python-exx Documentation

Release 0.2.0

Sam McHardy

Mar 07, 2018

Contents

This is an unofficial Python wrapper for the EXX exchange REST API v1. I am in no way affiliated with EXX, use at your own risk.

If you came here looking for the EXX exchange to purchase cryptocurrencies, then go here. If you want to automate interactions with EXX stick around.

Source code https://github.com/sammchardy/python-exx

Documentation https://python-exx.readthedocs.io/en/latest/

Make sure you update often and check the Changelog for new features and bug fixes.

CHAPTER 1

Features

- Implementation of all Market Data and Account endpoints.
- Simple handling of authentication
- No need to generate timestamps yourself, the wrapper does it for you
- Response exception handling

CHAPTER 2

Quick Start

Register an account with Exx.

Generate an API Key and save the key and secret.

```
pip install python-exx
```

```
from exx.client import Client
client = Client(api_key, api_secret)

# get market details
markets = client.get_markets()

# get market depth
depth = client.get_order_book('hsr_eth')

# get all symbol prices
prices = client.get_tickers()

# get a symbol prices
price = client.get_ticker('hsr_eth')

# place an order
order = client.create_order('eth_hsr', 'buy', '0.0012', '1023.2')

# cancel an order
res = client.cancel_order('eth_hsr', 1234567)

# get order details
order = client.get_order('eth_hsr', 1234567)

# get open orders
orders = client.get_open_orders('eth_hsr')

# get open buy orders
orders = client.get_open_orders('eth_hsr', 'buy')
```

```
# get second page of open sell orders
orders = client.get_open_orders('eth_hsr', 'sell', 2)
```

For more check out the documentation.

CHAPTER 3

Donate

If this library helped you out feel free to donate.

- ETH: 0xD7a7fDdCfA687073d7cC93E9E51829a727f9fE70
- LTC: LPC5vw9ajR1YndE1hYVeo3kJ9LdHjcRCUZ
- NEO: AVJB4ZgN7VgSUtArCt94y7ZYT6d5NDfpBo
- BTC: 1Dknp6L6oRZrHDECRedihPzx2sSfmvEBys

CHAPTER 4

Other Exchanges

If you use Binance check out my python-binance library.

If you use Quoinex or Qryptos check out my python-quoine library.

If you use Kucoin check out my python-kucoin library.

If you use Allcoin check out my python-allucoin library.

If you use IDEX check out my python-idex library.

If you use BigONE check out my python-bigone library.

4.1 Contents

4.1.1 Getting Started

Installation

`python-exx` is available on PYPI. Install with `pip`:

```
pip install python-exx
```

Register on EXX

Firstly register an account with EXX.

Generate an API Key

To use signed account methods you are required to create an API Key and save the key and secret.

Initialise the client

Pass your API Key and Secret

```
from exx.client import Client
client = Client(api_key, api_secret)
```

API Rate Limit

Each IP can send maximum of 1000 https requests per minute. If you exceed 1000 requests, the system will automatically block the IP for one hour. After an hour, the IP will be automatically unlocked.

Each user can send a maximum of 10 request within one second.

4.1.2 Changelog

v0.0.2 - 2018-02-02

Added

- User Agent header
- default requests timeout of 10s

Fixed

- Default page value for *get_open_orders*
- Handle empty response exception on *get_open_orders* and return empty list

v0.0.1 - 2018-01-18

Initial version

Added

- General, Market Data and Order endpoints

4.1.3 EXX API

client module

class exx.client.**Client**(*api_key*, *api_secret*)

Bases: object

API_URL = 'https://api.exx.com/data/v1'

PRIVATE_URL = 'https://trade.exx.com/api'

SIDE_BUY = 'buy'

SIDE_SELL = 'sell'

__init__(*api_key*, *api_secret*)

Exx API Client constructor

https://www.exx.com/help/restApi

Parameters

- **api_key** (*string*) – Api Token Id
- **api_secret** (*string*) – Api Secret

```
client = Client(api_key, api_secret)
```

cancel_order(*symbol*, *order_id*)
Cancel an order

Parameters

- **symbol** (*int*) – required e.g eth_hsr
- **order_id** – required e.g 123456789

```
res = client.cancel_order('eth_hsr', 123456789)
```

Returns dict

```
{
    "code": "100",
    "message": ""
}
```

Raises ExxResponseException, ExxAPIException

create_order(*symbol*, *order_type*, *price*, *amount*)
Cancel an order

Parameters

- **symbol** (*str*) – e.g eth_hsr
- **order_type** – type buy or sell
- **price** – price to trade at
- **amount** – amount to trade

```
order = client.create_order('eth_hsr', 'buy', '0.0012', '1023.2')
```

Returns dict

```
{
    "code": 100,
    "message": "",
    "id": "13877"
}
```

Raises ExxResponseException, ExxAPIException

get_balance()
Get your balance

```
orders = client.get_balance()
```

Returns dict


```
{
    "credits": [
        {
            "flatRatio": "0.1",
            "userRatio": "0.0985",
            "noticeRatio": "0.2",
            "levels": 10,
            "flatPrice": 11.01471399
        }
    ],
    "funds": {
        "BTS": {
            "total": "10",
            "freeze": "0",
            "balance": "10",
            "propTag": "BTS",
            "credit_quota": "121.938066",
            "credit_borrowed": "0",
            "credit_interest": "0"
        },
        "MONA": {
            "total": "0.966033",
            "freeze": "0.966033",
            "balance": "0",
            "propTag": "MONA",
            "credit_quota": "0",
            "credit_borrowed": "0",
            "credit_interest": "0"
        },
        ....
        "ETH": {
            "total": "10",
            "freeze": "0",
            "balance": "10",
            "propTag": "ETH",
            "credit_quota": "121.938066",
            "credit_borrowed": "0",
            "credit_interest": "0"
        },
        "LTC": {
            "total": "0",
            "freeze": "0",
            "balance": "0",
            "propTag": "LTC",
            "credit_quota": "121.938066",
            "credit_borrowed": "0",
            "credit_interest": "0"
        },
        "QTUM": {
            "total": "0.003",
            "freeze": "0.003",
            "balance": "0",
            "propTag": "QTUM",
            "credit_quota": "0",
            "credit_borrowed": "9.65",
            "credit_interest": "0.026252"
```

```
        }
    }
}
```

Raises ExxResponseException, ExxAPIException

get_market_trades(*symbol*)
Get the trades for the symbol

Parameters **symbol** (*str*) – required e.g eth_hsr

```
trades = client.get_market_trades('eth_hsr')
```

Returns list of dicts

```
[
    {
        "amount"     : 0.933,
        "price"      : 31.595,
        "tid"        : 2583932,      # Trade ID
        "type"       : "sell",       # Trade type
        "date"       : 2583932,
        "trade_type" : "ask",        # Order type
    }, ...
]
```

Raises ExxResponseException, ExxAPIException

get_markets()
Get a list of markets

```
markets = client.get_markets()
```

Returns dict of dicts

```
{
     "eos_btc":{
         "amountScale":2,
         "priceScale":6,
         "maxLevels":0,
         "isOpen":false
     },
     "etc_hsr":{
         "amountScale":3,
         "priceScale":3,
         "maxLevels":0,
         "isOpen":true
     },
}
```

Raises ExxResponseException, ExxAPIException

get_open_orders(*symbol*, *order_type=None*, *page=1*)
Get a list of open buy or sell orders, 10 at a time

Parameters

- **symbol** (*str*) – e.g eth_hsr
- **order_type** – optional - type buy or sell
- **page** (*int*) – page index starting at 1

```
# get first page of open orders
orders = client.get_open_orders('hsr_eth')

# get first page of buy orders
orders = client.get_open_orders('hsr_eth', 'buy')

# second page of sell orders
orders = client.get_open_orders('hsr_eth', 'sell', 2)
```

Returns list of dicts

```
{
    "fees": 0,
    "total_amount": 1,
    "trade_amount": 0,
    "price": 31,
    "currency": "eth_hsr",
    "id": "13877",
    "trade_money": 0,
    "type": "buy",
    "trade_date": 1509728383300,
    "status": 0
}
```

Raises ExxResponseException, ExxAPIException

get_order(*symbol*, *order_id*)

Cancel an order

Parameters

- **symbol** (*int*) – required e.g eth_hsr
- **order_id** – required e.g 123456789

```
order = client.get_order('eth_hsr', 123456789)
```

Returns dict

```
{
    "fees": 0,
    "total_amount": 1,
    "trade_amount": 0,
    "price": 31,
    "currency": "eth_hsr",
    "id": "13877",
    "trade_money": 0,
    "type": "buy",
    "trade_date": 1509728383300,
```

```
    "status": 0
}
```

Raises ExxResponseException, ExxAPIException

get_order_book(*symbol*)

Get the bid and asks for the symbol

Parameters **symbol** (*str*) – required e.g eth_hsr

```
markets = client.get_order_book('eth_hsr')
```

Returns dict

```
{
    "asks": [
        [
            "32.831",    # price
            "0.083"      # quantity
        ]...
    ],
    "bids": [
        [
            "30.434",    # price
            "10.766"     # quantity
        ]...
    ],
    "timestamp" : Timestamp
}
```

Raises ExxResponseException, ExxAPIException

get_ticker(*symbol*)

Get symbol price ticker

Parameters **symbol** (*str*) – required e.g eth_hsr

```
markets = client.get_ticker('eth_hsr')
```

Returns dict

```
{
    "ticker": {
        "vol": "1447.851",
        "last": "30.487000000",
        "sell": "30.499",
        "buy": "30.487",
        "weekRiseRate": -1.17,
        "riseRate": 9.45,
        "high": "30.812",
        "low": "27.855",
        "monthRiseRate": -0.99
    },
    "date": "1510383406453"
}
```

Raises ExxResponseException, ExxAPIException

get_tickers()
Get all price tickers

```
markets = client.get_markets()
```

Returns dict of dicts

```
{
    "bts_btc":{
        "vol":0.0,
        "last":0,
        "sell":0.0,
        "buy":0.0,
        "weekRiseRate":0.0,
        "riseRate":0.0,
        "high":0.0,
        "low":0,
        "monthRiseRate":0.0
    },
}
```

Raises ExxResponseException, ExxAPIException

exceptions module

exception exx.exceptions.**ExxAPIException**(*response*)
Bases: exceptions.Exception

Exception class to handle general API Exceptions

code values

message format

__init__(*response*)

exception exx.exceptions.**ExxRequestException**(*message*)
Bases: exceptions.Exception

__init__(*message*)

4.2 Index

- genindex

Python Module Index

e

Index

Symbols

A

C

E

G

P

S