

---

# Dremio client Documentation

*Release \_\_version\_\_ = '0.13.2'*

**Ryan Murray**

**Jun 04, 2020**



---

## Contents:

---

<b>1</b>	<b>Dremio client</b>	<b>1</b>
1.1	Features . . . . .	1
1.2	Status . . . . .	2
1.3	TODO . . . . .	2
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Stable release . . . . .	3
2.2	Conda release . . . . .	3
2.3	From sources . . . . .	3
<b>3</b>	<b>Usage</b>	<b>5</b>
3.1	Configuration . . . . .	5
<b>4</b>	<b>Utilities</b>	<b>7</b>
4.1	Async and sync queries . . . . .	7
<b>5</b>	<b>Command Line Interface</b>	<b>9</b>
5.1	Run a Query . . . . .	10
<b>6</b>	<b>dremio_client</b>	<b>11</b>
6.1	dremio_client package . . . . .	11
<b>7</b>	<b>Contributing</b>	<b>39</b>
7.1	Types of Contributions . . . . .	39
7.2	Get Started! . . . . .	40
7.3	Pull Request Guidelines . . . . .	41
7.4	Tips . . . . .	41
7.5	Deploying . . . . .	41
<b>8</b>	<b>Credits</b>	<b>43</b>
8.1	Development Lead . . . . .	43
8.2	Contributors . . . . .	43
<b>9</b>	<b>History</b>	<b>45</b>
9.1	0.1.0 (2019-07-04) . . . . .	45
9.2	0.2.4 (2019-08-13) . . . . .	45
9.3	0.3.0 (2019-08-14) . . . . .	45
9.4	0.5.0 (2019-10-30) . . . . .	45

9.5	0.6.3 (2020-01-29)	45
9.6	0.8.4 (2020-01-29)	46
9.7	0.8.8 (2020-02-19)	46
9.8	0.8.9 (2020-02-19)	46
9.9	0.8.11 (2020-03-09)	46
9.10	0.9.2 (2020-03-17)	46
9.11	0.10.1 (2020-03-19)	46
9.12	0.11.1 (2020-04-09)	46
9.13	0.12.0 (2020-04-22)	46
9.14	0.12.1 (2020-05-04)	47
9.15	0.13.0 (2020-05-21)	47
9.16	0.13.1 (2020-05-21)	47
9.17	0.13.2 (2020-05-22)	47
<b>10</b>	<b>Indices and tables</b>	<b>49</b>
	<b>Python Module Index</b>	<b>51</b>
	<b>Index</b>	<b>53</b>

# CHAPTER 1

---

## Dremio client

---

The **un-official** python client for Dremio's REST API. This enables both administrators and data scientists to get the most out of [Dremio](#) in Python

- Documentation: <https://dremio-client.readthedocs.io>.
- Github: [https://github.com/rymurr/dremio\\_client](https://github.com/rymurr/dremio_client)
- PyPI: [https://pypi.python.org/pypi/dremio\\_client](https://pypi.python.org/pypi/dremio_client)
- Free software: Apache Software License 2.0

## 1.1 Features

- Cross platform support
- All Pythons between 2.7 - 3.7 supported
- Full support for Dremio's [REST API](#)
- Optional Support for Dremio's [ODBC](#) or experimental [Arrow Flight](#) capabilities
- Rich config file support via [confuse](#) yaml config library. Simple to create a client (config stored in a yaml file)

```
from dremio_client import init
client = init() # initialise connectivity to Dremio via config file
catalog = client.data # fetch catalog
vds = catalog.space.vds.get() # fetch a specific dataset
```

(continues on next page)

(continued from previous page)

```
df = vds.query() # query the first 1000 rows of the dataset and return as ↵  
↵ a DataFrame  
pds = catalog.source.pds.get() # fetch a physical dataset  
pds.metadata_refresh() # refresh metadata on that dataset
```

- CLI interface for integration with scripts

```
$ dremio_client query --sql 'select * from sys.options'  
{'results':results}  
$ dremio_client refresh-metadata --table 'my.vds.name'  
{'result':'ok'}
```

- Catalog autocompletion in Jupyter Notebooks

## 1.2 Status

This is still alpha software and is relatively incomplete. Contributions in the form of Github Issues or Pull requests are welcome. See [CONTRIBUTING](#)

## 1.3 TODO

- see issues

### 2.1 Stable release

To install basic Dremio client, run this command in your terminal:

```
$ pip install dremio_client
```

This installs a minimal set of dependencies suitable for only interacting with the REST API and for restricted environments. To get full set of dependencies run:

```
$ pip install dremio_client[noarrow] # everything but pyarrow  
$ pip install dremio_client[full] # all dependencies
```

This is the preferred method to install Dremio client, as it will always install the most recent stable release.

If you don't have `pip` installed, this [Python installation guide](#) can guide you through the process.

### 2.2 Conda release

You can use conda to install Dremio Client by doing:

```
$ conda install dremio_client -c rymurr -c conda-forge
```

### 2.3 From sources

The sources for Dremio client can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/rymurr/dremio_client
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/rymurr/dremio_client/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```



To use Dremio client in a project:

```
import dremio_client
```

## 3.1 Configuration

The Dremio Client is configured using the `confuse` yaml based configuration library. This looks for a configuration file called `config.yaml` in:

- macOS: `~/.config/dremio_client` and `~/Library/Application Support/dremio_client`
- Other Unix: `~/.config/dremio_client` and `/etc/dremio_client`
- Windows: `%APPDATA%\dremio_client` where the `APPDATA` environment variable falls back to `%HOME%\AppData\Roaming` if undefined
- Via the environment variable `DREMIO_CLIENTDIR`

The default config file is as follows:

```
auth:
  type: basic # currently only basic is supported
  username: dremio
  password: dremio123
  timeout: 10
hostname: localhost
port: 9047
ssl: false
```

The `command line interface` can be configured with most of the above parameters via flags or by setting a config directory. The relevant configs can also be set via environment variables. These take precedence. The environment variable format is to append `DREMIO_` to a config parameter and nested configs are separated by a `_`. For example: `DREMIO_AUTH_TIMEOUT` maps to `auth.timeout` in the default configuration file above.



We provide several useful utilities to help with common tasks

## 4.1 Async and sync queries

To execute a query synchronously run the following:

```
from dremio_client import init
import pandas as pd
query = 'select * from sys.options'
client = init(simple_client=True)
results = client.query(self, query)
pd.DataFrame(results)
```

This will submit the sql query to the sql endpoint then poll the job-status endpoint until the job completed. If the job has succeeded a list of dicts is returned else it throws an exception.

One can execute many queries at once via async queries also:

```
from dremio_client import init
from concurrent.futures.thread import ThreadPoolExecutor
from concurrent.futures import as_completed

queries = ['select * from table0',
          'select * from table1',
          'select * from table2',
          'select * from table3',
          'select * from table4',
          'select * from table5'
          ]
client = init(simple_client=True)

def execute(query):
```

(continues on next page)

(continued from previous page)

```
client.query(query, asynchronous=True)

# example shamelessly taken from https://docs.python.org/3/library/
# concurrent.futures.html
with ThreadPoolExecutor(max_workers=4) as executor:
    futures = {executor.submit(execute, query): query for query in queries}
    for future in as_completed(futures):
        query = futures[future]
        try:
            result = future.result()
        except Exception as exc:
            print('%r generated an exception: %s' % (url, exc))
        else:
            print('%r returned %d rows' % (url, len(results)))
```

---

## Command Line Interface

---

The Dremio Client uses the [Click](#) library to generate its command line interface. This is installed as `dremio_client` by `pip`

All of the REST API calls are exposed via the command line interface. To see a list of what is available run:

```
$ dremio_client --help
Usage: dremio_client [OPTIONS] COMMAND [ARGS]...

Options:
  --config DIRECTORY    Custom config file.
  -h, --hostname TEXT    Hostname if different from config file
  -p, --port INTEGER     Hostname if different from config file
  --ssl                 Use SSL if different from config file
  -u, --username TEXT    username if different from config file
  -p, --password TEXT    password if different from config file
  --help                Show this message and exit.

Commands:
  catalog_item      return the details of a given catalog item if...
  delete_catalog    delete a catalog entity given by cid and tag...
  delete_pat        delete a personal access token
  graph             return graph info of PDS or VDS
  group             return group info
  job_results       return results for a given job id paginated...
  job_status        Return status of job for a given job id
  pat              return personal access token info for a given...
  query            execute a query given by sql and print...
  reflection        return the reflection set
  reflections       return the reflection set
  refresh_pds       refresh metadata/reflections for a given...
  set_catalog       set a catalog entity with the given json...
  set_pat          create a personal access token
  sql              Execute sql statement and return job id
  tags             returns tags for a given catalog entity id or...
  update_catalog    update a catalog entity (cid) given a json...
```

(continues on next page)

(continued from previous page)

user	<b>return</b> user info
votes	<b>return</b> reflection votes
wiki	returns wiki <b>for</b> a given catalog entity id or...
wlm_queues	<b>return</b> the list of wlm queues
wlm_rules	<b>return</b> the list of wlm rules

The config directory and a number of other settings can be configured from the command line and overwrite the default config locations.

## 5.1 Run a Query

We can run a query on the command line and have the results returned as json. We use `jq` to manipulate the json after it is returned. This could be combined into more complete scripts for maintenance and monitoring.

```
$ dremio_client query --help
Usage: dremio_client query [OPTIONS]

    execute a query given by sql and print results as json

Options:
  --sql TEXT  sql query to execute. [required]
  --help      Show this message and exit.
$ dremio_client query --sql 'select * from sys.options' | jq '. | length'
477
$ dremio_client query --sql 'select * from sys.options' | jq
[
  {
    "name": "vote.release.leadership.ms",
    "kind": "LONG",
    "type": "SYSTEM",
    "status": "DEFAULT",
    "num_val": 126000000,
    "string_val": null,
    "bool_val": null,
    "float_val": null
  },
  {
    "name": "vote.schedule.millis",
    "kind": "LONG",
    "type": "SYSTEM",
    ...
$ dremio_client query --sql 'select * from sys.options' | jq 'to_entries[] |
  ↪  "\(.value | .name) "'
"acceleration.orphan.cleanup_in_milliseconds"
"accelerator.enable.subhour.policies"
"accelerator.enable_agg_join"
...
```

## 6.1 dremio\_client package

### 6.1.1 Subpackages

#### dremio\_client.auth package

##### Submodules

#### dremio\_client.auth.basic module

`dremio_client.auth.basic.login` (*base\_url, username, password, timeout=10, verify=True*)

Log into dremio using basic auth :param base\_url: Dremio url :param username: username :param password: password :param timeout: optional timeout :param verify: If false ignore ssl errors :return: auth token

#### Module contents

`dremio_client.auth.basic_auth` (*base\_url, username, password, timeout=10, verify=True*)

Log into dremio using basic auth :param base\_url: Dremio url :param username: username :param password: password :param timeout: optional timeout :param verify: If false ignore ssl errors :return: auth token

`dremio_client.auth.config_auth` (*base\_url, config\_dict*)

Log into dremio using basic auth and looking for config :param base\_url: Dremio url :param config\_dict: config dict :param timeout: optional timeout :return: auth token

`dremio_client.auth.auth` (*base\_url, config\_dict*)

#### dremio\_client.conf package

## Submodules

### dremio\_client.conf.cli\_helper module

`dremio_client.conf.cli_helper.get_base_url_token(args=None)`

### dremio\_client.conf.config\_parser module

`dremio_client.conf.config_parser.build_config(args=None)`

## Module contents

`dremio_client.conf.build_config(args=None)`

`dremio_client.conf.get_base_url_token(args=None)`

`dremio_client.conf.to_dict(config)`

## dremio\_client.flight package

### Submodules

#### dremio\_client.flight.command\_pb2 module

#### dremio\_client.flight.flight\_auth module

`class dremio_client.flight.flight_auth.HttpDremioClientAuthHandler(username, password)`

Bases: `pyarrow._flight.ClientAuthHandler`

**authenticate** (*self*, *outgoing*, *incoming*)

Conduct the handshake with the server.

**outgoing** [ClientAuthSender] A channel to send messages to the server.

**incoming** [ClientAuthReader] A channel to read messages from the server.

**get\_token** (*self*)

Get the auth token for a call.

## Module contents

`dremio_client.flight.connect(hostname='localhost', port=47470, username='dremio', password='dremio123', tls_root_certs_filename=None)`

Connect to and authenticate against Dremio's arrow flight server. Auth is skipped if username is None

### Parameters

- **hostname** – Dremio coordinator hostname
- **port** – Dremio coordinator port
- **username** – Username on Dremio



- **password** – Password on Dremio
- **tls\_root\_certs\_filename** – use ssl to connect with root certs from filename

**Returns** arrow flight client

```
dremio_client.flight.query(sql, client=None, hostname='localhost', port=47470, user-
                           name='dremio', password='dremio123', pandas=True,
                           tls_root_certs_filename=False)
```

Run an sql query against Dremio and return a pandas dataframe or arrow table

Either host,port,user,pass tuple or a pre-connected client should be supplied. Not both

**Parameters**

- **sql** – sql query to execute on dremio
- **client** – pre-connected client (optional)
- **hostname** – Dremio coordinator hostname (optional)
- **port** – Dremio coordinator port (optional)
- **username** – Username on Dremio (optional)
- **password** – Password on Dremio (optional)
- **pandas** – return a pandas dataframe (default) or an arrow table
- **tls\_root\_certs\_filename** – use ssl to connect with root certs from filename

**Returns**

## dremio\_client.model package

### Submodules

#### dremio\_client.model.catalog module

```
dremio_client.model.catalog.catalog(token, base_url, flight_endpoint, ssl_verify=True)
```

#### dremio\_client.model.data module

```
class dremio_client.model.data.AccessControl(id=None, permissions=None)
```

Bases: object

```
to_json()
```

```
class dremio_client.model.data.AccessControlList(users=None, groups=None, ver-
                                                  sion=None)
```

Bases: object

```
to_json()
```

```
class dremio_client.model.data.Catalog(token=None, base_url=None,
                                       flight_endpoint=None, ssl_verify=True,
                                       dirty=False)
```

Bases: dict

```
commit()
```

```
delete()
```

**get** ()  
Return the value for key if key is in the dictionary, else default.

**insert** (entity\_type, name, commit=True, \*\*kwargs)

**keys** () → a set-like object providing a view on D's keys

**promote** (file\_format='parquet', \*\*kwargs)  
promote this file/folder to a PDS

---

**Note:** can only be run on a file or folder in a Source

---

**Parameters file\_format** – file format of the file/folder (only parquet, xls, excel, json, text are accepted)

```
:return None

refresh ()
remove ()
tags ()
to_json ()
wiki ()

class dremio_client.model.data.Dataset (token=None, base_url=None,
                                         flight_endpoint=None, ssl_verify=True, dirty=False,
                                         **kwargs)
    Bases: dremio_client.model.data.Catalog

    get_graph ()
    get_table ()
    query ()
    sql (sql)

class dremio_client.model.data.DatasetMetaData (entityType=None, id=None, path=None,
                                                  tag=None, type=None, fields=None,
                                                  createdAt=None, accelerationRe-
                                                  freshPolicy=None, sql=None, sql-
                                                  Context=None, format=None, ap-
                                                  proximateStatisticsAllowed=None,
                                                  accessControlList=None)
    Bases: object

    to_json ()

class dremio_client.model.data.File (token=None, base_url=None, flight_endpoint=None,
                                       ssl_verify=True, dirty=False, **kwargs)
    Bases: dremio_client.model.data.Catalog

class dremio_client.model.data.FileMetaData (entityType=None, id=None, path=None, ac-
                                              cessControlList=None)
    Bases: object

    to_json ()
```

```
class dremio_client.model.data.Folder (token=None, base_url=None, flight_endpoint=None,
                                         ssl_verify=True, dirty=False, **kwargs)
    Bases: dremio_client.model.data.Catalog

class dremio_client.model.data.FolderMetadata (entityType=None, id=None, path=None,
                                                tag=None, accessControlList=None)
    Bases: object

    to_json()

class dremio_client.model.data.Home (token=None, base_url=None, flight_endpoint=None,
                                      ssl_verify=True, dirty=False, **kwargs)
    Bases: dremio_client.model.data.Space

class dremio_client.model.data.MetadataPolicy (authTTLs=None, datasetRe-
                                                freshAfterMs=None, datasetEx-
                                                pireAfterMs=None, namesRe-
                                                freshMs=None, datasetUpdate-
                                                Mode=None)
    Bases: object

    to_json()

class dremio_client.model.data.PhysicalDataset (token=None, base_url=None,
                                                  flight_endpoint=None, ssl_verify=True,
                                                  dirty=False, **kwargs)
    Bases: dremio_client.model.data.Dataset

    metadata_refresh()
    refresh()

class dremio_client.model.data.QueueMetadata (id=None, tag=None, name=None,
                                                cpuTier=None, maxAllowedRun-
                                                ningJobs=None, maxStartTime-
                                                outMs=None)
    Bases: object

    to_json()

class dremio_client.model.data.ReflectionMetadata (entityType=None, id=None,
                                                    tag=None, name=None, en-
                                                    abled=None, createdAt=None,
                                                    updatedAt=None, type=None,
                                                    datasetId=None, current-
                                                    SizeBytes=None, totalSize-
                                                    Bytes=None, status=None, di-
                                                    mensionFields=None, measure-
                                                    Fields=None, displayFields=None,
                                                    distributionFields=None, partition-
                                                    Fields=None, sortFields=None,
                                                    partitionDistributionStrat-
                                                    egy=None)
    Bases: object

    to_json()
```

```

class dremio_client.model.data.ReflectionSummaryMetadata (entityType=None,
                                                         id=None,          create-
                                                         dAt=None,          up-
                                                         datedAt=None,
                                                         type=None, name=None,
                                                         datasetId=None, dataset-
                                                         Path=None,         dataset-
                                                         Type=None,         current-
                                                         SizeBytes=None,    to-
                                                         talSizeBytes=None,
                                                         enabled=None,      sta-
                                                         tus=None)

    Bases: object

    to_json()

class dremio_client.model.data.Root (token=None, base_url=None, flight_endpoint=None,
                                     ssl_verify=True, dirty=False)
    Bases: dremio_client.model.data.Catalog

    add(item)

    add_by_path(item, new_entity=True)

class dremio_client.model.data.RootMetaData (id=None)
    Bases: object

    to_json()

class dremio_client.model.data.RuleMetadata (name=None, conditions=None, accep-
                                             tId=None, acceptName=None, action=None,
                                             id=None)

    Bases: object

    to_json()

class dremio_client.model.data.Source (token=None, base_url=None, flight_endpoint=None,
                                       ssl_verify=True, dirty=False, **kwargs)
    Bases: dremio_client.model.data.Catalog

class dremio_client.model.data.SourceMetadata (entityType=None, id=None, name=None,
                                              description=None,          tag=None,
                                              type=None,         config=None,         create-
                                              dAt=None,          metadataPolicy=None,
                                              state=None,          accelerationGracePe-
                                              riodMs=None,          accelerationRefresh-
                                              PeriodMs=None,          accelerationNever-
                                              Expire=None,          accelerationNever-
                                              Refresh=None, path=None, accessCon-
                                              trolList=None)

    Bases: object

    to_json()

class dremio_client.model.data.SourceState (status=None, message=None)
    Bases: object

    to_json()

class dremio_client.model.data.Space (token=None, base_url=None, flight_endpoint=None,
                                      ssl_verify=True, dirty=False, **kwargs)
    Bases: dremio_client.model.data.Catalog

```

```
class dremio_client.model.data.SpaceMetadata (entityType=None, id=None, name=None,
                                              tag=None, path=None, accessControl-
                                              List=None)
```

```
Bases: object
```

```
to_json()
```

```
class dremio_client.model.data.TagsData (tags=None, version=None)
```

```
Bases: object
```

```
to_json()
```

```
class dremio_client.model.data.VirtualDataset (token=None, base_url=None,
                                              flight_endpoint=None, ssl_verify=True,
                                              dirty=False, **kwargs)
```

```
Bases: dremio_client.model.data.Dataset
```

```
class dremio_client.model.data.VoteMetadata (id=None, votes=None, datasetId=None,
                                              datasetPath=None, datasetType=None,
                                              datasetReflectionCount=None, entity-
                                              Type=None)
```

```
Bases: object
```

```
to_json()
```

```
class dremio_client.model.data.WikiData (text=None, version=None)
```

```
Bases: object
```

```
to_json()
```

```
dremio_client.model.data.create (item, token, base_url, flight_endpoint, trim_path=0,
                                ssl_verify=True, dirty=False)
```

```
dremio_client.model.data.create_folder (catalog, path)
```

```
dremio_client.model.data.create_space (catalog, name)
```

```
dremio_client.model.data.create_vds (catalog, path, sql, sqlContext)
```

```
dremio_client.model.data.get_path (item, trim_path)
```

```
dremio_client.model.data.make_reflection (data, summary=False)
```

```
dremio_client.model.data.make_tags (tags)
```

```
dremio_client.model.data.make_vote (vote)
```

```
dremio_client.model.data.make_wiki (wiki)
```

```
dremio_client.model.data.make_wlm_queue (queue)
```

```
dremio_client.model.data.make_wlm_rule (rule)
```

## dremio\_client.model.endpoints module

```
dremio_client.model.endpoints.cancel_job (token, base_url, jid, ssl_verify=True)
```

```
cancel running job with job id = jid
```

```
https://docs.dremio.com/rest-api/jobs/post-job.html
```

### Parameters

- **token** – auth token
- **base\_url** – sql query

- **jid** – id of the job to cancel
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

:exception DremioNotFoundException no job found :exception DremioBadRequestException job already finished

`dremio_client.model.endpoints.catalog(token, base_url, ssl_verify=True)`  
<https://docs.dremio.com/rest-api/catalog/get-catalog.html> populate the root dremio catalog

#### Parameters

- **token** – auth token from previous login attempt
- **base\_url** – base Dremio url
- **ssl\_verify** – ignore ssl errors if False

**Returns** json of root resource

`dremio_client.model.endpoints.catalog_item(token, base_url, cid=None, path=None, ssl_verify=True)`

fetch a specific catalog item by id or by path

<https://docs.dremio.com/rest-api/catalog/get-catalog-id.html>      <https://docs.dremio.com/rest-api/catalog/get-catalog-path.html>

#### Parameters

- **token** – auth token from previous login attempt
- **base\_url** – base Dremio url
- **cid** – unique dremio id for resource
- **path** – list ['space', 'folder', 'vds']
- **ssl\_verify** – ignore ssl errors if False

**Returns** json of resource

`dremio_client.model.endpoints.collaboration_tags(token, base_url, cid, ssl_verify=True)`

fetch tags for a catalog entry

<https://docs.dremio.com/rest-api/user/get-catalog-collaboration.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **cid** – id of a catalog entity
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.collaboration_wiki(token, base_url, cid, ssl_verify=True)`

fetch wiki for a catalog entry

<https://docs.dremio.com/rest-api/user/get-catalog-collaboration.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **cid** – id of a catalog entity
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.create_queue(token, base_url, json, ssl_verify=True)`  
create a single queue

<https://docs.dremio.com/rest-api/reflections/post-wlm-queue.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **json** – json document for new queue
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.create_reflection(token, base_url, json, ssl_verify=True)`  
create a single reflection

<https://docs.dremio.com/rest-api/reflections/post-reflection.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **json** – json document for new reflection
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.delete_catalog(token, base_url, cid, tag, ssl_verify=True)`  
remove a catalog item from Dremio

<https://docs.dremio.com/rest-api/catalog/delete-catalog-id.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **cid** – id of a catalog entity
- **tag** – version tag of entity
- **ssl\_verify** – ignore ssl errors if False

**Returns** None

`dremio_client.model.endpoints.delete_personal_access_token(token, base_url, uid, tid=None, ssl_verify=True)`  
create a pat for a given user

<https://docs.dremio.com/rest-api/user/delete-user-uid-token.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **uid** – id user
- **tid** – label of token (optional)
- **ssl\_verify** – ignore ssl errors if False

**Returns** updated catalog entity

`dremio_client.model.endpoints.delete_queue(token, base_url, queueid, ssl_verify=True)`  
delete a single queue by id

<https://docs.dremio.com/rest-api/reflections/delete-wlm-queue.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **queueid** – id of the queue to delete
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.delete_reflection(token, base_url, reflectionid, ssl_verify=True)`  
delete a single reflection by id

<https://docs.dremio.com/rest-api/reflections/delete-reflection.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **reflectionid** – id of the reflection to delete
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.graph(token, base_url, cid=None, ssl_verify=True)`  
Retrieves graph information about a specific catalog entity by id

<https://docs.dremio.com/rest-api/catalog/get-catalog-id-graph.html>

#### Parameters

- **token** – auth token from previous login attempt
- **base\_url** – base Dremio url
- **cid** – unique dremio id for resource
- **ssl\_verify** – ignore ssl errors if False

**Returns** json of resource

`dremio_client.model.endpoints.group(token, base_url, gid=None, name=None, ssl_verify=True)`  
fetch a group based on id or name



<https://docs.dremio.com/rest-api/reflections/get-group.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **gid** – unique dremio id for group
- **name** – name for a group
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.job_results` (*token, base\_url, job\_id, offset=0, limit=100, ssl\_verify=True*)

fetch job results

<https://docs.dremio.com/rest-api/jobs/get-job.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **job\_id** – job id (as returned by sql)
- **offset** – offset of result set to return
- **limit** – number of results to return (max 500)
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.job_status` (*token, base\_url, job\_id, ssl\_verify=True*)

fetch job status

<https://docs.dremio.com/rest-api/jobs/get-job.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **job\_id** – job id (as returned by sql)
- **ssl\_verify** – ignore ssl errors if False

**Returns** status object

`dremio_client.model.endpoints.modify_queue` (*token, base\_url, queueid, json, ssl\_verify=True*)

update a single queue by id

<https://docs.dremio.com/rest-api/reflections/put-wlm-queue.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **queueid** – id of the reflection to fetch
- **json** – json document for modified queue

- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.modify_reflection(token, base_url, reflectionid, json, ssl_verify=True)`

update a single reflection by id

<https://docs.dremio.com/rest-api/reflections/put-reflection.html>

**Parameters**

- **token** – auth token
- **base\_url** – sql query
- **reflectionid** – id of the reflection to fetch
- **json** – json document for modified reflection
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.modify_rules(token, base_url, json, ssl_verify=True)`

update wlm rules. Order of rules array is important!

The order of the rules is the order in which they will be applied. If a rule isn't included it will be deleted new ones will be created <https://docs.dremio.com/rest-api/reflections/put-wlm-rule.html>

**Parameters**

- **token** – auth token
- **base\_url** – sql query
- **json** – json document for modified reflection
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.personal_access_token(token, base_url, uid, ssl_verify=True)`

fetch a PAT for a user based on id

<https://docs.dremio.com/rest-api/user/get-user-id-token.html>

**Parameters**

- **token** – auth token
- **base\_url** – sql query
- **uid** – id of a user
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.promote_catalog(token, base_url, cid, json, ssl_verify=True)`

promote a catalog entity (only works on folders and files in sources)

<https://docs.dremio.com/rest-api/catalog/post-catalog-id.html>

**Parameters**

- **token** – auth token

- **base\_url** – sql query
- **cid** – id of catalog entity
- **json** – json document for new catalog entity
- **ssl\_verify** – ignore ssl errors if False

**Returns** updated catalog entity

`dremio_client.model.endpoints.reflection(token, base_url, reflectionid, ssl_verify=True)`  
fetch a single reflection by id

<https://docs.dremio.com/rest-api/reflections/get-reflection.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **reflectionid** – id of the reflection to fetch
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.reflections(token, base_url, summary=False, ssl_verify=True)`  
fetch all reflections

<https://docs.dremio.com/rest-api/reflections/get-reflection.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **summary** – fetch only the reflection summary
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.refresh_pds(token, base_url, pid, ssl_verify=True)`  
refresh a physical dataset and all its child reflections

<https://docs.dremio.com/rest-api/catalog/post-catalog-id-refresh.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **pid** – id of a catalog entity
- **ssl\_verify** – ignore ssl errors if False

**Returns** None

`dremio_client.model.endpoints.set_catalog(token, base_url, json, ssl_verify=True)`  
add a new catalog entity

<https://docs.dremio.com/rest-api/catalog/post-catalog.html>

#### Parameters

- **token** – auth token

- **base\_url** – sql query
- **json** – json document for new catalog entity
- **ssl\_verify** – ignore ssl errors if False

**Returns** new catalog entity

```
dremio_client.model.endpoints.set_collaboration_tags(token, base_url, cid, tags,  
                                                    ssl_verify=True)
```

set tags on a given catalog entity

<https://docs.dremio.com/rest-api/catalog/post-catalog-collaboration.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **cid** – id of a catalog entity
- **tags** – list of strings for tags
- **ssl\_verify** – ignore ssl errors if False

**Returns** None

```
dremio_client.model.endpoints.set_collaboration_wiki(token, base_url, cid, wiki,  
                                                    ssl_verify=True)
```

set wiki on a given catalog entity

<https://docs.dremio.com/rest-api/catalog/post-catalog-collaboration.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **cid** – id of a catalog entity
- **wiki** – text representing markdown for entity
- **ssl\_verify** – ignore ssl errors if False

**Returns** None

```
dremio_client.model.endpoints.set_personal_access_token(token, base_url, uid,  
                                                         label, lifetime=24,  
                                                         ssl_verify=True)
```

create a pat for a given user

<https://docs.dremio.com/rest-api/user/post-user-uid-token.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **uid** – id user
- **label** – label of token
- **lifetime** – lifetime in hours of token
- **ssl\_verify** – ignore ssl errors if False

**Returns** updated catalog entity

`dremio_client.model.endpoints.sql` (*token, base\_url, query, context=None, ssl\_verify=True*)  
submit job w/ given sql

<https://docs.dremio.com/rest-api/sql/post-sql.html>

#### Parameters

- **token** – auth token
- **base\_url** – base Dremio url
- **query** – sql query
- **context** – optional dremio context
- **ssl\_verify** – ignore ssl errors if False

**Returns** job id json object

`dremio_client.model.endpoints.update_catalog` (*token, base\_url, cid, json, ssl\_verify=True*)  
update a catalog entity

<https://docs.dremio.com/rest-api/catalog/put-catalog-id.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **cid** – id of catalog entity
- **json** – json document for new catalog entity
- **ssl\_verify** – ignore ssl errors if False

**Returns** updated catalog entity

`dremio_client.model.endpoints.user` (*token, base\_url, uid=None, name=None, ssl\_verify=True*)  
fetch user based on id or name <https://docs.dremio.com/rest-api/reflections/get-user.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **uid** – unique dremio id for user
- **name** – name for a user
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.votes` (*token, base\_url, ssl\_verify=True*)  
fetch all votes

<https://docs.dremio.com/rest-api/reflections/get-vote.html>

#### Parameters

- **token** – auth token
- **base\_url** – sql query
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.wlm_queues` (*token, base\_url, ssl\_verify=True*)  
fetch all wlm queues

<https://docs.dremio.com/rest-api/reflections/get-wlm-queue.html>

**Parameters**

- **token** – auth token
- **base\_url** – sql query
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

`dremio_client.model.endpoints.wlm_rules` (*token, base\_url, ssl\_verify=True*)  
fetch all wlm rules

<https://docs.dremio.com/rest-api/reflections/get-wlm-queue.html>

**Parameters**

- **token** – auth token
- **base\_url** – sql query
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

## Module contents

### dremio\_client.util package

#### Submodules

#### dremio\_client.util.query module

`dremio_client.util.query.refresh_metadata` (*token, base\_url, table, ssl\_verify=True*)  
Refresh the metadata of a given PDS

This requests a metadata refresh of a given Physical Dataset

**Parameters**

- **token** – API token from auth
- **base\_url** – base url of Dremio instance
- **table** – valid dremio table name
- **ssl\_verify** – verify ssl on web requests

**Raise** DremioException if job failed

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Returns** None

**Example**

```
>>> refresh_metadata('abc', 'http://localhost:9047', 'source.pds')
```

```
dremio_client.util.query.run(token, base_url, query, context=None, sleep_time=10,
                             ssl_verify=True)
```

Run a single sql query

This runs a single sql query against the rest api and returns a json document of the results

#### Parameters

- **token** – API token from auth
- **base\_url** – base url of Dremio instance
- **query** – valid sql query
- **context** – optional context in which to execute the query
- **sleep\_time** – seconds to sleep between checking for finished state
- **ssl\_verify** – verify ssl on web requests

**Raise** DremioException if job failed

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Returns** json array of result rows

#### Example

```
>>> run('abc', 'http://localhost:9047', 'select * from sys.options')
[{'record': '1'}, {'record': '2'}]
```

```
dremio_client.util.query.run_async(token, base_url, query, context=None, sleep_time=10,
                                   ssl_verify=True)
```

Run a single sql query asynchronously

This executes a single sql query against the rest api asynchronously and returns a future for the result

#### Parameters

- **token** – API token from auth
- **base\_url** – base url of Dremio instance
- **query** – valid sql query
- **context** – optional context in which to execute the query
- **sleep\_time** – seconds to sleep between checking for finished state
- **ssl\_verify** – verify ssl on web requests

**Raise** DremioException if job failed

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Returns** concurrent.futures.Future for the result

#### Example

```
>>> f = run_async('abc', 'http://localhost:9047', 'select * from sys.options')
>>> f.result()
[{'record': '1'}, {'record': '2'}]
```

## dremio\_client.util.refresh module

**Warning:** dremio\_client.util.refresh.refresh\_reflections\_of\_one\_dataset Caution using this function. It can has an critical impact on source system due to disabled reflection.

### Module contents

dremio\_client.util.**run**(token, base\_url, query, context=None, sleep\_time=10, ssl\_verify=True)

Run a single sql query

This runs a single sql query against the rest api and returns a json document of the results

#### Parameters

- **token** – API token from auth
- **base\_url** – base url of Dremio instance
- **query** – valid sql query
- **context** – optional context in which to execute the query
- **sleep\_time** – seconds to sleep between checking for finished state
- **ssl\_verify** – verify ssl on web requests

**Raise** DremioException if job failed

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Returns** json array of result rows

#### Example

```
>>> run('abc', 'http://localhost:9047', 'select * from sys.options')
[{'record': '1'}, {'record': '2'}]
```

dremio\_client.util.**run\_async**(token, base\_url, query, context=None, sleep\_time=10, ssl\_verify=True)

Run a single sql query asynchronously

This executes a single sql query against the rest api asynchronously and returns a future for the result

#### Parameters

- **token** – API token from auth
- **base\_url** – base url of Dremio instance
- **query** – valid sql query
- **context** – optional context in which to execute the query
- **sleep\_time** – seconds to sleep between checking for finished state
- **ssl\_verify** – verify ssl on web requests

**Raise** DremioException if job failed

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Returns** concurrent.futures.Future for the result

#### Example



```
>>> f = run_async('abc', 'http://localhost:9047', 'select * from sys.options')
>>> f.result()
[{'record': '1'}, {'record': '2'}]
```

`dremio_client.util.refresh_metadata(token, base_url, table, ssl_verify=True)`

Refresh the metadata of a given PDS

This requests a metadata refresh of a given Physical Dataset

#### Parameters

- **token** – API token from auth
- **base\_url** – base url of Dremio instance
- **table** – valid dremio table name
- **ssl\_verify** – verify ssl on web requests

**Raise** DremioException if job failed

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Returns** None

#### Example

```
>>> refresh_metadata('abc', 'http://localhost:9047', 'source.pds')
```

`dremio_client.util.promote_catalog(client, catalog, file_format='parquet', **kwargs)`

`dremio_client.util.refresh_vds_reflection_by_path(client, path=None)`

By providing a path from VDS the reflection of that vds will be refreshed Script will find which pds is responsible for the reflection and trigger the refresh based on pds

**Parameters** **path** – list ['space', 'folder', 'vds']

`dremio_client.util.refresh_reflections_of_one_dataset(client, path=None)`

By providing a path from dataset the reflection of that dataset will be refreshed through reenablenabled reflections of dataset and all reflections from derived dataset will be refreshed

**Parameters** **path** – list ['space', 'folder', 'vds']

## 6.1.2 Submodules

### 6.1.3 dremio\_client.cli module

Console script for dremio\_client.

`dremio_client.cli.print_version(ctx, param, value)`

### 6.1.4 dremio\_client.dremio\_client module

Main module.

**class** `dremio_client.dremio_client.DremioClient(config)`

Bases: object

**data**

**get\_item** (*cid=None, path=None*)

**group** (*gid=None, name=None*)

return details for a group

User must supply one of gid or name. gid takes precedence if both are supplied .. note:: can only be run by admin

**Parameters**

- **gid** – group id
- **name** – group name

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Raise** DremioNotFoundException group could not be found

**Returns** group info as a dict

**personal\_access\_token** (*uid*)

return a list of personal access tokens for a user

---

**Note:** can only be run for the logged in user

---

**Parameters** **uid** – user id

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Raise** DremioNotFoundException user could not be found

**Returns** personal access token list

**query** (*sql, pandas=True, method='flight'*)

**reflections**

**simple** ()

**user** (*uid=None, name=None*)

return details for a user

User must supply one of uid or name. uid takes precedence if both are supplied .. note:: can only be run by admin

**Parameters**

- **uid** – group id
- **name** – group name

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Raise** DremioNotFoundException user could not be found

**Returns** user info as a dict

**votes**

**wlm\_queues**

**wlm\_rules**

### 6.1.5 dremio\_client.dremio\_simple\_client module

```
class dremio_client.dremio_simple_client.SimpleClient(config)
    Bases: object

    cancel_job(jobid)
        cancel running job with job id = jobid

        https://docs.dremio.com/rest-api/jobs/post-job.html

        Parameters jobid – id of the job to cancel

        Returns result object

        :exception DremioNotFoundException no job found :exception DremioBadRequestException job already
        finished

    catalog()

    catalog_item(cid, path)

    collaboration_tag(cid)
        returns a list of tags for catalog entity

        Parameters cid – catalog entity id

        Raise DremioBadRequestException if tags can't exist on this entity

        Raise DremioUnauthorizedException if token is incorrect or invalid

        Raise DremioPermissionException user does not have permission

        Raise DremioNotFoundException user could not be found

        Returns list of tags

    collaboration_wiki(cid)
        returns a wiki details for catalog entity

        Parameters cid – catalog entity id

        Raise DremioUnauthorizedException if token is incorrect or invalid

        Raise DremioPermissionException user does not have permission

        Raise DremioNotFoundException user could not be found

        Returns wiki details

    create_queue(json)
        create a single queue

        https://docs.dremio.com/rest-api/reflections/post-wlm-queue.html

        Parameters json – json document for new queue

        Returns result object

    create_reflection(json)
        create a single reflection

        https://docs.dremio.com/rest-api/reflections/post-reflection.html

        Parameters json – json document for new reflection

        Returns result object
```

**delete\_catalog** (*cid, tag*)

remove a catalog item from Dremio

<https://docs.dremio.com/rest-api/catalog/delete-catalog-id.html>

**Parameters**

- **cid** – id of a catalog entity
- **tag** – version tag of entity

**Returns** None

**delete\_personal\_access\_token** (*uid*)

create a pat for a given user

<https://docs.dremio.com/rest-api/user/delete-user-uid-token.html>

**Parameters** **uid** – id user

**Returns** updated catalog entity

**delete\_queue** (*queueid*)

delete a single queue by id

<https://docs.dremio.com/rest-api/reflections/delete-wlm-queue.html>

**Parameters** **queueid** – id of the queue to delete

**Returns** result object

**delete\_reflection** (*reflectionid*)

delete a single reflection by id

<https://docs.dremio.com/rest-api/reflections/delete-reflection.html>

**Parameters** **reflectionid** – id of the reflection to fetch

**Returns** result object

**graph** (*cid*)

**group** (*gid=None, name=None*)

return details for a group

User must supply one of gid or name. gid takes precedence if both are supplied .. note:: can only be run by admin

**Parameters**

- **gid** – group id
- **name** – group name

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Raise** DremioNotFoundException group could not be found

**Returns** group info as a dict

**job\_results** (*jobid*)

**job\_status** (*jobid*)

**modify\_queue** (*queueid*, *json*)

update a single queue by id

<https://docs.dremio.com/rest-api/reflections/put-wlm-queue.html>**Parameters**

- **queueid** – id of the reflection to fetch
- **json** – json document for modified queue

**Returns** result object**modify\_reflection** (*reflectionid*, *json*)

update a single reflection by id

<https://docs.dremio.com/rest-api/reflections/put-reflection.html>**Parameters**

- **reflectionid** – id of the reflection to fetch
- **json** – json document for modified reflection

**Returns** result object**modify\_rules** (*json*)

update wlm rules. Order of rules array is important!

The order of the rules is the order in which they will be applied. If a rule isn't included it will be deleted new ones will be created <https://docs.dremio.com/rest-api/reflections/put-wlm-rule.html>

**Parameters** **json** – json document for modified reflection**Returns** result object**personal\_access\_token** (*uid*)

return a list of personal access tokens for a user

---

**Note:** can only be run for the logged in user

---

**Parameters** **uid** – user id**Raise** DremioUnauthorizedException if token is incorrect or invalid**Raise** DremioPermissionException user does not have permission**Raise** DremioNotFoundException user could not be found**Returns** personal access token list**promote\_catalog** (*cid*, *json*)

promote a catalog entity

<https://docs.dremio.com/rest-api/catalog/put-catalog-id.html>**Parameters**

- **cid** – id of catalog entity
- **json** – json document for new catalog entity

**Returns** updated catalog entity

**query** (*query*, *context=None*, *sleep\_time=10*, *asynchronous=False*)

Run a single sql query asynchronously

This executes a single sql query against the rest api asynchronously and returns a future for the result

**Parameters**

- **query** – valid sql query
- **context** – optional context in which to execute the query
- **sleep\_time** – seconds to sleep between checking for finished state
- **asynchronous** – boolean execute asynchronously

**Raise** DremioException if job failed

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Returns** concurrent.futures.Future for the result

**Example**

```
>>> client =
>>> f = query('select * from sys.options', asynchronous=True)
>>> f.result()
[{'record': '1'}, {'record': '2'}]
```

**reflection** (*reflectionid*)

**reflections** (*summary=False*)

**refresh\_metadata** (*table*)

Refresh the metadata for a given physical dataset

**Parameters** **table** – the physical dataset to be refreshed

**Raise** DremioException if job failed

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Returns** None

**refresh\_pds** (*pid*)

refresh a physical dataset and all its child reflections

<https://docs.dremio.com/rest-api/catalog/post-catalog-id-refresh.html>

**Parameters** **pid** – id of a catalog entity

**Returns** None

**refresh\_reflections\_of\_one\_dataset** (*path*)

Refresh the reflection for a given dataset Refresh will disable and enable reflection. As long reflection will be reenabled all queries will be redirected to source dataset. All VDS Reflection derived from this VDS will be refreshed as well

**Parameters** **path** – list ['space', 'folder', 'vds']

**Returns** None

**refresh\_vds\_reflection\_by\_path** (*path*)

Refresh the reflection for a given virtual dataset

**Parameters** **path** – list ['space', 'folder', 'vds']

**Returns** None

**set\_catalog** (*json*)  
add a new catalog entity

<https://docs.dremio.com/rest-api/catalog/post-catalog.html>

**Parameters** *json* – json document for new catalog entity

**Returns** new catalog entity

**set\_collaboration\_tag** (*cid, tags*)  
returns a list of tags for catalog entity

**Parameters**

- **cid** – catalog entity id
- **tags** – string list

**Raise** DremioBadRequestException if tags can't exist on this entity

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Raise** DremioNotFoundException user could not be found

**Returns** list of tags

**set\_personal\_access\_token** (*uid, label, lifetime=24*)  
create a pat for a given user

<https://docs.dremio.com/rest-api/user/post-user-uid-token.html>

**Parameters**

- **uid** – id user
- **label** – label of token
- **lifetime** – lifetime in hours of token

**Returns** updated catalog entity

**sql** (*query, context=None*)

**update\_catalog** (*cid, json*)  
update a catalog entity

<https://docs.dremio.com/rest-api/catalog/put-catalog-id.html>

**Parameters**

- **cid** – id of catalog entity
- **json** – json document for new catalog entity

**Returns** updated catalog entity

**user** (*uid=None, name=None*)  
return details for a user

User must supply one of uid or name. uid takes precedence if both are supplied .. note:: can only be run by admin

**Parameters**

- **uid** – group id
- **name** – group name

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Raise** DremioNotFoundException user could not be found

**Returns** user info as a dict

**votes ()**

return details all reflection votes

Summarizes all votes in the system <https://docs.dremio.com/rest-api/votes/get-vote.html> .. note:: can only be run by admin

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Returns** votes as a list of dicts

**wlm\_queues ()**

return details all workload management queues

Summarizes all workload management queues in the system <https://docs.dremio.com/rest-api/wlm/get-wlm-queue.html> .. note:: can only be run by admin .. note:: Enterprise only

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Raise** DremioNotFoundException queues not found

**Returns** queues as a list of dicts

**wlm\_rules ()**

return details all workload management rules

Summarizes all workload management rules in the system <https://docs.dremio.com/rest-api/wlm/get-wlm-rule.html> .. note:: can only be run by admin .. note:: Enterprise only

**Raise** DremioUnauthorizedException if token is incorrect or invalid

**Raise** DremioPermissionException user does not have permission

**Raise** DremioNotFoundException ruleset is not found

**Returns** rules as a list of dicts

## 6.1.6 dremio\_client.error module

**exception** `dremio_client.error.DremioBadRequestException` (*msg*, *original\_exception*, *response=None*)

Bases: `dremio_client.error.DremioException`

**exception** `dremio_client.error.DremioException` (*msg*, *original\_exception*, *response=None*)

Bases: `Exception`

base dremio exception

**exception** `dremio_client.error.DremioNotFoundException` (*msg*, *original\_exception*, *response=None*)

Bases: `dremio_client.error.DremioException`



**exception** `dremio_client.error.DremioPermissionException(msg, original_exception, response=None)`

Bases: `dremio_client.error.DremioException`

**exception** `dremio_client.error.DremioUnauthorizedException(msg, original_exception, response=None)`

Bases: `dremio_client.error.DremioException`

## 6.1.7 Module contents

`dremio_client.init(config_dir=None, simple_client=False, config_dict=None)`

create a new Dremio client object

This returns a rich client by default. This client abstracts the Dremio catalog into a a set of Python objects and enables <Tab> completion where possible. It also supports fetching datasets via flight or odbc

The simple client simply wraps the Dremio REST endpoints an returns dict objects

### Parameters

- **config\_dir** – optional directory to look for config in
- **simple\_client** – return the ‘simple’ client.
- **config\_dict** – dictionary of extra config arguments

**Returns** either a simple or rich client

### Example

```
>>> client = init('/my/config/dir')
```

`dremio_client.catalog(token, base_url, ssl_verify=True)`

<https://docs.dremio.com/rest-api/catalog/get-catalog.html> populate the root dremio catalog

### Parameters

- **token** – auth token from previous login attempt
- **base\_url** – base Dremio url
- **ssl\_verify** – ignore ssl errors if False

**Returns** json of root resource

`dremio_client.catalog_item(token, base_url, cid=None, path=None, ssl_verify=True)`

fetch a specific catalog item by id or by path

<https://docs.dremio.com/rest-api/catalog/get-catalog-id.html>

<https://docs.dremio.com/rest-api/catalog/get-catalog-path.html>

### Parameters

- **token** – auth token from previous login attempt
- **base\_url** – base Dremio url
- **cid** – unique dremio id for resource
- **path** – list ['space', 'folder', 'vds']
- **ssl\_verify** – ignore ssl errors if False

**Returns** json of resource

`dremio_client.sql` (*token, base\_url, query, context=None, ssl\_verify=True*)  
submit job w/ given sql

<https://docs.dremio.com/rest-api/sql/post-sql.html>

**Parameters**

- **token** – auth token
- **base\_url** – base Dremio url
- **query** – sql query
- **context** – optional dremio context
- **ssl\_verify** – ignore ssl errors if False

**Returns** job id json object

`dremio_client.job_status` (*token, base\_url, job\_id, ssl\_verify=True*)  
fetch job status

<https://docs.dremio.com/rest-api/jobs/get-job.html>

**Parameters**

- **token** – auth token
- **base\_url** – sql query
- **job\_id** – job id (as returned by sql)
- **ssl\_verify** – ignore ssl errors if False

**Returns** status object

`dremio_client.job_results` (*token, base\_url, job\_id, offset=0, limit=100, ssl\_verify=True*)  
fetch job results

<https://docs.dremio.com/rest-api/jobs/get-job.html>

**Parameters**

- **token** – auth token
- **base\_url** – sql query
- **job\_id** – job id (as returned by sql)
- **offset** – offset of result set to return
- **limit** – number of results to return (max 500)
- **ssl\_verify** – ignore ssl errors if False

**Returns** result object

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

## 7.1 Types of Contributions

### 7.1.1 Report Bugs

Report bugs at [https://github.com/rymurr/dremio\\_client/issues](https://github.com/rymurr/dremio_client/issues).

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

### 7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

## 7.1.4 Write Documentation

Dremio client could always use more documentation, whether as part of the official Dremio client docs, in docstrings, or even on the web in blog posts, articles, and such.

## 7.1.5 Submit Feedback

The best way to send feedback is to file an issue at [https://github.com/rymurr/dremio\\_client/issues](https://github.com/rymurr/dremio_client/issues).

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 7.2 Get Started!

Ready to contribute? Here's how to set up *dremio\_client* for local development.

1. Fork the *dremio\_client* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dremio_client.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv dremio_client
$ cd dremio_client/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 dremio_client tests
$ python setup.py test or py.test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

## 7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check [https://travis-ci.org/rymurr/dremio\\_client/pull\\_requests](https://travis-ci.org/rymurr/dremio_client/pull_requests) and make sure that the tests pass for all supported Python versions.

## 7.4 Tips

To run a subset of tests:

```
$ python -m pytest tests.test_dremio_client
```

## 7.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.



### 8.1 Development Lead

- Ryan Murray <[rymurr@gmail.com](mailto:rymurr@gmail.com)>

### 8.2 Contributors

- markfjohnson
- insatomcat
- narendrans
- viktordremio





#### 9.1 0.1.0 (2019-07-04)

- First code, unreleased.

#### 9.2 0.2.4 (2019-08-13)

- Alpha quality release with partial implementation of features. First PyPI release.

#### 9.3 0.3.0 (2019-08-14)

- Beta quality release for GET based endpoints

#### 9.4 0.5.0 (2019-10-30)

- Most endpoint supported
- flight support with tls & auth
- minor bugfixes

#### 9.5 0.6.3 (2020-01-29)

- All endpoints supported
- lots of bugfixes
- utilities for building spaces/folders vds

- removed recordclass library in favor of attrs

## 9.6 0.8.4 (2020-01-29)

- history re-written

## 9.7 0.8.8 (2020-02-19)

- minor bug fixes

## 9.8 0.8.9 (2020-02-19)

- ensure use correct name for Spaces

## 9.9 0.8.11 (2020-03-09)

- bug fixes when creating sources and folders

## 9.10 0.9.2 (2020-03-17)

- add promote endpoint
- update versions from pyup

## 9.11 0.10.1 (2020-03-19)

- fix ssl verify flag in cli

## 9.12 0.11.1 (2020-04-09)

- a number of bug fixes and improvements from the community

## 9.13 0.12.0 (2020-04-22)

- add graph endpoint and cli flag
- add more detailed exception handling

## 9.14 0.12.1 (2020-05-04)

- few bug fixes and API extensions

## 9.15 0.13.0 (2020-05-21)

- add version flag
- upgrades and additions to some commands

## 9.16 0.13.1 (2020-05-21)

- simplify delete cli command

## 9.17 0.13.2 (2020-05-22)

- bugfixes



## CHAPTER 10

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### d

- `dremio_client`, [37](#)
- `dremio_client.auth`, [11](#)
- `dremio_client.auth.basic`, [11](#)
- `dremio_client.cli`, [29](#)
- `dremio_client.conf`, [12](#)
- `dremio_client.conf.cli_helper`, [12](#)
- `dremio_client.conf.config_parser`, [12](#)
- `dremio_client.dremio_client`, [29](#)
- `dremio_client.dremio_simple_client`, [31](#)
- `dremio_client.error`, [36](#)
- `dremio_client.flight`, [12](#)
- `dremio_client.flight.flight_auth`, [12](#)
- `dremio_client.model`, [26](#)
- `dremio_client.model.catalog`, [13](#)
- `dremio_client.model.data`, [13](#)
- `dremio_client.model.endpoints`, [17](#)
- `dremio_client.util`, [28](#)
- `dremio_client.util.query`, [26](#)





## A

AccessControl (class in *dremio\_client.model.data*), 13

AccessControlList (class in *dremio\_client.model.data*), 13

add() (*dremio\_client.model.data.Root* method), 16

add\_by\_path() (*dremio\_client.model.data.Root* method), 16

auth() (in module *dremio\_client.auth*), 11

authenticate() (*dremio\_client.flight.flight\_auth.HttpDremioClientAuthHandler* method), 12

## B

basic\_auth() (in module *dremio\_client.auth*), 11

build\_config() (in module *dremio\_client.conf*), 12

build\_config() (in module *dremio\_client.conf.config\_parser*), 12

## C

cancel\_job() (*dremio\_client.dremio\_simple\_client.SimpleClient* method), 31

cancel\_job() (in module *dremio\_client.model.endpoints*), 17

Catalog (class in *dremio\_client.model.data*), 13

catalog() (*dremio\_client.dremio\_simple\_client.SimpleClient* method), 31

catalog() (in module *dremio\_client*), 37

catalog() (in module *dremio\_client.model.catalog*), 13

catalog() (in module *dremio\_client.model.endpoints*), 18

catalog\_item() (*dremio\_client.dremio\_simple\_client.SimpleClient* method), 31

catalog\_item() (in module *dremio\_client*), 37

catalog\_item() (in module *dremio\_client.model.endpoints*), 18

collaboration\_tag() (*dremio\_client.dremio\_simple\_client.SimpleClient* method), 31

collaboration\_tags() (in module *dremio\_client.model.endpoints*), 18

collaboration\_wiki() (*dremio\_client.dremio\_simple\_client.SimpleClient* method), 31

collaboration\_wiki() (in module *dremio\_client.model.endpoints*), 18

commit() (*dremio\_client.model.data.Catalog* method), 13

config\_auth() (in module *dremio\_client.auth*), 11

connect() (in module *dremio\_client.flight*), 12

create() (in module *dremio\_client.model.data*), 17

create\_folder() (in module *dremio\_client.model.data*), 17

create\_queue() (*dremio\_client.dremio\_simple\_client.SimpleClient* method), 31

create\_queue() (in module *dremio\_client.model.endpoints*), 19

create\_reflection() (*dremio\_client.dremio\_simple\_client.SimpleClient* method), 31

create\_reflection() (in module *dremio\_client.model.endpoints*), 19

create\_space() (in module *dremio\_client.model.data*), 17

create\_vds() (in module *dremio\_client.model.data*), 17

## D

data (*dremio\_client.dremio\_client.DremioClient* attribute), 29

Dataset (class in *dremio\_client.model.data*), 14

DatasetMetadata (class in *dremio\_client.model.data*), 14

delete() (*dremio\_client.model.data.Catalog* method), 13

delete\_catalog() (*dremio\_client.dremio\_simple\_client.SimpleClient* method), 31

delete\_catalog() (in module *dremio\_client.model.endpoints*), 19

`delete_personal_access_token()`  
     (*dremio\_client.dremio\_simple\_client.SimpleClient*  
     *method*), 32  
`delete_personal_access_token()` (in module  
     *dremio\_client.model.endpoints*), 19  
`delete_queue()` (*dremio\_client.dremio\_simple\_client.SimpleClient*  
     *method*), 32  
`delete_queue()` (in module  
     *dremio\_client.model.endpoints*), 20  
`delete_reflection()`  
     (*dremio\_client.dremio\_simple\_client.SimpleClient*  
     *method*), 32  
`delete_reflection()` (in module  
     *dremio\_client.model.endpoints*), 20  
`dremio_client` (module), 37  
`dremio_client.auth` (module), 11  
`dremio_client.auth.basic` (module), 11  
`dremio_client.cli` (module), 29  
`dremio_client.conf` (module), 12  
`dremio_client.conf.cli_helper` (module), 12  
`dremio_client.conf.config_parser` (mod-  
     ule), 12  
`dremio_client.dremio_client` (module), 29  
`dremio_client.dremio_simple_client` (mod-  
     ule), 31  
`dremio_client.error` (module), 36  
`dremio_client.flight` (module), 12  
`dremio_client.flight.flight_auth` (mod-  
     ule), 12  
`dremio_client.model` (module), 26  
`dremio_client.model.catalog` (module), 13  
`dremio_client.model.data` (module), 13  
`dremio_client.model.endpoints` (module), 17  
`dremio_client.util` (module), 28  
`dremio_client.util.query` (module), 26  
`DremioBadRequestException`, 36  
`DremioClient` (class in *dremio\_client.dremio\_client*),  
     29  
`DremioException`, 36  
`DremioNotFoundException`, 36  
`DremioPermissionException`, 36  
`DremioUnauthorizedException`, 37  
**F**  
`File` (class in *dremio\_client.model.data*), 14  
`FileMetaData` (class in *dremio\_client.model.data*), 14  
`Folder` (class in *dremio\_client.model.data*), 14  
`FolderMetaData` (class in *dremio\_client.model.data*),  
     15  
**G**  
`get()` (*dremio\_client.model.data.Catalog* method), 13  
`get_base_url_token()` (in module  
     *dremio\_client.conf*), 12  
`get_base_url_token()` (in module  
     *dremio\_client.conf.cli\_helper*), 12  
`get_graph()` (*dremio\_client.model.data.Dataset*  
     *method*), 14  
`get_item()` (*dremio\_client.dremio\_client.DremioClient*  
     *method*), 29  
`get_path()` (in module *dremio\_client.model.data*), 17  
`get_table()` (*dremio\_client.model.data.Dataset*  
     *method*), 14  
`get_token()` (*dremio\_client.flight.flight\_auth.HttpDremioClientAuthHa*  
     *method*), 12  
`graph()` (*dremio\_client.dremio\_simple\_client.SimpleClient*  
     *method*), 32  
`graph()` (in module *dremio\_client.model.endpoints*), 20  
`group()` (*dremio\_client.dremio\_client.DremioClient*  
     *method*), 29  
`group()` (*dremio\_client.dremio\_simple\_client.SimpleClient*  
     *method*), 32  
`group()` (in module *dremio\_client.model.endpoints*), 20  
**H**  
`Home` (class in *dremio\_client.model.data*), 15  
`HttpDremioClientAuthHandler` (class in  
     *dremio\_client.flight.flight\_auth*), 12  
**I**  
`init()` (in module *dremio\_client*), 37  
`insert()` (*dremio\_client.model.data.Catalog* method),  
     14  
**J**  
`job_results()` (*dremio\_client.dremio\_simple\_client.SimpleClient*  
     *method*), 32  
`job_results()` (in module *dremio\_client*), 38  
`job_results()` (in module  
     *dremio\_client.model.endpoints*), 21  
`job_status()` (*dremio\_client.dremio\_simple\_client.SimpleClient*  
     *method*), 32  
`job_status()` (in module *dremio\_client*), 38  
`job_status()` (in module  
     *dremio\_client.model.endpoints*), 21  
**K**  
`keys()` (*dremio\_client.model.data.Catalog* method), 14  
**L**  
`login()` (in module *dremio\_client.auth.basic*), 11  
**M**  
`make_reflection()` (in module  
     *dremio\_client.model.data*), 17  
`make_tags()` (in module *dremio\_client.model.data*),  
     17

[make\\_vote\(\)](#) (in module `dremio_client.model.data`), [17](#)  
[make\\_wiki\(\)](#) (in module `dremio_client.model.data`), [17](#)  
[make\\_wlm\\_queue\(\)](#) (in module `dremio_client.model.data`), [17](#)  
[make\\_wlm\\_rule\(\)](#) (in module `dremio_client.model.data`), [17](#)  
[metadata\\_refresh\(\)](#) (`dremio_client.model.data.PhysicalDataset` method), [15](#)  
[MetadataPolicy](#) (class in `dremio_client.model.data`), [15](#)  
[modify\\_queue\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [32](#)  
[modify\\_queue\(\)](#) (in module `dremio_client.model.endpoints`), [21](#)  
[modify\\_reflection\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [33](#)  
[modify\\_reflection\(\)](#) (in module `dremio_client.model.endpoints`), [22](#)  
[modify\\_rules\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [33](#)  
[modify\\_rules\(\)](#) (in module `dremio_client.model.endpoints`), [22](#)

## P

[personal\\_access\\_token\(\)](#) (`dremio_client.dremio_client.DremioClient` method), [30](#)  
[personal\\_access\\_token\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [33](#)  
[personal\\_access\\_token\(\)](#) (in module `dremio_client.model.endpoints`), [22](#)  
[PhysicalDataset](#) (class in `dremio_client.model.data`), [15](#)  
[print\\_version\(\)](#) (in module `dremio_client.cli`), [29](#)  
[promote\(\)](#) (`dremio_client.model.data.Catalog` method), [14](#)  
[promote\\_catalog\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [33](#)  
[promote\\_catalog\(\)](#) (in module `dremio_client.model.endpoints`), [22](#)  
[promote\\_catalog\(\)](#) (in module `dremio_client.util`), [29](#)

## Q

[query\(\)](#) (`dremio_client.dremio_client.DremioClient` method), [30](#)  
[query\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [33](#)  
[query\(\)](#) (`dremio_client.model.data.Dataset` method), [14](#)  
[query\(\)](#) (in module `dremio_client.flight`), [13](#)  
[QueueMetadata](#) (class in `dremio_client.model.data`), [15](#)

## R

[reflection\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [34](#)  
[reflection\(\)](#) (in module `dremio_client.model.endpoints`), [23](#)  
[ReflectionMetadata](#) (class in `dremio_client.model.data`), [15](#)  
[Reflections](#) (`dremio_client.dremio_client.DremioClient` attribute), [30](#)  
[reflections\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [34](#)  
[reflections\(\)](#) (in module `dremio_client.model.endpoints`), [23](#)  
[ReflectionSummaryMetadata](#) (class in `dremio_client.model.data`), [15](#)  
[refresh\(\)](#) (`dremio_client.model.data.Catalog` method), [14](#)  
[refresh\(\)](#) (`dremio_client.model.data.PhysicalDataset` method), [15](#)  
[refresh\\_metadata\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [34](#)  
[refresh\\_metadata\(\)](#) (in module `dremio_client.util`), [29](#)  
[refresh\\_metadata\(\)](#) (in module `dremio_client.util.query`), [26](#)  
[refresh\\_pds\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [34](#)  
[refresh\\_pds\(\)](#) (in module `dremio_client.model.endpoints`), [23](#)  
[refresh\\_reflections\\_of\\_one\\_dataset\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [34](#)  
[refresh\\_reflections\\_of\\_one\\_dataset\(\)](#) (in module `dremio_client.util`), [29](#)  
[refresh\\_vds\\_reflection\\_by\\_path\(\)](#) (`dremio_client.dremio_simple_client.SimpleClient` method), [34](#)  
[refresh\\_vds\\_reflection\\_by\\_path\(\)](#) (in module `dremio_client.util`), [29](#)  
[remove\(\)](#) (`dremio_client.model.data.Catalog` method), [14](#)  
[Root](#) (class in `dremio_client.model.data`), [16](#)  
[RootMetaData](#) (class in `dremio_client.model.data`), [16](#)  
[RuleMetadata](#) (class in `dremio_client.model.data`), [16](#)  
[run\(\)](#) (in module `dremio_client.util`), [28](#)  
[run\(\)](#) (in module `dremio_client.util.query`), [26](#)  
[run\\_async\(\)](#) (in module `dremio_client.util`), [28](#)

`run_async()` (in module `dremio_client.util.query`), 27

## S

`set_catalog()` (`dremio_client.dremio_simple_client.SimpleClient` method), 34

`set_catalog()` (in module `dremio_client.model.endpoints`), 23

`set_collaboration_tag()` (`dremio_client.dremio_simple_client.SimpleClient` method), 35

`set_collaboration_tags()` (in module `dremio_client.model.endpoints`), 24

`set_collaboration_wiki()` (in module `dremio_client.model.endpoints`), 24

`set_personal_access_token()` (`dremio_client.dremio_simple_client.SimpleClient` method), 35

`set_personal_access_token()` (in module `dremio_client.model.endpoints`), 24

`simple()` (`dremio_client.dremio_client.DremioClient` method), 30

`SimpleClient` (class in `dremio_client.dremio_simple_client`), 31

`Source` (class in `dremio_client.model.data`), 16

`SourceMetadata` (class in `dremio_client.model.data`), 16

`SourceState` (class in `dremio_client.model.data`), 16

`Space` (class in `dremio_client.model.data`), 16

`SpaceMetadata` (class in `dremio_client.model.data`), 16

`sql()` (`dremio_client.dremio_simple_client.SimpleClient` method), 35

`sql()` (`dremio_client.model.data.Dataset` method), 14

`sql()` (in module `dremio_client`), 37

`sql()` (in module `dremio_client.model.endpoints`), 24

## T

`tags()` (`dremio_client.model.data.Catalog` method), 14

`TagsData` (class in `dremio_client.model.data`), 17

`to_dict()` (in module `dremio_client.conf`), 12

`to_json()` (`dremio_client.model.data.AccessControl` method), 13

`to_json()` (`dremio_client.model.data.AccessControlList` method), 13

`to_json()` (`dremio_client.model.data.Catalog` method), 14

`to_json()` (`dremio_client.model.data.DatasetMetadata` method), 14

`to_json()` (`dremio_client.model.data.FileMetadata` method), 14

`to_json()` (`dremio_client.model.data.FolderMetadata` method), 15

`to_json()` (`dremio_client.model.data.MetadataPolicy` method), 15

`to_json()` (`dremio_client.model.data.QueueMetadata` method), 15

`to_json()` (`dremio_client.model.data.ReflectionMetadata` method), 15

`to_json()` (`dremio_client.model.data.ReflectionSummaryMetadata` method), 16

`to_json()` (`dremio_client.model.data.RootMetadata` method), 16

`to_json()` (`dremio_client.model.data.RuleMetadata` method), 16

`to_json()` (`dremio_client.model.data.SourceMetadata` method), 16

`to_json()` (`dremio_client.model.data.SourceState` method), 16

`to_json()` (`dremio_client.model.data.SpaceMetadata` method), 17

`to_json()` (`dremio_client.model.data.TagsData` method), 17

`to_json()` (`dremio_client.model.data.VoteMetadata` method), 17

`to_json()` (`dremio_client.model.data.WikiData` method), 17

## U

`update_catalog()` (`dremio_client.dremio_simple_client.SimpleClient` method), 35

`update_catalog()` (in module `dremio_client.model.endpoints`), 25

`user()` (`dremio_client.dremio_client.DremioClient` method), 30

`user()` (`dremio_client.dremio_simple_client.SimpleClient` method), 35

`user()` (in module `dremio_client.model.endpoints`), 25

## V

`VirtualDataset` (class in `dremio_client.model.data`), 17

`VoteMetadata` (class in `dremio_client.model.data`), 17

`votes` (`dremio_client.dremio_client.DremioClient` attribute), 30

`votes()` (`dremio_client.dremio_simple_client.SimpleClient` method), 36

`votes()` (in module `dremio_client.model.endpoints`), 25

## W

`wiki()` (`dremio_client.model.data.Catalog` method), 14

`WikiData` (class in `dremio_client.model.data`), 17

`wlm_queues` (`dremio_client.dremio_client.DremioClient` attribute), 30

`wlm_queues()` (`dremio_client.dremio_simple_client.SimpleClient` method), 36

`wlm_queues()` (in module `dremio_client.model.endpoints`), 25

`wlm_rules` (*dremio\_client.dremio\_client.DremioClient*  
    *attribute*), [30](#)  
`wlm_rules()` (*dremio\_client.dremio\_simple\_client.SimpleClient*  
    *method*), [36](#)  
`wlm_rules()` (in *module*  
    *dremio\_client.model.endpoints*), [26](#)