The Financial and Risk business of Thomson Reuters is now Refinitiv.

REFINITIV™

Datastream Web Service Getting started with Python

Introduction

Datastream is the world's leading time series database, enabling strategists, economists and research communities' access to the most comprehensive financial information available. With histories back to the 1950's, you can explore relationships between data series; perform correlation analysis, test investment and trading ideas and research countries, regions and industries.

Datastream content is available via the Python 3.6 or above using the Datastream Web Service (DSWS) tool.

This document provides examples on how to: access DSWS via Python and run simple requests. It also gives basic information on usage limits.

Requirements

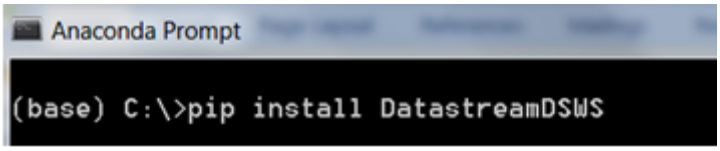
- A Datastream Child ID starting with Z, with access to DSWS API
- Python 3.6 or above

Getting started

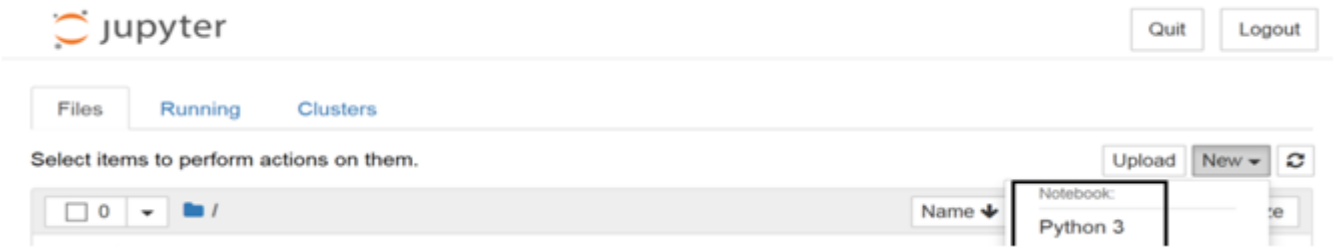
In order to use DSWS via Python you would need to install Python beta package for Datastream Web Service (DSWS) API available here: https://github.com/DatastreamDSWS/Datastream

The package is using pandas, requests, datetime and pytz library and it requires you to have DSWS service with your Datastream account (with valid Datastream child Id and password). If you don't have access to DSWS please contact your account representative.

To demonstrate how to get started using DSWS service via Python we will be using Anaconda platform. Once installed go to Start menu, search for and open Anaconda Prompt. In the new window type 'pip install DatastreamDSWS' as shown on below picture.



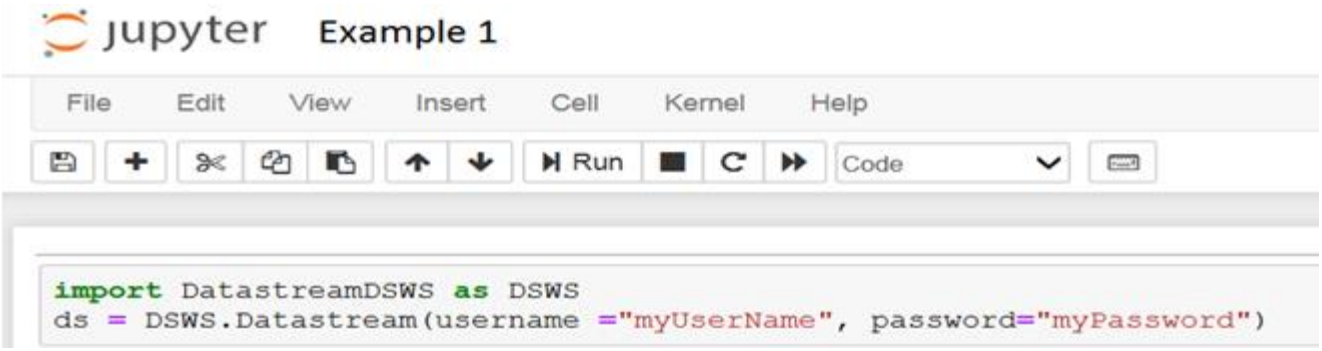
For examples of requests we will use Jupyter Notebook, a web- based, interactive computing notebook environment. Once the pip is installed go to Anaconda Navigator in Start menu and launch Jupyter Notebook. A new window will open in your Internet Browser:



To create a notebook click on the "New" and from drop down list select "Python", as presented in picture above. A new window will open with your notebook name, a menu bar, a toolbar and an empty code cell. To change the name double click on "Untitled" and type in preferable title (as seen below).

In order to start working on Datastream data you need to first import DatastreamDSWS and use your DSWS ID and password, as seen below. You can add a cell from the toolbar by:

1. Clicking on plus sign (+)
2. Clicking on "Run" after typing in the request



Running requests within Python

Once you have authenticated your account you can start requesting the data. You can add one cell at a time and run it or you can add below examples in separate cells and run them all by clicking on "Cell" in menu bar and selecting "Run all" from drop down list.

A typical request would be for a **snapshot request**. To create it you need to define **tickers** with instruments of your choice and **fields** with static data types and finish with parameter **kind=0**.

Below snapshot examples will show how the requests can be created following with output in Jupyter. All the examples are available in the ipynb file DatastreamDSWS_Basic Equity series available for download here.

Simple request for one instrument (VOD – for Vodafone Group) and one data type P (which represents the official closing price) would look like this:

ds.get_data(tickers='VOD', fields='P', kind=0)

	Instrument	Datatype	Value
0	VOD	P	127.46

The same request, with more data types (MV – Market Value, DY – Dividend Yield) can be done with RIC instead as long as it is enclosed in <>:

ds.get_data(tickers='<VOD.L>', fields=('P','MV','DY'), kind=0)

	Instrument	Datatype	Value	Dates
0	<VOD.L>	P	127.46	2019-06-17
1	<VOD.L>	MV	34061.74	2019-06-17
2	<VOD.L>	DY	6.14	2019-06-17

While creating a request for multiple instruments the data type parameters should be set in square brackets:

ds.get_data(tickers='@AAPL, @FB, @GOOGL, @MFST, @NXPI, U:JPM, U:XOM', fields= ['NAME'], kind=0)

	Instrument	Datatype	Value	Dates
0	@AAPL	NAME	APPLE	2019-06-17
1	@FB	NAME	FACEBOOK CLASS A	2019-06-17
2	@GOOGL	NAME	ALPHABET A	2019-06-17
3	@MFST	NAME	MEDIFIRST SOLUTIONS	2019-06-17
4	@NXPI	NAME	NXP SEMICONDUCTORS	2019-06-17
5	U:JPM	NAME	JP MORGAN CHASE & CO.	2019-06-17
6	U:XOM	NAME	EXXON MOBIL	2019-06-17

You can also create **static request –** a one off request for instruments and data types at one point in time. In this case you need to define **tickers**, **fields**, **date** and **kind=0**, as on below example.

ds.get_data (tickers='@AAPL, @FB, @GOOGL, @MSFT, U:JPM', fields=['P'], start='2018-01-01', kind=0)

	Instrument	Datatype	Value	Dates
0	@AAPL	P	169.23	2018-01-01
1	@FB	P	176.46	2018-01-01
2	@GOOGL	P	1053.40	2018-01-01
3	@MSFT	P	85.54	2018-01-01
4	U:JPM	P	106.94	2018-01-01

For this request output will show official closing price for five instruments on 1st of January 2018.

It is also possible to perform a time series request. The difference between static and time series requests is that in the latter you will use **start and end date** to define the period for which you need the selected data. Date can be relative (e.g. -10D, -2Y, 3M) or absolute (e.g. 2018-11-09) date format. **Frequency** in the request can be specified in days (D), weeks (W), months (M), quarters (Q) or years(Y).

Note: Without specifying the end date, the previous days' value will be returned

For example, to get daily data from 1st to 10th of January 2018, for ten instruments, with eight data types use:

ds.get_data (tickers='@AAPL, @FB, @GOOGL, @MSFT, @NXPI, U:JPM, U:XOM, U:BAC, U:BABA, U:V',

fields=['P', 'MV', 'PO', 'PH', 'PL', 'VO', 'DY', 'PE'], start='2018-01-01', end='2018-01-10', freq='D')

You can also request Worldscope, IBES or ESG data by using appropriate data types. Here is a request that will give you five years of Worldscope data in annual frequency, for previously mentioned indices:

ds.get_data(tickers='@AAPL, @FB, @GOOGL, @MSFT, @NXPI, U:JPM, U:XOM, U:BAC, U:BABA, U:V',

fields=['WC08311','WC18191', 'WC18100', 'WC08106', 'WC08376'], start='-5Y', freq='Y')

For six months of daily frequency IBES company level EPS1MN, SAL1MN data for given instruments use the following:

ds.get_data(tickers='@AAPL, @FB, @GOOGL, @MSFT, @NXPI, U:JPM, U:XOM, U:BAC, U:BABA, U:V',

fields=['EPS1MN', 'SAL1MN'], start='-1Y',end='-6M', freq='D')

Finally, requesting five years of yearly frequency ESG data for given instruments:

ds.get_data(tickers='@AAPL, @FB, @GOOGL, @MSFT, @NXPI, U:JPM, U:XOM, U:BAC, U:BABA, U:V', fields=['SOCOO01V','ENPIDP048','SOHRDP012','SOEQ','SOTDDP018','SODODP0012','ENRRDP033','ENERDP052','CGVSDP030'], start='-5Y', freq='Y')

Running other requests

In this beta package we have made possible creation of bundle requests, Next date or Release and Point in Time data requests, usage stats review for Desktop users, and more. Examples in the ipynb file of these requests are available for download here and this document presents more details on these types of requests, what they mean and how to create them.

Bundle request

Bundle request is used to retrieve multiple datasets in one request. Details on GetDataBundle method are available in Soap tutorial here. We have enabled the creation of such request in Python by using the following code:

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"))

ds.get_bundle_data(bundleRequest=reqs)

Note: There is a limit to the size of a bundle, detailed in section “size of requests” at bottom of page

Next date of release (NDoR)

The dates of the next releases (NDoR) are important both in the initial selection of a series, and once selected to be informed of when the series will next be updated. This data can be downloaded using a set of datatypes. These have the form DS.NDOR1 to DS.NDOR12 and will display the following fields (once an update has taken place the NDoR’s will rollover, so what was #2 today will be #1 tomorrow for example).

	Field	Description
1	DS.NDOR1_DATE	Date of release or start of range where the exact date is not available.
2	DS.NDOR1_DATE_LATEST	End of range when a range is given.
3	DS.NDOR1_TIME_GMT	Expected time of release. This is provided for 'official' times only – see next field.
4	DS.NDOR1_DATE_FLAG	Indicates whether dates are 'official' ones from the source; or estimated by Thomson Reuters where 'officials' not available.
6	DS.NDOR1_REF_PERIOD	Update period. Provides the latest observation or time period that is being issued on the given day of release.
5	DS.NDOR1_TYPE	Indicates whether the release is for a new reference period (NewValue) or an update to a period for which there has already been a release (ValueUpdate).

NDoRs are available for over 300,000 nationally sourced series and Markit PMIs. In Python you can use following:

ds.get_data(tickers='CNCONPRCF',fields=['DS.NDOR1'])

	Instrument	Datatype	Value
0	CNCONPRCF	DS.NDOR1_DATE	2019-06-19
1	CNCONPRCF	DS.NDOR1_DATE_LATEST	2019-06-19
2	CNCONPRCF	DS.NDOR1_TIME_GMT	12:30
3	CNCONPRCF	DS.NDOR1_DATE_FLAG	Official
4	CNCONPRCF	DS.NDOR1_REF_PERIOD	2019-05-15
5	CNCONPRCF	DS.NDOR1_TYPE	NewValue

Point in Time

Economics Point in Time database enables users to view economic data as it was originally reported upon release, then view how it changed over time. The database has over 10 years of history & in excess of 2000 major indicators from around the globe. This database has been enhanced so users of DFO & datafeeds can access a time series of the first three release dates.

These datatypes would be used alongside existing Datatypes REL1 REL2 & REL3 that display the first, second

& third release values.

Datatype	Description
DREL1	Date of first release
DREL2	Date of second release
DREL3	Date of third release

Below is example showing Canada Consumer Price with the First release date (DREL1):

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

Instrument	CNCONPRCF(DREL1)
Field	(X)
Dates	
2017-06-15	2017-07-21
2017-07-15	2017-08-18
2017-08-15	2017-09-22
2017-09-15	2017-10-20
2017-10-15	2017-11-17
2017-11-15	2017-12-21

List and functions/expressions request

Datastream also supports Constituent Lists of instruments, e.g. LFTSE100, LS&PCOMP, LDAXINDX, LSTOKYOSE, etc. List instruments are only supported in Snapshot mode, and only one list is permitted per request. The datatypes supplied with the list request are applied to all the constituents of the list. These lists can be searched for using DFO, EIKON, etc.

List request must carry **|L** after the instrument as in below example where the output will show us mnemonics for all constituents of the list LINSURIT (Insurance IT). Please set **|L to** simplify post-processing of the response.

ds.get_data(tickers="LINSURIT|L",fields =["MNEM"], kind=0)

	Instrument	Datatype	Value	Dates
0	923375	MNEM	I:G	2019-06-17
1	284294	MNEM	I:CASS	2019-06-17
2	51269L	MNEM	I:PST	2019-06-17
3	505136	MNEM	I:UNI	2019-06-17
4	929455	MNEM	I:US	2019-06-17

Similar to list all the expressions too must carry **|E** when being requested. When this is set the server performs different processing of the instrument field.

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

Instrument	PCH#(VOD(P),3M)
Field	
Dates	
2018-11-01	-17.82
2018-12-01	0.91
2019-01-01	-5.62
2019-02-01	-8.36
2019-03-01	-20.12
2019-04-01	-7.21
2019-05-01	1.66

Symbol substitution

You can simplify the use of expressions using the “Symbol substitution” feature in fields part of the request, where each requested instrument is substituted for X in any expression (e.g. in expression PCH#(VOD,-1M), VOD will be substituted by X).

ds.get_data(tickers='VOD', fields=['PCH#(X,-1M)'], start='-1D',kind=0)

	Instrument	Datatype	Value
0	VOD	PCH#(X,-1M)	0.93

Transposing

If you would like to change the layout of data, you can use the example below to transpose columns to rows. By using below "data1" – instrument, fields and date will be rearranged to rows. Similarly, with second example (data2), instruments, fields and date will be visible in row instead of column.

data1=ds.get_data(tickers='@AAPL', fields=['P'], start='-2D', end='-0D', freq='D')data1.transpose()

	Dates	2019-06-13	2019-06-14	2019-06-17
Instrument	**Field**			
@AAPL	**P**	194.15	192.74	193.89

data2 = ds.get_data(tickers='VOD', fields=['P','MV',], start='2017-01-01', kind=0)data2.transpose()

	0	1
Instrument	VOD	VOD
Datatype	P	MV
Value	199.85	53194.4
Dates	2017-01-01	2017-01-01

Usage statistics

If you are accessing content via the Datastream Web Service (DSWS) API for Desktop you might want to know your monthly usage in terms of data points used per month. You will do this by using 'STATS' as instruments and 'DS.USERSTATS' as data type. Only snapshot requests are supported and by default the current month's usage stats are returned. Previous months' data can be returned by simply adding a valid start date in request of any previous month.

ds.get_data(tickers='STATS', fields=['DS.USERSTATS'], kind=0)

	Instrument	Datatype	Value	Dates
0	STATS	User	Z	2019-06-21
1	STATS	Hits	226	2019-06-21
2	STATS	Requests	196	2019-06-21
3	STATS	Datatypes	1553	2019-06-21
4	STATS	Datapoints	103640	2019-06-21
5	STATS	Start Date	2019-06-01	2019-06-21
6	STATS	End Date	2019-06-30	2019-06-21

Size of requests

Users are advised of the maximum size limits of DSWS requests via Python:

- Maximum instruments per request 50
- Maximum datatypes per request 50
- Maximum items (instrument x datatypes) per request not to exceed 100

The above limits permit the following example permutations in **single request**:

- 50 instruments x 2 datatypes
- 2 instruments x 50 datatypes
- 1 constituent list x 50 datatypes (you can never request more than one constituent list)
- 10 instruments x 10 datatypes

When using **Bundle request**, where a collection of single request can be supplied, there are additional limits imposed on the number of items that can be requested across the bundle:

- The maximum number of Requests (reqs.append) per bundled request: 20
- The maximum number of items (instruments x datatypes) across all Requests: 500

The above limits permit the following example permutations in any one Bundle request:

- Up to 5 Requests each requesting 100 items
- 10 Requests each requesting 50 items
- 20 Requests each requesting 25 items