MyChem.info Documentation

Release v1

Su and Wu Labs

Apr 14, 2022

Contents

CHAPTER 1

Introduction

MyChem.info provides simple-to-use REST web services to query/retrieve chemical and drug annotation data. It's designed with an emphasis on **simplicity** and **performance**.

CHAPTER 2

Quick start

MyChem.info provides two simple web services: one for querying chemical compound or drug objects and the other for chemical/drug annotation retrieval by common IDs (e.g. inchikey, chebiID, drugbankID etc.). Both return results in JSON format.

2.1 Chemical/drug query service

2.1.1 URL

```
http://mychem.info/v1/query
```

2.1.2 Examples

```
http://mychem.info/v1/query?q=imatinib
http://mychem.info/v1/query?q=_exists_:drugbank
http://mychem.info/v1/query?q=drugbank.targets.uniprot:A9UF02&fields=drugbank
```

Hint: View nicely formatted JSON result in your browser with this handy add-on: JSON formatter for Chrome or JSONView for Firefox.

2.1.3 To learn more

- You can read the full description of our query syntax here.
- Try it live on interactive API page.
- Batch queries? Yes, you can. do it with a POST request.

2.2 Chemical/drug annotation service

2.2.1 URL

```
http://mychem.info/v1/chem/<chem_id>
```

`<chem_id>` can be any one of the following common chemical/drug identifiers:

- InChIKey,
- DrugBank accession number,
- ChEMBLID,
- ChEBI identifier,
- PubChem CID,
- UNII.

2.2.2 Examples

```
http://mychem.info/v1/chem/KTUFNOKKBVMGRW-UHFFFAOYSA-N
http://mychem.info/v1/chem/CHEBI:45783?fields=chebi
http://mychem.info/v1/chem/CHEMBL941?fields=chembl
http://mychem.info/v1/chem/BKJ8M8G5HI?fields=unii
http://mychem.info/v1/chem/DB00619?fields=drugbank
```

2.2.3 To learn more

- You can read the full description of our query syntax here.
- Try it live on interactive API page.
- Yes, batch queries via POST request as well.

CHAPTER 3

Documentation

3.1 Chemical annotation data

3.1.1 Data sources

We currently obtain chemical annotation data from several data resources and keep them up-to-date, so that you don't have to do it:

Total Chemicals loaded: **N/A**

Source	version	# of chemicals	key name*	data notes
AEOLUS	-	0	aeolus	notes
ChEBI	-	0	chebi	notes
ChEMBL	-	0	chembl	
DrugBank	-	0	drugbank	notes
DrugCentral	-	0	drugcentral	
ginas	-	0	ginas	
NDC	-	0	ndc	notes
PharmGKB	-	0	pharmgkb	
PubChem	-	0	pubchem	
SIDER	-	0	sider	notes
UNII	-	0	unii	

* key name: this is the key for the specific annotation data in a chemical object.

The most updated information can be accessed here.

Note: Each data source may have its own usage restrictions. Please refer to the data source pages above for their specific restrictions.

3.1.2 Chemical object

Chemical annotation data are both stored and returned as a chemical object, which is essentially a collection of fields (attributes) and their values:

```
{
  "_id": "KTUFNOKKBVMGRW-UHFFFAOYSA-N",
  "unii": {
    "_license": "http://bit.ly/2Pg8Oo9",
    "inchikey": "KTUFNOKKBVMGRW-UHFFFAOYSA-N",
    "ingredient_type": "INGREDIENT SUBSTANCE",
    "inn_id": "8031",
    "molecular_formula": "C29H31N7O",
    "ncit": "C62035",
    "preferred_term": "IMATINIB",
    "pubchem": "5291",
    "registry_number": "152459-95-5",
    "rxcui": "282388",
    "smiles":
↪"CN1CCN(CC2=CC=C(C=C2)C(=O)NC3=CC(NC4=NC=CC(=N4)C5=CC=CN=C5)=C(C)C=C3)CC1",
    "unii": "BKJ8M8G5HI"
  }
}
```

The example above omits many of the available fields. For a full example, check out this example chemical, or try the interactive API page.

3.1.3 _id field

Each individual chemical object contains an "**_id**" field as the primary key. Where possible, MyChem.info chemical objects use InChIKey (a 27 character hash of the International Chemical Identifier) as their "**_id**". If an InChIKey isn't available, any one of the following datasource IDs may be used:

- DrugBank accession number,
- ChEMBLID,
- ChEBI identifier,
- PubChem CID,
- UNII.

3.1.4 _score field

You will often see a "_score" field in the returned chemical object, which is the internal score representing how well the query matches the returned chemical object. It probably does not mean much in chemical annotation service when only one chemical object is returned. In chemical query service, by default, the returned chemical hits are sorted by the scores in descending order.

3.1.5 Available fields

The table below lists all of the possible fields that could be in a chemical object, as well as all of their parents (for nested fields). If the field is indexed, it may also be directly queried.

3.2 Data release notes

This page contains metadata about each MyChem.info data release. Click a link to see more.

3.2.1 MyChem Releases

3.3 Chemical query service

This page describes the reference for MyChem.info chemical query web service. It's also recommended to try it live on our interactive API page.

3.3.1 Service endpoint

```
http://mychem.info/v1/query
```

3.3.2 GET request

Query parameters

q

Required, passing user query. The detailed query syntax for parameter "**q**" we explained *below*.

fields

Optional, a comma-separated string to limit the fields returned from the matching chemical/drug hits. The supported field names can be found from any chemical object (e.g. here). Note that it supports dot notation, and wildcards as well, e.g., you can pass "drugbank", "drugbank.name", or "dbnsfp.products.*". If "fields=all", all available fields will be returned. Default: "all".

size

Optional, the maximum number of matching chemical hits to return (with a cap of 1000 at the moment). Default: 10.

from

Optional, the number of matching chemical hits to skip, starting from 0. Default: 0

Hint: The combination of "**size**" and "**from**" parameters can be used to get paging for large query:

```
q=drugbank.name:acid*&size=50                    first 50 hits
q=drugbank.name:acid*&size=50&from=50            the next 50 hits
```

fetch_all

Optional, a boolean, which when TRUE, allows fast retrieval of all unsorted query hits. The return object contains a **_scroll_id** field, which when passed as a parameter to the query endpoint, returns the next 1000 query results. Setting **fetch_all** = TRUE causes the results to be inherently unsorted, therefore the **sort** parameter is ignored. For more information see *examples using fetch_all here*. Default: FALSE.

scroll_id

Optional, a string containing the **_scroll_id** returned from a query request with **fetch_all** = TRUE. Supplying a valid **scroll_id** will return the next 1000 unordered results. If the next results are not obtained within 1 minute of the previous set of results, the **scroll_id** becomes stale, and a new one must be obtained with another query request with **fetch_all** = TRUE. All other parameters are ignored when the **scroll_id** parameter is supplied. For more information see *examples using scroll_id here*.

sort

Optional, the comma-separated fields to sort on. Prefix with "-" for descending order, otherwise in ascending order. Default: sort by matching scores in descending order.

facets

Optional, a single field or comma-separated fields to return facets, can only be used on non-free text fields. E.g. "facets=chembl.molecule_properties.full_mwt". See *examples of faceted queries here*.

facet_size

Optional, an integer (1 <= **facet_size** <= 1000) that specifies how many buckets to return in a faceted query.

callback

Optional, you can pass a "**callback**" parameter to make a JSONP call.

dotfield

Optional, can be used to control the format of the returned chemical object. If "dotfield" is true, the returned data object is returned flattened (no nested objects) using dotfield notation for key names. Default: false.

email

Optional, if you are regular users of our services, we encourage you to provide us an email, so that we can better track the usage or follow up with you.

Query syntax

Examples of query parameter "**q**":

Simple queries

search for everything:

```
q=imatinib                        # search all default fields for term
```

Fielded queries

```
q=chebi.xref.uniprot:P80175                # for matching value on a specific field

q=drugbank.name:(acid alcohol)             # multiple values for a field
q=drugbank.name:(acid OR alcohol)          # multiple values for a field using OR

q=_exists_:pubchem                         # having pubchem field
q=NOT _exists_:drugbank                    # missing drugbank field
```

Hint: For a list of available fields, see *here*.

Range queries

```
q=pubchem.exact_mass:<200
q=pubchem.exact_mass:>=500

q=pubchem.exact_mass:[200 TO 500]          # bounded (including 200 and 500)
q=pubchem.exact_mass:{200 TO 500}         # unbounded
```

Wildcard queries

Wildcard character "*" or "?" is supported in either simple queries or fielded queries:

```
q=drugbank.name:acid*
```

Note: Wildcard character can not be the first character. It will be ignored.

Scrolling queries

If you want to return ALL results of a very large query, sometimes the paging method described *above* can take too long. In these cases, you can use a scrolling query. This is a two-step process that turns off database sorting to allow very fast retrieval of all query results. To begin a scrolling query, you first call the query endpoint as you normally would, but with an extra parameter **fetch_all** = TRUE. For example, a GET request to:

```
http://mychem.info/v1/query?q=_exists_:drugbank&fields=drugbank.name&fetch_all=TRUE
```

Returns the following object:

```
{
  "_scroll_id":
↪"cXVlcnlUaGVuRmV0Y2g7MTA7Njg4ODAwOTI6SmU0ck9oMTZUUHFyRXlYSTNPS2pMZzs2ODg4MDA5MTpKZTRyT2gxNlRQcXJFe
↪",
  "max_score": 1.0,
  "took": 2042,
  "total": 11290,
  "hits": [
    {
      "_id": "SDUQYLNIPVEERB-QPPQHZFASA-N",
      "_score": 1.0,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Gemcitabine"
      }
    },
    {
      "_id": "SESFRYSPDFLNCH-UHFFFAOYSA-N",
      "_score": 1.0,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Benzyl Benzoate"
      }
    },
    .
    .
    .
  ],
}
```

At this point, the first 1000 hits have been returned (of ~11,000 total), and a scroll has been set up for your query. To get the next batch of 1000 unordered results, simply execute a GET request to the following address, supplying the _scroll_id from the first step into the **scroll_id** parameter in the second step:

```
http://mychem.info/v1/query?scroll_
↪id=cXVlcnlUaGVuRmV0Y2g7MTA7Njg4ODAwOTI6SmU0ck9oMTZUUHFyRXlYSTNPS2pMZzs2ODg4MDA5MTpKZTRyT2gxNlRQcXJ
```

Hint: Your scroll will remain active for 1 minute from the last time you requested results from it. If your scroll expires before you get the last batch of results, you must re-request the scroll_id by setting **fetch_all** = TRUE as in step 1.

Hint: When you need to use this "scrolling query" feature via "fetch_all" parameter, we recommend you to use our Python client "biothings_client".

Boolean operators and grouping

You can use **AND/OR/NOT** boolean operators and grouping to form complicated queries:

```
q=_exists_:drugbank AND _exists_:pubchem                          AND operator
q=_exists_:drugbank AND NOT _exists_:pubchem                      NOT operator
q=_exists_:drugbank OR (_exists_:chebi AND _exists_:pubchem)      grouping with␣
↪()
```

Escaping reserved characters

If you need to use these reserved characters in your query, make sure to escape them using a back slash (""):

```
+ - = && || > < ! ( ) { } [ ] ^ " ~ * ? : \ /
```

Returned object

A GET request like this:

```
http://mychem.info/v1/query?q=drugbank.name:acid&fields=drugbank.name
```

should return hits as:

```
{
  "max_score": 7.929331,
  "took": 102,
  "total": 1063,
  "hits": [
    {
      "_id": "BDAGIHXWWSANSR-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Formic Acid"
      }
    },
    {
      "_id": "BSYNRYMUTXBXSQ-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Acetylsalicylic acid"
      }
    },
    {
      "_id": "KGBXLFKZBHKPEV-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Boric acid"
      }
    },
    {
      "_id": "LPEPZBJOKDYZAD-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Flufenamic Acid"
```

(continues on next page)

(continued from previous page)

```
      }
    },
    {
      "_id": "JXMIBUGMYLQZGO-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Iotroxic acid"
      }
    },
    {
      "_id": "HXQVQGWHFRNKMS-UHFFFAOYSA-M",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Ethylmercurithiosalicylic acid"
      }
    },
    {
      "_id": "LOAUVZALPPNFOQ-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Quinaldic Acid"
      }
    },
    {
      "_id": "LDKRAXXVBWHMRH-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Phosphonoacetohydroxamic Acid"
      }
    },
    {
      "_id": "GWYFCOCPABKNJV-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Isovaleric Acid"
      }
    },
    {
      "_id": "HJZKOAYDRQLPME-UHFFFAOYSA-N",
      "_score": 7.929331,
      "drugbank": {
        "_license": "http://bit.ly/2PSfZTD",
        "name": "Oxidronic acid"
      }
    }
  ]
}
```

“**total**” in the output gives the total number of matching hits, while the actual hits are returned under “**hits**” field. “**size**” parameter controls how many hits will be returned in one request (default is 10). Adjust “**size**” parameter and “**from**” parameter to retrieve the additional hits.

Faceted queries

If you need to perform a faceted query, you can pass an optional "*facets*" parameter.

A GET request like this:

```
http://mychem.info/v1/query?q=drugbank.name:acid&fields=drugbank.name&facets=drugbank.
↪targets.organism&size=0
```

should return hits as:

```
{
  "facets": {
    "drugbank.targets.organism": {
      "other": 1782,
      "_type": "terms",
      "missing": 8,
      "total": 1483,
      "terms": [
        {
          "count": 545,
          "term": "human"
        },
        {
          "count": 250,
          "term": "strain"
        },
        {
          "count": 155,
          "term": "escherichia"
        },
        {
          "count": 154,
          "term": "coli"
        },
        {
          "count": 138,
          "term": "k12"
        },
        {
          "count": 79,
          "term": "atcc"
        },
        {
          "count": 54,
          "term": "pseudomonas"
        },
        {
          "count": 43,
          "term": "dsm"
        },
        {
          "count": 34,
          "term": "bacillus"
        },
        {
          "count": 31,
          "term": "sp"
```

(continues on next page)

(continued from previous page)

```
        }
      ]
    }
  },
  "max_score": 0.0,
  "took": 12,
  "total": 1063,
  "hits": []
}
```

3.3.3 Batch queries via POST

Although making simple GET requests above to our chemical query service is sufficient for most use cases, there are times you might find it more efficient to make batch queries (e.g., retrieving chemical annotation for multiple chemicals). Fortunately, you can also make batch queries via POST requests when you need:

```
URL: http://mychem.info/v1/query
HTTP method:  POST
```

Query parameters

q

Required, multiple query terms seperated by comma (also support "+" or white space), but no wildcard, e.g., 'q=SDUQYLNIPVEERB-QPPQHZFASA-N,SESFRYSPDFLNCH-UHFFFAOYSA-N'

scopes

Optional, specify one or more fields (separated by comma) as the search "scopes", e.g., "scopes=drugbank". The available "fields" can be passed to "**scopes**" parameter are *listed here*. Default:

fields

Optional, a comma-separated string to limit the fields returned from the matching chem hits. The supported field names can be found from any chemical object. Note that it supports dot notation, and wildcards as well, e.g., you can pass "drugbank", "drugbank.name", or "dbnsfp.products.*". If "fields=all", all available fields will be returned. Default: "all".

email

Optional, if you are regular users of our services, we encourage you to provide us an email, so that we can better track the usage or follow up with you.

Example code

Unlike GET requests, you can easily test them from browser, make a POST request is often done via a piece of code. Here is a sample python snippet using httplib2 module:

```
import httplib2
h = httplib2.Http()
headers = {'content-type': 'application/x-www-form-urlencoded'}
params = 'q=CHEBI:175901,CHEBI:41237&scopes=chebi.id&fields=drugbank.name'
res, con = h.request('http://mychem.info/v1/query', 'POST', params, headers=headers)
```

or this example using requests module:

```
import requests
params = {'q': 'CHEBI:175901,CHEBI:41237', 'scopes': 'chebi.id', 'fields': 'drugbank.
↪name'}
res = request.post('http://mychem.info/v1/query', params)
con = res.json()
```

Returned object

Returned result (the value of “con” variable above) from above example code should look like this:

```
[
  {
    "query": "CHEBI:175901",
    "_score": 16.388842,
    "drugbank": {
      "_license": "http://bit.ly/2PSfZTD",
      "name": "Gemcitabine"
    },
    "_id": "SDUQYLNIPVEERB-QPPQHZFASA-N"
  },
  {
    "query": "CHEBI:41237",
    "_score": 16.388842,
    "drugbank": {
      "_license": "http://bit.ly/2PSfZTD",
      "name": "Benzyl Benzoate"
    },
    "_id": "SESFRYSPDFLNCH-UHFFFAOYSA-N"
  }
]
```

Tip: “query” field in returned object indicates the matching query term.

If a query term has no match, it will return with “**notfound**” field as “**true**”:

```
[
  ...,
  {'query': '...',
   'notfound': true},
  ...
]
```

3.4 Chemical annotation service

This page describes the reference for the MyChem.info chemical annotation web service. It's also recommended to try it live on our interactive API page.

3.4.1 Service endpoint

```
http://mychem.info/v1/chem
```

3.4.2 GET request

Obtaining the chemical annotation via our web service is as simple as calling this URL:

```
http://mychem.info/v1/chem/<chemid>
```

chemid above is any one of several common chemical identifiers: InChIKey, DrugBank accession number, ChEMBLID, ChEBI identifier, PubChem CID, UNII.

By default, this will return the complete chemical annotation object in JSON format. See *here* for an example and *here* for more details. If the input **chemid** is not valid, 404 (NOT FOUND) will be returned.

Optionally, you can pass a "**fields**" parameter to return only the annotation you want (by filtering returned object fields):

```
http://mychem.info/v1/chem/KTUFNOKKBVMGRW-UHFFFAOYSA-N?fields=drugbank
```

"**fields**" accepts any attributes (a.k.a fields) available from the chemical object. Multiple attributes should be separated by commas. If an attribute is not available for a specific chemical object, it will be ignored. Note that the attribute names are case-sensitive.

Just like the chemical query service, you can also pass a "**callback**" parameter to make a JSONP call.

Query parameters

fields

Optional, can be a comma-separated fields to limit the fields returned from the chemical object. If "fields=all", all available fields will be returned. Note that it supports dot notation as well, e.g., you can pass "drugbank.name". Default: "fields=all".

callback

Optional, you can pass a "**callback**" parameter to make a JSONP call.

filter

Alias for "fields" parameter.

email

Optional, if you are regular users of our services, we encourage you to provide us an email, so that we can better track the usage or follow up with you.

Returned object

A GET request like this:

```
http://mychem.info/v1/chem/KTUFNOKKBVMGRW-UHFFFAOYSA-N?fields=pubchem
```

should return a chemical object below:

```
{
  "_id": "KTUFNOKKBVMGRW-UHFFFAOYSA-N",
  "_version": 1,
  "pubchem": {
    "_license": "http://bit.ly/2AqoLOc",
    "chiral_atom_count": 0,
    "chiral_bond_count": 0,
    "cid": 5291,
    "complexity": 706,
    "covalently-bonded_unit_count": 1,
    "defined_atom_stereocenter_count": 0,
    "defined_bond_stereocenter_count": 0,
    "exact_mass": 493.259,
    "formal_charge": 0,
    "heavy_atom_count": 37,
    "hydrogen_bond_acceptor_count": 7,
    "hydrogen_bond_donor_count": 2,
    "inchi": "InChI=1S/C29H31N7O/c1-21-5-10-25(18-27(21)34-29-31-13-11-26(33-29)24-4-
↪3-12-30-19-24)32-28(37)23-8-6-22(7-9-23)20-36-16-14-35(2)15-17-36/h3-13,18-19H,14-
↪17,20H2,1-2H3,(H,32,37)(H,31,33,34)",
    "inchi_key": "KTUFNOKKBVMGRW-UHFFFAOYSA-N",
    "isotope_atom_count": 0,
    "iupac": {
      "traditional": "4-[(4-methylpiperazino)methyl]-N-[4-methyl-3-[[4-(3-
↪pyridyl)pyrimidin-2-yl]amino]phenyl]benzamide"
    },
    "molecular_formula": "C29H31N7O",
    "molecular_weight": 493.615,
    "monoisotopic_weight": 493.259,
    "rotatable_bond_count": 7,
    "smiles": {
      "isomeric":
↪"CC1=C(C=C(C=C1)NC(=O)C2=CC=C(C=C2)CN3CCN(CC3)C)NC4=NC=CC(=N4)C5=CN=CC=C5"
    },
    "tautomers_count": 72,
    "topological_polar_surface_area": 86.3,
    "undefined_atom_stereocenter_count": 0,
    "undefined_bond_stereocenter_count": 0,
    "xlogp": 3.5
  }
}
```

3.4.3 Batch queries via POST

Although making simple GET requests above to our chemical query service is sufficient in most use cases, there are some times you might find it's easier to batch query (e.g., retrieving chemical annotations for multiple chemicals). Fortunately, you can also make batch queries via POST requests when you need:

```
URL: http://mychem.info/v1/chem
HTTP method:  POST
```

Query parameters

ids

Required. Accept multiple chemical ids separated by comma, e.g., "ids=SDUQYLNIPVEERB-QPPQHZFASA-N,SESFRYSPDFLNCH-UHFFFAOYSA-N,SHGAZHPCJJPHSC-ZVCIMWCZSA-N". Note that currently we only take the input ids up to **1000** maximum, the rest will be omitted.

fields

Optional, can be a comma-separated fields to limit the fields returned from the matching hits. If "fields=all", all available fields will be returned. Note that it supports dot notation as well, e.g., you can pass "drugbank" or "drugbank.name". Default: "all".

email

Optional, if you are regular users of our services, we encourage you to provide us an email, so that we can better track the usage or follow up with you.

Example code

Unlike GET requests, you can easily test them from browser, make a POST request is often done via a piece of code, still trivial of course. Here is a sample python snippe using httplib2 modulet:

```
import httplib2
h = httplib2.Http()
headers = {'content-type': 'application/x-www-form-urlencoded'}
params = 'ids=SDUQYLNIPVEERB-QPPQHZFASA-N,SESFRYSPDFLNCH-UHFFFAOYSA-N&fields=drugbank.
↪name'
res, con = h.request('http://mychem.info/v1/chem', 'POST', params, headers=headers)
```

or this example using requests module:

```
import requests
params = {'ids': 'SDUQYLNIPVEERB-QPPQHZFASA-N,SESFRYSPDFLNCH-UHFFFAOYSA-N', 'fields':
↪'drugbank.name'}
res = request.post('http://mychem.info/v1/chem', params)
con = res.json()
```

Returned object

Returned result (the value of “con” variable above) from above example code should look like this:

```
[
  {
    "_id": "SDUQYLNIPVEERB-QPPQHZFASA-N",
    "query": "SDUQYLNIPVEERB-QPPQHZFASA-N",
    "drugbank": {
      "_license": "http://bit.ly/2PSfZTD",
      "name": "Gemcitabine"
    }
  },
  {
    "_id": "SESFRYSPDFLNCH-UHFFFAOYSA-N",
    "query": "SESFRYSPDFLNCH-UHFFFAOYSA-N",
    "drugbank": {
      "_license": "http://bit.ly/2PSfZTD",
      "name": "Benzyl Benzoate"
    }
  }
]
```

3.5 Server response

The MyChem.info server returns a variety of query responses, and response status codes. They are listed here.

Note: These examples show query responses using the python requests package.

3.5.1 Status code *200*

A **200** status code indicates a successful query, and is accompanied by the query response payload.

```
In [1]: import requests

In [2]: r = requests.get('http://mychem.info/v1/query?q=_exists_:drugbank')

In [3]: r.status_code
Out[3]: 200

In [4]: data = r.json()

In [5]: data.keys()
Out[5]: dict_keys(['total', 'max_score', 'took', 'hits'])
```

3.5.2 Status code *400*

A **400** status code indicates an improperly formed query, and is accompanied by a response payload describing the source of the error.

```
In [6]: r = requests.get('http://mychem.info/v1/query?q=_exists_:drugbank&size=u')

In [7]: r.status_code
Out[7]: 400

In [8]: data = r.json()

In [9]: data
Out[9]:
{'error': "Expected 'size' parameter to have integer type.  Couldn't convert 'u' to␣
↪integer",
 'success': False}
```

3.5.3 Status code *404*

A **404** status code indicates either an unrecognized URL, as in (*/query* is misspelled */quer* resulting in an unrecognized URL):

```
In [10]: r = requests.get('http://mychem.info/v1/quer?q=_exists_:drugbank')

In [11]: r.status_code
Out[11]: 404
```

or, for the **/chem** endpoint, a **404** status code could be from querying for a nonexistent chemical ID, as in:

```
In [12]: r = requests.get('http://mychem.info/v1/chem/5')

In [13]: r.status_code
Out[13]: 404

In [14]: data = r.json()

In [15]: data
Out[15]:
{'error': "ID '5' not found",
 'success': False}
```

3.5.4 Status code *5xx*

Any **5xx** status codes are the result of uncaught query errors. Ideally, these should never occur. We routinely check our logs for these types of errors and add code to catch them, but if you see any status **5xx** responses, please submit a bug report to help@mychem.info.

3.6 Biothings_client python module

You can access the MyChem.info services programmatically with our biothings_client unified python client.

CHAPTER 4

Related links

- github repository