**DatastreamPy**

* Python wrapper for the Datastream Web Services API (DSWS)

To Connect to the Refinitiv Datastream database via Datastream Web Services, you need to have a Datastream subscription and a username/password to use this package.

Please note that this is an official beta package.

For support on this package, please contact Refinitiv team.

This package includes all functionalities required to get data from Datastream, in the form of Python dataframes.

The package by default refers to <https://product.datstream.com>. In case you need to change this to other url or Delivery Direct ip, the config file can be used and the path of the config file needs to be provided in the code (Refer below on configuring url path). Please refer Appendix 2 for config details.

**Software Requirements:**

Install Python 3 or higher versions.

**Python Packages to be installed:**

Pandas, requests, Urllib3, datetime, pytz

**Usage**

* **DatastreamPy – Package installation:**

pip install DatastreamPy

* **Import the package in the code and get data**. (Example given below)

import DatastreamPy as dsws

* **Configuring URL Path** (If different from <https://product.datastream.com>)

ds = dsws.Datastream(username='XXXXXXX', password='XXXXXXX', config='<your config path>')

* **Authenticate and use get\_data to request data**

ds = dsws.Datastream(username='XXXXXXX', password='XXXXXXX')

df = ds.get\_data(tickers='VOD',fields=['P'],start ='2017-01-01',end = '-5D')

print(df)

In case there are any *network related issues* while running the above code, please refer *Appendix 1*.

Output:

#Instrument Datatype Value Dates

0 VOD VO 36773.80 2017-01-01

1 VOD P 199.85 2017-01-01

**Time Series Requests**

df = ds.get\_data(tickers='VOD', fields=['P','MV','VO'],start='-10D',end='-0D', freq='D')

Output:

Instrument VOD

Field P MV VO

Dates

2019-12-04 146.08 39106.88 55532.5

2019-12-05 144.26 38619.65 64336.8

2019-12-06 144.38 38651.77 44165.4

2019-12-09 143.86 38512.55 60042.1

2019-12-10 142.74 38212.71 56569.8

2019-12-11 143.58 38437.59 61400.9

2019-12-12 144.34 38641.07 58731.2

2019-12-13 146.24 39149.71 72826.8

2019-12-16 148.32 39706.54 49237.2

2019-12-17 149.28 39963.54 56020.2

2019-12-18 150.50 40290.15 61011.9

**Using get\_bundle\_data**

ds = dsws.Datastream("xxxxxxx", "xxxxxxxxx")

reqs =[]

reqs.append(ds.post\_user\_request(tickers='VOD',fields=['VO','P'],start='2017-01-01', kind = 0))

reqs.append(ds.post\_user\_request(tickers='U:BAC',fields=['P'],start='1975-01-01', end='0D', freq = "Y"))

df = ds.get\_bundle\_data(bundleRequest=reqs)

print(df)

Output:

Instrument Datatype Value Dates

0 VOD VO 36773.80 2017-01-01

1 VOD P 199.85 2017-01-01,

Instrument Dates U:BAC

Field P

0 1975-01-01 0.9375

1 1976-01-01 1.2188

2 1977-01-01 1.5313

3 1978-01-01 1.4219

.....

**Retrieving data for a List**

import DatastreamPy as dsws

ds = dsws.Datastream(username="xxxxx", password="xxxxx")

df = ds.get\_data(tickers="LS&PCOMP|L",fields =["NAME"], kind=0)

print(df)

*(Note: 1. we should specify |L in tickers, for List.*

*2. the ‘kind’ param is set to 0 for Static requests. By default ‘kind’ is set to 1 i.e TimeSeries request)*

Output:

Instrument Datatype Value Dates

0 891399 NAME AMAZON.COM 2019-01-21

1 916328 NAME ABBOTT LABORATORIES 2019-01-21

2 545101 NAME AES 2019-01-21

3 777953 NAME ABIOMED 2019-01-21

......

**Retrieving data for Expressions**

import DatastreamPy as dsws

ds = dsws.Datastream(username="xxxxx", password="xxxxx")

df = ds.get\_data(tickers='PCH#(VOD(P),3M)|E', start="20181101",end="-1M", freq="M")

print(df)

*(Note: we should specify |E in tickers, for Expressions.)*

Output:

Instrument Dates PCH#(VOD(P), 3M)

Field

0 2018-11-01 -17.82

1 2018-12-01 0.91

**Symbol substitution**

df =ds.get\_data(tickers='VOD, U:JPM',fields=['PCH#(X(P),-3M)'], freq="M")

Output:

Instrument Dates VOD U:JPM

Field PCH#(X(P),-3M) PCH#(X(P),-3M)

0 2018-02-07 -3.07 14.2987

1 2018-03-07 -10.25 9.6635

2 2018-04-07 -13.85 0.6923

3 2018-05-07 0.24 -3.1009

4 2018-06-07 -8.30 -3.4254

5 2018-07-07 -6.37 -4.6109

......

**Retrieving data for NDOR**

df = ds.get\_data(tickers='USGDPâ€¦D',fields=['DS.NDOR1'])

Output:

Instrument Datatype Value

0 USGDP...D DS.NDOR1\_DATE 2019-02-11

1 USGDP...D DS.NDOR1\_DATE\_LATEST 2019-02-19

2 USGDP...D DS.NDOR1\_TIME\_GMT NA

3 USGDP...D DS.NDOR1\_DATE\_FLAG Estimated

4 USGDP...D DS.NDOR1\_REF\_PERIOD 2018-11-15

5 USGDP...D DS.NDOR1\_TYPE NewValue

**Retrieving data for Point In Time**

df = ds.get\_data(tickers='CNCONPRCF(DREL1)', fields=['(X)'], start='-2Y', end='0D', freq='M')

Output:

Instrument Dates CNCONPRCF(DREL1)

Field (X)

0 2017-02-15 2017-03-24

1 2017-03-15 2017-04-21

2 2017-04-15 2017-05-19

3 2017-05-15 2017-06-23

4 2017-06-15 2017-07-21

5 2017-07-15 2017-08-18

**Usage Stats**

df = ds.get\_data(tickers='STATS', fields=['DS.USERSTATS'], kind=0)

Output:

Instrument Datatype Value Dates

0 STATS User xxxxxxx 2019-02-08

1 STATS Hits 147 2019-02-08

2 STATS Requests 113 2019-02-08

3 STATS Datatypes 660 2019-02-08

4 STATS Datapoints 23213 2019-02-08

5 STATS Start Date 2019-02-01 2019-02-08

6 STATS End Date 2019-02-28 2019-02-08

**Returning the Instrument and Datatype names**

Using get\_data function:

df = ds.get\_data(tickers='VOD,BARC', fields=['PH','PL'], start= '-2D', retName=True)

(Set the retName param to True)

Output:

{'VOD': 'VODAFONE GROUP', 'BARC': 'BARCLAYS', 'PH': 'PRICE HIGH', 'PL': 'PRICE LOW'}

Instrument VOD BARC VOD BARC

Field PH PH PL PL

Dates

2020-08-31 NaN NaN NaN NaN

2020-09-01 111.56 111.16 106.35 105.86

2020-09-02 109.62 107.42 107.02 103.18

Using get\_bundle\_data function:

bReqs =[]

bReqs.append(ds.post\_user\_request(tickers='VOD,BARC', fields=['PH','PL'], start= '-2D', retName=True))

bReqs.append(ds.post\_user\_request(tickers='R:ABGJ', fields=['WC05350'], kind=1, start = '2015-12-31', end = '2016-12-31', freq='Y', retName=True))

df = ds.get\_bundle\_data(bReqs,retName=True)

(Note that retName is set to True in both post\_user\_request and get\_bundle\_data functions)

Output:

{'VOD': 'VODAFONE GROUP', 'BARC': 'BARCLAYS', 'PH': 'PRICE HIGH', 'PL': 'PRICE LOW'}

{'R:ABGJ': 'ABSA GROUP LTD', 'WC05350': 'FISCAL PERIOD END DATE'}

[Instrument VOD BARC VOD BARC

Field PH PH PL PL

Dates

2020-08-31 NaN NaN NaN NaN

2020-09-01 111.56 111.16 106.35 105.86

2020-09-02 109.62 107.42 107.02 103.18,

Instrument R:ABGJ

Field WC05350

Dates

2015-12-31 2015-12-31

2016-12-30 2016-12-31

**Acknowledgements**

Thanks to Joris (hoenie-ams) for his work on https://github.com/hoenie-ams/PyDSWS

**Appendix - 1**

**Network Issues**

As we have implemented ‘HTTPS’ connection, some of the clients may get issues while using the package.

Some of the common issues and possible resolutions are listed below.

Issue type 1:

ConnectionError(MaxRetryError("HTTPSConnectionPool(host='product.datastream.com', port=443): Max retries exceeded with url: /DSWSClient/V1/DSService.svc/rest/GetToken (Caused by NewConnectionError('<requests.packages.urllib3.connection.VerifiedHTTPSConnection object at 0x00000000159514A8>: Failed to establish a new connection: [Errno 11004] getaddrinfo failed',))",),), <traceback object at 0x0000000015A60DC8>)

Please contact the network Admin team to provide the Proxy details.

Provide the proxy details in the code as shown below:

import DatastreamPy as dsws

ds = dsws.Datastream(username='XXXXXXX', password='XXXXXXX', proxy='https://user:password@proxyip:port')

Issue type 2:

MaxRetryError: HTTPSConnectionPool(host='product.datastream.com', port=443): Max retries exceeded with url: /DSWSClient/V1/DSService.svc/rest/Token?username=xxxxx2&password=xxxxxx (Caused by SSLError(SSLError("bad handshake: Error([('SSL routines', 'tls\_process\_server\_certificate', 'certificate verify failed')])")))

Please find the SSL certificate path on your local machine and provide the same in the code as shown below:

import DatastreamPy as dsws

ds = dsws.Datastream(username='XXXXXXX', password='XXXXXXX', sslCer='path\certfile')

**Appendix – 2**

**Configuration of url:**

The config file should contain the url that you need to connect to. A config.ini file needs to be created and the contents of the file should be as shown below:

[url]

path=https://example.com

The config.ini file is provided on Git hub for your reference.

We should provide the path of this config file in the code as shown below

ds = dsws.Datastream(username='xxxxx', password='xxxxx', config='<path>/config.ini')