EasyIB

Release 0.0.3

ashpipe

Feb 21, 2023

CONTENTS

"Logo for 'EasyIB'" according to Midjourney

EasyIB is an unofficial python wrapper for Interactive Brokers Client Portal Web API. The motivation for the project was to build a Python wrapper that can run on Linux/cloud environments. Thus, Client Portal API was preferred over Trader Workstation (TWS) API.

Please see https://easyib.readthedocs.io for the full documentation.

CHAPTER

ONE

FEATURES

Notable functionality includes:

- Pull account info, portfolio, cash balance, the net value
- Pull historical market data
- Submit, modify, cancel orders
- Get order status, list of live orders
- Ping (tickle) server, get authentication status, re-authenticate

CHAPTER

TWO

HOW TO INSTALL

EasyIB assumes a gateway session is active and authenticated. Follow instructions at https://interactivebrokers.github.io/cpwebapi/ for authentication. A custom package such as Voyz/IBeam can also be used for setting up an active session. Part Time Larry has an excellent youtube tutorial on this topic: https://www.youtube.com/watch?v=O1OhiiCx6Ho.

EasyIB was developed under the Voyz/Ibeam docker image environment.

Once a gateway session is running, `pip` command can be used to install EasyIB:

```
pip install easyib
```

CHAPTER

THREE

QUICK START

3.1 Historical data

```
import easyib

ib = easyib.REST() # default parameters: url="https://localhost:5000", ssl=False

bars = ib.get_bars("AAPL", period="1w", bar="1d")
print(bars)
```

3.2 Submitting an order

```
list_of_orders = [
    {
        "conid": ib.get_conid("AAPL"),
        "orderType": "MKT",
        "side": "BUY",
        "quantity": 7,
        "tif": "GTC",
    }
]

order = ib.submit_orders(list_of_orders)
print(order)
```

CHAPTER

FOUR

REFERENCE

For the complete reference, please visit https://easyib.readthedocs.io/en/latest/reference.html.

4.1 REST

By default, EasyIB assumes the gateway session is open at https://localhost:5000 without an SSL certificate. A custom URL and SSL certificate can be set by:

```
ib = easyib.REST(url="https://localhost:5000", ssl=False)
```

4.2 API REST Methods

Documentation of available functions is at https://easyib.readthedocs.io/en/latest/reference.html.

See the official documentation of the End Point at https://www.interactivebrokers.com/api/doc.html.

REST Method	End Point	Result
get_accounts()	Get portfolio/accounts	list
switch_account(accountId: str)	Post iserver/account/{accountId}	dict
get_cash()	Get portfolio/{accountId}/ledger	float
get_netvalue()	Get portfolio/{accountId}/ledger	float
get_conid(symbol: str, instrument_filters: Dict = None, contract_filters: Dict = {"isUS": True})	Get trsv/stocks	int
get_fut_conids(symbol: str)	Get trsv/futures	list
get_portfolio()	Get portfolio/{accountId}/positions/0	dict
reply_yes(id: str)	Post iserver/reply/{id}	dict
submit_orders(list_of_orders: list, reply_yes=True)	Post iserver/account/{acountId}/orders	dict
get_order(orderId: str)	Get iserver/account/order/satus/	dict
get_live_orders(filters=None)	Get iserver/account/orders	dict
cancel_order(orderId: str)	Delete iserver/account/{accountId}/order/{orderId}	dict
modify_order(orderId=None, order=None, reply_yes=True)	Post iserver/account/{accountId}/order/{orderId}	dict
get_bars(symbol: str, period="1w", bar="1d", outsideRth=False, conid="default")	Get iserver/marketdata/history	dict
ping_server()	Post tickle	dict
get_auth_status()	Post iserver/auth/status	dict
re_authenticate()	Post iserver/reauthenticate	None
log_out()	Post logout	None

CHAPTER

FIVE

CONTENTS

5.1 Reference

Reference for easyib.REST class.

class easyib.**REST**(*url='https://localhost:5000'*, *ssl=False*)

Allows to send REST API requests to Interactive Brokers Client Portal Web API.

Parameters

- **url** (*str, optional*) – Gateway session link, defaults to "https://localhost:5000"
- **ssl** (*bool, optional*) – Usage of SSL certificate, defaults to False

5.1.1 Account info

REST.**get_accounts**() → list

Returns account info

Returns
list of account info

Return type
list

REST.**switch_account**(*accountId: str*) → dict

Switch selected account to the input account

Parameters
accountId (*str*) – account ID of the desired account

Returns
Response from the server

Return type
dict

REST.**get_portfolio**() → dict

Returns portfolio of the selected account

Returns
Portfolio

Return type
dict

REST.**get_cash**() → float

Returns cash balance of the selected account

Returns
cash balance

Return type
float

REST.**get_netvalue**() → float

Returns net value of the selected account

Returns
Net value in USD

Return type
float

5.1.2 Instrument info

REST.**get_conid**(*symbol: str*, *instrument_filters: Optional[dict] = None*, *contract_filters: dict = {'isUS': True}*) → int

Returns contract id of the given stock instrument

Parameters

- **symbol** (*str*) – Symbol of the stock instrument
- **instrument_filters** (*Dict, optional*) – Key-value pair of filters to use on the returned instrument data, e.g) {"name": "ISHARES NATIONAL MUNI BOND E", "assetClass": "STK"}
- **contract_filters** (*Dict, optional*) – Key-value pair of filters to use on the returned contract data, e.g) {"isUS": True, "exchange": "ARCA"}

Returns
contract id

Return type
int

REST.**get_fut_conids**(*symbol: str*) → list

Returns list of contract id objects of a future instrument.

Parameters
symbol (*str*) – symbol of a future instrument

Returns
list of contract id objects

Return type
list

REST.**get_bars**(*symbol: str*, *period='1w'*, *bar='1d'*, *outsideRth=False*, *conid: str = 'default'*) → dict

Returns market history for the given instrument. conid should be provided for futures and options.

Parameters

- **symbol** (*str*) – Symbol of the stock instrument

- **period** (*str, optional*) – Period for the history, available time period– {1-30}min, {1-8}h, {1-1000}d, {1-792}w, {1-182}m, {1-15}y, defaults to "1w"
- **bar** (*str, optional*) – Granularity of the history, possible value– 1min, 2min, 3min, 5min, 10min, 15min, 30min, 1h, 2h, 3h, 4h, 8h, 1d, 1w, 1m, defaults to "1d"
- **outsideRth** (*bool, optional*) – For contracts that support it, will determine if historical data includes outside of regular trading hours., defaults to False
- **conid** (*str or int, optional*) – conid should be provided separately for futures or options. If not provided, it is assumed to be a stock.

Returns
Response from the server

Return type
dict

5.1.3 Orders

REST.**submit_orders**(*list_of_orders: list*, *reply_yes=True*) → dict

Submit a list of orders

Parameters

- **list_of_orders** (*list*) – List of order dictionaries. For each order dictionary, see here for more details.
- **reply_yes** (*bool, optional*) – Replies yes to returning messages or not, defaults to True

Returns
Response to the order request

Return type
dict

REST.**modify_order**(*orderId: Optional[str] = None*, *order: Optional[dict] = None*, *reply_yes=True*) → dict

Modify submitted order

Parameters

- **orderId** (*str*) – Order ID of the submitted order, defaults to None
- **order** (*dict*) – Order dictionary, defaults to None
- **reply_yes** (*bool, optional*) – Replies yes to the returning messages, defaults to True

Returns
Response from the server

Return type
dict

REST.**cancel_order**(*orderId: str*) → dict

Cancel the submitted order

Parameters
orderId (*str*) – Order ID for the input order

Returns
Response from the server

Return type
dict

REST.**get_order**(*orderId: str*) → dict

Returns details of the order

Parameters
orderId (*str*) – Order ID of the submitted order

Returns
Details of the order

Return type
dict

REST.**get_live_orders**(*filters: Optional[list] = None*) → dict

Returns list of live orders

Parameters
filters (*list, optional*) – List of filters for the returning response. Available items – "inactive" "pending_submit" "pre_submitted" "submitted" "filled" "pending_cancel" "cancelled" "warn_state" "sort_by_time", defaults to []

Returns
list of live orders

Return type
dict

REST.**reply_yes**(*id: str*) → dict

Replies yes to a single message generated while submitting or modifying orders.

Parameters
id (*str*) – message ID

Returns
Returned message

Return type
dict

5.1.4 Communication with server

REST.**ping_server**() → dict

Tickle server for maintaining connection

Returns
Response from the server

Return type
dict

REST.**get_auth_status**() → dict

Returns authentication status

Returns
Status dictionary

Return type
dict

REST.**re_authenticate**() → None

Attempts to re-authenticate when authentication is lost

REST.**log_out**() → None

Log out from the gateway session

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX