FXCM REST API

Quick Start Guide

JavaScript and node.js

1.Introduction

FXCM provides a RESTful API (henceforth the “API”) to interact with its trading platform. Among others, it allows the retrieval of historical data as well as of streaming data. In addition, it allows to place different types of orders and to read out account information. The overall goal is to allow the implementation automated, algortithmic trading programs.

In this documentation, you learn how to use JavaScript and node.js in order to connect to the API, get live feed of currency pairs, retrieve historical prices, retrieve acct / open position / closed position table, place market or entry order and modify prices of entry orders.

2. Prerequisites

To get started with the API, a demo account with FXCM is sufficient. You can open such an account under https://www.fxcm.com/uk/algorithmic-trading/api-trading/.

Also you will need to install socket.io library and then reference it in your code.

You can download the code you need from this link
https://github.com/fxcm/fxcm-api-rest-nodejs-example

This is a node.js client which is created to help you use our REST API with ease.

Download the code and use a text editor of your choice to see the node.js program for convenience.

3. First steps

After you’ve done the required steps properly you are good to go.

Now it’s time to paste the token of your account in the *config.sample.js* file.

```
var config = {};

config.token = "PASTE_YOUR_TOKEN_HERE"; // get this from
http://tradingstation.fxcm.com/
config.trading_api_host = 'api-demo.fxcm.com';
config.trading_api_port = 443;
config.trading_api_proto = 'https'; // http or https

module.exports = config;
```

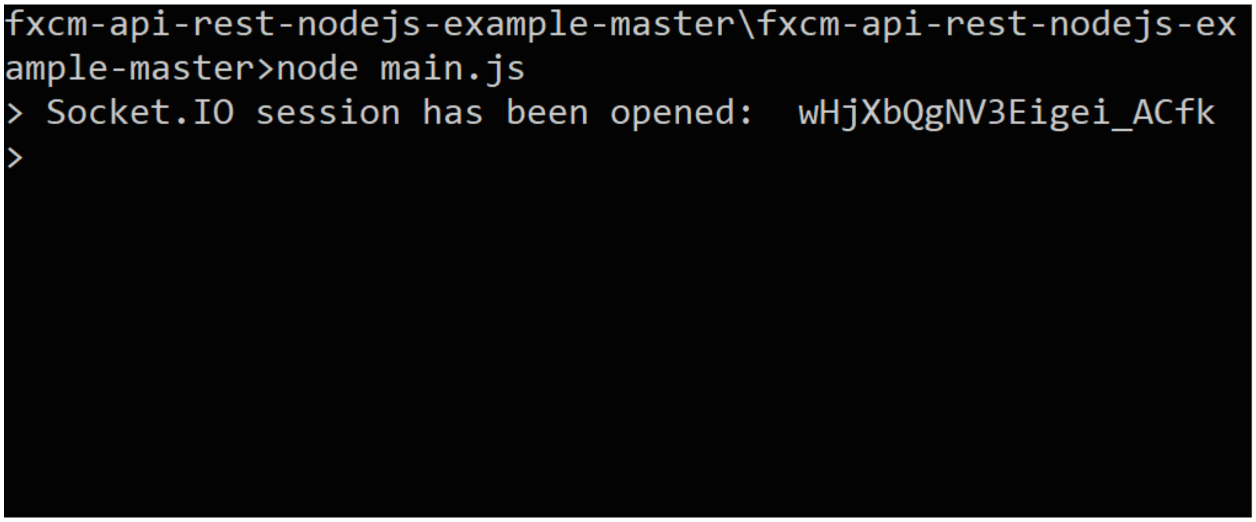
Then you should rename or copy the *config.sample.js* file into *config.js*.

You are able to change the host, port and protocol also.

4. Use our node.js client

Open the command prompt of your PC and navigate do the folder you’ve saved the node.js client in.

Then run it by typing *node main.js*



```
fxcm-api-rest-nodejs-example-master\fxcm-api-rest-nodejs-ex
ample-master>node main.js
> Socket.IO session has been opened:  wHjXbQgNV3Eigei_ACfk
>
```

The *Socket.IO session has been opened* message means that you’ve established a connection with no errors and you are ready to use the API.

You can do so by copying the commands from the *commands.txt* file.

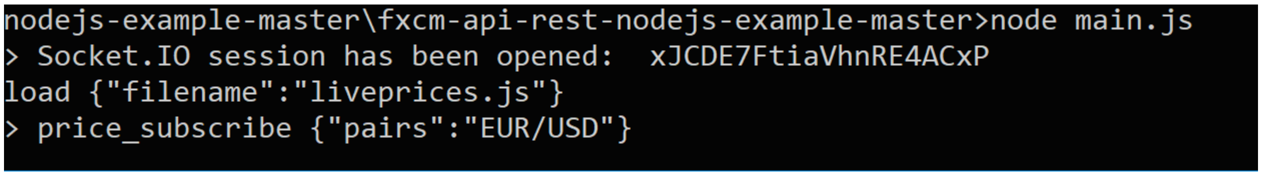
Examples:

Get live feed (EUR/USD):

For the sake of convenience you can copy the required commands from here:

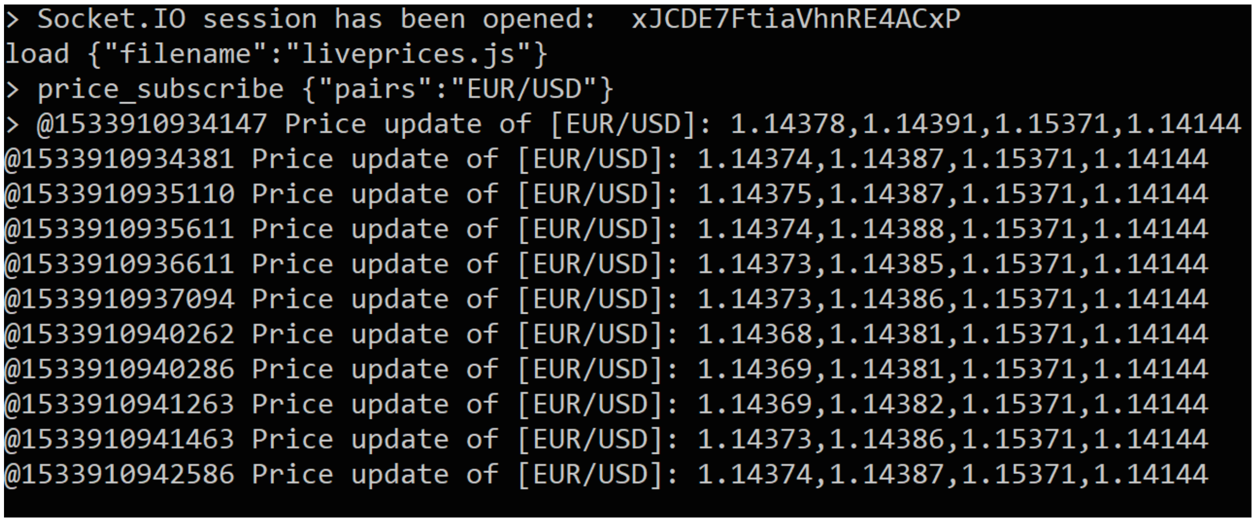
load {"filename":"liveprices.js"}

price_subscribe {"pairs":"EUR/USD"}



```
nodejs-example-master\fxcm-api-rest-nodejs-example-master>node main.js
> Socket.IO session has been opened:  xJCDE7FtiaVhnRE4ACxP
load {"filename":"liveprices.js"}
> price_subscribe {"pairs":"EUR/USD"}
```

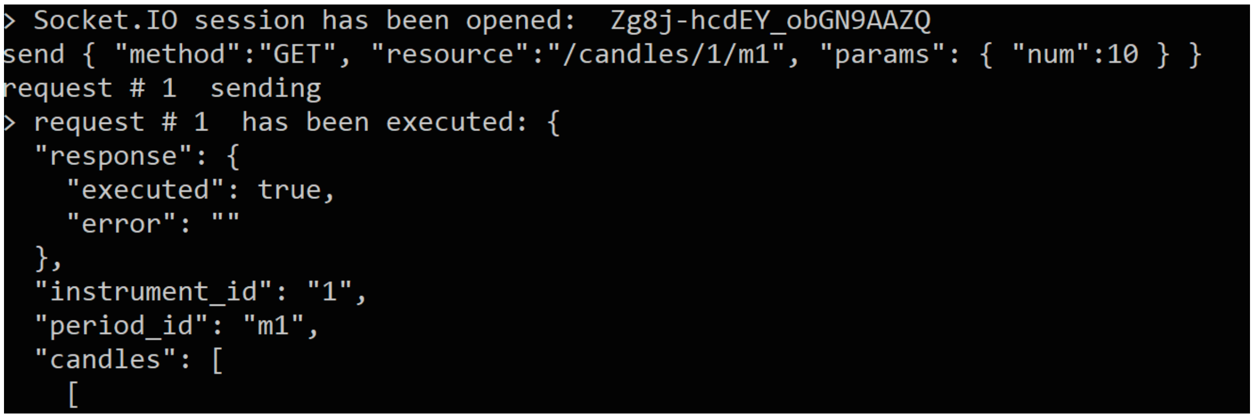
You should get a result that looks like this:



```
> Socket.IO session has been opened:  xJCDE7FtiaVhnRE4ACxP
load {"filename":"liveprices.js"}
> price_subscribe {"pairs":"EUR/USD"}
> @1533910934147 Price update of [EUR/USD]: 1.14378,1.14391,1.15371,1.14144
@1533910934381 Price update of [EUR/USD]: 1.14374,1.14387,1.15371,1.14144
@1533910935110 Price update of [EUR/USD]: 1.14375,1.14387,1.15371,1.14144
@1533910935611 Price update of [EUR/USD]: 1.14374,1.14388,1.15371,1.14144
@1533910936611 Price update of [EUR/USD]: 1.14373,1.14385,1.15371,1.14144
@1533910937094 Price update of [EUR/USD]: 1.14373,1.14386,1.15371,1.14144
@1533910940262 Price update of [EUR/USD]: 1.14368,1.14381,1.15371,1.14144
@1533910940286 Price update of [EUR/USD]: 1.14369,1.14381,1.15371,1.14144
@1533910941263 Price update of [EUR/USD]: 1.14369,1.14382,1.15371,1.14144
@1533910941463 Price update of [EUR/USD]: 1.14373,1.14386,1.15371,1.14144
@1533910942586 Price update of [EUR/USD]: 1.14374,1.14387,1.15371,1.14144
```

Retrieve historical price (m1):

send { "method":"GET", "resource":"/candles/1/m1", "params": { "num":10 } }

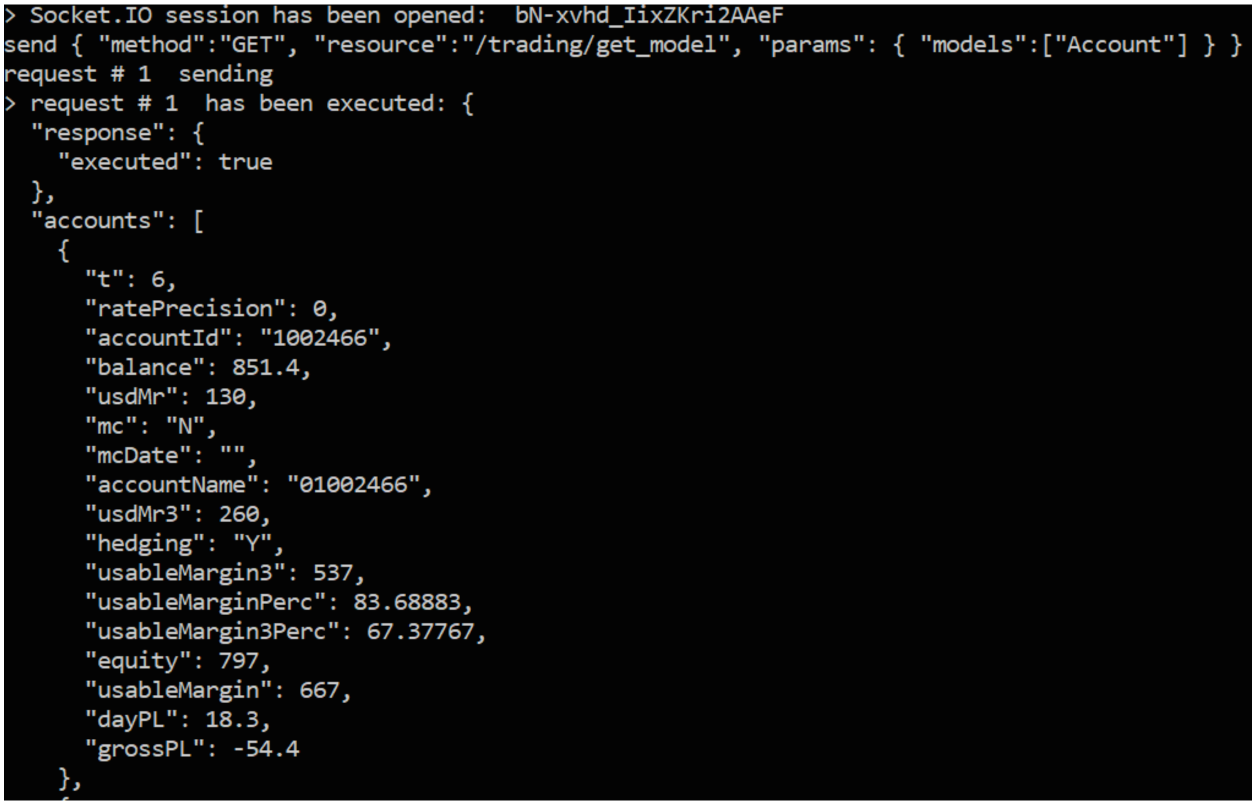


```
> Socket.IO session has been opened:  Zg8j-hcdEY_obGN9AAZQ
send { "method":"GET", "resource":"/candles/1/m1", "params": { "num":10 } }
request # 1  sending
> request # 1  has been executed: {
  "response": {
    "executed": true,
    "error": ""
  },
  "instrument_id": "1",
  "period_id": "m1",
  "candles": [
    [
```

Retrieve account / open position / closed position table :

Account

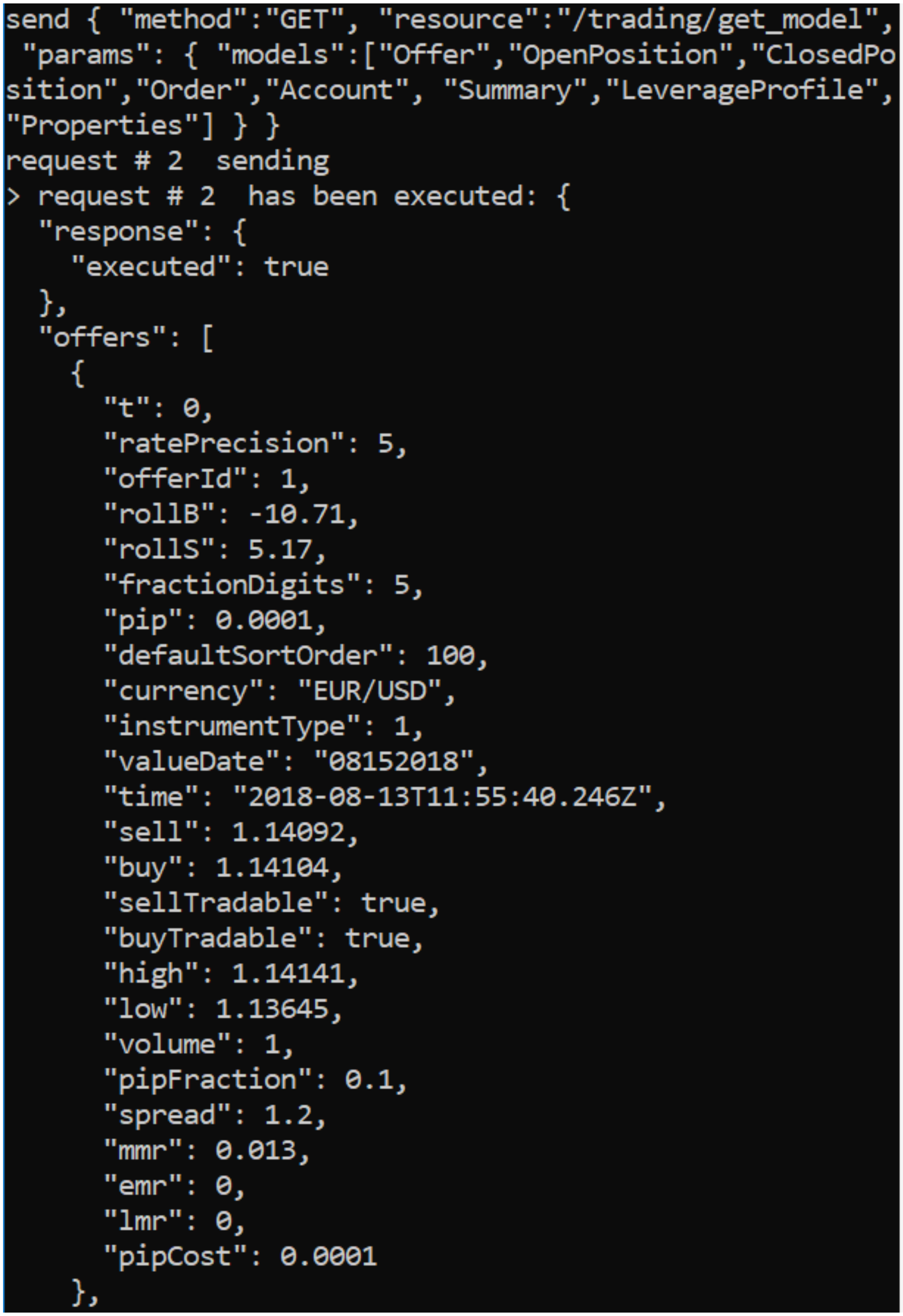
send { "method":"GET", "resource":"/trading/get_model", "params": { "models":["Account"] } }



```
> Socket.IO session has been opened:  bN-xvhd_IixZKri2AAeF
send { "method":"GET", "resource":"/trading/get_model", "params": { "models":["Account"] } }
request # 1  sending
> request # 1  has been executed: {
  "response": {
    "executed": true
  },
  "accounts": [
    {
      "t": 6,
      "ratePrecision": 0,
      "accountId": "1002466",
      "balance": 851.4,
      "usdMr": 130,
      "mc": "N",
      "mcDate": "",
      "accountName": "01002466",
      "usdMr3": 260,
      "hedging": "Y",
      "usableMargin3": 537,
      "usableMarginPerc": 83.68883,
      "usableMargin3Perc": 67.37767,
      "equity": 797,
      "usableMargin": 667,
      "dayPL": 18.3,
      "grossPL": -54.4
    },
```

Open and Close positions

```
send { "method":"GET", "resource":"/trading/get_model", "params": {
"models":["Offer","OpenPosition","ClosedPosition","Order","Account",
"Summary","LeverageProfile","Properties"] } }
```



```
send { "method":"GET", "resource":"/trading/get_model",
 "params": { "models":["Offer","OpenPosition","ClosedPo
sition","Order","Account", "Summary","LeverageProfile",
"Properties"] } }
request # 2  sending
> request # 2  has been executed: {
  "response": {
    "executed": true
  },
  "offers": [
    {
      "t": 0,
      "ratePrecision": 5,
      "offerId": 1,
      "rollB": -10.71,
      "rollS": 5.17,
      "fractionDigits": 5,
      "pip": 0.0001,
      "defaultSortOrder": 100,
      "currency": "EUR/USD",
      "instrumentType": 1,
      "valueDate": "08152018",
      "time": "2018-08-13T11:55:40.246Z",
      "sell": 1.14092,
      "buy": 1.14104,
      "sellTradable": true,
      "buyTradable": true,
      "high": 1.14141,
      "low": 1.13645,
      "volume": 1,
      "pipFraction": 0.1,
      "spread": 1.2,
      "mmr": 0.013,
      "emr": 0,
      "lmr": 0,
      "pipCost": 0.0001
    },
```

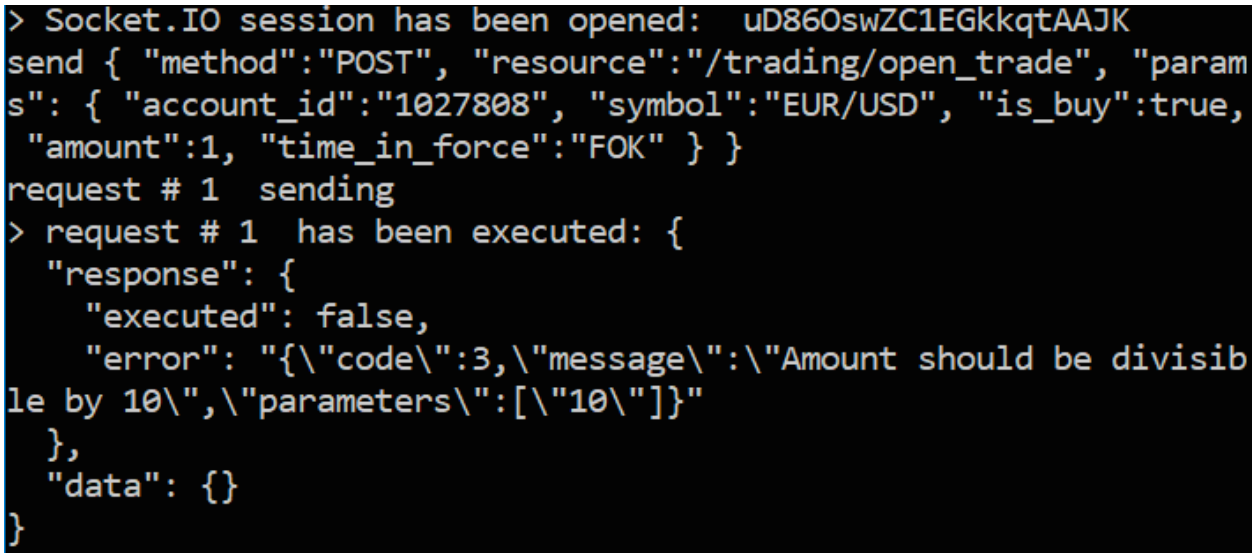
Place market or entry order:

Market order

```
send { "method":"POST", "resource":"/trading/open_trade", "params": { "account_id":"1027808", "symbol":"EUR/USD", "is_buy":true, "amount":1, "time_in_force":"FOK" } }
```

Please note that "account_id" is a number that you can retrieve from Account table as "accountId".

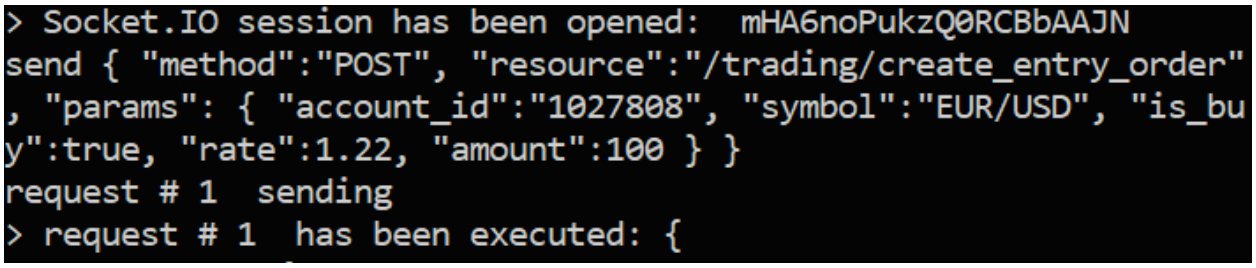
Different accounts have different minimum trade size, and amount must be divisible by that number:



```
> Socket.IO session has been opened:  uD86OswZC1EGkkqtAAJK
send { "method":"POST", "resource":"/trading/open_trade", "param
s": { "account_id":"1027808", "symbol":"EUR/USD", "is_buy":true,
 "amount":1, "time_in_force":"FOK" } }
request # 1  sending
> request # 1  has been executed: {
  "response": {
    "executed": false,
    "error": "{\"code\":3,\"message\":\"Amount should be divisib
le by 10\",\"parameters\":[\"10\"]}"
  },
  "data": {}
}
```

Entry order

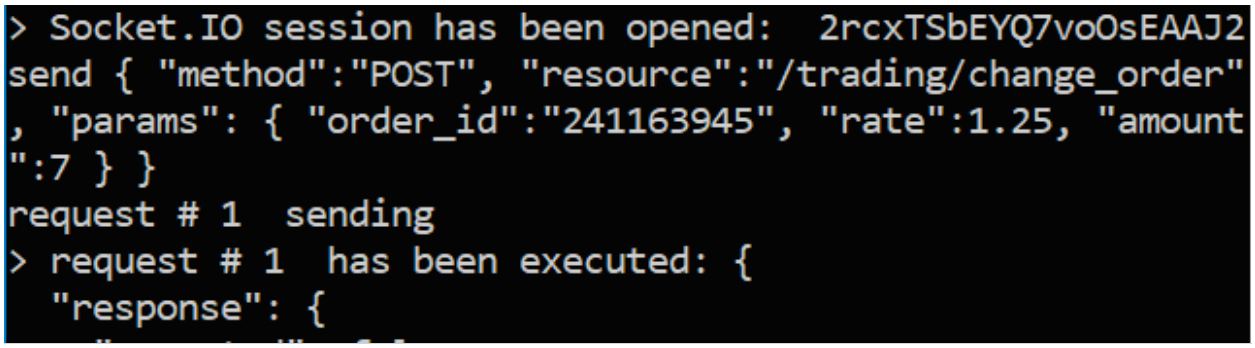
```
send { "method":"POST", "resource":"/trading/create_entry_order", "params": { "account_id":"1027808", "symbol":"EUR/USD", "is_buy":true, "rate":1.22, "amount":100 } }
```



```
> Socket.IO session has been opened:  mHA6noPukzQ0RCBbAAJN
send { "method":"POST", "resource":"/trading/create_entry_order"
, "params": { "account_id":"1027808", "symbol":"EUR/USD", "is_bu
y":true, "rate":1.22, "amount":100 } }
request # 1  sending
> request # 1  has been executed: {
```

Modify the price of entry order:

send { "method":"POST", "resource":"/trading/change_order", "params": { "order_id":"241163945", "rate":1.25, "amount":7 } }



```
> Socket.IO session has been opened:  2rcxTSbEYQ7voOsEAAJ2
send { "method":"POST", "resource":"/trading/change_order"
, "params": { "order_id":"241163945", "rate":1.25, "amount
":7 } }
request # 1  sending
> request # 1  has been executed: {
  "response": {
```

5. Conclusion

This is the JavaScript/node.js startup guide for the FXCM REST API.

Always keep in mind that your account should be valid and you have to choose the correct commands from the commands.txt file. Also be careful with the input parameters you type in the commands.