OpenAPI Introduction

Overview

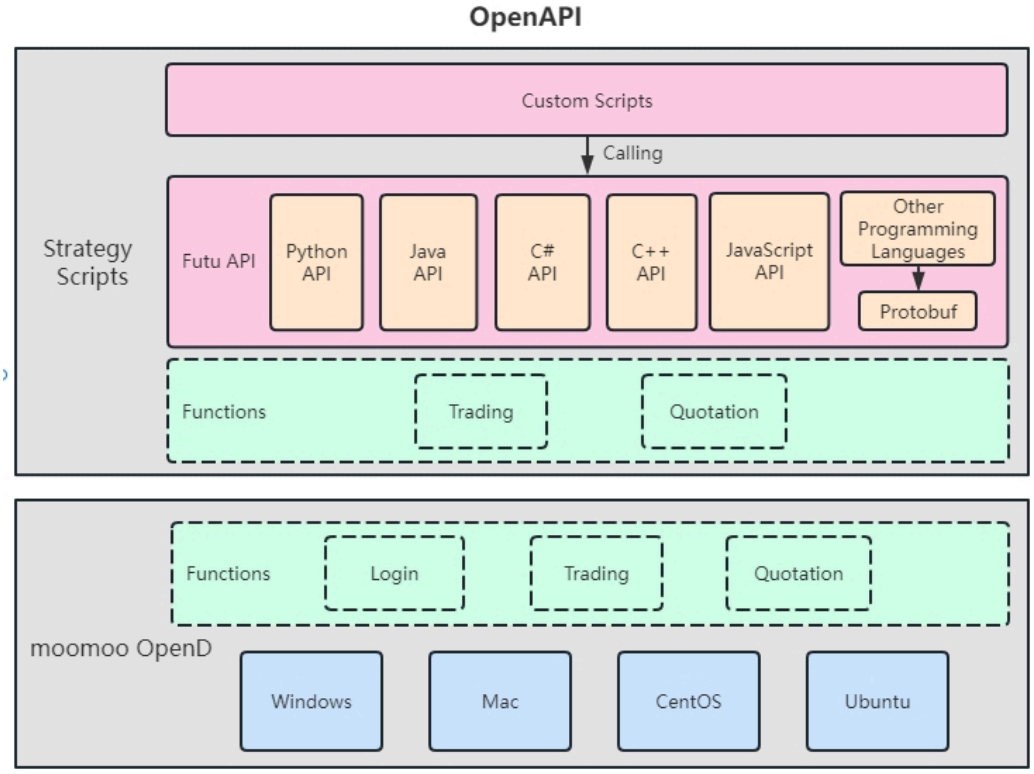
OpenAPI provides wide varieties of market data and trading services for your programmed trading to meet the needs of every developer's programmed trading and help your Quant dreams.

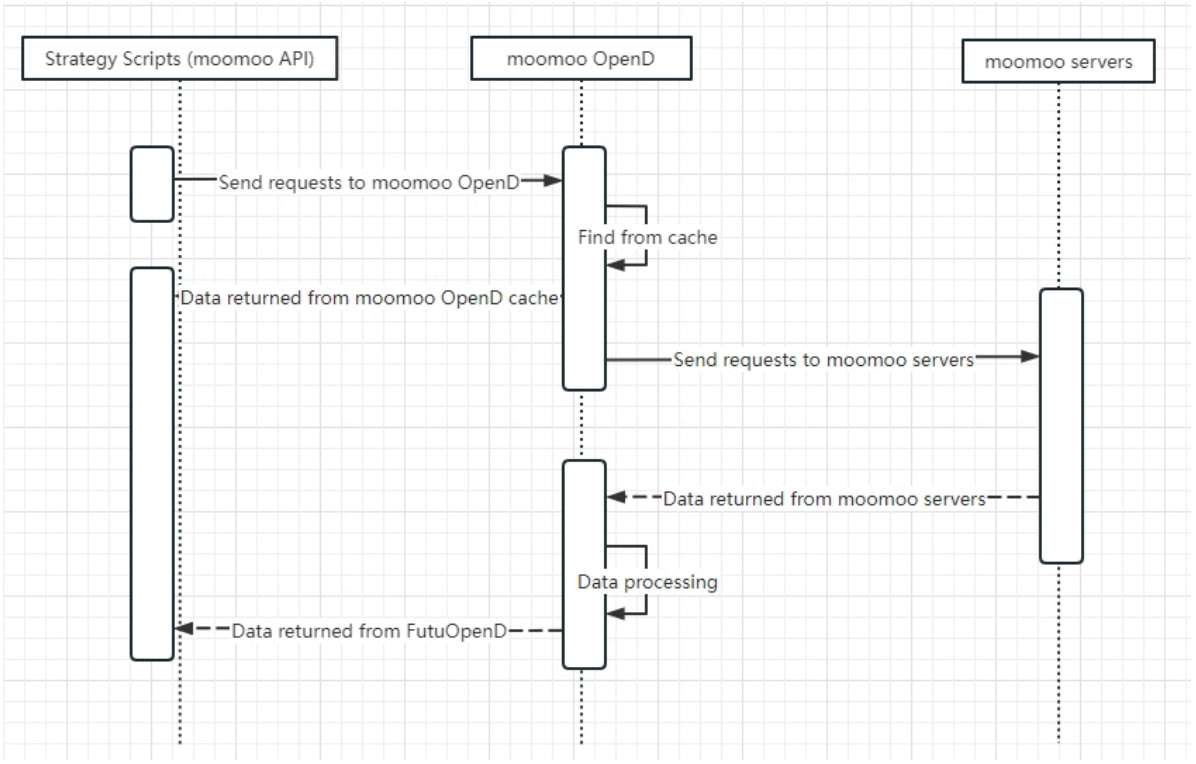
Moomoo users can click here to learn more.

OpenAPI consists of *OpenD* and *moomoo API*:

- *OpenD* is the gateway program of *moomoo API*, running on your local computer or cloud server. It is responsible for transferring the protocol requests to moomoo servers, and returning the processed data.
- moomoo API is an API SDK encapsulated by moomoo, including mainstream programming languages (Python, Java, C#, C++, JavaScript), to reduce the difficulty of your trading strategy development. If the language you want to use is not listed above, you can still interface with the protocol yourself to complete the trading strategy development.

Diagrams below illustrate the architecture of OpenAPI.





The first time using OpenAPI, you need to finish the following two steps:

The first step is to install and start a gateway program **OpenD** locally or in the cloud.

OpenD exposes the interfaces in the way of TCP, which is responsible for transferring the protocol requests to moomoo servers and returning the processed data. The protocol interface has nothing to do with the type of programming language.

The second step is to download moomoo API and complete **Environment Setup**.

For your convenience, moomoo encapsulates API SDK for mainstream programming languages (hereinafter referred to as moomoo API).

Account

OpenAPI involves two types of accounts, *moomoo ID* and *universal account*.

moomoo ID

moomoo ID is your user account (moomoo ID), which can be used in moomoo APP and OpenAPI.
You can use your *moomoo ID* and *login password* to log in to OpenD and obtain market data.

Universal Account

Universal account allows trading across multiple markets (including Hong Kong stocks, US stocks, A-shares, and funds) in various currencies. There's no need for multiple accounts.
Universal Accounts come in two forms:

- Securities Universal Account: Trade stocks, ETFs, options, and other securities across different markets.
- Futures Universal Account: Trade futures, including Hong Kong, US CME Group, Singapore, and Japanese futures.

Functionality

There are 2 functions of OpenAPI: quotation and trading.

Quotation Functions

Quotation Data Categories

Including stocks, indices, options and futures from HK, US and A-share market. Find the specific types of support in the table below. You need authorities for each kinds of market data. For more details on how to obtain authorities, please click here.

Market	Contract	Moomoo Users
HK Market	Stocks, ETFs, Warrants, CBBCs, Inline Warrants	✓
	Options	✓
	Futures	✓
	Indices	✓
	Plates	✓
US Market	Stocks, ETFs	✓
	OTC Securities	X
	Options	✓
	Futures	✓
	Indices	X
	Plates	✓
A-share Market	Stocks, ETFs	✓
	Indices	✓
	Plates	✓
Singapore Market	Stocks, ETFs, Warrants, REITs, DLCs	X
	Futures	X
Japanese Market	Stocks, ETFs, REITs	X
	Futures	X
Australian Market	Stocks, ETFs	X
Global Markets	Forex	X

Method to Obtain Market Data

- Subscribe and receive pushed real-time quote, candlestick, tick-by-tick and order book.
- Request for the latest market snapshot, historical candlesticks etc.

Trading Functions

Trading Capacity

Including stocks, options and futures from HK, US, A-share, Singapore and Japanese markets. Find the specific types of support in the table below:

Market	Contracts	Paper Trading	Live Trading						
			FUTU HK	Moomoo US	Moomoo SG	Moomoo AU	Moomoo MY	Moomoo CA	Moomoo JP
HK Market	Stocks, ETFs, Warrants, CBBCs, Inline Warrants	✓	✓	✓	✓	✓	✓	X	X
	Options	✓	✓	X	X	X	X	X	X
	Futures	✓	✓	X	X	X	X	X	X
US Market	Stocks, ETFs	✓	✓	✓	✓	✓	✓	✓	✓
	Options	✓	✓	✓	✓	✓	✓	✓	✓
	Futures	✓	✓	X	✓	X	✓	X	X
A-share Market	China Connect Securities stocks	✓	✓	✓	✓	X	✓	X	X
	Non-China Connect Securities stocks	✓	X	X	X	X	X	X	X
Singapore Market	Stocks, ETFs, Warrants, REITs, DLCs	X	X	X	X	X	X	X	X
	Futures	✓	✓	X	✓	X	X	X	X
Japanese Market	Stocks, ETFs, REITs	X	X	X	X	X	X	X	X
	Futures	✓	✓	X	X	X	X	X	X
Australian Market	Stocks, ETFs	X	X	X	X	X	X	X	X
Canadian Market	Stocks, ETFs	X	X	X	X	X	X	X	X

Method of Trading

The trading interfaces are used for both live trading and paper trading.

Features

1. Full platform and multi-language

- OpenD supports Windows, MacOS, CentOS, Ubuntu
- moomoo API supports Python, Java, C#, C++, JavaScript, etc.

2. Stable speed and free

- Stable technical architecture, directly connected to the exchanges
- The fastest order is 0.0014s
- There is no additional charge for trading via OpenAPI

3. Abundant investment varieties

- Supporting real-time market data, live trading, and simulated trading in multiple markets including United States, Hong Kong, etc.

4. Professional institutional services

- Customized market data and trading solutions

Authorities and Limitations

Login Limitations

Opening Accounts

You need to finish opening your trading accounts on moomoo APP, before logging in to OpenAPI.

Compliance Confirmation

After the first login, you need to complete *API Questionnaire and Agreements* before you can continue to use OpenAPI. Click here for moomoo users.

Quotation Data

There are several limitations for market quotation data as follow:

- **Quote Right** -- The authority to obtain the relevant market data.
- **Interface Frequency Limitations** -- Frequency limits of calling interfaces.
- **Subscription Quota** -- Number of real-time quotes subscribed at the same time.
- **Historical Candlestick Quota** -- The total number of subjects pulling the historical candlestick per 30 days.

Quote Right

You need the corresponding permission to obtain data of each market through OpenAPI. The permission of OpenAPI is not exactly the same as that of APP. Different levels correspond to different time delay, order book levels, and the permission to use the interface.

You need to buy a quotation card before you can obtain the quotation of some varieties, the specific way to obtain is shown in the table below.

Market	Security Type	Quote Right Acquisition Method

HK Market	Securities (including stocks, ETFs, warrants, CBBCs, Inline Warrants)	* Chinese mainland customers: LV2 market quotes for free. Purchase **HK Stocks Advanced Full Market Quotes** for SF market quotes * Non-Chinese mainland customers: LV1 market quotes for free. Purchase **HK stocks LV2 advanced market** for LV2 market quotes. Purchase **HK Stocks Advanced Full Market Quotes** for SF market quotes
	Indices	
	Plates	
	Options	* Chinese mainland customers: LV2 market quotes for free during promotion period. * Non-Chinese mainland customers: LV1 market quotes for free. Purchase **HK stock options futures LV2 advanced market** for LV2 market data
	Futures	
US Market	Securities (Covers NYSE, NYSE-American and Nasdaq listed equities, ETFs)	* Not shared with App market data permissions. Purchase **Nasdaq Basic** for LV1 market quotes (basic quotes, 24H). * Not shared with App market data permissions. Purchase **Nasdaq Basic+TotalView** for LV2 market quotes (Full Order Book for 24H trading).
	Plates	
	OTC Securities	Unsupported.
	Options (Covers US stock options, US index options)	* Customers who meet the threshold : get LV1 market data for free * Customers who do not meet the threshold : Purchase **OPRA Options Real-time Quote** for LV1 market data
	Futures	* For clients who have a futures account. For CME Group quotes , please access the **CME Group Futures LV2** For CME quotes, please access the **CME Futures LV2** For CBOT quotes, please access the **CBOT Futures LV2** For NYMEX quotes, please access the **NYMEX Futures LV2** For NYMEX quotes, please access the **COMEX Futures LV2** * For clients who do not have a futures account: Unsupported.

	Indices	Unsupported.
A-share Market	Securities (including stocks, ETFs)	* Chinese mainland customers: LV1 market data for free. * Non-Chinese mainland customers/institutional customers: Unsupported.
	Indices	
	Plates	
Singapore Market	Futures	Unsupported.
Japanese Market	Futures	Unsupported.

> **Tips**
>
> In the above table, the Chinese mainland customers and the Non-Chinese mainland customers are distinguished by the login IP address of OpenD.

Interface Frequency Limitations

In order to protect the server from malicious attacks, there are frequency limitations for all interfaces that need to send requests to moomoo servers. The frequency limitation rules for each API are different. For more information, please see **Interface Limitations** at the bottom of each API page.

Example:
The limitation rule of **Get Market Snapshot** is: A maximum of 60 requests every 30 seconds. You can request a uniform request every 0.5 seconds. You can also quickly request 60 times, rest for 30 seconds, and then request the next round. If the frequency limitation is exceeded, an error will be returned by the interface.

Subscription Quota & Historical Candlestick Quota

The limitation rules of subscription quota and historical candlestick quota as follows:

User Type	Subscription Quota	Historical Candlestick Quota

Finished Opening trading accounts.	100	100
Total asset > 10,000 HKD.	300	300
Satisfy 1 of the items following: 1. Total asset > 500,000 HKD; 2. The number of monthly filled orders > 200; 3. Monthly trading volume > 2 million HKD.	1000	1000
Satisfy 1 of the items following: 1. Total asset > 5 million HKD; 2. The number of monthly filled orders > 2000; 3. Monthly trading volume > 20 million HKD.	2000	2000

1. Total asset

Total asset, refers to all your assets in moomoo, including securities, futures, funds and bonds assets, converted into HKD according to the spot exchange rate.

2. The monthly number of filled orders

It is calculated by taking the larger value of the number of filled orders the last natural month and that of the current natural month, that is:

max (the number of filled orders of the last natural month, the number of filled orders of the current natural month)

3. Monthly Trading volume

It is calculated by taking the larger value of the total trading volume of your last natural month and that of the current natural month, which is converted into HKD according to the spot exchange rate, that is:

max (the total trading volume of the last natural month, the total trading volume of the current natural month)

The calculation of futures trading value needs to be multiplied by the adjustment factor (0.1 by default). The formula for calculating futures trading volume is as follows:

Futures trading value = ∑ (volume of a single transaction * trading price * contract multiplier * exchange rate * adjustment factor)

4. Subscription Quota

It is applicable to the real-time data interface that can only be obtained after a subscription. One type of subscription for each stock takes up 1 subscription quota, and canceling the

subscription will release the occupied quota.
Example:
Suppose your Subscription Quota is 100. When you subscribe to real-time order book for HK.00700, real-time ticker for US.AAPL, and real-time quotation for SH.600519 at the same time, the Subscription Quota will occupy 3, and the remaining Subscription Quota will be 97. At this time, if you cancel the real-time order book subscription of HK.00700, your Subscription Quota will become 2, and the remaining Subscription Quota will become 98.

5. Historical Candlestick Quota

Suitable for **Get Historical Candlesticks** interface. In the last 30 days, requests for historical candlesticks of each stock will occupy one Historical Candlestick Quota. Repeated requests for historical candlestick of the same stock within the last 30 days will not be counted repeatedly. Meanwhile, subscribing to candlestick of different periods for the same stockonly occupies one quota and will not be accumulated repeatedly. Example:
Suppose your Historical Candlestick Quota is 100, and today is July 5, 2020. You have requested historical candlesticks for a total of 60 stocks between June 5, 2020 and July 5, 2020. The remaining Historical Candlestick Quota is 40.

> **Tips**
>
> - Subscription Quota and Historical Candlestick Quota are automatically assigned and do not need to be applied manually.
> - For newly deposited accounts, the quota will automatically take effect within 2 hours.
> - *Asset in Transit* will not be calculated in quota assign.

Trading Functions

- When you trade in a specific market, you need to first confirm whether a trading account has been opened in that market.
  For example: you can only trade US stocks under the US trading account, but not under the HK trading account.

Fee

Quote

LV2 HK market quotes and A-share LV1 market quotes are free for Chinese mainland customers.
For some variaties, you need to buy quotation cards before obtaining market data. For more details of quotation cards prices, please click **Quote Right** and go to data store.

Trade

There is no extra fee for tradings through OpenAPI. The transaction fee is the same as that of APP. You can check the specific charging plans from the following table:

Securities Firm	Charging Options
FUTU HK	Charging Options
Moomoo US	Charging Options
Moomoo SG	Charging Options
Moomoo AU	Charging Options
Moomoo MY	Charging Options
Moomoo CA	Charging Options
Moomoo JP	Charging Options

Visualization OpenD

OpenD provides two operation modes: visualization and command line. Here is a description of Visualization OpenD which is relatively simple to operate.

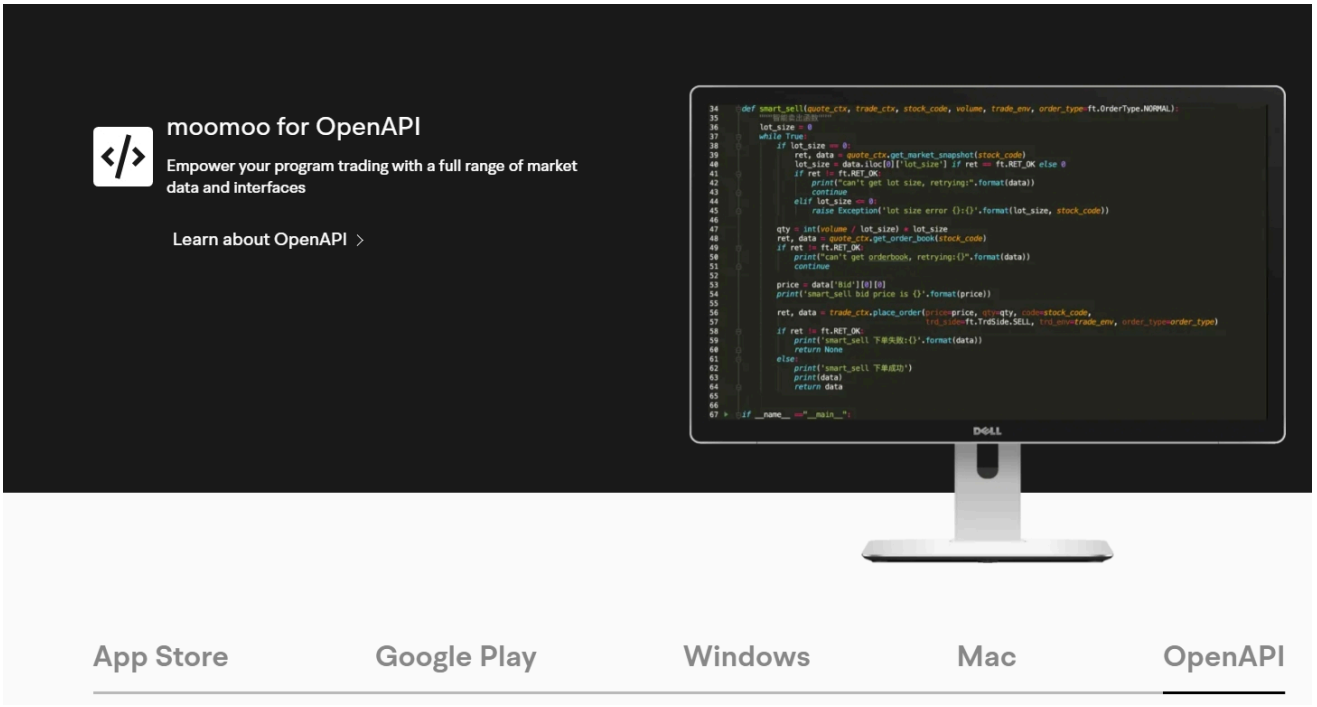
Please refer to **Command Line OpenD** for more informations for your interest.

Visualization OpenD

Step 1: Download

Visualization OpenD can be runned under 4 operating systems: Windows、MacOS、CentOS、Ubuntu.

- You can download through **moomoo official website**

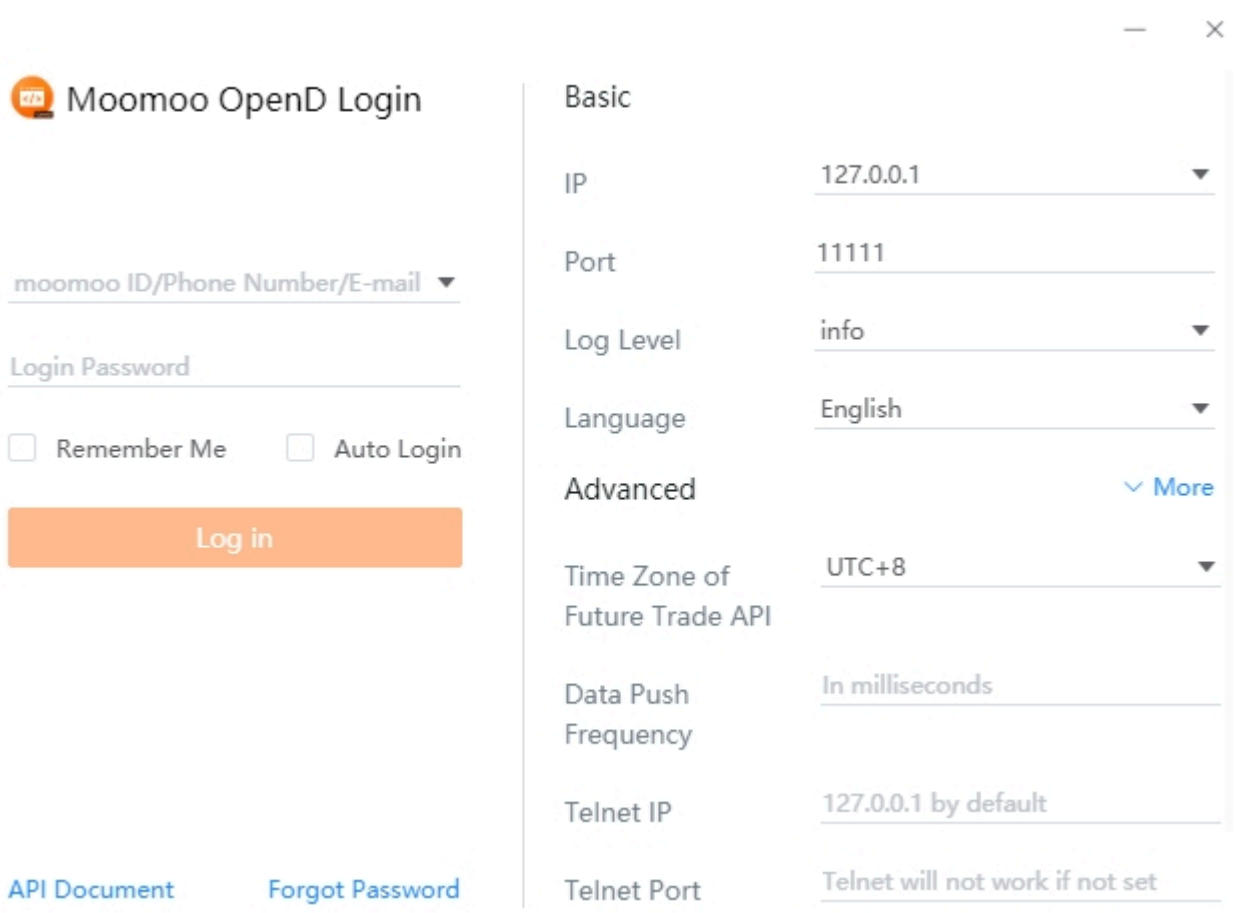


Step 2: Installation

- Extract the file and find the corresponding installation file to install OpenD.
- OpenD is installed in the `% appdata%` directory by default under Windows System.

Step 3: Configuration

- The Visualization OpenD launch configuration is on the right side of the graphical interface, as shown in the following figure:



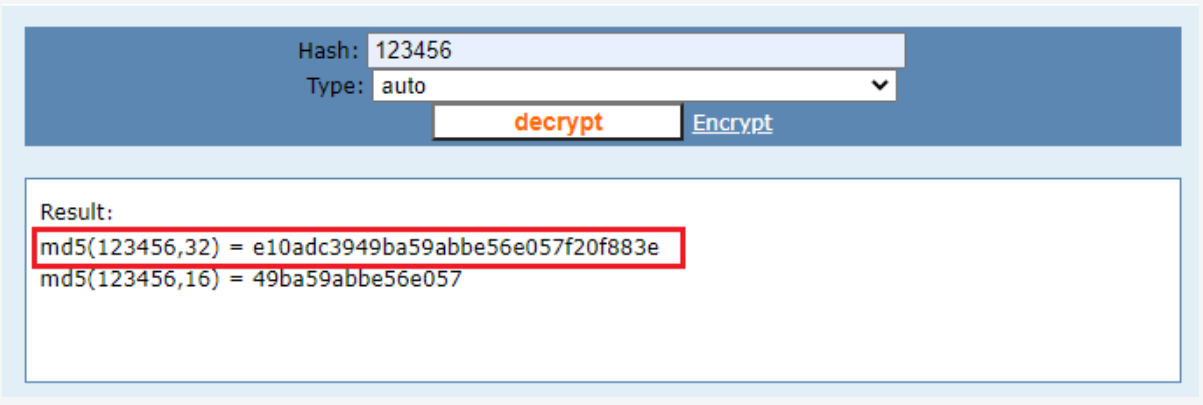
Configuration item list：

Configuration Item	Description
IP	API listening IP address.
Port	API listening port.
Log Level	Log level of OpenD.
Language	Language.
Time Zone of Future Trade API	Specify the futures trading API time zone.
Data Push Frequency	API subscription data push frequency control.
Telnet IP	Listening address of remote operation command.
Telnet Port	Listening port of remote operation command.

Configuration Item	Description
Encrypted Private Key	Absolute path of RSA Encrypted Private Key.
WebSocket IP	WebSocket listening address.
WebSocket Port	WebSocket listening port.
WebSocket Certificate	WebSocket certificate file path.
WebSocket Private Key	WebSocket certificate private key file path.
WebSocket Authentication Key	Cipher text of key (32-bit MD5 encrypted hexadecimal).

Tips

- Visual OpenD provides services by launching command line OpenD, interacted through WebSocket, so the WebSocket function must be started.
- To ensure safety of your trading accounts, if the listening address is not local, you must configure a private key to use the trading interface. The quote interface is not subject to this restriction.
- When the WebSocket listening address is not local, you need to configure SSL to start it, and a password should not be set during the certificate private key generation.
- Ciphertext is represented in hexadecimal after plaintext is encrypted by 32-bit MD5, which can be calculated by searching online MD5 encryption (note that there may be a risk of records colliding with libraries calculated through third-party websites) or by downloading MD5 computing tools. The 32-bit MD5 ciphertext is shown in the red box area (e10adc3949ba59abbe56e057f20f883e):



> - OpenD reads OpenD.xml in the same directory by default. On MacOS, due to the system protection mechanism, OpenD.app will be assigned a random path at run time, so that the original path can not be found. At this point, there are the following methods:
>
>   - Execute fixrun.sh under tar package
>   - Specify the configuration file path with the command line parameter `-cfg_file`, as described below
>
> - The log level defaults to the info level. During the system development phase, it is not recommended to close the log or modify the log to the warning, error, fatal level to prevent failure to locate problems.

Step 4: Login

- Enter your account number and password to login.
  You need to complete the questionnaire evaluation and agreement confirmation when you log in for the first time.
  You can see your account information and **quote right**, After logging in successfully.

Environment Setup

> **Notice**
>
> Ways of building programming environment are different for different programming languages.

Python Environment

Environment Requirement

- Operating system requirements:
  - 32-bit or 64-bit operating system of Windows 7/10
  - 64-bit operating system of Mac 10.11 and above
  - 64-bit operating system of CentOS 7 and above
  - 64-bit operating system of Ubuntu 16.04 and above
- Python version requirements:
  - Python 3.6 or above

Environment Building

1. Install Python

To avoid running failures due to environmental problems, we recommend Python version 3.8.

Download page: Download Python

▶ Tips

After the installation, execute the following command to see if the installation is successful: `python -V` (Windows) or `python3 -V` (Linux/Mac)

2. Install PyCharm (Optional)

We recommend that using PyCharm as your Python IDE.

3. Install TA-Lib (Optional)

TA-Lib is a functional library widely used in program trading for technical analysis of market data. It provides a variety of technical analysis functions to facilitate our quantitative investment.

Installation method: directly use pip installation in cmd

```
$ pip install TA-Lib
```

> **提示**
>
> - Installation of TA-Lib is not necessary, you can skip this step

Program Samples

Python Example

Step 1: Download and install OpenD

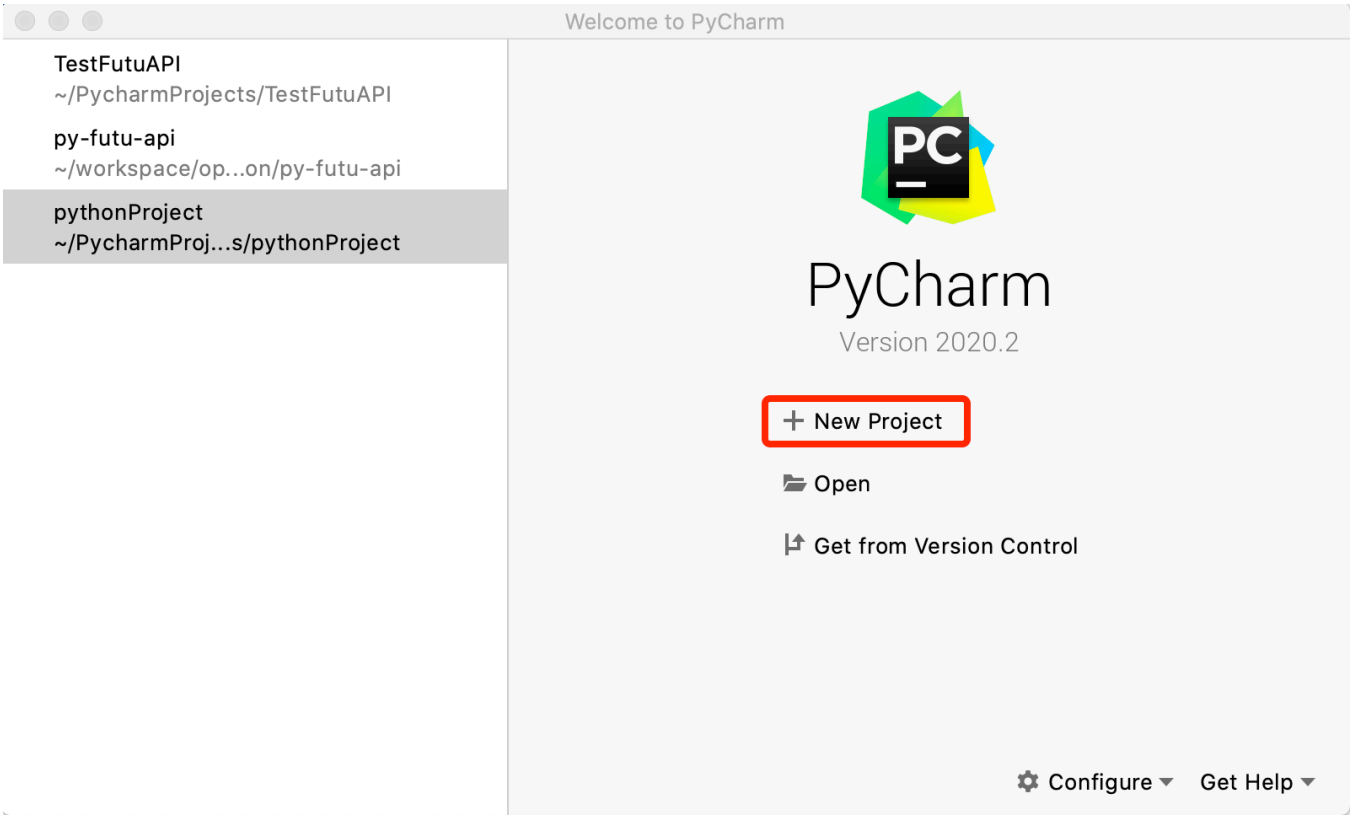
Please refer to here to finish downloading, installing and logging in OpenD.

Step 2: Download Python API

- Method 1: Use pip install in cmd.
  - Initial installation: Windows: `$ pip install moomoo-api`, Linux/Mac `$ pip3 install moomoo-api`.
  - Secondary upgrade: Windows: `$ pip install moomoo-api --upgrade`, Linux/Mac `$ pip3 install moomoo-api --upgrade`.
- Method 2: Download latest version of Python API from moomoo official website.

Step 3: Create New Project

Open PyCharm and click 'New Project' from 'Welcome to PyCharm' window. If you have already created a project, you can open the project directly.



Step 4: Create new file

Create new Python file under the project, and copy the sample code below to that file. The sample code includes viewing the market snapshot and placing an order through paper trading account.

```
from moomoo import *

quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)  # Create quote objec
print(quote_ctx.get_market_snapshot('HK.00700'))  # Get market snapshot for HK.0
quote_ctx.close() # Close object to prevent the number of connextions from runni


trd_ctx = OpenSecTradeContext(host='127.0.0.1', port=11111)  # Create trade obje
print(trd_ctx.place_order(price=500.0, qty=100, code="HK.00700", trd_side=TrdSid

trd_ctx.close()  # Close object to prevent the number of connextions from running
```

Step 5: Running file

Run the project, and you can see the returned message of a successful run as follows:

```
2020-11-05 17:09:29,705 [open_context_base.py] _socket_reconnect_and_wait_ready:2
2020-11-05 17:09:29,705 [open_context_base.py] on_connected:344: Connected : conn
2020-11-05 17:09:29,706 [open_context_base.py] _handle_init_connect:445: InitConn
(0,        code          update_time  last_price  open_price  high_price  ...  a
0  HK.00700  2020-11-05 16:08:06       625.0       610.0       625.0  ...

[1 rows x 132 columns])
2020-11-05 17:09:29,739 [open_context_base.py] _socket_reconnect_and_wait_ready:2
2020-11-05 17:09:29,739 [network_manager.py] work:366: Close: conn_id=1
2020-11-05 17:09:29,739 [open_context_base.py] on_connected:344: Connected : conn
2020-11-05 17:09:29,740 [open_context_base.py] _handle_init_connect:445: InitConn
(0,        code stock_name trd_side order_type order_status  ... dealt_avg_price
0  HK.00700       腾讯控股      BUY     NORMAL   SUBMITTING  ...             0.0

[1 rows x 16 columns])
2020-11-05 17:09:32,843 [network_manager.py] work:366: Close: conn_id=2
(0,        code stock_name trd_side      order_type order_status  ... dealt_avg_p
0  HK.00700       腾讯控股      BUY  ABSOLUTE_LIMIT    SUBMITTED  ...

[1 rows x 16 columns])
```

Strategy Setup

> **Tips**
>
> - The content of this trading strategy is not an investment advice. It is for learning purposes only.

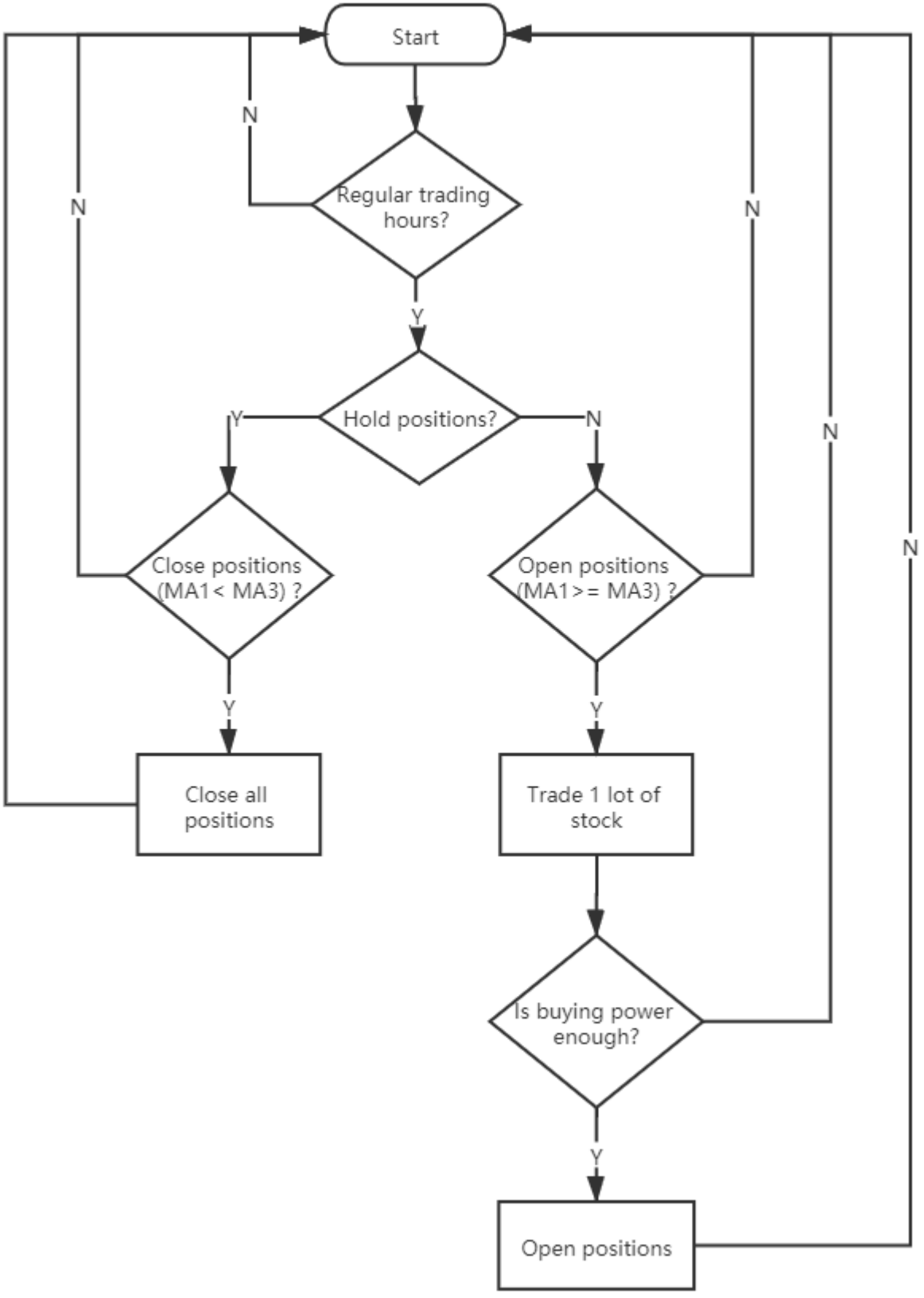
Strategy Introduction

Contruct a Double Moving Averaging Strategy.

That is, using the 1 minute candlestick of an underlying stock, to calculate two moving averages of different periods, MA1 and MA3. The values of MA1 and MA3 are tracked to determine the timing of buying and selling.

When MA1 >= MA3, the underlying stock is judged to be strong and the market is considered to be a bull market, which shows a long signal.
When MA1 < MA3, the underlying stock is judged to be weak and the market is considered to be a bear market, which shows a short signal.

Flow Chart



Code Sample

- **Example**

```python
from moomoo import *

############################ Global Variables ############################
MOOMOOOPEND_ADDRESS = '127.0.0.1'  # mooomoo OpenD listening address
MOOMOOOPEND_PORT = 11111  # mooomoo OpenD listening port

TRADING_ENVIRONMENT = TrdEnv.SIMULATE  # Trading environment: REAL / SIMULATE
TRADING_MARKET = TrdMarket.HK  # Transaction market authority, used to filter ac
TRADING_PWD = '123456'  # Trading password, used to unlock trading for real tradi
TRADING_PERIOD = KLType.K_1M  # Underlying trading time period
TRADING_SECURITY = 'HK.00700'  # Underlying trading security code
FAST_MOVING_AVERAGE = 1  # Parameter for fast moving average
SLOW_MOVING_AVERAGE = 3  # Parameter for slow moving average

quote_context = OpenQuoteContext(host=MOOMOOOPEND_ADDRESS, port=MOOMOOOPEND_PORT)
trade_context = OpenSecTradeContext(filter_trdmarket=TRADING_MARKET, host=MOOMOOO


# Unlock trade
def unlock_trade():
    if TRADING_ENVIRONMENT == TrdEnv.REAL:
        ret, data = trade_context.unlock_trade(TRADING_PWD)
        if ret != RET_OK:
            print('Unlock trade failed: ', data)
            return False
        print('Unlock Trade success!')
    return True


# Check if it is regular trading time for underlying security
def is_normal_trading_time(code):
    ret, data = quote_context.get_market_state([code])
    if ret != RET_OK:
        print('Get market state failed: ', data)
        return False
    market_state = data['market_state'][0]
    '''
    MarketState.MORNING            HK and A-share morning
    MarketState.AFTERNOON          HK and A-share afternoon, US opening hours
    MarketState.FUTURE_DAY_OPEN    HK, SG, JP futures day market open
    MarketState.FUTURE_OPEN        US futures open
    MarketState.FUTURE_BREAK_OVER  Trading hours of U.S. futures after break
    MarketState.NIGHT_OPEN         HK, SG, JP futures night market open
    '''
```

```
    if market_state == MarketState.MORNING or \
                    market_state == MarketState.AFTERNOON or \
                    market_state == MarketState.FUTURE_DAY_OPEN  or \
                    market_state == MarketState.FUTURE_OPEN  or \
                    market_state == MarketState.FUTURE_BREAK_OVER  or \
                    market_state == MarketState.NIGHT_OPEN:
        return True
    print('It is not regular trading hours.')
    return False


# Get positions
def get_holding_position(code):
    holding_position = 0
    ret, data = trade_context.position_list_query(code=code, trd_env=TRADING_ENVI
    if ret != RET_OK:
        print('Get holding position failed: ', data)
        return None
    else:
        for qty in data['qty'].values.tolist():
            holding_position += qty
        print('[Holding Position Status] The holding position quantity of {} is:
    return holding_position



# Query for candlesticks, calculate moving average value and judge bull or bear
def calculate_bull_bear(code, fast_param, slow_param):
    if fast_param <= 0 or slow_param <= 0:
        return 0
    if fast_param > slow_param:
        return calculate_bull_bear(code, slow_param, fast_param)
    ret, data = quote_context.get_cur_kline(code=code, num=slow_param + 1, ktype=
    if ret != RET_OK:
        print('Get candlestick value failed: ', data)
        return 0
    candlestick_list = data['close'].values.tolist()[::-1]
    fast_value = None
    slow_value = None
    if len(candlestick_list) > fast_param:
        fast_value = sum(candlestick_list[1: fast_param + 1]) / fast_param
    if len(candlestick_list) > slow_param:
        slow_value = sum(candlestick_list[1: slow_param + 1]) / slow_param
    if fast_value is None or slow_value is None:
        return 0
```

```
    return 1 if fast_value >= slow_value else -1


# Get ask1 and bid1 from order book
def get_ask_and_bid(code):
    ret, data = quote_context.get_order_book(code, num=1)
    if ret != RET_OK:
        print('Get order book failed: ', data)
        return None, None
    return data['Ask'][0][0], data['Bid'][0][0]


# Open long positions
def open_position(code):
    # Get order book data
    ask, bid = get_ask_and_bid(code)

    # Get quantity
    open_quantity = calculate_quantity()

    # Check whether buying power is enough
    if is_valid_quantity(TRADING_SECURITY, open_quantity, ask):
        # Place order
        ret, data = trade_context.place_order(price=ask, qty=open_quantity, code=
                                              order_type=OrderType.NORMAL, trd_en
                                              remark='moving_average_strategy')
        if ret != RET_OK:
            print('Open position failed: ', data)
    else:
        print('Maximum quantity that can be bought less than transaction quantity


# Close position
def close_position(code, quantity):
    # Get order book data
    ask, bid = get_ask_and_bid(code)

    # Check quantity
    if quantity == 0:
        print('Invalid order quantity.')
        return False

    # Close position
    ret, data = trade_context.place_order(price=bid, qty=quantity, code=code, tr
                    order_type=OrderType.NORMAL, trd_env=TRADING_ENVIRONMENT, rema
```

```
    if ret != RET_OK:
        print('Close position failed: ', data)
        return False
    return True


# Calculate order quantity
def calculate_quantity():
    price_quantity = 0
    # Use minimum lot size
    ret, data = quote_context.get_market_snapshot([TRADING_SECURITY])
    if ret != RET_OK:
        print('Get market snapshot failed: ', data)
        return price_quantity
    price_quantity = data['lot_size'][0]
    return price_quantity


# Check the buying power is enough for the quantity
def is_valid_quantity(code, quantity, price):
    ret, data = trade_context.acctradinginfo_query(order_type=OrderType.NORMAL, c
                                                   trd_env=TRADING_ENVIRONMENT)
    if ret != RET_OK:
        print('Get max long/short quantity failed: ', data)
        return False
    max_can_buy = data['max_cash_buy'][0]
    max_can_sell = data['max_sell_short'][0]
    if quantity > 0:
        return quantity < max_can_buy
    elif quantity < 0:
        return abs(quantity) < max_can_sell
    else:
        return False


# Show order status
def show_order_status(data):
    order_status = data['order_status'][0]
    order_info = dict()
    order_info['Code'] = data['code'][0]
    order_info['Price'] = data['price'][0]
    order_info['TradeSide'] = data['trd_side'][0]
    order_info['Quantity'] = data['qty'][0]
    print('[OrderStatus]', order_status, order_info)

```

```python

############################ Fill in the functions below to finish your trading s
# Strategy initialization. Run once when the strategy starts
def on_init():
    # unlock trade (no need to unlock for paper trading)
    if not unlock_trade():
        return False
    print('************  Strategy Starts ***********')
    return True


# Run once for each tick. You can write the main logic of the strategy here
def on_tick():
    pass


# Run once for each new candlestick. You can write the main logic of the strategy
def on_bar_open():
    # Print seperate line
    print('****************************************')

    # Only trade during regular trading hours
    if not is_normal_trading_time(TRADING_SECURITY):
        return

    # Query for candlesticks, and calculate moving average value
    bull_or_bear = calculate_bull_bear(TRADING_SECURITY, FAST_MOVING_AVERAGE, SLO

    # Get positions
    holding_position = get_holding_position(TRADING_SECURITY)

    # Trading signals
    if holding_position == 0:
        if bull_or_bear == 1:
            print('[Signal] Long signal. Open long positions.')
            open_position(TRADING_SECURITY)
        else:
            print('[Signal] Short signal. Do not open short positions.')
    elif holding_position > 0:
        if bull_or_bear == -1:
            print('[Signal] Short signal. Close positions.')
            close_position(TRADING_SECURITY, holding_position)
        else:
            print('[Signal] Long signal. Do not add positions.')

```

```python

# Run once when an order is filled
def on_fill(data):
    pass


# Run once when the status of an order changes
def on_order_status(data):
    if data['code'][0] == TRADING_SECURITY:
        show_order_status(data)


################################ Framework code, which can be ignored ############
class OnTickClass(TickerHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        on_tick()


class OnBarClass(CurKlineHandlerBase):
    last_time = None
    def on_recv_rsp(self, rsp_pb):
        ret_code, data = super(OnBarClass, self).on_recv_rsp(rsp_pb)
        if ret_code == RET_OK:
            cur_time = data['time_key'][0]
            if cur_time != self.last_time and data['k_type'][0] == TRADING_PERIOD
                if self.last_time is not None:
                    on_bar_open()
                self.last_time = cur_time


class OnOrderClass(TradeOrderHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret, data = super(OnOrderClass, self).on_recv_rsp(rsp_pb)
        if ret == RET_OK:
            on_order_status( data)


class OnFillClass(TradeDealHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret, data = super(OnFillClass, self).on_recv_rsp(rsp_pb)
        if ret == RET_OK:
            on_fill(data)


# Main function
```

```
if __name__ == '__main__':
    # Strategy initialization
    if not on_init():
        print('Strategy initialization failed, exit script!')
        quote_context.close()
        trade_context.close()
    else:
        # Set up callback functions
        quote_context.set_handler(OnTickClass())
        quote_context.set_handler(OnBarClass())
        trade_context.set_handler(OnOrderClass())
        trade_context.set_handler(OnFillClass())

        # Subscribe tick-by-tick, candlestick and order book of the underlying tr
        quote_context.subscribe(code_list=[TRADING_SECURITY], subtype_list=[SubTy
```

- Output

```
************  Strategy Starts ***********
****************************************
[Position] The position of HK.00700 is 0
[Signal] Long signal. Open long positions.
[OrderStatus] SUBMITTING {'Code': 'HK.00700', 'Price': 597.5, 'TradeSide': 'BUY'
[OrderStatus] SUBMITTED {'Code': 'HK.00700', 'Price': 597.5, 'TradeSide': 'BUY',
[OrderStatus] FILLED_ALL {'Code': 'HK.00700', 'Price': 597.5, 'TradeSide': 'BUY'
****************************************
[Position] The position of HK.00700 is 100.0
[Signal] Short signal. Close positions.
[OrderStatus] SUBMITTING {'Code': 'HK.00700', 'Price': 596.5, 'TradeSide': 'SELL
[OrderStatus] SUBMITTED {'Code': 'HK.00700', 'Price': 596.5, 'TradeSide': 'SELL'
[OrderStatus] FILLED_ALL {'Code': 'HK.00700', 'Price': 596.5, 'TradeSide': 'SELL
```

Overview

- OpenD, which can be runned on your local computer or cloud server, is the gateway program of moomoo API. It is responsible for transferring protocol requests to moomoo servers and returning the processed data. It is a necessary prerequisite for running moomoo API programs.
- OpenD can be runned under 4 operating systems: Windows, MacOS, CentOS and Ubuntu.
- You need to log in to OpenD with your *moomoo ID*, *Email*, *Phone number* and *login password*.
- After a successful login into OpenD, the socket service is started for moomoo API to connect and communicate.

Running Mode

There are 2 modes to run OpenD, you can choose 1 of them below:

- Visualisation OpenD: Provide interface applications, easy to operate, especially suitable for beginners. Please refer to **Visualization OpenD** for installation and operation.
- Command Line OpenD: Provide command line execution program, which needs to be configured by yourself, which is suitable for users who are familiar with the command line or running on the server for a long time. Please refer to **Command Line OpenD** for installation and operation.

Operation While Running

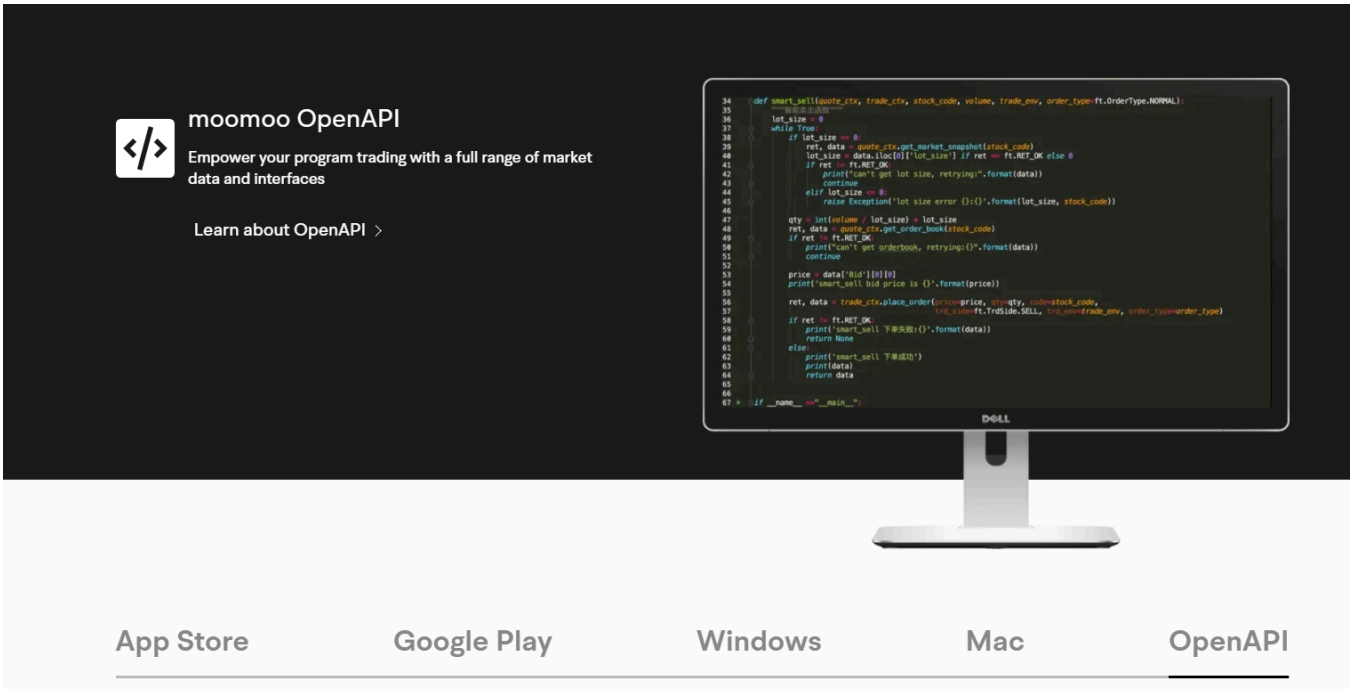
While OpenD is running, you can view user quota, quote right, connection status, delay statistics, and operate closing API connection, re-login, logging out etc. with Operation Command.
For more information, please see the following table:

Method	Visualisation OpenD	Command Line OpenD
Direct Method	through the UI interface	Send **Operation Command** through command line
Indirect Medhod	Send **Operation Command** through Telnet	Send **Operation Command** through Telnet

Command Line OpenD

Step 1: Download

- You can download through moomoo official website.



Step 2: Decompression

- Extract the file downloaded in the previous step and find the OpenD configuration file OpenD.xml and the program packaged data file Appdata.dat in the folder.
  - OpenD.xml is used to configure the startup parameters of the OpenD program. If it does not exist, the program cannot start correctly.
  - Appdata.dat is a large amount of data information the program needs to use, packaging data to reduce the time of downloading data while starting OpenD. If it does not exist, the program can not start correctly.
- Command line OpenD supports user-defined file paths, refer to Command line startup parameters。

Step 3: Parameter Configuration

- Open and edit the configuration file OpenD.xml as the picture below. For general use, you only need to change your account and login password, and other options can be modified

according to the instructions in the following table.

```
<moomoo_opend>
    <!-- 基础参数 -->
    <!-- Basic parameters -->
        <!-- 协议监听地址,不填默认127.0.0.1 -->
        <!-- Listening address. 127.0.0.1 by default -->
        <ip>127.0.0.1</ip>
        <!-- API接口协议监听端口 -->
        <!-- API interface protocol listening port -->
        <api_port>11111</api_port>
        <!-- 登录帐号 -->
        <!-- Login account -->
        <login_account>100000</login_account>
        <!-- 登录密码32位MD5加密16进制 -->
        <!-- Login password, 32-bit MD5 encrypted hexadecimal -->
        <!-- <login_pwd_md5>6e55f158a827b1a1c4321a245aaaad88</login_pwd_md5> -->
        <!-- 登录密码明文，密码密文存在情况下只使用密文 -->
        <!-- Plain text of login password. When cypher text exists, the cypher text will be used. -->
        <login_pwd>123456</login_pwd>
        <!-- FutuOpenD语言，en: 英文，chs: 简体中文 -->
        <!-- FutuOpenD language. en: English, chs: Simplified Chinese -->
        <lang>chs</lang>
    <!-- 进阶参数 -->
    <!-- Advanced parameters -->
        <!-- FutuOpenD日志等级，no, debug, info, warning, error, fatal -->
        <!-- FutuOpenD log level: no, debug, info, warning, error, fatal -->
        <log_level>info</log_level>
        <!-- API推送协议格式，0: pb, 1: json -->
        <!-- API push protocol format. 0: pb, 1: json -->
        <push_proto_type>0</push_proto_type>
        <!-- API订阅数据推送频率控制，单位毫秒，目前不包括K线和分时，不设置则不限制频率-->
        <!-- Data Push Frequency, in milliseconds. Candlesticks and timeframes are not included. If not set, t
        <!-- <qot_push_frequency>1000</qot_push_frequency> -->
        <!-- Telnet监听地址,不填默认127.0.0.1 -->
        <!-- Telnet listening address. 127.0.0.1 by default -->
        <!-- <telnet_ip>127.0.0.1</telnet_ip> -->
        <!-- Telnet监听端口 -->
        <!-- Telnet listening port -->
        <!-- <telnet_port>22222</telnet_port> -->
        <!-- API协议加密私钥文件路径,不设置则不加密 -->
        <!-- File path for private key for API protocol enctyption. If not set, it will not be encrypted. -->
        <!-- <rsa_private_key>D:\rsa</rsa_private_key> -->
        <!-- 是否接收到价提醒推送，0: 不接收，1: 接收 -->
        <!-- Whether to receive the price reminder push. 0: not receive, 1: receive -->
        <price_reminder_push>1</price_reminder_push>
        <!-- 被踢后是否自动抢权限，0: 否，1: 是 -->
        <!-- 开启该选项后，要想在其他终端上获取行情权限需要在10秒连续踢FutuOpenD两次行情权限（终端登录算一次）
        <!-- Whether to automatically grab highest quote right after being kicked, 0: No, 1: Yes -->
        <!-- When this parameter is set as 1, if you want to get highest quote right on APP, you need to kick
```

Configuration item list：

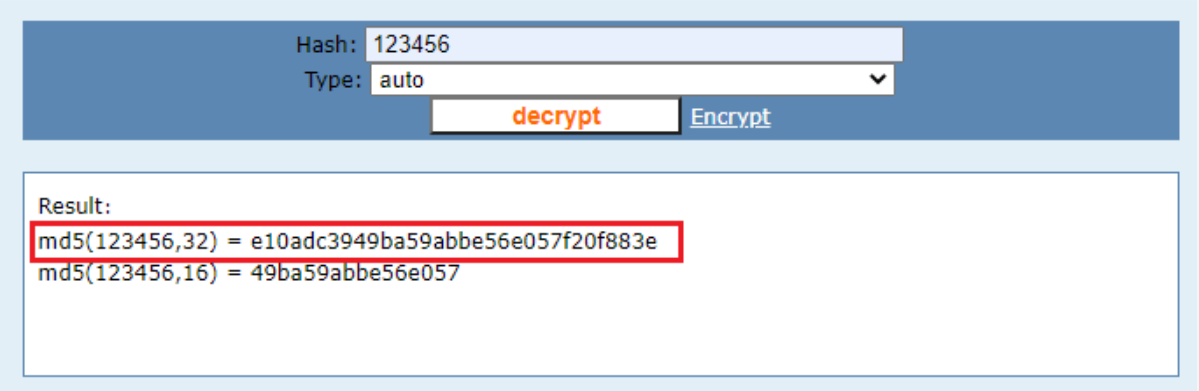
Configuration Item	Description
ip	listening address.
api_port	API protocol receiving port.
login_account	Login account.
login_pwd	Login password in plaintext.
login_pwd_md5	Login password ciphertext (32-bit MD5 encrypted hexadecimal).
Lang	Language.

Configuration Item	Description
log_level	Log level of OpenD.
push_proto_type	API protocol type.
qot_push_frequency	API subscription data push frequency
telnet_ip	Remote operation command listening address.
telnet_port	Remote operation command listening port.
rsa_private_key	API protocol RSA encrypted private key (PKCS#1) file absolute path.
price_reminder_push	Whether to receive the price reminder.
auto_hold_quote_right	Whether to automatically grab quote right after being kicked.
future_trade_api_time_zone	Specify the futures trading API time zone.
websocket_ip	WebSocket listening address.
websocket_port	WebSocket service listening port.
websocket_key_md5	Key ciphertext (32-bit MD5 encrypted hexadecimal).
websocket_private_key	WebSocket certificate private key file path.
websocket_cert	WebSocket certificate file path.
pdt_protection	Whether to turn on the Pattern Day Trade Protection.
dtcall_confirmation	Whether to turn on the Day-Trading Call Warning.

Tips

- To ensure safety of your trading accounts, if the listening address is not local, you must configure a private key to use the trading interface. The quote interface is not subject to this restriction.
- When the WebSocket listening address is not local, you need to configure SSL to start it, and a password should not be set during the certificate private key generation.

- Ciphertext is represented in hexadecimal after plaintext is encrypted by 32-bit MD5, which can be calculated by searching online MD5 encryption (note that there may be a risk of records colliding with libraries calculated through third-party websites) or by downloading MD5 computing tools. The 32-bit MD5 ciphertext is shown in the red box area (e10adc3949ba59abbe56e057f20f883e):



- OpenD reads OpenD.xml in the same directory by default. On MacOS, due to the system protection mechanism, OpenD.app will be assigned a random path at run time, so that the original path can not be found. At this point, there are the following methods:
  - Execute fixrun.sh under tar package
  - Specify the configuration file path with the command line parameter `-cfg_file`, as described below
- The log level defaults to the info level. During the system development phase, it is not recommended to close the log or modify the log to the warning, error, fatal level to prevent failure to locate problems.

Step 4: Command Line Startup

- On the command line, change the directory to the folder which OpenD is located, and run the following command to start Command Line OpenD with configuration from OpenD.xml.
  - Windows: `OpenD`
  - Linux: `./OpenD`
  - MacOS: `./OpenD.app/Contents/MacOS/OpenD`

▶ Command Line Startup Parameters

- If the same parameters exist on both the command line and the configuration file, the command line parameters take precedence. For details of the parameters, please see the following table:

parameter list:

Configuration Item	Description
login_account	Login account.
login_pwd	Plaintext of login password.
login_pwd_md5	Login password ciphertext (32-bit MD5 encrypted hexadecimal).
cfg_file	The absolute path of OpenD configuration file.
console	Whether to display the console.
lang	OpenD language
api_ip	API service listening address.
api_port	API listening port.
help	Output startup command line parameters and exit the program.
log_level	Log level of OpenD.
no_monitor	Whether to start the daemon.
websocket_ip	WebSocket listening address.
websocket_port	WebSocket service listening port.
websocket_private_key	WebSocket certificate private key file path.
websocket_cert	WebSocket certificate file path.
websocket_key_md5	Key ciphertext (32-bit MD5 encrypted hexadecimal).
price_reminder_push	Whether to receive the price reminder.
auto_hold_quote_right	Whether to automatically grab quote right after being kicked.

Configuration Item	Description
future_trade_api_time_zone	Specify the futures *Trade API* time zone. 

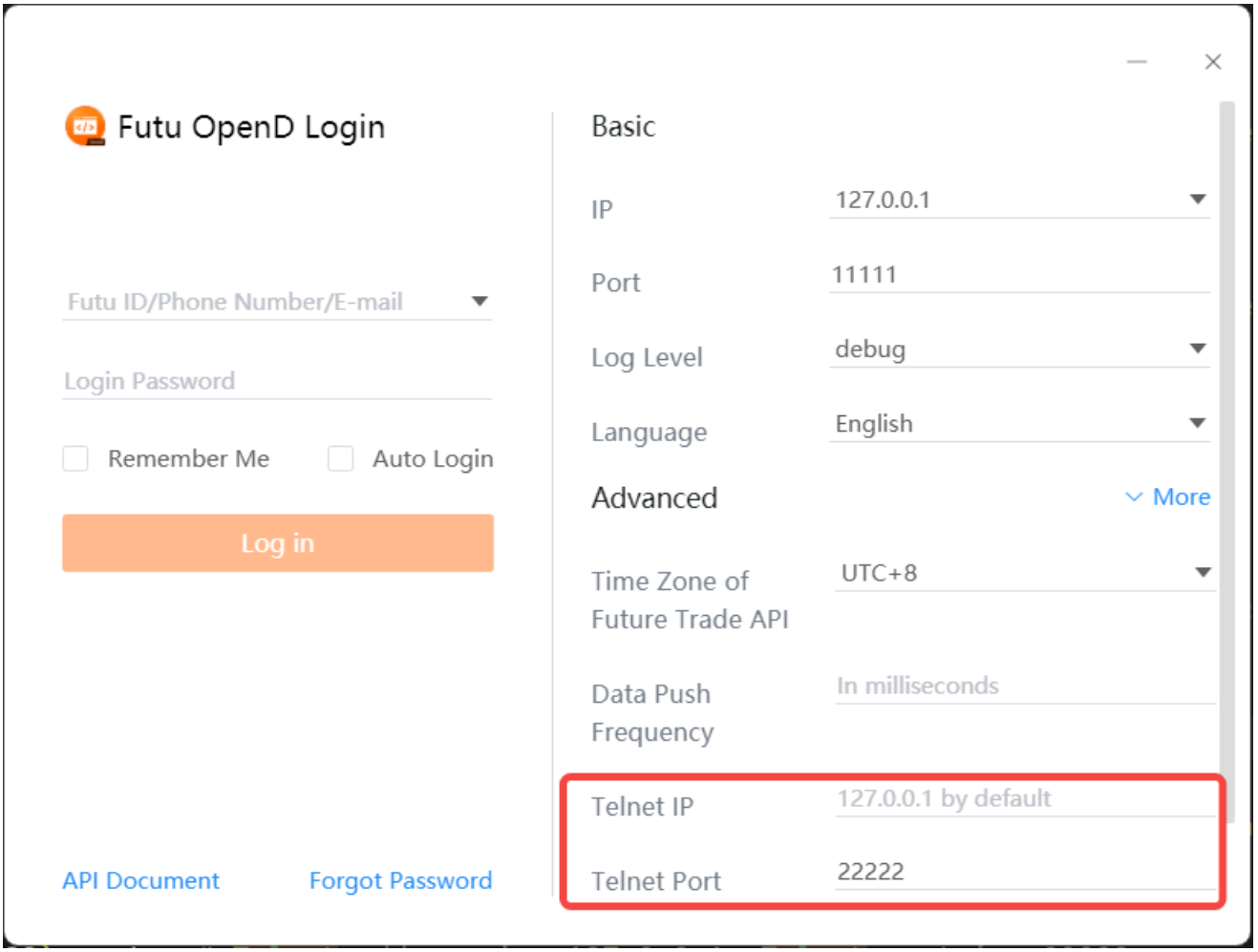
:::

Operation Command

You can do operate OpenD by sending Operation Command from the command line or Telent.

Command format: `cmd -param_key1=param_value1 -param_key2=param_value2`
Using the following example to describe how to use Telnet: `help -cmd=exit`

1. Configure Telnet address and Telnet port in the OpenD set up parameter.



FutuOpenD.xml

```
<futu_opend>
    <!-- 基础参数 -->
    <!-- Basic parameters -->
        <!-- 协议监听地址,不填默认127.0.0.1 -->
        <!-- Listening address. 127.0.0.1 if not specified -->  // Listening address. 127.0.0.1 by default
        <ip>127.0.0.1</ip>
        <!-- API接口协议监听端口 -->
        <!-- API interface protocol listening port -->
        <api_port>11111</api_port>
        <!-- 登录帐号 -->
        <login_account>100000</login_account>
        <!-- 登录密码32位MD5加密16进制 -->
        <!-- <login_pwd_md5>6e55f158a827b1a1c4321a245aaaad88</login_pwd_md5> -->
        <!-- 登录密码明文，密码密文存在情况下只使用密文 -->
        <login_pwd>123456</login_pwd>
        <!-- FutuOpenD语言，en：英文，chs：简体中文 -->
        <lang>chs</lang>
    <!-- 进阶参数 -->
    <!-- Advanced parameters -->
        <!-- FutuOpenD日志等级，no,debug,info,warning,error,fatal -->
        <log_level>info</log_level>
        <!-- API推送协议格式，0:pb, 1:json -->
        <!-- API push protocol format, 0:pb, 1:json -->
        <push_proto_type>0</push_proto_type>
        <!-- API订阅数据推送频率控制，单位毫秒，目前不包括K线和分时，不设置则不限制频率-->
        <!-- API subscription data push frequency control, in milliseconds, currently does not include K-line and time-sharing, if not
        <!-- <qot_push_frequency>1000</qot_push_frequency> -->
        <!-- Telnet监听地址,不填默认127.0.0.1 -->
        <!-- Telnet listening address, default 127.0.0.1 if not filled in -->  // Telnet listening address, 127.0.0.1 by default
        <telnet_ip>127.0.0.1</telnet_ip>
        <!-- Telnet监听端口 -->
        <!-- Telnet listening port -->
        <telnet_port>22222</telnet_port>
        <!-- API协议加密私钥文件路径,不设置则不加密 -->
        <!-- API protocol encrypted private key file path, if not set, it will not be encrypted -->  // File path for private key for
        <!-- <rsa_private_key>D:\rsa</rsa_private_key> -->
        <!-- 是否接收到价提醒推送，0：不接收，1：接收 -->
        <!-- Whether to receive the price reminder push, 0: not receive, 1: receive -->
```

2. Start OpenD (it will also start Telnet).
3. Via Telnet，send the command `help -cmd=exit` to OpenD。

```py
from telnetlib import Telnet
with Telnet('127.0.0.1', 22222) as tn:  # Telnet address is: 127.0.0.1, Telnet p
    tn.write(b'help -cmd=exit\r\n')
    reply = b''
    while True:
        msg = tn.read_until(b'\r\n', timeout=0.5)
        reply += msg
        if msg == b'':
            break
    print(reply.decode('gb2312'))
```

Command Help

`help -cmd=exit`

View the detailed information of the specified command, output the command list if no parameter is specified

- Parameters:
  - cmd: command

Exit the Program

`Exit`

Exit OpenD

Request Mobile Phone Verification Code

`req_phone_verify_code`

Requested mobile phone verification code. Security verification is required when the device lock is enabled and the device is logged in at the first time.

- Frequency limitations:
  - Maximal 1 request every 60 seconds

Enter the Phone Verification Code

`Input_phone_verify_code -code=123456`

Enter the phone verification code and continue the login process.

- Parameters:
  - code: mobile phone verification code
- Frequency limitations:
  - Maximal 10 requests every 60 seconds

Request Graphic Verification Code

`req_pic_verify_code`

Request a graphic verification code. When you enter the wrong login password multiple times, you need to enter the graphic verification code.

- Frequency limitations:

- Maximal 10 requests every 60 seconds

Enter Graphic Verification Code

```
Input_pic_verify_code -code=1234
```

Enter the graphic verification code and continue the login process.

- Parameters:
  - code: Graphic verification code
- Frequency limitations:
  - Maximal 10 requests every 60 seconds

Relogin

```
relogin -login_pwd=123456
```

This command can be used when the user is required to log in again when the login password is changed or the device lock is opened midway. You can only relogin to the current account, and changing accounts is not supported. The password parameter is mainly used to the situation that the login password had been modified. If login_pwd is not set, the login password at startup will be used.

- Parameters:
  - login_pwd: login password in plaintext
  - login_pwd_md5: login password in ciphertext (32-bit MD5 encrypted hexadecimal)
- Frequency limitations:
  - Maximal 10 requests every hour

Time Delay Between Detection and Connection Point

```
ping
```

Delay before detection and connection point

- Frequency limitations:
  - Maximal 10 requests every 60 seconds

Display Delay Statistics Report

```
show_delay_report -detail_report_path=D:/detail.txt -push_count_type=sr2cs
```

Display delay statistics report, including push delay, request delay and order delay. Data is cleaned up at 6:00 Beijing time every day.

- Parameters:
  - detail_report_path: file output path (MAC system only supports absolute path, not relative path), optional parameter, if not specified, output to the console
  - push_count_type: the type of push delay (sr2ss, ss2cr, cr2cs, ss2cs, sr2cs), sr2cs by default.
    - sr refers to the server receiving time (currently only HK stocks support this time)
    - ss refers to the server sending time
    - cr refers to OpenD receiving time
    - cs refers to OpenD sending time

Close API Connection

```
close_api_conn -conn_id=123456
```

Close an API connection, if not specified, close all connections

- Parameters:
  - conn_id: API connection ID

Show Subscription Status

```
show_sub_info -conn_id=123456 -sub_info_path=D:/detail.txt
```

Display the subscription status of a connection, if not specified, display all connections

- Parameters:
  - conn_id: API connection ID

- sub_info_path: file output path (MAC system only supports absolute path, not relative path), optional parameter, if not specified, output to the console

Request the Highest Quotation Permission

`request_highest_quote_right`

When the advanced quotation authority is occupied by other devices (such as desktop/mobile terminal), you can use this command to request the highest quotation authority again (And then, other devices that are logged in will not be able to use advanced quote).

- Frequency limitations:
  - Maximal 10 requests every 60 seconds

Update

`update`

Update

Overview

Module		Interface Name	Function Description
Real-time Data	Subscription	subscribe	Subscribe real-time market data
		unsubscribe	Unsubscribe subscriptions
		unsubscribe_all	Unsubscribe all subscriptions
		query_subscription	Get subscription information
	Push and Callback	StockQuoteHandlerBase	Real-time quote callback
		OrderBookHandlerBase	Real-time order book callback
		CurKlineHandlerBase	Real-time candlestick callback
		TickerHandlerBase	Real-time tick-by-tick callback
		RTDataHandlerBase	Real-time time frame callback
		BrokerHandlerBase	Real-time broker queue callback
	Get	get_market_snapshot	Get market snapshot
		get_stock_quote	Get real-time quote
		get_order_book	Get real-time order book
		get_cur_kline	Get real-time candlestick
		get_rt_data	Get real-time time frame data
		get_rt_ticker	Get real-time tick-by-tick

		get_broker_queue	Get real-time broker queue
Basic Data		get_market_state	Get market status of securities
		get_capital_flow	Get capital flow
		get_capital_distribution	Get capital distribution
		get_owner_plate	Get the stock ownership plate
		request_history_kline	Get historical candlesticks
		get_rehab	Get the stock adjustment factor
Related Derivatives		get_option_expiration_date	Query all expiration dates of option chains through the underlying stock.
		get_option_chain	Get the option chain from an underlying stock
		get_warrant	Get filtered warrant (for HK market only)
		get_referencestock_list	Get related data of securities
		get_future_info	Get futures contract information
Market Filter		get_stock_filter	Filter stocks by condition
		get_plate_stock	Get the list of stocks in the plate
		get_plate_list	Get plate list
		get_stock_basicinfo	Get stock basic information
		get_ipo_list	Get IPO information of a specific market
		get_global_state	Get global status

	request_trading_days	Get trading calendar
Customization	get_history_kl_quota	Get usage details of historical candlestick quota
	set_price_reminder	Add, delete, modify, enable, and disable price reminders for specified stocks
	get_price_reminder	Get a list of price reminders set for the specified stock or market
	get_user_security_group	Get a list of groups from the user watchlist
	get_user_security	Get a list of a specified group from watchlist
	modify_user_security	Modify the specific group from the watchlist
	PriceReminderHandlerBase	The price reminder notification callback

Quote Object

Create and Initialize the Connection

`OpenQuoteContext(host='127.0.0.1', port=11111, is_encrypt=None)`

- **Introduction**

  Create and initialize market connection

- **Parameters**

Parameter	Type	Description
host	str	OpenD listening address.
port	int	OpenD listening port.
is_encrypt	bool	Whether to enable encryption.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111, is_encrypt=False)
quote_ctx.close() # After using the connection, remember to close it to prevent
```

Close Connection

`close()`

- **Introduction**

Close the interface quotation object. By default, the threads created inside the moomoo API will prevent the process from exiting, and the process can exit normally only after all Contexts

are closed. But through set_all_thread_daemon, all internal threads can be set as daemon threads. At this time, even if the close of Context is not called, the process can exit normally.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

Start-up

`start()`

- **Introduction**

  Start to receive push data asynchronously

Stop

`stop()`

- **Introduction**

  Stop receiving push data asynchronously

Subscribe and Unsubscribe

subscribe

```
subscribe(code_list, subtype_list, is_first_push=True, subscribe_push=True, is_detailed_orderbook=False, extended_time=False, session=Session.NONE)
```

- **Description**

  To subscribe to the real-time information required for registration, specify the stock and subscription data types.
  HK market (including underlying stocks, warrants, CBBCs, options, futures) subscriptions require LV1 and above permissions. Subscriptions are not supported under BMP permissions.
  US market (including underlying stocks, ETFs) subscriptions for overnight quotes require LV1 and above permissions. Subscriptions are not supported under BMP permissions.

- **Parameters**

Parameter	Type	Description
code_list	list	A list of stock codes that need to be subscribed.
subtype_list	list	List of data types that need to be subscribed.
is_first_push	bool	Whether to push the cached data immediately after a successful subscription.
subscribe_push	bool	Whether to push data after subscription.
is_detailed_orderbook	bool	Whether to subscribe to the detailed order book.
extended_time	bool	Whether to allow pre-market and after-hours data of US stocks.

Parameter	Type	Description
session	Session	US stocks quotes session

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
err_message	NoneType	If ret == RET_OK, None is returned.
	str	If ret != RET_OK, error description is returned.

- Example

```
import time
from moomoo import *
class OrderBookTest(OrderBookHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, data = super(OrderBookTest,self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("OrderBookTest: error, msg: %s"% data)
            return RET_ERROR, data
        print("OrderBookTest ", data) # OrderBookTest's own processing logic
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = OrderBookTest()
quote_ctx.set_handler(handler) # Set real-time swing callback
quote_ctx.subscribe(['US.AAPL'], [SubType.ORDER_BOOK]) # Subscribe to the order
time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
quote_ctx.close() # Close the current link, OpenD will automatically cancel the
```

- Output

```
OrderBookTest  {'code': 'US.AAPL', 'name': 'Apple', 'svr_recv_time_bid': '2025-04
```

unsubscribe

`unsubscribe(code_list, subtype_list, unsubscribe_all=False)`

- Description

  unsubscribe

- Parameters

Parameter	Type	Description
code_list	list	A list of stock codes to unsubscribe.
subtype_list	list	List of data types that need to be subscribed.
unsubscribe_all	bool	Cancel all subscriptions.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
err_message	NoneType	If ret == RET_OK, None is returned.
	str	If ret != RET_OK, error description is returned.

- Example

```
from moomoo import *
import time
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

print('current subscription status :', quote_ctx.query_subscription()) # Query th
ret_sub, err_message = quote_ctx.subscribe(['US.AAPL'], [SubType.QUOTE, SubType.T
# First subscribed to the two types of QUOTE and TICKER. After the subscription
if ret_sub == RET_OK: # Subscription successful
    print('subscribe successfully! current subscription status :', quote_ctx.quer
    time.sleep(60) # You can unsubscribe at least 1 minute after subscribing
    ret_unsub, err_message_unsub = quote_ctx.unsubscribe(['US.AAPL'], [SubType.QU
    if ret_unsub == RET_OK:
        print('unsubscribe successfully! current subscription status:', quote_ctx
    else:
```

```
        print('unsubscription failed!', err_message_unsub)
else:
    print('subscription failed', err_message)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
current subscription status : (0, {'total_used': 0, 'remain': 1000, 'own_used': 0
subscribe successfully! current subscription status : (0, {'total_used': 2, 'rema
unsubscribe successfully! current subscription status: (0, {'total_used': 1, 'rem
```

unsubscribe_all

`unsubscribe_all()`

- Description

  Unsubscribe all subscriptions

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
err_message	NoneType	If ret == RET_OK, None is returned.
	str	If ret != RET_OK, error description is returned.

- Example

```
from moomoo import *
import time
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

print('current subscription status :', quote_ctx.query_subscription()) # Query th
ret_sub, err_message = quote_ctx.subscribe(['US.AAPL'], [SubType.QUOTE, SubType.T
# First subscribed to the two types of QUOTE and TICKER. After the subscription i
if ret_sub == RET_OK: # Subscription successful
```

```
        print('subscribe successfully! current subscription status :', quote_ctx.quer
        time.sleep(60) # You can unsubscribe at least 1 minute after subscribing
        ret_unsub, err_message_unsub = quote_ctx.unsubscribe_all() # Cancel all subs
        if ret_unsub == RET_OK:
            print('unsubscribe all successfully! current subscription status:', quote
        else:
            print('Failed to cancel all subscriptions! ', err_message_unsub)
    else:
        print('subscription failed', err_message)
    quote_ctx.close() # After using the connection, remember to close it to prevent
```

- **Output**

```
current subscription status : (0, {'total_used': 0, 'remain': 1000, 'own_used': 0
subscribe successfully! current subscription status : (0, {'total_used': 2, 'rema
unsubscribe all successfully! current subscription status: (0, {'total_used': 0,
```

Interface Limitations

- Supports subscriptions of multiple real-time data types, refer to SubType, each stock subscription for one one quota.
- Please refer to Subscription Quota & Historical Candlestick Quota for Subscription Quota rules.
- You can unsubscribe after subscribing after at least one minute.
- Due to the large amount of SF market data in Hong Kong stocks, in order to ensure the speed of the SF market and the processing performance of OpenD, currently SF authorized users are restricted to subscribing to the order book and broker queue of only 50 securities products (including hkex stocks, warrants, bulls and bears) at the same time, the remaining subscription quota can still be used to subscribe to other types, such as: tickers and brokerage etc.
- HK options and futures do not support subscription to *TICKER* type under LV1 authority.

Get Subscription Status

`query_subscription(is_all_conn=True)`

- **Description**

  Get subscription information

- **Parameters**

Parameter	Type	Description
is_all_conn	bool	Whether to return the subscription status of all connections. 

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	dict	If ret == RET_OK, subscription information data is returned.
	str	If ret != RET_OK, error description is returned.

  - subscription information data format as follows:

```
{
    'total_used': subscription quota used by all connections,
    'own_used': The subscription quota used by the current connection,
    'remain': remaining subscription quota,
    'sub_list': The stock list corresponding to each subscription type,
    {
        'Subscription type': A list of all subscribed stocks under this subscrip
        …
    }
}
```

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

quote_ctx.subscribe(['HK.00700'], [SubType.QUOTE])
ret, data = quote_ctx.query_subscription()
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
{'total_used': 1, 'remain': 999, 'own_used': 1, 'sub_list': {'QUOTE': ['HK.00700
```

Real-time Quote Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  Real-time quotation callback, asynchronous processing of real-time quotation push of subscribed stocks. After receiving the real-time quote data push, it will call back to this function. You need to override on_recv_rsp in the derived class.

- **Parameters**

Parameter	Type	Description
rsp_pb	Qot_UpdateBasicQot_pb2.Response	This parameter does not need to be processed in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, quotation data is returned.
	str	If ret != RET_OK, error description is returned.

  - quotation data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
data_date	str	Date.

Field	Type	Description
data_time	str	Time of latest price.
last_price	float	Latest price.
open_price	float	Open.
high_price	float	High.
low_price	float	Low.
prev_close_price	float	Yesterday's close.
volume	int	Volume.
turnover	float	Turnover.
turnover_rate	float	Turnover rate.
amplitude	int	Amplitude.
suspension	bool	Whether trading is suspended.
listing_date	str	Listing date.
price_spread	float	Spread.
dark_status	DarkStatus	Grey market transaction status.
sec_status	SecurityStatus	Stock status.
strike_price	float	Strike price.
contract_size	float	Contract size.
open_interest	int	Number of open positions.
implied_volatility	float	Implied volatility.
premium	float	Premium.

Field	Type	Description
delta	float	Greek value Delta.
gamma	float	Greek value Gamma.
vega	float	Greek value Vega.
theta	float	Greek value Theta.
rho	float	Greek value Rho.
index_option_type	IndexOptionType	Index option type.
net_open_interest	int	Net open contract number.
expiry_date_distance	int	The number of days from the expiry date.
contract_nominal_value	float	Contract nominal amount.
owner_lot_multiplier	float	Equal number of underlying stocks.
option_area_type	OptionAreaType	Option type (by exercise time).
contract_multiplier	float	Contract multiplier.
pre_price	float	Pre-market price.
pre_high_price	float	Pre-market high.
pre_low_price	float	Pre-market low.
pre_volume	int	Pre-market volume.
pre_turnover	float	Pre-market turnover.
pre_change_val	float	Pre-market change.
pre_change_rate	float	Pre-market change rate.

Field	Type	Description
pre_amplitude	float	Pre-market amplitude.
after_price	float	After-hours price.
after_high_price	float	After-hours high.
after_low_price	float	After-hours low.
after_volume	int	After-hours volume.
After_turnover	float	After-hours turnover.
after_change_val	float	After-hours change.
after_change_rate	float	After-hours change rate.
after_amplitude	float	After-hours amplitude.
overnight_price	float	Overnight price.
overnight_high_price	float	Overnight high.
overnight_low_price	float	Overnight low.
overnight_volume	int	Overnight volume.
overnight_turnover	float	Overnight turnover.
overnight_change_val	float	Overnight change.
overnight_change_rate	float	Overnight change rate.
overnight_amplitude	float	Overnight amplitude.
last_settle_price	float	Yesterday's close.
position	float	Holding position.
position_change	float	Daily position change.

- Example

```
import time
from moomoo import *

class StockQuoteTest(StockQuoteHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, data = super(StockQuoteTest,self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("StockQuoteTest: error, msg: %s"% data)
            return RET_ERROR, data
        print("StockQuoteTest ", data) # StockQuoteTest's own processing logic
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = StockQuoteTest()
quote_ctx.set_handler(handler) # Set real-time quote callback
ret, data = quote_ctx.subscribe(['US.AAPL'], [SubType.QUOTE]) # Subscribe to the
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
quote_ctx.close() # Close the current link, OpenD will automatically cancel the
```

- Output

```
StockQuoteTest        code name data_date data_time  last_price  open_price  high
0  US.AAPL   Apple                              0.0         0.0         0.0
```

> **Tips**
>
> - This interface provides the function of continuously obtaining pushed data. If you need to obtain real-time data at one time, please refer to the Get Real-time Quote of Securities API.
> - For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface.

Real-time Order Book Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  Real-time quotation callback, asynchronous processing of real-time quotation push of subscribed stocks. It will call back to this function after receiving the push of real-time disk data. You need to override on_recv_rsp in the derived class.

- **Parameters**

Parameter	Type	Description
rsp_pb	Qot_UpdateOrderBook_pb2.Response	This parameter does not need to be processed directly in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	dict	If ret == RET_OK, plate data is returned.
	str	If ret != RET_OK, error description is returned.

  - Order Book format as follows：

Field	Type	Description
code	str	Stock code.
name	str	Stock name.

Field	Type	Description
svr_recv_time_bid	str	The time when moomoo server receives order book of bid from the exchange.
svr_recv_time_ask	str	The time when moomoo server receives order book of ask from the exchange.
Bid	list	Each tuple contains the following information：Bid price, bid volume, order qty of bid, order details of bid.
Ask	list	Each tuple contains the following information：Ask price, ask volume, order qty of ask, order details of ask.

The format of Bid and Ask fields as follows：

```
'Bid': [ (bid_price1, bid_volume1, order_num, {'orderid1': order_volume1, 'orderid
'Ask': [ (ask_price1, ask_volume1, order_num, {'orderid1': order_volume1, 'orderid
```

- Example

```
import time
from moomoo import *
class OrderBookTest(OrderBookHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, data = super(OrderBookTest,self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("OrderBookTest: error, msg: %s" % data)
            return RET_ERROR, data
        print("OrderBookTest ", data) # OrderBookTest's own processing logic
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = OrderBookTest()
quote_ctx.set_handler(handler) # Set real-time swing callback
ret, data = quote_ctx.subscribe(['US.AAPL'], [SubType.ORDER_BOOK]) # Subscribe to
if ret == RET_OK:
    print(data)
else:
```

```
18      print('error:', data)
19  time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
20  quote_ctx.close() # Close the current link, OpenD will automatically cancel the
```

- Output

```
1   OrderBookTest  {'code': 'US.AAPL', 'name': 'Apple', 'svr_recv_time_bid': '', 'sv
```

> Tips
>
> - This interface provides the function of continuously obtaining pushed data. If you need to obtain real-time data at one time, please refer to the Get Real-time Orderbook API.
> - For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface.
> - Real-time order book callback for US stocks, will continuously update data during the current trading session, with no need to set the session parameter.

Real-time Candlestick Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  Real-time candlestick callback, asynchronous processing of real-time candlestick push for subscribed stocks.

  After receiving real-time candlestick data push, it will call back to this function. You need to override on_recv_rsp in the derived class.

- **Parameters**

Parameter	Type	Description
rsp_pb	Qot_UpdateKL_pb2.Response	This parameter does not need to be processed directly in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, IPO data is returned.
	str	If ret != RET_OK, error description is returned.

  - IPO data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.

Field	Type	Description
time_key	str	Time.
open	float	Open.
close	float	Close.
high	float	High.
low	float	Low.
volume	int	Volume.
turnover	float	Turnover.
pe_ratio	float	P/E ratio.
turnover_rate	float	Turnover rate.
last_close	float	Yesterday's close.
k_type	KLType	Candlestick type.

- Example

```
import time
from moomoo import *
class CurKlineTest(CurKlineHandlerBase):
     def on_recv_rsp(self, rsp_pb):
        ret_code, data = super(CurKlineTest,self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("CurKlineTest: error, msg: %s"% data)
            return RET_ERROR, data
        print("CurKlineTest ", data) # CurKlineTest's own processing logic
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = CurKlineTest()
quote_ctx.set_handler(handler) # Set real-time candlestick callback
ret, data = quote_ctx.subscribe(['US.AAPL'], [SubType.K_1M], session=Session.ALL
if ret == RET_OK:
    print(data)
```

```
    else:
        print('error:', data)
    time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
    quote_ctx.close() # Close the current link, OpenD will automatically cancel the
```

- Output

```
CurKlineTest        code name             time_key    open    close    high    lo
0  US.AAPL    APPLE  2025-04-07 05:15:00  180.39  180.26  180.46  180.2    1322
```

Tips

- This interface provides the function of continuously obtaining pushed data. If you need to obtain real-time data at one time, please refer to the Get Real-time Candlestick API.
- For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface.
- **Options** related candlestick data, only supports 1 day, 1 minute, 5 minutes, 15 minutes and 60 minutes.

Real-time Time Frame Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  Real-time Time Frame callback, asynchronous processing of real-time Time Frame push of subscribed stocks. After receiving the real-time Time Frame data push, it will call back to this function. You need to override on_recv_rsp in the derived class.

- **Parameters**

Parameter	Type	Description
rsp_pb	Qot_UpdateRT_pb2.Response	This parameter does not need to be processed directly in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, Time Frame data is returned.
	str	If ret != RET_OK, error description is returned.

  - Time Frame data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
time	str	Time.

Field	Type	Description
is_blank	bool	Data status.
opened_mins	int	How many minutes have passed from 0 o'clock.
cur_price	float	Current price.
last_close	float	Yesterday's close.
avg_price	float	Average price.
volume	float	Volume.
turnover	float	Transaction amount.

- Example

```
import time
from moomoo import *

class RTDataTest(RTDataHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, data = super(RTDataTest, self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("RTDataTest: error, msg: %s"% data)
            return RET_ERROR, data
        print("RTDataTest ", data) # RTDataTest's own processing logic
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = RTDataTest()
quote_ctx.set_handler(handler) # Set real-time Time Frame push callback
ret, data = quote_ctx.subscribe(['US.AAPL'], [SubType.RT_DATA], session=Session.A
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
quote_ctx.close() # Close the current link, OpenD will automatically cancel the
```

- Output

```
RTDataTest        code name                time  is_blank  opened_mins  cur_pric
0  US.AAPL    APPLE  2025-04-07 05:24:00     False          324     179.53      18
```

Tips

- This interface provides the function of continuously obtaining pushed data. If you need to obtain real-time data at one time, please refer to the **Get Real-time Time Frame Data** API.
- For the difference between get real-time data and real-time data callback, please refer to **How to Get Real-time Quotes Through Subscription Interface**.

Real-time Tick-by-Tick Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  Real-time callback, asynchronously processing the real-time push of subscribed stocks. After receiving real-time data push, it will call back to this function. You need to override on_recv_rsp in the derived class.

- **Parameters**

Parameter	Type	Description
rsp_pb	Qot_UpdateTicker_pb2.Response	This parameter does not need to be processed directly in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, tick-by-tick data is returned.
	str	If ret != RET_OK, error description is returned.

  - Tick-by-tick data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
sequence	int	Sequence number.

Field	Type	Description
time	str	Transaction time.
price	float	Transaction price.
volume	int	Volume.
turnover	float	Transaction amount.
ticker_direction	TickerDirect	Tick-By-Tick direction.
type	TickerType	Tick-By-Tick type.
push_data_type	PushDataType	Source of data.

- Example

```
import time
from moomoo import *

class TickerTest(TickerHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, data = super(TickerTest,self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("TickerTest: error, msg: %s"% data)
            return RET_ERROR, data
        print("TickerTest ", data) # TickerTest's own processing logic
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = TickerTest()
quote_ctx.set_handler(handler) # Set real-time push callback
ret, data = quote_ctx.subscribe(['US.AAPL'], [SubType.TICKER], session=Session.A
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
quote_ctx.close() # Close the current link, OpenD will automatically cancel the
```

- Output

```
TickerTest        code name                    time   price  volume  turnover ti
0  US.AAPL   APPLE  2025-04-07 05:25:44.116  179.81       9   1618.29          NI

```

Tips

- This interface provides the function of continuously obtaining pushed data. If you need to obtain real-time data at one time, please refer to the Get Real-time Tick-By-Tick API.
- For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface.
- After the market connection is reconnected, during the disconnected period of OpenD pulls, the nearest (up to 50) Tick-By-Tick data is pushed, which can be distinguished by the Tick-By-Tick push type field

Real-time Broker Queue Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  Real-time broker queue callback, asynchronous processing of real-time broker queue push of subscribed stocks. After receiving the real-time broker queue data push, it will call back to this function. You need to override on_recv_rsp in the derived class.

- **Parameters**

Parameter	Type	Description
rsp_pb	Qot_UpdateBroker_pb2.Response	This parameter does not need to be processed directly in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	tuple	If ret == RET_OK, broker queue data is returned.
	str	If ret != RET_OK, error description is returned.

  - Broker queue data format as follows:

Field	Type	Description
stock_code	str	Stock code.
bid_frame_table	pd.DataFrame	Data from bid.
ask_frame_table	pd.DataFrame	Data from ask.

- Bid_frame_table format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
bid_broker_id	int	Bid broker ID.
bid_broker_name	str	Bid broker name.
bid_broker_pos	int	Broker level.
order_id	int	Exchange order ID.
order_volume	int	Order volume.

- Ask_frame_table format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
ask_broker_id	int	Ask Broker ID.
ask_broker_name	str	Ask Broker name.
ask_broker_pos	int	Broker level.
order_id	int	Exchange order ID.
order_volume	int	Order volume.

- **Example**

```
import time
from moomoo import *

```

```
class BrokerTest(BrokerHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, err_or_stock_code, data = super(BrokerTest, self).on_recv_rsp(
        if ret_code != RET_OK:
            print("BrokerTest: error, msg: {}".format(err_or_stock_code))
            return RET_ERROR, data
        print("BrokerTest: stock: {} data: {} ".format(err_or_stock_code, data))
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = BrokerTest()
quote_ctx.set_handler(handler) # Set real-time broker push callback
ret, data = quote_ctx.subscribe(['HK.00700'], [SubType.BROKER]) # Subscribe to t
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
quote_ctx.close() # Close the current link, OpenD will automatically cancel the
```

- Output

```
BrokerTest: stock: HK.00700 data: [        code     name  bid_broker_id
0   HK.00700  TENCENT           5338            J.P. Morgan Broking (Hong Kong) L
..       ...      ...            ...
36  HK.00700  TENCENT           8305  Futu Securities International (Hong Kong) L

[37 rows x 7 columns],         code     name  ask_broker_id
0   HK.00700  TENCENT           1179  Huatai Financial Holdings (Hong Kong) Limi
..       ...      ...            ...
39  HK.00700  TENCENT           6996  China Investment Information Services Limi

[40 rows x 7 columns]]
```

Tips

- This interface provides the function of continuously obtaining pushed data. If you need to obtain real-time data at one time, please refer to the Get Real-time Orderbook API.
- For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface.

- Under the LV1 HK market quotes, the broker queue market data is not supported.

Get Market Snapshot

`get_market_snapshot(code_list)`

- **Description**

  Get market snapshot

- **Parameters**

Parameter	Type	Description
code_list	list	Stock list

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, stock snapshot data is returned.
	str	If ret != RET_OK, error description is returned.

  - Stock snapshot data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
update_time	str	Current update time.
last_price	float	Latest price.
open_price	float	Open.

Field	Type	Description
high_price	float	High.
low_price	float	Low.
prev_close_price	float	Yesterday's close.
volume	int	Volume.
turnover	float	Turnover.
turnover_rate	float	Turnover rate.
suspension	bool	Is suspended or not.
listing_date	str	Listing date.
equity_valid	bool	Is stock or not.
issued_shares	int	Total shares.
total_market_val	float	Total market value.
net_asset	int	Net asset value.
net_profit	int	Net profit.
earning_per_share	float	Earnings per share.
outstanding_shares	int	Shares outstanding.
net_asset_per_share	float	Net assets per share.
circular_market_val	float	Circulation market value.
ey_ratio	float	Yield rate.
pe_ratio	float	P/E ratio.

Field	Type	Description
pb_ratio	float	P/B ratio.
pe_ttm_ratio	float	P/E ratio TTM.
dividend_ttm	float	Dividend TTM, dividend.
dividend_ratio_ttm	float	Dividend rate TTM.
dividend_lfy	float	Dividend LFY, dividend of the previous year.
dividend_lfy_ratio	float	Dividend rate LFY.
stock_owner	str	The code of the underlying stock to which the warrant belongs or the code of the underlying stock of the option.
wrt_valid	bool	Is warrant or not.
wrt_conversion_ratio	float	Conversion ratio.
wrt_type	WrtType	Warrant type.
wrt_strike_price	float	Strike price.
wrt_maturity_date	str	Maturity date.
wrt_end_trade	str	Last trading time.
wrt_leverage	float	Leverage ratio.
wrt_ipop	float	in/out of the money.
wrt_break_even_point	float	Breakeven point.
wrt_conversion_price	float	Conversion price.

Field	Type	Description
wrt_price_recovery_ratio	float	Price recovery ratio.
wrt_score	float	Comprehensive score of warrant.
wrt_code	str	The underlying stock of the warrant (This field has been deprecated and changed to stock_owner.).
wrt_recovery_price	float	Warrant recovery price.
wrt_street_vol	float	Warrant Outstanding quantity.
wrt_issue_vol	float	Warrant issuance.
wrt_street_ratio	float	Outstanding percentage.
wrt_delta	float	Delta value of warrant.
wrt_implied_volatility	float	Warrant implied volatility.
wrt_premium	float	Warrant premium.
wrt_upper_strike_price	float	Upper bound price.
wrt_lower_strike_price	float	lower bound price.
wrt_inline_price_status	PriceType	in/out of bounds
wrt_issuer_code	str	Issuer code.
option_valid	bool	Is option or not.
option_type	OptionType	Option type.

Field	Type	Description
strike_time	str	The option exercise date. 
option_strike_price	float	Strike price.
option_contract_size	float	Number of stocks per contract.
option_open_interest	int	Total open contract number.
option_implied_volatility	float	Implied volatility.
option_premium	float	Premium.
option_delta	float	Greek value Delta.
option_gamma	float	Greek value Gamma.
option_vega	float	Greek value Vega.
option_theta	float	Greek value Theta.
option_rho	float	Greek value Rho.
index_option_type	IndexOptionType	Index option type.
option_net_open_interest	int	Net open contract number.
option_expiry_date_distance	int	The number of days from the expiry date, a negative number means it has expired.
option_contract_nominal_value	float	Contract nominal amount. 

Field	Type	Description
option_owner_lot_multiplier	float	Equal number of underlying stocks.
option_area_type	OptionAreaType	Option type (by exercise time).
option_contract_multiplier	float	Contract multiplier.
plate_valid	bool	Is plate or not.
plate_raise_count	int	Number of stocks that raises in the plate.
plate_fall_count	int	Number of stocks that falls in the plate.
plate_equal_count	int	Number of stocks that dose not change in price in the plate.
index_valid	bool	Is index or not.
index_raise_count	int	Number of stocks that raises in the plate.
index_fall_count	int	Number of stocks that falls in the plate.
index_equal_count	int	Number of stocks that dose not change in the plate.
lot_size	int	The number of shares per lot, stock options represent the number of shares per contract , and

Field	Type	Description
		futures represent contract multipliers.
price_spread	float	The current upward price difference.
ask_price	float	Ask price.
bid_price	float	Bid price.
ask_vol	float	Ask volume.
bid_vol	float	Bid volume.
enable_margin	bool	Whether financing is available (Deprecated).
mortgage_ratio	float	Stock mortgage rate (Deprecated).
long_margin_initial_ratio	float	The initial margin rate of financing (Deprecated).
enable_short_sell	bool	Whether short-selling is available (Deprecated).
short_sell_rate	float	Short-selling reference rate (Deprecated).
short_available_volume	int	Remaining quantity that can be sold short (Deprecated).
short_margin_initial_ratio	float	The initial margin rate for short selling (Deprecated).
sec_status	SecurityStatus	Stock status.

Field	Type	Description
amplitude	float	Amplitude.
avg_price	float	Average price.
bid_ask_ratio	float	The Committee.
volume_ratio	float	Volume ratio.
highest52weeks_price	float	Highest price in 52 weeks.
lowest52weeks_price	float	Lowest price in 52 weeks .
highest_history_price	float	Highest historical price.
lowest_history_price	float	Lowest historical price.
pre_price	float	Pre-market price.
pre_high_price	float	Highest pre-market price.
pre_low_price	float	Lowest pre-market price.
pre_volume	int	Pre-market volume.
pre_turnover	float	Pre-market turnover.
pre_change_val	float	Pre-market change.
pre_change_rate	float	Pre-market change rate.
pre_amplitude	float	Pre-market amplitude.
after_price	float	After-hours price.
after_high_price	float	Highest price after-hours.
after_low_price	float	Lowest price after-hours.

Field	Type	Description
after_volume	int	After-hours trading volume.
after_turnover	float	After-hours turnover.
after_change_val	float	After-hours change.
after_change_rate	float	After-hours change rate.
after_amplitude	float	After-hours amplitude.
overnight_price	float	Overnight price.
overnight_high_price	float	Overnight high.
overnight_low_price	float	Overnight low.
overnight_volume	int	Overnight volume.
overnight_turnover	float	Overnight turnover.
overnight_change_val	float	Overnight change.
overnight_change_rate	float	Overnight change rate.
overnight_amplitude	float	Overnight amplitude.
future_valid	bool	Is futures or not.
future_last_settle_price	float	Yesterday's close.
future_position	float	Holding position.
future_position_change	float	Change in position.
future_main_contract	bool	Is future main contract or not.

Field	Type	Description
future_last_trade_time	str	The last trading time.
trust_valid	bool	Is fund or not.
trust_dividend_yield	float	Dividend rate.
trust_aum	float	Asset scale.
trust_outstanding_units	int	Total circulation.
trust_netAssetValue	float	Net asset value.
trust_premium	float	Premium.
trust_assetClass	AssetClass	Asset class.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_market_snapshot(['HK.00700', 'US.AAPL'])
if ret == RET_OK:
    print(data)
    print(data['code'][0])    # Take the first stock code
    print(data['code'].values.tolist())   # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
    code  name              update_time  last_price  open_price  high_price  low_p
0  HK.00700  TENCENT      2025-04-07 16:09:07      435.40      441.80      462.40
1   US.AAPL    APPLE  2025-04-07 05:30:43.301      188.38      193.89      199.88

   wrt_issue_vol  wrt_street_ratio  wrt_delta  wrt_implied_volatility  wrt_premiu
0            NaN               NaN        NaN                     NaN          Na
1            NaN               NaN        NaN                     NaN          Na
```

```
8
9      trust_outstanding_units  trust_netAssetValue  trust_premium trust_assetClass p
10 0                       NaN                  NaN            NaN              N/A
11 1                       NaN                  NaN            NaN              N/A
12 HK.00700
13 ['HK.00700', 'US.AAPL']
```

> **Interface Limitations**
>
> - Request up to 60 snapshots every 30 seconds
> - For each request, the maximum number of stock codes supported by the parameter *code_list* is 400.

Get Real-time Quote

`get_stock_quote(code_list)`

- **Description**

  To get real-time quotes of subscribed securities, you must subscribe first.

- **Parameters**

Parameter	Type	Description
code_list	list	Stock list. Data type of elements in the list is str.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, quotation data is returned.
	str	If ret != RET_OK, error description is returned.

  - quotation data format as follows:

Field	Type	Description
code	str	Stock code.
data_date	str	Date.
data_time	str	Time of latest price. 
last_price	float	Latest price.
open_price	float	Open.

Field	Type	Description
high_price	float	High.
low_price	float	Low.
prev_close_price	float	Yesterday's close.
volume	int	Volume.
turnover	float	Turnover.
turnover_rate	float	Turnover rate.
amplitude	int	Amplitude.
suspension	bool	Whether trading is suspended.
listing_date	str	Listing date.
price_spread	float	Spread.
dark_status	DarkStatus	Grey market transaction status.
sec_status	SecurityStatus	Stock status.
strike_price	float	Strike price.
contract_size	float	Contract size.
open_interest	int	Number of open positions.
implied_volatility	float	Implied volatility.
premium	float	Premium.
delta	float	Greek value Delta.
gamma	float	Greek value Gamma.
vega	float	Greek value Vega.

Field	Type	Description
theta	float	Greek value Theta.
rho	float	Greek value Rho.
index_option_type	IndexOptionType	Index option type.
net_open_interest	int	Net open contract number. 
expiry_date_distance	int	The number of days from the expiry date. 
contract_nominal_value	float	Contract nominal amount. 
owner_lot_multiplier	float	Equal number of underlying stocks. 
option_area_type	OptionAreaType	Option type (by exercise time).
contract_multiplier	float	Contract multiplier.
pre_price	float	Pre-market price.
pre_high_price	float	Pre-market high.
pre_low_price	float	Pre-market low.
pre_volume	int	Pre-market volume.
pre_turnover	float	Pre-market turnover.
pre_change_val	float	Pre-market change.
pre_change_rate	float	Pre-market change rate. 
pre_amplitude	float	Pre-market amplitude. 
after_price	float	After-hours price.
after_high_price	float	After-hours high.

Field	Type	Description
after_low_price	float	After-hours low.
after_volume	int	After-hours volume.
After_turnover	float	After-hours turnover.
after_change_val	float	After-hours change.
after_change_rate	float	After-hours change rate.
after_amplitude	float	After-hours amplitude.
overnight_price	float	Overnight price.
overnight_high_price	float	Overnight high.
overnight_low_price	float	Overnight low.
overnight_volume	int	Overnight volume.
overnight_turnover	float	Overnight turnover.
overnight_change_val	float	Overnight change.
overnight_change_rate	float	Overnight change rate.
overnight_amplitude	float	Overnight amplitude.
last_settle_price	float	Yesterday's close.
position	float	Holding position.
position_change	float	Daily position change.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

```

```
ret_sub, err_message = quote_ctx.subscribe(['US.AAPLE'], [SubType.QUOTE], subscri
# Subscribe to the K line type first. After the subscription is successful, OpenD
if ret_sub == RET_OK: # Subscription successful
    ret, data = quote_ctx.get_stock_quote(['US.AAPL']) # Get real-time data of s
    if ret == RET_OK:
        print(data)
        print(data['code'][0]) # Take the first stock code
        print(data['code'].values.tolist()) # Convert to list
    else:
        print('error:', data)
else:
    print('subscription failed', err_message)
quote_ctx.close() # Close the current connection, OpenD will automatically cancel
```

- Output

```
  code name   data_date     data_time  last_price  open_price  high_price  low_pric
0  US.AAPL   APPLE  2025-04-07  05:37:21.794      188.38      193.89      199.88
US.AAPL
['US.AAPL']
```

> **Tips**
>
> - This API provides the function to obtain real-time data at one time. If you need to obtain pushed data continuously, please refer to the Real-time Quote Callback API.
> - For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface.

Get Real-time Order Book

`get_order_book(code, num=10)`

- **Description**

  To get the real-time order book of subscribed stocks, you must subscribe first.

- **Parameters**

Parameter	Type	Description
code	str	Stock code.
name	str	Stock name.
num	int	The requested number of price levels.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	dict	If ret == RET_OK, plate data is returned.
	str	If ret != RET_OK, error description is returned.

  - Order Book format as follows：

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
svr_recv_time_bid	str	The time when moomoo server receives order book of bid from the exchange.

Field	Type	Description
svr_recv_time_ask	str	The time when moomoo server receives order book of ask from the exchange.
Bid	list	Each tuple contains the following information：Bid price, bid volume, order qty of bid, order details of bid.
Ask	list	Each tuple contains the following information：Ask price, ask volume, order qty of ask, order details of ask.

The format of Bid and Ask fields as follows：

```
'Bid': [ (bid_price1, bid_volume1, order_num, {'orderid1': order_volume1, 'orderid
'Ask': [ (ask_price1, ask_volume1, order_num, {'orderid1': order_volume1, 'orderid
```

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
ret_sub = quote_ctx.subscribe(['US.AAPL'], [SubType.ORDER_BOOK], subscribe_push=
# First subscribe to the order type. After the subscription is successful, OpenD
if ret_sub == RET_OK:  # Successfully subscribed
    ret, data = quote_ctx.get_order_book('US.AAPL', num=3)  # Get 3 files of rea
    if ret == RET_OK:
        print(data)
    else:
        print('error:', data)
else:
    print('subscription failed')
quote_ctx.close()  # Close the current connection, OpenD will automatically canc
```

- **Output**

```
{'code': 'US.AAPL', 'name': 'APPLE', 'svr_recv_time_bid': '2025-04-07 05:39:20.3
```

Interface Limitations

- The time field in which the moomoo server receives data from the exchange. Only supports A-share Market stocks, HK stocks, ETFs, warrants, bulls and bears, and this data is only available at the opening time.
- The time field in which the moomoo server receives data from the exchange. The receiving time of some data is zero, such as server restart or cached data pushed for the first time.

Tips

- This API provides the function of obtaining real-time data at one time. If you need to obtain pushed data continuously, please refer to the **Real-time OrderBook Callback** API.
- For the difference between get real-time data and real-time data callback, please refer to **How to Get Real-time Quotes Through Subscription Interface**.
- The real-time order book data will be returned during the current trading session for US stocks, with no need to set the session parameter.

Get Real-time Candlestick

```
get_cur_kline(code, num, ktype=SubType.K_DAY, autype=AuType.QFQ)
```

- **Description**

  Get real-time candlestick data of subscribed stocks, you must subscribe first.

- **Parameters**

Parameter	Type	Description
code	str	Stock code.
num	int	The number of candlesticks.
ktype	KLType	Candlestick type.
autype	AuType	Type of adjustment.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, IPO data is returned.
	str	If ret != RET_OK, error description is returned.

  - IPO data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.

Field	Type	Description
time_key	str	Time.
open	float	Open.
close	float	Close.
high	float	High.
low	float	Low.
volume	int	Volume.
turnover	float	Turnover.
pe_ratio	float	P/E ratio.
turnover_rate	float	Turnover rate.
last_close	float	Yesterday's close.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret_sub, err_message = quote_ctx.subscribe(['US.AAPL'], [SubType.K_DAY], subscri
# First subscribe to the candlestick type. After the subscription is successful,
if ret_sub == RET_OK:  # Successfully subscribed
    ret, data = quote_ctx.get_cur_kline('US.AAPL', 2, SubType.K_DAY, AuType.QFQ)
    if ret == RET_OK:
        print(data)
        print(data['turnover_rate'][0])   # Take the first turnover rate
        print(data['turnover_rate'].values.tolist())   # Convert to list
    else:
        print('error:', data)
else:
    print('subscription failed', err_message)
quote_ctx.close()  # Close the current link, OpenD will automatically cancel the
```

- Output

```
code name             time_key    open   close    high     low      volume      tu
0  US.AAPL   APPLE  2025-04-03 00:00:00  205.54  203.19  207.49  201.25  1034190(
1  US.AAPL   APPLE  2025-04-04 00:00:00  193.89  188.38  199.88  187.34  1259109
0.00689
[0.00689, 0.00838]
```

Interface Limitations

- This interface is to obtain real-time candlestick, which can obtain the nearest 1000 at most. To get historical candlestick, please refer to Get historical candlestick.
- Only a stock of daily timeframe and above have P/E ratio and turnover ratio fields.
- **Options** related candlestick data, only supports 1 day, 1 minute, 5 minutes, 15 minutes and 60 minutes.

Tips

- This API provides the function of obtaining candlestick data at one time. If you need to obtain pushed data continuously, please refer to the Real-time Candlestick Callback API.
- For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface.

Get Real-time Time Frame Data

`get_rt_data(code)`

- **Description**

  Obtain real-time tick-by-tick data for a specified stock. (Require real-time data subscription.)

- **Parameters**

Parameter	Type	Description
code	str	Stock code.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, Time Frame data is returned.
	str	If ret != RET_OK, error description is returned.

  - Time Frame data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
time	str	Time.
is_blank	bool	Data status.
opened_mins	int	How many minutes have passed from 0 o'clock.

Field	Type	Description
cur_price	float	Current price.
last_close	float	Yesterday's close.
avg_price	float	Average price.
volume	float	Volume.
turnover	float	Transaction amount.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
ret_sub, err_message = quote_ctx.subscribe(['US.AAPL'], [SubType.RT_DATA], subscr
# Subscribe to the Time Frame data type first. After the subscription is success
if ret_sub == RET_OK:   # Successfully subscribed
    ret, data = quote_ctx.get_rt_data('US.AAPL')   # Get Time Frame data once
    if ret == RET_OK:
        print(data)
    else:
        print('error:', data)
else:
    print('subscription failed', err_message)
quote_ctx.close()   # Close the current link, OpenD will automatically cancel the
```

- Output

```
code  name                time  is_blank  opened_mins  cur_price  last_close   a
0    US.AAPL   APPLE  2025-04-06 20:01:00     False         1201     183.00
..       ...    ...                  ...       ...          ...        ...
586  US.AAPL   APPLE  2025-04-07 05:47:00     False          347     181.26

[587 rows x 10 columns]
```

> Tips

- This API provides the function of obtaining real-time data at one time. If you need to obtain pushed data continuously, please refer to the **Real-time Time Frame Callback** API.
- For the difference between get real-time data and real-time data callback, please refer to **How to Get Real-time Quotes Through Subscription Interface**.

Get Real-time Tick-by-Tick

`get_rt_ticker(code, num=500)`

- Description

  To get real-time tick-by-tick of subscribed stocks. (Require real-time data subscription.)

- Parameters

Parameter	Type	Description
code	str	Stock code.
num	int	Number of recent tick-by-tick.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, tick-by-tick data is returned.
	str	If ret != RET_OK, error description is returned.

  - Tick-by-tick data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
sequence	int	Sequence number.
time	str	Transaction time.

Field	Type	Description
price	float	Transaction price.
volume	int	Volume.
turnover	float	Transaction amount.
ticker_direction	TickerDirect	Tick-By-Tick direction.
type	TickerType	Tick-By-Tick type.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret_sub, err_message = quote_ctx.subscribe(['US.AAPL'], [SubType.TICKER], subscri
# First subscribe to each type. After the subscription is successful, OpenD will
if ret_sub == RET_OK: # Subscription successful
     ret, data = quote_ctx.get_rt_ticker('US.AAPL', 2) # Get the last 2 transacti
     if ret == RET_OK:
         print(data)
         print(data['turnover'][0]) # Take the first transaction amount
         print(data['turnover'].values.tolist()) # Convert to list
     else:
         print('error:', data)
else:
     print('subscription failed', err_message)
quote_ctx.close() # Close the current link, OpenD will automatically cancel the
```

- Output

```
code name                    time   price  volume  turnover ticker_direction
0  US.AAPL   APPLE  2025-04-07 05:50:23.745  181.70       2    363.40          NI
1  US.AAPL   APPLE  2025-04-07 05:50:24.170  181.73       1    181.73          NI
363.4
[363.4, 181.73]
```

Interface Limitations

- You can get up to the latest 1000 tick-by-tick data, more historical tick-by-tick data is not yet available
- Under the authority of LV1 HK futures and options market, tick-by-tick data is not available

Tips

- This API provides the function of obtaining real-time data at one time. If you need to obtain pushed data continuously, please refer to the **Real-time Tick-By-Tick Callback** API.
- For the difference between get real-time data and real-time data callback, please refer to **How to Get Real-time Quotes Through Subscription Interface**.

Get Real-time Broker Queue

get_broker_queue

`get_broker_queue(code)`

- **Description**

  Obtain real-time data of market participants on the order book. (Require real-time data subscription.)

- **Parameters**

Parameter	Type	Description
code	str	Stock code.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
bid_frame_table	pd.DataFrame	If ret == RET_OK, queue of bid brokers is returned.
	str	If ret != RET_OK, error description is returned.
ask_frame_table	pd.DataFrame	If ret == RET_OK, queue of ask brokers is returned.
	str	If ret != RET_OK, error description is returned.

  - Queue of bid brokers format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
bid_broker_id	int	Bid broker ID.
bid_broker_name	str	Bid broker name.
bid_broker_pos	int	Broker level.
order_id	int	Exchange order ID.
order_volume	int	Order volume.

- Queue of ask brokers format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
ask_broker_id	int	Ask Broker ID.
ask_broker_name	str	Ask Broker name.
ask_broker_pos	int	Broker level.
order_id	int	Exchange order ID.
order_volume	int	Order volume.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
ret_sub, err_message = quote_ctx.subscribe(['HK.00700'], [SubType.BROKER], subscr
# First subscribe to the broker queue type. After the subscription is successful,
if ret_sub == RET_OK: # Subscription successful
```

```
        ret, bid_frame_table, ask_frame_table = quote_ctx.get_broker_queue('HK.00700
        if ret == RET_OK:
            print(bid_frame_table)
        else:
            print('error:', bid_frame_table)
    else:
        print(err_message)
    quote_ctx.close() # Close the current connection, OpenD will automatically cancel
```

- Output

```
        code     name  bid_broker_id                                bid_broker_na
    0   HK.00700  TENCENT           5338           J.P. Morgan Broking (Hong Kong) L
    ..       ...      ...            ...
    36  HK.00700  TENCENT           8305  Futu Securities International (Hong Kong) L

    [37 rows x 7 columns]
```

> **Tips**
>
> - This API provides the function of obtaining real-time data at one time. If you need to obtain pushed data continuously, please refer to the Real-time Broker Queue Callback API.
> - For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface.
> - Under the LV1 HK market quotes, the broker queue market data is not supported.

Get Market Status of Securities

`get_market_state(code_list)`

- **Description**

  Get market status of underlying security

- **Parameters**

Parameter	Type	Description
code_list	list	A list of security codes that need to query for market status. 

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, market status data is returned.
	str	If ret != RET_OK, error description is returned.

  - Market status data format as follows:

Field	Type	Description
code	str	Security code.
stock_name	str	Security name.
market_state	MarketState	Market state.

- **Example**

```python
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_market_state(['SZ.000001', 'HK.00700'])
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
    code          stock_name   market_state
0  SZ.000001    Ping An Bank  AFTERNOON
1  HK.00700     Tencent       AFTERNOON
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds
> - The maximum number of stock codes for each request is 400.

Get Capital Flow

`get_capital_flow(stock_code, period_type = PeriodType.INTRADAY, start=None, end=None)`

- Description

  Get the flow of a specific stock

- Parameters

Parameter	Type	Description
stock_code	str	Stock code.
period_type	PeriodType	Period Type.
start	str	Start time.
end	str	End time.

  - The combination of ***start*** and ***end*** is as follows

start type	end type	Description
str	str	***start*** and ***end*** are the specified dates respectively.
None	str	***start*** is 365 days before ***end***.
str	None	***end*** is 365 days after ***start***.
None	None	***end*** is the current date, ***start*** is 365 days before.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.

data	pd.DataFrame	If ret == RET_OK, capital flow data is returned.
	str	If ret != RET_OK, error description is returned.

- Capital flow data format as follows:

Field	Type	Description
in_flow	float	Net inflow of capital.
main_in_flow	float	Block Orders Net Inflow.
super_in_flow	float	Extra-large Orders Net Inflow.
big_in_flow	float	Large Orders Net Inflow.
mid_in_flow	float	Medium Orders Net Inflow.
sml_in_flow	float	Small Orders Net Inflow.
capital_flow_item_time	str	Start time string.
last_valid_time	str	Last valid time string of data.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_capital_flow("HK.00700", period_type = PeriodType.INTRA
if ret == RET_OK:
    print(data)
    print(data['in_flow'][0]) # Take the first net inflow of capital
    print(data['in_flow'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent
```

- **Output**

```
    last_valid_time       in_flow  ...  main_in_flow  capital_flow_item_time
0               N/A -1.857915e+08  ... -1.066828e+08     2021-06-08 00:00:00
..              ...           ...  ...           ...                     ...
245             N/A  2.179240e+09  ...  2.143345e+09     2022-06-08 00:00:00

[246 rows x 8 columns]
-185791500.0
[-185791500.0, -18315000.0, -672100100.0, -714394350.0, -698391950.0, -818886750
..                   ...           ...                   ...
2031460.0, 638067040.0, 622466600.0, -351788160.0, -328529240.0, 715415020.0, 767
```

Interface Limitations

- A maximum of 30 requests per 30 seconds
- Supported for stocks, warrants and funds only
- Historical period (day, month, year) Only provides data for the latest 1 year; Intraday period only provides data for the latest day.
- Data with historical period (day, month, year), is only supported for the last 2 years. While Data with intraday period is only supported for the latest day.
- Output data only includes tha data during Regular Trading Hours, not the data during Pre and Post-Market Hours.

Get Capital Distribution

`get_capital_distribution(stock_code)`

- **Description**

  Access to capital distribution

- **Parameters**

Parameter	Type	Description
stock_code	str	Stock code.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, stock fund distribution data is returned.
	str	If ret != RET_OK, error description is returned.

  - Stock fund distribution data format as follows:

Field	Type	Description
capital_in_super	float	Inflow capital quota, extra-large order.
capital_in_big	float	Inflow capital quota, large order.
capital_in_mid	float	Inflow capital quota, midium order.
capital_in_small	float	Inflow capital quota, small order.
capital_out_super	float	Outflow capital quota, extra-large order.

Field	Type	Description
capital_out_big	float	Outflow capital quota, large order.
capital_out_mid	float	Outflow capital quota, midium order.
capital_out_small	float	Outflow capital quota, small order.
update_time	str	Updated time string.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_capital_distribution("HK.00700")
if ret == RET_OK:
    print(data)
    print(data['capital_in_big'][0]) # Take the amount of inflow capital of the
    print(data['capital_in_big'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent
```

- Output

```
   capital_in_super  capital_in_big  ...  capital_out_small          update_time
0      2.261085e+09    2.141964e+09  ...       2.887413e+09  2022-06-08 15:59:59

[1 rows x 9 columns]
2141963720.0
[2141963720.0]
```

> **Interface Limitations**
>
> - A maximum of 30 requests per 30 seconds
> - Only support stocks, warrants and funds.
> - For more capital flow introduction, please refer to here.

- Output data only includes tha data during Regular Trading Hours, not the data during Pre and Post-Market Hours.

Get Plates of Stocks

`get_owner_plate(code_list)`

- **Description**

  Get the information of plates to which the stocks belong

- **Parameters**

Parameter	Type	Description
code_list	list	Stock code list.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, data of the sector is returned.
	str	If ret != RET_OK, error description is returned.

  - Data of the sector format as follows:

Field	Type	Description
code	str	Securities code.
name	str	Stock name.
plate_code	str	Plate code.
plate_name	str	Plate name.
plate_type	Plate	Plate type.

- Example

```python
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

code_list = ['HK.00001']
ret, data = quote_ctx.get_owner_plate(code_list)
if ret == RET_OK:
    print(data)
    print(data['code'][0]) # Take the first stock code
    print(data['plate_code'].values.tolist()) # Convert plate code to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
    code          name          plate_code                          plate_name
0   HK.00001  CKH HOLDINGS  HK.HSI Constituent  ConstituentStocks in Hang Seng In
..       ...           ...                 ...
8   HK.00001  CKH HOLDINGS           HK.BK1983                                 HK

[9 rows x 5 columns]
HK.00001
['HK.HSI Constituent', 'HK.GangGuTong', 'HK.BK1000', 'HK.BK1061', 'HK.BK1107', 'H
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds
> - The maximum number of stocks of each request list is 200
> - Only supports stocks and indices

Get Historical Candlesticks

`request_history_kline(code, start=None, end=None, ktype=KLType.K_DAY, autype=AuType.QFQ, fields=[KL_FIELD.ALL], max_count=1000, page_req_key=None, extended_time=False)`

- **Description**

  Get historical candlesticks

- **Parameters**

Parameter	Type	Description
code	str	Stock code.
start	str	Start time.
end	str	End time.
ktype	KLType	Candlestick type.
autype	AuType	Type of adjustment.
fields	KL_FIELD	List of fields to be returned.
max_count	int	The maximum number of candlesticks returned in this request.
page_req_key	bytes	The key of the page request. If the number of candlesticks between start and end is more than max_count, then None should be passed at the first time you call this interface, and the page_req_key returned by the last call must be passed in the subsequent pagerequests.
extended_time	bool	Need pre-market and after-hours data for US stocks or not. False: not need, True: need.

Parameter	Type	Description
session	Session	Get US stocks historical k-line in session

- The combination of ***start*** and ***end*** is as follows

Start type	End type	Description
str	str	***start*** and ***end*** are the specified dates respectively.
None	str	***start*** is 365 days before ***end***.
str	None	***end*** is 365 days after ***start***.
None	None	***end*** is the current date, ***start*** is 365 days before.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, historical candlestick data is returned.
	str	If ret != RET_OK, error description is returned.
page_req_key	bytes	The key of the next page request.

- Historical candlestick data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
time_key	str	Candlestick time.

Field	Type	Description
open	float	Open.
close	float	Close.
high	float	High.
low	float	Low.
pe_ratio	float	P/E ratio.
turnover_rate	float	Turnover rate.
volume	int	Volume.
turnover	float	Turnover.
change_rate	float	Change rate.
last_close	float	Yesterday's close.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
ret, data, page_req_key = quote_ctx.request_history_kline('US.AAPL', start='2019
if ret == RET_OK:
    print(data)
    print(data['code'][0]) # Take the first stock code
    print(data['close'].values.tolist()) # The closing price of the first page i
else:
    print('error:', data)
while page_req_key != None: # Request all results after
    print('************************************')
    ret, data, page_req_key = quote_ctx.request_history_kline('US.AAPL', start='
    if ret == RET_OK:
        print(data)
    else:
        print('error:', data)
```

```
print('All pages are finished!')
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
code  name            time_key       open      close       high        low  pe_r
0  US.AAPL   APPLE  2019-09-11 00:00:00  52.631194  53.963447  53.992409  52.5491
..       ...   ...                  ...        ...        ...        ...
4  US.AAPL   APPLE  2019-09-17 00:00:00  53.087346  53.265945  53.294907  52.8846

[5 rows x 13 columns]
US.AAPL
[53.9634465, 53.84156475, 52.7953125, 53.072865, 53.265945]
************************************
       code  name            time_key       open      close       high        lo
0  US.AAPL   APPLE  2019-09-18 00:00:00  53.352831  53.76554  53.784847  52.9618
All pages are finished!
```

> **Interface Restrictions**
>
> - Candlestick data with timeframes of **60 minutes and below**, is only supported for the last 8 years. **Daily** candlestick data is supported for the last 20 years. **Daily above** candlestick data is not restricted.
> - We will issue historical candlestick quota based on your account assets and transaction conditions. Therefore, you can only obtain historical candlestick data for a limited number of stocks within 30 days. For specific rules, please refer to Subscription Quota & Historical Candlestick Quota. The historical candlestick quota you consume on that day will be automatically released after 30 days.
> - A maximum of 60 requests per 30 seconds. Note: If you obtain data by page, this frequency limit rule is only applicable to the first time calling the interface, and subsequent pages request frequency is unlimited.
> - *Change rate*, only supports timeframes of daily and above.
> - **Options** related candlestick data, only supports 1 day, 1 minute, 5 minutes, 15 minutes and 60 minutes.
> - The pre-market, after-hours and overnight candlestick of US stocks, only supports timeframes of 60 minutes and below. Since the pre-market, after-hours and overnight session of the US stock market are irregular trading hours, the candlestick data for this period may be less than 2 years.

- *Turnover* of US stocks, only supports data after 2015-10-12.

Get Adjustment Factor

`get_rehab(code)`

- **Description**

  Get the stock adjustment factor

- **Parameters**

Parameter	Type	Description
code	str	Stock code.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, data for adjustment is returned.
	str	If ret != RET_OK, error description is returned.

  - Data for adjustment format as follows:

Field	Type	Description
ex_div_date	str	Ex-dividend date.
split_base	float	Split numerator.
split_ert	float	Split dominator.
join_base	float	Joint numerator.
join_ert	float	Joint dominator.

Field	Type	Description
split_ratio	float	Split ratio.
per_cash_div	float	Dividend per share.
bounce_base	float	Bounce numerator.
bounce_ert	float	Bounce dominator.
per_share_div_ratio	float	Bounce ratio.
transfer_base	float	Conversion numerator.
transfer_ert	float	Conversion dominator.
per_share_trans_ratio	float	Conversion ratio.
allot_base	float	Allotment numerator.
allot_ert	float	Allotment dominator.
allotment_ratio	float	Allotment ratio.
allotment_price	float	Issuance price.
add_base	float	Additional issuance numerator.
add_ert	float	Additional issuance dominator.
stk_spo_ratio	float	Additional issuance ratio.
stk_spo_price	float	Additional issuance price.
spin_off_base	float	Spin-off numerator.
spin_off_ert	float	Spin-off dominator.
spin_off_ratio	float	Spin-off ratio.
forward_adj_factorA	float	Forward adjustment factor A.

Field	Type	Description
forward_adj_factorB	float	Forward adjustment factor B.
backward_adj_factorA	float	Backward adjustment factor A.
backward_adj_factorB	float	Backward adjustment factor B.

Price after forward adjustment = price before forward adjustment * Forward adjustment factor A + Forward adjustment factor B
Price after backward adjustment = price before backward adjustment * Backward adjustment factor A + Backward adjustment factor B

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_rehab("HK.00700")
if ret == RET_OK:
    print(data)
    print(data['ex_div_date'][0]) # Take the first ex-dividend date
    print(data['ex_div_date'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
    ex_div_date  split_ratio  per_cash_div  per_share_div_ratio  per_share_trans_
0   2005-04-19          NaN          0.07                  NaN
..         ...          ...           ...                  ...
15  2019-05-17          NaN          1.00                  NaN

[16 rows x 16 columns]
2005-04-19
['2005-04-19', '2006-05-15', '2007-05-09', '2008-05-06', '2009-05-06', '2010-05-0
```

> Interface Limitations

- A maximum of 60 requests per 30 seconds

Get Option Expiration Date

`get_option_expiration_date(code, index_option_type=IndexOptionType.NORMAL)`

- Description

  Query all expiration dates of option chains through the underlying stock. To obtain the complete option chain, please use it in combination with Get Option Chain.

- Parameters

Parameter	Type	Description
code	str	Stock code.
index_option_type	IndexOptionType	Index option type.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, option expiration date data is returned.
	str	If ret != RET_OK, error description is returned.

  - Option expiration date data format as follows:

Field	Type	Description
strike_time	str	Exercise date.
option_expiry_date_distance	int	The number of days from the expiry date.
expiration_cycle	ExpirationCycle	Expiration cycle.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
ret, data = quote_ctx.get_option_expiration_date(code='HK.00700')
if ret == RET_OK:
    print(data)
    print(data['strike_time'].values.tolist())  # Convert to list
else:
    print('error:', data)
quote_ctx.close()  # After using the connection, remember to close it to prevent
```

- Output

```
  strike_time  option_expiry_date_distance expiration_cycle
0  2021-04-29                           4              N/A
1  2021-05-28                          33              N/A
2  2021-06-29                          65              N/A
3  2021-07-29                          95              N/A
4  2021-09-29                         157              N/A
5  2021-12-30                         249              N/A
6  2022-03-30                         339              N/A
['2021-04-29', '2021-05-28', '2021-06-29', '2021-07-29', '2021-09-29', '2021-12-
```

Interface Limitations

- A maximum of 60 requests per 30 seconds

Get Option Chain

```
get_option_chain(code, index_option_type=IndexOptionType.NORMAL, start=None, end=None, option_type=OptionType.ALL, option_cond_type=OptionCondType.ALL, data_filter=None)
```

- **Description**

  Query the option chain from an underlying stock. This interface only returns the static information of the option chain. If you need to obtain dynamic information such as quotation or trading, please use the security code returned by this interface to subscribe the required security.

- **Parameters**

Parameter	Type	Description
code	str	Code of underlying stock.
index_option_type	IndexOptionType	Index option type.
start	str	Start date, for expiration date.
end	str	End date (including this day), for expiration date.
option_type	OptionType	Option type for call/put.
option_cond_type	OptionCondType	Option type for in/out of the money.
data_filter	*OptionDataFilter*	Data filter condition.

  - The combination of ***start*** and ***end*** is as follows:

Start type	End type	Description
str	str	***start*** and ***end*** are the specified dates respectively.

Start type	End type	Description
None	str	***start*** is 30 days before ***end***.
str	None	***end*** is 30 days after ***start***.
None	None	***start*** is the current date, ***end*** is 30 days later.

- *OptionDataFilter* fields are as follows

Field	Type	Description
implied_volatility_min	float	Min value of implied volatility.
implied_volatility_max	float	Max value of implied volatility.
delta_min	float	Min value of Greek value Delta.
delta_max	float	Max value of Greek value Delta.
gamma_min	float	Min value of Greek value Gamma.
gamma_max	float	Max value of Greek value Gamma.
vega_min	float	Min value of Greek value Vega.
vega_max	float	Max value of Greek value Vega.
theta_min	float	Min value of Greek value Theta.
theta_max	float	Max value of Greek value Theta.
rho_min	float	Min value of Greek value Rho.
rho_max	float	Max value of Greek value Rho.
net_open_interest_min	float	Min value of net open contract number.
net_open_interest_max	float	Max value of net open contract number.
open_interest_min	float	Min value of open contract number.

Field	Type	Description
open_interest_max	float	Max value of open contract number.
vol_min	float	Min value of Volume.
vol_max	float	Max value of Volume.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, option chain data is returned.
	str	If ret != RET_OK, error description is returned.

  - Option chain data format as follows:

Field	Type	Description
code	str	Security code.
name	str	Security name.
lot_size	int	Number of shares per lot, number of shares per contract for options.
stock_type	SecurityType	Stock type.
option_type	OptionType	Option type.
stock_owner	str	Underlying stock.
strike_time	str	Exercise date.
strike_price	float	Strike price.

Field	Type	Description
suspension	bool	Whether is suspended.
stock_id	int	Stock ID.
index_option_type	IndexOptionType	Index option type.
expiration_cycle	ExpirationCycle	Expiration cycle type.
option_standard_type	OptionStandardType	Option standard type.
option_settlement_mode	OptionSettlementMode	Option settlement mode.

- Example

```python
from moomoo import *
import time
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
ret1, data1 = quote_ctx.get_option_expiration_date(code='HK.00700')

filter1 = OptionDataFilter()
filter1.delta_min = 0
filter1.delta_max = 0.1

if ret1 == RET_OK:
    expiration_date_list = data1['strike_time'].values.tolist()
    for date in expiration_date_list:
        ret2, data2 = quote_ctx.get_option_chain(code='HK.00700', start=date, end
        if ret2 == RET_OK:
            print(data2)
            print(data2['code'][0])  # Take the first stock code
            print(data2['code'].values.tolist())  # Convert to list
        else:
            print('error:', data2)
        time.sleep(3)
else:
    print('error:', data1)
quote_ctx.close()  # After using the connection, remember to close it to prevent
```

- Output

```
                     code                 name  lot_size stock_type option_type s
0     HK.TCH210429C350000   腾讯 210429 350.00 购       100       DRVT        CAL
1     HK.TCH210429P350000   腾讯 210429 350.00 沽       100       DRVT         PU
2     HK.TCH210429C360000   腾讯 210429 360.00 购       100       DRVT        CAL
3     HK.TCH210429P360000   腾讯 210429 360.00 沽       100       DRVT         PU
4     HK.TCH210429C370000   腾讯 210429 370.00 购       100       DRVT        CAL
5     HK.TCH210429P370000   腾讯 210429 370.00 沽       100       DRVT         PU
HK.TCH210429C350000
['HK.TCH210429C350000', 'HK.TCH210429P350000', 'HK.TCH210429C360000', 'HK.TCH2104
...
                   code                 name  lot_size stock_type option_type stoc
0   HK.TCH220330C490000  腾讯 220330 490.00 购       100       DRVT        CALL
1   HK.TCH220330P490000  腾讯 220330 490.00 沽       100       DRVT         PUT
2   HK.TCH220330C500000  腾讯 220330 500.00 购       100       DRVT        CALL
3   HK.TCH220330P500000  腾讯 220330 500.00 沽       100       DRVT         PUT
4   HK.TCH220330C510000  腾讯 220330 510.00 购       100       DRVT        CALL
5   HK.TCH220330P510000  腾讯 220330 510.00 沽       100       DRVT         PUT
HK.TCH220330C490000
['HK.TCH220330C490000', 'HK.TCH220330P490000', 'HK.TCH220330C500000', 'HK.TCH2201
```

Interface Limitations

- A maximum of 10 requests per 30 seconds
- The upper limit of the incoming time span is 30 days

Tips

- This interface does not support the query of expired option chains, please enter today or future date to the **End date** parameter.
- Open interest (OI) is updated daily and the specific timing depends on the exchange.
  - For U.S. stock options, the data is updated during the PRE_MARKET session.
  - For Hong Kong stock options, the data is updated after the Regular Trading Hours.

Get Filtered Warrant

`get_warrant(stock_owner='', req=None)`

- **Description**

  Get Filtered Warrant (only warrants, CBBCs and Inline Warrants of HK market are surpported)

- **Parameters**

Parameter	Type	Description
stock_owner	str	Code of the underlying stock.
req	*WarrantRequest*	Filter parameter combination.

  - *WarrantRequest*'s details as follows:

Field	Type	Description
begin	int	Data start point
num	int	The number of requested data.
sort_field	SortField	According to which field to sort.
ascend	bool	The sort direction.
type_list	list	Warrant Type Filter List.
issuer_list	list	Issuer filter list.
maturity_time_min	str	The start time of the maturity date filter range.
maturity_time_max	str	The end time of the maturity date filter range.

Field	Type	Description
ipo_period	IpoPeriod	Listing period.
price_type	PriceType	In/out of the money.
status	WarrantStatus	Warrant Status.
cur_price_min	float	The filter lower limit (closed interval) of the latest price.
cur_price_max	float	The filter upper limit (closed interval) of the latest price.
strike_price_min	float	The lower filter limit (closed interval) of the strike price.
strike_price_max	float	The upper filter limit (closed interval) of the strike price.
street_min	float	The lower limit (closed interval) of Outstanding percentage.
street_max	float	The upper limit (closed interval) of Outstanding percentage.
conversion_min	float	The lower filter limit (closed interval) of the conversion ratio.
conversion_max	float	The upper filter limit (closed interval) of the conversion ratio.
vol_min	int	The lower filter limit (closed interval) of the volume.
vol_max	int	The upper filter limit (closed interval) of the volume.
premium_min	float	The lower filter limit (closed interval) of premium value.

Field	Type	Description
premium_max	float	The upper filter limit (closed interval) of premium value.
leverage_ratio_min	float	The lower filter limit (closed interval) of the leverage ratio.
leverage_ratio_max	float	The upper filter limit (closed interval) of the leverage ratio.
delta_min	float	The lower filter limit (closed interval) of the hedge value Delta.
delta_max	float	The upper filter limit (closed interval) of the hedge value Delta.
implied_min	float	The lower filter limit (closed interval) of the implied volatility.
implied_max	float	The upper filter limit (closed interval) of the implied volatility.
recovery_price_min	float	The lower filter limit (closed interval) of the recovery price.
recovery_price_max	float	The upper filter limit (closed interval) of the recovery price.
price_recovery_ratio_min	float	The lower filter limit (closed interval) of the price recovery ratio.
price_recovery_ratio_max	float	The upper filter limit (closed interval) of the price recovery ratio.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, warrant data is returned.
	str	If ret != RET_OK, error description is returned.

  - Warrant data format as follows:

Field	Type	Description
warrant_data_list	pd.DataFrame	Warrant data after filtering.
last_page	bool	Weather is the last page.
all_count	int	The total number of warrants in the filtered result.

    - Warrant_data_list's detail as follows:

Field	Type	Description
stock	str	Warrant code.
stock_owner	str	Underlying stock.
type	WrtType	Warrant type.
issuer	Issuer	Issuer.
maturity_time	str	Maturity date.
list_time	str	Listing time.
last_trade_time	str	Last trading day.
recovery_price	float	Recovery price.

Field	Type	Description
conversion_ratio	float	Conversion ratio.
lot_size	int	Quantity per lot.
strike_price	float	Strike price.
last_close_price	float	Yesterday's close.
name	str	Name.
cur_price	float	Current price.
price_change_val	float	Price change.
status	WarrantStatus	Warrant status.
bid_price	float	Bid price.
ask_price	float	Ask price.
bid_vol	int	Bid volume.
ask_vol	int	Ask volume.
volume	unsigned int	Volume.
turnover	float	Turnover.
score	float	Comprehensive score.
premium	float	Premium. 
break_even_point	float	Breakeven point.
leverage	float	Leverage ratio.
ipop	float	In/out of the money. 
price_recovery_ratio	float	Price recovery ratio. 

Field	Type	Description
conversion_price	float	Conversion price.
street_rate	float	Outstanding percentage.
street_vol	int	Outstanding quantity.
amplitude	float	Amplitude.
issue_size	int	Issue size.
high_price	float	High.
low_price	float	Low.
implied_volatility	float	Implied volatility.
delta	float	Hedging value.
effective_leverage	float	Effective leverage.
upper_strike_price	float	Upper bound price.
lower_strike_price	float	Lower bound price.
inline_price_status	PriceType	In/out of bounds.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

req = WarrantRequest()
req.sort_field = SortField.TURNOVER
req.type_list = WrtType.CALL
req.cur_price_min = 0.1
req.cur_price_max = 0.2
ret, ls = quote_ctx.get_warrant("HK.00700", req)
if ret == RET_OK: # First judge whether the interface return is normal, and then
    warrant_data_list, last_page, all_count = ls
```

```
    print(len(warrant_data_list), all_count, warrant_data_list)
    print(warrant_data_list['stock'][0]) # Take the first warrant code
    print(warrant_data_list['stock'].values.tolist()) # Convert to list
else:
    print('error: ', ls)

req = WarrantRequest()
req.sort_field = SortField.TURNOVER
req.issuer_list = ['UB','CS','BI']
ret, ls = quote_ctx.get_warrant(Market.HK, req)
if ret == RET_OK:
    warrant_data_list, last_page, all_count = ls
    print(len(warrant_data_list), all_count, warrant_data_list)
else:
    print('error: ', ls)

quote_ctx.close()  # After using the connection, remember to close it to prevent
```

- Output

```
2 2
    stock        name stock_owner  type issuer maturity_time   list_time last_tra
0   HK.20306  MBTENCT@EC2012A    HK.00700  CALL     MB    2020-12-01  2019-06-27
1   HK.16545  SGTENCT@EC2102B    HK.00700  CALL     SG    2021-02-26  2020-07-14
HK.20306
['HK.20306', 'HK.16545']

200 358
    stock        name stock_owner    type issuer maturity_time   list_time last_t
0    HK.19839   PINGANRUIYINLINGYIGOUAC    HK.02318    CALL     UB    2020-12-31
1    HK.20084   PINGANZHONGYINLINGYIGOUAC    HK.02318    CALL     BI    2020-12-1
......
198  HK.56886   UB#HSI  RC2301F   HK.800000    BULL     UB    2023-01-30  2020-01
199  HK.56895   UB#XIAMIRC2012D    HK.01810    BULL     UB    2020-12-30  2020-01
```

Interface Limitations

- Hong Kong stock BMP permission does not support calling this API
- A maximum of 60 requests per 30 seconds
- The maximum number of data per request is 200

Get Related Data of a Specific Security

`get_referencestock_list(code, reference_type)`

- **Description**

  Get related data of securities, such as: obtaining warrants related to underlying stocks, obtaining contracts related to futures

- **Parameters**

Parameter	Type	Description
code	str	Stock code.
reference_type	SecurityReferenceType	Related data type to be obtained.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, related data of security is returned.
	str	If ret != RET_OK, error description is returned.

  - Related data of security fotmat as follows:

Field	Type	Description
code	str	Security code.
lot_size	int	The number of shares per lot, contract multiplier for futures.
stock_type	SecurityType	Security type.

Field	Type	Description
stock_name	str	Security name.
list_time	str	Time of listing.
wrt_valid	bool	Whether it is a warrant.
wrt_type	WrtType	Warrant type.
wrt_code	str	The underlying stock.
future_valid	bool	Whether it is a future.
future_main_contract	bool	Whether the future main contract.
future_last_trade_time	str	Last trading time.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

# Get warrants related to the underlying stock
ret, data = quote_ctx.get_referencestock_list('HK.00700', SecurityReferenceType.W
if ret == RET_OK:
    print(data)
    print(data['code'][0]) # Take the first stock code
    print(data['code'].values.tolist()) # Convert to list
else:
    print('error:', data)
print('**************************************')
# Port related contracts
ret, data = quote_ctx.get_referencestock_list('HK.A50main', SecurityReferenceType
if ret == RET_OK:
    print(data)
    print(data['code'][0]) # Take the first stock code
    print(data['code'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
        code  lot_size stock_type stock_name   list_time  wrt_valid wrt_type  wrt
0     HK.24719      1000    WARRANT     TENGXUNDONGYAJIUSIGUA  2018-07-20       T
...        ...       ...        ...        ...         ...        ...      ...
1617  HK.63402     10000    WARRANT     GS#TENCTRC2108Y  2020-11-26       True

[1618 rows x 11 columns]
HK.24719
['HK.24719', 'HK.27886', 'HK.28621', 'HK.14339', 'HK.27952', 'HK.18693', 'HK.203
...        ...       ...        ...        ...         ...        ...      ...
'HK.63402']
****************************************
        code  lot_size stock_type         stock_name list_time  wrt_valid  wrt_ty
0  HK.A50main      5000     FUTURE      A50 Future Main(DEC0)                Fals
..         ...       ...        ...                 ...       ...        ...
5  HK.A502106      5000     FUTURE      A50 JUN1               False       NaN

[6 rows x 11 columns]
HK.A50main
['HK.A50main', 'HK.A502011', 'HK.A502012', 'HK.A502101', 'HK.A502103', 'HK.A5021
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds
> - When obtaining warrants related to the underlying stock, it is not subject to the above frequency restriction

Get Futures Contract Information

`get_future_info(code_list)`

- **Description**

  Get futures contract information

- **Parameters**

Parameter	Type	Description
code_list	list	Futures code list. Data type of elements in the list is str.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, futures contract data is returned.
	str	If ret != RET_OK, error description is returned.

  - Futures contract data format as follows:

Field	Type	Description
code	str	Future code.
name	str	Future name.
owner	str	Subject.
exchange	str	Exchange.
type	str	Contract type.

Field	Type	Description
size	float	Contract size.
size_unit	str	Contract size unit.
price_currency	str	Quote currency.
price_unit	str	Price unit.
min_change	float	Price change step.
min_change_unit	str	Unit of price change step.
trade_time	str	Trading time.
time_zone	str	Time zone.
last_trade_time	str	The last trading time.
exchange_format_url	str	Exchange format url address.
origin_code	str	Original future code.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_future_info(["HK.MPImain", "HK.HAImain"])
if ret == RET_OK:
    print(data)
    print(data['code'][0]) # Take the first stock code
    print(data['code'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent
```

- Output

```
    code      name       owner exchange  type     size size_unit price_currency p
0  HK.MPImain  MPI Future Main(NOV0)    Hang Seng Mainland Properties Index     H
1  HK.HAImain  HAI Future Main(NOV0)    HK.06837     HKEX  Single Stock  10000.0
HK.MPImain
['HK.MPImain', 'HK.HAImain']
```

Interface Limitations

- A maximum of 30 requests for obtaining futures contract data interface every 30 seconds
- The maximum number of futures is 200, in the code list for each request.

Filter Stocks by Condition

```
get_stock_filter(market, filter_list, plate_code=None, begin=0, num=200)
```

- **Description**

  Filter stocks by condition

- **Parameters**

Parameter	Type	Description
market	Market	Market identifier.
filter_list	list	The list of filter conditions.
plate_code	str	Plate code.
begin	int	Data starting point.
num	int	The number of requested data.

  - The relevant parameters of the *SimpleFilter* object are as follows:

Field	Type	Description
stock_field	StockField	Simple filter properties.
filter_min	float	The lower limit of the interval (closed interval).
filter_max	float	The upper limit of the interval (closed interval).
is_no_filter	bool	Whether the field does not require filtering.
sort	SortDir	Sort direction.

  - The relevant parameters of the *AccumulateFilter* object are as follows:

Field	Type	Description
stock_field	StockField	Cumulative filter properties.
filter_min	float	The lower limit of the interval (closed interval).
filter_max	float	The upper limit of the interval (closed interval).
is_no_filter	bool	Whether the field does not require filtering.
sort	SortDir	Sort direction.
days	int	Accumulative days of filtering data.

- The relevant parameters of the *FinancialFilter* object are as follows:

Field	Type	Description
stock_field	StockField	Financial filter properties.
filter_min	float	The lower limit of the interval (closed interval).
filter_max	float	The upper limit of the interval (closed interval).
is_no_filter	bool	Whether the field does not require filtering.
sort	SortDir	Sort direction.
quarter	FinancialQuarter	Accumulation time of financial report.

- The relevant parameters of the *CustomIndicatorFilter* object are as follows:

Field	Type	Description
stock_field1	StockField	Custom indicator filter properties.
stock_field1_para	list	Custom indicator parameter.

Field	Type	Description
relative_position	RelativePosition	Relative position.
stock_field2	StockField	Custom indicator filter properties.
stock_field2_para	list	Custom indicator parameter.
value	float	Custom value.
ktype	KLType	K line type KLType (only supports K_60M, K_DAY, K_WEEK, K_MON four time periods).
consecutive_period	int	Filters data whose consecutive periods are all eligible.
is_no_filter	bool	Whether the field does not require filtering. True: no filtering, False: filtering. No filtering by default.

- The relevant parameters of the *PatternFilter* object are as follows:

Field	Type	Description
stock_field	StockField	Pattern filter properties.
ktype	KLType	K line type KLType (only supports K_60M, K_DAY, K_WEEK, K_MON four time periods).
consecutive_period	int	Filters data whose consecutive periods are all eligible.
is_no_filter	bool	Whether the field does not require filtering. True: no filtering, False: filtering. No filtering by default.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, stock selection data is returned.
	str	If ret != RET_OK, error description is returned.

- Stock selection data format as follows:

Field	Type	Description
last_page	bool	Is it the last page.
all_count	int	Total number of lists.
stock_list	list	Stock selection data.

  - *FilterStockData*'s data format as follows:

Field	Type	Description
stock_code	str	Stock code.
stock_name	str	Stock name.
cur_price	float	Current price.
cur_price_to_highest_52weeks_ratio	float	(Current price - high in 52 weeks)/high in 52 weeks.
cur_price_to_lowest_52weeks_ratio	float	(Current price - low in 52 weeks)/low in 52 weeks.
high_price_to_highest_52weeks_ratio	float	(Today's high - high in 52 weeks)/high in 52 weeks.
low_price_to_lowest_52weeks_ratio	float	(Today's low - low in 52 weeks)/low in 52 weeks.

Field	Type	Description
volume_ratio	float	Volume ratio.
bid_ask_ratio	float	The committee.
lot_price	float	Price per lot.
market_val	float	Market value.
pe_annual	float	P/E ratio.
pe_ttm	float	P/E ratio TTM.
pb_rate	float	P/B ratio.
change_rate_5min	float	Price change in five minutes.
change_rate_begin_year	float	Price change of this year.
ps_ttm	float	P/S rate TTM.
pcf_ttm	float	P/CF rate TTM.
total_share	float	Total number of shares.
float_share	float	Shares outstanding.
float_market_val	float	Market capitalization.
change_rate	float	Price change rate.
amplitude	float	Amplitude.
volume	float	Average daily volume.
turnover	float	Average daily turnover.
turnover_rate	float	Turnover rate.

Field	Type	Description
net_profit	float	Net profit.
net_profix_growth	float	Net profit growth rate.
sum_of_business	float	Operating income.
sum_of_business_growth	float	Year-on-year growth rate of operating income.
net_profit_rate	float	Net interest rate.
gross_profit_rate	float	Gross profit rate.
debt_asset_rate	float	Asset-liability ratio.
return_on_equity_rate	float	Return on net assets.
roic	float	Return on invested capital.
roa_ttm	float	Return on Assets TTM.
ebit_ttm	float	Earnings before interest and tax TTM.
ebitda	float	Earnings before interest and tax, depreciation and amortization.
operating_margin_ttm	float	Operating profit margin TTM.
ebit_margin	float	EBIT profit margin.
ebitda_margin	float	EBITDA profit margin.
financial_cost_rate	float	Financial cost rate.
operating_profit_ttm	float	Operating profit TTM.

Field	Type	Description
shareholder_net_profit_ttm	float	Net profit attributable to the parent company.
net_profit_cash_cover_ttm	float	Proportion of cash income in profit.
current_ratio	float	Current ratio.
quick_ratio	float	Quick ratio.
current_asset_ratio	float	Current asset ratio.
current_debt_ratio	float	Current debt ratio.
equity_multiplier	float	Equity multiplier.
property_ratio	float	Property ratio.
cash_and_cash_equivalents	float	Cash and cash equivalents.
total_asset_turnover	float	Total asset turnover rate.
fixed_asset_turnover	float	Fixed asset turnover rate.
inventory_turnover	float	Inventory turnover rate.
operating_cash_flow_ttm	float	Operating cash flow TTM.
accounts_receivable	float	Net accounts receivable.
ebit_growth_rate	float	EBIT year-on-year growth rate.
operating_profit_growth_rate	float	Operating profit year-on-year growth rate.
total_assets_growth_rate	float	Year-on-year growth rate of total assets.

Field	Type	Description
profit_to_shareholders_growth_rate	float	Year-on-year growth rate of net profit attributable to the parent.
profit_before_tax_growth_rate	float	Year-on-year growth rate of total profit.
eps_growth_rate	float	EPS year-on-year growth rate.
roe_growth_rate	float	ROE year-on-year growth rate.
roic_growth_rate	float	ROIC year-on-year growth rate.
nocf_growth_rate	float	Year-on-year growth rate of operating cash flow.
nocf_per_share_growth_rate	float	Year-on-year growth rate of operating cash flow per share.
operating_revenue_cash_cover	float	Operating cash income ratio.
operating_profit_to_total_profit	float	operating profit percentage.
basic_eps	float	Basic earnings per share.
diluted_eps	float	Diluted earnings per share.
nocf_per_share	float	Net operating cash flow per share.
price	float	latest price
ma	float	Simple moving average

Field	Type	Description
ma5	float	5-day simple moving average
ma10	float	10-day simple moving average
ma20	float	20-day simple moving average
ma30	float	30-day simple moving average
ma60	float	60-day simple moving average
ma120	float	120-day simple moving average
ma250	float	250-day simple moving average
rsi	float	RSI 
ema	float	exponential moving average 
ema5	float	5-day exponential moving average
ema10	float	10-day exponential moving average
ema20	float	20-day exponential moving average
ema30	float	30-day exponential moving average
ema60	float	60-day exponential moving average
ema120	float	120日-day exponential moving average

Field	Type	Description
ema250	float	250日-day exponential moving average
kdj_k	float	K value of KDJ indicator
kdj_d	float	D value of KDJ indicator
kdj_j	float	J value of KDJ indicator
macd_diff	float	DIFF value of MACD indicator
macd_dea	float	DEA value of MACD indicator
macd	float	MACD value of MACD indicator
boll_upper	float	UPPER value of BOLL indicator
boll_middler	float	MIDDLER value of BOLL indicator
boll_lower	float	LOWER value of BOLL indicator

- **Example**

```
from moomoo import *
import time

quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
simple_filter = SimpleFilter()
simple_filter.filter_min = 2
simple_filter.filter_max = 1000
simple_filter.stock_field = StockField.CUR_PRICE
simple_filter.is_no_filter = False
```

```
# simple_filter.sort = SortDir.ASCEND

financial_filter = FinancialFilter()
financial_filter.filter_min = 0.5
financial_filter.filter_max = 50
financial_filter.stock_field = StockField.CURRENT_RATIO
financial_filter.is_no_filter = False
financial_filter.sort = SortDir.ASCEND
financial_filter.quarter = FinancialQuarter.ANNUAL

custom_filter = CustomIndicatorFilter()
custom_filter.ktype = KLType.K_DAY
custom_filter.stock_field1 = StockField.KDJ_K
custom_filter.stock_field1_para = [10,4,4]
custom_filter.stock_field2 = StockField.KDJ_K
custom_filter.stock_field2_para = [9,3,3]
custom_filter.relative_position = RelativePosition.MORE
custom_filter.is_no_filter = False

nBegin = 0
last_page = False
ret_list = list()
while not last_page:
    nBegin += len(ret_list)
    ret, ls = quote_ctx.get_stock_filter(market=Market.HK, filter_list=[simple_f
    if ret == RET_OK:
        last_page, all_count, ret_list = ls
        print('all count = ', all_count)
        for item in ret_list:
            print(item.stock_code)  # Get the stock code
            print(item.stock_name)  # Get the stock name
            print(item[simple_filter])   # Get the value of the variable correspo
            print(item[financial_filter])   # Get the value of the variable corre
            print(item[custom_filter])  # Get the value of custom_filter
    else:
        print('error: ', ls)
        break
    time.sleep(3)  # Sleep for 3 seconds to avoid trigger frequency limitation

quote_ctx.close()  # After using the connection, remember to close it to prevent
```

- Output

```
39 39 [ stock_code:HK.08103  stock_name:hmvod Limited  cur_price:2.69  current_r
HK.08103
hmvod Limited
2.69
2.69
4.413
...
HK.00306
Kwoon Chung Bus
2.29
2.29
49.769
```

Tips

- Use Get sub-plate list function to get the sub-plate code, the plates supported by conditional stock selection are respectively
  1. The industry plate and concept plate of HK market.
  2. Industry plate of US market.
  3. Shanghai and Shenzhen's industry plate, conceptual plate and geographic plate.
- Supported plate index codes

Code	Description
HK.Motherboard	Main plate of HK market
HK.GEM	Growth Enterprise Market of HK market
HK.BK1911	Main plate of H-Share
HK.BK1912	Growth Enterprise Market of H-share
US.NYSE	New York Stock Exchange
US.AMEX	American Exchange
US.NASDAQ	NASDAQ
SH.3000000	Shanghai main plate

Code	Description
SZ.3000001	Shenzhen main plate
SZ.3000004	Shenzhen Growth Enterprise Market

Interface Limitations

- A maximum of 10 requests per 30 seconds
- At most 200 filter results are returned per page
- It is recommended that the filter conditions do not exceed 250, otherwise "business processing timeout did not return" may appear
- The maximum number of the same filter condition for cumulative filter properties is 10
- If you use dynamic data such as "current price" as the sorting field, the sorting of the data may change between multiple pages
- Non-similar indicators do not support comparison, and are limited to the establishment of comparison relationships between similar indicators, and comparisons across different types of indicators will cause errors. For example: MA5 and MA10 can establish a relationship. MA5 and EMA10 cannot establish a relationship.
- The same type of filter conditions of the custom indicator attribute exceeds the upper limit of 10
- Simple attributes, financial attributes, and morphological attributes do not support repeated designation of filter conditions for the same field
- Stock filter function currently does not support irregular trading hours (i.e.pre-market, post-market and overnight). All results are based on regular trading hours data.

Get the List of Stocks in The Plate

`get_plate_stock(plate_code, sort_field=SortField.CODE, ascend=True)`

- Description

  Get the list of stocks in the plate, or get the constituent stocks of the stock index

- Parameters

Parameter	Type	Description
plate_code	str	Plate code.
sort_field	SortField	Sort field.
ascend	bool	Sort direction.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, stock data of the plate is returned.
	str	If ret != RET_OK, error description is returned.

  - Stock data of the plate format as follows:

Field	Type	Description
code	str	Stock code.
lot_size	int	The number of shares per lot, or contract multiplier for futures.
stock_name	str	Stock name.

Field	Type	Description
stock_type	SecurityType	Stock type.
list_time	str	Time of listing.
stock_id	int	Stock ID.
main_contract	bool	Whether future main contract.
last_trade_time	str	Last trading time.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_plate_stock('HK.BK1001')
if ret == RET_OK:
    print(data)
    print(data['stock_name'][0]) # Take the first stock name
    print(data['stock_name'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent
```

- Output

```
    code  lot_size stock_name  stock_owner  stock_child_type stock_type   list_t
0   HK.00462      4000       Natural dairy          NaN              NaN      S
..       ...       ...        ...          ...                ...        ...
9   HK.06186      1000          China Feihe Limited          NaN              N

[10 rows x 10 columns]
Natural Dairy
['Natural Dairy', 'China Modern Dairy', 'Yashili International', 'YuanShengTai Da
```

> Interface Limitations

- A maximum of 10 requests per 30 seconds

▶ Commonly used sectors and index codes

Get Plate List

`get_plate_list(market, plate_class)`

- **Description**

  Obtain a list of stock sectors

- **Parameters**

Parameter	Type	Description
market	Market	Market identification.
plate_class	Plate	Plate classification.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, data of the plate list is returned.
	str	If ret != RET_OK, error description is returned.

  - Data of the plate list format as follows:

Field	Type	Description
code	str	Plate code.
plate_name	str	Plate name.
plate_id	str	Plate ID.

- **Example**

```python
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_plate_list(Market.HK, Plate.CONCEPT)
if ret == RET_OK:
    print(data)
    print(data['plate_name'][0]) # Take the first plate name
    print(data['plate_name'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
    code plate_name plate_id
0   HK.BK1000      Short Collection   BK1000
..        ...         ...      ...
77  HK.BK1999      Funeral Concept    BK1999

[78 rows x 3 columns]
Short Collection
['Short Collection','Ali concept stocks','Xiongan concept stocks','Apple concept
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds

Get Stock Basic Information

`get_stock_basicinfo(market, stock_type=SecurityType.STOCK, code_list=None)`

- **Description**

  Get Stock Basic Information

- **Parameters**

Parameter	Type	Description
market	Market	Market type.
stock_type	SecurityType	Stock type. It does not support SecurityType.DRVT.
code_list	list	Stock list.

  Note: when both *market* and *code_list* exist, *market* is ignored and only *code_list* is effective.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, stock static data is returned.
	str	If ret != RET_OK, error description is returned.

  - Stock static data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.

Field	Type	Description
lot_size	int	Number of shares per lot, number of shares per contract for options, contract multipliers for futures.
stock_type	SecurityType	Stock type.
stock_child_type	WrtType	Warrant type.
stock_owner	str	The code of the underlying stock to which the warrant belongs, or the code of the underlying stock of the option.
option_type	OptionType	Option type.
strike_time	str	The option exercise date.
strike_price	float	Option strike price.
suspension	bool	Whether the option is suspended.
listing_date	str	Listing time.
stock_id	int	Stock ID.
delisting	bool	Whether is delisted or not.
index_option_type	str	Index option type.
main_contract	bool	Whether is future main contract.
last_trade_time	str	Last trading time.
exchange_type	ExchType	Exchange Type.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
```

```python
ret, data = quote_ctx.get_stock_basicinfo(Market.HK, SecurityType.STOCK)
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
print('******************************************')
ret, data = quote_ctx.get_stock_basicinfo(Market.HK, SecurityType.STOCK, ['HK.069
if ret == RET_OK:
    print(data)
    print(data['name'][0]) # Take the first stock name
    print(data['name'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
        code             name  lot_size stock_type stock_child_type stock_owner c
0      HK.00001     CK Hutchison       500      STOCK              N/A
...         ...              ...       ...        ...              ...          .
2592   HK.09979     GREENTOWN MANAGEMENT HOLDINGS COMPANY LIMITED      1000

[2593 rows x 16 columns]
******************************************
        code            name  lot_size stock_type stock_child_type stock_owner op
0  HK.06998     JHBP       500      STOCK              N/A
1  HK.00700     Tencent       100      STOCK                 N/A
JHBP
['JHBP', 'Tencent']
```

Tips

- When input stocks are not recognized by the program (including stocks that have been delisted a long time ago and non-existent stocks), this interface still returns stock information. The "delisted" field is used to indicate that the stock does exist or not. The unified processing is: the code is displayed normally, the stock name is displayed as "unknown stock", and the other fields are default values (The integer type defaults to 0, and the string defaults to an empty string.).
- This interface is different from other market information interfaces. When other interfaces get input stocks that the program cannot recognize, they will reject the

request and return the error description "unknown stock".

Get IPO Information

`get_ipo_list(market)`

- **Description**

  Get IPO information of a specific market

- **Parameters**

Parameter	Type	Description
market	Market	Market identification.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, IPO data is returned.
	str	If ret != RET_OK, error description is returned.

  - IPO data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
list_time	str	Listing date, expected listing date for US stocks.
list_timestamp	float	Listing date timestamp, expected listing date timestamp for US stocks.

Field	Type	Description
apply_code	str	Subscription code (applicable to A-shares).
issue_size	int	Total number of issuance (applicable to A-shares); Total quantity of issuance (applicable to US stocks).
online_issue_size	int	Online issuance (applicable to A-shares).
apply_upper_limit	int	Subscription limit (applicable for A-shares).
apply_limit_market_value	int	The market value required for maximium subscription (applicable to A-shares).
is_estimate_ipo_price	bool	Weather to estimate the issuance price (applicable to A-shares).
ipo_price	float	Issuance price. (applicable to A-shares).
industry_pe_rate	float	Industry P/E ratio (applicable to A-shares).
is_estimate_winning_ratio	bool	Whether to estimate the winning rate (applicable to A-shares).
winning_ratio	float	Winning rate. (applicable to A-shares).
issue_pe_rate	float	Issue P/E ratio (applicable to A-shares).
apply_time	str	Subscription date string (applicable to A-shares).
apply_timestamp	float	Subscription date timestamp (applicable to A-shares).
winning_time	str	Time string of announcement date (applicable to A-shares).
winning_timestamp	float	Timestamp of announcement date (applicable to A-shares).

Field	Type	Description
is_has_won	bool	Whether the winning number has been announced (applicable to A-shares).
winning_num_data	str	The winning number (applicable to A-shares).
ipo_price_min	float	Lowest offer price (applicable to HK stocks); lowest issue price (applicable to US stocks).
ipo_price_max	float	Highest offer price (applicable to HK stocks); highest issue price (applicable to US stocks).
list_price	float	List price (applicable to HK stocks).
lot_size	int	Number of shares per lot.
entrance_price	float	Entrance fee (applicable to HK stocks).
is_subscribe_status	bool	Is it a subscription status.
apply_end_time	str	Subscription deadline string (applicable to HK stocks).
apply_end_timestamp	float	Subscription deadline timestamp.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_ipo_list(Market.HK)
if ret == RET_OK:
    print(data)
    print(data['code'][0]) # Take the first stock code
    print(data['code'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent
```

- Output

```
    code      name   list_time  list_timestamp apply_code issue_size online_issue
0  HK.06666  Evergrande Property Services Group Limited  2020-12-02    1.606838e-
1  HK.02110                    Yue Kan Holdings Limited  2020-12-07    1.607270e-
HK.06666
['HK.06666', 'HK.02110']
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds

Get global market status

`get_global_state()`

- **Description**

  Get global status

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	dict	If ret == RET_OK, global status is returned.
	str	If ret != RET_OK, error description is returned.

  - Global status format as follows:

Field	Type	Description
market_sz	MarketState	Shenzhen market state.
market_sh	MarketState	Shanghai market state.
market_hk	MarketState	Hong Kong market status.
market_hkfuture	MarketState	Hong Kong futures market status. 
market_usfuture	MarketState	US futures market status.
market_us	MarketState	United States market state. 
market_sgfuture	MarketState	Singapore futures market status. 

Field	Type	Description
market_jpfuture	MarketState	Japanese futures market status.
server_ver	str	OpenD version number.
trd_logined	bool	True: Logged into the trading server, False: Not logged into the trading server.
qot_logined	bool	True: logged into the market server, False: Not logged into the market server.
timestamp	str	Current Greenwich timestamp.
local_timestamp	float	Local timestamp for OpenD.
program_status_type	ProgramStatusType	Current status.
program_status_desc	str	Additional description.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
print(quote_ctx.get_global_state())
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
(0, {'market_sz': 'REST', 'market_us': 'AFTER_HOURS_END', 'market_sh': 'REST', 'r
```

Get Trading Calendar

```
request_trading_days(market=None, start=None, end=None, code=None)
```

- **Description**

  Request trading calendar via market or code.
  Note that the trading day is obtained by excluding weekends and holidays from natural days, and the temporary market closed data is not excluded.

- **Parameters**

Parameter	Type	Description
market	TradeDateMarket	Market type.
start	str	Start date.
end	str	End date.
code	str	Security code.

  Note: when both *market* and *code* exist, *market* is ignored and only *code* is effective.

  - The combination of ***start*** and ***end*** is as follows

Start type	End type	Description
str	str	***start*** and ***end*** are the specified dates respectively.
None	str	***start*** is 365 days before ***end***.
str	None	***end*** is 365 days after ***start***.
None	None	***start*** is 365 days before, ***end*** is the current date.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	list	If ret == RET_OK, data of the trading day is returned. Data type of elements in the list is dict.
	str	If ret != RET_OK, error description is returned.

- Data of the trading day's format as follows:

Field	Type	Description
time	str	Time.
trade_date_type	TradeDateType	Trading day type.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.request_trading_days(TradeDateMarket.HK, start='2020-04-01
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent
```

- **Output**

```
[{'time': '2020-04-01', 'trade_date_type': 'WHOLE'}, {'time': '2020-04-02', 'trad
```

> **Interface Limitations**
>
> - A maximum of 30 requests per 30 seconds
> - The historical trading calendar provides data for the past 10 years, and the future trading calendar is available until December 31 this year.

Get Details of Historical Candlestick Quota

`get_history_kl_quota(get_detail=False)`

- **Description**

  Get usage details of historical candlestick quota

- **Parameters**

Parameter	Type	Description
get_detail	bool	Whether to return the detailed record of historical candlestick pulled.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	tuple	If ret == RET_OK, historical candlestick quota is returned.
	str	If ret != RET_OK, error description is returned.

  - Historical candlestick quota format as follows:

Field	Type	Description
used_quota	int	Used quota.
remain_quota	int	Remaining quota.
detail_list	list	Detailed records of historical candlestick data pulled, including stock code and pulling time.

- detail_list, the data column format is as follows

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
request_time	str	The time string of the last pull.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_history_kl_quota(get_detail=True)  # Setting True means
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- **Output**

```
(2, 98,  {'code': 'HK.00123', 'name': 'YUEXIU PROPERTY', 'request_time': '2023-0
```

> **Interface Restrictions**
>
> - We will issue historical candlestick quotas based on the assets and tradings of your account. Therefore, you can only obtain historical candlestick data for a limited number of stocks within 30 days. For specific rules, please refer to Subscription Quota & Historical Candlestick Quota.
> - The historical candlestick quota you consume on that day will be automatically released after 30 days.

Set Price Reminder

`set_price_reminder(code, op, key=None, reminder_type=None, reminder_freq=None, value=None, note=None, reminder_session_list=NONE)`

- **Description**

  Add, delete, modify, enable, and disable price reminders for specified stocks

- **Parameters**

Parameter	Type	Description
code	str	Stock code
op	SetPriceReminderOp	Operation type.
key	int	Identification, do not need to fill in the case of adding all or deleting all.
reminder_type	PriceReminderType	The type of price reminder, this input parameter will be ignored when delete, enable, or disable.
reminder_freq	PriceReminderFreq	The frequency of price reminder, this input parameter will be ignored when delete, enabled, or disable.
value	float	Reminder value, the input parameter will be ignored when delete, enable, or disable.
note	str	The note set by the user, note supports no more than 20 Chinese characters, the input parameter will

Parameter	Type	Description
		be ignored when delete, enable, or disable.
reminder_session_list	list	The session for US stocks price reminder, this input parameter will be ignored when delete, enable, or disable.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
key	int	If ret == RET_OK, The price reminder key of the operation is returned. When deleting all reminders of a specific stock, 0 is returned.
	str	If ret != RET_OK, error description is returned.

- Example

```
from moomoo import *
import time
class PriceReminderTest(PriceReminderHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, content = super(PriceReminderTest,self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("PriceReminderTest: error, msg: %s" % content)
            return RET_ERROR, content
        print("PriceReminderTest ", content)
        return RET_OK, content
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = PriceReminderTest()
quote_ctx.set_handler(handler)
ret, data = quote_ctx.get_market_snapshot(['US.AAPL'])
if ret == RET_OK:
    bid_price = data['bid_price'][0] # Get real-time bid price
    ask_price = data['ask_price'][0] # Get real-time selling price
```

```
    # Set a reminder for AAPL(24H) when the selling price is lower than (ask_pri
    ret_ask, ask_data = quote_ctx.set_price_reminder(code='US.AAPL', op=SetPriceR
    if ret_ask == RET_OK:
        print('When the selling price is lower than (ask_price-1), remind that th
    else:
        print('error:', ask_data)
    # Set a reminder for AAPL(24H) when the bid price is higher than (bid_price+1
    ret_bid, bid_data = quote_ctx.set_price_reminder(code='US.AAPL', op=SetPriceR
    if ret_bid == RET_OK:
        print('When the bid price is higher than (bid_price+1), the reminder is s
    else:
        print('error:', bid_data)
time.sleep(15)
quote_ctx.close()
```

- **Output**

```
When the selling price is lower than (ask_price-1), the reminder is set successfu
When the bid price is higher than (bid_price+1), the reminder is set successfully
```

> **Tips**
>
> - Trading volume in API is based on shares. A-shares are shown in lots in moomoo Client.
> - The type of price alert has minimum precision, as follows:
>
>   TURNOVER_UP: The minimum precision of the turnover is 10 (Yuan, Hong Kong dollar, US dollar). The value passed in will be automatically rounded down to an integer multiple of the minimum precision. If you set 00700 transaction volume 102 yuan reminder, you will get 00700 transaction volume 100 yuan reminder. After setting; if you set 00700 transaction volume 8 yuan reminder, you will get 00700 transaction volume 0 yuan reminder after setting.
>
>   VOLUME_UP: The minimum accuracy of A-share trading volume is 1000 shares, and the minimum accuracy of other market stock trading volume is 10 shares. The value passed in will be automatically rounded down to an integer multiple of the minimum precision.

BID_VOL_UP, ASK_VOL_UP: The minimum precision for buying and selling of A-shares is 100 shares. The value passed in will be automatically rounded down to an integer multiple of the minimum precision.

The precision of the remaining price alert types supports up to 3 decimal places

Interface Limitations

- A maximum of 60 requests per 30 seconds
- The upper limit of reminders that can be set for each type of each stock is 10

Get Price Reminder List

`get_price_reminder(code=None, market=None)`

- **Description**

  Get a list of price reminders set for the specified stock or market

- **Parameters**

Parameter	Type	Description
code	str	Specified stock code.
market	Market	Specified market type.

  Note: Choose either code or market, and code takes precedence if both exist.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, price reminder data is returned.
	str	If ret != RET_OK, error description is returned.

  - Price reminder data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
key	int	Identification, used to modify the price reminder.

Field	Type	Description
reminder_type	PriceReminderType	The type of price reminder.
reminder_freq	PriceReminderFreq	The frequency of price reminder.
value	float	Remind value.
enable	bool	Whether to enable.
note	str	Note.
reminder_session_list	list	Price reminder session list for US stocks

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_price_reminder(code='US.AAPL')
if ret == RET_OK:
    print(data)
    print(data['key'].values.tolist())   # Convert to list
else:
    print('error:', data)
print('****************************************')
ret, data = quote_ctx.get_price_reminder(code=None, market=Market.US)
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # If the price remind list is not empty
        print(data['code'][0])    # Take the first stock code
        print(data['code'].values.tolist())   # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
code name                 key   reminder_type reminder_freq   value  enable note
0  US.AAPL   APPLE  1744021708234288125    BID_PRICE_UP        ALWAYS  184.37
```

```
1  US.AAPL   APPLE  1744022257052794489    BID_PRICE_UP        ALWAYS  185.50
2  US.AAPL   APPLE  1744021708211891867  ASK_PRICE_DOWN        ALWAYS  182.54
3  US.AAPL   APPLE  1744022257023211123  ASK_PRICE_DOWN        ALWAYS  183.70
[1744021708234288125, 1744022257052794489, 1744021708211891867, 1744022257023211
****************************************
      code name                 key   reminder_type reminder_freq   value  enabl
0  US.AAPL   APPLE  1744021708234288125    BID_PRICE_UP        ALWAYS  184.37
1  US.AAPL   APPLE  1744022257052794489    BID_PRICE_UP        ALWAYS  185.50
2  US.AAPL   APPLE  1744021708211891867  ASK_PRICE_DOWN        ALWAYS  182.54
3  US.AAPL   APPLE  1744022257023211123  ASK_PRICE_DOWN        ALWAYS  183.70
4  US.NVDA  NVIDIA  1739697581665326308      PRICE_DOWN        ALWAYS  102.00
US.AAPL
['US.AAPL', 'US.AAPL', 'US.AAPL', 'US.AAPL', 'US.NVDA']
```

Interface Limitations

- A maximum of 10 requests per 30 seconds

Get The Watchlist

`get_user_security(group_name)`

- **Description**

  Get a list of a specified group from watchlist

- **Parameters**

Parameter	Type	Description
group_name	str	The name of the specified group from watchlist.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, watchlist is returned.
	str	If ret != RET_OK, error description is returned.

  - Watchlist data format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.
lot_size	int	Number of shares per lot, number of shares per contract for options, contract multiplier for futures.
stock_type	SecurityType	Stock type.

Field	Type	Description
stock_child_type	WrtType	Warrant type.
stock_owner	str	The code of the underlying stock to which the warrant belongs, or the code of the underlying stock of the option.
option_type	OptionType	Option type.
strike_time	str	The option exercise date.
strike_price	float	Option strike price.
suspension	bool	Whether the option is suspended.
listing_date	str	Listing date.
stock_id	int	Stock ID.
delisting	bool	Whether is delisted.
main_contract	bool	Whether is future main contract.
last_trade_time	str	Last trading time.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_user_security("A")
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # If the user security list is not empty
        print(data['code'][0]) # Take the first stock code
        print(data['code'].values.tolist()) # Convert to list
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent
```

- Output

```
    code    name  lot_size stock_type stock_child_type stock_owner option_type st
0  HK.HSImain  HSI Future Main(NOV0)        50     FUTURE              N/A
1  HK.00700    Tencent Holdings       100      STOCK             N/A
HK.HSImain
['HK.HSImain', 'HK.00700']
```

Tips

The corresponding Chinese and English names of the system group are as follows

Chinese	English
全部	All
沪深	CN
港股	HK
美股	US
期权	Options
港股期权	HK options
美股期权	US options
特别关注	Starred
期货	Futures

Interface Limitations

- A maximum of 10 requests per 30 seconds
- Does not support position (Positions), fund treasure (Mutual Fund), foreign exchange (Forex) group query

Get Groups From Watchlist

`get_user_security_group(group_type = UserSecurityGroupType.ALL)`

- Description

  Get a list of groups from the user watchlist

- Parameters

Parameter	Type	Description
group_type	UserSecurityGroupType	Group type.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, group data of watchlist is returned.
	str	If ret != RET_OK, error description is returned.

  - Group data of watchlist format as follows:

Field	Type	Description
group_name	str	Group name.
group_type	UserSecurityGroupType	Group type.

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.get_user_security_group(group_type = UserSecurityGroupType
```

```
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- Output

```
        group_name group_type
0          Options     SYSTEM
..             ...        ...
12               C     CUSTOM

[13 rows x 2 columns]
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds

Modify the Watchlist

`modify_user_security(group_name, op, code_list)`

- **Description**

  Modify the specific group from the watchlist (you cannot modify the system group)

- **Parameters**

Parameter	Type	Description
group_name	str	The name of the group from the watchlist that needs to be modified.
op	ModifyUserSecurityOp	Operation type.
code_list	list	Stock list.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
msg	str	If ret == RET_OK, "success" is returned.
		If ret != RET_OK, error description is returned.

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)

ret, data = quote_ctx.modify_user_security("A", ModifyUserSecurityOp.ADD, ['HK.00
if ret == RET_OK:
    print(data) # Return success
else:
```

```
    print('error:', data)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

- **Output**

```
success
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds
> - You can only modify custom groups, not the system group
> - There is an upper limit on the number of "all" watchlist: 500 for untraded customers and 2000 for traded clients (when adding stocks to other groups, the "all" watchlist will also increase synchronously)
> - If there are multiple groups with a same name, the first group will be operated

Price Reminder Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  The price reminder notification callback, asynchronously handles the notification push that has been set to the price reminder. After receiving the real-time price notification, it will call back to this function. You need to override on_recv_rsp in the derived class.

- **Parameters**

Parameter	Type	Description
rsp_pb	Qot_UpdatePriceReminder_pb2.Response	This parameter does not need to be processed directly in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	dict	If ret == RET_OK, price reminder is returned.
	str	If ret != RET_OK, error description is returned.

  - Price reminder format as follows:

Field	Type	Description
code	str	Stock code.
name	str	Stock name.

Field	Type	Description
price	float	Current price.
change_rate	str	Current change rate.
market_status	PriceReminderMarketStatus	The time period for triggering.
content	str	Text content of price reminder.
note	str	Note.
key	int	Price reminder identification.
reminder_type	PriceReminderType	The type of price reminder.
set_value	float	The reminder value set by the user.
cur_value	float	The value when the reminder was triggered.

- Example

```
import time
from moomoo import *

class PriceReminderTest(PriceReminderHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, content = super(PriceReminderTest,self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("PriceReminderTest: error, msg: %s" % content)
            return RET_ERROR, content
        print("PriceReminderTest ", content) # PriceReminderTest's own processing
        return RET_OK, content
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = PriceReminderTest()
quote_ctx.set_handler(handler) # Set price reminder notification callback
time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
quote_ctx.close() # Close the current connection, OpenD will automatically cance
```

- Output

```
PriceReminderTest  {'code': 'US.AAPL', 'name': 'APPLE', 'price': 185.750, 'change
```

Tips

- This interface provides the function of continuously obtaining pushed data. If you need to obtain real-time data at one time, please refer to the Get Price Reminder List API.
- For the difference between get real-time data and real-time data callback, please refer to How to Get Real-time Quotes Through Subscription Interface API.

Quotation Definitions

Cumulative Filter Properties

> StockField

- `NONE`

  unknown

- `CHANGE_RATE`

  Yield 

- `AMPLITUDE`

  Amplitude 

- `VOLUME`

  Average daily trading volume 

- `TURNOVER`

  Average daily turnover 

- `TURNOVER_RATE`

  Turnover rate 

Asset Types

> AssetClass

- `UNKNOW`

  Unknown

- `STOCK`

  Stocks
- `BOND`

  Bonds
- `COMMODITY`

  Commodities
- `CURRENCY_MARKET`

  Currency markets
- `FUTURE`

  Futures
- `SWAP`

  Swaps

Corporate Action

Dark Disk Status

> **DarkStatus**

- `NONE`

  No grey market trading
- `TRADING`

  Ongoing grey market trading
- `END`

  Grey market trading finished

Financial Filter Properties

> StockField

- `NONE`

  unknown

- `NET_PROFIT`

  Net profit 

- `NET_PROFIX_GROWTH`

  Net profit growth rate 

- `SUM_OF_BUSINESS`

  Operating income 

- `SUM_OF_BUSINESS_GROWTH`

  Year-on-year growth rate of operating income 

- `NET_PROFIT_RATE`

  Net profit rate 

- `GROSS_PROFIT_RATE`

  Gross profit margin 

- `DEBT_ASSET_RATE`

  Asset-liability ratio 

- `RETURN_ON_EQUITY_RATE`

  Return on equity 

- `ROIC`

Return on invested capital

- `ROA_TTM`

  Return on assets TTM

- `EBIT_TTM`

  Earnings before interest and tax TTM

- `EBITDA`

  Earnings before interest, tax, depreciation and amortization

- `OPERATING_MARGIN_TTM`

  Operating profit margin TTM

- `EBIT_MARGIN`

  EBIT margin

- `EBITDA_MARGIN`

  EBITDA margin

- `FINANCIAL_COST_RATE`

  Financial cost rate

- `OPERATING_PROFIT_TTM`

  Operating profit TTM

- `SHAREHOLDER_NET_PROFIT_TTM`

  Net profit attributable to the parent company

- `NET_PROFIT_CASH_COVER_TTM`

  The proportion of cash income in profit

- `CURRENT_RATIO`

  Current ratio

- `QUICK_RATIO`

  Quick ratio 
- `CURRENT_ASSET_RATIO`

  Liquidity rate 
- `CURRENT_DEBT_RATIO`

  Current debt ratio 
- `EQUITY_MULTIPLIER`

  Equity multiplier 
- `PROPERTY_RATIO`

  Equity ratio 
- `CASH_AND_CASH_EQUIVALENTS`

  Cash and cash equivalent 
- `TOTAL_ASSET_TURNOVER`

  Total asset turnover rate 
- `FIXED_ASSET_TURNOVER`

  Fixed asset turnover rate 
- `INVENTORY_TURNOVER`

  Inventory turnover rate 
- `OPERATING_CASH_FLOW_TTM`

  Operating cash flow TTM 
- `ACCOUNTS_RECEIVABLE`

  Net accounts receivable 
- `EBIT_GROWTH_RATE`

Year-on-year growth rate of EBIT

- `OPERATING_PROFIT_GROWTH_RATE`

  Year-on-year growth rate of operating profit
- `TOTAL_ASSETS_GROWTH_RATE`

  Year-on-year growth rate of total assets
- `PROFIT_TO_SHAREHOLDERS_GROWTH_RATE`

  Year-on-year growth rate of net profit attributed to parent company owner
- `PROFIT_BEFORE_TAX_GROWTH_RATE`

  Year-on-year growth rate of total profit
- `EPS_GROWTH_RATE`

  Year-on-year growth rate of EPS
- `ROE_GROWTH_RATE`

  Year-on-year growth rate of ROE
- `ROIC_GROWTH_RATE`

  Year-on-year growth rate of ROIC
- `NOCF_GROWTH_RATE`

  Year-on-year growth rate of operating cash flow
- `NOCF_PER_SHARE_GROWTH_RATE`

  Year-on-year growth rate of operating cash flow per share
- `OPERATING_REVENUE_CASH_COVER`

  Operating cash cover ratio
- `OPERATING_PROFIT_TO_TOTAL_PROFIT`

  Operating profit ratio

- `BASIC_EPS`

  Basic earnings per share

- `DILUTED_EPS`

  Diluted earnings per share

- `NOCF_PER_SHARE`

  Net operating cash flow per share

Financial Filter Properties Period

> FinancialQuarter

- `NONE`

  unknown

- `ANNUAL`

  annual report

- `FIRST_QUARTER`

  First quarter report

- `INTERIM`

  Interim report

- `THIRD_QUARTER`

  Third quarter report

- `MOST_RECENT_QUARTER`

  Latest quarter report

Custom indicator attributes

> **StockField**

- `NONE`

  Unknown

- `PRICE`

  latest price

- `MA5`

  Simple moving average

- `MA5`

  5-day simple moving average (Not recommended)

- `MA10`

  10-day simple moving average (Not recommended)

- `MA20`

  20-day simple moving average (Not recommended)

- `MA30`

  30-day simple moving average (Not recommended)

- `MA60`

  60-day simple moving average (Not recommended)

- `MA120`

  120-day simple moving average (Not recommended)

- `MA250`

  250-day simple moving average (Not recommended)

- `RSI`

  RSI 

- `EMA`

  Exponential moving average

- `EMA5`

  5-day exponential moving average (Not recommended)

- `EMA10`

  10-day exponential moving average (Not recommended)

- `EMA20`

  20-day exponential moving average (Not recommended)

- `EMA30`

  30-day exponential moving average (Not recommended)

- `EMA60`

  60-day exponential moving average (Not recommended)

- `EMA120`

  120-day exponential moving average (Not recommended)

- `EMA250`

  250-day exponential moving average (Not recommended)

- `KDJ_K`

  K value of KDJ indicator 

- `KDJ_D`

  D value of KDJ indicator 

- `KDJ_J`

  J value of KDJ indicator 

- `MACD_DIFF`

DIFF value of MACD indicator

- `MACD_DEA`

  DEA value of MACD indicator

- `MACD`

  MACD value of MACD indicator

- `BOLL_UPPER`

  UPPER value of BOLL indicator

- `BOLL_MIDDLER`

  MIDDLER value of BOLL indicator

- `BOLL_LOWER`

  LOWER value of BOLL indicator

- `VALUE`

  Custom value (stock_field1 does not support this field)

Relative position

> **RelativePosition**

- `NONE`

  Unknown

- `MORE`

  Stock_field1 is greater than stock_field2

- `LESS`

  Stock_field1 is less than stock_field2

- `CROSS_UP`

Stock_field1 cross over stock_field2

- `CROSS_DOWN`

  Stock_field1 cross below stock_field2

Pattern attributes

> PatternField

- `NONE`

  未知

- `MA_ALIGNMENT_LONG`

  MA bullish alignment (MA5 > MA10 > MA20 > MA30 > MA60 for two consecutive days, and the closing price of the day is greater than the closing price of the previous day)

- `MA_ALIGNMENT_SHORT`

  MA bearish alignment (MA5 < MA10 < MA20 < MA30 < MA60 for two consecutive days, and the closing price of the day is less than the closing price of the previous day)

- `EMA_ALIGNMENT_LONG`

  EMA bullish alignment (EMA5 > EMA10 > EMA20 > EMA30 > EMA60 for two consecutive days, and the closing price of the day is greater than the closing price of the previous day)

- `EMA_ALIGNMENT_SHORT`

  EMA bearish alignment (EMA5 < EMA10 < EMA20 < EMA30 < MA60 for two consecutive days, and the closing price of the day is less than the closing price of the previous day)

- `RSI_GOLD_CROSS_LOW`

  RSI low golden cross (short-term RSI crosses over long-term RSI below 50 (short-term RSI of the previous day is less than long-term RSI, short-term RSI of the current day is greater than long-term RSI))

- `RSI_DEATH_CROSS_HIGH`

RSI high dead cross (short-term RSI crosses below long-term RSI above 50 (short-term RSI of the previous day is greater than long-term RSI, short-term RSI of the current day is less than long-term RSI))

- `RSI_TOP_DIVERGENCE`

  RSI top divergence (two adjacent candlestick peaks, the CLOSE of the later peak > the CLOSE of the earlier peak, the RSI12 value of the later peak < the RSI12 value of the earlier peak)

- `RSI_BOTTOM_DIVERGENCE`

  RSI bottom divergence (two adjacent candlestick troughs, the CLOSE of the later trough < the CLOSE of the earlier trough, the RSI12 value of the later trough > the RSI12 value of the earlier trough)

- `KDJ_GOLD_CROSS_LOW`

  KDJ low golden cross (D value is less than or equal to 30, and the K value of the previous day is less than the D value, and the K value of the day is greater than the D value)

- `KDJ_DEATH_CROSS_HIGH`

  KDJ high death cross (D value is greater than or equal to 70, and the K value of the previous day is greater than the D value, and the K value of the day is less than the D value)

- `KDJ_TOP_DIVERGENCE`

  KDJ top divergence (two adjacent candlestick peaks, the CLOSE of the later peak > the CLOSE of the earlier peak, the J value of the later peak < the J value of the earlier peak)

- `KDJ_BOTTOM_DIVERGENCE`

  KDJ bottom divergence (two adjacent candlestick troughs, the CLOSE of the later trough < the CLOSE of the earlier trough, the J value of the later trough > the J value of the earlier trough)

- `MACD_GOLD_CROSS_LOW`

  MACD golden cross (DIFF crosses over DEA (DIFF is less than DEA of the previous day, and DIFF is greater than DEA of the current day))

- `MACD_DEATH_CROSS_HIGH`

MACD dead cross (DIFF crosses below DEA (DIFF is greater than DEA of the previous day, and DIFF is less than DEA of the current day))

- `MACD_TOP_DIVERGENCE`

  MACD top divergence (two adjacent candlestick peaks, the CLOSE of the later peak > the CLOSE of the earlier peak, the MACD value of the later peak < the MACD value of the earlier peak)

- `MACD_BOTTOM_DIVERGENCE`

  MACD bottom deviation (two adjacent candlestick troughs, the CLOSE of the later trough < the CLOSE of the earlier trough, the MACD value of the later trough > the MACD value of the earlier trough)

- `BOLL_BREAK_UPPER`

  Break up bollinger upper bound (the stock price of the previous day was lower than the upper bound, and the stock price of the current day is greater than the upper bound)

- `BOLL_BREAK_LOWER`

  Break up bollinger lower bound (the stock price of the previous day was greater than the lower bound, and the stock price of the current day is less than the lower bound)

- `BOLL_CROSS_MIDDLE_UP`

  Cross over bollinger mid line (the stock price of the previous day was lower than the mid line, and the stock price of the current day is greater than the mid line)

- `BOLL_CROSS_MIDDLE_DOWN`

  Cross below bollinger mid line (the stock price of the previous day was greater than the mid line, and the stock price of the current day is less than the mid line)

Watchlist Group Type

> **UserSecurityGroupType**

- `NONE`

  unknown

- `CUSTOM`

  Custom groups

- `SYSTEM`

  System groups

- `ALL`

  All groups

Index Option Category

> **IndexOptionType**

- `NONE`

  unknown

- `NORMAL`

  Ordinary index option

- `SMALL`

  Small Index Options

Listing Time

> **IpoPeriod**

- `NONE`

  unknown

- `TODAY`

  Listed today

- `TOMORROW`

To be listed tomorrow

- `NEXTWEEK`

  To be listed next week

- `LASTWEEK`

  Has been listed last week

- `LASTMONTH`

  Has been listed last month

Warrant Issuer

> **Issuer**

- `UNKNOW`

  unknown

- `SG`

  Societe Generale

- `BP`

  BNP Paribas

- `CS`

  Credit Suisse

- `CT`

  Citi Bank

- `EA`

  The Bank of East Aisa

- `GS`

Goldman Sachs

- `HS`

  HSBC

- `JP`

  JPMorgan Chase

- `MB`

  Macquarie Bank

- `SC`

  Standard Chartered Bank

- `UB`

  Union Bank of Switzerland

- `BI`

  Bank of China

- `DB`

  Deutsche Bank

- `DC`

  Daiwa Bank

- `ML`

  Merrill Lynch

- `NM`

  Nomura Bank

- `RB`

  Rabobank

- `RS`

  The Royal Bank of Scotland
- `BC`

  Barclays
- `HT`

  Haitong Bank
- `VT`

  Bank Vontobel
- `KC`

  KBC Bank
- `MS`

  Morgan Stanley
- `GJ`

  Guotai Junan
- `XZ`

  DBS Bank
- `HU`

  Huatai
- `KS`

  Korea Investment
- `CI`

  CITIC Securities

Candlestick Field

> KL_FIELD

- `ALL`

  All

- `DATE_TIME`

  Time

- `HIGH`

  High

- `OPEN`

  Open

- `LOW`

  Low

- `CLOSE`

  Close

- `LAST_CLOSE`

  Close yesterday

- `TRADE_VOL`

  Volume

- `TRADE_VAL`

  Turnover

- `TURNOVER_RATE`

  Turnover rate

- `PE_RATIO`

  P/E ratio

- `CHANGE_RATE`

  Yield

Candlestick Type

> KLType

- `NONE`

  unknown

- `K_1M`

  1 minute candlestick

- `K_DAY`

  1 day candlestick

- `K_WEEK`

  1 week candlestick 

- `K_MON`

  1 month candlestick 

- `K_YEAR`

  1 year candlestick 

- `K_5M`

  5 minutes candlestick

- `K_15M`

  15 minutes candlestick

- `K_30M`

  30 minutes candlestick 

- `K_60M`

  60 minutes candlestick

- `K_3M`

  3 minutes candlestick

- `K_QUARTER`

  1 quarter candlestick

Period Type

> PeriodType

- `INTRADAY`

  Intraday

- `DAY`

  Day

- `WEEK`

  Week

- `MONTH`

  Month

Price Reminder Market Status

> PriceReminderMarketStatus

- `UNKNOW`

  unknown

- `OPEN`

Market opens

- `US_PRE`

  Pre-market of US stocks

- `US_AFTER`

  After-hours of US stocks

- `US_OVERNIGHT`

  Overnight trading session of US stocks

Watchlist Operation

> **ModifyUserSecurityOp**

- `NONE`

  Unknown

- `ADD`

  Add

- `DEL`

  Delete

- `MOVE_OUT`

  Remove from group

Option Type (by Exercise Time)

> **OptionAreaType**

- `NONE`

  unknown

- `AMERICAN`

  American Option

- `EUROPEAN`

  European Option

- `BERMUDA`

  Bermuda Option

Option in/out of The Money

> **OptionCondType**

- `ALL`

  All

- `WITHIN`

  In the money

- `OUTSIDE`

  Out of the money

Option Type (by Direction)

> **OptionType**

- `ALL`

  all

- `CALL`

  Call option

- `PUT`

Put option

Plate Set Type

> **Plate**

- `ALL`

  All plates

- `INDUSTRY`

  Industry plate

- `REGION`

  Regional plate 

- `CONCEPT`

  Concept plate

- `OTHER`

  Other plates 

Price Reminder Frequency

> **PriceReminderFreq**

- `NONE`

  Unknown

- `ALWAYS`

  Keep reminding

- `ONCE_A_DAY`

  Once a day

- `ONCE`

  Only remind once

Price Reminder Type

> **PriceReminderType**

- `NONE`

  Unknown

- `PRICE_UP`

  Price rise to

- `PRICE_DOWN`

  Price fall to

- `CHANGE_RATE_UP`

  Daily increase rate exceeds 

- `CHANGE_RATE_DOWN`

  Daily decline rate exceeds 

- `FIVE_MIN_CHANGE_RATE_UP`

  Increate rate in 5 minutes exceeds 

- `FIVE_MIN_CHANGE_RATE_DOWN`

  Decline rate in 5 minutes exceeds 

- `VOLUME_UP`

  Volume exceeds

- `TURNOVER_UP`

  Turnover exceeds

- `TURNOVER_RATE_UP`

  Turnover rate exceeds 
- `BID_PRICE_UP`

  Bid price higher than
- `ASK_PRICE_DOWN`

  Ask price lower than
- `BID_VOL_UP`

  Bid volume higher than
- `ASK_VOL_UP`

  Ask volume higher than
- `THREE_MIN_CHANGE_RATE_UP`

  Increate rate in 3 minutes exceeds 
- `THREE_MIN_CHANGE_RATE_DOWN`

  Decline rate in 3 minutes exceeds 

Warrant in/out of the Money

> **PriceType**

- `UNKNOW`

  Unknown
- `OUTSIDE`

  Out of the money
- `WITH_IN`

  In the money

Quote Push Type

> PushDataType

- `UNKNOW`

  Unknown

- `REALTIME`

  Real-time data

- `BYDISCONN`

  Pull supplementary data (up to 50) during disconnection from Futu server

- `CACHE`

  Non-real-time non-supplementary data

Quote Market

> Market

- `NONE`

  Unknown market

- `HK`

  HK market

- `US`

  US market

- `SH`

  Shanghai market

- `SZ`

  Shenzhen market

- `SG`

  Singapore market

- `JP`

  Japanese market

- `AU`

  Australian market

- `MY`

  Malaysian market

- `CA`

  Canadian market

- `FX`

  Forex market

Market State

> **MarketState**

Corresponding time period of each market state, click here to learn more

- `NONE`

  No trading

- `AUCTION`

  Pre-market trading

- `WAITING_OPEN`

Waiting for opening

- `MORNING`

  Morning session

- `REST`

  Lunch break

- `AFTERNOON`

  Afternoon session / Regular trading hours for U.S stock market

- `CLOSED`

  Market closed

- `PRE_MARKET_BEGIN`

  Pre-market trading of U.S stock market

- `PRE_MARKET_END`

  Pre-market ending of U.S stock market

- `AFTER_HOURS_BEGIN`

  After-hours trading of U.S stock market

- `AFTER_HOURS_END`

  Market closed of U.S. stock market

- `OVERNIGHT`

  Overnight trading of U.S. stock market

- `NIGHT_OPEN`

  Night market trading hours

- `NIGHT_END`

  Night market closed

- `NIGHT`

  Night market trading hours for U.S. index options

- `TRADE_AT_LAST`

  Late trading hours for U.S. index options

- `FUTURE_DAY_OPEN`

  Day market trading hours

- `FUTURE_DAY_BREAK`

  Day market break

- `FUTURE_DAY_CLOSE`

  Day market closed

- `FUTURE_DAY_WAIT_OPEN`

  Futures market wait for opening

- `HK_CAS`

  After-hours bidding for HK stocks

- `FUTURE_NIGHT_WAIT`

  Futures night market wait for opening (Obsolete)

- `FUTURE_AFTERNOON`

  Futures afternoon (Obsolete)

- `FUTURE_SWITCH_DATE`

  Waiting for U.S. futures opening

- `FUTURE_OPEN`

  Trading hours of U.S. futures

- `FUTURE_BREAK`

Break of U.S. futures

- `FUTURE_BREAK_OVER`

  Trading hours of U.S. futures after break

- `FUTURE_CLOSE`

  Market closed of U.S. futures

- `STIB_AFTER_HOURS_WAIT`

  After-hours matching period on the Sci-tech innovation plate (Obsolete)

- `STIB_AFTER_HOURS_BEGIN`

  After-hours trading on the Sci-tech innovation plate begins (Obsolete)

- `STIB_AFTER_HOURS_END`

  After-hours trading on the Sci-tech innovation plate ends (Obsolete)

US Stock Session

> Session

- `NONE`

  Unknown

- `RTH`

  US Stocks Regular trading hours

- `ETH`

  US Stocks Pre/Post + regular trading hours

- `OVERNIGHT`

  US Stocks Overnight trading hours (only applied to Trade API)

- `ALL`

US Stocks 24H trading hours (applied to Quote API & Trade API)

Quote Authorities

> **QotRight**

- `UNKNOW`

  Unknown

- `BMP`

  BMP (subscription is not supported for this permission)

- `LEVEL1`

  Level1

- `LEVEL2`

  Level2

- `SF`

  HK Securities FullTick Quotes

- `NO`

  No permission

Associated * Data Type

> **SecurityReferenceType**

- `UNKNOW`

  Unknown

- `WARRANT`

  Warrants for stocks

- `FUTURE`

  Contracts related to futures main

Candlestick Adjustment Type

> **AuType**

- `NONE`

  Actual

- `QFQ`

  Adjust forward

- `HFQ`

  Adjust backward

Stock Status

> **SecurityStatus**

- `NONE`

  Unknown

- `NORMAL`

  Normal status

- `LISTING`

  To be listed

- `PURCHASING`

  Purchasing

- `SUBSCRIBING`

Subscribing

- `BEFORE_DRAK_TRADE_OPENING`

  Before the grey market trading opens

- `DRAK_TRADING`

  Ongoing grey market trading

- `DRAK_TRADE_END`

  Grey market trading closed

- `TO_BE_OPEN`

  To be open

- `SUSPENDED`

  Suspended

- `CALLED`

  Called

- `EXPIRED_LAST_TRADING_DATE`

  Expired latest trading date

- `EXPIRED`

  Expired

- `DELISTED`

  Delisted

- `CHANGE_TO_TEMPORARY_CODE`

  During the company action, the trading was closed and transferred to the temporary code trading

- `TEMPORARY_CODE_TRADE_END`

Temporary trading ends

- `CHANGED_PLATE_TRADE_END`

  Plate changed, the old code is not available for trading

- `CHANGED_CODE_TRADE_END`

  The code has been changed, the old code is not available for trading

- `RECOVERABLE_CIRCUIT_BREAKER`

  Recoverable circuit breaker

- `UN_RECOVERABLE_CIRCUIT_BREAKER`

  Unrecoverable circuit breaker

- `AFTER_COMBINATION`

  After-hours matchmaking

- `AFTER_TRANSATION`

  After-hours trading

Stock Type

> **SecurityType**

- `NONE`

  Unknown

- `BOND`

  Bonds

- `BWRT`

  Blanket warrants

- `STOCK`

Stocks

- `ETF`

  ETFs

- `WARRANT`

  Warrants

- `IDX`

  Indexs

- `PLATE`

  Plates

- `DRVT`

  Options

- `PLATESET`

  Plate sets

- `FUTURE`

  Futures

Set Price Reminder Operation Type

> **SetPriceReminderOp**

- `NONE`

  Unknown

- `ADD`

  Add

- `DEL`

Delete

- `ENABLE`

  Enable

- `DISABLE`

  Disable

- `MODIFY`

  Modify

- `DEL_ALL`

  Delete all (delete all price alerts under the specified stock)

Sort Direction

> **SortDir**

- `NONE`

  Not sorted

- `ASCEND`

  Ascending

- `DESCEND`

  Descending

Sort Field

> **SortField**

- `NONE`

  Unknown

- `CODE`

  Code
- `CUR_PRICE`

  Latest price
- `PRICE_CHANGE_VAL`

  Price changed
- `CHANGE_RATE`

  Yield
- `STATUS`

  Status
- `BID_PRICE`

  Bid price
- `ASK_PRICE`

  Ask price
- `BID_VOL`

  Bid volume
- `ASK_VOL`

  Ask volume
- `VOLUME`

  Volume
- `TURNOVER`

  Turnover
- `AMPLITUDE`

Amplitude

- `SCORE`

  Comprehensive score

- `PREMIUM`

  Premium

- `EFFECTIVE_LEVERAGE`

  Effective leverage

- `DELTA`

  Hedging value 

- `IMPLIED_VOLATILITY`

  Implied volatility 

- `TYPE`

  Type

- `STRIKE_PRICE`

  Strike price

- `BREAK_EVEN_POINT`

  Break even point

- `MATURITY_TIME`

  Maturity date

- `LIST_TIME`

  Listing date

- `LAST_TRADE_TIME`

  Lastest trading day

- `LEVERAGE`

  Leverage ratio
- `IN_OUT_MONEY`

  In/out of the money %
- `RECOVERY_PRICE`

  Recovery price
- `CHANGE_PRICE`

  Change price
- `CHANGE`

  Change ratio
- `STREET_RATE`

  Outstanding percentage (the propotioin of retail investors)
- `STREET_VOL`

  Outstanding quantity (the volume held by retail investors)
- `WARRANT_NAME`

  Warrant name
- `ISSUER`

  Issuer
- `LOT_SIZE`

  Lot size
- `ISSUE_SIZE`

  Issue size
- `UPPER_STRIKE_PRICE`

Upper bound 

- `LOWER_STRIKE_PRICE`

  Lower bound 

- `INLINE_PRICE_STATUS`

  In/out of bounds 

- `PRE_CUR_PRICE`

  Latest price of pre-market

- `AFTER_CUR_PRICE`

  Latest price of after-hours

- `PRE_PRICE_CHANGE_VAL`

  Pre-market changes

- `AFTER_PRICE_CHANGE_VAL`

  After-hours changes

- `PRE_CHANGE_RATE`

  Pre-market change rate %

- `AFTER_CHANGE_RATE`

  After-hours change rate %

- `PRE_AMPLITUDE`

  Pre-market amplitude %

- `AFTER_AMPLITUDE`

  After-hours amplitude %

- `PRE_TURNOVER`

  Pre-market turnover

- `AFTER_TURNOVER`

  After-hours turnover

- `LAST_SETTLE_PRICE`

  Last settle price

- `POSITION`

  Position

- `POSITION_CHANGE`

  Daily increase of position

Simple Filter Properties

> StockField

- `NONE`

  unknown

- `STOCK_CODE`

  Stock code, does not accept list inputs as an interval

- `STOCK_NAME`

  Stock name, does not accept list inputs as an interval

- `CUR_PRICE`

  The latest price 

- `CUR_PRICE_TO_HIGHEST52_WEEKS_RATIO`

  (CP - WH52) / WH52
  CP: Current price
  WH52: 52-week high
  Corresponding to the “percentage from 52-week high” on the PC terminal 

- `CUR_PRICE_TO_LOWEST52_WEEKS_RATIO`

(CP - WL52) / WL52

CP: Current price

WL52: 52-week low

Corresponding to the “percentage from 52-week low” on the PC terminal

- `HIGH_PRICE_TO_HIGHEST52_WEEKS_RATIO`

  (TH - WH52) / WH52

  TH: Today's high

  WH52: 52-week high

- `LOW_PRICE_TO_LOWEST52_WEEKS_RATIO`

  (TL - WL52) / WL52

  TL: Today's low

  WL52: 52-week low

- `VOLUME_RATIO`

  Volume ratio

- `BID_ASK_RATIO`

  Bid-ask ratio

- `LOT_PRICE`

  Price per lot

- `MARKET_VAL`

  Market value

- `PE_ANNUAL`

  Trailing P/E

- `PE_TTM`

  P/E TTM

- `PB_RATE`

  P/B ratio

- `CHANGE_RATE_5MIN`

  Change rate in 5 minutes

- `CHANGE_RATE_BEGIN_YEAR`

  Price change rate from this year

- `PS_TTM`

  P/S TTM

- `PCF_TTM`

  PCF TTM

- `TOTAL_SHARE`

  Total number of shares

- `FLOAT_SHARE`

  Shares outstanding

- `FLOAT_MARKET_VAL`

  Market value outstanding

Subscription Type

> **SubType**

- `NONE`

  Unknown

- `QUOTE`

  Basic quote

- `ORDER_BOOK`

  Order book

- `TICKER`

  Tick-by-tick

- `RT_DATA`

  Time Frame

- `K_DAY`

  Daily candlesticks

- `K_5M`

  5 minutes candlesticks

- `K_15M`

  15 minutes candlesticks

- `K_30M`

  30 minutes candlesticks

- `K_60M`

  60 minutes candlesticks

- `K_1M`

  1 minute candlesticks

- `K_WEEK`

  Weekly candlesticks

- `K_MON`

  Monthly candlesticks

- `BROKER`

Broker's queue

- `K_QURATER`

  Seasonal candlesticks

- `K_YEAR`

  Annual candlesticks

- `K_3M`

  3 minutes candlesticks

Transaction Direction

> TickerDirect

- `NONE`

  unknown

- `BUY`

  Active buy 

- `SELL`

  Active sell 

- `NEUTRAL`

  Neutral transaction 

Tick-by-Tick Transaction Type

> TickerType

- `UNKNOWN`

  Unknown

- `AUTO_MATCH`

  Regular sale
- `LATE`

  Pre-market trade
- `NON_AUTO_MATCH`

  Non-regular sale
- `INTER_AUTO_MATCH`

  Regular sale for same broker
- `INTER_NON_AUTO_MATCH`

  Non-regular sale for same broker
- `ODD_LOT`

  Odd lot trade
- `AUCTION`

  Auction trade
- `BULK`

  Bunched trade
- `CRASH`

  Cash trade
- `CROSS_MARKET`

  Intermarket sweep
- `BULK_SOLD`

  Bunched sold trade
- `FREE_ON_BOARD`

Price variation trade

- `RULE127_OR155`

  Rule 127 (NYSE only) or Rule 155 (NYSE MKT only)

- `DELAY`

  Delay the transaction

- `MARKET_CENTER_CLOSE_PRICE`

  Market center close price

- `NEXT_DAY`

  Next day

- `MARKET_CENTER_OPENING`

  Market center opening trade

- `PRIOR_REFERENCE_PRICE`

  Prior reference price

- `MARKET_CENTER_OPEN_PRICE`

  Market center open price

- `SELLER`

  Seller

- `T`

  Form T(pre-open and post-close market trade)

- `EXTENDED_TRADING_HOURS`

  Extended trading hours/sold out of sequence

- `CONTINGENT`

  Contingent trade

- `AVERAGE_PRICE`

  Average price trade

- `OTC_SOLD`

  Sold(out of sequence)

- `ODD_LOT_CROSS_MARKET`

  Odd lot cross trade

- `DERIVATIVELY_PRICED`

  Derivatively priced

- `REOPENINGP_RICED`

  Re-Opening price

- `CLOSING_PRICED`

  Closing price

- `COMPREHENSIVE_DELAY_PRICE`

  Consolidated late price per listing packet

- `OVERSEAS`

  One party to the transaction is not a member of the Hong Kong Stock Exchange and is an over-the-counter transaction

Check The Market on The Trading Day

> **TradeDateMarket**

- `NONE`

  Unknown

- `HK`

  HK market

- `US`

  US market 
- `CN`

  A-share market
- `NT`

  Northbound Trading
- `ST`

  Southbound Trading
- `JP_FUTURE`

  Japanese future market
- `SG_FUTURE`

  Singapore future market

Type of Trading Day

> **TradeDateType**

- `WHOLE`

  Whole day trading
- `MORNING`

  Trading in the morning, closed in the afternoon
- `AFTERNOON`

  Trading in the afternoon, closed in the morning

Warrant Status

> **WarrantStatus**

- `NONE`

  Unknown

- `NORMAL`

  Normal status

- `SUSPEND`

  Suspended

- `STOP_TRADE`

  Stop trading

- `PENDING_LISTING`

  Waiting to be listed

Warrant Type

> **WrtType**

- `NONE`

  Unknown

- `CALL`

  Long warrants

- `PUT`

  Short warrants

- `BULL`

  Call warrants

- `BEAR`

Put warrants

- `INLINE`

  Inline Warrants

Exchange Type

> **ExchType**

- `NONE`

  Unknown

- `HK_MAINBOARD`

  HKEx·Main Board

- `HK_GEMBOARD`

  HKEx·GEM

- `HK_HKEX`

  HKEx

- `US_NYSE`

  NYSE

- `US_NASDAQ`

  NASDAQ

- `US_PINK`

  OTC Mkt

- `US_AMEX`

  AMEX

- `US_OPTION`

US

- `US_NYMEX`

  NYMEX
- `US_COMEX`

  COMEX
- `US_CBOT`

  CBOT
- `US_CME`

  CME
- `US_CBOE`

  CBOE
- `CN_SH`

  SH Stock Ex
- `CN_SZ`

  SZ Stock Ex
- `CN_STIB`

  STAR
- `SG_SGX`

  SGX
- `JP_OSE`

  OSE

Security Identification

Security

```
message Security
{
    required int32 market = 1; //QotMarket, quote market
    required string code = 2; //Code
}
```

Candlestick data

KLine

```
message KLine
{
    required string time = 1; //String of timestamp (Format: yyyy-MM-dd HH:mm:ss
    required bool isBlank = 2; //Whether it is a point with empty content, if it
    optional double highPrice = 3; //High
    optional double openPrice = 4; //Open
    optional double lowPrice = 5; //Low
    optional double closePrice = 6; //Close
    optional double lastClosePrice = 7; //Close yesterday
    optional int64 volume = 8; //Volume
    optional double turnover = 9; //Turnover
    optional double turnoverRate = 10; // Turnover rate (this field is in decima
    optional double pe = 11; //P/E ratio
    optional double changeRate = 12; //Yield (This field is in percentage form, s
    optional double timestamp = 13; //Timestamp
}
```

Option Specific Fields of The Underlying Quote

OptionBasicQotExData

```
message OptionBasicQotExData
{
    required double strikePrice = 1; //Strike price
    required int32 contractSize = 2; //Contract size (integer)
    optional double contractSizeFloat = 17; //Contract size (float)
```

```
    required int32 openInterest = 3; //Number of open positions
    required double impliedVolatility = 4; //Implied volatility (This field is i
    required double premium = 5; //Premium (This field is in percentage form, so
    required double delta = 6; //Greek value Delta
    required double gamma = 7; //Greek value Gamma
    required double vega = 8; //Greek value Vega
    required double theta = 9; //Greek value Theta
    required double rho = 10; //Greek value Rho
    optional int32 netOpenInterest = 11; //Net open contract number , only HK opt
    optional int32 expiryDateDistance = 12; //The number of days from the expiry
    optional double contractNominalValue = 13; //Contract nominal amount , only H
    optional double ownerLotMultiplier = 14; //Equal number of underlying stocks,
    optional int32 optionAreaType = 15; //OptionAreaType, option type (by exercis
    optional double contractMultiplier = 16; //Contract multiplier
    optional int32 indexOptionType = 18; //Qot_Common.IndexOptionType, index opti
}
```

Futures Specific Fields of The Base Quote

FutureBasicQotExData

```
message FutureBasicQotExData
{
    required double lastSettlePrice = 1; //Close yesterday
    required int32 position = 2; //Hold position
    required int32 positionChange = 3; //Daily change in position
    optional int32 expiryDateDistance = 4; //The number of days from the expirati
}
```

Basic Quotation

BasicQot

```
message BasicQot
{
    required Security security = 1; //Stock
    optional string name = 24; // stock name
    required bool isSuspended = 2; //whether trading is suspended
```

```
    required string listTime = 3; //listed date string (This field is deprecated
    required double priceSpread = 4; //Spread
    required string updateTime = 5; //Update time string of the latest price (Fo
    required double highPrice = 6; //High
    required double openPrice = 7; //Open
    required double lowPrice = 8; //low
    required double curPrice = 9; //The latest price
    required double lastClosePrice = 10; //Close yesterday
    required int64 volume = 11; //Volume
    required double turnover = 12; //Turnover
    required double turnoverRate = 13; //Turnover rate (This field is in percenta
    required double amplitude = 14; //Amplitude (This field is in percentage for
    optional int32 darkStatus = 15; //Grey market trading status
    optional OptionBasicQotExData optionExData = 16; //Option specific field
    optional double listTimestamp = 17; //Time stamp of listing date (This field
    optional double updateTimestamp = 18; //Update timestamp of the latest price
    optional PreAfterMarketData preMarket = 19; //Pre-market data
    optional PreAfterMarketData afterMarket = 20; //After-hours data
    optional int32 secStatus = 21; //Security status
    optional FutureBasicQotExData futureExData = 22; //Futures specific field
}
```

Before And After Data

PreAfterMarketData

```
//US stocks support pre-market and after-hours data
//The Sci-tech Innovation Plate only supports after-hours data: trading volume, t
message PreAfterMarketData
{
    optional double price = 1; //Pre-market or after-hours## Price
    optional double highPrice = 2; //Pre-market or after-hours## High
    optional double lowPrice = 3; //Pre-market or after-hours## Low
    optional int64 volume = 4; //Pre-market or after-hours## Volume
    optional double turnover = 5; //Pre-market or after-hours## Turnover
    optional double changeVal = 6; //Pre-market or after-hours## Change in price
    optional double changeRate = 7; //Pre-market or after-hours## Yield (This fie
    optional double amplitude = 8; //Pre-market or after-hours## Amplitude (This
}
```

Time Frame Data

TimeShare

```
message TimeShare
{
    required string time = 1; //Time string (Format: yyyy-MM-dd HH:mm:ss)
    required int32 minute = 2; //Minutes after 0 o'clock
    required bool isBlank = 3; //Whether the content is empty, if true, it conte
    optional double price = 4; //Current price
    optional double lastClosePrice = 5; //Close yesterday
    optional double avgPrice = 6; //Average price
    optional int64 volume = 7; //Volume
    optional double turnover = 8; //Turnover
    optional double timestamp = 9; //Timestamp
}
```

Basic Static Information of Securities

SecurityStaticBasic

```

message SecurityStaticBasic
{
    required Qot_Common.Security security = 1; //Stock
    required int64 id = 2; //Stock ID
    required int32 lotSize = 3; //Lot size, the option type represents the numbe
    required int32 secType = 4; //Qot_Common.SecurityType, stock type
    required string name = 5; //Stock name
    required string listTime = 6; //Listing time string (This field is deprecate
    optional bool delisting = 7; //Delisted or not
    optional double listTimestamp = 8; //Listing timestamp (This field is deprec
    optional int32 exchType = 9; //Qot_Common.ExchType, Exchange Type
}
```

Warrant Additional Static Information

WarrantStaticExData

```
message WarrantStaticExData
{
    required int32 type = 1; //Qot_Common.WarrantType, Warrant Type
    required Qot_Common.Security owner = 2; //The underlying stock
}
```

Option Additional Static Information

OptionStaticExData

```
message OptionStaticExData
{
    required int32 type = 1; //Qot_Common.OptionType, option
    required Qot_Common.Security owner = 2; //Underlying stock
    required string strikeTime = 3; //Exercise date (Format: yyyy-MM-dd)
    required double strikePrice = 4; //Strike price
    required bool suspend = 5; //Suspended or not
    required string market = 6; //Issuance market name
    optional double strikeTimestamp = 7; //Exercise date timestamp
    optional int32 indexOptionType = 8; //Qot_Common.IndexOptionType, type of inc
    optional int32 expirationCycle = 9; // Qot_Common.ExpirationCycle, type of op
    optional int32 optionStandardType = 10; // Qot_Common.OptionStandardType, typ
    optional int32 optionSettlementMode = 11; // OptionSettlementMode, mode of op
}
```

Additional Static Information About Futures

FutureStaticExData

```
message FutureStaticExData
{
    required string lastTradeTime = 1; //The lastest trading day, only future no
    optional double lastTradeTimestamp = 2; //The lastest trading day timestamp,
    required bool isMainContract = 3; //Futures main contract or not
}
```

Securities Static Information

SecurityStaticInfo

```
message SecurityStaticInfo
{
    required SecurityStaticBasic basic = 1; //Basic security information
    optional WarrantStaticExData warrantExData = 2; //Additional information for
    optional OptionStaticExData optionExData = 3; //Additional information for o
    optional FutureStaticExData futureExData = 4; //Additional information for f
}
```

Brokerage

Broker

```
message Broker
{
    required int64 id = 1; //Broker ID
    required string name = 2; //Broker name
    required int32 pos = 3; //Broker position

    //The following fields are specific to SF quote
    optional int64 orderID = 4; //Exchange order ID, which is different from the
    optional int64 volume = 5; //Number of shares in order
}
```

Tick-by-Tick

Ticker

```
message Ticker
{
    required string time = 1; //Time string (Format: yyyy-MM-dd HH:mm:ss)
    required int64 sequence = 2; //Unique identification
    required int32 dir = 3; //TickerDirection, buy or sell direction
    required double price = 4; //Price
    required int64 volume = 5; //Volume
    required double turnover = 6; // turnover
    optional double recvTime = 7; //Local timestamp of received push data, used t
    optional int32 type = 8; //TickerType, type by pen
    optional int32 typeSign = 9; //Pattern-by-stroke type sign
    optional int32 pushDataType = 10; //Used to distinguish push situations, this
    optional double timestamp = 11; //time stamp}
}
```

Transaction File Details

OrderBookDetail

```
message OrderBookDetail
{
    required int64 orderID = 1; //Exchange order ID, which is different from the
    required int64 volume = 2; //Number of shares in order
}
```

Order Book

OrderBook

```
message OrderBook
{
    required double price = 1; //Order price
    required int64 volume = 2; //Order quantity
    required int32 orederCount = 3; //Number of commissioned orders
    repeated OrderBookDetail detailList = 4; //Order information, unique to HK SE
}
```

Changes in Holdings

ShareHoldingChange

```
message ShareHoldingChange
{
    required string holderName = 1; //Holder name (institution name or fund name
    required double holdingQty = 2; //Current number of holdings
    required double holdingRatio = 3; //Current shareholding percentage (This fie
    required double changeQty = 4; //The number of changes from the previous time
    required double changeRatio = 5; //The percentage of change from the last tim
    required string time = 6; //Release time (Format: yyyy-MM-dd HH:mm:ss)
    optional double timestamp = 7; //Timestamp
}
```

Single Subscription Type Information

SubInfo

```
message SubInfo
{
    required int32 subType = 1; //Qot_Common.SubType, subscription type
    repeated Qot_Common.Security securityList = 2; //Subscribe to securities of t
}
```

Single Connection Subscription Information

ConnSubInfo

```
message ConnSubInfo
{
    repeated SubInfo subInfoList = 1; //The connection subscription information
    required int32 usedQuota = 2; //The subscription quota that the connection ha
    required bool isOwnConnData = 3; //Used to distinguish whether it is self-co
}
```

Plate Information

PlateInfo

```
message PlateInfo
{
    required Qot_Common.Security plate = 1; //Plate
    required string name = 2; //Plate name
    optional int32 plateType = 3; //Plate type, only 3207 (to get the plate to wh
}
```

Adjustment Information

Rehab

```
message Rehab
{
    required string time = 1; //Time string (Format: yyyy-MM-dd)
    required int64 companyActFlag = 2; //CompanyAct combination flag bit, which s
    required double fwdFactorA = 3; //Adjustments factor A
    required double fwdFactorB = 4; //Adjustments factor B
```

```
    required double bwdFactorA = 5; //Adjustments factor A
    required double bwdFactorB = 6; //Adjustments factor B
    optional int32 splitBase = 7; //Stock split (for example, 1 split 5, Base is
    optional int32 splitErt = 8;
    optional int32 joinBase = 9; //Reverse stock split (for example, 50 in 1, Bas
    optional int32 joinErt = 10;
    optional int32 bonusBase = 11; //Bonus shares (for example, 10 free 3, Base
    optional int32 bonusErt = 12;
    optional int32 transferBase = 13; //Transfer bonus shares (for example, 10 to
    optional int32 transferErt = 14;
    optional int32 allotBase = 15; //Allotment (for example, 10 get 2 free, allot
    optional int32 allotErt = 16;
    optional double allotPrice = 17;
    optional int32 addBase = 18; //Additional shares (for example, 10 get 2 free,
    optional int32 addErt = 19;
    optional double addPrice = 20;
    optional double dividend = 21; //Cash dividend (for example, if every 10 shar
    optional double spDividend = 22; //Special dividend (for example, if a specia
    optional double timestamp = 23; //Timestamp
}
```

> - For CompanyAct combination flag bit, refer to CompanyAct.

Expiration Cycle

> **ExpirationCycle**

- `NONE`

  Unknown

- `WEEK`

  Weekly options

- `MONTH`

  Monthly options

- `END_OF_MONTH`

End-Of-Monthly options

- `QUARTERLY`

  Quarterly options

- `WEEKMON`

  Monthly options-Monday

- `WEEKTUE`

  Monthly options-Tuesday

- `WEEKWED`

  Monthly options-Wednesday

- `WEEKTHU`

  Monthly options-Thursday

- `WEEKFRI`

  Monthly options-Friday

Option Standard Type

> **OptionStandardType**

- `NONE`

  Unknown

- `STANDARD`

  standard options

- `NON_STANDARD`

  non-standard options

Option Settlement Mode

> **OptionSettlementMode**

- `NONE`

  Unknown

- `AM`

  Asian Pricing

- `PM`

  Path-Dependent

Stockholder (Deprecated)

Stock Holder

> **StockHolder**

- `NONE`

  Unknown

- `INSTITUTE`

  Institute

- `FUND`

  Fund

- `EXECUTIVE`

  Executives

Overview

Module	Interface Name	Function Description
Account	get_acc_list	Get account list
	unlock_trade	Unlock trading
Asset and Position	accinfo_query	Get account financial information
	acctradinginfo_query	Get maximum tradable quantity
	position_list_query	Get positions list
	Trd_GetMarginRatio	Get margin data
	Get Cash Flow Summary	Get Account Cash Flow Data
Order	place_order	Place order
	modify_order	Modify or cancel order
	order_list_query	Get order list
	order_fee_query	Get order fees
	history_order_list_query	Get historical order list
	TradeOrderHandlerBase	Order callback
	SubAccPush	Trade data callback
Deal	deal_list_query	Get today's executed trades
	history_deal_list_query	Get historical executed trades
	TradeDealHandlerBase	Trade execution callback

Transaction Objects

Create the connection

`OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, host='127.0.0.1', port=11111, is_encrypt=None, security_firm=SecurityFirm.FUTUINC)`

`OpenFutureTradeContext(host='127.0.0.1', port=11111, is_encrypt=None, security_firm=SecurityFirm.FUTUINC)`

- **Description**

  According to the transaction variaties, select a correct account, and create its transaction object.

Transaction Objects	Accounts
OpenSecTradeContext	Securities account
OpenFutureTradeContext	Futures account

- **Parameters**

Parameter	Type	Description
filter_trdmarket	TrdMarket	Filter accounts according to the transaction market authority.
host	str	The IP listened by OpenD.
port	int	The port listened by OpenD.
is_encrypt	bool	Whether to enable encryption.
security_firm	SecurityFirm	Specified security firm

- **Example**

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
trd_ctx.close() # After using the connection, remember to close it to prevent the
```

Close the connection

`close()`

- **Description**

  Close the trading object. By default, the threads created inside the moomoo API will prevent the process from exiting, and the process can exit normally only after all Contexts are closed. But through set_all_thread_daemon, all internal threads can be set as daemon threads. At this time, even if close of Context is not called, the process can exit normally.

- **Example**

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
trd_ctx.close() # After using the connection, remember to close it to prevent the
```

Get the List of Trading Accounts

`get_acc_list()`

- **Description**

  Get a list of trading accounts. Before calling other trading interfaces, please obtain this list first and confirm that the trading account to be operated is correct.

- **Parameters**

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, trading account list is returned.
	str	If ret != RET_OK, error description is returned.

  - Trading account list format as follows:

Field	Type	Description
acc_id	int	Trading account.
trd_env	TrdEnv	Trading environment.
acc_type	TrdAccType	Account type.
uni_card_num	str	Universal account number, same as the display in the mobile terminal.
card_num	str	Trading account number
sim_acc_type	SimAccType	Simulate account type.

Field	Type	Description
security_firm	SecurityFirm	Securities firm to which the account belongs.
trdmarket_auth	list	Transaction market authority.
acc_status	TrdAccStatus	Account status.
acc_role	TrdAccRole	Account Structure
jp_acc_type	list	JP sub account type

- **Description**

  After the paper trading of HK/US stock options is opened, this function will return 2 paper trading accounts when obtaining the list of HK/US trading accounts. The first one is the original account, and the second one is the option paper trading account. Currently, the US paper trading accounts from OpenAPI are different with those showed on the mobile app. Click here for more details.

- **Example**

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
ret, data = trd_ctx.get_acc_list()
if ret == RET_OK:
    print(data)
    print(data['acc_id'][0])  # Get the first account ID
    print(data['acc_id'].values.tolist())  # convert to list
else:
    print('get_acc_list error: ', data)
trd_ctx.close()
```

- **Output**

```
              acc_id   trd_env acc_type       uni_card_num            card_num
0  281756420273981734      REAL   MARGIN  10018561211263256   1001100530724347
1             3450310  SIMULATE     CASH                N/A                N/A
2             3548732  SIMULATE   MARGIN                N/A                N/A
```

```
281756420273981734
[281756420273981734, 3450310, 3548732]
```

Unlock Trade

`unlock_trade(password=None, password_md5=None, is_unlock=True)`

- **Description**

  Lock or unlock trade

- **Parameters**

Parameter	Type	Description
password	str	Transaction password.
password_md5	str	32-bit MD5 encryption of transaction password (all lowercase).
is_unlock	bool	Lock or unlock.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
msg	NoneType	If ret == RET_OK, None is returned.
	str	If ret != RET_OK, error description is returned.

- **Example**

```
from moomoo import *
pwd_unlock = '123456'
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
ret, data = trd_ctx.unlock_trade(pwd_unlock)
if ret == RET_OK:
    print('unlock success!')
else:
```

```
8       print('unlock_trade failed: ', data)
9   trd_ctx.close()
```

- **Output**

```
1   unlock success!
```

> **TIP**
>
> - When using live trading accounts, you need to **unlock trade before** calling *Place Order* or *Modify or Cancel Orders* interface, but when using paper trading accounts, you do not need to **unlock trade**.
> - Locking or unlocking the transaction is an operation on OpenD. As long as one connection is unlocked, all other connections can call the transaction interface
> - It is strongly recommended that customers who connect to OpenD via the external network for live trading use encrypted channels, refer to **Enable protocol encryption**
> - OpenAPI does not support moomoo token. If you have activated moomoo token, it will fail to unlock. You need to turn off the token and then use OpenAPI to unlock.

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds under a single user ID.

Get Account Funds

```
accinfo_query(trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, refresh_cache=False, currency=Currency.HKD, asset_category=AssetCategory.NONE)
```

- Description

  Query fund data such as net asset value, securities market value, cash, and purchasing power of trading accounts.

- Parameters

Parameter	Type	Description
trd_env	TrdEnv	Trading environment
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
refresh_cache	bool	Whether to refresh the cache.
currency	Currency	The display currency of the funds.
asset_category	AssetCategory	Asset category

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, fund data is returned.
	str	If ret != RET_OK, error description is returned.

- Fund data format as follows:

Field	Type	Description
power	float	Maximum Buying Power.
max_power_short	float	Short Buying Power.
net_cash_power	float	Cash Buying Power.
total_assets	float	Total Net Assets.
securities_assets	float	Security Assets
fund_assets	float	Fund Assets
bond_assets	float	Bond Assets
cash	float	Cash.
market_val	float	Securities Market Value.
long_mv	float	Long Market Value.
short_mv	float	Short Market Value.
pending_asset	float	Asset in Transit.
interest_charged_amount	float	Interest Charged Amount.
frozen_cash	float	Funds on Hold.
avl_withdrawal_cash	float	Withdrawable Cash.
max_withdrawal	float	Maximum Withdrawal.
currency	Currency	The currency used for this query.
available_funds	float	Available funds.
unrealized_pl	float	Unrealized gain or loss.

Field	Type	Description
realized_pl	float	Realized gain or loss.
risk_level	CltRiskLevel	Risk control level.
risk_status	CltRiskStatus	Risk status.
initial_margin	float	Initial Margin.
margin_call_margin	float	Margin-call Margin.
maintenance_margin	float	Maintenance Margin.
hk_cash	float	HKD Cash.
hk_avl_withdrawal_cash	float	HKD Withdrawable Cash.
hkd_net_cash_power	float	HKD Cash Buying Power.
hkd_assets	float	HK Net Assets Value.
us_cash	float	USD Cash.
us_avl_withdrawal_cash	float	USD Withdrawable Cash.
usd_net_cash_power	float	USD Cash Buying Power.
usd_assets	float	US Net Assets Value.
cn_cash	float	CNH Cash.
cn_avl_withdrawal_cash	float	CNH Withdrawable Cash.
cnh_net_cash_power	float	CNH Cash Buying Power.
cnh_assets	float	CN Net Assets Value.
jp_cash	float	JPY Cash.
jp_avl_withdrawal_cash	float	JPY Withdrawable Cash.

Field	Type	Description
jpy_net_cash_power	float	JPY Cash Buying Power.
jpy_assets	float	JP Net Assets Value.
sg_cash	float	SGD Cash.
sg_avl_withdrawal_cash	float	SGD Withdrawable Cash.
sgd_net_cash_power	float	SGD Cash Buying Power.
sgd_assets	float	SG Net Assets Value.
au_cash	float	AUD Cash.
au_avl_withdrawal_cash	float	AUD Withdrawable Cash.
aud_net_cash_power	float	AUD Cash Buying Power.
aud_assets	float	AU Net Assets Value.
ca_cash	float	CAD Cash.
ca_avl_withdrawal_cash	float	CAD Withdrawable Cash.
cad_net_cash_power	float	CAD Cash Buying Power.
cad_assets	float	CA Net Assets Value.
my_cash	float	MYR Cash.
my_avl_withdrawal_cash	float	MYR Withdrawable Cash.
myr_net_cash_power	float	MYR Cash Buying Power.
myr_assets	float	MY Net Assets Value.
is_pdt	bool	Is it marked as a PDT.
pdt_seq	string	Day Trades Left.

Field	Type	Description
beginning_dtbp	float	Beginning DTBP.
remaining_dtbp	float	Remaining DTBP.
dt_call_amount	float	Day-trading Call Amount.
dt_status	DtStatus	Day-trading Status.

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
ret, data = trd_ctx.accinfo_query()
if ret == RET_OK:
    print(data)
    print(data['power'][0])  # Get the first buying power
    print(data['power'].values.tolist())  # convert to list
else:
    print('accinfo_query error: ', data)
trd_ctx.close()  # Close the current connection
```

- Output

- Output

```
power  max_power_short  net_cash_power  total_assets  securities_assets  fund_ass
0  465453.903307    465453.903307            0.0   289932.0404        197028.220
465453.903307
[465453.903307]
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds under a single account ID (acc_id)
> - It will be restricted by the frequency limit for this interface, only when refresh_cache is True

Query the Maximum Quantity that Can be Bought or Sold

```
acctradinginfo_query(order_type, code, price, order_id=None, adjust_limit=0, trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, session=Session.NONE, jp_acc_type=SubAccType.JP_GENERAL, position_id=NONE)
```

- **Description**

  Query the maximum quantity that can be bought or sold under a specifictrading account, and you can also query the maximum changeable quantity of a specific order under a specifictrading account.

  Cash account request options are not supported.

- **Parameters**

Parameter	Type	Description
order_type	OrderType	Order type.
code	str	Security code.
price	float	Quotation.
order_id	str	Order ID.
adjust_limit	float	Price adjustment range.
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
session	Session	US stocks Trading Session
jp_acc_type	SubAccType	JP sub account type

Parameter	Type	Description
position_id	int	Position ID

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, account list is returned.
	str	If ret != RET_OK, error description is returned.

  - Account list format as follows:

Field	Type	Description
max_cash_buy	float	Buy on cash.
max_cash_and_margin_buy	float	Buy on margin.
max_position_sell	float	Sell on position.
max_sell_short	float	Short sell.
max_buy_back	float	Short positions.
long_required_im	float	Initial margin change when buying one contract of an asset.
short_required_im	float	Initial margin change when selling one contract of an asset.
session	Session	Order session (Only applied to US stocks)

- **Example**

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', pc
ret, data = trd_ctx.acctradinginfo_query(order_type=OrderType.NORMAL, code='US.AA
if ret == RET_OK:
    print(data)
    print(data['max_cash_and_margin_buy'][0])  # Get maximum quantity that can be
else:
    print('acctradinginfo_query error: ', data)
trd_ctx.close()  # Close the current connection
```

- Output

```
    max_cash_buy  max_cash_and_margin_buy  max_position_sell  max_sell_short  max
0            0.0                  1500.0                0.0             0.0
1500.0
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds under a single account ID (acc_id).

> **Tips**
>
> - The cash account does not support trading derivatives, so it is unsupported to query options through the cash account.
> - Maximum quantity that can be bought for futures should be calculated by yourself. The formula is floor(Max buying power/Initial margin change when buying one contract of an asset). Max buying power is from Get Account Funds.

Get Positions

```
position_list_query(code='', position_market=TrdMarket.NONE, pl_ratio_min=None, pl_ratio_max=None, trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, refresh_cache=False, asset_category=AssetCategory.NONE)
```

- Description

  Query the holding position list of a specific trading account

- Parameters

Parameter	Type	Description
code	str	Security symbol.
position_market	TrdMarket	Filter positions by market.
pl_ratio_min	float	The lower limit of the current gain or loss ratio filter.
pl_ratio_max	float	The upper limit of the current gain or loss ratio filter.
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
refresh_cache	bool	Whether to refresh the cache.
asset_category	AssetCategory	Asset category

- Return

Field	Type	Description

ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, list of positions is returned.
	str	If ret != RET_OK, error description is returned.

- List of positions format as follows:

Field	Type	Description
position_side	PositionSide	Position direction.
code	str	Security code.
stock_name	str	Security name.
position_market	TrdMarket	Position market.
qty	float	The number of holdings.
can_sell_qty	float	Available quantity.
currency	Currency	Transaction currency.
nominal_price	float	Market price.
cost_price	float	Diluted Cost (for securities account). Average Cost (for futures account).
cost_price_valid	bool	Whether the cost price is valid.
average_cost	float	Average cost price
diluted_cost	float	Diluted cost price
market_val	float	Market value.
pl_ratio	float	Proportion of gain or loss(under diluted cost price)

Field	Type	Description
pl_ratio_valid	bool	Whether the gain or loss ratio is valid.
pl_ratio_avg_cost	float	Proportion of gain or loss(under average cost price)
pl_val	float	Gain or loss.
pl_val_valid	bool	Whether the gain or loss is valid.
today_pl_val	float	Gain or loss today.
today_trd_val	float	Transaction amount today.
today_buy_qty	float	Total volume purchased today.
today_buy_val	float	Total amount purchased today.
today_sell_qty	float	Total volume sold today.
today_sell_val	float	Total amount sold today.
unrealized_pl	float	Unrealized gain or loss.
realized_pl	float	Realized gain or loss.
position_id	int	Position ID

- Example

```
from futu import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', pc
ret, data = trd_ctx.position_list_query()
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # If the position list is not empty
        print(data['stock_name'][0])  # Get the first stock name of the holding p
        print(data['stock_name'].values.tolist())  # Convert to list
else:
```

```
    print('position_list_query error: ', data)
trd_ctx.close()  # Close the current connection
```

- Output

```
       code stock_name position_market    qty  can_sell_qty  cost_price  cost_pri
0  US.AAPL   Apple Inc.              US  400.0         400.0      53.975
Apple Inc.
['Apple Inc.']
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds under a single account ID (acc_id).
> - Call this interface, only when the cache is refreshed, will it be restricted by the frequency limit

Get Margin Data

`get_margin_ratio(code_list)`

- **Description**

  Query the margin data of stocks.

- **Parameters**

Parameter	Type	Description
code_list	list	Stock list.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, margin data is returned.
	str	If ret != RET_OK, error description is returned.

  - Margin data format as follows:

Field	Type	Description
code	str	Stock code
is_long_permit	bool	Is marginable trading allowed.
is_short_permit	bool	Is shortable trading allowed.
short_pool_remain	float	Short pool remaining.
short_fee_rate	float	Borrow rate.

Field	Type	Description
alert_long_ratio	float	Marginable alert margin.
alert_short_ratio	float	Shortable alert margin.
im_long_ratio	float	Marginable initial margin.
im_short_ratio	float	Shortable initial margin.
mcm_long_ratio	float	Marginable margin call margin.
mcm_short_ratio	float	Shortable margin call margin.
mm_long_ratio	float	Marginable maintenance margin.
mm_short_ratio	float	Marginable maintenance margin.

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
ret, data = trd_ctx.get_margin_ratio(code_list=['US.AAPL','US.FUTU'])
if ret == RET_OK:
    print(data)
     print(data['is_long_permit'][0])  # Get whether marginable trading allowed f
    print(data['im_short_ratio'].values.tolist())  # Convert to list
else:
    print('error:', data)
trd_ctx.close()  # After using the connection, remember to close it to prevent th
```

- Output

```
       code  is_long_permit  is_short_permit  short_pool_remain  short_fee_rate
0  US.AAPL             True             True          1826900.0            0.89
1  US.FUTU             True             True          1150600.0            0.95
True
[60.0, 50.0]
```

Interface Limitations

- A maximum of 10 requests per 30 seconds under a single user ID.
- For each request, the maximum number of stocks supported by the parameter is 100.
- Stocks and ETFs of US, HK and A-share markets are supported.

Get Account Cash Flow

`get_acc_cash_flow(clearing_date='', trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, cashflow_direction=CashFlowDirection.NONE)`

- Description

  Query the cash flow list of a specified trading account on a specified date. This includes all transactions that affect cash balances, such as deposits/withdrawals, fund transfers, currency exchanges, buying/selling financial assets, margin interest, and securities lending interest.

- Parameters

Parameter	Type	Description
clearing_date	str	Clearing date.
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
cashflow_direction	CashFlowDirection	Filter by the direction of cash flow (e.g., inflow/outflow).

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, account cash flow list is returned.
	str	If ret != RET_OK, error description is returned.

  - Account cash flow list format as follows:

Field	Type	Description
cashflow_id	int	Cash flow ID
clearing_date	str	Clearing date.
settlement_date	str	Settlement date.
currency	Currency	Transaction currency.
cashflow_type	str	Cash flow type.
cashflow_direction	CashFlowDirection	Cash flow direction.
cashflow_amount	float	Cash flow amount (positive:inflow, negative:outflow).
cashflow_remark	str	Remarks.

- Example

```
from futu import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
ret, data = trd_ctx.get_acc_cash_flow(clearing_date='2025-02-18', trd_env=TrdEnv
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # If account cash flow list is not empty
        print(data['cashflow_type'][0])  # Get direction of the first cash flow
        print(data['cashflow_amount'].values.tolist())  # Convert to list
else:
    print('get_acc_cash_flow error: ', data)
trd_ctx.close()

```

- Output

```
   cashflow_id     clearing_date     settlement_date     currency     cashflow_ty
0  16308           2025-02-27        2025-02-28          HKD             Others
1  16357           2025-02-27        2025-03-03          HKD             Others
2  16360           2025-02-27        2025-02-27          USD          Fund Redempt
```

```
3  16384          2025-02-27       2025-02-27          HKD          Fund Redempt
Others
[0.00, -104000.00, 23000.00, 104108.96]
```

Interface Limitations

- A maximum of 20 requests per 30 seconds under a single account ID (acc_id).
- Cash flows are arranged in chronological order.
- Can not query cash flow list through paper trading accounts and moomoo US accounts.

Place Orders

```
place_order(price, qty, code, trd_side, order_type=OrderType.NORMAL, adjust_limit=0, trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, remark=None, time_in_force=TimeInForce.DAY, fill_outside_rth=False, aux_price=None, trail_type=None, trail_value=None, trail_spread=None, session=Session.NONE, jp_acc_type=SubAccType.JP_GENERAL, position_id=NONE)
```

- **Description**

  Place order

  > **Tips**
  >
  > The Python API is synchronous, but the network transport is asynchronous. When the receiving time interval is very short between the response packet of place_order and Order Fill Push Callback or Order Push Callback, it may happen that the response packet of place_order returns first, but the callback function is called first. For example: Order Push Callback may be called first, and then the place_order interface returns.

- **Parameters**

Parameter	Type	Description
price	float	Order price.
qty	float	Order quantity.
code	str	Code.
trd_side	TrdSide	Transaction direction.
order_type	OrderType	Order type.
adjust_limit	float	Price adjustment range.

Parameter	Type	Description
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
remark	str	Remark.
time_in_force	TimeInForce	Valid period.
fill_outside_rth	bool	Whether allow to execute the order during pre-market or after-hours market trades.
aux_price	float	Trigger price.
trail_type	TrailType	Trailing type.
trail_value	float	Trailing amount/ratio.
trail_spread	float	Specify spread.
session	Session	US stocks Trading Session
jp_acc_type	SubAccType	JP sub account type
position_id	int	Position ID

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, order list is returned.
	str	If ret != RET_OK, error description is returned.

  - Order list format as follows:

Field	Type	Description
trd_side	TrdSide	Trading direction.
order_type	OrderType	Order type.
order_status	OrderStatus	Order status.
order_id	str	Order ID.
code	str	Security code.
stock_name	str	Security name.
qty	float	Order quantity.
price	float	Order price.
create_time	str	Create time.
updated_time	str	Last update time.
dealt_qty	float	Deal quantity
dealt_avg_price	float	Average deal price.
last_err_msg	str	The last error description.
remark	str	Identification of remarks when placing an order.
time_in_force	TimeInForce	Valid period.
fill_outside_rth	bool	Whether pre-market and after-hours are needed.
session	Session	Order session (Only applied to US stocks)
aux_price	float	Traget price.
trail_type	TrailType	Trailing type.

Field	Type	Description
trail_value	float	Trailing amount/ratio.
trail_spread	float	Specify spread.

- Example

```
from moomoo import *
pwd_unlock = '123456'
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
ret, data = trd_ctx.unlock_trade(pwd_unlock)  # If you use a live trading accoun
if ret == RET_OK:
    ret, data = trd_ctx.place_order(price=510.0, qty=100, code="US.AAPL", trd_si
    if ret == RET_OK:
        print(data)
        print(data['order_id'][0])  # Get the order ID of the placed order
        print(data['order_id'].values.tolist())  # Convert to list
    else:
        print('place_order error: ', data)
else:
    print('unlock_trade failed: ', data)
trd_ctx.close()
```

- Output

```

       code stock_name trd_side order_type order_status          order_id    qty
0  US.AAPL       Apple Inc.      BUY     NORMAL   SUBMITTING  38196006548709500
38196006548709500
['38196006548709500']
```

> **Interface Limitations**
>
> - A maximum of 15 requests per 30 seconds under a single account ID (acc_id), and the interval between two consecutive requests cannot be less than 0.02 seconds.

- When using live trading accounts, you need to **unlock trade** **before** calling *Place Order* interface, but when using paper trading accounts, you do not need to **unlock trade**.

Tips

- Required parameters for each order type: **Click here** to learn more.
- Each broker sets limits on shares or amounts for single orders of various trading products. Exceeding these limits may result in order failures: **Click here** to learn more.
- Locking position is not supported for **shortable securities**, that means you can not hold a long position and a short position at the same time.
- If you want to **close out position** of a **shortable securities**, you need to get the direction of the position and send an opposite order with the same quantity.
- If you want to **reversing trade** of a **shortable securities**, there are 2 steps: 1. you need to get the direction of the position and send an opposite order with the same quantity. 2. Send an opposite order again to open the reverse trade. For example: If you want to reverse trade of 1 long position of HK.HSI2012, you need to close the long position first and then sell short the contract.
- Only limit orders can be allowed during US stocks 24 Hour Trading Hour. You can choose Day, Good-Till-Cancelled (GTC) as the time-in-force. 24-hour order runs from Sunday 8:00 PM to Friday 8:00 PM ET, covering regular trading hours plus pre-market, post-market, and overnight trading sessions. You can place orders anytime during this period.
- Paper trading of US stocks does not support irregular trading hours (including pre/post-market and overnight).

Modify or Cancel Orders

`modify_order(modify_order_op, order_id, qty, price, adjust_limit=0, trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, aux_price=None, trail_type=None, trail_value=None, trail_spread=None)`

- **Description**

  Modify the price and quantity of orders, cancel orders, delete orders, enable or disable orders, etc.
  For HKCC market, it is invalid to change orders, except that cancelling orders is supported.

- **Parameters**

Parameter	Type	Description
modify_order_op	ModifyOrderOp	Modify order operation type.
order_id	str	Order ID.
qty	float	The quantity after the order is changed.
price	float	The price after the order is changed.
adjust_limit	float	Price adjustment range.
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
aux_price	float	Trigger price.
trail_type	TrailType	Trailing type.
trail_value	float	Trailing amount/ratio.

Parameter	Type	Description
trail_spread	float	Specify spread.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, modification information is returned.
	str	If ret != RET_OK, error description is returned.

  - Modification information format as follows:

Field	Type	Description
trd_env	TrdEnv	Trading environment.
order_id	str	Order ID.

- Example

```
from moomoo import *
pwd_unlock = '123456'
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
ret, data = trd_ctx.unlock_trade(pwd_unlock)  # If you use a live trading accoun
if ret == RET_OK:
    order_id = "8851102695472794941"
    ret, data = trd_ctx.modify_order(ModifyOrderOp.CANCEL, order_id, 0, 0)
    if ret == RET_OK:
        print(data)
        print(data['stock_name'][0])  # Get the order ID of the modified order
        print(data['stock_name'].values.tolist())  # Convert to list
    else:
        print('modify_order error: ', data)
else:
    print('unlock_trade failed: ', data)
trd_ctx.close()
```

- Output

```
    trd_env             order_id
0    REAL      8851102695472794941
8851102695472794941
['8851102695472794941']
```

`cancel_all_order(trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, trdmarket=TrdMarket.NONE)`

- Description

  Cancel all orders. Paper trading and HKCC trading accounts do not support all cancellations.

- Parameters

Parameter	Type	Description
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
trdmarket	TrdMarket	Transaction market selection.

- Return

Field	Type	Description
ret	int	Returned value. On success, ret == RET_OK. On error, ret != RET_OK.
data	str	If ret == RET_OK, modification information is returned.
		If ret != RET_OK, error description is returned.

  - Modification information format as follows:

Field	Type	Description
trd_env	TrdEnv	Trading environment
order_id	str	Order number

- Example

```
from moomoo import *
pwd_unlock = '123456'
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
ret, data = trd_ctx.unlock_trade(pwd_unlock)  # If you use a live trading accoun
if ret == RET_OK:
    ret, data = trd_ctx.cancel_all_order()
    if ret == RET_OK:
        print(data)
    else:
        print('cancel_all_order error: ', data)
else:
    print('unlock_trade failed: ', data)
trd_ctx.close()
```

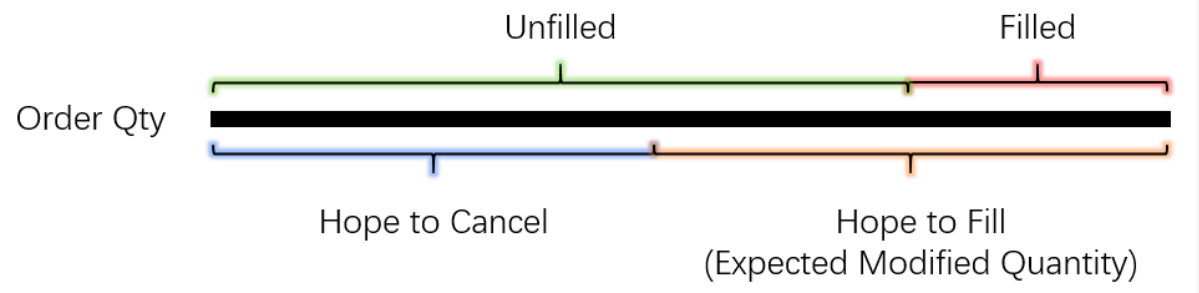
- Output

```
success
```

> **Interface Limitations**
>
> - A maximum of 20 requests per 30 seconds under a single account ID (acc_id), and the time interval between two consecutive requests should not be less than 0.04 seconds.
> - When using live trading accounts, you need to unlock trade **before** calling *Modify or Cancel Orders* interface, but when using paper trading accounts, you do not need to unlock trade.

> **Tip**

- For the execution of **modify the order**, to learn more about the required parameters for each order type, please **click here**.
- If you want to **modify the quantity of the order**, the parameter **qty** should be equal to the total quantity of the expected filled.
  For example:
  The quantity of an order is *N* shares, with *n* shares filled. For the unfilled *(N-n)* shares, if you want to cancel *x* shares, the parameter **modify_order_op** should be *NORMAL*, **qty** should be *(N-x)*.



- If you want to cancel an order, the parameter **modify_order_op** should be *CANCEL*.
  For example:
  The quantity of an order is *N* shares, with *n* shares filled. If you want to cancel the unfilled *(N-n)* shares, **modify_order_op** should be *CANCEL*, and **qty** and **price** will be ignored.

Get open Orders

`order_list_query(order_id="", order_market=TrdMarket.NONE, status_filter_list=[], code='', start='', end='', trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, refresh_cache=False)`

- Description

  Query the open order list of the specified trading account

- Parameters

Parameter	Type	Description
order_id	str	Order id.
order_market	TrdMarket	Filter orders by security market.
status_filter_list	list	Order status filter conditions.
code	str	Security symbol.
start	str	Start time.
end	str	End time.
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
refresh_cache	bool	Whether to refresh the cache.

- Return

Field	Type	Description

ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, order list is returned.
	str	If ret != RET_OK, error description is returned.

- Order list format as follows:

Field	Type	Description
trd_side	TrdSide	Trading direction.
order_type	OrderType	Order type.
order_status	OrderStatus	Order status.
order_id	str	Order ID.
code	str	Security code.
stock_name	str	Security name.
order_market	TrdMarket	Order market.
qty	float	Order quantity.
price	float	Order price.
currency	Currency	Transaction currency.
create_time	str	Create time.
updated_time	str	Last update time.
dealt_qty	float	Deal quantity
dealt_avg_price	float	Average deal price.
last_err_msg	str	The last error description.

Field	Type	Description
remark	str	Identification of remarks when placing an order.
time_in_force	TimeInForce	Valid period.
fill_outside_rth	bool	Whether pre-market and after-hours are needed.
session	Session	Order session (Only applied to US stocks)
aux_price	float	Traget price.
trail_type	TrailType	Trailing type.
trail_value	float	Trailing amount/ratio.
trail_spread	float	Specify spread.
jp_acc_type	SubAccType	JP sub account type

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
ret, data = trd_ctx.order_list_query()
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # If the order list is not empty
        print(data['order_id'][0])  # Get the first order ID of the order list to
        print(data['order_id'].values.tolist())  # Convert to list
else:
    print('order_list_query error: ', data)
trd_ctx.close()
```

- Output

```
         code stock_name order_market    trd_side          order_type   order_sta
0   US.AAPL                     US         BUY          NORMAL  CANCELLED_ALL  60
```

```
6644468615272262086
['6644468615272262086']
```

Interface Limitations

- A maximum of 10 requests per 30 seconds under a single account ID (acc_id).
- It will be restricted by the frequency limit for this interface only when the cache is refreshed

Tips

- Open orders are arranged in chronological order: earlier orders return first, followed by later orders.

Get Historical Orders

`history_order_list_query(status_filter_list=[], code='', order_market=TrdMarket.NONE, start='', end='', trd_env=TrdEnv.REAL, acc_id=0, acc_index=0)`

- **Description**

  Query the historical order list of a specified trading account

- **Parameters**

Parameter	Type	Description
status_filter_list	list	Order status filter conditions.
code	str	Security symbol.
order_market	TrdMarket	Filter orders by security market.
start	str	Start time.
end	str	End time
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.

  - The combination of ***start*** and ***end*** is as follows

Start type	End type	Description
str	str	***start*** and ***end*** are the specified dates respectively.
None	str	***start*** is 90 days before ***end***.

Start type	End type	Description
str	None	***end*** is 90 days after ***start***.
None	None	***start*** is 90 days before, ***end*** is the current date.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, order list is returned.
	str	If ret != RET_OK, error description is returned.

  - Order list format as follows:

Field	Type	Description
trd_side	TrdSide	Trading direction.
order_type	OrderType	Order type.
order_status	OrderStatus	Order status.
order_id	str	Order ID.
code	str	Security code.
stock_name	str	Security name.
order_market	TrdMarket	Order market.
qty	float	Order quantity.
price	float	Order price.
currency	Currency	Transaction currency.

Field	Type	Description
create_time	str	Create time.
updated_time	str	Last update time.
dealt_qty	float	Deal quantity
dealt_avg_price	float	Average deal price.
last_err_msg	str	The last error description.
remark	str	Identification of remarks when placing an order.
time_in_force	TimeInForce	Valid period.
fill_outside_rth	bool	Whether pre-market and after-hours are needed.
session	Session	Order session (Only applied to US stocks)
aux_price	float	Traget price.
trail_type	TrailType	Trailing type.
trail_value	float	Trailing amount/ratio.
trail_spread	float	Specify spread.
jp_acc_type	SubAccType	JP sub account type

- **Example**

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
ret, data = trd_ctx.history_order_list_query()
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # If the order list is not empty
        print(data['order_id'][0])  # Get Order ID of the first holding position
```

```
        print(data['order_id'].values.tolist())  # Convert to list
else:
    print('history_order_list_query error: ', data)
trd_ctx.close()
```

- Output

```
        code stock_name  order_market  trd_side          order_type   order_stat
0   US.AAPL                  US          BUY          NORMAL  CANCELLED_ALL  664
6644468615272262086
['6644468615272262086']
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds under a single account ID (acc_id).

> **Tips**
>
> - Historical orders are arranged in reverse chronological order: later orders return first, followed by earlier orders.

Orders Push Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  In response to orders push, asynchronously process the order status information pushed by OpenD. After receiving the order status information pushed by OpenD, this function is called.. You need to override on_recv_rsp in the derived class.

- **Parameters**

Parameter	Type	Description
rsp_pb	Trd_UpdateOrder_pb2.Response	This parameter does not need to be processed in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, order list is returned.
	str	If ret != RET_OK, error description is returned.

  - Order list format as follows:

Field	Type	Description
trd_side	TrdSide	Trading direction.
order_type	OrderType	Order type.
order_status	OrderStatus	Order status.

Field	Type	Description
order_id	str	Order ID.
code	str	Security code.
stock_name	str	Security name.
qty	float	Order quantity.
price	float	Order price.
currency	Currency	Transaction currency.
create_time	str	Create time.
updated_time	str	Last update time.
dealt_qty	float	Deal quantity
dealt_avg_price	float	Average deal price.
last_err_msg	str	The last error description.
remark	str	Identification of remarks when placing an order.
time_in_force	TimeInForce	Valid period.
fill_outside_rth	bool	Whether pre-market and after-hours are needed.
session	Session	Order session (Only applied to US stocks)
aux_price	float	Traget price.
trail_type	TrailType	Trailing type.
trail_value	float	Trailing amount/ratio.
trail_spread	float	Specify spread.

- Example

```
from moomoo import *
from time import sleep
class TradeOrderTest(TradeOrderHandlerBase):
    """ order update push"""
    def on_recv_rsp(self, rsp_pb):
        ret, content = super(TradeOrderTest, self).on_recv_rsp(rsp_pb)
        if ret == RET_OK:
            print("* TradeOrderTest content={}\n".format(content))
        return ret, content

trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
trd_ctx.set_handler(TradeOrderTest())
print(trd_ctx.place_order(price=518.0, qty=100, code="US.AAPL", trd_side=TrdSide

sleep(15)
trd_ctx.close()
```

- Output

```
* TradeOrderTest content=  trd_env      code stock_name  dealt_avg_price  dealt_
0    REAL  US.AAPL        Apple Inc.              0.0        0.0  100.0  7262526
```

Get Order Fee

`order_fee_query(order_id_list=[], acc_id=0, acc_index=0, trd_env=TrdEnv.REAL)`

- **介绍**

  Get specified orders' fee details. (Minimum version requirement: 8.2.4218)

- **参数**

Parameter	Type	Description
order_id_list	list	Order id list.
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.

- **返回**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, order fee list is returned.
	str	If ret != RET_OK, error description is returned.

  - Order list format as follows：

字段	类型	说明
order_id	str	Order ID
fee_amount	float	Total fee of the order.

字段	类型	说明
fee_details	list	Fee details of the order.

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
ret1, data1 = trd_ctx.history_order_list_query(status_filter_list=[OrderStatus.F
if ret1 == RET_OK:
    if data1.shape[0] > 0:  # If the order list is not empty
        ret2, data2 = trd_ctx.order_fee_query(data1['order_id'].values.tolist())
        if ret2 == RET_OK:
            print(data2)
            print(data2['fee_details'][0])  # Get fee details of the first order
        else:
            print('order_fee_query error: ', data2)
else:
    print('order_list_query error: ', data1)
trd_ctx.close()
```

- Output

```
                                          order_id  fee_amount
0  v3_20240314_12345678_MTc4NzA5NzY5OTA3ODAzMzMwN       10.46  [(Commission, 5.8
1  v3_20240318_12345678_MTM5Nzc5MDYxNDY1NDM1MDI1M        2.25  [(Commission, 0.9
[('Commission', 5.85), ('Platform Fee', 2.7), ('ORF', 0.11), ('OCC Fee', 0.18),
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds under a single account ID (acc_id).
> - Only orders after 2018-01-01 are supported.
> - Can not query order fee through paper trading accounts.
> - Can not query order fee through Moomoo CA accounts.

Subscribe to Transaction Push

Python does not need to subscribe to transaction push

Get Today's Deals

`deal_list_query(code="", deal_market= TrdMarket.NONE, trd_env=TrdEnv.REAL, acc_id=0, acc_index=0, refresh_cache=False)`

- Description

  Query today's deal list of a specific trading account.
  This feature is only available for live trading and not for paper trading.

- Parameters

Parameter	Type	Description
code	str	Security symbol.
deal_market	TrdMarket	Filter deals by security market.
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.
refresh_cache	bool	Whether to refresh the cache.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, transaction list is returned.
	str	If ret != RET_OK, error description is returned.

  - Transaction list format as follows:

Field	Type	Description
trd_side	TrdSide	Trading direction.
deal_id	str	Deal number.
order_id	str	Order ID.
code	str	Security code.
stock_name	str	Security name.
deal_market	TrdMarket	Deal market.
qty	float	Quantity of shares bought/sold on this fill.
price	float	Fill price.
create_time	str	Create time.
counter_broker_id	int	Counter broker ID.
counter_broker_name	str	Counter broker name.
status	DealStatus	Deal status.
jp_acc_type	SubAccType	JP sub account type

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', pc
ret, data = trd_ctx.deal_list_query()
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # If the order fill list is not empty
        print(data['order_id'][0])  # Get the first order ID of the transaction
        print(data['order_id'].values.tolist())  # Convert to list
else:
```

```
    print('deal_list_query error: ', data)
trd_ctx.close()
```

- Output

```
    code stock_name                             deal_market     deal_id
0  US.AAPL       Apple Inc.                       HK  5056208452274069375  466529
4665291631090960915
['4665291631090960915']
```

Interface Limitations

- A maximum of 10 requests per 30 seconds under a single account ID (acc_id).
- It will be restricted by the frequency limit for this interface only when refresh_cache is True

Tips

- Today's deals are arranged in chronological order: earlier deals return first, followed by later deals.

Get Historical Deals

`history_deal_list_query(code='', deal_market=TrdMarket.NONE, start='', end='', trd_env=TrdEnv.REAL, acc_id=0, acc_index=0)`

- **Description**

  Query historical deal list of a specific trading account.
  This feature is only available for live trading and not for paper trading.

- **Parameters**

Parameter	Type	Description
code	str	Security symbol.
deal_market	TrdMarket	Filter deals by security market.
start	str	Start time.
end	str	End time.
trd_env	TrdEnv	Trading environment.
acc_id	int	Trading account ID.
acc_index	int	The account number in the trading account list.

  - The combination of ***start*** and ***end*** is as follows

Start type	End type	Description
str	str	***start*** and ***end*** are the specified dates respectively.
None	str	***start*** is 90 days before ***end***.
str	None	***end*** is 90 days after ***start***.

Start type	End type	Description
None	None	***start*** is 90 days before, ***end*** is the current date.

- Return

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, transaction list is returned.
	str	If ret != RET_OK, error description is returned.

  - Transaction list format as follows:

Field	Type	Description
trd_side	TrdSide	Trading direction.
deal_id	str	Deal number.
order_id	str	Order ID.
code	str	Security code.
stock_name	str	Security name.
deal_market	TrdMarket	Deal market.
qty	float	Quantity of shares bought/sold on this fill.
price	float	Fill price.
create_time	str	Create time.
counter_broker_id	int	Counter broker ID.
counter_broker_name	str	Counter broker name.

Field	Type	Description
status	DealStatus	Deal status.
jp_acc_type	SubAccType	JP sub account type

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
ret, data = trd_ctx.history_deal_list_query()
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # If the order fill list is not empty
        print(data['deal_id'][0])  # Get the first deal ID of the history order
        print(data['deal_id'].values.tolist())  # Convert to list
else:
    print('history_deal_list_query error: ', data)
trd_ctx.close()  # Close the current connection
```

- Output

```
    code      stock_name      deal_market      deal_id             order_id      
0  US.AAPL   Apple Inc.           US  5056208452274069375  4665291631090960915  
5056208452274069375
['5056208452274069375']
```

> **Interface Limitations**
>
> - A maximum of 10 requests per 30 seconds under a single account ID (acc_id).

> **Tips**
>
> - Historical deals are arranged in reverse chronological order: later deals return first, followed by earlier deals.

Deals Push Callback

`on_recv_rsp(self, rsp_pb)`

- **Description**

  In response to the transaction push, asynchronously process the transaction status information pushed by OpenD. After receiving the order fill information pushed by OpenD, this function is called. You need to override on_recv_rsp in the derived class.
  This feature is only available for live trading and not for paper trading.

- **Parameters**

Parameter	Type	Description
rsp_pb	Trd_UpdateOrderFill_pb2.Response	This parameter does not need to be processed in the derived class.

- **Return**

Field	Type	Description
ret	RET_CODE	Interface result.
data	pd.DataFrame	If ret == RET_OK, transaction list is returned.
	str	If ret != RET_OK, error description is returned.

  - Transaction list format as follows:

Field	Type	Description
trd_side	TrdSide	Trading direction.
deal_id	str	Deal number.

Field	Type	Description
order_id	str	Order ID.
code	str	Security code.
stock_name	str	Security name.
qty	float	Quantity of shares bought/sold on this fill. 
price	float	Fill price.
create_time	str	Create time. 
counter_broker_id	int	Counter broker ID. 
counter_broker_name	str	Counter broker name. 
status	DealStatus	Deal status.

- Example

```
from moomoo import *
from time import sleep
class TradeDealTest(TradeDealHandlerBase):
    """ order update push"""
    def on_recv_rsp(self, rsp_pb):
        ret, content = super(TradeDealTest, self).on_recv_rsp(rsp_pb)
        if ret == RET_OK:
            print("TradeDealTest content={}".format(content))
        return ret, content

trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
trd_ctx.set_handler(TradeDealTest())
print(trd_ctx.place_order(price=595.0, qty=100, code="US.AAPL", trd_side=TrdSide

sleep(15)
trd_ctx.close()
```

- Output

```
TradeDealTest content=  trd_env      code stock_name             deal_id
0    REAL  US.AAPL      Apple Inc.  2511067564122483295  8561504228375901919  100
```

Trading Definitions

Account Risk Control Level

> CltRiskLevel

- `NONE`

  Unknown

- `SAFE`

  Safe

- `WARNING`

  Warning

- `DANGER`

  Danger

- `ABSOLUTE_SAFE`

  Absolutely safe

- `OPT_DANGER`

  Danger 

> **Tips**
>
> - It is recommanded to use risk_status field to get risk status of futures account, see CltRiskStatus

Currency Type

> **Currency**

- `NONE`

  Unknown currency

- `HKD`

  HK dollar

- `USD`

  US dollar

- `CNH`

  Offshore RMB

- `JPY`

  Japanese Yen

- `SGD`

  SG dollar

- `AUD`

  Australian dollar

- `CAD`

  Canadian dollar

- `MYR`

  Malaysian Ringgit

TrailType

TrailType

- `NONE`

Unknown

- `RATIO`

  Trailing ratio

- `AMOUNT`

  Trailing amount

Modify Order Operation

> **ModifyOrderOp**

- `NONE`

  Unknown operation

- `NORMAL`

  Modify order

- `CANCEL`

  Cancel 

- `DISABLE`

  Disable 

- `ENABLE`

  Enable 

- `DELETE`

  Delete 

Transaction Status

> **DealStatus**

- `OK`

  Transaction success

- `CANCELLED`

  Transaction cancelled

- `CHANGED`

  Transaction changed

Order Status

> **OrderStatus**

- `NONE`

  Unknown status

- `WAITING_SUBMIT`

  Queued 

- `SUBMITTING`

  Submitting 

- `SUBMITTED`

  Working 

- `FILLED_PART`

  Partially filled 

- `FILLED_ALL`

  Fully filled

- `CANCELLED_PART`

  Partially cancelled

- `CANCELLED_ALL`

  Fully cancelled

- `FAILED`

  Failed. Rejected by server.

- `DISABLED`

  Deactivated 

- `DELETED`

  Deleted, only unfilled orders can be deleted 

Order Type

> **Tips**
>
> - Order types supported in live trading.
> - Paper trade only supports limit orders (NORMAL) and market orders (MARKET).

> **OrderType**

- `NONE`

  Unknown type.

- `NORMAL`

  Limit orders.

- `MARKET`

  Market orders.

- `ABSOLUTE_LIMIT`

  Absolute limit orders. 

- `AUCTION`

At-auction market orders.

- `AUCTION_LIMIT`

  At-auction limit orders.

- `SPECIAL_LIMIT`

  Special limit orders.

- `SPECIAL_LIMIT_ALL`

  AON special limit orders.

- `STOP`

  Stop orders.

- `STOP_LIMIT`

  Stop Limit orders.

- `MARKET_IF_TOUCHED`

  Market if Touched orders.

- `LIMIT_IF_TOUCHED`

  Limit if Touched orders.

- `TRAILING_STOP`

  Trailing Stop orders.

- `TRAILING_STOP_LIMIT`

  Trailing Stop Limit orders.

- `TWAP_LIMIT`

  Time Weighted Average Price Limit orders (HK and US securities only).

- `TWAP`

  Time Weighted Average Price Market orders (US securities only).

- `VWAP_LIMIT`

  Volume Weighted Average Price Limit orders (HK and US securities only).
- `VWAP`

  Volume Weighted Average Price Market orders (US securities only).

Position Direction

> **PositionSide**

- `NONE`

  Unknown position
- `LONG`

  Long position, by default
- `SHORT`

  Short position

Account Type

> **TrdAccType**

- `NONE`

  Unknown type
- `CASH`

  Cash account
- `MARGIN`

  Margin account
- `TFSA`

Canadian TFSA account

- `RRSP`

  Canadian RRSP account

- `SRRSP`

  Canadian SRRSP account

- `DERIVATIVE`

  Japanese derivative account

Trading Environment

> **TrdEnv**

- `SIMULATE`

  Simulated environment

- `REAL`

  Real environment

Transaction Market

> **TrdMarket**

- `NONE`

  Unknown market

- `HK`

  Hong Kong market

- `US`

  US market

- `CN`

  A-share market 
- `HKCC`

  HKCC market 
- `FUTURES`

  Futures market
- `FUTURES_SIMULATE_US`

  US futures simulated market 
- `FUTURES_SIMULATE_HK`

  Hong Kong futures simulated market 
- `FUTURES_SIMULATE_SG`

  Singapore futures simulated market 
- `FUTURES_SIMULATE_JP`

  Japan futures simulated market 
- `HKFUND`

  HK fund market 
- `USFUND`

  US fund market 
- `SG`

  SG market 
- `JP`

  JP market 
- `AU`

AU market

- `MY`

  MY market

- `CA`

  CA market

Account Status

> **TrdAccStatus**

- `ACTIVE`

  Active account

- `DISABLED`

  Disabled account

Account Structure

> **TrdAccRole**

- `NONE`

  Unknown

- `MASTER`

  Master account

- `NORMAL`

  Normal account

- `IPO`

  Malaysian IPO account

Transaction Securities Market

Transaction Direction

> TrdSide

- `NONE`

  Unknown direction

- `BUY`

  Buy

- `SELL`

  Sell

- `SELL_SHORT`

  Sell short 

- `BUY_BACK`

  Buy back 

> **Tips**
>
> It is recommanded that only use `Buy` or `Sell` as the input parameter of direction of **place_order** interface.
> `BuyBack` and `SellShort` is only used as the display field for **Get Order List** , **Get History Order List**, **Orders Push Callback**, **Get Today's Deals**, **Get Historical Deals** and **Deals Push Callback** interface.

Order Validity Period

> TimeInForce

- `DAY`

  Good for the day

- `GTC`

  Good until cancel

Securities Firm to Which the Account Belongs

> SecurityFirm

- `NONE`

  Unknown

- `FUTUSECURITIES`

  FUTU HK

- `FUTUINC`

  Moomoo US

- `FUTUSG`

  Moomoo SG

- `FUTUAU`

  Moomoo AU

- `FUTUCA`

  Moomoo CA

- `FUTUMY`

  Moomoo MY

- `FUTUJP`

Moomoo JP

Simulate Account Type

> **SimAccType**

- `NONE`

  Unknown

- `STOCK`

  Stock simulation account

- `OPTION`

  Option simulation account

- `FUTURES`

  Futures simulation account

Account Risk Control Status

> **CltRiskStatus**

- `NONE`

  Unknown

- `LEVEL1`

  Very Safe

- `LEVEL2`

  Safe

- `LEVEL3`

  Safe

- `LEVEL4`

  Low Risk
- `LEVEL5`

  Medium Risk
- `LEVEL6`

  High Risk
- `LEVEL7`

  Warning
- `LEVEL8`

  Margin Call
- `LEVEL9`

  Margin Call

Day-trading Status

> **DtStatus**

- `NONE`

  Unknown
- `Unlimited`

  Unlimited 
- `EM_Call`

  EM-Call 
- `DT_Call`

  DT-Call 

Cash Flow Direction

> **CashFlowDirection**

- `NONE`

  Unknown

- `IN`

  Cash Inflow

- `OUT`

  Cash Outflow

JP Sub Account Type

> **SubAccType**

- `NONE`

  Unknown

- `JP_GENERAL`

  General - long

- `JP_TOKUTEI`

  Specified - long

- `JP_NISA_GENERAL`

  General NISA

- `JP_NISA_TSUMITATE`

  Tsumitate NISA

- `JP_GENERAL_SHORT`

  General - short

- `JP_TOKUTEI_SHORT`

  Specified - short

- `JP_HONPO_GENERAL`

  Domestic Margin Trading Collateral - General

- `JP_GAIKOKU_GENERAL`

  Foreign Margin Trading Collateral - General

- `JP_HONPO_TOKUTEI`

  Domestic Margin Trading Collateral - Specified

- `JP_GAIKOKU_TOKUTEI`

  Foreign Margin Trading Collateral - Specified

- `JP_DERIVATIVE_LONG`

  Derivatives Sub-account - Long

- `JP_DERIVATIVE_SHORT`

  Derivatives Sub-account - Short

- `JP_HONPO_DERIVATIVE_GENERAL`

  Domestic Derivatives Margin Sub-account - General

- `JP_GAIKOKU_DERIVATIVE_GENERAL`

  Foreign Derivatives Margin Sub-account - General

- `JP_HONPO_DERIVATIVE_TOKUTEI`

  Domestic Derivatives Margin Sub-account - Specific

- `JP_GAIKOKU_DERIVATIVE_TOKUTEI`

Foreign Derivatives Margin Sub-account - Specific

Asset Category

> AssetCategory

- `NONE`

  Unknown

- `JP`

  Domestic

- `US`

  Foreign

Transaction Category

TrdCategory

```
enum TrdCategory
{
    TrdCategory_Unknown = 0; //Unknown
    TrdCategory_Security = 1; //Securities
    TrdCategory_Future = 2; //Futures
}
```

Account Cash Information

AccCashInfo

```
message AccCashInfo
{
    optional int32 currency = 1; //Currency type, refer to Currency
    optional double cash = 2; //Cash balance
    optional double availableBalance = 3; //Available cash withdrawal amount
```

```
    optional double netCashPower = 4;      // Net cash power
}
```

Account Assets Information by Market

AccMarketInfo

```
message AccCashInfo
{
    optional int32 trdMarket = 1;         // Trading market, refer to TrdMarket
    optional double assets = 2;           // Account assets information by market
}
```

Transaction Protocol Public Header

TrdHeader

```
message TrdHeader
{
  required int32 trdEnv = 1; //Trading environment, refer to the enumeration defi
  required uint64 accID = 2; //Trading account, trading account should match to t
  required int32 trdMarket = 3; //Trading market, refer to the enumeration defini
  optional int32 jpAccType = 4; //JP sub account type, refer to TrdSubAccType
}
```

Trading Account

TrdAcc

```
message TrdAcc
{
  required int32 trdEnv = 1; //Trading environment, refer to the enumeration defi
  required uint64 accID = 2; //Trading account
  repeated int32 trdMarketAuthList = 3; //The trading market permissions supporte
```

```
    optional int32 accType = 4; //Account type, refer to TrdAccType
    optional string cardNum = 5; //card number
    optional int32 securityFirm = 6; //security firm, refer to SecurityFirm
    optional int32 simAccType = 7; //simulate account type, see SimAccType
    optional string uniCardNum = 8; //Universal account number
    optional int32 accStatus = 9; //Account status, refer to TrdAccStatus
    optional int32 accRole = 10; //Account Structure, used to distinguish between r
    repeated int32 jpAccType = 11; //JP sub account type, refer to TrdSubAccType
}
```

Account Funds

Funds

```
message Funds
{
    required double power = 1; //Maximum Buying Power (Minimum OpenD version requi
    required double totalAssets = 2; //Net Assets
    required double cash = 3; //Cash (Only Single market accounts use this field.
    required double marketVal = 4; //Securities Market Value (only applicable to s
    required double frozenCash = 5; //Funds on Hold
    required double debtCash = 6; //Interest Charged Amount (Minimum OpenD version
    required double avlWithdrawalCash = 7; //Withdrawable Cash (Only Single market

    optional int32 currency = 8;            //The currency used for this query (onl
    optional double availableFunds = 9;     //Available funds (only applicable to
    optional double unrealizedPL = 10;      //Unrealized gain or loss (only applica
    optional double realizedPL = 11;        //Realized gain or loss (only applicabl
    optional int32 riskLevel = 12;           //Risk control level (only applicable
    optional double initialMargin = 13;      //Initial Margin (only applicable to
    optional double maintenanceMargin = 14;  //Maintenance Margin (Minimum OpenD ve
    repeated AccCashInfo cashInfoList = 15;  //Cash information by currency (only a
    optional double maxPowerShort = 16; //Short Buying Power (Minimum OpenD versio
    optional double netCashPower = 17;  //Cash Buying Power  (Only Single market ac
    optional double longMv = 18;        //Long Market Value (Minimum OpenD version
    optional double shortMv = 19;       //Short Market Value (Minimum OpenD versio
    optional double pendingAsset = 20;  //Asset in Transit (Minimum OpenD version
    optional double maxWithdrawal = 21;          //Maximum Withdrawal (only applica
    optional int32 riskStatus = 22;              //Risk status (only applicable to
    optional double marginCallMargin = 23;       //Margin-call Margin (Minimum Oper

```

```
    optional bool isPdt = 24;                //Is it marked as a PDT. True: It is a
    optional string pdtSeq = 25;             //Day Trades Left. Only applicable to
    optional double beginningDTBP = 26;      //Beginning DTBP. Only applicable to s
    optional double remainingDTBP = 27;      //Remaining DTBP. Only applicable to s
    optional double dtCallAmount = 28;       //Day-trading Call Amount. Only applic
    optional int32 dtStatus = 29;                 //Day-trading Status. Only applica

    optional double securitiesAssets = 30; // Net asset value of securities
    optional double fundAssets = 31; // Net asset value of fund
    optional double bondAssets = 32; // Net asset value of bond

repeated AccMarketInfo marketInfoList = 33; //Account assets information by marke
}
```

Account Holding

Position

```
message Position
{
    required uint64 positionID = 1; //Position ID, a unique identifier of a posit
    required int32 positionSide = 2; //Position direction, refer to the enumerati
    required string code = 3; //Code
    required string name = 4; //Name
    required double qty = 5; //Holding quantity, 2 decimal places, the same belo
    required double canSellQty = 6; //Available quantity. Available quantity = Ho
    required double price = 7; //Market price, 3 decimal places, 2 decimal places
    optional double costPrice = 8; //Diluted Cost (for securities account). Avera
    required double val = 9; //Market value, 3 decimal places, value of this fie
    required double plVal = 10; //Amount of profit or loss, 3 decimal places,  2
    optional double plRatio = 11; //Percentage of profit or loss(under diluted co
    optional int32 secMarket = 12; //The market to which the securities belong, 

    //The following is the statistics of this position today
    optional double td_plVal = 21; //Today's profit or loss, 3 decimal places, th
    optional double td_trdVal = 22; //Today's trading volume, not applicable for
    optional double td_buyVal = 23; //Total value bought today, not applicable fo
    optional double td_buyQty = 24; //Total amount bought today, not applicable 
    optional double td_sellVal = 25; //Total value sold today, not applicable for
    optional double td_sellQty = 26; //Total amount sold today, not applicable fo

```

```
    optional double unrealizedPL = 28; //Unrealized profit or loss (only applica
    optional double realizedPL = 29; //Realized profit or loss (only applicable
    optional int32 currency = 30;        // Currency type, refer to Currency
    optional int32 trdMarket = 31;  //Trading market, refer to the enumeration d

    optional double dilutedCostPrice = 32;  //diluted cost price, applicable for
    optional double averageCostPrice = 33;  //average cost price, not applicable
    optional double averagePlRatio = 34;  //Percentage of profit or loss(under a
}
```

Order

Order

```
message Order
{
    required int32 trdSide = 1; //Trading direction, refer to TrdSide enumeratio
    required int32 orderType = 2; //Order type, refer to enumeration definition
    required int32 orderStatus = 3; //Order status, refer to enumeration definit
    required uint64 orderID = 4; //Order number
    required string orderIDEx = 5; //Extended order number (only for checking th
    required string code = 6; //code
    required string name = 7; //Name
    required double qty = 8; //Order quantity,  3 decimal places, option unit is
    optional double price = 9; //Order price, 3 decimal places
    required string createTime = 10; //Create time, strictly in accordance with
    required string updateTime = 11; //The last update time, strictly according
    optional double fillQty = 12; //Filled quantity, 2 decimal place accuracy, t
    optional double fillAvgPrice = 13; //Average price of the fill, no precision
    optional string lastErrMsg = 14; //The last error description, if there is a
    optional int32 secMarket = 15; //The market to which the securities belong,
    optional double createTimestamp = 16; //Timestamp for creation
    optional double updateTimestamp = 17; //Timestamp for last update
    optional string remark = 18; //User remark string, the maximum length is 64
    optional double auxPrice = 21; //Trigger price
    optional int32 trailType = 22; //Trailing type, see Trd_Common.TrailType enu
    optional double trailValue = 23; //Trailing amount / ratio
    optional double trailSpread = 24; //Specify spread
    optional int32 currency = 25;        // Currency type, refer to Currency
    optional int32 trdMarket = 26;  //Trading market, refer to the enumeration d
    optional int32 session = 27; //Trading session, refer to the enumeration def
```

```
    optional int32 jpAccType = 28; //JP sub account type, refer to TrdSubAccType
}
```

Order Fee Item

OrderFeeItem

```
message OrderFeeItem
{
    optional string title = 1; //Fee title
    optional double value = 2; //Fee Value
}
```

Order Fee

OrderFee

```
message OrderFee
{
    required string orderIDEx = 1; //Server order id
    optional double feeAmount = 2; //Fee amount
    repeated OrderFeeItem feeList = 3; //Fee details
}
```

Order Fill

OrderFill

```
message OrderFill
{
    required int32 trdSide = 1; //Trading direction, refer to enumeration defini
    required uint64 fillID = 2; //OrderFill ID
    required string fillIDEx = 3; //Extended OrderFill ID (only for checking the
    optional uint64 orderID = 4; //Order ID
```

```
    optional string orderIDEx = 5; //Extended order ID (only when checking the pr
    required string code = 6; //code
    required string name = 7; //Name
    required double qty = 8; //Filled quantity, 2 decimal place accuracy, the opt
    required double price = 9; //Price of the fill, 3 decimal places
    required string createTime = 10; //Create time (transaction time), in strict
    optional int32 counterBrokerID = 11; //Counter Broker ID, valid for Hong Kong
    optional string counterBrokerName = 12; //Counter Broker Name, valid for Hong
    optional int32 secMarket = 13; //Securities belong to the market, refer to en
    optional double createTimestamp = 14; //Create a timestamp
    optional double updateTimestamp = 15; //last update timestamp
    optional int32 status = 16; //Deal status, refer to enumeration definition of
    optional int32 trdMarket = 17;  //Trading market, refer to enumeration defini
    optional int32 jpAccType = 18; //JP sub account type, refer to TrdSubAccType
}
```

Maximum Trading Quantity

MaxTrdQtys

```
message MaxTrdQtys
{
    //Due to the current server's implementation, it is required to sell the hold
    required double maxCashBuy = 1; //Buy on cash. (Maximum quantity that can be
    optional double maxCashAndMarginBuy = 2; //Buy on margin. (Maximum quantity t
    required double maxPositionSell = 3; //Sell on position. (Maximum quantity ca
    optional double maxSellShort = 4; //Short sell. (Maximum quantity can be shor
    optional double maxBuyBack = 5; //Short positions. (Buyback required quantity
    optional double longRequiredIM = 6;         //Initial margin change when buyi
    optional double shortRequiredIM = 7;        //Initial margin change when sell
}
```

Cash Flow Summary Info

FlowSummaryInfo

```
message FlowSummaryInfo
{
```

```
    optional string clearingDate = 1; //clearing date
    optional string settlementDate = 2; //settlement date
    optional int32 currency = 3; //currency
    optional string cashFlowType = 4; //cash flow type
    optional int32 cashFlowDirection = 5; //cash flow direction, refer to TrdCas
    optional double cashFlowAmount = 6; //amount
    optional string cashFlowRemark = 7; //remark
    optional uint64 cashFlowID = 8; //cash flow ID
}
```

Filter Conditions

TrdFilterConditions

```
message TrdFilterConditions
{
  repeated string codeList = 1; //Code filtering, only returns the products for
  repeated uint64 idList = 2; //ID primary key filter, only returns the products
  optional string beginTime = 3; //Start time, strictly in accordance with YYYY-M
  optional string endTime = 4; //The end time, strictly in accordance with YYYY-M
  repeated string orderIDExList = 5; // The server order id list, which can be us
  optional int32 filterMarket = 6; //Trading market filter, refer to enumeration
}
```

Basic Functions

Set Interface Information(deprecated)

`set_client_info(client_id, client_ver)`

- **Introduction**

  Set calling interface information (unnecessary).

- **Parameters**

  - client_id: the identification of the client
  - client_ver: the version number of the client

```
from moomoo import *
SysConfig.set_client_info("MymoomooAPI", 0)
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close()
```

Set Protocol Format

`set_proto_fmt(proto_fmt)`

- **Introduction**

  Set the communication protocol body format, Protobuf and Json formats are currently supported , default ProtoBuf, unnecessary interface

- **Parameters**

  - proto_fmt: protocol format, refer to ProtoFMT

- **Example**

```
from moomoo import *
SysConfig.set_proto_fmt(ProtoFMT.Protobuf)
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close()
```

Set Protocol Encryption Globally

`Enable_proto_encrypt(is_encrypt)`

- **Introduction** Setting protocol encryption can help users protect their requests and returned contents globally. For more information about Protocol Encryption Process, please check here.
- **Parameters**

Parameter	Type	Description
is_encrypt	bool	Enable encryption or not.

- **Example**

```
from moomoo import *
SysConfig.enable_proto_encrypt(True)
SysConfig.set_init_rsa_file("conn_key.txt")   # rsa private key file path
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close()
```

Set the Path of Private Key

`set_init_rsa_file(file)`

- **Introduction**

  Set the Path of Private Key in moomoo API. For more information about Protocol Encryption Process, please check here.

- **Parameters**

Parameter	Type	Description
file	str	Private key file path.

- **Example**

```
from moomoo import *
SysConfig.enable_proto_encrypt(True)
SysConfig.set_init_rsa_file("conn_key.txt")   # rsa private key file path
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close()
```

Set Thread Mode

`set_all_thread_daemon(all_daemon)`

- **Introduction** Whether to set all internally threads to be daemon threads.

- If it is set to be daemon threads, then after the main thread exits, the process also exits.
  For example, when using the real-time callback API, you need to make sure the main thread survives by yourself. Otherwise, when the main thread exits, the process also exits and you will no longer receive the push data.
- If it is set to a non-daemon thread, the process will not exit after the main thread exits. For example, if you do not call close() to close the connection after creating a quote or trade object, the process will not exit even if the main thread exits.

- **Parameters**

Parameter	Type	Description
all_daemon	bool	Whether to set threads to be daemon threads.

- **Example**

```
from moomoo import *
SysConfig.set_all_thread_daemon(True)
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
# the process will exit without calling quote_ctx.close(),
```

Set Callback

`set_handler(handler)`

- **Introduction**

  Set asynchronous callback processing object

- **Parameters**

  - handler: callback processing object

Class	Description
SysNotifyHandlerBase	OpenD notification processing base class
StockQuoteHandlerBase	Quote processing base class
OrderBookHandlerBase	Order book processing base class
CurKlineHandlerBase	Real-time candlestick processing base class
TickerHandlerBase	Tick-By-Tick processing base class
RTDataHandlerBase	Time Frame data processing base class
BrokerHandlerBase	Broker queue processing base class

Class	Description
PriceReminderHandlerBase	Price reminder processing base class
TradeOrderHandlerBase	Order processing base class
TradeDealHandlerBase	Order fill processing base class

- Example

```
import time
from moomoo import *
class OrderBookTest(OrderBookHandlerBase):
    def on_recv_rsp(self, rsp_pb):
        ret_code, data = super(OrderBookTest,self).on_recv_rsp(rsp_pb)
        if ret_code != RET_OK:
            print("OrderBookTest: error, msg: %s" % data)
            return RET_ERROR, data
        print("OrderBookTest ", data) # OrderBookTest's own processing logic
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = OrderBookTest()
quote_ctx.set_handler(handler) # Setting real-time order book callback
quote_ctx.subscribe(['HK.00700'], [SubType.ORDER_BOOK]) # Subscribe to the order book type, OpenD st
time.sleep(15) # Set the script to receive OpenD push duration to 15 seconds
quote_ctx.close() # Close the current connection, OpenD will automatically cancel the subscription o
```

Get Connection ID

`get_sync_conn_id()`

- Introduction

  Get the connection ID, the value will be available after the connection is successfully initialized

- Return

  - conn_id: connection ID

- Example

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.get_sync_conn_id()
quote_ctx.close() # After using the connection, remember to close it to prevent the number of connec
```

Event Notification Callback

`SysNotifyHandlerBase`

- **Introduction**

  Notify OpenD of some important news, such as disconnection, etc.

- **Protocol ID**

  1003

- **Return**

Field	Type	Description
ret	int	Returned value. On success, ret == RET_OK. On error, ret != RET_OK.
data	tuple	If ret == RET_OK, **event notification data** is returned.
	str	If ret != RET_OK, error description is returned.

  - The format of **event notification** data is as follows:

Field	Type	Description	
notify_type	SysNotifyType	Notification data type	
sub_type	ProgramStatusType	Subtype. If notify_type == SysNotifyType.PROGRAM_STATUS, program status type is returned.	
	GtwEventType	1581	Subtype. If notify_type == SysNotifyType.GTW_EVENT, OpenD event type is returned.
	0	If notify_type !=SysNotifyType.PROGRAM_STATUS and notify_type !=SysNotifyType.GTW_EVENT, no useful information is returned.	
msg	dict	Event information. If notify_type == SysNotifyType.CONN_STATUS, **connection status event information** is returned.	
		Event information. If notify_type == SysNotifyType.QOT_RIGHT, **quote**	

		right event information is returned.

- The format of **connection status event information** is as follows(The value of connection status is a bool type, with True for normal, and False for disconnected):

```
{
    'qot_logined': bool1,
    'trd_logined': bool2,
}
```

- The format of **quote right event information** is as follows(the type of quote right refers to Quote Right):

```
{
    'hk_qot_right': value1,
    'hk_option_qot_right': value2,
    'hk_future_qot_right': value3,
    'us_qot_right': value4,
    'us_option_qot_right': value5,
    'us_future_qot_right': value6,  // deprecated
    'cn_qot_right': value7,
     'us_index_qot_right': value8,
     'us_otc_qot_right': value9,
     'sg_future_qot_right': value10,
     'jp_future_qot_right': value11,
     'us_future_qot_right_cme': value12,
     'us_future_qot_right_cbot': value13,
     'us_future_qot_right_nymex': value14,
     'us_future_qot_right_comex': value15,
     'us_future_qot_right_cboe': value16,
}
```

- Example

```
import time
from moomoo import *

class SysNotifyTest(SysNotifyHandlerBase):
    def on_recv_rsp(self, rsp_str):
        ret_code, data = super(SysNotifyTest, self).on_recv_rsp(rsp_str)
        notify_type, sub_type, msg = data
        if ret_code != RET_OK:
            logger.debug("SysNotifyTest: error, msg: {}".format(msg))
            return RET_ERROR, data
        if (notify_type == SysNotifyType.GTW_EVENT):  #  OpenD event notification
            print("GTW_EVENT, type: {} msg: {}".format(sub_type, msg))
        elif (notify_type == SysNotifyType.PROGRAM_STATUS):  # Notification of change in program sta
```

```
            print("PROGRAM_STATUS, type: {} msg: {}".format(sub_type, msg))
        elif (notify_type == SysNotifyType.CONN_STATUS):  ## Notification of change in connection st
            print("CONN_STATUS, qot: {}".format(msg['qot_logined']))
            print("CONN_STATUS, trd: {}".format(msg['trd_logined']))
        elif (notify_type == SysNotifyType.QOT_RIGHT):  # Notification of change in quote right
            print("QOT_RIGHT, hk: {}".format(msg['hk_qot_right']))
            print("QOT_RIGHT, hk_option: {}".format(msg['hk_option_qot_right']))
            print("QOT_RIGHT, hk_future: {}".format(msg['hk_future_qot_right']))
            print("QOT_RIGHT, us: {}".format(msg['us_qot_right']))
            print("QOT_RIGHT, us_option: {}".format(msg['us_option_qot_right']))
            print("QOT_RIGHT, us_future: {}".format(msg['us_future_qot_right']))
            print("QOT_RIGHT, cn: {}".format(msg['cn_qot_right']))
            print("QOT_RIGHT, us_index: {}".format(msg['us_index_qot_right']))
              print("QOT_RIGHT, us_otc: {}".format(msg['us_otc_qot_right']))
              print("QOT_RIGHT, sg_future: {}".format(msg['sg_future_qot_right']))
              print("QOT_RIGHT, jp_future: {}".format(msg['jp_future_qot_right']))
            print("QOT_RIGHT, us_future_cme: {}".format(msg['us_future_qot_right_cme']))
            print("QOT_RIGHT, us_future_cbot: {}".format(msg['us_future_qot_right_cbot']))
            print("QOT_RIGHT, us_future_nymex: {}".format(msg['us_future_qot_right_nymex']))
            print("QOT_RIGHT, us_future_comex: {}".format(msg['us_future_qot_right_comex']))
            print("QOT_RIGHT, us_future_cboe: {}".format(msg['us_future_qot_right_cboe']))
        return RET_OK, data

quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = SysNotifyTest()
quote_ctx.set_handler(handler)   # Set real-time swing callback
time.sleep(15)  # Set the script to receive OpenD push duration to 15 seconds
quote_ctx.close()  # After using the connection, remember to close it to prevent the number of conne
```

General Definitions

Interface Result

> RET_CODE

- `RET_OK`

  Success

- `RET_ERROR`

  Failed

Protocol Format

> ProtoFMT

- `Protobuf`

  Google Protobuf

- `Json`

  Json

Packet Encryption Algorithm

Program Status Type

> ProgramStatusType

- `NONE`

  Unknown

- `LOADED`

  The necessary modules have been loaded

- `LOGING`

  Logging in

- `NEED_PIC_VERIFY_CODE`

  Need graphic verification code

- `NEED_PHONE_VERIFY_CODE`

  Need mobile phone verification code

- `LOGIN_FAILED`

  Login failed

- `FORCE_UPDATE`

  The client version is too low

- `NESSARY_DATA_PREPARING`

  Pulling necessary information

- `NESSARY_DATA_MISSING`

  Missing necessary information

- `UN_AGREE_DISCLAIMER`

  Disclaimer is not agreed

- `READY`

  Ready to use

- `FORCE_LOGOUT`

  OpenD was forced to log out

OpenD Event Notification Type

> GtwEventType

- `LocalCfgLoadFailed`

  Failed to load the local configuration file

- `APISvrRunFailed`

  Failed to run the OpenD monitoring service

- `ForceUpdate`

  Force upgrade of the OpenD

- `LoginFailed`

  Failed to log in to moomoo servers

- `UnAgreeDisclaimer`

  Did not agree to the disclaimer, unable to run

- `LOGIN_FAILED`

  Login failed

- `NetCfgMissing`

  Missing network connection configuration

- `KickedOut`

  Login kicked offline

- `LoginPwdChanged`

  Login password has been changed

- `BanLogin`

  This account is not allowed to log in by moomoo servers

- `NeedPicVerifyCode`

  Need graphic verification code

- `NeedPhoneVerifyCode`

  Need mobile verification code

- `AppDataNotExist`

  Program package data loss

- `NessaryDataMissing`

  The necessary data is not synchronized successfully

- `TradePwdChanged`

  Transaction password change notice

- `EnableDeviceLock`

  Need to enable device lock

System Notification Type

> SysNotifyType

- `GTW_EVENT`

  Gateway event

- `PROGRAM_STATUS`

  Program status changes

- `CONN_STATUS`

  Status of Connection to moomoo servers has been changed

- `QOT_RIGHT`

  Quotes authority changed

Package Unique Identifier

PacketID

```
message PacketID
{
  required uint64 connID = 1; //The current TCP connection ID, the unique identi
  required uint32 serialNo = 2; //Increment serial number
}
```

Program Status

ProgramStatus

```
message ProgramStatus
{
  required ProgramStatusType type = 1; //Current status
  optional string strExtDesc = 2; //Additional description
}
```

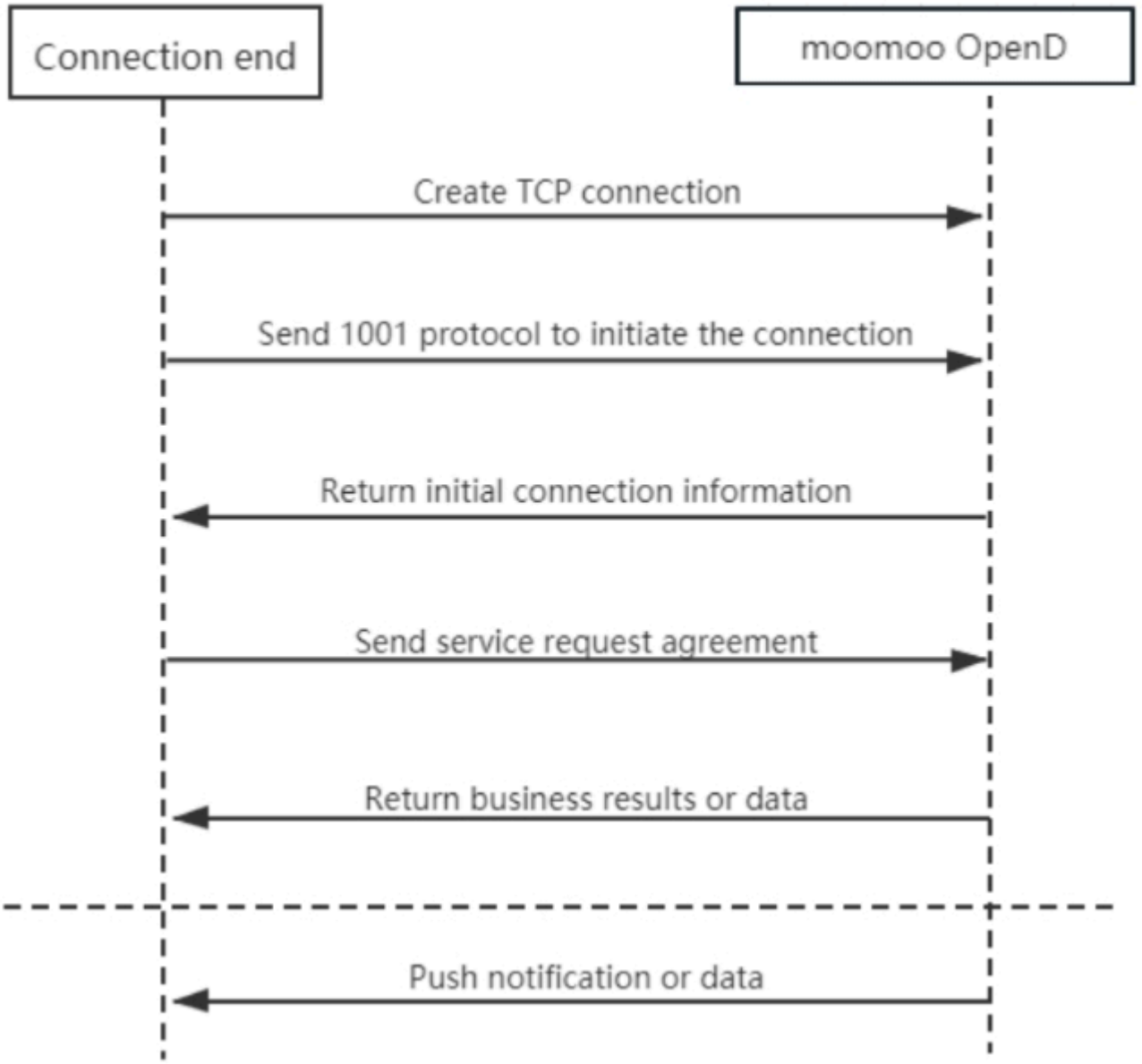
Protocol Introduction

moomoo API is an API SDK, encapsulated by moomoo including mainstream programming languages (Python, Java, C #, C++, JavaScript) to make it easy for you to call and reduce the difficulty of trading strategy development.
This part mainly introduces the underlying protocol of communication between script and OpenD service, which is suitable for users who do not use the above five programming languages.

> **Tips**
>
> - If you are using a programming language that is one of the five mainstream programming languages mentioned above, you can skip this part.

Protocol Request Process

- Create a connection
- Initialize the connection
- Request data or receive pushed data
- Send KeepAlive protocol periodically to keep connected



Protocol Design

The protocol data includes the protocol header and the protocol body. The protocol header is fixed, and the protocol body is determined according to the specific protocol.

Protocol Header

```
struct APIProtoHeader
{
    u8_t szHeaderFlag[2];
    u32_t nProtoID;
    u8_t nProtoFmtType;
    u8_t nProtoVer;
```

```
    u32_t nSerialNo;
    u32_t nBodyLen;
    u8_t arrBodySHA1[20];
    u8_t arrReserved[8];
};
```

Field	Description
szHeaderFlag	Packet header start flag, fixed as "FT"
nProtoID	Protocol ID
nProtoFmtType	Protocol type, 0 for Protobuf, 1 for Json
nProtoVer	Protocol version, used for iterative compatibility, currently 0
nSerialNo	Packet serial number, used to correspond to the request packet and return packet, and it is required to be incremented
nBodyLen	Body length
arrBodySHA1	SHA1 hash value of the original data of the packet body (after decryption)
arrReserved	Reserved 8-byte extension

> **Tips**
>
> - *u8_t* refer to 8-bit unsigned integer, *u32_t* refer to 32-bit unsigned integer
> - *OpenD* internal processing uses *Protobuf*, so the protocol format recommends using *Protobuf*, to reduce *Json* conversion overhead.
> - The *nProtoFmtType* field specifies the data type of the package body, and the corresponding protocol type will be returned when the package is returned. The data type of the push protocol is specified by the *OpenD* configuration file
> - *arrBodySHA1* is used to verify the consistency of the requested data before and after network transmission, and must be filled in correctly
> - The binary stream of the protocol header uses little-endian byte order, that is, generally there is no need to use *ntohl* and other related functions to convert the data

Protocol Body

Packet Body Structure of Protobuf Request

```
message C2S
{
    required int64 req = 1;
}

message Request
{
    required C2S c2s = 1;
}
```

Packet Body Structure of Protobuf Response

```
message S2C
{
    required int64 data = 1;
}

message Response
{
    required int32 retType = 1 [default = -400]; //RetType, result of return
    optional string retMsg = 2;
    optional int32 errCode = 3;
    optional S2C s2c = 4;
}
```

Field	Description
c2s	Request parameter structure
req	Request parameters, actually defined according to the protocol
retType	Request result

Field	Description
retMsg	The reason for the failed request
errCode	The corresponding error code for failed request
s2c	Response data structure, some protocols do not return data if there is no such field
data	Response data, actually defined according to the protocol

> **Tips**
>
> - The package body format type request package is specified by *nProtoFmtType* field from protocol header, and the *OpenD* initiative push format is set in InitConnect.
> - The original protocol file format is defined in *Protobuf* format. If you need *json* format transmission, it is recommended to use the *protobuf3* interface to directly convert to *json*.
> - The enumeration value field definition uses signed integer, and the comment indicates the corresponding enumeration. The enumeration is generally defined in *Common.proto, Qot_Common.proto, Trd_Common.proto* files.
> - The price, percentage and other data in the protocol are transmitted in floating point type. Direct use will cause accuracy problems. It needs to be rounded according to the accuracy (if not specified in the protocol, the default is 3 decimal places) before use.

Heartbeat Keep Alive

```
syntax = "proto2";
package KeepAlive;
option java_package = "com.moomoo.openapi.pb";
option go_package = "github.com/moomooopen/mmapi4go/pb/keepalive";

import "Common.proto";

message C2S
{
     required int64 time = 1; //Greenwich timestamp when the client sends the pac
}
```

```
12
13  message S2C
14  {
15      required int64 time = 1; //Greenwich timestamp when the server returned the
16  }
17
18  message Request
19  {
20      required C2S c2s = 1;
21  }
22
23  message Response
24  {
25      required int32 retType = 1 [default = -400]; //RetType, return result
26      optional string retMsg = 2;
27      optional int32 errCode = 3;
28
29      optional S2C s2c = 4;
30  }
```

- **Introduction**

  Heartbeat keep alive

- **Protocol ID**

  1004

- **Introduction**

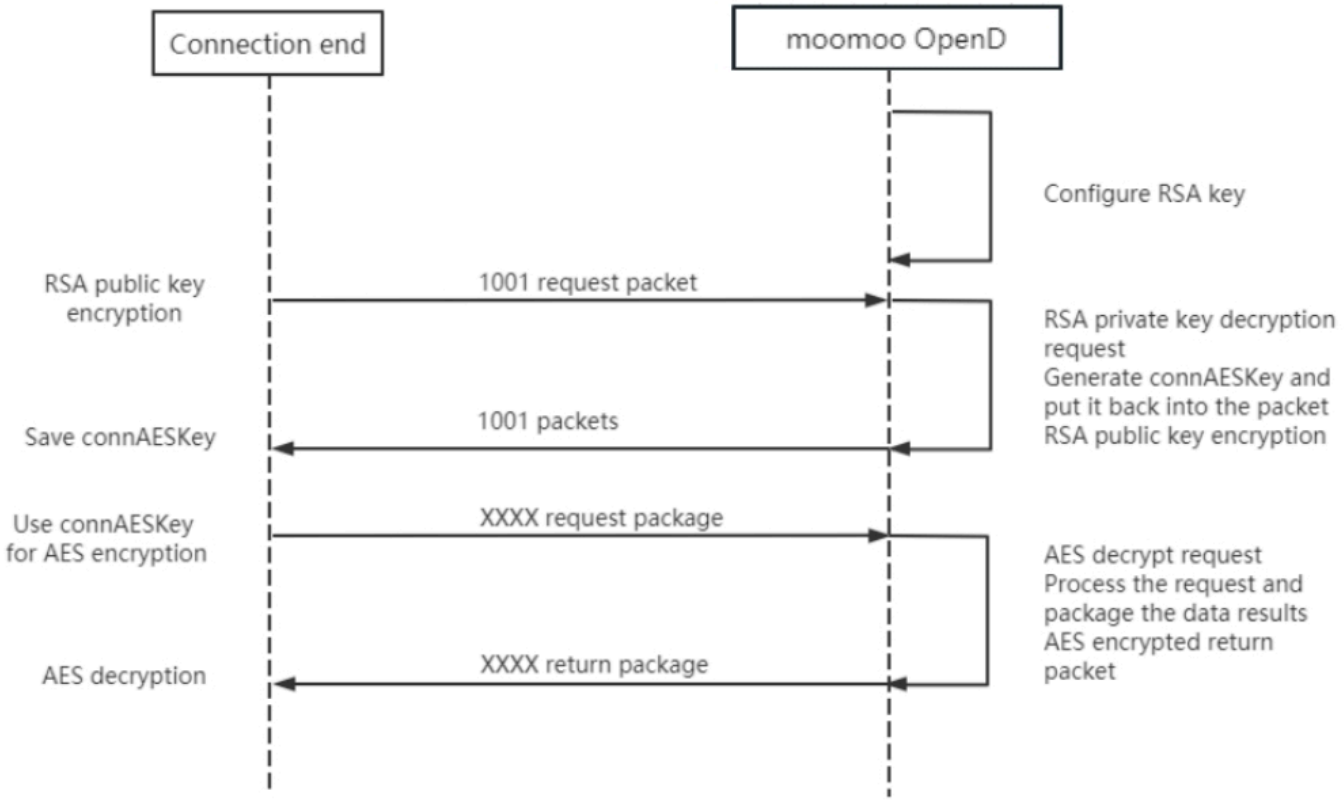
  According to the heartbeat keeping alive interval returned by the **initialization protocol**, send the heartbeeat keep alive protocol to OpenD.

Encrypted Communication Process

- If OpenD is configured with encryption, **InitConnect** must use **RSA** public key encryption to initialize the connection protocol, and other subsequent protocols use the random key returned by InitConnect for AES encrypted communication.
- The encryption process of OpenD draws on the SSL protocol. Considering that services and applications are generally deployed locally, we simplifies the related processes. OpenD

shares the same RSA private key file with the access Client. Please save and distribute the private key file properly.

- Go to this URL to generate a random RSA key pair online. The key format must be PCKS#1, the key length can be 512, 1024, and do not set password. Copy and save the generated private key to a file, and then configure the path of the private key file to the **rsa_private_key** configuration item agreed upon in OpenD Configuration.
- **It is recommended that users who have real trade configure encryption to avoid leakage of account and trade information.**



RSA Encryption and Decryption

- OpenD configuration Convention **rsa_private_key** is the path of the private key file
- OpenD shares the same private key file with the access client
- RSA encryption and decryption is only used for InitConnect requests, and is used to securely obtain symmetric encryption key of other request protocols
- The RSA key of OpenD is 1024-bit, the filling method is PKCS1, public key encryption, private key decryption, public key can be generated by private key

Send Data Encryption

- RSA encryption rules: If the number of key bits is key_size, the maximum length of a single encryption string is (key_size)/8-11. The current number of bits is 1024, and the length of one encryption can be set to 100.
- Divide the plaintext data into one or several segments of up to 100 bytes for encryption, and the final encrypted data is spliced by all segmented encrypted data.

Receive Data Decryption

- RSA decryption also follows the segmentation rule. For a 1024-bit key, the length of each segment to be decrypted is 128-byte.
- Divide the ciphertext data into one or several segments of up to 128 bytes for decryption, and the final decrypted data is spliced by all segmented decrypted data.

AES Encryption and Decryption

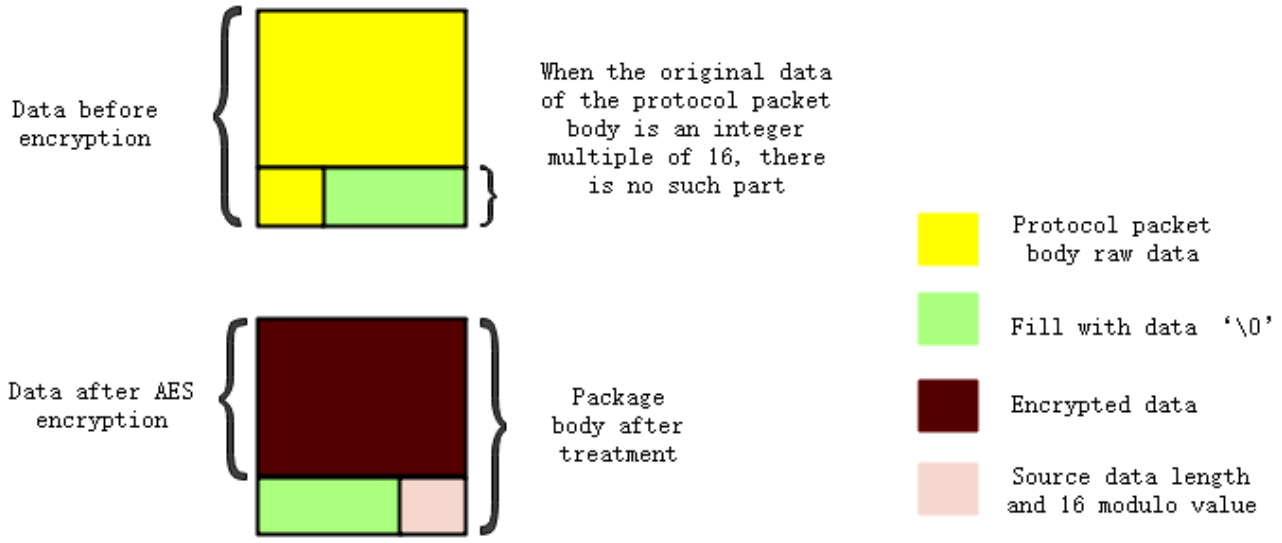
- The encryption key is returned by the InitConnect protocol
- The ecb encryption mode of AES is used by default.

Send Data Encryption

- AES encryption requires that the length of the source data must be an integer multiple of 16, so it needs to be aligned with ‘0’ before encryption. Record mod_len for source data length and 16 module.
- Because it is possible to modify the source data before encryption, it is necessary to add a 16-byte padding data block at the end of the encrypted data. The last byte is assigned mod_len, and the remaining bytes are assigned the value '0'. The encrypted data and additional populated data blocks are spliced as the body data to be sent in the end.

Receive Data Decryption

- For protocol body data, first take out the last byte and record it as mod_len, then truncate the body to the 16-byte padding data block before decrypting it (corresponding to the encrypted padding extra data block logic).
- When mod_len is 0, the above decrypted data is the body data returned by the protocol, otherwise the tail (16-mod_len) length of the data used for filling and alignment needs to be truncated.


Data before encryption
When the original data of the protocol packet body is an integer multiple of 16, there is no such part
Protocol packet body raw data
Fill with data '\0'
Data after AES encryption
Package body after treatment
Encrypted data
Source data length and 16 modulo value

OpenD Related

Q1: OpenD automatically exited due to failure to complete "Questionnaire Evaluation and Agreement Confirmation"

A: You need to carryout relevant questionnaire evaluation and agreement confirmation before you can use OpenD. Please go to complete.

Q2: OpenD exited due to "the program's own data does not exist"

A: Generally, the copy of the own data fails due to permission problems. You can try to copy the file extracted from *Appdata.dat* in the program directory to the program data directory.

- Windows program data directory: `%appdata%/com.moomoo.OpenD/F3CNN`
- Non-windows program data directory: `~/.com.moomoo.OpenD/F3CNN`

Q3: OpenD service failed to start

A: Please check:

1. Whether there are other programs occupying the configured port;
2. Is there a OpenD configured with the same port already running?

Q4: How to verify the mobile phone verification code?

A: On the OpenD interface or remotely to the Telnet port, enter the command 'input_phone_verify_code -code=123456`.

> **Tips**
>
> - 123456 is the mobile phone verification code received

- there is a space before '-code=123456'

Q5: Are other programming languages supported?

A: OpenD provides a socket-based protocol. Currently we provide and maintain Python, C++, Java, C# and JavaScript interfaces, **download entry**.

If the above languages still cannot meet your needs, you can connect to the Protobuf protocol by yourself.

Q6: Verify the device lock multiple times on the same device

A: The device ID is randomly generated and stored in the \com.moomoo.OpenD\F3CNN\Device.dat file.

> **Tips**
>
> 1. If the file is deleted or damaged, OpenD will regenerate a new device ID and then verify the device lock.
> 2. In addition, users of mirror copy deployment need to be aware that if the Device.dat content of multiple machines is the same, it will also cause these machines to verify the device lock multiple times. Delete the Device.dat file to solve it.

Q7: Does OpenD provide a Docker image?

A: Not currently available.

Q8: Can one account log in to multiple OpenD?

A: One account can log in to OpenD or other client terminals on multiple machines, and up to 10 OpenD terminals are allowed to log in at the same time. At the same time, there is a restriction of "market kicking", and only one OpenD can obtain the highest authority market.

For example, if two terminals log into the same account, there can only be one HK stock LV2 quotation and the other HK stock BMP quotation.

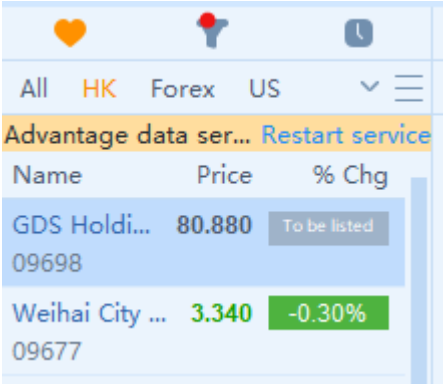
Q9: How to control the market permissions of OpenD and other clients (desktop and mobile)?

A: In accordance with the regulations of the exchange, there will be a restriction on “market kicking” when multiple terminals are online at the same time, and only one terminal can obtain the highest authority market. The auto_hold_quote_right parameter is built-in in the startup parameters of the command line version of OpenD, which is used to flexibly configure market permissions. When this parameter option is enabled, OpenD will automatically retrieve it after the market permission is robbed. If it is robbed again within 10 seconds, other terminals will obtain the highest market quotation authority (OpenD will not rob again).

Q10: How to give priority to the OpenD market authority?

A:

1. Configure the OpenD startup parameter auto_hold_quote_right to 1;
2. Make sure not to grab the highest authority twice in a row within 10 seconds on the mobile or desktop moomoo (login counts once, and click "Restart Quotes" to count the second time).



Q11: How to give priority to the market authority of the mobile terminal (or desktop terminal)?

A: Set OpenD startup parameter **auto_hold_quote_right** to 0, and login with mobile or PC moomoo after OpenD.

Q12: Use the Visualization OpenD to remember the password to log in. After a long time hang up, it prompts that the connection is disconnected. Do I need to log in again?

A: Using the Visualization OpenD, if you choose to remember the password to log in, you will use the token recorded locally. Due to the time limit of the token, when the token expires, if there is network fluctuation or moomoo background release, it may cause the situation that it cannot be automatically connected after disconnecting from the background. Therefore, if you want visiulization OpenD for a long time to hang up, it is recommended to manually enter the password to log in, and OpenD will automatically handle this situation.

Q13: How to request official engineers to investigate logs when encountering product defects?

A:

1. Contact OpenAPI developers via QQ/WeChat to facilitate instant communication and file transferation.
2. Detailed description: the time when the error occurred, OpenD version number, moomoo API version number, script language name, interface name or protocol number, short code or screenshot with detailed input and return.
3. If necessary, OpenD log must be provided to facilitate location and confirmation of problems. Transaction issues require info log level, and market issues require debug log level. The log level log_level can be configured in ***OpenD.xml*** **Configure**. After configuration, OpenD needs to be restarted to take effect. After the problem recurs, package the log and send it to moomoo R&D personnel.

> **Tips**
>
> The log path is as follows:

> windows: `%appdata%/com.moomoo.OpenD/Log`
>
> Non-windows: `~/.com.moomoo.OpenD/Log`

Q14: Script cannot connect to OpenD

A: Please try to check first:

1. Whether the port connected by the script is consistent with the port configured by OpenD.
2. Since the upper limit of OpenD connection is 128, is there any useless connection that is not closed?
3. Check whether the listening address is correct. If the script and OpenD are not on the same machine, the OpenD listening address needs to be set to 0.0.0.0.

Q15: Disconnected after being connected for a while

A: If it is a self-docking protocol, check whether there is a regular heartbeat to maintain the connection.

Q16: I can't connect to OpenD when I run Python scripts in multiprocessing mode through the multiprocessing module under Linux?

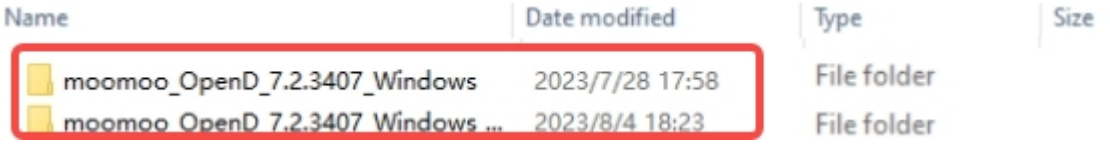
A: After the process is created by default in the Linux/Mac environment, the thread created inside py-moomoo-api in the parent process will disappear in the child process, resulting in an internal program error. You can use spawn to start the process:

```py
import multiprocessing as mp
mp.set_start_method('spawn')
p = mp.Process(target=func)
```

Q17: How to log in to two OpenD at the same time on one computer?

A: Visualization OpenD does not support, but Command Line OpenD supports.

1. Unzip the file downloaded from the official website, and copy the entire Command Line OpenD folder (e.g. ***moomoo_OpenD_5.2.1408_Windows***). Take Windows as an example, other systems can do the same operation.

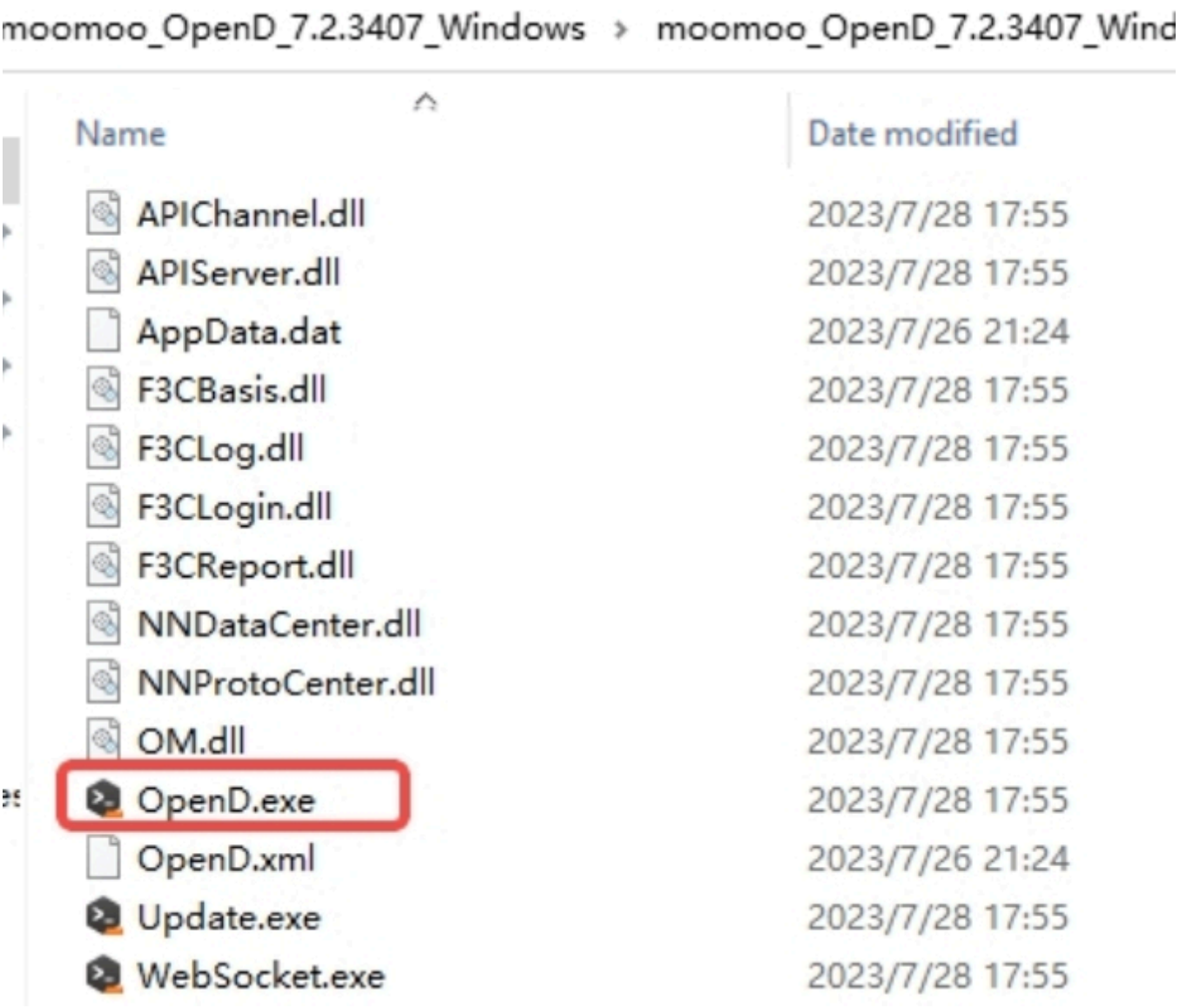


2. Configure two ***OpenD.xml*** files that are placed in two Command Line OpenD folders. Configure items as follow:

Configuration file 1: api_port = 11111, login_account = Login Account 1, login_pwd = Login Password 1

Configuration file 2: api_port = 11112, login_account = Login Account 2, login_pwd = Login Password 2

3. Run the two OpenD.exe.



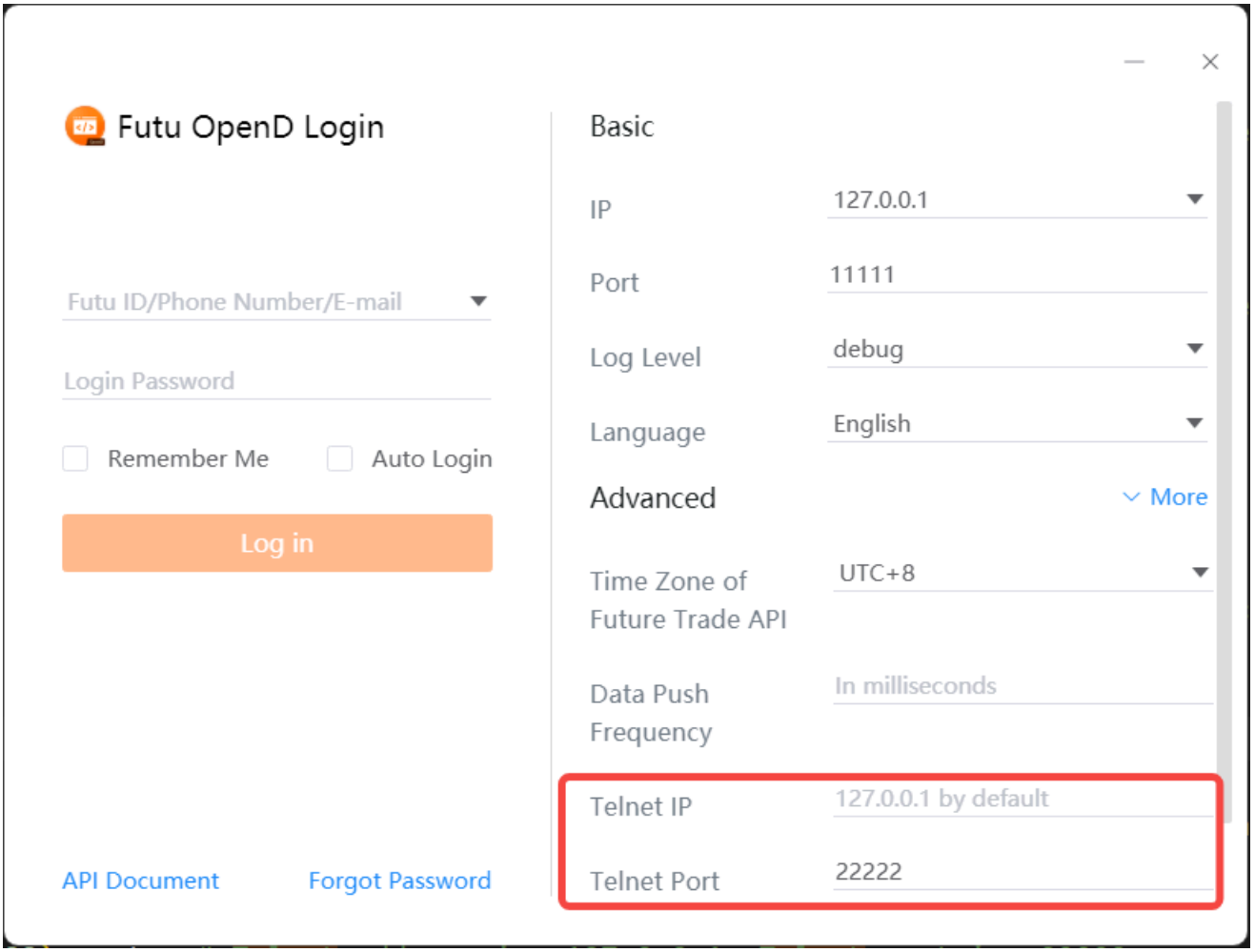
4. When calling the interface, note that the parameter **`port`** (OpenD listening address) should corresponds to the parameter **`api_port`** in the *OpenD.xml file*.
   For example:

```py
from moomoo import *

# Send requests to OpenD logged in account 1
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111, is_encrypt=False)
quote_ctx.close() # After using the connection, remember to close it to prevent t

# Send requests to OpenD logged in account 2
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11112, is_encrypt=False)
quote_ctx.close() # After using the connection, remember to close it to prevent t
```

Q18: How do I execute the operation and maintenance commands for grabbing permissions through scripts when the market permission is kicked off by other clients?

A:

1. Configure Telnet address and Telnet port.



FutuOpenD.xml

```
<futu_opend>
    <!-- 基础参数 -->
    <!-- Basic parameters -->
        <!-- 协议监听地址,不填默认127.0.0.1 -->
        <!-- Listening address. 127.0.0.1 if not specified -->  // Listening address. 127.0.0.1 by default
        <ip>127.0.0.1</ip>
        <!-- API接口协议监听端口 -->
        <!-- API interface protocol listening port -->
        <api_port>11111</api_port>
        <!-- 登录帐号 -->
        <login_account>100000</login_account>
        <!-- 登录密码32位MD5加密16进制 -->
        <!-- <login_pwd_md5>6e55f158a827b1a1c4321a245aaaad88</login_pwd_md5> -->
        <!-- 登录密码明文，密码密文存在情况下只使用密文 -->
        <login_pwd>123456</login_pwd>
        <!-- FutuOpenD语言，en：英文，chs：简体中文 -->
        <lang>chs</lang>
    <!-- 进阶参数 -->
    <!-- Advanced parameters -->
        <!-- FutuOpenD日志等级，no,debug,info,warning,error,fatal -->
        <log_level>info</log_level>
        <!-- API推送协议格式，0:pb, 1:json -->
        <!-- API push protocol format, 0:pb, 1:json -->
        <push_proto_type>0</push_proto_type>
        <!-- API订阅数据推送频率控制，单位毫秒，目前不包括K线和分时，不设置则不限制频率-->
        <!-- API subscription data push frequency control, in milliseconds, currently does not include K-line and time-sharing, if not
        <!-- <qot_push_frequency>1000</qot_push_frequency> -->
        <!-- Telnet监听地址,不填默认127.0.0.1 -->
        <!-- Telnet listening address, default 127.0.0.1 if not filled in -->  // Telnet listening address, 127.0.0.1 by default
        <telnet_ip>127.0.0.1</telnet_ip>
        <!-- Telnet监听端口 -->
        <!-- Telnet listening port -->
        <telnet_port>22222</telnet_port>
        <!-- API协议加密私钥文件路径,不设置则不加密 -->
        <!-- API protocol encrypted private key file path, if not set, it will not be encrypted -->  // File path for private key for
        <!-- <rsa_private_key>D:\rsa</rsa_private_key> -->
        <!-- 是否接收到价提醒推送，0：不接收，1：接收 -->
        <!-- Whether to receive the price reminder push, 0: not receive, 1: receive -->
```

2. Start OpenD (it will also start Telnet).
3. After finding that the market quotation authority has been robbed, you can refer to the following code example and send the `request_highest_quote_right` command to OpenD via Telnet.

```py
from telnetlib import Telnet
with Telnet('127.0.0.1', 22222) as tn: # Telnet address is: 127.0.0.1, Telnet por
    tn.write(b'request_highest_quote_right\r\n')
    reply = b''
    while True:
        msg = tn.read_until(b'\r\n', timeout=0.5)
        reply += msg
        if msg == b'':
            break
    print(reply.decode('gb2312'))
```

Q19: OpenD automatic upgrade failed

A: The automatic update of OpenD failed to be executed by the `update` command. Possible reasons:

- The file is occupied by other processes: you can try to close other OpenD processes, or restart the system and execute `update` again If the above still cannot be solved, you can download the update by yourself through Official Website.

Q20: Fail to launch the visualization OpenD on ubuntu22?

A: When running the visualization OpenD on certain Linux distributions (such as Ubuntu 22.04), you may encounter the error: `dlopen(): error loading libfuse.so.2`. This occurs because libfuse is not installed by default on these systems. Typically you can resolve this issue by installing libfuse manually. For example, you can install it via the commane line on Ubuntu22.04 with:

```
sudo apt update
sudo apt install -y libfuse2
```

Once successfully installed, you will be able to run the visualization OpenD normally. Please refer to https://docs.appimage.org/user-guide/troubleshooting/fuse.html for more details.

Q21: How to run the command line OpenD in the background on Linux?

A: First, switch to the directory where OpenD is located, configure OpenD.xml, and then execute the following command.

```
nohup ./OpenD &
```

Quote related

Q1: Subscription failed

A: When the subscription interface returns an error, there are two common situations.

- Insufficient subscription quota

  Please refer to Subscription Quota & Historical Candlestick Quota for the subscription quota rules.

- Insufficient quota right

  The quota right that supports subscription are shown in the following table:

Market	Contracts	Quota Right for Subscription
HK Market	Secrities	LV1, LV2, SF
	Options	LV1, LV2
	Futures	LV1, LV2
US Market	Secrities	LV1, LV2
	Options	LV1
	Futures	LV1, LV2
A-Share Market	Secrities	LV1

  Please refer to Quote Right for the access method.

  Note: If your account has the above-mantioned quota rights, but the subscription still fails. The possible reason is that the quota right has been kicked out by other terminals.

Q2: Unsubscribe failed

A: You can unsubscribe after you subscribe for at least one minute.

Q3: The unsubscribe was successful but the quota was not released

A: The quota is released after all connections are unsubscribed to the market.

For example: Connection A and Connection B are both subscribing to HK.00700's listing data. After Connection A is unsubscribed, because Connection B is still calling HK.00700's listing data, the OpenD quota will not be released until all connections have been unsubscribed the listing data of HK.00700.

Q4: Will the quota be released if the script connection is closed if the subscription is less than one minute?

A: No. After the connection is closed, the target type whose subscription duration is less than one minute will be automatically unsubscribed after reaching one minute, and the corresponding subscription quota will be released.

Q5: What is the specific restriction logic for requesting frequency restriction?

A: At most n times within 30 seconds, it means that the interval between the first request and the n+1th request must be greater than 30 seconds.

Q6: What is the reason why self-selected stocks cannot be added?

A: Please check whether the upper limit is exceeded first, or delete part of the self-selected stocks.

Q7: Why is the US stock quotation on the OpenAPI side different from the national comprehensive quotation on the client side?

A: Since US stock trade are scattered on many exchanges, moomoo provides two basic quotations for US stocks, one is Nasdaq Basic (quotes on the Nasdaq exchange), and the other is a comprehensive quotation for the United States (all 13 exchanges in the United States). However, OpenAPI's US stock quote currently only support Nasdaq Basic purchases through quotation card, and do not support comprehensive US quote.
Therefore, if you purchase both the US comprehensive quotation card on the APP side and the Nasdaq Basic quotation card that is only used for OpenAPI, there may indeed be a difference in the quotation between the APP side and the OpenAPI side.
Therefore, if you notice a discrepancy between the opening price of U.S. stocks and the price displayed on the App, this is because the OpenAPI real-time upstream market data only retrieves Nasdaq Basic.

Q8: Where can I buy OpenAPI quotation cards?

A:

- HK market
  - HK stocks LV2 advanced market (only non-Chinese mainland IP)
  - HK stock options futures LV2 advanced market (only non-Mainland China IP)
  - HK stocks LV2 + option futures LV2 market (non-Mainland China IP only)
  - HK Stocks Advanced Full Market Quotes (SF Quotes)
- US market
  - US stocks Nasdaq Basic
  - US stocks Nasdaq Totalview
  - US options OPRA real-time data

Q9: Why sometimes, the response of the get interface to obtain real-time data is slow?

A: Because the get interface for real-time data needs to be subscribed first, and it depends on the push to OpenD from the background. If the user uses the get interface to request immediately after subscribing, OpenD may not receive the background push yet. In order to prevent this from happening, the get interface has built-in waiting logic, and the push will be returned to the script immediately after receiving the push within 3 seconds, and empty data will be returned to the script if the background push is not received for more than 3 seconds. The get interfaces include: get_rt_ticker, get_rt_data, get_cur_kline, get_order_book, get_broker_queue, get_stock_quote. Therefore, when you find that the response of the get interface for obtaining real-time data is relatively slow, you can first check whether it is the cause of no trade data.

Q10: What kind of data can be obtained after purchasing the OpenAPI Nasdaq Basic quotation card?

A: After the Nasdaq Basic quotation card purchase is activated, the categories that can be obtained include Nasdaq, NYSE, NYSE MKT stocks listed on the exchange (including US stocks and ETF, excluding US stock futures and US stock options). Supported data interfaces include: snapshots, historical candlestick, real-time ticker subscriptions, real-time one-stage subscriptions, real-time candlestick subscriptions, real-time quotation subscriptions, real-time Time Frame subscriptions, and price reminders.

Q11: How many levels does each market category support?

A:

Quotes category	LV1	LV2	SF
HK stocks (including Stock, Warrants, bulls and bears, and inbound securities)	/	10	full stock + thousand details
HK stock options futures	1	10	/
US stocks (including ETF)	1	60	/
US stock options	1	/	/
US futures	/	40	/

Quotes category	LV1	LV2	SF
A-shares	5	/	/

Q12：Why does OpenD still have no quote right after I purchase and activate the quotation card?

A:

1. The quote right of OpenAPI is not exactly the same as that of APP. Some quotation cards are only applicable to the APP side (e.g., OpenAPI quotation cards for US stocks need to be purchased separately). Please confirm that the card you purchased is applicable to OpenD first. We have listed **all** the quotation cards applicable to OpenAPI in the section *Authorities and Limitations*. Please click here.

2. After activating the quotation card, your quote right will be effective immediately. Please check **after restarting OpenD**.

Q13：How to Get Real-time Quotes Through Subscription Interface?

The First Stop：Subscription

Pass the code of underlying security and data type to Subscription Interface to finish subscribing.

Subscription interface supports requesting real-time quote, real-time order book, real-time tick-by-tick, real-time Time Frame, real-time candlesticks and real-time broker queue. After a successful subscription, OpenD will continuously receive real-time data from Futo Server.

Attention: The subscription quota is allocated by your total capital, trading amount and trading volume. Please refer to Subscription Quota & Historical Candlestick Quota for details. If your subscription quota is not enough, please check if there is any useless subscriptions in the quota. Unsubscribe to release the subscription quota in time.

The Second Step：Obtain Data

We provide two methods to obtain subscribed data from OpenD:

Methos 1: Real-time data Callback
Set corresponding callback functions to process the pushed data asyncronously.

After the callback function is set, OpenD will immediately push the received real-time data to the callback function of the script for processing.

If the underlying security is very active, you may get a large amount of pushed data with high frequency. If you want to slower the push frequency of OpenD, we recommand you to config push frequency(`qot_push_frequency`) in OpenD Startup Parameter

The interfaces involved in mode 1 include: Real-time Quote Callback, Real-time Order Book Callback, Real-time Candlestick Callback, Real-time Time Frame Callback, Real-time Tick-by-Tick Callback, Real-time Broker Queue Callback.

Method 2: Get Real-time Data
Through the access to real-time data interface, you can use scripts to get the latest data received by OpenD. This approach is more flexible, and scripts do not need to deal with massive pushes. As long as OpenD continues to receive push from servers, the script can obtain the data on demand.

As the data is taken from the pushed data received by OpenD, there is no frequency limit for this type of interface.

The interfaces involved in mode 1 include: Get Real-time Quote of Securities, Get Real-time Order Book, Get Real-time Candlestick, Get Real-time Time Frame Data, Get Real-time Tick-by-Tick, Get Real-time Broker Queue.

Q14：What time period corresponds to each market state?

A:

Market	Security Type	Market State	Time Period (Local time)
HK Market	Securities (including stocks, ETFs,	* NONE: No trading	CST 08:55 - 09:00
		* ACTION: Pre-market trading	CST 09:00 - 09:20

	warrants, CBBCs, Inline Warrants)	* WAITING_OPEN: Waiting for opening	CST 09:20 - 09:30
		* MORNING: Morning session	CST 09:30 - 12:00
		* REST: Lunch break	CST 12:00 - 13:00
		* AFTERNOON: Afternoon session	CST 13:00 - 16:00
		* HK_CAS: After-hours bidding for HK stocks (The market state corresponding to the addition of CAS mechanism to the Hong Kong stock market)	CST 16:00 - 16:08
		* CLOSED: Market closed	CST 16:08 - 08:55 (T+1)
	Options, Futures (Day Market only)	* NONE: Waiting for options opening	CST 08:55 - 09:30
		* MORNING: Morning session	CST 09:30 - 12:00
		* REST: Lunch break	CST 12:00 - 13:00
		* AFTERNOON: Afternoon session	CST 13:00 - 16:00
		* CLOSED: Market closed	CST 16:00 - 08:55 (T+1)
	Futures (Day and Night Market)	* FUTURE_DAY_WAIT_FOR_OPEN: Futures market wait for opening	Different trading time for different species
		* NIGHT_OPEN: Night market trading hours	
		* NIGHT_END: Night market closed	
		* FUTURE_DAY_WAIT_FOR_OPEN: Futures market wait for opening	

		* FUTURE_DAY_OPEN: Day market trading hours	
		* FUTURE_DAY_CLOSE: Day market closed	
US Market	Securities (including stocks, ETFs)	* PRE_MARKET_BEGIN: Pre-market trading	EST 04:00 - 09:30
		* AFTERNOON: Regular trading hours	EST 09:30 - 16:00
		* AFTER_HOURS_BEGIN: After-hours trading	EST 16:00 - 20:00
		* AFTER_HOURS_END: Market closed of U.S. stock market	EST 20:00 - 04:00 （T+1）
		* OVERNIGHT: Overnight trading session of U.S. stock market	EST 20:00 - 04:00 （T+1）
	Options	* NONE: Waiting for options opening	Different trading time for different species
		* REST：Lunch break	
		* AFTERNOON: Regular trading hours	
		* TRADE_AT_LAST: Late trading hours	
		* NIGHT: Night market trading hours	
		* CLOSED: Market closed	
	Futures	* NONE: Waiting for U.S. futures opening	Different trading time for different species
		* FUTURE_OPEN: Trading hours of U.S. futures	

		* FUTURE_BREAK: Break of U.S. futures	
		* FUTRUE_BREAK_OVER: Trading hours of U.S. futures after break	
		* FUTURE_CLOSE: Market closed of U.S. futures	
A-share Market	Securities (including stocks, ETFs)	* NONE: No trading	CST 08:55 - 09:15
		* Action: Pre-market trading	CST 09:15 - 09:25
		* WAITING_OPEN: Waiting for opening	CST 09:25 - 09:30
		* MORNING: Morning session	CST 09:30 - 11:30
		* REST: Lunch break	CST 11:30 - 13:00
		* AFTERNOON: Afternoon session	CST 13:00 - 15:00
		* CLOSED: Market closed	CST 15:00 - 08:55 (T+1)
Singapore Market	Futures	* FUTURE_DAY_WAIT_FOR_OPEN: Futures market wait for opening	Different trading time for different species
		* NIGHT_OPEN: Night market trading hours	
		* NIGHT_END: Night market closed	
		* FUTURE_DAY_OPEN: Day market trading hours	
		* FUTURE_DAY_CLOSE: Day market closed	

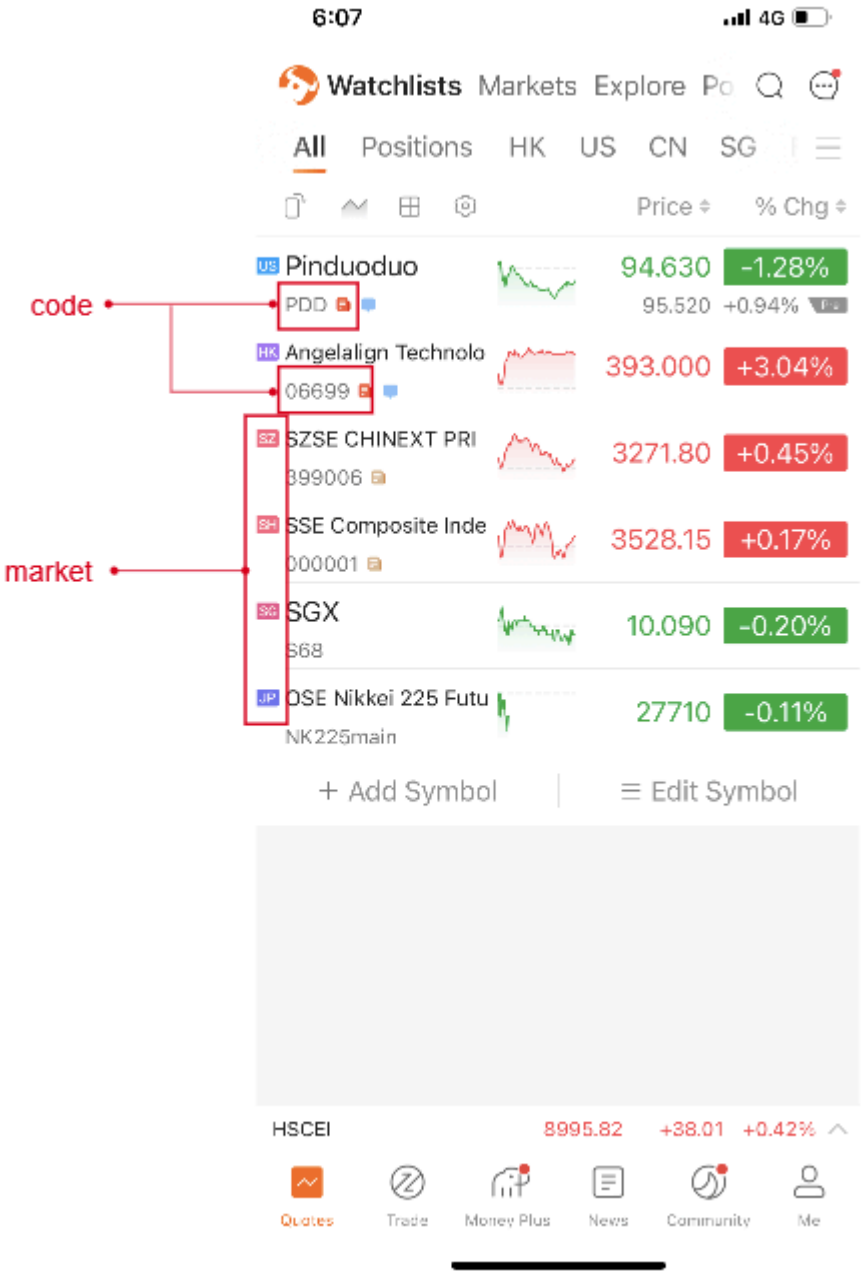
Japanese Market	Futures	* FUTURE_DAY_WAIT_FOR_OPEN: Futures market wait for opening	JST 16:25（T-1）- 16:30（T-1）
		* NIGHT_OPEN: Night market trading hours	JST 16:30（T-1） - 05:30
		* NIGHT_END: Night market closed	JST 05:30 - 08:45
		* FUTURE_DAY_OPEN: Day market trading hours	JST 08:45 - 15:15
		* FUTURE_DAY_CLOSE: Day market closed	JST 15:15 - 16:25

* CST, EST, JST represent China time, US Eastern time, and Japan time respectively.

Q15：Parameter format of stock code.

A：

- For users with different programming languages, parameter format of stock code is different.
  - **Python users**
    Format of stock code: `market.code` .
    For example: Tencent. Parameter `code` should be passed in 'HK.00700'.
  - **Non-Python users**
    For stock structure, refer to Security.
    For example: Tencent. Parameter `market` should be passed in QotMarket_HK_Security, parameter `code` should be passed in '00700'.
- Quick inquiries. View the code and market through APP: Quotes > Watchlists > All.
  For Quote Market, refer to here.



Q16：Stock Price Adjustment

A:

Overview

Price adjustment refers to adjusting stock price and trading volume after corporate actions, so that the price chart can better represent actual price moves and trading volume.
Corporate actions such as stock split, reverse stock split, bonus issue, rights issue, allotment, secondary offering, and dividend payment can affect the stock price. Price adjustment

eliminates the impact of corporate actions on stock price and trading volume, and maintains the continuity of the stock price moves.

> **Tips**
>
> - The information on this page is mainly intended for the China A-share market.

Glossary

- Corporate action: Actions on equity and stock conducted by a listed company that affect the company's stock price and number of shares.
- Default adjustment: Keep the current stock price unchanged, and use it as the benchmark to re-calculate all previous stock prices.
- Cumulative adjustment: Keep the stock price before the earliest corporate action unchanged, and use it as the benchmark to re-calculate all future stock prices.
- Price adjustment factor: The ratio used to re-calculate the adjusted and cumulative stock prices and number of shares after a corporate action. There are two types of price adjustment factors: the default adjustment factor for calculating the adjusted price and the cumulative adjustment factor for calculating the cumulative price.
- Ex-div and pay date: The next trading day of the registration date. The stock exchange must calculate the adjusted stock price before market open on the ex-and-pay date. It is also the date on which dividends are distributed to shareholders and changes in the number of shares take place.

Price Adjustment Methods

There are two price adjustment methods: two-step method and continuous multiplication. OpenAPI uses different adjustment methods for different markets.

- Two-step method: The stock price is adjusted based on corporate actions; there are 2 factors in this method: factor A for cash dividends and factor B for all other corporate actions.
- Continuous multiplication: The stock price is adjusted the continuously multiplying the adjustment factors. This method can be seen as a special case of two-step method with factor B as 0.

> **Tips:**

- OpenAPI uses continuous multiplication for calculating the adjusted price of US stocks, with the price adjustment factor B set to 0.
- OpenAPI uses two-step method for stocks other than US stocks (China A-shares, Hong Kong stocks, Singapore stocks, etc.) and for calculating the cumulative price of US stocks.

Calculation Formulae

Single Adjustment

- Default adjustment:
  Adjusted price = Actual price × Default adjustment factor A + Default adjustment factor B
- Cumulative adjustment:
  Cumulative price = Actual price × Cumulative adjustment factor A + Cumulative adjustment factor B

Multiple Price Adjustments

- Default adjustment: In chronological order, select the adjustment factors later than the adjustment date, and first use earlier adjustment factors for calculation. Take a double adjustment as an example:

$$Price_n^{adjusted} = (Price_n * FactorA_{n+1}^{adjusted} + FactorB_{n+1}^{adjusted}) * FactorA_{n+2}^{adjusted} + FactorB_{n+2}^{adjusted}$$

$Price_n^{adjusted}$: Adjusted price of the day

$Price_n$: Actual price of the day

$FactorA_{n+1}^{adjusted}$: Default adjustment factor A of the next day

$FactorB_{n+1}^{adjusted}$: Default adjustment factor B of the next day

$FactorA_{n+2}^{adjusted}$: Default adjustment factor A of the next two days

$FactorB_{n+2}^{adjusted}$: Default adjustment factor B of the next two days

- Cumulative adjustment: In reverse chronological order, select the adjustment factors earlier than or on the calculation date, and first use later adjustment factors for calculation. Take a double adjustment as an example:

$$Price_n^{cumulative} = (Price_n * FactorA_n^{cumulative} + FactorB_n^{backward}) * FactorA_{n-1}^{cumulative} + FactorB_{n-1}^{cumulative}$$

$Price_n^{cumulative}$: Cumulative price of the day

$Price_n$: Actual price of the day

$FactorA_n^{cumulative}$: Cumulative adjustment factor A of the day

$FactorB_n^{cumulative}$: Cumulative adjustment factor B of the day

$FactorA_{n-1}^{cumulative}$: Cumulative adjustment factor A of the previous day

$FactorB_{n-1}^{cumulative}$: Cumulative adjustment factor B of the previous day

Examples

Example of a single adjustment

Take the stock of Muyuan Foods as an example:

- Screening weighting factors are as follows:

Ex-Div and Pay Date	Stock Symbol	Corporate Action Details	Default Adjustment Factor A	Default Adjustment Factor B
06/03/2021	SZ.002714	10-share dividends: 4 shares and ￥14.61 (tax included)	0.71429	-1.04357

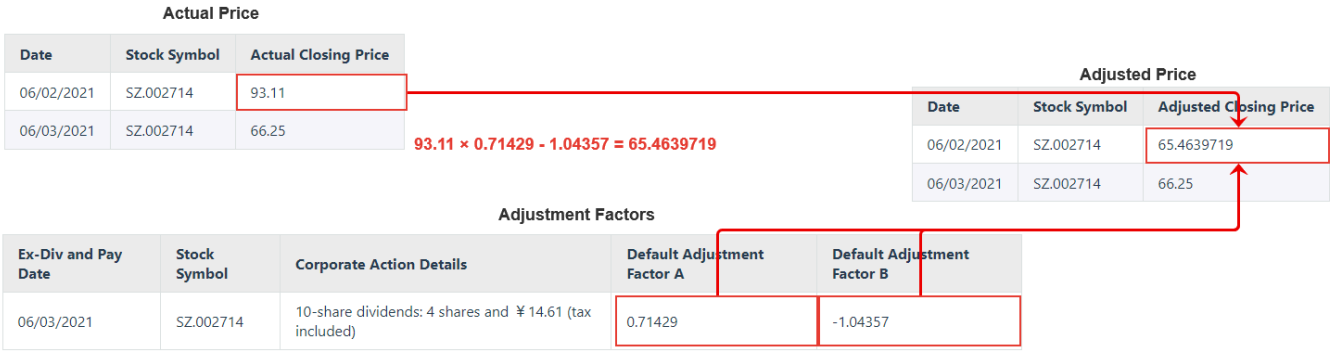
- Data on actual price:

Date	Stock Symbol	Actual Closing Price
06/02/2021	SZ.002714	93.11
06/03/2021	SZ.002714	66.25

- Data on adjusted prices:

Date	Stock Symbol	Adjusted Closing Price
06/02/2021	SZ.002714	65.4639719
06/03/2021	SZ.002714	66.25

- Method for calculating adjusted prices:
  Muyuan Foods conducted a stock split and paid cash dividends on 2021/06/03 (4 shares and ￥14.61 for every 10 shares owned), and here is how to calculate the adjusted closing price on 06/02/2021: Adjusted price (65.4639719) = Actual price (93.11) × Default adjustment factor A (0.71429) + Default adjustment factor B (-1.04357)



Example of multiple cumulative adjustment

Following on the previous example, here is how to calculate the cumulative price of Muyuan Foods on 06/02/2021:

- Adjustment factors are as follows:

Ex-Date	Stock Symbol	Corporate Action Details	Cumulative Factor A	Cumulative Factor B
07/04/2014	SZ.002714	10-share dividends: ￥2.34 (tax included)	1	0.234
06/10/2015	SZ.002714	10-share dividends: 10 shares and ￥0.61 tax included)	2	0.061
07/08/2016	SZ.002714	10-share dividends: 10 shares and ￥3.53 tax included) (tax included)	2	0.353
07/11/2017	SZ.002714	10-share dividends: 8 shares and ￥6.9 (tax included)	1.8	0.69
07/03/2018	SZ.002714	10-share dividends: ￥6.91 (tax included)	1	0.691

Ex-Date	Stock Symbol	Corporate Action Details	Cumulative Factor A	Cumulative Factor B
07/04/2019	SZ.002714	10-share dividends: ￥0.5 (tax included)	1	0.05
06/04/2020	SZ.002714	10-share dividends: 7 shares and ￥5.5 (tax included)	1.7	0.55

- Data on actual prices:

Date	Stock Symbol	Actual Price
06/02/2021	SZ.002714	93.11

- Data on cumulative prices:

Date	Stock Symbol	Cumulative Price
06/02/2021	SZ.002714	1152.7226

- Method for calculating cumulative prices:
  To calculate the cumulative price of Muyuan Foods on June 2, 2021, all the corporate actions by June 2, 2021 need to be taken into account. The detailed calculations are as follows:

Actual Price

Date	Stock Symbol	Actual Price
02/06/2021	SZ.002714	93.11

Cumulative Price

Date	Stock Symbol	Cumulative Price
02/06/2021	SZ.002714	1152.7226

((((((93.11 x 1.7 + 0.55) x 1 + 0.05) x 1 + 0.691) x 1.8 + 0.69) x 2 + 0.353) x 2 + 0.061) x 1 + 0.234 = 1152.7226

Adjustment Factors

Ex-Date	Stock Symbol	Corporate Action Details	Cumulative Factor A	Cumulative Factor B
07/04/2014	SZ.002714	10-share dividends: ￥2.34 (tax included)	1	0.234
06/10/2015	SZ.002714	10-share dividends: 10 shares and ￥0.61 tax included)	2	0.061
07/08/2016	SZ.002714	10-share dividends: 10 shares and ￥3.53 tax included) (tax included)	2	0.353
07/11/2017	SZ.002714	10-share dividends: 8 shares and ￥6.9 (tax included)	1.8	0.69
07/03/2018	SZ.002714	10-share dividends: ￥6.91 (tax included)	1	0.691
07/04/2019	SZ.002714	10-share dividends: ￥0.5 (tax included)	1	0.05
06/04/2020	SZ.002714	10-share dividends: 7 shares and ￥5.5 (tax included)	1.7	0.55

Transaction related

Q1: How to use paper trading?

A:

Overview

Paper trading is a simulation that allows you to practice trading without the risk of using real money.

Trading time

Paper trading only supports trading during regular trading hours, and does not support trading outside regular trading hours, US market pre-market and after-hours, HK market and China A-shares market Opening and Closing Auction. For details, please click **Rules of paper trading**.

Categories supported

For categories that OpenAPI supports by paper trading, please click **here**.

Unlock Trade

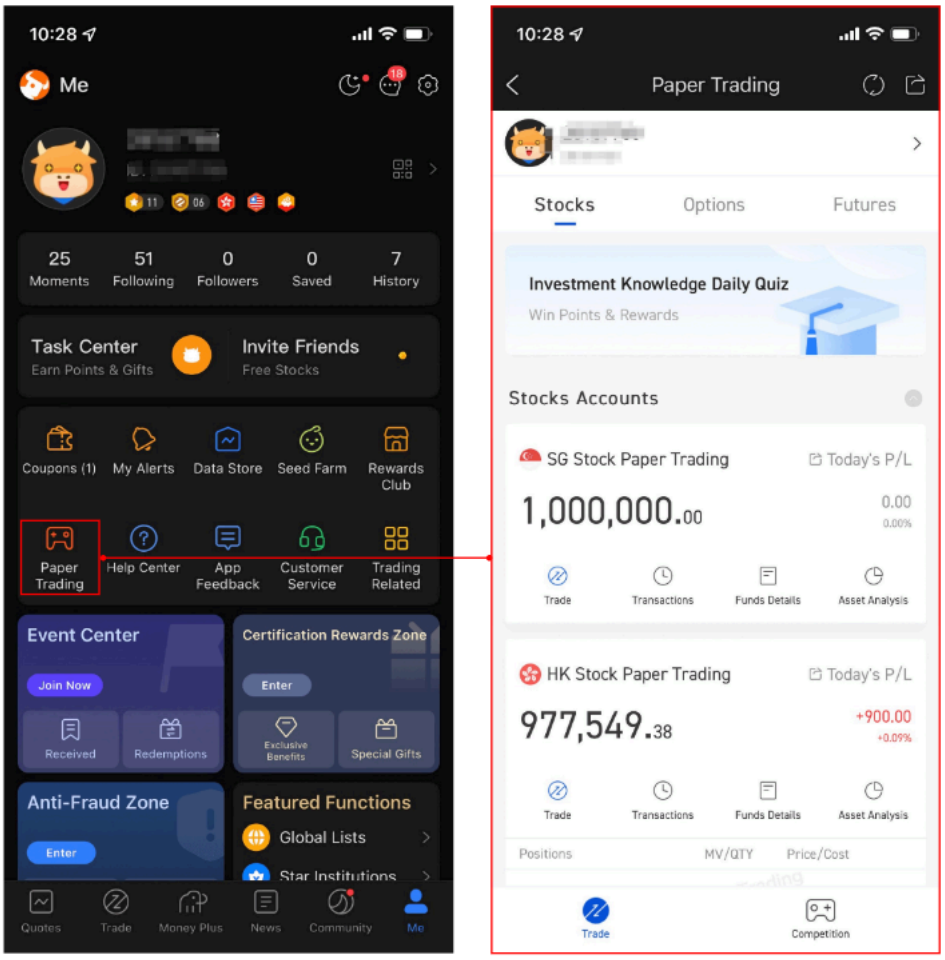
Different from live trading, you do not need to unlock the account to place orders or modify or cancel orders when using paper trading.

Orders

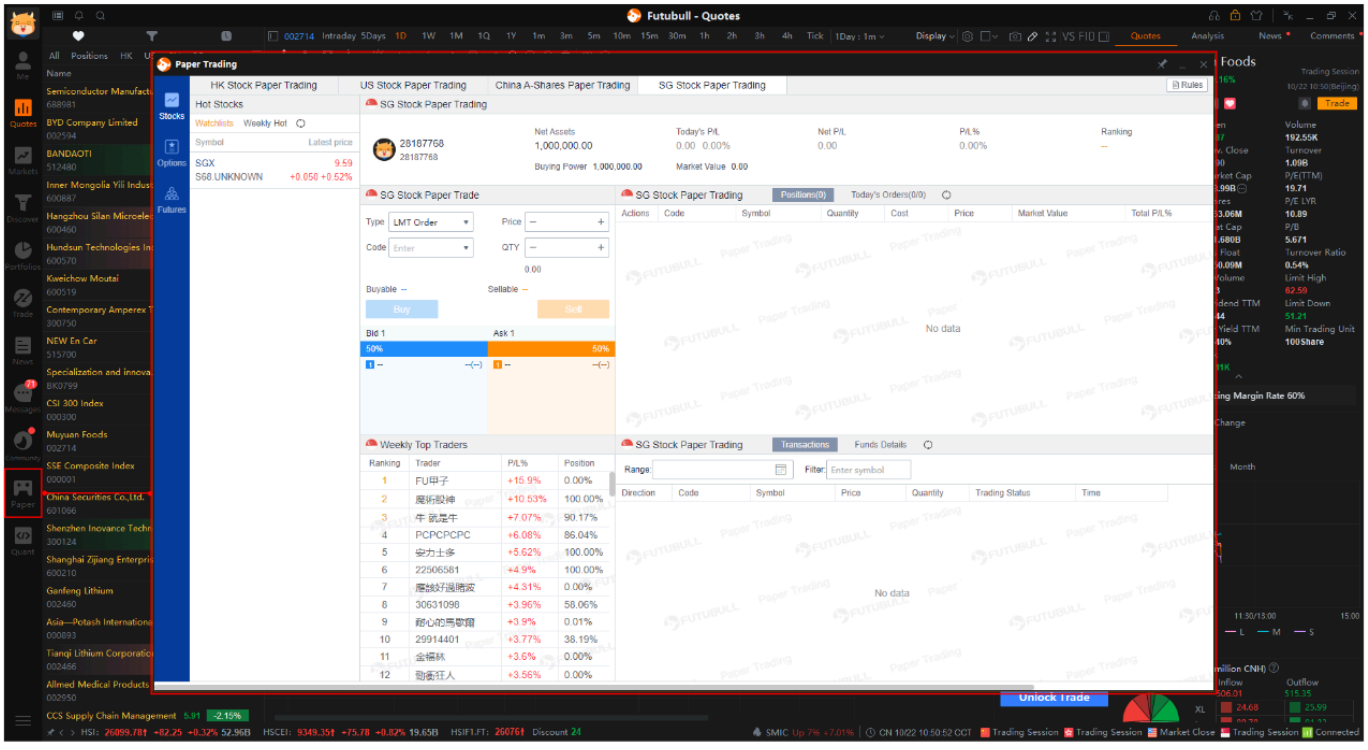
1. Order Types: limit order and market order.
2. Modify Order Operation: Paper trading does not support enabling, disabling, and deleting the order, but supports modifying and canceling the order.
3. Deals: Paper trading does not support deals related operations, including **Get today's deals**, **Get historical deals**, and **Respond to the transaction push**.
4. Valid Period: Paper trading only supports good for day order when setting valid period.
5. Short Selling: Options and futures support short selling. Only US stocks support short selling.

Platform

1. Mobile clients: Me — Paper Trading.



2. Desktop clients: Left side tab *Paper* .



3. Web clients: Paper Trading Website.

4. OpenAPI: When calling the interface, set the parameter trading environment to the simulated environment. Click How to use paper trading through OpenAPI for detail.

> Tips
>
> - The four platforms shown above use the same paper trading accounts.

How to use paper trading through OpenAPI?

Create Connection

Firstly, create the corresponding connection. When the underlyings are stocks or options, please use `OpenSecTradeContext`. When the underlyings are futures, please use `OpenFutureTradeContext`.

Get the List of Trading Accounts

Use the interface Get the List of Trading Accounts to view trading accounts (including paper trading accounts and live trading accounts). Take Python as an example: When the returned parameter `trd_env` is `SIMULATE`, it means the corresponding account is a paper trading account.

- Example: Stocks and Options

```py
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, host='127.0.0.1', port=11111, security_firm=SecurityFirm.FUTUSECUR
#trd_ctx = OpenFutureTradeContext(host='127.0.0.1', port=11111, is_encrypt=None, security_firm=SecurityFirm.FUTUSECURITIES)
ret, data = trd_ctx.get_acc_list()
if ret == RET_OK:
    print(data)
    print(data['acc_id'][0])  # get the first account id
    print(data['acc_id'].values.tolist())  # convert to list format
else:
    print('get_acc_list error: ', data)
trd_ctx.close()
```

- Output

```py
              acc_id   trd_env acc_type          card_num   security_firm  \
0  281756480572583411      REAL   MARGIN  1001318721909873  FUTUSECURITIES
1             9053218  SIMULATE     CASH               N/A             N/A
2             9048221  SIMULATE   MARGIN               N/A             N/A

  sim_acc_type  trdmarket_auth
0          N/A  [HK, US, HKCC]
1        STOCK            [HK]
2       OPTION            [HK]
```

> Tips
>
> - In paper trading, stock accounts and options accounts are distinguished. Stock accounts can only trade stocks, and options accounts can only trade options; take Python as an example: `sim_acc_type` in the returned field is `STOCK`, which means stock account; `OPTION` means option account.

- Example: Futures

```py
from moomoo import *
#trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, host='127.0.0.1', port=11111, security_firm=SecurityFirm.FUTUSECU
trd_ctx = OpenFutureTradeContext(host='127.0.0.1', port=11111, is_encrypt=None, security_firm=SecurityFirm.FUTUSECURITIES)
ret, data = trd_ctx.get_acc_list()
if ret == RET_OK:
    print(data)
    print(data['acc_id'][0])  # get the first account id
    print(data['acc_id'].values.tolist())  # convert to list format
```

```py
else:
    print('get_acc_list error: ', data)
trd_ctx.close()
```

- Output

```
    acc_id   trd_env acc_type card_num security_firm sim_acc_type  \
0  9497808  SIMULATE   MARGIN      N/A           N/A      FUTURES
1  9497809  SIMULATE   MARGIN      N/A           N/A      FUTURES
2  9497810  SIMULATE   MARGIN      N/A           N/A      FUTURES
3  9497811  SIMULATE   MARGIN      N/A           N/A      FUTURES

          trdmarket_auth
0  [FUTURES_SIMULATE_HK]
1  [FUTURES_SIMULATE_US]
2  [FUTURES_SIMULATE_SG]
3  [FUTURES_SIMULATE_JP]
```

Place Orders

When using the interface Place Orders, set the trading environment to the simulated environment. Take Python as an example: `trd_env = TrdEnv.SIMULATE`.

- Example

```py
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, host='127.0.0.1', port=11111, security_firm=SecurityFirm.FUTUSECUR
ret, data = trd_ctx.place_order(price=510.0, qty=100, code="HK.00700", trd_side=TrdSide.BUY, trd_env=TrdEnv.SIMULATE)
if ret == RET_OK:
    print(data)
else:
    print('place_order error: ', data)
trd_ctx.close()
```

- Output

```
    code stock_name    trd_side order_type    order_status  order_id qty  price    create_time    updated_time  dealt_qty
0  HK.00700 Tencent   BUY  NORMAL    SUBMITTING    4642000476506964749    100.0    510.0    2021-10-09 11:34:54    2021-10-09
```

Modify or Cancel Orders

When using the Interface Modify or Cancel Orders, set the trading environment to the simulated environment. Take Python as an example: `trd_env = TrdEnv.SIMULATE`.

- Example

```py
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, host='127.0.0.1', port=11111, security_firm=SecurityFirm.FUTUSECUR
order_id = "4642000476506964749"
ret, data = trd_ctx.modify_order(ModifyOrderOp.CANCEL, order_id, 0, 0, trd_env=TrdEnv.SIMULATE)
if ret == RET_OK:
    print(data)
else:
    print('modify_order error: ', data)
trd_ctx.close()
```

- Output

```
    trd_env            order_id
0  SIMULATE  464200047650696474 9
```

Get Historical Orders

When using the Interface Get Historical Orders, set the trading environment to the simulated environment. Take Python as an example: `trd_env = TrdEnv.SIMULATE` .

- Example

```py
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, host='127.0.0.1', port=11111, security_firm=SecurityFirm.FUTUSECUR
ret, data = trd_ctx.history_order_list_query(trd_env=TrdEnv.SIMULATE)
if ret == RET_OK:
    print(data)
else:
    print('history_order_list_query error: ', data)
trd_ctx.close()
```

- Output

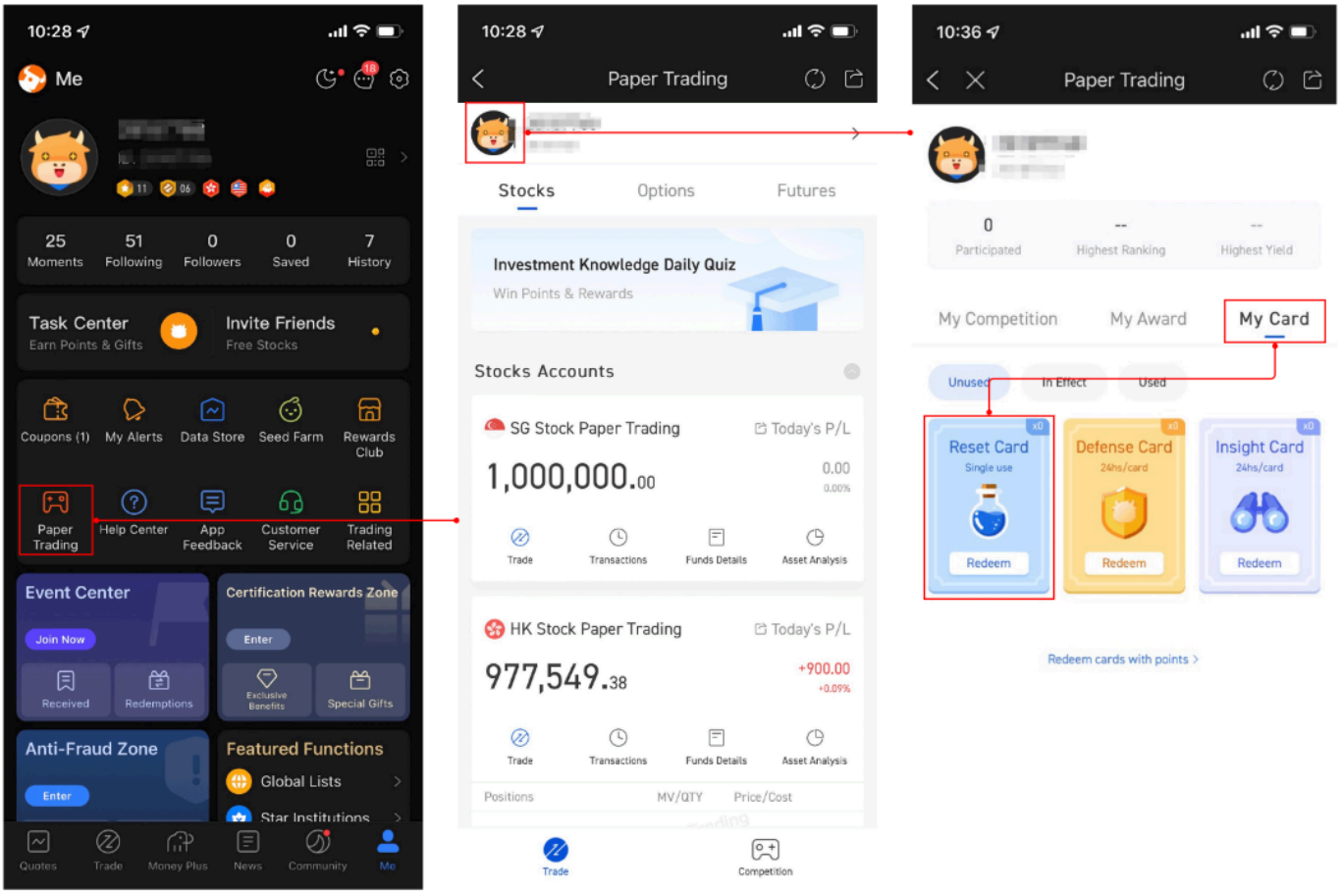
```
    code stock_name    trd_side order_type    order_status  order_id qty  price    create_time   updated_time  dealt_qty
0   HK.00700 Tencent   BUY  ABSOLUTE_LIMIT    CANCELLED_ALL 464200047650696474    100.0    510.0    2021-10-09 11:34:54
```

How to reset the paper trading account?

Currently, OpenAPI does not support resetting the paper trading account. You can use the reset card on the mobile clients. After the reset, net assets would be restored to the initial value and the historical orders would be emptied.

Specific process

Modify clients: Me — Paper Trading — My Icon — My Card — Reset Card

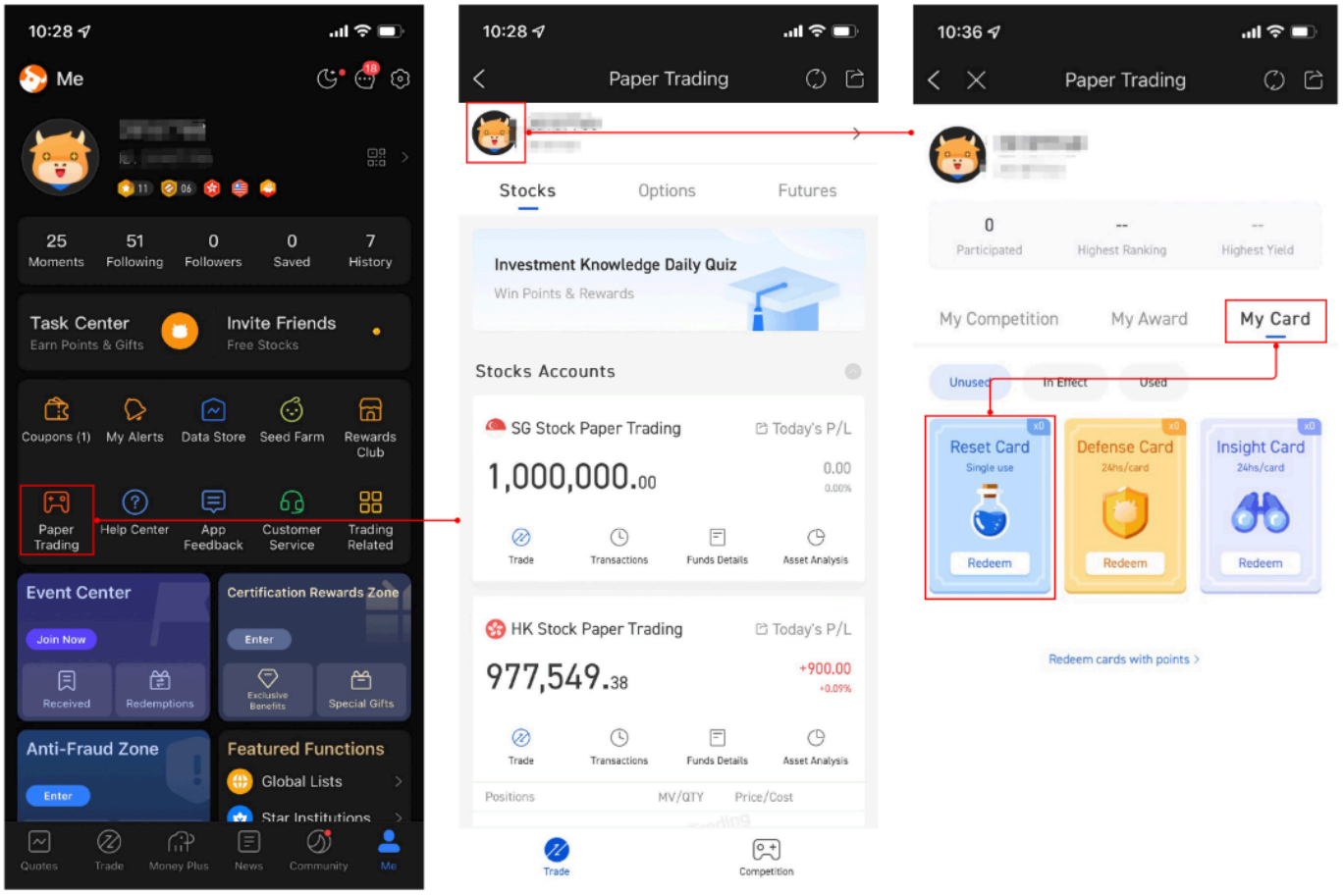


How to reset the paper trading account?

Currently, OpenAPI does not support resetting the paper trading account. You can use the reset card on the mobile clients. After the reset, net assets would be restored to the initial value and the historical orders would be emptied.

Specific process

Modify clients: Me — Paper Trading — My Icon — My Card — Reset Card



Q2: If support A-share trading or not?

A: Paper trading supports A-share trading. However, real trade can only be used to trade some A-shares through A-shares connect. For details, please refer to List of HKCC.

Q3: Trading directions supported by each market

A: Except for futures, other stocks only support the two trading directions of BUY and SELL. In the case of a short position, SELL is passed in, and the direction of the resulting order is short selling.

Q4: Order types supported in each market in real environment

A:

Market	Variety	Limit Orders	Market Orders	At-auction Limit Orders	At-auction Market Orders	Absolute Limit Orders	Special Limit Orders	AON Special Limit Orders	Stop Orders	Stop Limit Orders	Market if Touched Orders	Limit if Touched Orders
HK	Securities (including stocks, ETFs, warrants, CBBCs, Inline Warrants)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Options	✓	X	-	-	-	-	-	X	✓	X	✓

	Futures	✓	✓	-	✓	-	-	-	✓	✓	✓	✓
US	Securities (including stocks, ETFs)	✓	✓	-	-	-	-	-	✓	✓	✓	✓
	Options	✓	✓	-	-	-	-	-	✓	✓	✓	✓
	Futures	✓	✓	-	-	-	-	-	✓	✓	✓	✓
HKCC	Securities (including stocks, ETFs)	✓	X	-	-	-	-	-	X	✓	X	✓
Singapore	Futures	✓	✓	-	-	-	-	-	✓	✓	✓	✓
Japanese	Futures	✓	✓	-	-	-	-	-	✓	✓	✓	✓

Q5: Order operations supported by each market

A:

- HK stocks support order modification, cancellation, entry into force, invalidation, and deletion
- US stocks only support order modification and cancellation
- HKCC only supports cancellation of orders
- Futures supports order modification, cancellation, and deletion

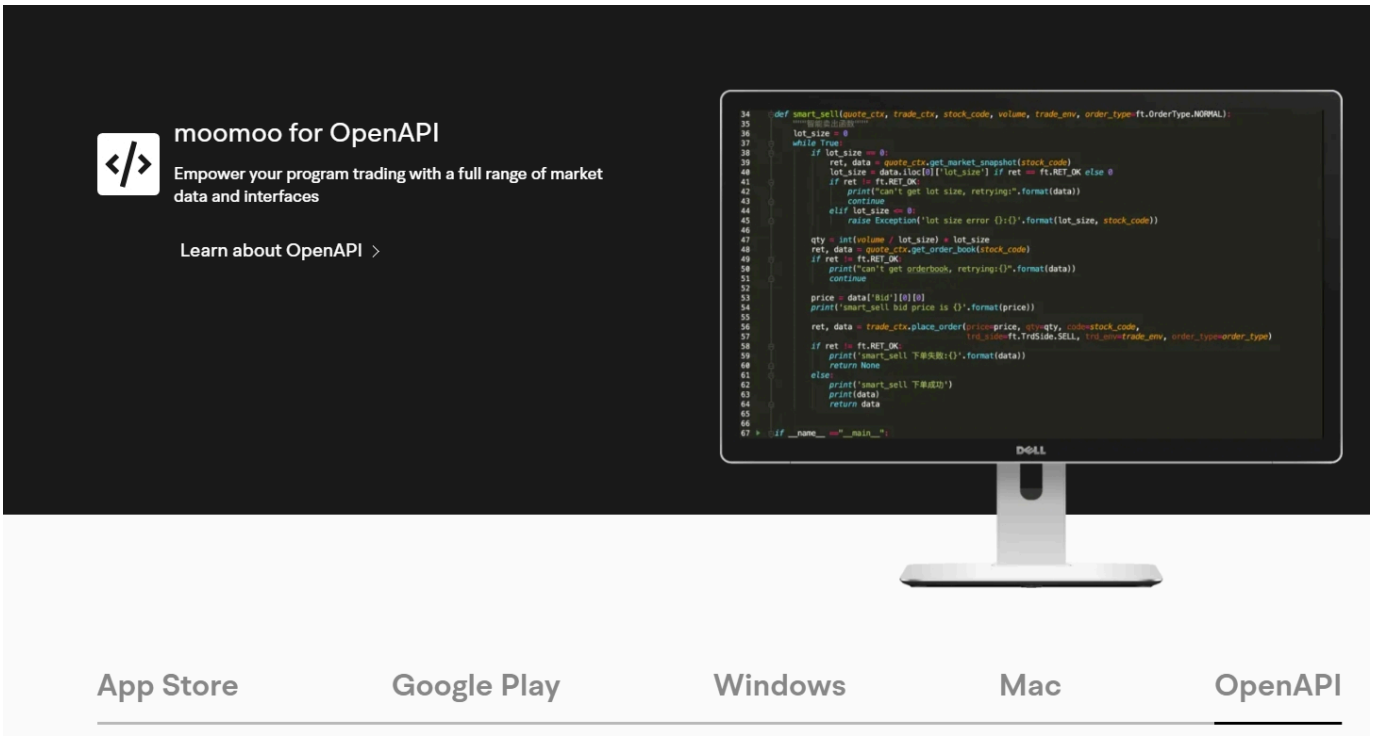
Q6: How to use OpenD startup parameter future_trade_api_time_zone?

A: Since the types of futures supported for trading account are distributed in multiple exchanges around the world, and the time zones of the exchanges are different, the time display of the futures trading API has become a problem. The future_trade_api_time_zone parameter has been added to the OpenD startup parameters, allowing futures traders in different regions of the world to flexibly specify the time zone. The default time zone is UTC+8. If you are more accustomed to Eastern Time, you only need to configure this parameter to UTC-5.

> **Tips**
>
> - This parameter is only valid for futures trading interface objects. The time zone of HK stock trading, US stock trading, and HKCC trading interface objects is still displayed in accordance with the time zone of the exchange.
> - The interfaces affected by this parameter include: responding to order push callbacks, responding to transaction push callbacks, querying today's orders, querying historical orders, querying current transactions, querying historical transactions, and placing orders.

Q7: Can I see the order placed through OpenAPI, in APP?

A：Yes, you can.
After the order is successfully placed through OpenAPI, you can view today's orders, order status change in the trade page of APP, and you can also receive **Order Notice** in the APP.



Q8: Which trading targets support Off-Market order?

A：All orders can only be filled during the market opening period.

Orders made outside market hours and extended hours trading are queued and fulfilled either at or near the beginning of extended hours trading or at or near the market open, according to your instructions. These orders may be named as off market orders or overnight order.

OpenAPI supports Off-Market order for a part of trading targets (APP supports much more trading targets' Off-Market order), as follows:

Market	Contracts	Paper Trading	Live Trading						
			FUTU HK	Moomoo Financial Inc.	Moomoo Financial Singapore Pte. Ltd.	Moomoo AU	Moomoo MY	Moomoo CA	Moomoo JP
HK Market	Securities (including stocks, ETFs, warrants, CBBCs, Inline Warrants)	✓	✓	✓	✓	✓	✓	X	X
	Options	✓	X	X	X	X	X	X	X
	Futures	X	X	X	X	X	X	X	X
US Market	Securities (including stocks, ETFs)	✓	X	X	X	X	✓	✓	✓
	Options	✓	X	X	X	X	✓	✓	✓
	Futures	X	X	X	X	X	✓	X	X

A-share Market	HKCC stocks	✓	X	X	X	X	X	X	X
	Non-HKCC stocks	✓	X	X	X	X	X	X	X
Singapore Market	Futures	X	X	X	X	X	X	X	X
Japanese Market	Securities (including stocks, ETFs, REITS)	X	X	X	X	X	X	X	X
	Futures	✓	✓	X	X	X	X	X	X
Australian Market	Securities (including stocks, ETFs)	X	X	X	X	X	X	X	X
Canadian Market	Securities	X	X	X	X	X	X	X	X

> Tip
>
> - ✓：support Off-Market order
> - X：do not support Off-Market order（or non-tradable）

Q9: For each order type，mandatory parameters of PlaceOrder and broker limits for the single order.

A1: Mandatory parameters of PlaceOrder.

Parameters	Limit Orders	Market Orders	At-auction Limit Orders	At-auction Market Orders	Absolute Limit Orders	Special Limit Orders	AON Special Limit Orders	Stop Orders	Stop Limit Orders	Market if Touched Orders	Limit if Touched Orders	Trailing Stop Orders
price	✓		✓		✓	✓	✓		✓		✓	
qty	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
code	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
trd_side	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
order_type	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
trd_env	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
aux_price								✓	✓	✓	✓	
trail_type												✓
trail_value												✓
trail_spread												

`Python users` should note that, place_order does not set a default value for price. For the five types of orders mentioned above, you still need to pass in price, which can be any value.

A2: The broker sets limits on shares or amounts for single orders of various trading products. Exceeding these limits may result in order failures. See the table below for details.

Broker	Product	Quantity Limit Per Order	Amount Limit Per Order
FUTU HK	China A-Shares	1,000,000 Shares	￥5,000,000
	US Stocks	500,000 Shares	$5,000,000
	Hong Kong Stock Futures or Options	3,000 Contracts	Unlimited
moomoo US	US Stocks	500,000 Shares	$10,000,000
moomoo SG	US Stocks	500,000 Shares	$5,000,000
moomoo AU	US Stocks	Unlimited	Unlimited

Q10: For each order type, when modifying the order, mandatory parameters of ModifyOrder as follows.

Parameters	Limit Orders	Market Orders	At-auction Limit Orders	At-auction Market Orders	Absolute Limit Orders	Special Limit Orders	AON Special Limit Orders	Stop Orders	Stop Limit Orders	Market if Touched Orders	Limit if Touched Orders	Traili Sto Orde
modify_order_op	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
order_id	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
price	✓		✓		✓	✓	✓		✓		✓	
qty	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
trd_env	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
aux_price								✓	✓	✓	✓	
trail_type												✓
trail_value												✓
trail_spread												

`Python users` should note that, modify_order does not set a default value for price. For the five types of orders mentioned above, you still need to pass in price, which can be any value.

Q11: The Trade API returns "The current securities account has not yet agreed to the disclaimer."?

A:
Click the link below to confirm the agreement, and restart OpenD to use trading functions normally.

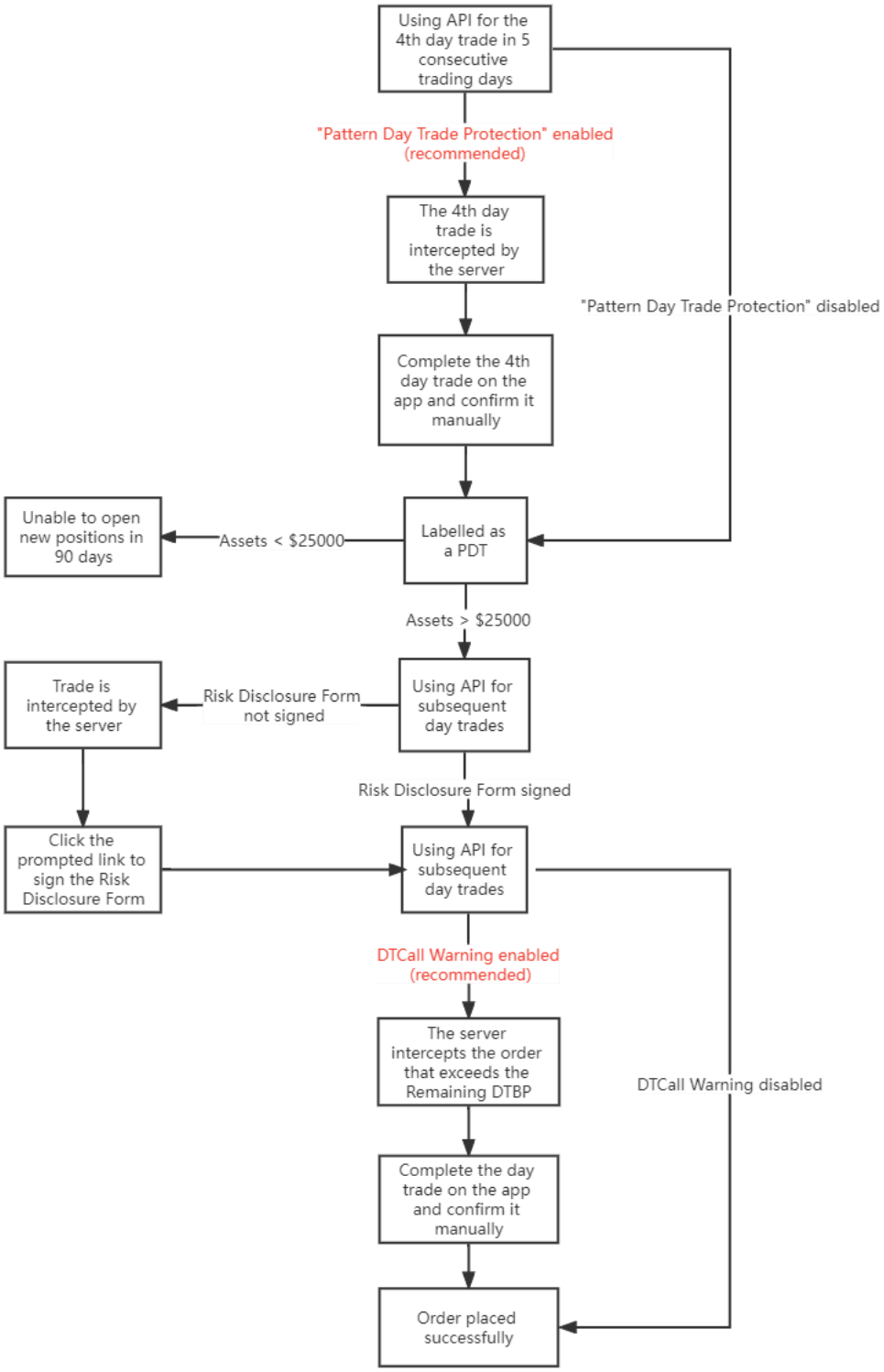
Securities Firm	Aggrement Link
FUTU HK	Click here
Moomoo US	Click here
Moomoo SG	Click here
Moomoo AU	Click here
Moomoo CA	Click here
Moomoo MY	Click here
Moomoo JP	Click here

Q12: Pattern Day Trader (PDT)

Overview

When clients use moomoo US accounts for intraday trading, they are subject to regulations by the US Financial Industry Regulatory Authority (FINRA). This is a regulatory requirement for US brokers and has nothing to do with the market to which a stock being traded belongs. The trading accounts of brokers in other countries or regions, such as moomoo HK and moomoo SG accounts, are not subject to this restriction. If a client conducts over 3 day trades in any 5 consecutive trading days, the client will be labelled as a pattern day trader (PDT).
For more details, refer to **Help Center - Day Trade Rules**.

Day Trading Flowchart



How to turn off "Pattern Day Trade Protection", if I'm willing to be labelled as a PDT and do not want the quant trading program to be interrupted?

A:

To prevent you from being unintentionally labelled as a PDT, the server will automatically intercept your 4th day trade in any 5 consecutive trading days. If you are willing to be labelled as a PDT and do not want the server to intercept your trade, you can take the following step:

Via Command Line OpenD, modify the value of the startup parameter "pdt_protection" to "0".

```
<!-- moomoo US 专用参数 -->
<!-- Specific parameters for moomoo US -->
<!-- 是否开启 防止被标记为日内交易者 的功能，0：否，1：是-->
    <!-- 开启功能后，我们会在您将要被标记 PDT 时阻止您的下单，但不确保您一定不被标记。若您被标记 PDT，当您的账户权益小于$
    <!-- Whether to turn on the Pattern Day Trade Protection, 0: No, 1: Yes  -->
    <!-- When this parameter is set as 1, we will prevent you from placing orders which might mark you as a Pattern Day T
    <pdt_protection>1</pdt_protection>
```

NOTE: You will not be able to establish new positions when you are labelled as a PDT and your account equity is below $25000.

How to turn off the Day-Trading Call Warning?

A:

Once you are labelled as a PDT, you need to pay attention to the day trading buying power (DTBP) of your account. When the DTBP is insufficient, you will receive a DTCall. The server will intercept your order that exceeds the DTBP. If you still want to place the order and do not want the server to intercept it, you can take the following step:

Via Command Line OpenD, modify the value of the startup parameter "dtcall_confirmation" to "0".

```
<!-- 是否开启 日内交易保证金追缴预警 的功能，0：否，1：是 -->
<!-- 开启功能后，我们会在您即将开仓下单超出剩余日内交易购买力前阻止您的下单。提醒您当前开仓订单的市值大于您的剩余日内交易购买力，若您在今日平仓当前标的，
<!-- Whether to turn on the Day-Trading Call Warning, 0: No, 1: Yes  -->
<!-- When this parameter is set as 1, we will prevent you from placing orders which might exceed your remaining day-trading buying power. We will alert y
<dtcall_confirmation>1</dtcall_confirmation>
```

NOTE: If the market value of a newly established position exceeds your remaining DTBP and you close the position in the same day, you will receive a DTCall, which can only be met by depositing funds.

How to check my DTBP?

A:

Via Get Account Funds Interface, you can request values related to day trading, such as Day Trades Left, Beginning DTBP, Remaining DTBP, etc.

Q13: How to track the status of orders?

A:

The two interfaces can be uesed to track the status of orders, after which have been placed.

Trading Enviroment	Interfaces
Real	Orders Push Callback, Deals Push Callback
Simulate	Orders Push Callback

Note: Non-python users need to Subscribe to Transaction Push before using the above two interfaces.

Orders Push Callback:

Feedback changes of the entire order. The order push will be triggered when the following 8 fields change:
`Order status` , `Order price` , `Order quantity` , `Deal quantity` , `Traget price` , `Trailing type` , `Trailing amount/ratio` , `Specify spread`

Therefore, when you place, modify, cancel, enable, or disable the order, or when an advanced order is triggered or an order has transaction changes, it will cause orders push. You just need to call the Orders Push Callback to listen for these messages.

Deals Push Callback:

Feedback changes of a transaction. The order push will be triggered when the following field change:
`Deal status`

Fot example: Suppose a limit order of 900 shares is divided into 3 transactions before it is completely filled, with each transaction being 200, 300 and 400 shares.

Q14: Why does the order interface return "The minimum tick size for this product is xxx. Please enter an integer multiple of the minimum tick size before submitting"?

A:
Different exchanges have different rules on order price spreads. If the price of a submitted order does not follow relevant rules, the order will be rejected.

Rules on Price Spread

Hong Kong Market

Refer to the official HKEX Spread Table

China A-Shares

Stock price spread: 0.01

US Market

Stock Price Spreads:

Price	Spread
Below $1	$0.0001
$1 or above	$0.01

Option Price Spreads:

Price	Spread
$0.10 - $3.00	$0.01 or $0.05
$3.00+	$0.05 or $0.10

Futures Price Spreads:
Different contracts have different price spreads, which can be obtained via the `Price change step` of Get Futures Contract Information interface.

How to ensure an order price meets spread rules?

- Method 1: Valid order prices can be obtained via the Get Real-time Order Book interface, since the prices of orders on the order book must be valid.
- Method 2: Auto-adjust an order price to a valid value via the `Price adjustment range` parameter in the Place Orders interface.

  How it works:

  Suppose the Adjust Limit is set to 0.0015. A positive value means that OpenD will auto-adjust upward the price of a submitted order to a valid value within +0.15% of the original price.

Suppose the current market price of Tencent Holdings is 359.600, so the spread is 0.200 according to the HKEX Spread Table. Let's say an order priced at 359.678 is submitted. In this case, the nearest upward valid price is 359.800, which means the order price only needs to be adjusted by 0.034%. The adjustment satisfies the Adjust Limit, so the final price of the submitted order is 359.800.

If the actual adjustment exceeds the Adjust Limit, OpenD will fail to auto-adjust the price, and the order submission will still return the error prompt "The minimum tick size for this product is xxx. Please enter an integer multiple of the minimum tick size before submitting".

Q15: Why did it say "Insufficient Buying Power" when I place a market order with enough buying power in my account?

A:

Why it indicates insufficient buying power when you place a market order

- For the sake of risk management, the system poses a higher buying power coefficient on market orders. With the same order parameters, a market order takes up more buying power than a limit order.
- Depending on different product types and market conditions, the risk management system dynamically adjusts the buying power coefficient of market orders. Therefore, when placing a market order, if you calculate the maximum buyable quantity using your maximum buying power, you are likely to get an inaccurate result.

How to get the correct buyable quantity

Instead of calculating it, you can obtain the correct buyable quantity through the [Query the Maximum Quantity that Can be Bought or Sold](../trade/get-max-trd-qtys.html) API.

How to buy as much as possible

You can place a limit order at the BBO, instead of a market order. In particular, the BBO means the best bid (or Bid 1) in the case of a sell order, or the best ask (or Ask 1) for a buy order.

Q16: Why can't I see the API papper trading orders on the mobile app?

A:
On all the mobile, desktop and website, the US stock paper trading account has been upgraded from the **US Paper Trading Accounts** to the **US Paper Trading Margin Accounts**.
The OpenAPI has not yet been upgraded (planning phase). At present, only the old US Paper Trading Account is available for use. Please note that this old account cannot be displayed on other clients, so use it with caution.

Q17: Instructions for Using Trade API Parameters

1. What is the Transaction Object?

Under your user ID, there is generally a margin universal account with several sub-accounts (usually two, a univeral securities account and a universal futures accoun; also a universal forex account if needed). Some users or institutional clients may open multiple universal accounts with multiple brokers.
Creating a transaction object is the process of initially screening sub-accounts.

- When calling get_acc_list using OpenSecTradeContext, only trading securities accounts will be returned.
- When calling get_acc_list using OpenFutureTradeContext, only trading futures accounts will be returned.

The `security_firm` is used to filter accounts belonging to the corresponding securities firm, and the `filter_trdmarket` is used to filter accounts with the corresponding trading market permissions.

1.1 security_firm

The brokers currently supported by OpenAPI are as follows.
When calling get_acc_list, it will return the real account of the securities firm corresponding to security_firm and all paper trading accounts (paper trading has no concept of brokers, so no matter what security_firm is passed, all paper trading accounts will be returned).
The default value of security_firm is FUTUSECURITIES. You can leave this parameter blank for FUTU HK accounts, but you need to modify this parameter when you want to obtain accounts from other brokers.

- Example 1

```py
trd_ctx = OpenSecTradeContext(security_firm=SecurityFirm.FUTUSECURITIES)
ret, data = trd_ctx.get_acc_list()
print(data)
```

- Output

```py
              acc_id   trd_env acc_type      uni_card_num          card_num   security_firm sim_acc_type                trdma
0  281756478396547854      REAL   MARGIN  1001200163530138  1001369091153722  FUTUSECURITIES          N/A  [HK, US, HKCC, HKFUN
1             3450309  SIMULATE     CASH               N/A               N/A             N/A        STOCK
2             3548731  SIMULATE   MARGIN               N/A               N/A             N/A       OPTION
3  281756455998014447      REAL   MARGIN               N/A  1001100320482767  FUTUSECURITIES          N/A
```

- Example 2

```py
trd_ctx = OpenSecTradeContext(security_firm=SecurityFirm.FUTUSG)
ret, data = trd_ctx.get_acc_list()
print(data)
```

- Output

```py
    acc_id   trd_env acc_type uni_card_num card_num security_firm sim_acc_type trdmarket_auth acc_status
0  3450309  SIMULATE     CASH          N/A      N/A           N/A        STOCK           [HK]     ACTIVE
1  3548731  SIMULATE   MARGIN          N/A      N/A           N/A       OPTION           [HK]     ACTIVE
```

1.2 filter_trdmarket

The trading markets supported by OpenAPI are as follows.
When calling get_acc_list, it will return all accounts with trading permissions in the filter_trdmarket market; when the filter_trdmarket is passed as NONE, the market will not be filtered and all accounts will be returned.
The default trdmarket is HK. Under the universal account system, this parameter is used to filter paper trading accounts in different markets.

- Example 1

```py
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US)
ret, data = trd_ctx.get_acc_list()
print(data)
```

- Output

```py
              acc_id   trd_env acc_type      uni_card_num          card_num   security_firm sim_acc_type                trdma
0  281756478396547854      REAL   MARGIN  1001200163530138  1001369091153722  FUTUSECURITIES          N/A  [HK, US, HKCC, HKFUN
1             3450310  SIMULATE   MARGIN               N/A               N/A             N/A        STOCK
2             3548732  SIMULATE   MARGIN               N/A               N/A             N/A       OPTION
3  281756460292981743      REAL   MARGIN               N/A  1001100520714263  FUTUSECURITIES          N/A
```

- Example 2

```py
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.NONE)
ret, data = trd_ctx.get_acc_list()
print(data)
```

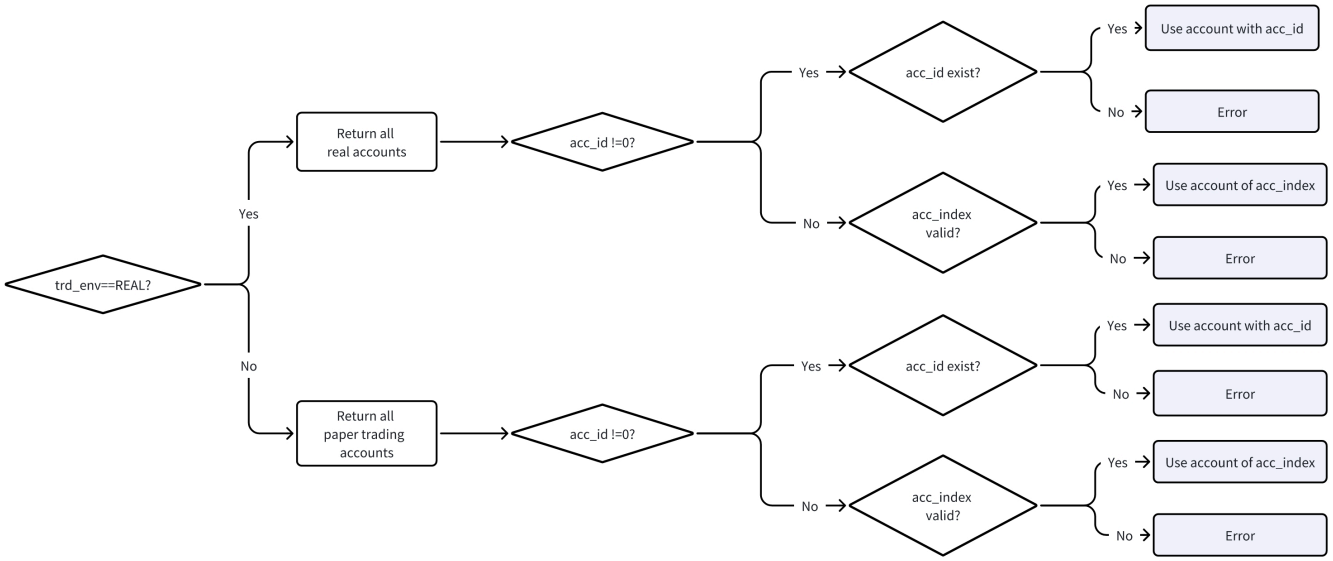
- Output

```py
                acc_id   trd_env acc_type      uni_card_num          card_num   security_firm sim_acc_type                trdm
0   281756478396547854      REAL   MARGIN  1001200163530138  1001369091153722  FUTUSECURITIES          N/A  [HK, US, HKCC, HKFU
1              3450309  SIMULATE     CASH               N/A               N/A             N/A        STOCK
2              3450310  SIMULATE   MARGIN               N/A               N/A             N/A        STOCK
3              3450311  SIMULATE     CASH               N/A               N/A             N/A        STOCK
4              3548732  SIMULATE   MARGIN               N/A               N/A             N/A       OPTION
5              3548731  SIMULATE   MARGIN               N/A               N/A             N/A       OPTION
6   281756455998014447      REAL   MARGIN               N/A  1001100320482767  FUTUSECURITIES          N/A
7   281756460292981743      REAL   MARGIN               N/A  1001100520714263  FUTUSECURITIES          N/A
8   281756468882916335      REAL   MARGIN               N/A  1001100610464507  FUTUSECURITIES          N/A
9   281756507537621999      REAL     CASH               N/A  1001100910390035  FUTUSECURITIES          N/A
10  281756550487294959      REAL     CASH               N/A  1001101010406844  FUTUSECURITIES          N/A
```

> **Tips**
>
> When the filter_trdmarket is passed NONE, all trading accounts will be returned. Row 0 is the active real universal account, rows 1-5 are paper trading accounts, and rows 6-10 are disabled real accounts which are all single-market accounts, that have been replaced by the universal account (row 0). However, historical orders and deals are still in these disabled accounts, and you can query them via these accounts.
> There is no filter_trdmarket in the OpenFutureTradeContext, but security_firm, which has the same function as that in OpenSecTradeContext.

2. Trade API Parameters

When using specific trading API (such as place orders, get open orders), the `trd_env`, `acc_index` and `acc_id` parameters will first filter and confirm a unique account, and then implement the corresponding interface function for this account.



> **Summary**
>
> 1. Filter out real or paper trading accounts according to trd_env.
> 2. Among the results, the account specified by acc_id is prioritized.
> 3. If acc_id is 0, select the corresponding account through acc_index.
> 4. Error: The specified acc_id does not exist, or the acc_index is out of range.

3. Examples

3.1 Place Orders through Universal securities accounts

```py
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.NONE, security_firm=SecurityFirm.FUTUSECURITIES)
ret, data = trd_ctx.unlock_trade("123123")
if ret == RET_OK:
    print("unlock success!")
    ret, data = trd_ctx.place_order(45, 200, 'HK.00700', TrdSide.BUY,
                                    order_type=OrderType.NORMAL,
                                    trd_env=TrdEnv.REAL,
                                    acc_id=0)
    print(data)
```

3.2 Get Open Orders through Universal futures accounts

```py
trd_ctx = OpenFutureTradeContext(security_firm=SecurityFirm.FUTUSECURITIES)

ret, data = trd_ctx.order_list_query(trd_env=TrdEnv.REAL,
                                     acc_id=0)
print(data)
```

3.3 Get Account Funds through HK Cash Account (Paper Trading)

```py
# filter_trdmarket: TrdMarket.HK
# trd_env: TrdEnv.SIMULATE
# acc_index: 0
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK)
ret, data = trd_ctx.accinfo_query(trd_env=TrdEnv.SIMULATE, acc_index=0)
print(data)
```

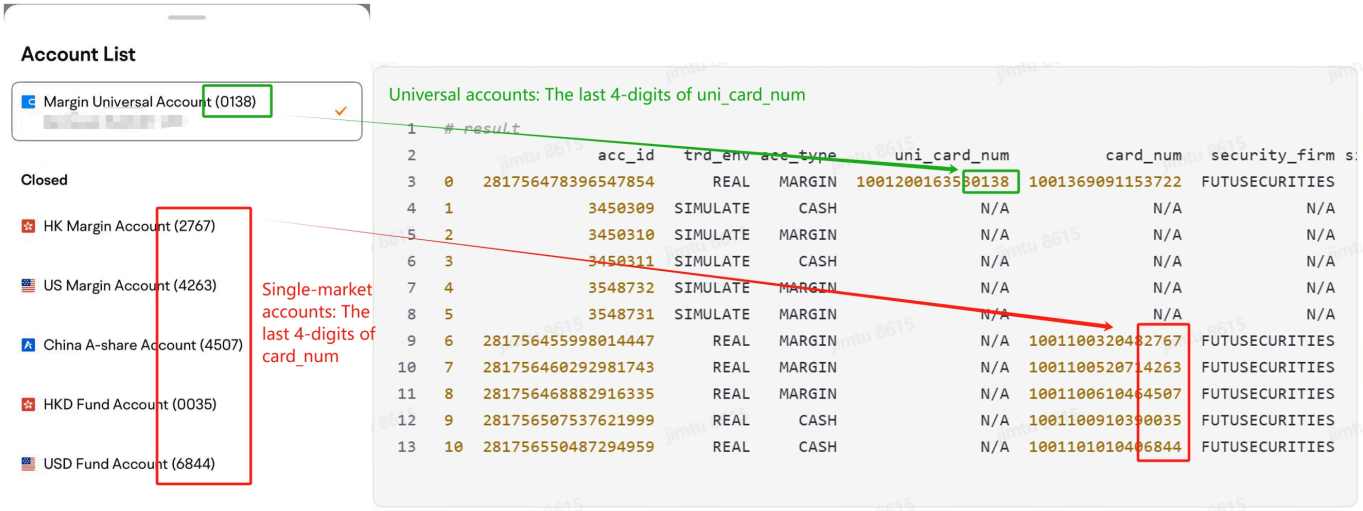
3.4 Trade Options through US Margin Account (Paper Trading)

```py
# Only two accounts returned after filtering by filter_trdmarket and trd_env
# acc_index = 0: US Cash Account (Trading stocks)
# acc_index = 1: US Margin Account (Trading options)
# acc_index: 1
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US)
ret, data = trd_ctx.place_order(10, 1, code="US.AAPL250618P550000",trd_side=TrdSide.BUY,
                                trd_env=TrdEnv.SIMULATE,
                                acc_index=1)
print(data)
```

3.5 Query the Max Quantity that can be Bought or Sold through JP Futures Paper Trading

```py
# Print the outcome of get_acc_list, the acc_id of JP Futures Paper Trading is 6271199
# Pass this acc_id when querying the max quantity that can be bought/sold
trd_ctx = OpenFutureTradeContext()
ret, data = trd_ctx.acctradinginfo_query(order_type=OrderType.NORMAL,
                                         price=5000,
                                         trd_env=TrdEnv.SIMULATE,
                                         acc_id=6271199,
                                         code="JP.NK225main")

print(data)
```

4. How to map the accounts in OpenAPI to those in the APP?



The accounts on the APP only show the last 4-digits of the card number.

According to the result of get_acc_list, the columns uni_card_num and card_num, are corresponding to the card number of Universal account and Single-market account (disabled), respectively.

The account obtained in the API can be matched with that on the APP through the last 4 digits of the card number.

Others

Q1：How to build C++ API?

A: moomoo-api c++ SDK is supported on Windows/MacOS/Linux. Pre-built libs are provided for the common build environment on each platform:

OS	Building Environment
Windows	Visual Studio 2013
Centos 7	g++ 4.8.5
Ubuntu 16.04	g++ 5.4.0
MacOS	XCode 11

If different compiler version is used, or different protobuf version is used, MMAPI and protobuf may be re-built. MMAPI source directory layout is:

```
MMAPI directory structure:
+---Bin                                 Libs for common build environment
+---Include                             Public headers, source files generated fro
+---Sample                              Sample project
\---Src
    +---MMAPI                           MMAPI source
    +---protobuf-all-3.5.1.tar.gz       protobuf source
```

Build steps：

1. Build protobuf to generate libprotobuf static lib and protoc executable.
2. Generated C++ source files from proto files.
3. Build MMAPI to generate libMMAPI static lib

Step1: Build protobuf：

- Windows：

    - Install CMake
    - Open Visual Studio command prompt, change directory to protobuf/cmake
    - Run：cmake -G "Visual Studio 12 2013" -DCMAKE_INSTALL_PREFIX=install -Dprotobuf_BUILD_TESTS=OFF This will generate Visual Studio 2013 solution file. Change -G parameter for other Visual Studio versions.
    - Open Visual Studio solution file, set platform toolset to v120_xp, then build.
- Linux (Refer to protobuf/src/README)
    - Run ./autogen.sh
    - Run CXXFLAGS="-std=gnu++11" ./configure --disable-shared
    - Run make
    - Put generated libprotobuf.a in Bin/Linux
- MacOS (Refer to protobuf/src/README)
    - Install dependencies with brew：autoconf automake libtool
    - Run ./configure CC=clang CXX="clang++ -std=gnu++11 -stdlib=libc++" --disable-shared

Step2: Generate C++ sources from proto files

- Use protoc to convert protofiles under Include/Proto to C++ source files. For example, the following command converts Common.proto to Common.pb.h and Common.pb.cc:
    - protoc -I="path-to-MMAPI/Include/Proto" --cpp_out="." path-to-MMAPI/Include/Proto/Common.proto
- Put the generated .h and .cc files in Include/Proto

Step3: Build MMAPI

- Windows：Create Visual Studio C++ static lib project，add source files under Src/MMAPI and Include，and set platform toolset to v120_xp.
- Mac：Create XCode C++ static lib project，add source files under Src/MMAPI and Include
- Linux：Use cmake to build MMAPI static lib, run following command under path-to-MMAPI/Src:
    - cmake -DTARGET_OS=Linux

Q2: Is there more complete strategy examples for reference?

A:

- Python strategy examples are in the /moomoo/examples/ folder. You can find the path of Python API by executing the following command:

```
import moomoo
print(moomoo.__file__)
```

- The C# strategy examples are in the /MMAPI4NET/Sample/ folder
- The Java strategy examples are in the /MMAPI4J/sample/ folder
- The C++ strategy examples are under the /MMAPI4CPP/Sample/ folder
- The JavaScript strategy examples are in the /MMAPI4JS/sample/ folder

Q3: Import error when using python API

First Scene:
I have already installed moomoo-api, but still get error: No module named 'moomoo'?
It is possible that the interpreter your IDE currently uses is not the interpreter of the moomoo-api module you installed. In other words, your may have more than two Python environments installed on your computer. You can do the following 2 steps:

1. Run the codes below to get the path of the current interpreter:

```
import sys
print(sys.executable)
```

Example diagram:

```
In[3]: import sys
  ...: print(sys.executable)
D:\software\anaconda3\python.exe
```

2. Run `$ D:\software\anaconda3\python.exe -m pip install moomoo-api` in command line (The first half of the command comes from the result of step 1). This will install a moomoo-api module in the current interpreter.

Q4: Import successful, but you still cannot call the relevant interface.

A: Usually in this case, you need to check if the ‘moomoo’ that was successfully imported is a correct moomoo API.

First Scene: There may be a file with the same name as 'moomoo'.

1. The current file name is moomoo.py
2. There is another file named moomoo.py under the path of the current file.
3. There is a folder named `/moomoo` under the path of the current file.

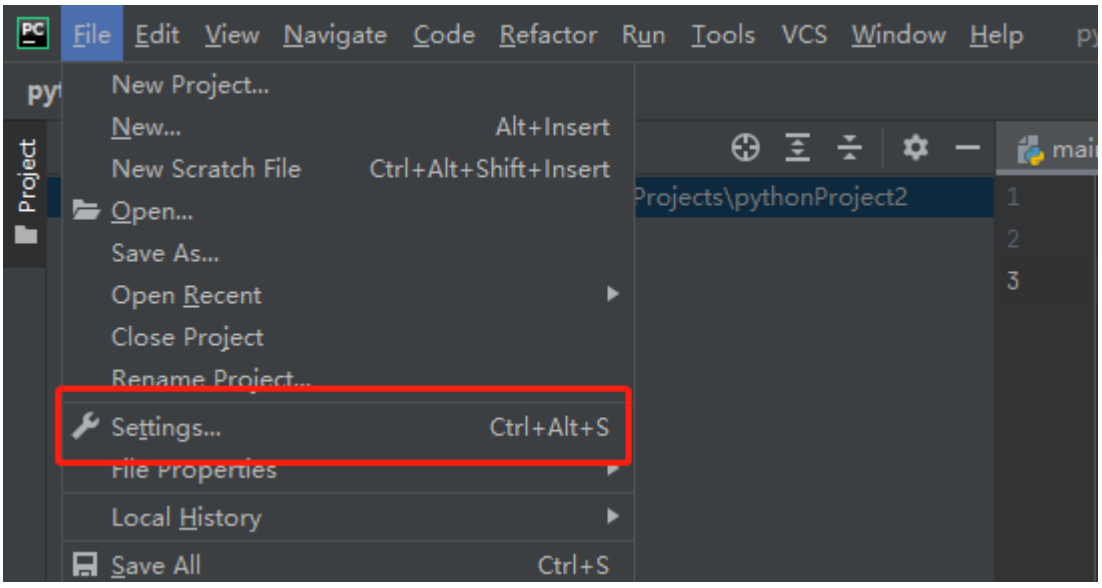
Therefore, we strongly recommend that you do not name files / folders / projects as *moomoo*.

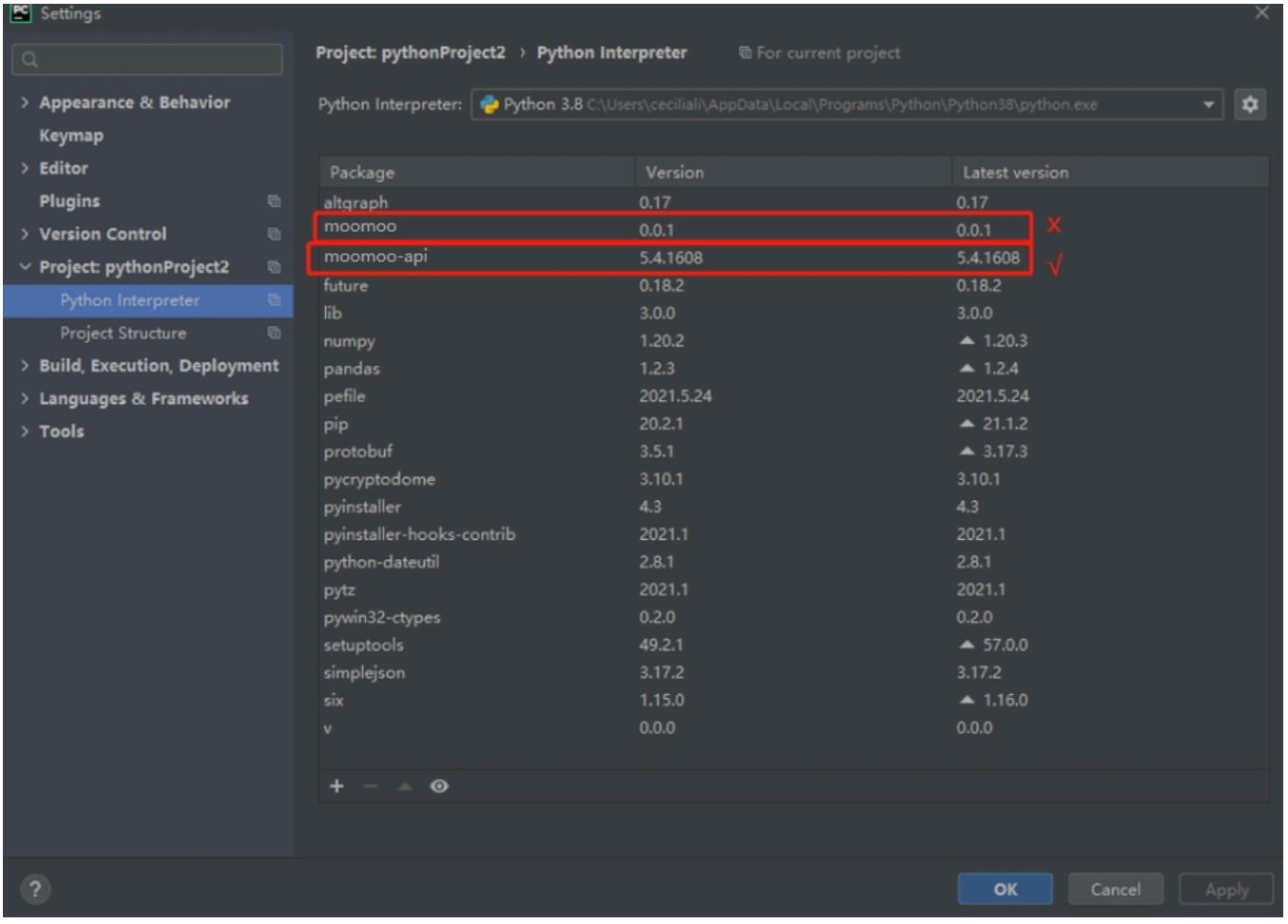
Second Scene: A third-party library called 'moomoo' was installed by mistake.

The correct name of the moomoo API library is `moomoo-api`, not 'moomoo'.

If you have installed a third-party library named 'moomoo', please uninstall it and **install moomoo-api**.

Take PyCharm as an example: Check the installation of libraries.





Q5: Protocol Encryption-Related

A:

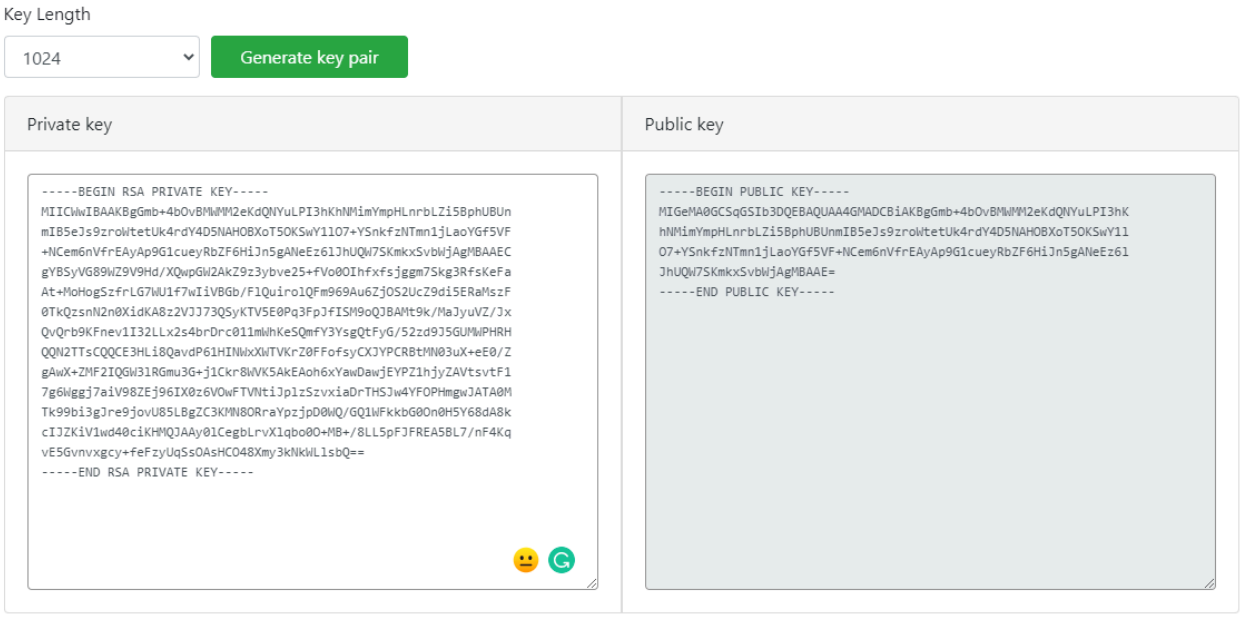
Overview

To ensure privacy and confidentiality, you can use the asymmetric encryption algorithm RSA to encrypt the request and return between Strategy Scripts (moomoo API) and OpenD.
If Strategy Scripts (moomoo API) is on the same computer as OpenD, it is usually not necessary to encrypt.

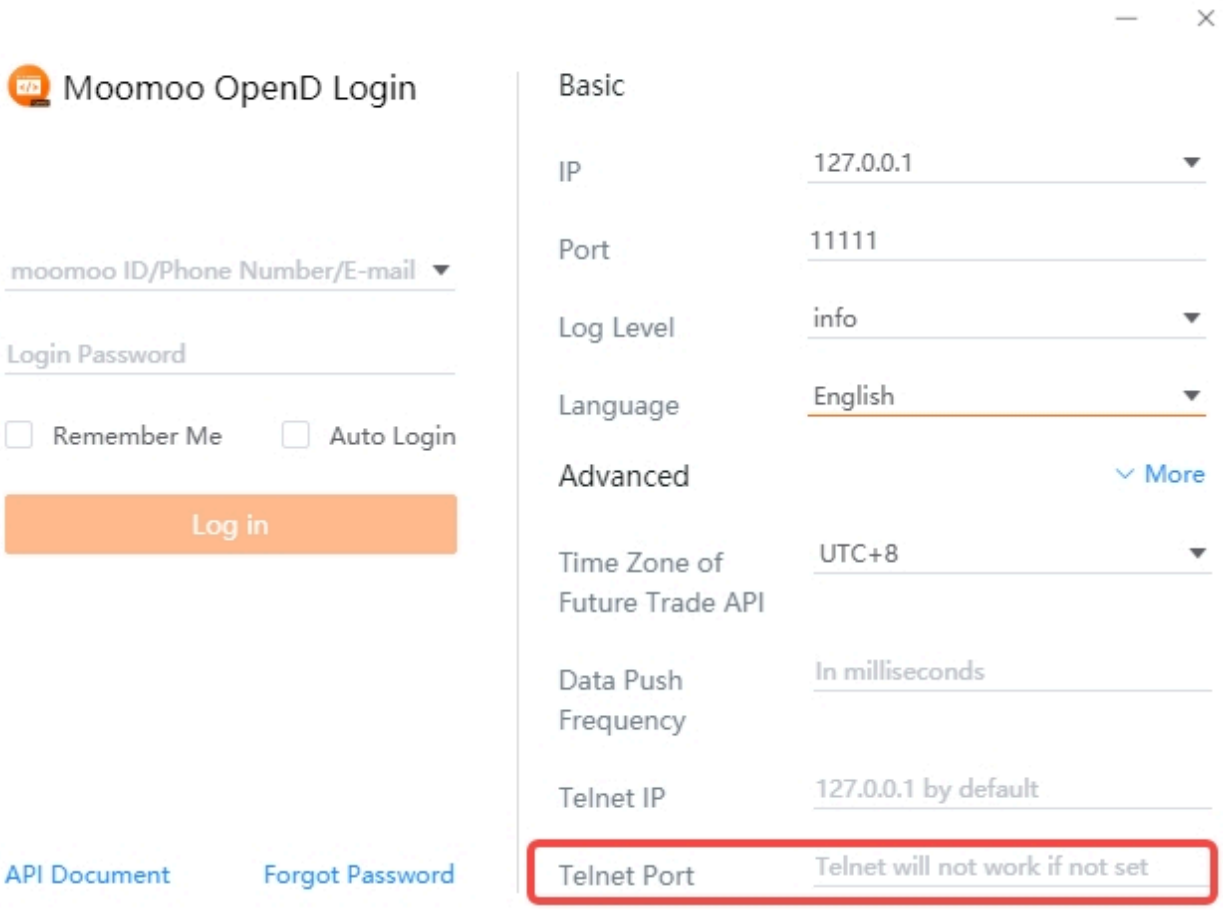
Protocol Encryption Process

You can try to solve this problem with the following steps:

1. Generate the key file automatically through a third-party web platform.
   - To be specific: Search 'Online RSA Key Generator' on Baidu or Google. Set Key Format as PKCS#1. Set Key Length as 1024 bit. No password required. Then click the bottom 'Generate key pair'



2. Copy and paste the private key into a text file. Save it to a specified path of the computer which OpenD is located in.
3. Specify the path of the RSA private key file on the computer which OpenD is located in. The path is the specified path mentioned in Step 2.
   - Method 1: Specify the path mentioned in Step 2 through 'Encryption Private Key' in Visualization OpenD. As shown below:



- Method 2: Specify the path mentioned in Step 2 through the code `rsa_private_key` in **Command Line OpenD**. As shown below:

```
<moomoo_opend>
    <!-- 基础参数 -->
    <!-- Basic parameters -->
        <!-- 协议监听地址,不填默认127.0.0.1 -->
        <!-- Listening address. 127.0.0.1 by default -->
        <ip>127.0.0.1</ip>
        <!-- API接口协议监听端口 -->
        <!-- API interface protocol listening port -->
        <api_port>11111</api_port>
        <!-- 登录帐号 -->
        <!-- Login account -->
        <login_account>100000</login_account>
        <!-- 登录密码32位MD5加密16进制 -->
        <!-- Login password, 32-bit MD5 encrypted hexadecimal -->
        <!-- <login_pwd_md5>6e55f158a827b1a1c4321a245aaaad88</login_pwd_md5> -->
        <!-- 登录密码明文，密码密文存在情况下只使用密文 -->
        <!-- Plain text of login password. When cypher text exists, the cypher text will be used. -->
        <login_pwd>123456</login_pwd>
        <!-- FutuOpenD语言，en: 英文，chs: 简体中文 -->
        <!-- FutuOpenD language. en: English, chs: Simplified Chinese -->
        <lang>chs</lang>
    <!-- 进阶参数 -->
    <!-- Advanced parameters -->
        <!-- FutuOpenD日志等级，no, debug, info, warning, error, fatal -->
        <!-- FutuOpenD log level: no, debug, info, warning, error, fatal -->
        <log_level>info</log_level>
        <!-- API推送协议格式，0: pb, 1: json -->
        <!-- API push protocol format. 0: pb, 1: json -->
        <push_proto_type>0</push_proto_type>
        <!-- API订阅数据推送频率控制，单位毫秒，目前不包括K线和分时，不设置则不限制频率-->
        <!-- Data Push Frequency, in milliseconds. Candlesticks and timeframes are not included. If not se
        <!-- <qot_push_frequency>1000</qot_push_frequency> -->
        <!-- Telnet监听地址,不填默认127.0.0.1 -->
        <!-- Telnet listening address. 127.0.0.1 by default -->
        <!-- <telnet_ip>127.0.0.1</telnet_ip> -->
        <!-- Telnet监听端口 -->
        <!-- Telnet listening port -->
        <!-- <telnet_port>22222</telnet_port> -->
        <!-- API协议加密私钥文件路径,不设置则不加密 -->
        <!-- File path for private key for API protocol enctyption. If not set, it will not be encrypted.
        <!-- <rsa_private_key>D:\rsa</rsa_private_key> -->
        <!-- 是否接收到价提醒推送，0: 不接收，1: 接收 -->
```

4. Save the text file in step 2 to a specified path of the computer which Strategy Scripts (moomoo API) are located in, and **set the path of private key** in Strategy Scripts.
5. Enable protocol encryption. There are two ways to enable protocol encryption.
   - Method 1: Encrypt the context independently (general). You can set encryption through the parameter `is_encrypt` when creating and initializing the connection in **Quote Object** or **Transaction Objects**.
   - Method 2: Encrypt the context globally (only Python). You can set encryption through the interface **enable_proto_encrypt**.

> **Tips**
>
> - When specifying the path of RSA private key in OpenD or in Strategy Scripts (moomoo API), the path needs to be complete and include the file name.
> - It is not necessary to save RSA public key which can be calculated by private key.

Q6: Why is the DataFrame data I got incomplete?

A: When printing pandas.DataFrame data, if there are too many columns and rows, pandas will collapse the data by default, resulting in an incomplete display.

Therefore, it is not OpenD's fault. You can add the following code in front of your Python script to solve the problem.

```
import pandas as pd
pd.options.display.max_rows=5000
pd.options.display.max_columns=5000
pd.options.display.width=1000
```

Q7: How to solve the problem that "Cannot open libFTAPIChannel.dylib" through C++ API on Mac?

A: Execute the following command in the directory where the file "libFTAPIChannel.dylib" is stored: `$ xattr -r -d com.apple.quarantine libAPIChannel.dylib` .

Q8: For Python users, why do large log files continue to be generated under the log folder, after the log level is set to no in the OpenD configuration file?

A： The *log_level* parameter in OpenD parameter configuration is only used to control the logs generated by OpenD. Python API also generates logs by default.
If you do not like it, you can add the following codes to your Python script:

```
logger.file_level = logging.FATAL  # Used to stop Python API log files generating
logger.console_level = logging.FATAL  # Used to stop printing Python log in runni
```

Q9: For versions 5.4 and above, the library name and configuration method of Java API have been changed.

A:

- If you are a user of Java API 5.3 and below, please note the following changes when updating the version.

Changes to the configuration process:

1. Download moomoo API from moomoo official website.
2. Decompress the downloaded file. `/MMAPI4J` is the directory of Java API. Add `/lib/moomoo-api-.x.y.z.jar` file to your project settings. To establish a moomoo-api project, please refer to here.

Changes to the directory:

1. For the Java version of moomoo API, the library name is changed from ftapi4j.jar to `moomoo-api-x.y.z.jar`, where "x.y.z" represents the version number.
2. For the third-party library, the dependencies of /lib/jna.jar and /lib/jna-platform.jar are removed, and the dependencies of `/lib/bcprov-jdk15on-1.68.jar` and `/lib/bcpkix-jdk15on-1.68.jar` are added.

```
+---mmapi4j                    MMAPI4J source code. If the JDK version used
+---lib                        The folder with common libraries
|    moomoo-api-x.y.z.jar        Java version of moomoo API
|    bcprov-jdk15on-1.68.jar   Third-party library, for encryption and decry
|    bcpkix-jdk15on-1.68.jar   Third-party library, for encryption and decry
|    protobuf-java-3.5.1.jar   Third-party library, for parsing protobuf dat
+---sample                     Sample project
+---resources                  The default generated directory of the maven
```

- If you are a new user to the moomoo API, we provide a more convenient way to configure Java API via maven repository for you. About the configuration process, please refer to here.

Q10: For Python users, when using pyinstaller to package scripts that need to run api, an error is reported: Common_pb2 module cannot be found.

A: You can try to solve this problem with the following steps.
Step 1. Suppose you need to package main.py. Using a command-line statement and run the statement: pyinstaller path\main.py, without the "- F" parameter.

```
pyinstaller path\main.py
```

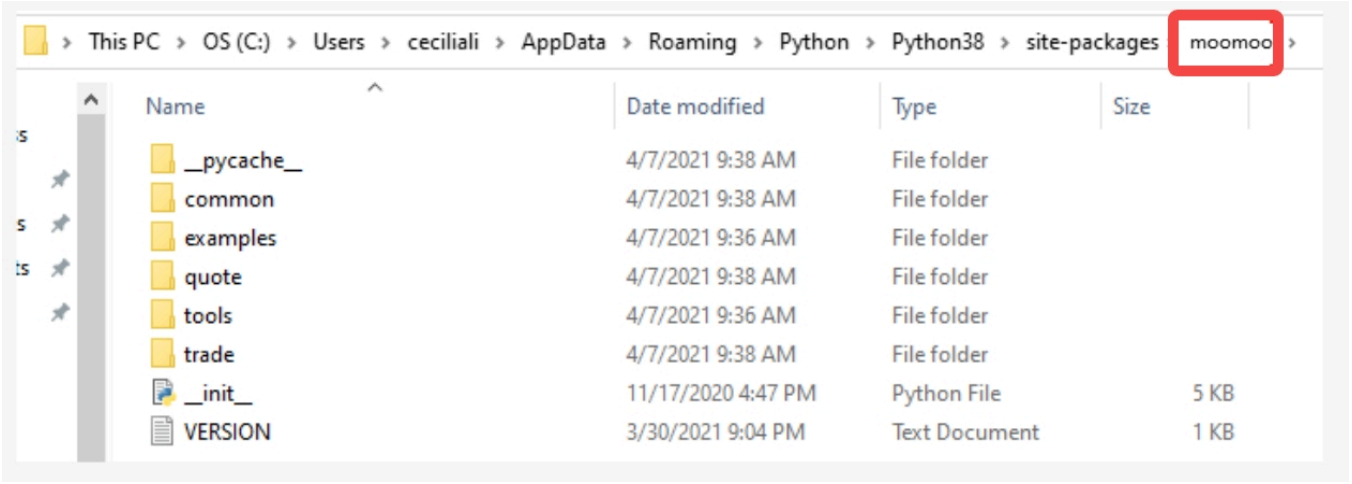
After main.py is packaged, the /main folder will be created in the /dist directory where it is located. main.exe is in this folder.

Step 2. Run the following code to find the installation path of moomoo-api: /path/moomoo.

```
import moomoo
print(moomoo.__file__)
```

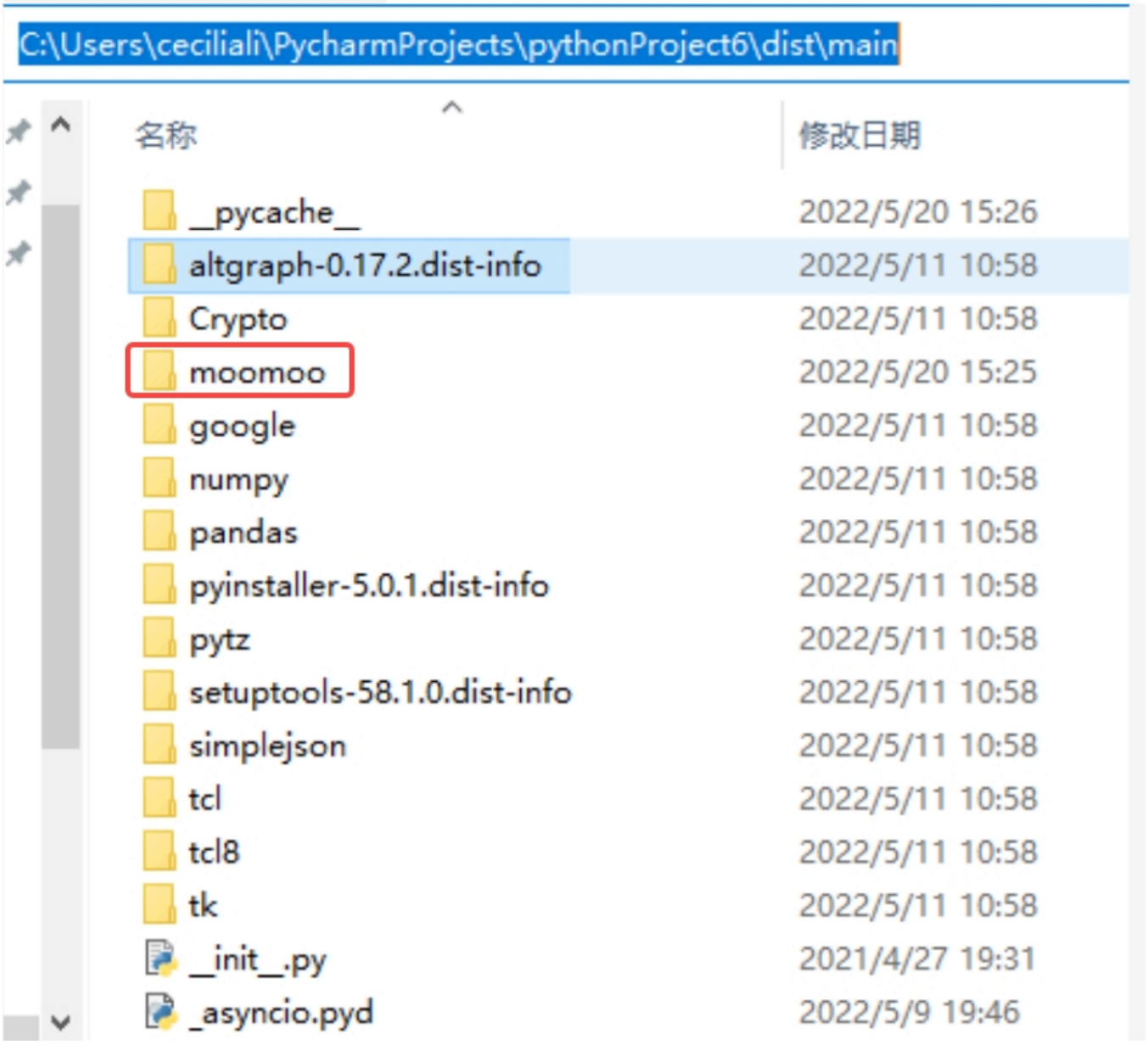
Results:

```
C:\Users\ceciliali\Anaconda3\lib\site-packages\moomoo\__init__.py
```



Step 3. Copy all the files in the /common/pb to /main.

Step 4. Create a folder in the /main and name it moomoo. Copy the `/path/moomoo/VERSION.txt` file to /main/moomoo.



Step 5. Try running the statement **pyinstaller main.py** again.

Q11: Why the interface result is success, but the return did not behave as expected?

A:

- A successful interface result means that server has successfully received and responded to your request, but the return may not behave as your expected.

  Example: If you call the subscribe during non-trading hours, your request can be responded successfully, but the exchange will not update the ticker data during this period. So you will temporarily not receive real-time data until trading hours.

- The interface result (definition: **Interface Result**) can be viewed from the field returned. A field of 0 means the interface result success, a non-zero means the interface result failed.

  For python user, the following two code statements are equivalent:

```
if ret_code == RET_OK:
```

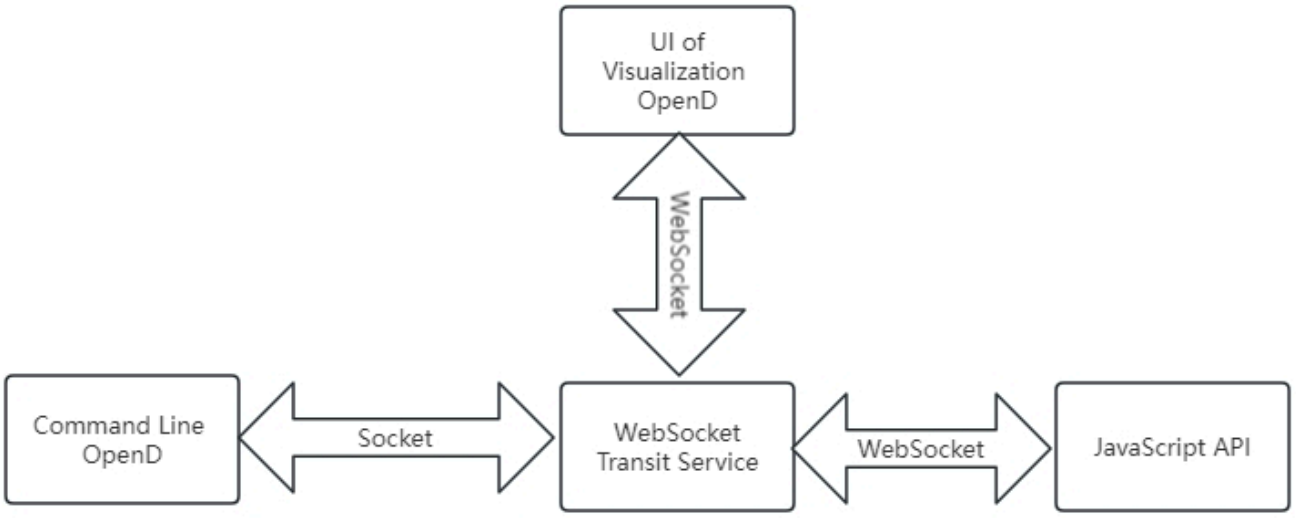
```
if ret_code == 0:
```

Q12: WebSocket Related

Overview

In OpenAPI, WebSocket is mainly used in the following two aspects:

- In Visualization OpenD, WebSocket is used to communicate between the UI interface and the underlying Command Line OpenD.
- The communication between JavaScript API and OpenD uses WebSocket.



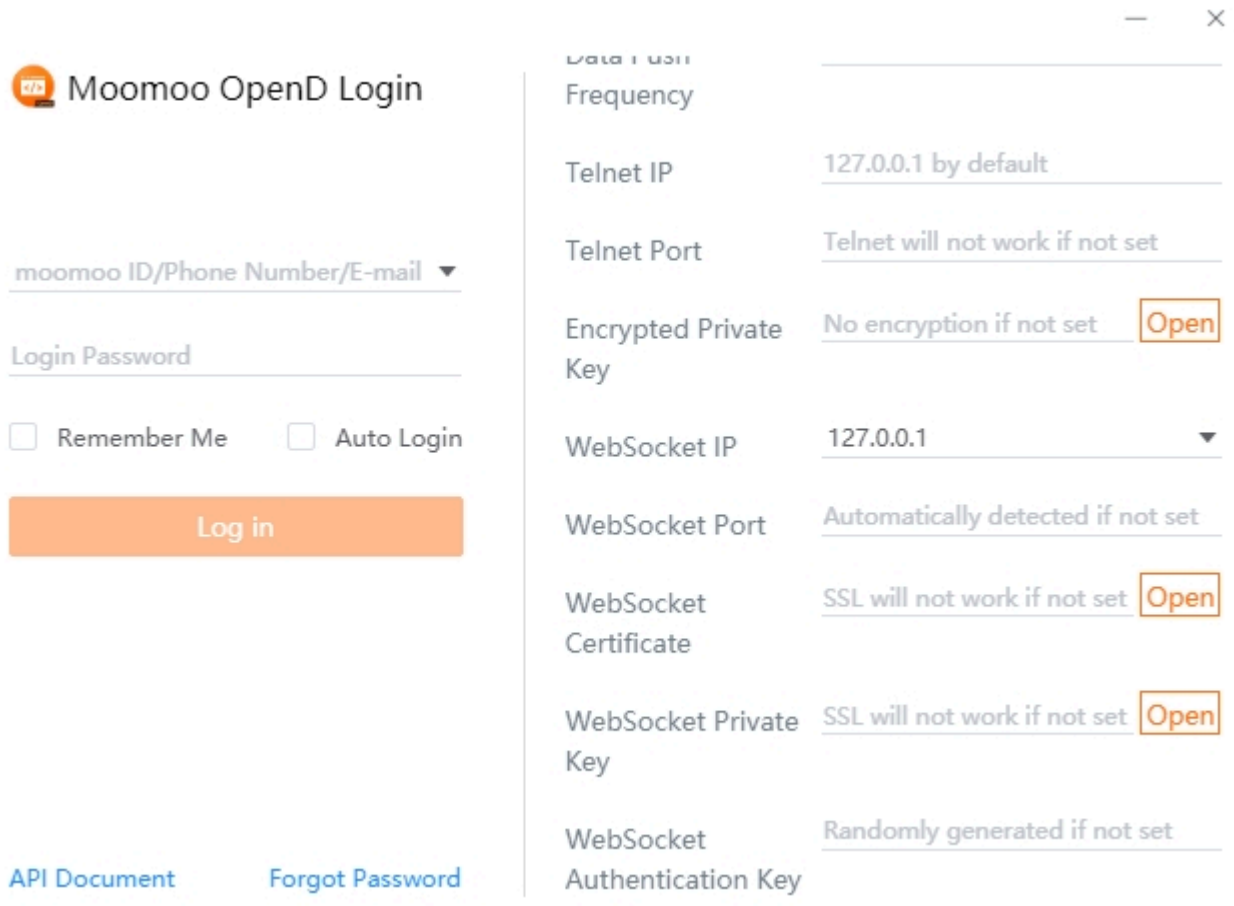
- When WebSocket starts, Command Line OpenD establishes a Socket connection (TCP) with the **MMWebSocket transit service**. This connection uses the default **listening address** and **API protocol listening port**.

- At the same time, JavaScript API will establish a WebSocket connection (HTTP) with the **MMWebSocket transit service**. This connection will use the **WebSocket listening address** and **WebSocket port**.

Usage

To ensure the security of your account, when WebSocket listens non-local requests, we strongly recommend that you enable SSL and configure the **WebSocket authentication key**

SSL is enabled by configuring the **WebSocket certificate** and the **WebSocket private key**. Command Line OpenD can set the file path by configuring OpenD.xml or configuring command line parameters. Visualization OpenD clicks the "more" drop-down menu to see the confifuration item.



> **Tips**
>
> If the certificate is self-signed, you need to install the certificate on the machine where the JavaScript API is called, or set not to verify the certificate.

Generate Self-signed Certificate

It is not convenient to expand the details of self-signed certificate generation in this document, please check it yourself. Simple and available build steps are provided here:

1. Install openssl.
2. Modify openssl.cnf and add the IP address or domain name under the alt_names node on the machine where OpenD locates.
   For example: IP.2 = xxx.xxx.xxx.xxx, DNS.2 = www.xxx.com
3. Generate private key and certificate (PEM)。

The certificate generation parameters are as follows：

```
openssl req -x509 -newkey rsa:2048 -out moomoo.cer -outform PEM -keyout moomoo.key -days 10000 -verbose -config openssl.cnf -nodes -sha256 -subj "/CN=moomoo CA" -reqexts v3_req -extensions v3_req
```

> **Tips**
>
> - openssl.cnf needs to be placed under the system path, or an absolute path needs to be specified in the build parameters.
> - Note that while generating a private key, you need to specify that the password is not set (-notes).

Attach the local self-signed certificate and the configuration file that generates the certificate for testing:

- openssl.cnf
- moomoo.cer
- moomoo.key

Q13: Where are the quote servers and the trade servers of OpenAPI?

A：

- Quote:

Futu ID	Quote Server Location
Futubull ID	Tencent Cloud Guangzhou and Hong Kong

Futu ID	Quote Server Location
moomoo ID	Tencent Cloud Virginia, USA and Singapore

- Trade:

Securities Firm	Trade Server Location
FUTU HK	Tencent Cloud Hong Kong
Moomoo US	Tencent Cloud Virginia, USA
Moomoo SG	Tencent Cloud Singapore
Moomoo AU	Tencent Cloud Singapore
Moomoo MY	Ali Cloud Malaysia
Moomoo CA	AWS Cloud Canada
Moomoo JP	Tencent Cloud Japan