FXCM REST API

Quick start for Python

1. Overview:

FXCM offers a web-based REST API which can be used to establish secure connectivity with FXCM’s trading systems for the purpose of receiving market data and trading.

This document provides the easiest way to use the REST API for the basic steps of trading.

To get started with the API and the example code, a demo account with FXCM is sufficient. You can open such an account under <https://www.fxcm.com/uk/forex-trading-demo/>. Also a PyCharm(IDE for python) and Anaconda version 3 should be installed. Python 3.6 is the version that is used in the sample.

1. Prerequisite:

* A token should be revoked from the Trading Station on <https://tradingstation.fxcm.com/>
* The Socked.IO library should be installed using Python: <https://pypi.python.org/pypi/socketIO-client>
* The Python Example should be cloned or downloaded and loaded into the PyCharm from <https://github.com/fxcm/fxcm-api-rest-python3-example>

1. Get connected.

In order to connect you should perform the following steps:

* Add the following imports to the project as it is shown in the “fxcm\_rest\_api\_token.py” file from the <https://github.com/fxcm/fxcm-api-rest-python3-example>:

**from** collections **import** namedtuple  
**import** requests  
**from** socketIO\_client **import** SocketIO  
**import** logging  
**import** json  
**import** uuid  
**import** threading  
**from** dateutil.parser **import** parse  
**from** datetime **import** datetime  
**import** time  
**import** types

* Link your account through the account token that you have revoked from <https://tradingstation.fxcm.com/>

trader = fxcm\_rest\_api.Trader(**’YOUR\_TOKEN\_HERE’**, **'demo'**) *# demo for demo*trader.login()

For the example purposes we will enter the following rows and the previous imports will be used through the “fxcm\_rest\_api\_token”:

**import** fxcm\_rest\_api\_token  
  
trading\_symbol = **'EUR/USD'**trader = fxcm\_rest\_api\_token.Trader(**YOURTOKEN** , **'demo'**) *# demo for demo*trader.login()  
trade\_amount = 10  
trade\_start\_buy\_price = SHOULD BE MUCH LOWER THAN THE CURRENT PRICE  
trade\_start\_sell\_price = SHOULD BE MUCH HIGHER THAN THE CURRENT PRICE

1. Get live feed (EUR/USD as example)

live\_feed = trader.subscribe\_symbol(**'EUR/USD'**)  
print(live\_feed)

As an only parameter you need to enter the symbol you are trading

1. Retrieve historical price

historical\_prices = trader.candles(**"EUR/USD"**, **"m1"**, 15, dt\_fmt=**"%Y/%m/%d %H:%M:%S"**)[**'candles'**]   
**for** candle **in** historical\_prices:  
 print(candle)

Parameters:

* “EUR/USD” is the symbol you are trading
* “m1” is the period fitted in a single candle. The example is 1 minute you can choose between (minutes) m1, m5, m15, m30, (hours) H1, H2, H3, H4, H6, H8, (day )D1, (week) W1,(month) M1
* 15 is the amount of candles that you will request to se
* dt\_fmt=**"%Y/%m/%d %H:%M:%S"** is actually not a parameter, but changing the date format to an adequate one

1. Retrieve acct/ open position / closed table.

All tree requests Retrieve account, open positions information and closed positions information are simple and intuitive lines of code thanks to the fxcm\_rest\_api\_token used in the example:

6.1.

theAccountID = trader.account\_id

6.2.

open\_positions = trader.open\_positions  
**for** position **in** open\_positions:  
 print(position)

6.3.

closed\_positions = trader.closed\_positions  
**for** position **in** closed\_positions:

print(position)

1. Place market/ entry order. (just simple one)
   1. Market order:

trader.open\_trade(theAccountID, trading\_symbol, **True**, 10)

Parameters:

* theAccountID you can extract as shown in step 6.1
* trading\_symbol is the symbol you will start a trade in
* True actually stand there as an answer of the question if this is going to be a BUY trade or a SELL trade
* 10 is the amount of lots you are going to trade
  1. Entry order:

trader.create\_entry\_order(theAccountID , trading\_symbol , **True** , trade\_start\_buy\_price, 10, **True**,  
 **'Entry'**, **'GTC'** )

Parameters:

* theAccountID you can extract as shown in step 6.1
* trading\_symbol is the symbol you will start a trade in
* True actually stand there as an answer of the question if this is going to be a BUY trade or a SELL trade. So True equals BUY and False equals SELL
* **trade\_start\_buy\_price** or **trade\_start\_sell\_price** (as seen in 3. Get started) should actually be the price that you want to create your Entry order with
* 10 is the amount of lots you are going to trade
* The second True is for choosing if the amount of the trade is in PIPs
* ‘Entry’ is for the type of the trade
* ‘GTC’ stands for “Good till close” is a parameter for when to start the entry

1. Modify price of entry order.

If you want to change the parameters of an entry order you have to use the ID and this is how you extract them:

open\_orders = []  
orders = trader.orders  
**for** order **in** orders:  
 open\_orders.append(order[**'orderId'**])

Entry order change:

trader.change\_order(open\_orders[1], trade\_start\_buy\_price, 0, 15)

Parameters:

* First parameter is the order ID
* Second parameter is the actual price that you want to change your entry price to
* Third parameter is the range of the price
* 15 stands for the amount of lots you want to use in the changed entry