This module is a thin wrapper around Enverus Drillinginfo’s Developer API (formerly known as Direct Access). It handles authentication and token management, pagination and network-related error handling/retries. It also provides a simple, convenient method to write results to CSV.

direct-access-py is built and tested on Python 3.6 but should work on Python 2.7 and up.

1.1 Installation

The easiest way to install direct-access-py is from the Python Package Index using pip:

```
$ pip install directaccess
```

To install it manually, simply download the repository from Github:

```
$ git clone https://github.com/wchatx/direct-access-py.git
$ cd directaccess/
$ python setup.py install
```

1.2 Notes

The directaccess module only supports the JSON format from the API. The query method returns a generator of API responses as dictionaries.

Version 2 of the API uses “soft deletes”. Records marked as deleted will have a populated DeletedDate field. If these records are not important for your workflow, you should always provide deleteddate='null' as a keyword argument to the V2 query method.

It is also important to note that your API credentials should be treated like any other password. Take care to not check them into public code repositories or expose them outside of your organization.
If you find a problem with this module, have a feature request or just need a little help getting started, please open an issue! If you’re having trouble with the Enverus Drillinginfo Developer API, you should contact support.

1.3 Quick Start

1.3.1 Direct Access Version 1

For version 1 of the API, create an instance of the DirectAccessV1 class and provide it your API key

```python
from directaccess import DirectAccessV1
d1 = DirectAccessV1(api_key='your-api-key')
```

**Warning**: Direct Access Version 1 will reach the end of its life in July, 2020. Please upgrade your application as Version 1 will be inaccessible after that date. A future version of this module will drop support for Version 1.

Provide the query method the dataset as the first argument and any query parameters as keyword arguments. See valid dataset names and query params in the Direct Access documentation. The query method returns a generator of API responses as dicts.

```python
for row in d1.query('legal-Leases', county_parish='Reeves', state_province='TX'):
    print(row)
```

1.3.2 Direct Access Version 2

For version 2 of the API, create an instance of the DirectAccessV2 class, providing it your API key, client id and client secret. The returned access token will be available as an attribute on the instance (`d2.access_token`) and the Authorization header is set automatically

```python
from directaccess import DirectAccessV2
d2 = DirectAccessV2(
    api_key='your-api-key',
    client_id='your-client-id',
    client_secret='your-client-secret',
)
```

Like with the V1 class, provide the query method the dataset and query params. All query parameters must match the valid parameters found in the Direct Access documentation and be passed as keyword arguments.

```python
for row in d2.query('well-origins', county='REEVES', pagesize=10000):
    print(row)
```

1.4 Version 2 Concepts

1.4.1 Filter Functions

Direct Access version 2 supports filter functions. These can be passed as strings on the keyword arguments.
Some common filters are greater than (gt()), less than (lt()), null, not null (not(null)) and between (btw()). See the Direct Access documentation for a list of all available filters.

```python
# Get well records updated after 2018-08-01 and without deleted dates
for row in d2.query('well-origins', updateddate='gt(2018-08-01)', deleteddate='null'):
    print(row)

# Get permit records with approved dates between 2018-03-01 and 2018-06-01
for row in d2.query('permits', approveddate='btw(2018-03-01,2018-06-01)'):
    print(row)
```

### 1.4.2 Fields keyword

You can use the `fields` keyword to limit the returned fields in your queries. This has the benefit of limiting the API responses to only those fields needed for your workflow and will significantly improve the speed of your queries.

```python
for row in d2.query('rigs', fields='DrillType,LeaseName,PermitDepth'):
    print(row)
```

### 1.4.3 Escaping

When making requests containing certain characters like commas, use a backslash to escape them.

```python
# Escaping the comma before LLC
for row in d2.query('producing-entities', curropername='PERCUSSION PETROLEUM OPERATING, LLC'):
    print(row)
```
2.1 DirectAccessV1

2.2 DirectAccessV2

class directaccess.DirectAccessV2(client_id, client_secret, api_key, retries=5, back-off_factor=1, links=None, access_token=None, **kwargs)

Client for Enverus Drillinginfo Developer API Version 2

count (dataset, **options)

    Get the count of records given a dataset and query options

    Parameters

    • dataset – a valid dataset name. See the Direct Access documentation for valid values
    • options – query parameters as keyword arguments

    Returns record count as int

ddl (dataset, database)

    Get DDL statement for dataset. Must provide exactly one of mssql or pg for database argument. mssql is Microsoft SQL Server, pg is PostgreSQL

    Parameters

    • dataset – a valid dataset name. See the Direct Access documentation for valid values
    • database – one of mssql or pg.

    Returns a DDL statement from the Direct Access service as str

docs (dataset)

    Get docs for dataset

    Parameters dataset – a valid dataset name. See the Direct Access documentation for valid values
Returns docs response for dataset as list[dict] or None if ?docs is not supported on the dataset

get_access_token()
Get an access token from /tokens endpoint. Automatically sets the Authorization header on the class instance’s session. Raises DAAuthException on error

Returns token response as dict

static in_(items)
Helper method for providing values to the API’s in() filter function.

The API currently supports GET requests to dataset endpoints. When providing a large list of values to the API’s in() filter function, it’s necessary to chunk up the values to avoid URLs larger than 2048 characters. The query method of this class handles the chunking transparently; this helper method simply stringifies the input items into the correct syntax.

def d2 = DirectAccessV2(client_id, client_secret, api_key)
    # Query well-origins
    well_origins_query = d2.query(
        dataset='well-origins',
        deleteddate='null',
        pagesize=100000
    )
    # Get all UIDs for well-origins
    uid_parent_ids = [x['UID'] for x in well_origins_query]
    # Provide the UIDs to wellbores endpoint
    wellbores_query = d2.query(
        dataset='wellbores',
        deleteddate='null',
        pagesize=100000,
        uidparent=d2.in_(uid_parent_ids)
    )

Parameters items (list) – list or generator of values to provide to in() filter function

Returns str to provide to DirectAccessV2 query method

query (dataset, **options)
Query Direct Access V2 dataset

Accepts a dataset name and a variable number of keyword arguments that correspond to the fields specified in the ‘Request Parameters’ section for each dataset in the Direct Access documentation.

This method only supports the JSON output provided by the API and yields dicts for each record.

Parameters

• dataset – a valid dataset name. See the Direct Access documentation for valid values

• options – query parameters as keyword arguments

Returns query response as generator

to_csv (query, path, log_progress=True, **kwargs)
Write query results to CSV. Optional keyword arguments are provided to the csv writer object, allowing control over delimiters, quoting, etc. The default is comma-separated with csv.QUOTE_MINIMAL

```python
d2 = DirectAccessV2(client_id, client_secret, api_key)
query = d2.query('rigs', deleteddate='null', pagesize=1500)
# Write tab-separated file
d2.to_csv(query, '/path/to/rigs.csv', delimiter='\t')
```
to_dataframe (dataset, converters=None, log_progress=True, **options)

Write query results to a pandas Dataframe with properly set dtypes and index columns.

This works by requesting the DDL for dataset and manipulating the text to build a list of dtypes, date columns and the index column(s). It then makes a query request for dataset to ensure we know the exact fields to expect, (ie, if fields was a provided query parameter and the result will have fewer fields than the DDL).

For endpoints with composite primary keys, a pandas MultiIndex is created.

This method is potentially fragile. The API’s docs feature is preferable but not yet available on all endpoints.

Query results are written to a temporary CSV file and then read into the dataframe. The CSV is removed afterwards.

pandas version 0.24.0 or higher is required for use of the Int64 dtype allowing integers with NaN values. It is not possible to coerce missing values for columns of dtype bool and so these are set to object dtype.

d2 = DirectAccessV2(client_id, client_secret, api_key)
# Create a Texas permits dataframe, removing commas from Survey names and replacing the state abbreviation with the complete name.
df = d2.to_dataframe(
    dataset='permits',
    deleteddate='null',
    pagesize=100000,
    stateprovince='TX',
    converters={
        'StateProvince': lambda x: 'TEXAS',
        'Survey': lambda x: x.replace(',', '')
    }
)
df.head(10)

Parameters

- dataset (str) – a valid dataset name. See the Direct Access documentation for valid values
- converters (dict) – Dict of functions for converting values in certain columns. Keys can either be integers or column labels.
- log_progress (bool) – whether to log progress. if True, log a message with current written count
- options – query parameters as keyword arguments

Returns pandas dataframe
CHAPTER 3

Indices and tables

- genindex
- modindex
- search
C
count() (directaccess.DirectAccessV2 method), 5

D
ddl() (directaccess.DirectAccessV2 method), 5
DirectAccessV2 (class in directaccess), 5
docs() (directaccess.DirectAccessV2 method), 5

G
get_access_token() (directaccess.DirectAccessV2 method), 6

I
in_() (directaccess.DirectAccessV2 static method), 6

Q
query() (directaccess.DirectAccessV2 method), 6

T
to_csv() (directaccess.DirectAccessV2 method), 6
to_dataframe() (directaccess.DirectAccessV2 method), 7