Connect - Python

User Guide

Prepared by Andrew McSween

Version 1.4

10/13/21

Table of Contents

1 About This Document

This document is the User Guide for the Connect - Python Capability. The document includes details on how to get started and the available requests. For additional details, please review the Jupyter Notebook samples on the ICE Connect Python website.

2 Getting Started

2.1 Pre-Requisites

- Python - Version 3.4 or later
- ICE XL - Version 4.9.6 or later
- Python Library: pywin32 - Allows for the communication between Python and .Net Libraries
- Python Library: icepython - Connect Library for data requests

2.2 Installation

- Ensure Python 3.4 (or later) is properly installed on the client machine
- Ensure the latest ICE XL version is installed, v. 4.9.6 or later
    - Installer found here
- Install pywin32 using pip command
    - Open command prompt
    - Type: python -m pip --user pywin32
    - Note: "--user" is not needed if you're an admin with an admin installation of python
    - Confirm that the library has installed
- Install icepython library using pip command
    - From command prompt go to your ICE XL directory
    - Type: cd "%localappdata%\ICE Data Services\ICE XL\bin"
    - Now install the icepython library
    - Type: python -m pip install theice.com_ICEPython-0.0.3-py3-none-any.whl
    - Confirm that the library has installed

2.3 Importing the Library

- Within Python, import the icepython library
- Type: "import icepython as ice" from within Python
- Example Code:

```
In [2]: #Sample wrapper package import
        import icepython as ice # ICE Python wrapper
        import pandas as pd
        import matplotlib as mp
```

2.4 Additional Notes

- ICE XL must be installed and authenticated. The Connect - Python functionality uses ICE XL for authentication
- Any data limits or entitlements are shared between ICE XL and Connect - Python
- All Python requests must happen locally to the machine running ICE XL

- The Python library is installed and updated through the ICE XL installer

3 Snapshot Quotes

3.1 Snapshot Quotes Details

- Snapshot quotes returns the latest value for streaming content
- Number of symbols is limited to 500 per request
- Number of requests are limited to 10 per second

3.2 Arguments

- Request takes a set of symbols, a set of fields, and an additional argument to keep the data streaming to the publisher
- All arguments are required
    - Symbols can be set as an array

```
In [6]: symbols = ['IBM', 'GOOG', 'ICE', 'AMD']
        print(symbols)

        ['IBM', 'GOOG', 'ICE', 'AMD']
```

    - Fields can be set as an array

```
In [7]: fields = ['bid', 'ask', 'last', 'volume']
        print(fields)

        ['bid', 'ask', 'last', 'volume']
```

    - Setting the subscription to 'True' means subsequent requests are faster - the Publisher will continue to receive updates, and the latest updates are retrieved locally.
    - Setting the subscription to 'False' means the publisher will unsubscribe the subscription after the initial request and will not impact your subscription limit. Future requests will require re-subscribing.

3.3 Example

```
In [2]: #Sample simple request for quotes
        #Arguments for get_quotes are Symbols, fields, and the subscription state for those requests.
        #The subscription state of True will keep an active connection to our servers for update; any subsequent request is local.
        data = ice.get_quotes(['ice','brn 1!-ice'],'last',True)
        print(data)
```

4 Snapshot Time Series

4.1 Snapshot Time Series Details

- Time series request returns a tuple where the date is in the first column, followed by the symbols and fields requested
- If multiple fields are requested, the output is sorted so all fields for one symbol are placed together
- Output has column headers with the symbols and fields, e.g. "IBM.Last"

4.2 Arguments

- All Arguments are required

- Symbols can be set as an array
- Fields can be set as an array
- Granularity is one value
  - D = Daily
  - W = Weekly
  - M = Monthly
  - I1 = 1 min
  - I30 = 30 min
  - I60 = 1 hour
- Start_date can take a date, or date & time
- End_date can take a date or date & time

4.3 Example

```
In [ ]: #Sample Time Series Request
        my_data = ice.get_timeseries(symbols, #Set of symbols in the request
                                     fields, #Set of feilds in the request
                                     granularity='D', #defines the timeseries granularity
                                     start_date='2020-01-01', #Start date
                                     end_date='2020-12-31') #End date
```

5 Snapshot Time & Sales

5.1 Snapshot Time & Sales Details

- Snapshot Time & Sales requests are limited to 1 per second
- Snapshot Time & Sales requests are limited to 10 symbols per request
- Data is returned in a tuple with date and time in the left most column, with symbols and fields listed in the headers of the columns to the right
- If multiple fields are requested, the output is sorted so all fields for one symbol are placed together

5.2 Arguments

- All Arguments are required
  - Symbols can be set as an array
  - Fields can be set as an array
  - Start_date can take a date, or date & time
  - End_date can take a date or date & time

5.3 Example

```
In [ ]: #Sample Time & Sales Request
        my_data = ice.get_timesales(symbols, #Set of symbols in the request
                                    fields, #Set of feilds in the request
                                    start_date='2020-12-30T15:30:00', #Start date and time
                                    end_date='2020-12-31T15:30:00') #End date and time
```

6 Ancillary Requests

6.1 Autolisting function

- Autolisting allows for the request of the set of tenors of a given product
- Results should be returned in an array, which can be used in future requests

- Monthly futures can be requested with one "*" and the product "Root"
  - "*BRN-ICE" will return all futures
- Spreads can be requested with two "*" and the product "Root"
  - "**BRN-ICE" will return front-back calendar spreads
  - "**BRN:BRN-ICE" will return all calendar spreads

6.2 Example:

```
In [ ]: #Sample Autolist
        symbols = ice.get_autolist('*BRN-ICE')
```

6.3 Hibernation functions

- The ICE XL Publisher will hibernate when not in use.
- Connection will occur as soon as a request is made and take the ICE XL Publisher out of hibernation mode
  - Connecting prior to the first request will speed up initial requests to the server
- Any active subscriptions will prevent the ICE XL Publisher from entering hibernation mode.
- There are two functions relating to Hibernation
  - get_hibernation() returns the current state of the publisher
  - set_hibernation() sets the ICE Publisher to a desired state

6.4 Arguments

- Set_hibernation(VAL) can take **True** or **False** values

6.5 Examples:

```
In [15]: ice.get_hibernation()
Out[15]: 'False'
```

```
In [25]: ice.set_hibernation(True)
```