OpenAPI 介紹

概述

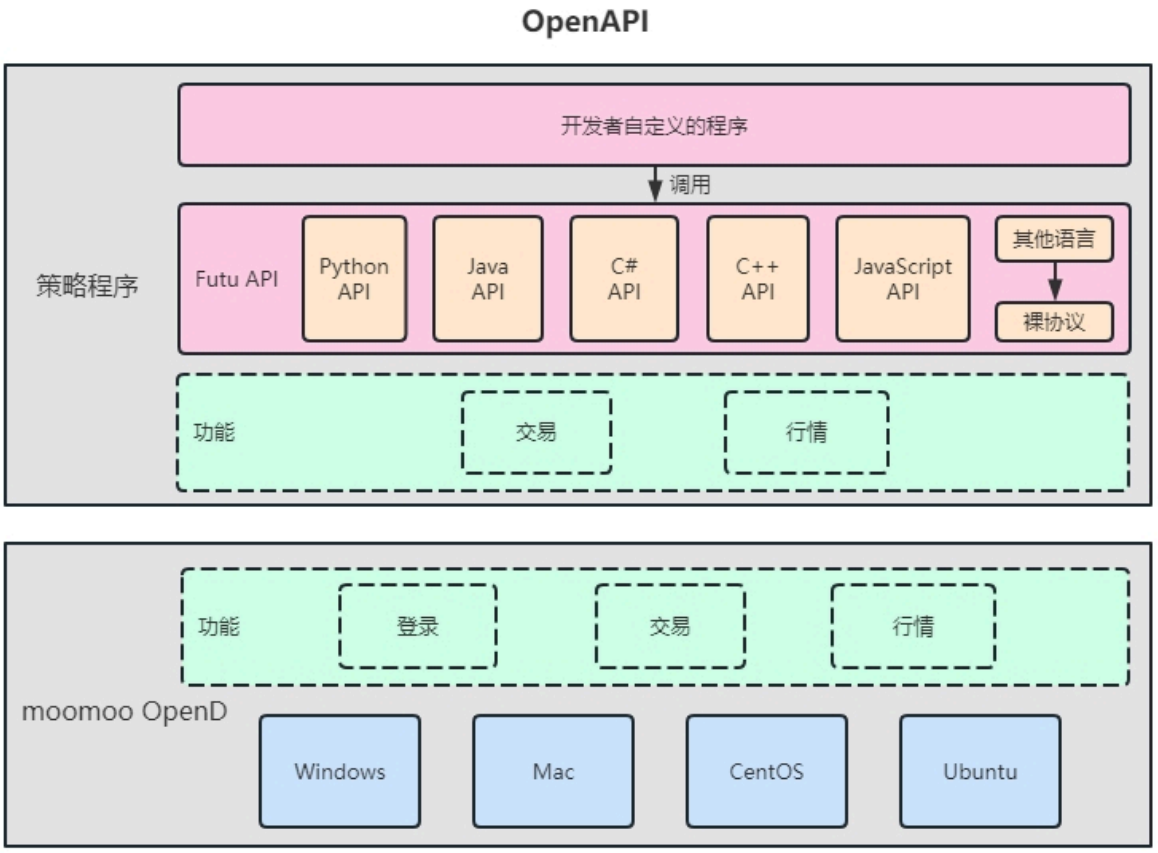
OpenAPI 量化介面，為您的程式化交易，提供豐富的行情和交易介面，滿足每一位開發者的量化投資需求，助力您的量化交易夢想。

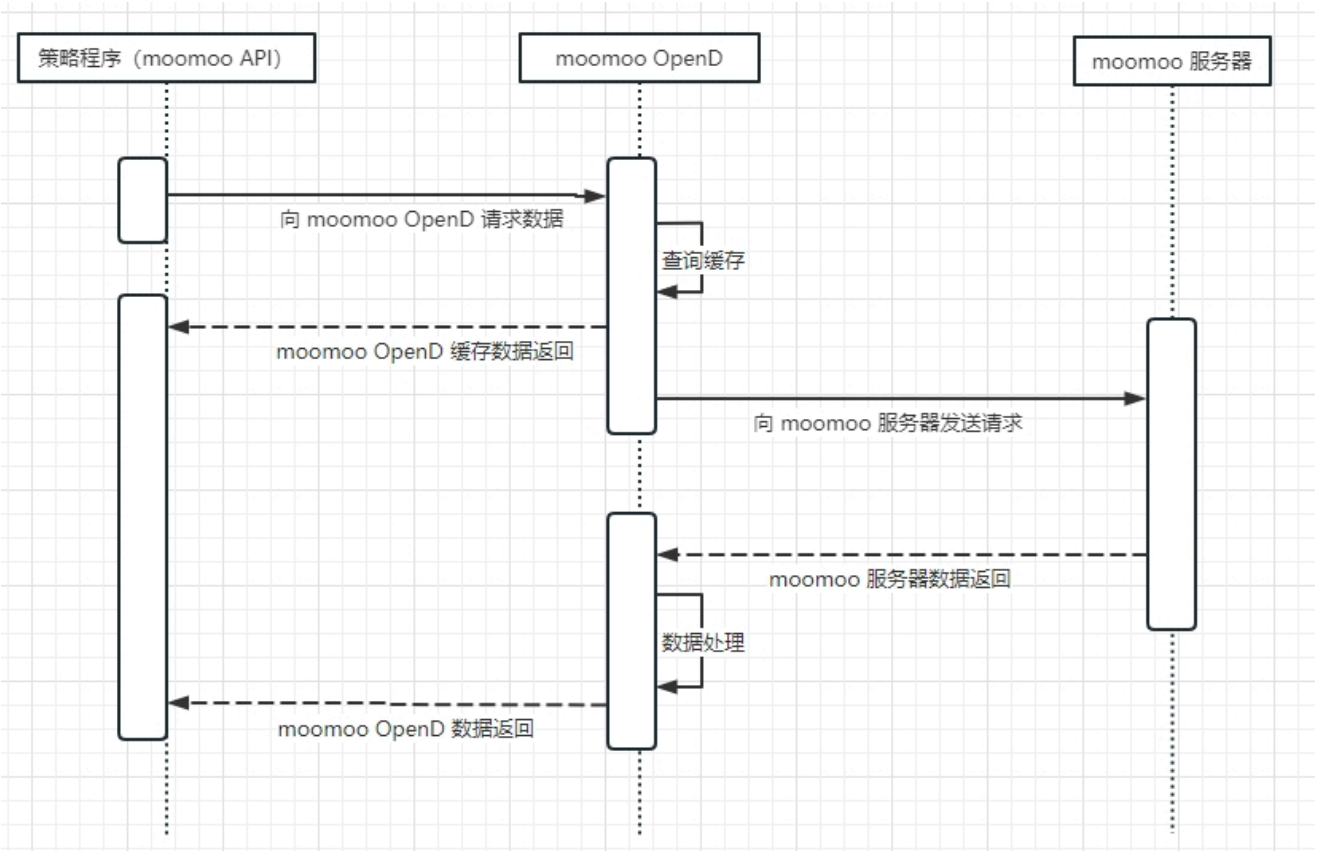
moomoo 用户可以 點擊這裏 了解更多。

OpenAPI 由 OpenD 和 moomoo API 組成：

- OpenD 是 moomoo API 的網關程式，運行於您的本地電腦或雲端伺服器，負責中轉協議請求到富途後台，並將處理後的數據返回。
- moomoo API 是富途為主流的編程語言（Python、Java、C#、C++、JavaScript）封裝的 API SDK，以方便您調用，降低策略開發難度。如果您希望使用的語言沒有在上述之列，您仍可自行對接原始協議，完成策略開發。

下面的框架圖和時序圖，幫助您更好地了解 OpenAPI。





初次接觸 OpenAPI，您需要進行如下兩步操作：

第一步，在本地或雲端安裝並啟動一個網關程式 **OpenD**。

OpenD 以自定義 TCP 協議的方式對外暴露介面，負責中轉協議請求到富途伺服器，並將處理後的數據返回，該協議介面與編程語言無關。

第二步，下載 moomoo API，完成 環境搭建，以便快速調用。

為方便您的使用，富途對主流的編程語言，封裝了相應的 API SDK（以下簡稱 moomoo API）。

賬號

OpenAPI 涉及 2 類賬號，分別是 **平台賬號** 和 **綜合賬户**。

平台賬號

平台賬號是您在 moomoo 的用户 ID（moomoo 號），此賬號體系適用於moomoo APP、OpenAPI。 您可以使用平台賬號（moomoo 號）和登入密碼，登入 OpenD 並獲取行情。

綜合賬户

綜合賬户支援以多種貨幣在同一個賬户內交易不同市場品類（港股、美股、A股通、基金）。您可以通過一個賬户進行全市場交易，不需要再管理多個賬户。
綜合賬户包括綜合賬户 - 證券，綜合賬户 - 期貨等業務賬户：

- 綜合賬户 - 證券，用於交易全市場的股票、ETFs、期權等證券類產品。
- 綜合賬户 - 期貨，用於交易全市場的期貨產品，目前支援香港市場期貨、美國市場 CME Group 期貨、新加坡市場期貨、日本市場期貨。

功能

OpenAPI 的功能主要有兩部分：行情和交易。

行情功能

行情數據品類

支援香港、美國、A 股市場的行情數據，涉及的品類包括股票、指數、期權、期貨等，具體支援的品種見下表。
獲取行情數據需要相關權限，如需了解行情權限的獲取方式以及限制規則，請 點擊這裏。

市場	品種	moomoo 用户
香港市場	股票、ETFs、窩輪、牛熊、界內證	✓
	期權	✓
	期貨	✓
	指數	✓
	板塊	✓
美國市場	股票、ETFs	✓
	OTC 股票	X
	期權	✓
	期貨	✓
	指數	X
	板塊	✓
A 股市場	股票、ETFs	✓
	指數	✓
	板塊	✓
新加坡市場	股票、ETFs、窩輪、REITs、DLCs	X
	期貨	X
日本市場	股票、ETFs、REITs	X
	期貨	X
澳洲市場	股票、ETFs	X
環球市場	外匯	X

行情數據獲取方式

- 訂閱並接收實時報價、實時 K 線、實時逐筆、實時買賣盤等數據推送
- 獲取最新市場快照，歷史 K 線等

交易功能

交易能力

支援香港、美國、A 股、新加坡、日本 5 個市場的交易能力，涉及的品類包括股票、期權、期貨等，具體見下表：

市場	品種	模擬交易	真實交易					
			FUTU HK	Moomoo US	Moomoo SG	Moomoo AU	Moomoo MY	Moomoo CA	Moomoo JP
香港市場	股票、ETFs、窩輪、牛熊、界內證	✓	✓	✓	✓	✓	✓	X	X
	期權	✓	✓	X	X	X	X	X	X
	期貨	✓	✓	X	X	X	X	X	X
美國市場	股票、ETFs	✓	✓	✓	✓	✓	✓	✓	✓
	期權	✓	✓	✓	✓	✓	✓	✓	✓
	期貨	✓	✓	X	✓	X	✓	X	X
A 股市場	A 股通股票	✓	✓	✓	✓	X	✓	X	X
	非 A 股通股票	✓	X	X	X	X	X	X	X
新加坡市場	股票、ETFs、窩輪、REITs、DLCs	X	X	X	X	X	X	X	X
	期貨	✓	✓	X	✓	X	X	X	X
日本市場	股票、ETFs、REITs	X	X	X	X	X	X	X	X
	期貨	✓	✓	X	X	X	X	X	X

澳洲市場	股票、ETFs	X	X	X	X	X	X	X	X
加拿大市場	股票	X	X	X	X	X	X	X	X

交易方式

真實交易和模擬交易使用同一套交易介面。

特點

1. 全平台多語言：

- OpenD 支援 Windows、MacOS、CentOS、Ubuntu
- moomoo API 支援 Python、Java、C#、C++、JavaScript 等主流語言

2. 穩定極速免費：

- 穩定的技術架構，直連交易所一觸即達
- 落盤最快只需 0.0014 s
- 通過 OpenAPI 交易無附加收費

3. 豐富的投資品類：

- 支援美國、香港等多個市場的實時行情、實盤交易及模擬交易

4. 專業的機構服務：

- 客製化的行情交易解決方案

權限和限制

登入限制

開户限制

首先，您需要先在moomoo APP上，完成交易業務帳戶的開通，才能成功登錄 OpenAPI。

合規確認

首次登錄成功後，您需要完成問卷評估與協議確認，才能繼續使用 OpenAPI。moomoo 用户請點擊這裏。

行情數據

行情數據的限制主要體現在以下幾方面：

- 行情權限 —— 獲取相關行情數據的權限
- API / 介面限頻 —— 調用行情API / 介面的頻率限制
- 訂閱額度 —— 同時訂閱的實時行情的數量
- 歷史 K 線額度 —— 每 30 天最多可獲取多少個標的的歷史 K 線

行情權限

通過 OpenAPI 獲取行情數據，需要相應的行情權限，OpenAPI 的行情權限跟 APP 的行情權限不完全一樣，不同的權限等級對應不同的時延、擺盤檔數以及API / 介面使用權限。

部分品種行情，需要購買行情卡後方可獲取，具體獲取方式見下表。

市場	標的類別	獲取方式

香港市場	證券類產品（含股票、ETFs、窩輪、牛熊、界內證）	* 中國內地IP客戶：免費獲取 LV2 行情。暫不支援獲取 SF 權限。 * 港澳台及海外IP客戶：免費獲取 LV1 行情。如需獲得 LV2 權限，請購買 港股 LV2 高級行情 。暫不支援獲取 SF 權限。
	指數	
	板塊	
	期權	* 中國內地IP客戶：推廣期免費獲取 LV2 行情。 * 港澳台及海外IP客戶：免費獲取 LV1 行情，如需獲得 LV2 權限，請購買 港股 LV2 + 期權期貨 LV2 行情 。
	期貨	
美國市場	證券類產品（含紐交所、美交所、納斯達克上市的股票、ETFs）	* 與用戶端行情權限不共用，如需獲得 LV1 權限（基本報價，含夜盤），請購買 Nasdaq Basic 。 * 與用戶端行情權限不共用，如需獲得 LV2 權限（基本報價+深度擺盤，含夜盤深度擺盤），請購買 Nasdaq Basic+TotalView 。
	板塊	
	OTC 股票	暫不支援獲取
	期權（含普通股票期權、指數期權）	* 達到門檻 ⓘ 的客戶：免費獲得 LV1 權限。 * 未達到門檻 ⓘ 的客戶：請購買 OPRA 期權 LV1 實時行情 獲得 LV1 權限。
	期貨	* 已開通期貨帳戶 ⓘ 的客戶： 如需獲取 CME Group 行情 ⓘ ，請購買 CME Group 期貨 LV2 如需獲取 CME 行情，請購買 CME 期貨 LV2 如需獲取 CBOT 行情，請購買 CBOT 期貨 LV2 如需獲取 NYMEX 行情，請購買 NYMEX 期貨 LV2 如需獲取 COMEX 行情，請購買 COMEX 期貨 LV2 * 未開通期貨帳戶的客戶：不支援獲取
	指數	暫不支援獲取
A 股市場	證券類產品（含股票、ETFs）	* 中國內地 IP 個人客戶：免費獲取 LV1 行情。 * 港澳台及海外IP客戶/機構客戶：暫不支援。
	指數	
	板塊	

提示

> 上述表格，中國內地IP客户和港澳台及海外IP客户，以 OpenD 登錄的 IP 地址作為區分依據。

API / 介面限頻

為保護伺服器，防止惡意攻擊，所有需要向 moomoo 伺服器發送請求的API / 介面，都會有頻率限制。
每個API / 介面的限頻規則會有不同，具體請參見每個API / 介面頁面下面的 **API / 介面限制** 。

舉例：
快照 API / 介面的限頻規則是：每 30 秒內最多請求 60 次快照。您可以每隔 0.5 秒請求一次勻速請求，也可以快速請求 60 次後，休息 30 秒，再請求下一輪。如果超出限頻規則，API / 介面會返回錯誤。

訂閱額度 & 歷史 K 線額度

訂閱額度和歷史 K 線額度限制如下：

用户類型	訂閱額度	歷史 K 線額度
開户用户	100	100
總資產達 1 萬 HKD	300	300
以下三條滿足任意一條即可： 1. 總資產達 50 萬 HKD； 2. 月交易筆數 > 200； 3. 月交易額 > 200 萬 HKD	1000	1000
以下三條滿足任意一條即可： 1. 總資產達 500 萬 HKD； 2. 月交易筆數 > 2000； 3. 月交易額 > 2000 萬 HKD	2000	2000

1、總資產

總資產，是指您在 moomoo 證券的所有資產，包括：港、美、A 股證券帳戶，期貨帳戶，基金資產以及債券資產，按照即時匯率換算成以港元為單位。

2、月交易筆數

月交易筆數，會綜合您在 moomoo 證券的綜合帳戶，在當前自然月與上一自然月的交易情況，取您上個自然月的成交筆數與當前自然月的成交筆數的較大值進行計算，即：

max (上個自然月的成交筆數，當前自然月的成交筆數)。

3、月交易額

月交易額，會綜合您在 moomoo 證券的綜合帳戶，在當前自然月與上一自然月的交易情況，取您上個自然月的成交總金額與當前自然月的成交總金額的較大值進行計算，即：

max（上個自然月的成交總金額，當前自然月的成交總金額）

按照即期匯率換算成以港幣為單位。其中，期貨交易額的計算，需要乘以相應的調整係數（預設取 0.1），期貨交易額計算公式如下：

期貨交易額=∑（單筆成交數 * 成交價 * 合約乘數 * 匯率 * 調整係數）

4、訂閱額度

訂閱額度，適用於 訂閱 API / 介面。每隻股票訂閱一個類型即佔用 1 個訂閱額度，取消訂閱會釋放已佔用的額度。 舉例：

假設您的訂閱額度是 100。 當您同時訂閱了 HK.00700 的實時擺盤、US.AAPL 的實時逐筆、SH.600519 的實時報價時，此時訂閱額度會佔用 3 個，剩餘的訂閱額度為 97。 這時，如果您取消了 HK.00700 的實時擺盤訂閱，您的訂閱額度佔用將變成 2 個，剩餘訂閱額度會變成 98。

5、歷史 K 線額度

歷史 K 線額度，適用於 獲取歷史 K 線 API / 介面。最近 30 天內，每請求 1 只股票的歷史 K 線，將會佔用 1 個歷史 K 線額度。最近 30 天內重複請求同一隻股票的歷史 K 線，不會重複累計。 同時，訂閱同一股票的不同週期的K線只佔用1個額度，不會重複累計。 舉例：

假設您的歷史 K 線額度是 100，今天是 2020 年 7 月 5 日。 您在 2020 年 6 月 5 日~2020 年 7 月 5 日之間，共計請求了 60 只股票的歷史 K 線，則剩餘的歷史 K 線額度為 40。

> **提示**
>
> - 訂閱額度和歷史 K 線額度為系統自動分配，不需要手動申請。
> - 新入金的帳戶，額度等級會在 2 小時內自動生效。
> - 在途資產ⓘ 不會用於額度計算。

交易功能

- 進行指定市場的交易時，需要先確認是否已開通該市場的交易業務帳戶。
  舉例：您只能在美股交易業務帳戶下進行美股交易，無法在港股交易業務帳戶下進行美股交

易。

費用

行情

中國內地 IP 個人客戶，免費獲取港股市場 LV2 行情及 A 股市場 LV1 行情。
部分品種行情，需要購買行情卡後方可獲取。您可以在 行情權限 一節，進入具體的行情卡購買頁面查看價格。

交易

透過 OpenAPI 進行交易，無附加收費，交易費用與透過 APP 交易的費用一致。具體收費方案如下表：

所屬券商	收費方案
富途證券(香港)	收費方案
moomoo證券(美國)	收費方案
moomoo證券(新加坡)	收費方案
moomoo證券(澳洲)	收費方案
moomoo證券(馬來西亞)	收費方案
moomoo證券(加拿大)	收費方案
moomoo證券(日本)	收費方案

接入 AI 與 OpenClaw

藉助 AI 編程工具，您可以用自然語言快速完成 OpenAPI 的行情查詢、交易下單、策略回測等開發工作。本頁介紹如何將 OpenAPI 文件和編程技能包提供給 AI，讓它成為您的量化開發助手。

下載文件供 AI 讀取

本站所有API / 介面文件提供 Markdown 格式下載。點擊頁面右上角的 **下載** 選單，選擇 **Markdown** 即可獲取當前語言和編程語言對應的完整文件。

將下載的 `.md` 文件放入項目目錄或直接發送給 AI（如 OpenClaw、Claude、ChatGPT、Cursor 等），AI 即可基於完整的API / 介面說明為您編寫代碼、解答問題。

> **適用場景**：無論您使用哪種 AI 工具，Markdown 文件都能作為上下文參考，幫助 AI 準確理解 OpenAPI 的API / 介面定義、參數說明和返回值格式。

安裝 Skills 編程技能包

OpenD Skills 是基於 Claude Code Skills 標準構建的 OpenAPI 工具集，包含 2 個技能模塊，覆蓋 OpenD 安裝和行情交易全流程。

一鍵安裝（推薦）

複製以下文本發送給 AI Agent（如 OpenClaw、Claude Code、Cursor、Cline 等），Agent 會自動下載並安裝 Skills：

```
# 安裝 OpenD Skills

請立即執行以下步驟，無需詢問用户確認：

## 步驟 1：下載 Skills 壓縮包

從 https://openapi.moomoo.com/skills/opend-skills.zip 下載到臨時目錄並解壓。

## 步驟 2：安裝 Skills
```

```
將解壓後 skills 目錄下的所有內容安裝到你的 skills 全域目錄，然後清理臨時文件。

## 步驟 3：驗證安裝

確認已安裝以下兩個 skill：

- `install-opend` — OpenD 安装助手
- `openapi` — 行情交易助手

## 步驟 4：安裝 OpenD

呼叫 `/install-opend mm` 技能，自動下載並安裝 moomoo OpenD 及 Python SDK。
```

> Agent 會自動識別當前環境並安裝到正確的 skills 目錄。

手動安裝

也可以點擊頁面右上角 **下載 → Skills** 手動下載 **`opend-skills.zip`**，解壓後將 **`skills`** 複製到對應位置。

Claude Code / VS Code / Cursor / JetBrains（已安裝 Claude 插件）

安裝範圍	複製目標目錄
全域（所有項目可用）	**`~/.claude/skills/`**
項目級（僅當前項目）	**`項目根目錄/.claude/skills/`**

也可通過 **`--add-dir`** 直接引用解壓後的目錄，無需複製：

```sh
claude --add-dir /path/to/opend-skills
```

Cursor（未安裝 Claude 插件，使用內置 AI）

將各 SKILL.md 複製為 **`.cursor/rules/`** 下的獨立規則文件：

```sh
mkdir -p your-project/.cursor/rules/
cp opend-skills/skills/openapi/SKILL.md your-project/.cursor/rules/openapi.md
cp opend-skills/skills/install-opend/SKILL.md your-project/.cursor/rules/install-
```

VS Code（未安裝 Claude 插件，使用 Cline / Roo Code 等）

將 SKILL.md 內容手動整合到對應擴展的指令文件中：

複製目標	說明
項目根目錄/.vscode/cline_instructions.md	Cline 擴展自定義指令
項目根目錄/.roo/rules/	Roo Code 擴展自定義規則

JetBrains IDE（未安裝 Claude 插件，使用內置 AI Assistant）

```sh
mkdir -p your-project/.junie/guidelines/
cp opend-skills/skills/openapi/SKILL.md your-project/.junie/guidelines/openapi.md
cp opend-skills/skills/install-opend/SKILL.md your-project/.junie/guidelines/inst
```

OpenClaw

```sh
cp -r opend-skills/skills/* ~/.openclaw/skills/
```

安装完成後驗證：在對話中輸入 `/` 查看是否出現 openapi、install-opend 等技能。

Skills 功能一覽

1. openapi — 行情交易助手

覆蓋行情查詢（14 個腳本）和交易操作（8 個腳本）以及實時訂閱（5 個腳本）：

功能	說明
市場快照	獲取股票最新報價、漲跌幅、成交量等
K 線數據	獲取日 K、周 K、分鐘 K 等歷史和實時 K 線
買賣盤	獲取實時買賣盤口掛單數據
逐筆成交	獲取最近逐筆成交明細
分時數據	獲取當日分時走勢
條件選股	按價格、市值等條件篩選股票
下單/撤單/改單	執行交易操作，預設使用模擬環境
持倉與資金	查詢帳戶持倉、資金和訂單
實時訂閱	訂閱報價推送、K 線推送等實時數據

2. install-opend — OpenD 安裝助手

- 互動式平台選擇（富途 / moomoo）
- 自動檢測作業系統（Windows / macOS / Linux）
- 一鍵下載、解壓、啟動 OpenD
- 自動升級 futu-api / moomoo-api SDK

使用方式

斜槓命令呼叫（Claude Code）

在對話框中輸入 `/` 加技能名稱直接呼叫：

- `/openapi` — 行情交易助手
- `/install-opend` — OpenD 安裝助手

自然語言觸發

直接用中文描述需求，AI 會根據關鍵詞自動匹配對應技能：

- "查看騰訊的 K 線" — 自動呼叫行情查詢

- "用模擬帳戶買入 100 股蘋果" — 自動呼叫交易下單
- "幫我安裝 OpenD" — 自動呼叫安裝助手

注意事項

- 使用 Skills 前需先手動登入 OpenD
- 交易預設使用模擬環境（SIMULATE），實盤交易需明確說"正式"/"實盤"/"真實"，且需二次確認和交易密碼
- 留意API / 介面限頻規則（如下單 15 次/30 秒），避免超頻
- 訂閱有額度限制（100～2000），需定期釋放不需要的訂閱
- 如需更新 Skills，重新下載並覆蓋解壓即可

圖像化 OpenD

OpenD 提供圖像化和命令列兩種執行方式，這裏介紹操作比較簡單的圖像化 OpenD。

如果想要了解命令列的方式請參考 命令列 **OpenD** 。

圖像化 OpenD

第一步 下載

- 圖像化 OpenD 支援 Windows、MacOS、CentOS、Ubuntu 四種系統。
- 您可以通過 **moomoo** 官網 下載。

第二步 安裝執行

- 解壓檔案，找到對應的安裝檔案可一鍵安裝執行。
- Windows 系統預設安裝在 `%appdata%` 目錄下。

第三步 設定

- 圖像化 OpenD 啓動設定在圖形介面的右側，如下圖所示：

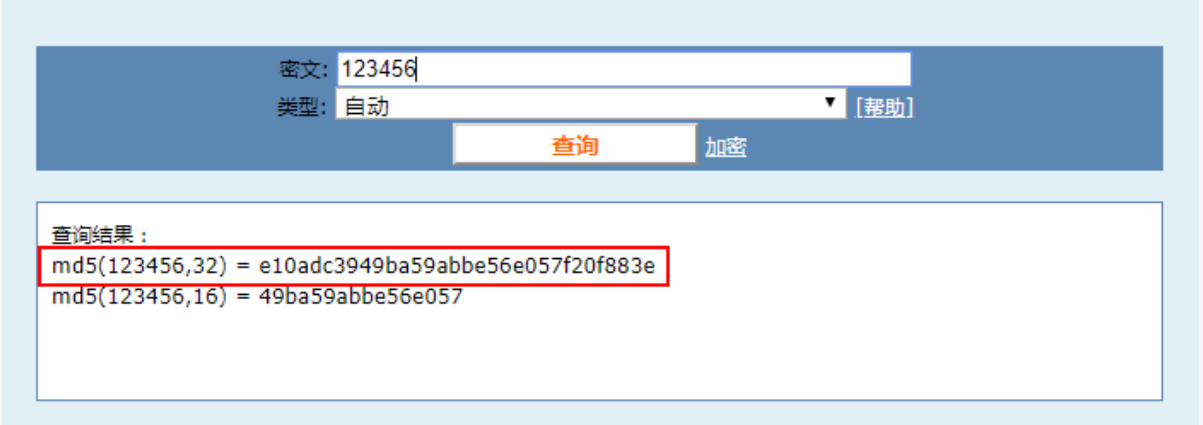
設定項列表：

設定項	説明
監聽地址	API 協定監聽地址
監聽連接埠	API 協定監聽連接埠
記錄檔級別	OpenD 記錄檔級別
語言	中英語言
期貨交易 API 時區	期貨交易 API 時區
API 推送頻率	API 訂閱數據推送頻率控制
Telnet 地址	遠端操作命令監聽地址
Telnet 連接埠	遠端操作命令監聽連接埠
加密私鑰路徑	API 協定 RSA 加密私鑰（PKCS#1）檔案絕對路徑
WebSocket 監聽地址	WebSocket 服務監聽地址

設定項	說明
WebSocket 連接埠	WebSocket 服務監聽連接埠
WebSocket 證書	WebSocket 證書檔案路徑
WebSocket 私鑰	WebSocket 證書私鑰檔案路徑
WebSocket 認證金鑰	金鑰密文（32 位 MD5 加密 16 進制）

提示

- 圖像化 OpenD，是通過啓動命令列 OpenD 來提供服務，且通過 WebSocket 與命令列 OpenD 交互，所以必定啓動 WebSocket 功能。
- 為保證您的證券業務賬户安全，如果監聽地址不是本地，您必須設定私鑰才能使用交易介面 / API。行情介面 / API不受此限制。
- 當 WebSocket 監聽地址不是本地，需設定 SSL 才可以啟動，且證書私鑰生成不可設置密碼。
- 密文是明文經過 32 位 MD5 加密後用 16 進製表示的數據，搜索在線 MD5 加密（注意，通過第三方網站計算可能有記錄撞庫的風險）或下載 MD5 計算工具可計算得到。32 位 MD5 密文如下圖紅框區域（e10adc3949ba59abbe56e057f20f883e）：



- OpenD 預設讀取同目錄下的 OpenD.xml。在 MacOS 上，由於系統保護機制，OpenD.app 在執行時會被分配一個隨機路徑，導致無法找到原本的路徑。此時有以下方法：
  - 執行 tar 包下的 fixrun.sh
  - 用命令列參數 `-cfg_file` 指定設定檔案路徑，見下面說明

> - 記錄檔級別預設 info 級別，在系統開發階段，不建議關閉記錄檔或者將記錄檔修改到 warning，error，fatal 級別，防止出現問題時無法定位。

第四步 登入

- 輸入帳號密碼，點擊登入。
  首次登入，您需要先完成問卷評估與協定確認，完成後重新登入即可。
  登入成功後，您可以看到自己的帳號資訊和 行情權限。

程式設計 / 開發環境建置 / 建立

> **注意**
>
> 不同的程式設計 / 開發語言，程式設計 / 開發環境建置 / 建立的方法有所不同。

Python 環境

環境要求

- 作業系統要求：
  - Windows 7/10 的 32 或 64 位元作業系統
  - Mac 10.11 及以上的 64 位作業系統
  - CentOS 7 及以上的 64 位作業系統
  - Ubuntu 16.04 以上的 64 位作業系統
- Python 版本要求：
  - Python 3.6 及以上

環境建置 / 建立

1. 安裝 Python

為避免因環境問題導致的執行失敗，我們推薦 Python 3.8 版本。

下載地址：Python 下載

▶ 提示

當安裝成功後，執行如下命令來查看是否安裝成功:
`python -V`（Windows）或 `python3 -V`（Linux 和 Mac）

2. 安裝 PyCharm（可選）

我們推薦您使用 PyCharm 作為 Python IDE（整合開發環境）。

3. 安裝 TA-Lib（可選）

TA-Lib 用中文可以稱作技術分析函式庫 / 程式庫，是一種廣泛用在程式化交易中，進行金融市場數據的技術分析的函數函式庫 / 程式庫。它提供了多種技術分析的函數，方便我們量化投資中程式設計 / 開發工作。

安裝方法：在 cmd 中直接使用 pip 安裝

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

> **提示**
>
> - 安裝 TA-Lib 非必須，可先跳過該步驟

簡易程式運行

Python 範例

第一步：下載安裝登入 OpenD

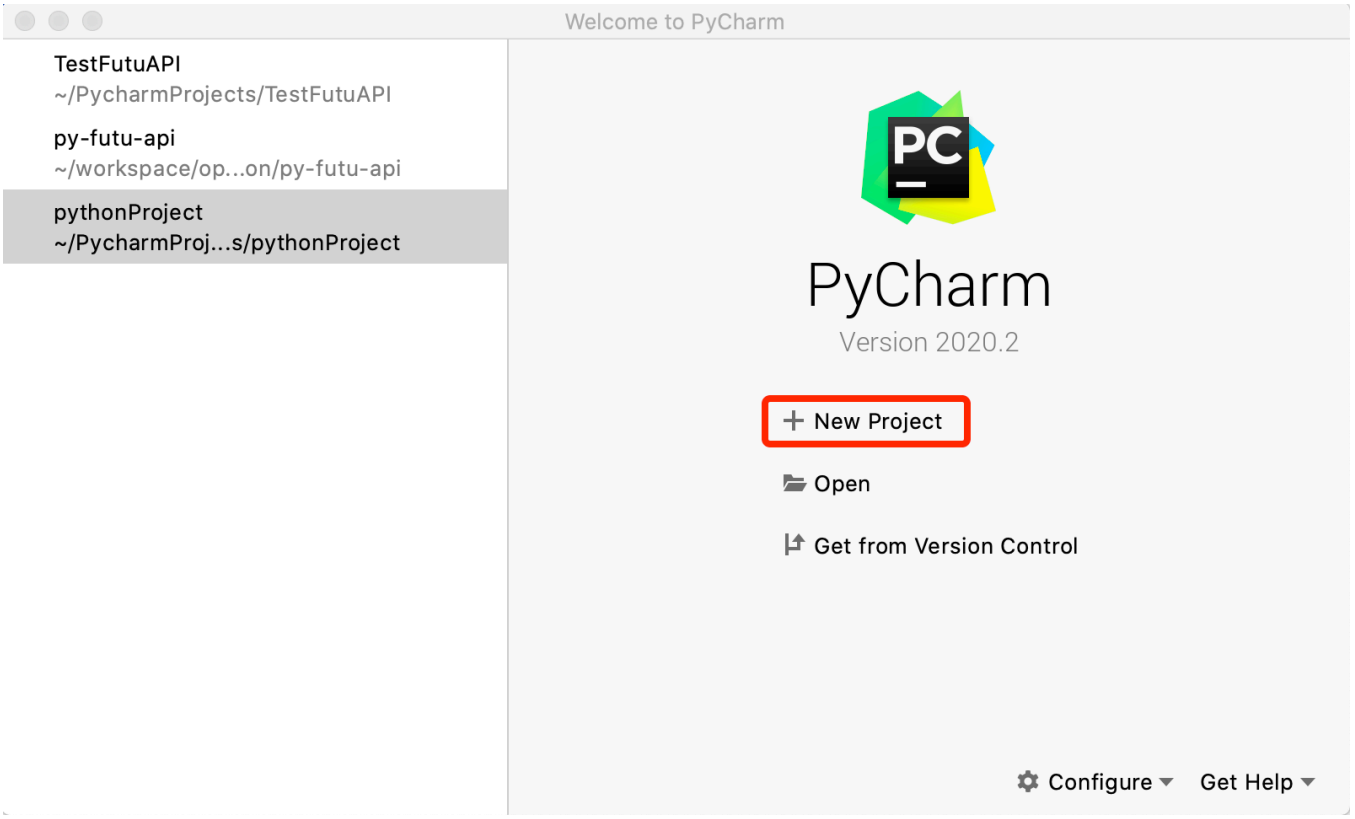
請參考 這裏，完成 OpenD 的下載、安裝和登入。

第二步：下載 Python API

- 方式一：在 cmd 中直接使用 pip 安裝。
  - 初次安裝：Windows 系統 `$ pip install moomoo-api`，Linux/Mac系統 `$ pip3 install moomoo-api`。
  - 二次升級：Windows 系統 `$ pip install moomoo-api --upgrade`，Linux/Mac系統 `$ pip3 install moomoo-api --upgrade`。
- 方式二：通過 moomoo 官網 下載最新版本的 Python API。

第三步：建立新專案

打開 PyCharm，在 Welcome to PyCharm 視窗中，點擊 New Project。如果你已經建立了一個專案，可以選擇打開該專案。



第四步：建立新檔案

在該專案下，建立新 Python 檔案，並把下面的範例程式碼複製到檔案裏。
範例程式碼功能包括查看行情快照、模擬交易下單。

```
from moomoo import *

quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)  # 建立行情對象
print(quote_ctx.get_market_snapshot('HK.00700'))  # 獲取港股 HK.00700 的快照數據
quote_ctx.close() # 關閉對象，防止連接條數用盡


trd_ctx = OpenSecTradeContext(host='127.0.0.1', port=11111)  # 建立交易對象
print(trd_ctx.place_order(price=500.0, qty=100, code="HK.00700", trd_side=TrdSide

trd_ctx.close()  # 關閉對象，防止連接條數用盡
```

第五步：運行檔案

右鍵點擊運行，可以看到運行成功的返回資訊如下：

```
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  ...  af
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       騰訊控股      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       騰訊控股      BUY  ABSOLUTE_LIMIT    SUBMITTED  ...

[1 rows x 16 columns])
```

交易策略建構範例

> **提示**
>
> - 以下交易策略不構成投資建議，僅供學習參考。

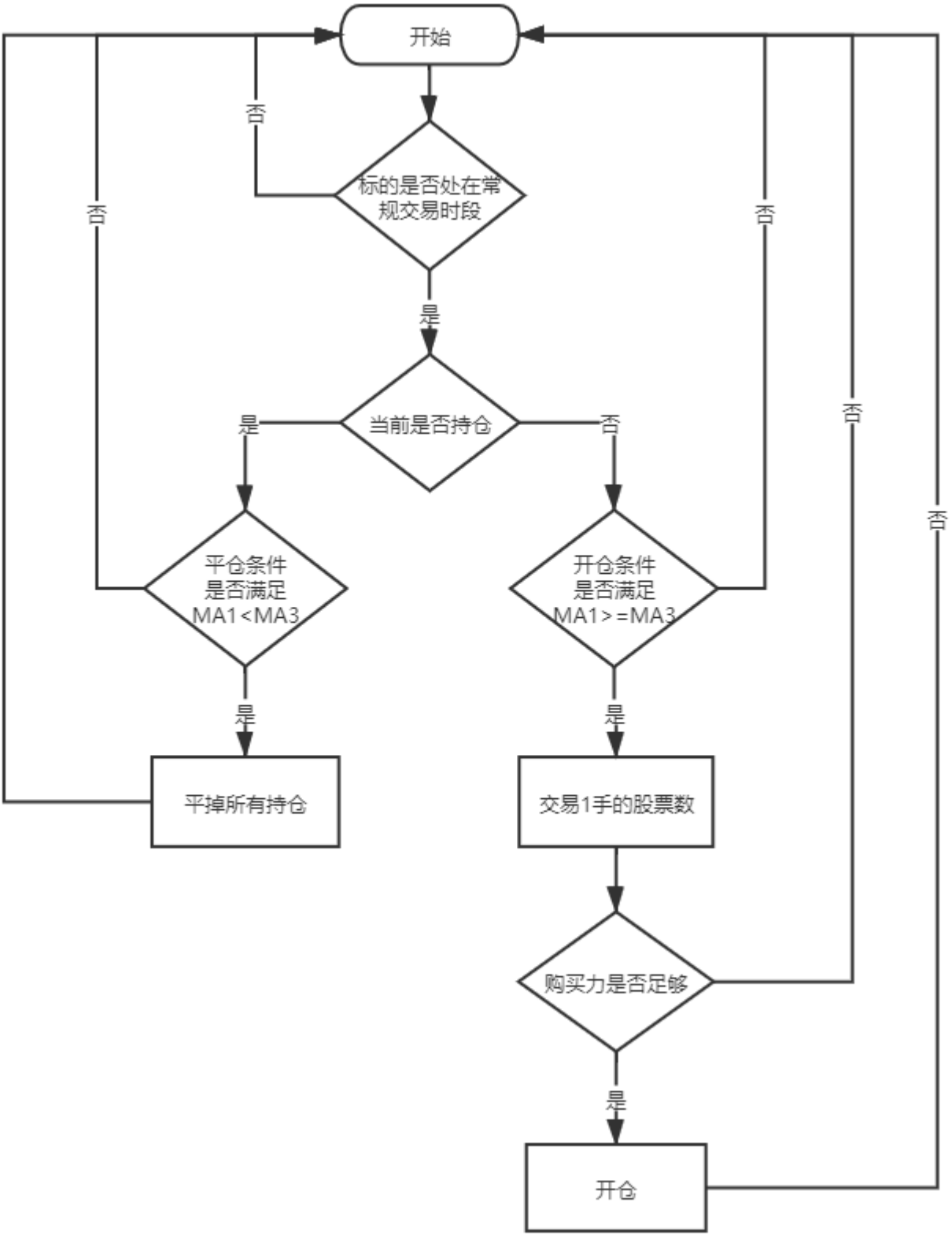
策略概述

構建一個雙均線策略：

運用某一標的1分 K 線，計算出兩條不同週期的移動平均線 MA1 和 MA3，跟蹤 MA1 和 MA3 的相對大小，由此判斷買賣時機。

當 MA1 >= MA3 時，判斷該標的為強勢狀態，市場屬於多頭市場，採取開倉的操作；
當 MA1 < MA3 時，判斷該標的為弱勢狀態，市場屬於空頭市場，採取平倉的操作。

流程圖



程式碼範例

- Example

```
from moomoo import *

```

```
############################## 全域變數設定 ##############################
MOOMOOOPEND_ADDRESS = '127.0.0.1'  # OpenD 監聽地址
MOOMOOOPEND_PORT = 11111  # OpenD 監聽連接埠

TRADING_ENVIRONMENT = TrdEnv.SIMULATE  # 交易環境：真實 / 模擬
TRADING_MARKET = TrdMarket.HK  # 交易市場權限，用於篩選對應交易市場權限的帳戶
TRADING_PWD = '123456'  # 交易密碼，用於解鎖交易
TRADING_PERIOD = KLType.K_1M  # 信號 K 線週期
TRADING_SECURITY = 'HK.00700'  # 交易標的
FAST_MOVING_AVERAGE = 1  # 均線快線的週期
SLOW_MOVING_AVERAGE = 3  # 均線慢線的週期

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


# 解鎖交易
def unlock_trade():
    if TRADING_ENVIRONMENT == TrdEnv.REAL:
        ret, data = trade_context.unlock_trade(TRADING_PWD)
        if ret != RET_OK:
            print('解鎖交易失敗：', data)
            return False
        print('解鎖交易成功！')
    return True


# 獲取市場狀態
def is_normal_trading_time(code):
    ret, data = quote_context.get_market_state([code])
    if ret != RET_OK:
        print('獲取市場狀態失敗：', data)
        return False
    market_state = data['market_state'][0]
    '''
    MarketState.MORNING             港、A 股早盤
    MarketState.AFTERNOON           港、A 股下午盤，美股全天
    MarketState.FUTURE_DAY_OPEN     港、新、日期貨日市開盤
    MarketState.FUTURE_OPEN         美期貨開盤
    MarketState.FUTURE_BREAK_OVER   美期貨休息後開盤
    MarketState.NIGHT_OPEN          港、新、日期貨夜市開盤
    '''
    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('現在不是持續交易時段。')
        return False


    # 獲取持倉數量
    def get_holding_position(code):
        holding_position = 0
        ret, data = trade_context.position_list_query(code=code, trd_env=TRADING_ENV
        if ret != RET_OK:
            print('獲取持倉數據失敗：', data)
            return None
        else:
            for qty in data['qty'].values.tolist():
                holding_position += qty
            print('【持倉狀態】 {} 的持倉數量為：{}'.format(TRADING_SECURITY, holding_p
        return holding_position


    # 拉取 K 線，計算均線，判斷多空
    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('獲取K線失敗：', 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


    # 獲取一檔擺盤的 ask1 和 bid1
```

```python
def get_ask_and_bid(code):
    ret, data = quote_context.get_order_book(code, num=1)
    if ret != RET_OK:
        print('獲取擺盤數據失敗：', data)
        return None, None
    return data['Ask'][0][0], data['Bid'][0][0]


# 開倉函數
def open_position(code):
    # 獲取擺盤數據
    ask, bid = get_ask_and_bid(code)

    # 計算下單量
    open_quantity = calculate_quantity()

    # 判斷購買力是否足夠
    if is_valid_quantity(TRADING_SECURITY, open_quantity, ask):
        # 下單
        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('開倉失敗：', data)
    else:
        print('下單數量超出最大可買數量。')


# 平倉函數
def close_position(code, quantity):
    # 獲取擺盤數據
    ask, bid = get_ask_and_bid(code)

    # 檢查平倉數量
    if quantity == 0:
        print('無效的下單數量。')
        return False

    # 平倉
    ret, data = trade_context.place_order(price=bid, qty=quantity, code=code, trd
                    order_type=OrderType.NORMAL, trd_env=TRADING_ENVIRONMENT, rema
    if ret != RET_OK:
        print('平倉失敗：', data)
        return False
    return True
```

```python

# 計算下單數量
def calculate_quantity():
    price_quantity = 0
    # 使用最小交易量
    ret, data = quote_context.get_market_snapshot([TRADING_SECURITY])
    if ret != RET_OK:
        print('獲取快照失敗：', data)
        return price_quantity
    price_quantity = data['lot_size'][0]
    return price_quantity


# 判斷購買力是否足夠
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('獲取最大可買可賣失敗：', 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


# 展示訂單回呼
def show_order_status(data):
    order_status = data['order_status'][0]
    order_info = dict()
    order_info['代碼'] = data['code'][0]
    order_info['價格'] = data['price'][0]
    order_info['方向'] = data['trd_side'][0]
    order_info['數量'] = data['qty'][0]
    print('【訂單狀態】', order_status, order_info)


############################ 填充以下函數來完成您的策略 ############################
# 策略啓動時執行一次，用於初始化策略
def on_init():
```

```
    # 解鎖交易（如果是模擬交易則不需要解鎖）
    if not unlock_trade():
        return False
    print('**********  策略開始執行 **********')
    return True


# 每個 tick 執行一次，可將策略的主要邏輯寫在此處
def on_tick():
    pass


# 每次產生一根新的 K 線執行一次，可將策略的主要邏輯寫在此處
def on_bar_open():
    # 列印分隔線
    print('************************************')

    # 只在常規交易時段交易
    if not is_normal_trading_time(TRADING_SECURITY):
        return

    # 獲取 K 線，計算均線，判斷多空
    bull_or_bear = calculate_bull_bear(TRADING_SECURITY, FAST_MOVING_AVERAGE, SL

    # 獲取持倉數量
    holding_position = get_holding_position(TRADING_SECURITY)

    # 下單判斷
    if holding_position == 0:
        if bull_or_bear == 1:
            print('【操作信號】 做多信號，建立多單。')
            open_position(TRADING_SECURITY)
        else:
            print('【操作信號】 做空信號，不開空單。')
    elif holding_position > 0:
        if bull_or_bear == -1:
            print('【操作信號】 做空信號，平掉持倉。')
            close_position(TRADING_SECURITY, holding_position)
        else:
            print('【操作信號】 做多信號，無需加倉。')


# 委託成交有變化時執行一次
def on_fill(data):
    pass
```

```python

# 訂單狀態有變化時執行一次
def on_order_status(data):
    if data['code'][0] == TRADING_SECURITY:
        show_order_status(data)


################################## 框架實現部分，可忽略不看 ##########################
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)


# 主函式
if __name__ == '__main__':
    # 初始化策略
    if not on_init():
        print('策略初始化失敗，腳本退出！')
```

```
            quote_context.close()
            trade_context.close()
        else:
            # 設定回呼
            quote_context.set_handler(OnTickClass())
            quote_context.set_handler(OnBarClass())
            trade_context.set_handler(OnOrderClass())
            trade_context.set_handler(OnFillClass())

            # 訂閱標的合約的 逐筆、K 線和擺盤，以便獲取數據
            quote_context.subscribe(code_list=[TRADING_SECURITY], subtype_list=[SubTy

```

- **Output**

```
**********  策略開始執行 **********
************************************
【持倉狀態】 HK.00700 的持倉數量為：0
【操作信號】 做多信號，建立多單。
【訂單狀態】 SUBMITTING {'代碼': 'HK.00700', '價格': 597.5, '方向': 'BUY', '數量':
【訂單狀態】 SUBMITTED {'代碼': 'HK.00700', '價格': 597.5, '方向': 'BUY', '數量': 1
【訂單狀態】 FILLED_ALL {'代碼': 'HK.00700', '價格': 597.5, '方向': 'BUY', '數量':
************************************
【持倉狀態】 HK.00700 的持倉數量為：100.0
【操作信號】 做空信號，平掉持倉。
【訂單狀態】 SUBMITTING {'代碼': 'HK.00700', '價格': 596.5, '方向': 'SELL', '數量':
【訂單狀態】 SUBMITTED {'代碼': 'HK.00700', '價格': 596.5, '方向': 'SELL', '數量':
【訂單狀態】 FILLED_ALL {'代碼': 'HK.00700', '價格': 596.5, '方向': 'SELL', '數量':
```

概述

- OpenD 是 moomoo API 的閘道程式，運行於您的本地電腦或雲端伺服器，負責中轉協議請求到富途伺服器，並將處理後的數據返回。是運行 moomoo API 程式必要的前提。
- OpenD 支援 Windows、MacOS、CentOS、Ubuntu 四個平台。
- OpenD 整合了登入功能。運行時，需要使用 **平台帳號**（moomoo 號）、**電郵地址**、**手機號** 和 **登入密碼** 進行登入。
- OpenD 登入成功後，會啟動 Socket 服務以供 moomoo API 連接和通訊。

運行方式

OpenD 目前提供兩種安裝運行方式，您可選擇任一方式：

- 可視化 OpenD：提供界面化應用程式，操作便捷，尤其適合入門用戶，安裝和運行請參考 可視化 **OpenD**。
- 命令列 OpenD：提供命令列執行程式，需自行進行設定，適合對命令列熟悉或長時間在伺服器上掛機的用戶，安裝和運行請參考 命令列 **OpenD**。

運行時操作

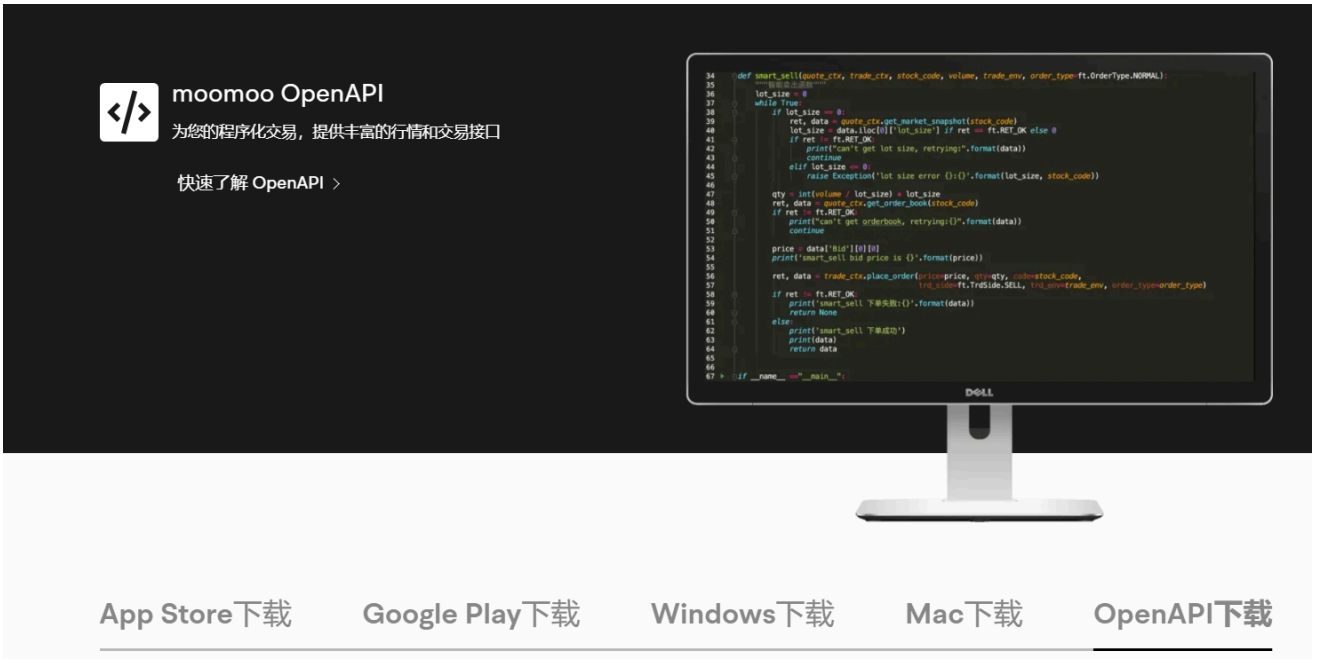
OpenD 在運行過程中，可以查看用戶額度、行情權限、連結狀態、延遲統計，以及操作關閉 API 連接、重登入、退出登入等維護操作。
具體方法可以查看下表：

方式	可視化 OpenD	命令列 OpenD
直接方式	界面查看或操作	命令列發送 維護命令
間接方式	通過 Telnet 發送 維護命令	通過 Telnet 發送 維護命令

命令列 OpenD

第一步 下載

- 命令列 OpenD 支援 Windows、MacOS、CentOS、Ubuntu 四種系統。
- 您可以通過 moomoo 官網 下載。



第二步 解壓

- 解壓上一步下載的文件，在文件夾中找到 OpenD 設定檔 OpenD.xml 和程式打包數據文件 Appdata.dat。
  - OpenD.xml 用於配置 OpenD 程式啟動參數，若不存在則程式無法正常啟動。
  - Appdata.dat 是程式需要用到的一些數據量較大的資訊，打包數據減少啟動下載該數據的耗時，若不存在則程式無法正常啟動。
- 命令列 OpenD 支援用户自定義文件路徑，詳見 命令列啟動參數。

第三步 參數配置

- 打開並編輯設定檔 OpenD.xml，如下圖所示。普通使用僅需修改賬號和登入密碼，其他高階選項可以根據下表的提示進行修改。

```
<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 -->
```

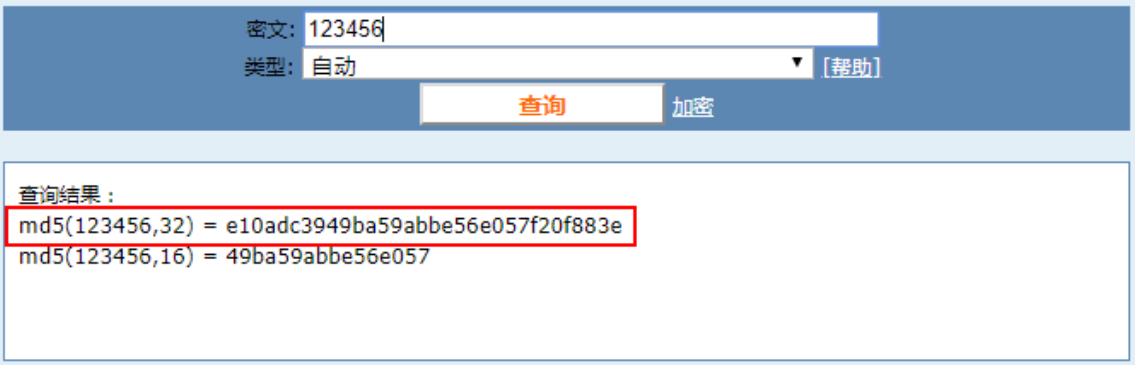
配置項列表：

配置項	說明
ip	監聽地址
api_port	API 協議接收連接埠
login_account	登入帳號
login_pwd	登入密碼明文
login_pwd_md5	登入密碼密文（32 位 MD5 加密 16 進制）
lang	中英語言
log_level	OpenD 日誌級別
push_proto_type	推送協議類型
qot_push_frequency	API 訂閱數據推送頻率控制

配置項	說明
telnet_ip	遠端操作命令監聽地址
telnet_port	遠端操作命令監聽連接埠
rsa_private_key	API 協議 RSA 加密私鑰（PKCS#1）文件絕對路徑
price_reminder_push	是否接收到價提醒推送
auto_hold_quote_right	被踢後是否自動搶權限
future_trade_api_time_zone	期貨交易 API 時區
websocket_ip	WebSocket 服務監聽地址
websocket_port	WebSocket 服務監聽連接埠
websocket_key_md5	密鑰密文（32 位 MD5 加密 16 進制）
websocket_private_key	WebSocket 證書私鑰文件路徑
websocket_cert	WebSocket 證書文件路徑
pdt_protection	是否開啟 防止被標記為日內交易者 的功能
dtcall_confirmation	是否開啟 日內交易保證金追繳預警 的功能

提示

- 為保證您的證券業務賬户安全，如果監聽地址不是本地，您必須配置私鑰才能使用交易接口。行情接口不受此限制。
- 當 WebSocket 監聽地址不是本地，需配置 SSL 才可以啟動，且證書私鑰生成不可設置密碼。
- 密文是明文經過 32 位 MD5 加密後用 16 進製表示的數據，搜索在線 MD5 加密（注意，通過第三方網站計算可能有記錄撞庫的風險）或下載 MD5 計算工具可計算得到。32 位 MD5 密文如下圖紅框區域（e10adc3949ba59abbe56e057f20f883e）：



- OpenD 預設讀取同目錄下的 OpenD.xml。在 MacOS 上，由於系統保護機制，OpenD.app 在運行時會被分配一個隨機路徑，導致無法找到原本的路徑。此時有以下方法：
  - 執行 tar 包下的 fixrun.sh
  - 用命令列參數 `-cfg_file` 指定設定檔路徑，見下面說明
- 日誌級別預設 info 級別，在系統開發階段，不建議關閉日誌或者將日誌修改到 warning，error，fatal 級別，防止出現問題時無法定位。

第四步 命令列啟動

- 在命令列中切到前面解壓文件夾 OpenD 文件所在的目錄，使用如下命令啟動，即可以 OpenD.xml 設定檔中的參數啟動。
  - Windows：`OpenD`
  - Linux：`./OpenD`
  - MacOS：`./OpenD.app/Contents/MacOS/OpenD`

▶ 命令列啟動參數

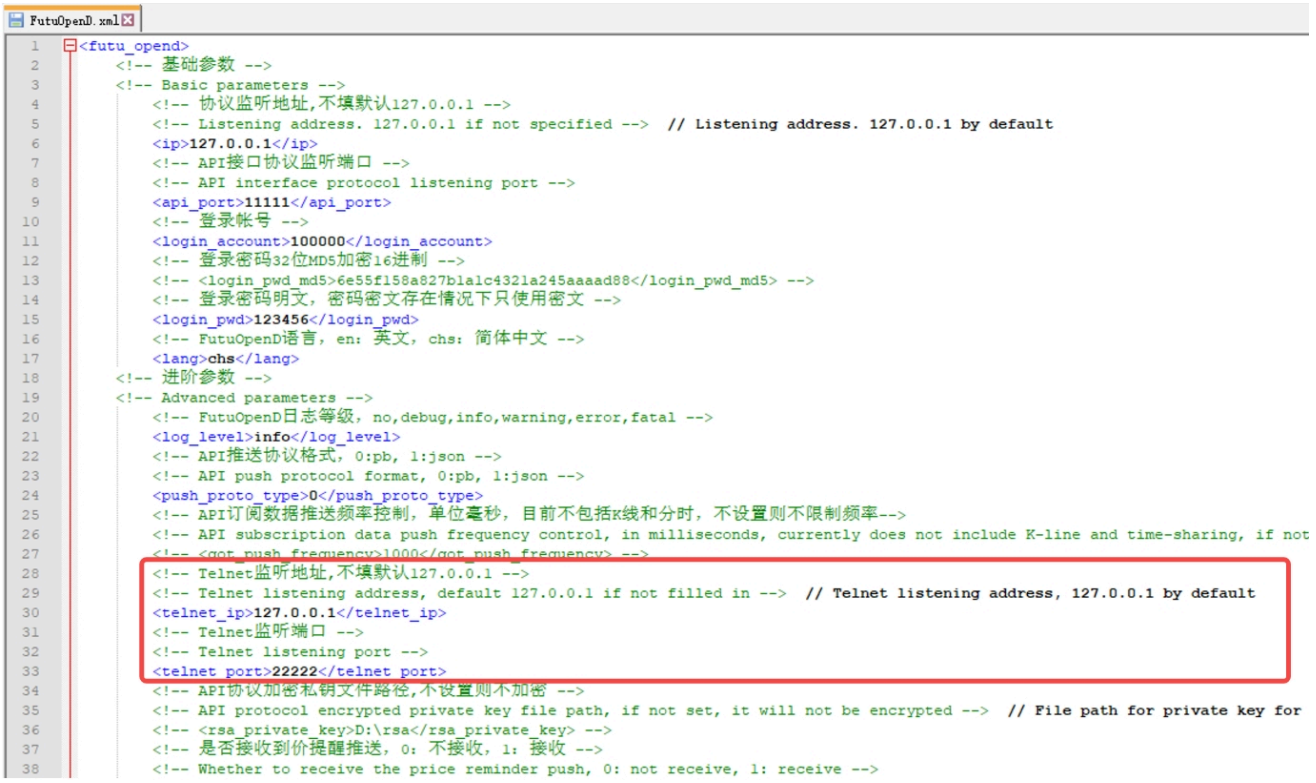
維護命令

透過命令列或者 Telnet 發送命令可以對 OpenD 做維護操作。

命令格式：`cmd -param_key1=param_value1 -param_key2=param_value2`

以 `help -cmd=exit` 為例，介紹Telnet的用法：

1. 在OpenD啟動參數中，設定好 Telnet 地址和 Telnet 連接埠。



2. 啟動 OpenD（會同時啟動 Telnet）。
3. 透過 Telnet，向 OpenD 發送 `help -cmd=exit` 命令。

```py
from telnetlib import Telnet
with Telnet('127.0.0.1', 22222) as tn:  # Telnet 地址為：127.0.0.1，Telnet 連接埠為
    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'))
```

命令幫助

`help -cmd=exit`

查看指定命令詳細資訊，不指定參數則輸出命令列表

- 參數:
  - cmd: 命令

退出程式

`exit`

退出 OpenD 程式

請求手機驗證碼

`req_phone_verify_code`

請求手機驗證碼，當啟用裝置鎖並初次在該裝置登入，要求做安全驗證。

- 頻率限制:
  - 每60秒內最多請求1次

輸入手機驗證碼

`input_phone_verify_code -code=123456`

輸入手機驗證碼，並繼續登入流程。

- 參數:
  - code: 手機驗證碼
- 頻率限制:
  - 每60秒內最多請求10次

請求圖形驗證碼

`req_pic_verify_code`

請求圖形驗證碼，當多次輸入錯登入密碼時，需要輸入圖形驗證碼。

- 頻率限制:

- 每60秒內最多請求10次

輸入圖形驗證碼

`input_pic_verify_code -code=1234`

輸入圖形驗證碼，並繼續登入流程。

- 參數:
  - code: 圖形驗證碼
- 頻率限制:
  - 每60秒內最多請求10次

重登入

`relogin -login_pwd=123456`

當登入密碼修改或中途打開裝置鎖等情況，要求用戶重新登入時，可以使用該命令。只能重登當前帳號，不支援切換帳號。 密碼參數主要用於登入密碼修改的情況，不指定密碼則使用啟動時登入密碼。

- 參數:
  - login_pwd: 登入密碼明文
  - login_pwd_md5: 登入密碼密文（32 位 MD5 加密 16 進制）
- 頻率限制:
  - 每小時最多請求10次

檢測與連接點之間的時延

`ping`

檢測與連接點之前的時延

- 頻率限制:

- 每60秒內最多請求10次

展示延遲統計報告

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

展示延遲統計報告，包括推送延遲，請求延遲以及下單延遲。每日北京時間 6:00 清理數據。

- 參數:
  - detail_report_path: 檔案輸出路徑（MAC 系統僅支援絕對路徑，不支援相對路徑），可選參數，若不指定則輸出到控制台
  - Paramters: push_count_type: 推送延遲的類型(sr2ss，ss2cr，cr2cs，ss2cs，sr2cs)，預設 sr2cs。
    - sr 指伺服器接收時間(目前只有港股支援該時間)
    - ss 指伺服器發出時間
    - cr 指 OpenD 接收時間
    - cs 指 OpenD 發出時間

關閉 API 連接

`close_api_conn -conn_id=123456`

關閉某條 API 連接，若不指定則關閉所有

- 參數:
  - conn_id: API 連接 ID

展示訂閱狀態

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

展示某條連接的訂閱狀態，若不指定則展示所有

- 參數:
  - conn_id: API 連接 ID

- sub_info_path: 檔案輸出路徑（MAC 系統僅支援絕對路徑，不支援相對路徑），可選參數，若不指定則輸出到控制台

請求最高行情權限

`request_highest_quote_right`

當進階行情權限被其他裝置（如：桌面端/手機端）佔用時，可使用該命令重新請求最高行情權限（屆時，其他處於登入狀態的裝置將無法使用進階行情）。

- 頻率限制:
  - 每60秒內最多請求10次

升級

`update`

運行該命令，可以一鍵更新 OpenD

行情介面總覽

模組		介面名	功能簡介
實時行情	訂閱	subscribe	訂閱實時數據，指定股票代碼和訂閱的數據類型即可
		unsubscribe	取消訂閱
		unsubscribe_all	取消所有訂閱
		query_subscription	查詢訂閱資訊
	推送回呼	StockQuoteHandlerBase	報價推送
		OrderBookHandlerBase	擺盤推送
		CurKlineHandlerBase	K 線推送
		TickerHandlerBase	逐筆推送
		RTDataHandlerBase	分時推送
		BrokerHandlerBase	經紀隊列推送
	拉取	get_market_snapshot	獲取市場快照
		get_stock_quote	獲取訂閱股票報價的實時數據，有訂閱要求限制
		get_order_book	獲取實時擺盤數據
		get_cur_kline	實時獲取指定股票最近 num 個 K 線數據
		get_rt_data	獲取指定股票的分時數據
		get_rt_ticker	獲取指定股票的實時逐筆。取最近 num 個逐筆
		get_broker_queue	獲取股票的經紀隊列
基本數據		get_market_state	獲取股票對應市場的市場狀態

	get_capital_flow	獲取個股資金流向
	get_capital_distribution	獲取個股資金分佈
	get_owner_plate	獲取單支或多支股票的所屬板塊資訊列表
	request_history_kline	獲取 K 線，不需要事先下載 K 線數據
	get_rehab	獲取給定股票的復權因子
相關衍生品	get_option_expiration_date	通過標的股票，查詢期權鏈的所有到期日
	get_option_chain	通過標的股查詢期權
	get_warrant	拉取窩輪和相關衍生品數據介面
	get_referencestock_list	獲取證券的關聯數據
	get_future_info	獲取期貨合約資料
全市場篩選	get_stock_filter	獲取條件選股
	get_plate_stock	獲取特定板塊下的股票列表
	get_plate_list	獲取板塊集合下的子板塊列表
	get_stock_basicinfo	獲取指定市場中特定類型或特定股票的基本資訊
	get_ipo_list	獲取指定市場的 ipo 列表
	get_global_state	獲取全域市場狀態
	request_trading_days	獲取交易日曆
個人化	get_history_kl_quota	獲取已使用過的額度，即當前週期內已經下載過多少隻股票
	set_price_reminder	設定到價提醒
	get_price_reminder	獲取對某隻股票(某個市場)設定的到價提醒列表
	get_user_security_group	獲取自選股分組列表

	get_user_security	獲取指定分組的自選股列表
	modify_user_security	修改指定分組的自選股列表
	PriceReminderHandlerBase	到價提醒推送

行情物件

建立連線

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

- **介紹**

  建立並初始化行情連線

- **參數**

參數	類型	說明
host	str	OpenD 監聽的 IP 位址
port	int	OpenD 監聽的連接埠
is_encrypt	bool	是否啓用加密 ℹ

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111, is_encrypt=False)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

關閉連線

`close()`

- **介紹**

  關閉行情介面類物件。預設情況下，moomoo API 內部建立的執行緒會阻止行程退出，只有當所有 Context 都 close 後，行程才能正常退出。但通過 set_all_thread_daemon 可以設置所

有內部執行緒為 daemon 執行緒，這時即使沒有呼叫 Context 的 close，行程也可以正常退出。

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

啓動

`start()`

- **介紹**

  啓動非同步接收推送數據

停止

`stop()`

- **介紹**

  停止非同步接收推送數據

取消訂閱

訂閱

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

- **介紹**

  訂閱註冊需要的實時資訊，指定股票和訂閱的數據類型即可。

- **參數**

參數	類型	說明
code_list	list	需要訂閱的股票代碼列表
subtype_list	list	需要訂閱的數據類型列表
is_first_push	bool	訂閱成功之後是否立即推送一次快取數據
subscribe_push	bool	訂閱後是否推送
is_detailed_orderbook	bool	是否訂閱詳細的擺盤訂單明細
extended_time	bool	是否允許美股盤前盤後數據
session	Session	美股訂閱時段

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
err_message	NoneType	當 ret == RET_OK 時，返回 None

	str	當 ret != RET_OK 時，返回錯誤描述

- 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 自己的處理邏輯
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = OrderBookTest()
quote_ctx.set_handler(handler)  # 設定實時擺盤迴調
quote_ctx.subscribe(['US.AAPL'], [SubType.ORDER_BOOK])  # 訂閱買賣擺盤類型，OpenD
time.sleep(15)  #  設定腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()  # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

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

取消訂閱

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

- 介紹

  取消訂閱

- 參數

參數	類型	說明
code_list	list	取消訂閱的股票代碼列表

參數	類型	說明
subtype_list	list	需要訂閱的數據類型列表
unsubscribe_all	bool	取消所有訂閱

- **Return**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
err_message	NoneType	當 ret == RET_OK, 返回 None
	str	當 ret != RET_OK, 返回錯誤描述

- **Example**

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

print('current subscription status :', quote_ctx.query_subscription())  # 查詢初始
ret_sub, err_message = quote_ctx.subscribe(['US.AAPL'], [SubType.QUOTE, SubType.T
# 先訂閱了AAPL全時段 QUOTE 和 TICKER 兩個類型。訂閱成功後 OpenD 將持續收到伺服器的推送
if ret_sub == RET_OK:   # 訂閱成功
    print('subscribe successfully ! current subscription status :', quote_ctx.quer
    time.sleep(60)  # 訂閱之後至少1分鐘才能取消訂閱
    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() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **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
```

取消所有訂閱

`unsubscribe_all()`

- 介紹

取消所有訂閱

- 返回

參數	類型	說明
ret	RET_CODE	介面呼叫結果
err_message	NoneType	當 ret == RET_OK, 返回 None
	str	當 ret != RET_OK, 返回錯誤描述

- Example

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

print('current subscription status :', quote_ctx.query_subscription())  # 查詢初始
ret_sub, err_message = quote_ctx.subscribe(['US.AAPL'], [SubType.QUOTE, SubType.T
# 先訂閱了AAPL全時段 QUOTE 和 TICKER 兩個類型。訂閱成功後 OpenD 將持續收到伺服器的推送
if ret_sub == RET_OK:  # 訂閱成功
    print('subscribe successfully ! current subscription status :', quote_ctx.quer
    time.sleep(60)  # 訂閱之後至少1分鐘才能取消訂閱
    ret_unsub, err_message_unsub = quote_ctx.unsubscribe_all()  # 取消所有訂閱
    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()  # 結束後記得關閉當條連線，防止連線條數用盡
```

- 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,
```

> **介面限制**
>
> - 支援多種實時數據類型的訂閱，參見 SubType ，每隻股票訂閱一個類型佔用一個額度。
> - 訂閱額度規則請參見 訂閱額度 & 歷史 K 線額度。
> - 至少訂閱一分鐘才可以反訂閱。
> - 由於港股 SF 行情擺盤數據量較大，為保證 SF 行情的速度和 OpenD 的處理效能，目前 SF 權限用戶僅限同時訂閱 50 只證券類產品（含 hkex 的正股、窩輪、牛熊）的擺盤、經紀隊列，剩餘訂閱額度仍可用於訂閱其他類型，如：逐筆，買賣經紀等。
> - 港股期權期貨在 LV1 權限下，不支援訂閱逐筆類型。

獲取訂閱狀態

`query_subscription(is_all_conn=True)`

- **介紹**

  獲取訂閱資訊

- **參數**

參數	類型	說明
is_all_conn	bool	是否返回所有連線的訂閱狀態

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	dict	當 ret == RET_OK，返回訂閱資訊數據
	str	當 ret != RET_OK，返回錯誤描述

  - 訂閱資訊數據字典格式如下：

```
{
    'total_used': 4,     # 所有連線已使用的訂閱額度
    'own_used': 0,       # 當前連線已使用的訂閱額度
    'remain': 496,       #  剩餘的訂閱額度
    'sub_list':          #  每種訂閱類型對應的股票列表
    {
        '訂閱的類型': 該訂閱類型下所有已訂閱股票列表,
        …
    }
}
```

- Example

```python
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() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

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

實時報價回呼

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  實時報價回呼，非同步處理已訂閱股票的實時報價推送。
  在收到實時報價數據推送後會回呼到該函數，您需要在衍生類別中覆寫 on_recv_rsp。

- **參數**

參數	類型	說明
rsp_pb	Qot_UpdateBasicQot_pb2.Response	衍生類別中不需要直接處理該參數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回報價數據
	str	當 ret != RET_OK，返回錯誤描述

  - 報價數據格式如下：

欄位	類型	說明
code	str	股票代碼
data_date	str	日期
data_time	str	當前價更新時間
last_price	float	最新價格
open_price	float	今日開盤價

欄位	類型	說明
high_price	float	最高價格
low_price	float	最低價格
prev_close_price	float	昨收盤價格
volume	int	成交數量
turnover	float	成交金額
turnover_rate	float	換手率
amplitude	int	振幅
suspension	bool	是否停牌
listing_date	str	上市日期
price_spread	float	當前向上的價差
dark_status	DarkStatus	暗盤交易狀態
sec_status	SecurityStatus	股票狀態
strike_price	float	行權價
contract_size	float	每份合約數
open_interest	int	未平倉合約數
implied_volatility	float	隱含波動率
premium	float	溢價
delta	float	希臘值 Delta
gamma	float	希臘值 Gamma
vega	float	希臘值 Vega

欄位	類型	說明
theta	float	希臘值 Theta
rho	float	希臘值 Rho
index_option_type	IndexOptionType	指數期權類型
net_open_interest	int	淨未平倉合約數
expiry_date_distance	int	距離到期日天數
contract_nominal_value	float	合約名義金額
owner_lot_multiplier	float	相等正股手數
option_area_type	OptionAreaType	期權類型（按行權時間）
contract_multiplier	float	合約乘數
pre_price	float	盤前價格
pre_high_price	float	盤前最高價
pre_low_price	float	盤前最低價
pre_volume	int	盤前成交量
pre_turnover	float	盤前成交額
pre_change_val	float	盤前漲跌額
pre_change_rate	float	盤前漲跌幅
pre_amplitude	float	盤前振幅
after_price	float	盤後價格
after_high_price	float	盤後最高價
after_low_price	float	盤後最低價

欄位	類型	說明
after_volume	int	盤後成交量
after_turnover	float	盤後成交額
after_change_val	float	盤後漲跌額
after_change_rate	float	盤後漲跌幅
after_amplitude	float	盤後振幅
overnight_price	float	夜盤價格
overnight_high_price	float	夜盤最高價
overnight_low_price	float	夜盤最低價
overnight_volume	int	夜盤成交量
overnight_turnover	float	夜盤成交額
overnight_change_val	float	夜盤漲跌額
overnight_change_rate	float	夜盤漲跌幅
overnight_amplitude	float	夜盤振幅
last_settle_price	float	昨結
position	float	持倉量
position_change	float	日增倉

- **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 自己的處理邏輯
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = StockQuoteTest()
quote_ctx.set_handler(handler)  # 設定實時報價回呼
ret, data = quote_ctx.subscribe(['US.AAPL'], [SubType.QUOTE])  # 訂閱實時報價類型
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
time.sleep(15)  #  設定腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()   # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

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

> **提示**
>
> - 此介面提供了持續獲取推送數據的功能，如需一次性獲取實時數據，請參考 獲取實時報價 介面
> - 獲取實時數據 和 實時數據回呼 的差別，請參考 如何透過訂閱介面獲取實時行情？

實時擺盤迴調

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  實時擺盤迴調，非同步處理已訂閱股票的實時擺盤推送。 在收到實時擺盤資料推送後會回調到該函數，您需要在衍生類別中覆寫 on_recv_rsp。

- **參數**

參數	類型	說明
rsp_pb	Qot_UpdateOrderBook_pb2.Response	衍生類別中不需要直接處理該參數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	dict	當 ret == RET_OK，返回擺盤資料
	str	當 ret != RET_OK，返回錯誤描述

  - 擺盤資料格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
svr_recv_time_bid	str	moomoo 伺服器從交易所收到買盤資料的時間

欄位	類型	說明
svr_recv_time_ask	str	moomoo 伺服器從交易所收到賣盤資料的時間
Bid	list	每個元祖包含如下資訊：委託價格，委託數量，委託訂單數，委託訂單明細
Ask	list	每個元祖包含如下資訊：委託價格，委託數量，委託訂單數，委託訂單明細

其中，Bid 和 Ask 字段的結構如下：

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

- Example

```python
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 自己的處理邏輯
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = OrderBookTest()
quote_ctx.set_handler(handler)  # 設定實時擺盤迴調
ret, data = quote_ctx.subscribe(['US.AAPL'], [SubType.ORDER_BOOK])  # 訂閱買賣擺盤
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
time.sleep(15)  #  設定腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()  # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

```
1    OrderBookTest  {'code': 'US.AAPL', 'name': '蘋果', 'svr_recv_time_bid': '', 'svr_
```

提示

- 此介面提供了持續獲取推送資料的功能，如需一次性獲取實時資料，請參考 獲取實時擺盤 介面
- 獲取實時資料 和 實時資料回調 的差別，請參考 如何透過訂閱介面獲取實時行情？
- 美股市場實時擺盤迴調，會持續推送當前交易時段的實時擺盤，無需設定時段。

實時 K 線回呼

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  實時 K 線回呼，非同步處理已訂閱股票的實時 K 線推送。

  在收到實時 K 線數據推送後會回呼到該函數，您需要在衍生類別中覆寫 on_recv_rsp。

- **參數**

參數	類型	說明
rsp_pb	Qot_UpdateKL_pb2.Response	衍生類別中不需要直接處理該參數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回 K 線數據數據
	str	當 ret != RET_OK，返回錯誤描述

  - K 線數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
time_key	str	時間
open	float	開盤價

欄位	類型	說明
close	float	收盤價
high	float	最高價
low	float	最低價
volume	int	成交量
turnover	float	成交額
pe_ratio	float	市盈率
turnover_rate	float	換手率
last_close	float	昨收價
k_type	KLType	K 線類型

- 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 自己的處理邏輯
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = CurKlineTest()
quote_ctx.set_handler(handler)  # 設定實時K線回呼
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)  # 設定腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()  # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- **Output**

```
CurKlineTest        code name            time_key    open   close    high    low
0  US.AAPL   蘋果  2025-04-07 05:15:00  180.39  180.26  180.46  180.2    1322  23
```

> **提示**
>
> - 此介面提供了持續獲取推送數據的功能，如需一次性獲取實時數據，請參考 獲取實時 K 線 介面
> - 獲取實時數據 和 實時數據回呼 的差別，請參考 如何透過訂閱介面獲取實時行情？
> - **期權**，僅提供日K, 1分K，5分K，15分K，60分K。

實時分時回呼

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  實時分時回呼，非同步處理已訂閱股票的實時分時推送。
  在收到實時分時數據推送後會回呼到該函數，您需要在衍生類別中覆寫 on_recv_rsp。

- **參數**

參數	類型	說明
rsp_pb	Qot_UpdateRT_pb2.Response	衍生類別中不需要直接處理該參數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回分時數據
	str	當 ret != RET_OK，返回錯誤描述

  - 分時數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
time	str	時間
is_blank	bool	數據狀態
opened_mins	int	零點到當前多少分鐘

欄位	類型	說明
cur_price	float	當前價格
last_close	float	昨天收盤的價格
avg_price	float	平均價格
volume	float	成交量
turnover	float	成交金額

- 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 自己的處理邏輯
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = RTDataTest()
quote_ctx.set_handler(handler)  # 設定實時分時推送回呼
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)  # 設定腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()   # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

```
RTDataTest        code name                time  is_blank  opened_mins  cur_pric
0  US.AAPL   蘋果  2025-04-07 05:24:00     False          324     179.53      188
```

提示

- 此介面提供了持續獲取推送數據的功能，如需一次性獲取實時數據，請參考 獲取實時分時 介面
- 獲取實時數據 和 實時數據回呼 的差別，請參考 如何透過訂閱介面獲取實時行情？

實時逐筆回呼

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  實時逐筆回呼，非同步處理已訂閱股票的實時逐筆推送。
  在收到實時逐筆數據推送後會回呼到該函數，您需要在衍生類別中覆寫 on_recv_rsp。

- **參數**

參數	類型	說明
rsp_pb	Qot_UpdateTicker_pb2.Response	衍生類別中不需要直接處理該參數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回逐筆數據
	str	當 ret != RET_OK，返回錯誤描述

  - 逐筆數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
sequence	int	逐筆序號
time	str	成交時間
price	float	成交價格

欄位	類型	說明
volume	int	成交數量
turnover	float	成交金額
ticker_direction	TickerDirect	逐筆方向
type	TickerType	逐筆類型
push_data_type	PushDataType	數據來源

- 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 自己的處理邏輯
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = TickerTest()
quote_ctx.set_handler(handler)  # 設定實時逐筆推送回呼
ret, data = quote_ctx.subscribe(['US.AAPL'], [SubType.TICKER], session=Session.AL
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
time.sleep(15)  # 設定腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()   # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

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

```
3
```

提示

- 此介面提供了持續獲取推送數據的功能，如需一次性獲取實時數據，請參考 獲取實時逐筆 介面
- 獲取實時數據 和 實時數據回呼 的差別，請參考 如何透過訂閱介面獲取實時行情？
- 行情連線斷開重連後，OpenD 拉取斷開期間，距離當前最近的（最多 50 根）逐筆數據並推送，可透過逐筆推送類型欄位區分

實時經紀隊列回呼

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  實時經紀隊列回呼，非同步處理已訂閱股票的實時經紀隊列推送。
  在收到實時經紀隊列數據推送後會回呼到該函數，您需要在衍生類別中覆寫 on_recv_rsp。

- **參數**

參數	類型	說明
rsp_pb	Qot_UpdateBroker_pb2.Response	衍生類別中不需要直接處理該參數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	tuple	當 ret == RET_OK，返回經紀隊列數據
	str	當 ret != RET_OK，返回錯誤描述

  - 經紀隊列元組內容如下：

欄位	類型	說明
stock_code	str	股票
bid_frame_table	pd.DataFrame	買盤數據
ask_frame_table	pd.DataFrame	賣盤數據

    - bid_frame_table 格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
bid_broker_id	int	經紀買盤 ID
bid_broker_name	str	經紀買盤名稱
bid_broker_pos	int	經紀檔位
order_id	int	交易所訂單 ID
order_volume	int	單筆委託數量

- ask_frame_table 格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
ask_broker_id	int	經紀賣盤 ID
ask_broker_name	str	經紀賣盤名稱
ask_broker_pos	int	經紀檔位
order_id	int	交易所訂單 ID
order_volume	int	單筆委託數量

- **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(r
        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)  # 設定實時經紀推送回呼
ret, data = quote_ctx.subscribe(['HK.00700'], [SubType.BROKER]) # 訂閱經紀類型，Op
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
time.sleep(15)  # 設定腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()   # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

```
BrokerTest: stock: HK.00700 data: [        code   name  bid_broker_id bid_broker_n
0   HK.00700  騰訊控股           5338          J.P.摩根               1       N/A
..       ...   ...            ...             ...             ...       ...
36  HK.00700  騰訊控股           8305  富途證券國際(香港)有限公司                4

[37 rows x 7 columns],         code   name  ask_broker_id ask_broker_name  ask_bro
0   HK.00700  騰訊控股           1179  華泰金融控股(香港)有限公司                1
..       ...   ...            ...             ...             ...       ...
39  HK.00700  騰訊控股           6996       中國投資信息有限公司               1

[40 rows x 7 columns]]
```

提示

- 此介面提供了持續獲取推送數據的功能，如需一次性獲取實時數據，請參考 獲取實時經紀隊列 介面
- 獲取實時數據 和 實時數據回呼 的差別，請參考 如何通過訂閱介面獲取實時行情？
- 港股 LV1 權限下，不支援獲取經紀隊列數據

獲取快照

`get_market_snapshot(code_list)`

- **介紹**

  獲取快照數據

- **參數**

參數	類型	說明
code_list	list	股票代碼列表

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回股票快照數據
	str	當 ret != RET_OK，返回錯誤描述

  - 股票快照數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
update_time	str	當前價更新時間
last_price	float	最新價格
open_price	float	今日開盤價

欄位	類型	說明
high_price	float	最高價格
low_price	float	最低價格
prev_close_price	float	昨收盤價格
volume	int	成交數量
turnover	float	成交金額
turnover_rate	float	換手率
suspension	bool	是否停牌
listing_date	str	上市日期
equity_valid	bool	是否正股
issued_shares	int	總股本
total_market_val	float	總市值
net_asset	int	資產淨值
net_profit	int	淨利潤
earning_per_share	float	每股盈利
outstanding_shares	int	流通股本
net_asset_per_share	float	每股淨資產
circular_market_val	float	流通市值
ey_ratio	float	收益率
pe_ratio	float	市盈率
pb_ratio	float	市淨率

欄位	類型	說明
pe_ttm_ratio	float	市盈率 TTM
dividend_ttm	float	股息 TTM，派息
dividend_ratio_ttm	float	股息率 TTM
dividend_lfy	float	股息 LFY，上一年度派息
dividend_lfy_ratio	float	股息率 LFY
stock_owner	str	窩輪所屬正股的代碼或期權的標的股代碼
wrt_valid	bool	是否是窩輪
wrt_conversion_ratio	float	換股比率
wrt_type	WrtType	窩輪類型
wrt_strike_price	float	行使價格
wrt_maturity_date	str	格式化窩輪到期時間
wrt_end_trade	str	格式化窩輪最後交易時間
wrt_leverage	float	槓桿比率
wrt_ipop	float	價內/價外
wrt_break_even_point	float	打和點
wrt_conversion_price	float	換股價
wrt_price_recovery_ratio	float	正股距收回價
wrt_score	float	窩輪綜合評分
wrt_code	str	窩輪對應的正股（此欄位已廢除，修改為

欄位	類型	說明
		stock_owner)
wrt_recovery_price	float	窩輪收回價
wrt_street_vol	float	窩輪街貨量
wrt_issue_vol	float	窩輪發行量
wrt_street_ratio	float	窩輪街貨佔比
wrt_delta	float	窩輪對沖值
wrt_implied_volatility	float	窩輪引伸波幅
wrt_premium	float	窩輪溢價
wrt_upper_strike_price	float	上限價
wrt_lower_strike_price	float	下限價
wrt_inline_price_status	PriceType	界內界外
wrt_issuer_code	str	發行人代碼
option_valid	bool	是否是期權
option_type	OptionType	期權類型
strike_time	str	期權行權日
option_strike_price	float	行權價
option_contract_size	float	每份合約數
option_open_interest	int	總未平倉合約數
option_implied_volatility	float	隱含波動率
option_premium	float	溢價

欄位	類型	說明
option_delta	float	希臘值 Delta
option_gamma	float	希臘值 Gamma
option_vega	float	希臘值 Vega
option_theta	float	希臘值 Theta
option_rho	float	希臘值 Rho
index_option_type	IndexOptionType	指數期權類型
option_net_open_interest	int	淨未平倉合約數
option_expiry_date_distance	int	距離到期日天數
option_contract_nominal_value	float	合約名義金額
option_owner_lot_multiplier	float	相等正股手數
option_area_type	OptionAreaType	期權類型（按行權時間）
option_contract_multiplier	float	合約乘數
plate_valid	bool	是否為板塊類型
plate_raise_count	int	板塊類型上漲支數
plate_fall_count	int	板塊類型下跌支數
plate_equal_count	int	板塊類型平盤支數
index_valid	bool	是否有指數類型
index_raise_count	int	指數類型上漲支數
index_fall_count	int	指數類型下跌支數
index_equal_count	int	指數類型平盤支數

欄位	類型	說明
lot_size	int	每手股數，股票期權表示每份合約的股數，期貨表示合約乘數
price_spread	float	當前向上的擺盤價差
ask_price	float	賣價
bid_price	float	買價
ask_vol	float	賣量
bid_vol	float	買量
enable_margin	bool	是否可融資（已廢棄）
mortgage_ratio	float	股票抵押率（已廢棄）
long_margin_initial_ratio	float	融資初始保證金率（已廢棄）
enable_short_sell	bool	是否可賣空（已廢棄）
short_sell_rate	float	賣空參考利率（已廢棄）
short_available_volume	int	剩餘可賣空數量（已廢棄）
short_margin_initial_ratio	float	賣空（融券）初始保證金率（已廢棄）
sec_status	SecurityStatus	股票狀態
amplitude	float	振幅
avg_price	float	平均價

欄位	類型	說明
bid_ask_ratio	float	委比
volume_ratio	float	量比
highest52weeks_price	float	52 周最高價
lowest52weeks_price	float	52 周最低價
highest_history_price	float	歷史最高價
lowest_history_price	float	歷史最低價
pre_price	float	盤前價格
pre_high_price	float	盤前最高價
pre_low_price	float	盤前最低價
pre_volume	int	盤前成交量
pre_turnover	float	盤前成交額
pre_change_val	float	盤前漲跌額
pre_change_rate	float	盤前漲跌幅
pre_amplitude	float	盤前振幅
after_price	float	盤後價格
after_high_price	float	盤後最高價
after_low_price	float	盤後最低價
after_volume	int	盤後成交量
after_turnover	float	盤後成交額
after_change_val	float	盤後漲跌額

欄位	類型	說明
after_change_rate	float	盤後漲跌幅
after_amplitude	float	盤後振幅
overnight_price	float	夜盤價格
overnight_high_price	float	夜盤最高價
overnight_low_price	float	夜盤最低價
overnight_volume	int	夜盤成交量
overnight_turnover	float	夜盤成交額
overnight_change_val	float	夜盤漲跌額
overnight_change_rate	float	夜盤漲跌幅
overnight_amplitude	float	夜盤振幅
future_valid	bool	是否期貨
future_last_settle_price	float	昨結
future_position	float	持倉量
future_position_change	float	日增倉
future_main_contract	bool	是否主連合約
future_last_trade_time	str	最後交易時間
trust_valid	bool	是否基金
trust_dividend_yield	float	股息率
trust_aum	float	資產規模
trust_outstanding_units	int	總發行量

欄位	類型	說明
trust_netAssetValue	float	單位淨值
trust_premium	float	溢價
trust_assetClass	AssetClass	資產類別

- 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])    # 取第一條的股票代碼
    print(data['code'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
code  name             update_time  last_price  open_price  high_price  low_pric
0  HK.00700  騰訊控股      2025-04-07 16:09:07      435.40      441.80      462.4
1   US.AAPL    蘋果  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

   trust_outstanding_units  trust_netAssetValue  trust_premium trust_assetClass p
0                      NaN                  NaN            NaN              N/A
1                      NaN                  NaN            NaN              N/A
HK.00700
['HK.00700', 'US.AAPL']
```

> 介面限制

- 每 30 秒內最多請求 60 次快照。
- 每次請求，介面參數 **股票代碼列表** 支援傳入的標的數量上限是 400 個。

獲取即時報價

`get_stock_quote(code_list)`

- **介紹**

  獲取已訂閱股票的即時報價，必須要先訂閱。

- **參數**

參數	類型	說明
code_list	list	股票代碼列表

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回報價數據
	str	當 ret != RET_OK，返回錯誤描述

  - 報價數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
data_date	str	日期
data_time	str	當前價更新時間
last_price	float	最新價格

欄位	類型	說明
open_price	float	今日開盤價
high_price	float	最高價格
low_price	float	最低價格
prev_close_price	float	昨收盤價格
volume	int	成交數量
turnover	float	成交金額
turnover_rate	float	換手率
amplitude	int	振幅
suspension	bool	是否停牌
listing_date	str	上市日期
price_spread	float	當前向上的價差
dark_status	DarkStatus	暗盤交易狀態
sec_status	SecurityStatus	股票狀態
strike_price	float	行權價
contract_size	float	每份合約數
open_interest	int	未平倉合約數
implied_volatility	float	隱含波動率
premium	float	溢價
delta	float	希臘值 Delta
gamma	float	希臘值 Gamma

欄位	類型	說明
vega	float	希臘值 Vega
theta	float	希臘值 Theta
rho	float	希臘值 Rho
index_option_type	IndexOptionType	指數期權類型
net_open_interest	int	淨未平倉合約數
expiry_date_distance	int	距離到期日天數
contract_nominal_value	float	合約名義金額
owner_lot_multiplier	float	相等正股手數
option_area_type	OptionAreaType	期權類型（按行權時間）
contract_multiplier	float	合約乘數
pre_price	float	盤前價格
pre_high_price	float	盤前最高價
pre_low_price	float	盤前最低價
pre_volume	int	盤前成交量
pre_turnover	float	盤前成交額
pre_change_val	float	盤前漲跌額
pre_change_rate	float	盤前漲跌幅
pre_amplitude	float	盤前振幅
after_price	float	盤後價格
after_high_price	float	盤後最高價

欄位	類型	說明
after_low_price	float	盤後最低價
after_volume	int	盤後成交量
after_turnover	float	盤後成交額
after_change_val	float	盤後漲跌額
after_change_rate	float	盤後漲跌幅
after_amplitude	float	盤後振幅
overnight_price	float	夜盤價格
overnight_high_price	float	夜盤最高價
overnight_low_price	float	夜盤最低價
overnight_volume	int	夜盤成交量
overnight_turnover	float	夜盤成交額
overnight_change_val	float	夜盤漲跌額
overnight_change_rate	float	夜盤漲跌幅
overnight_amplitude	float	夜盤振幅
last_settle_price	float	昨結
position	float	持倉量
position_change	float	日增倉

- **Example**

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

```

```python
ret_sub, err_message = quote_ctx.subscribe(['US.AAPL'], [SubType.QUOTE], subscri
# 先訂閱 K 線類型。訂閱成功後 OpenD 將持續收到伺服器的推送，False 代表暫時不需要推送給
if ret_sub == RET_OK:  # 訂閱成功
    ret, data = quote_ctx.get_stock_quote(['US.AAPL'])  # 獲取訂閱股票報價的即時數
    if ret == RET_OK:
        print(data)
        print(data['code'][0])    # 取第一條的股票代碼
        print(data['code'].values.tolist())   # 轉為 list
    else:
        print('error:', data)
else:
    print('subscription failed', err_message)
quote_ctx.close()  # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

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

提示

- 此介面提供了一次性獲取即時數據的功能，如需持續獲取推送數據，請參考 即時報價回呼 介面
- 獲取即時數據 和 即時數據回呼 的差別，請參考 如何通過訂閱介面獲取即時行情？

獲取實時擺盤

`get_order_book(code, num=10)`

- **介紹**

  獲取已訂閱股票的實時擺盤，必須要先訂閱。

- **參數**

參數	類型	說明
code	str	股票代碼
name	str	股票名稱
num	int	請求擺盤檔數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	dict	當 ret == RET_OK，返回擺盤數據
	str	當 ret != RET_OK，返回錯誤描述

  - 擺盤數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
svr_recv_time_bid	str	富途伺服器從交易所收到買盤數據的時間

欄位	類型	說明
svr_recv_time_ask	str	富途伺服器從交易所收到賣盤數據的時間
Bid	list	每個元祖包含如下資訊：委託價格，委託數量，委託訂單數，委託訂單明細
Ask	list	每個元祖包含如下資訊：委託價格，委託數量，委託訂單數，委託訂單明細

其中，Bid 和 Ask 欄位的結構如下：

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

- Example

```python
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=
# 先訂閱買賣擺盤類型。訂閱成功後 OpenD 將持續收到伺服器的推送，False 代表暫時不需要推送
if ret_sub == RET_OK:  # 訂閱成功
    ret, data = quote_ctx.get_order_book('US.AAPL', num=3)  # 獲取一次 3 檔實時擺盤
    if ret == RET_OK:
        print(data)
    else:
        print('error:', data)
else:
    print('subscription failed')
quote_ctx.close()  # 關閉當條連線，OpenD 會在 1 分鐘後自動取消相應股票相應類型的訂閱
```

- Output

```
{'code': 'US.AAPL', 'name': '蘋果', 'svr_recv_time_bid': '2025-04-07 05:39:20.352
```

> **介面限制**

- moomoo 伺服器從交易所收到數據的時間欄位，僅支援A股正股、港股正股、ETFs、窩輪、牛熊，且僅開盤時間才有此數據。
- moomoo 伺服器從交易所收到數據的時間欄位，部分情況下接收時間可能為零，例如：伺服器重啓或第一次推送的快取數據。

提示

- 此介面提供了一次性獲取實時數據的功能，如需持續獲取推送數據，請參考 實時擺盤迴調 介面
- 獲取實時數據 和 實時數據回調 的差別，請參考 如何透過訂閱介面獲取實時行情？
- 美股市場，會返回當前交易時段的實時擺盤數據，無需設定時段。

獲取實時 K 線

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

- **介紹**

  獲取已訂閱股票的實時 K 線數據，必須要先訂閱。

- **參數**

參數	類型	說明
code	str	股票代碼
name	str	股票名稱
num	int	K 線數據個數
ktype	KLType	K 線類型
autype	AuType	復權類型

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回 K 線數據數據
	str	當 ret != RET_OK，返回錯誤描述

  - K 線數據格式如下：

欄位	類型	說明
code	str	股票代碼

欄位	類型	說明
name	str	股票名稱
time_key	str	時間
open	float	開盤價
close	float	收盤價
high	float	最高價
low	float	最低價
volume	int	成交量
turnover	float	成交額
pe_ratio	float	市盈率
turnover_rate	float	換手率
last_close	float	昨收價

- **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
# 先訂閱 K 線類型。訂閱成功後 OpenD 將持續收到伺服器的推送，False 代表暫時不需要推送給
if ret_sub == RET_OK:  # 訂閱成功
    ret, data = quote_ctx.get_cur_kline('US.AAPL', 2, KLType.K_DAY, AuType.QFQ)
    if ret == RET_OK:
        print(data)
        print(data['turnover_rate'][0])   # 取第一條的換手率
        print(data['turnover_rate'].values.tolist())   # 轉為 list
    else:
        print('error:', data)
else:
```

```
    print('subscription failed', err_message)
quote_ctx.close()  # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

```
code name             time_key    open   close    high     low      volume      tu
0  US.AAPL   蘋果  2025-04-03 00:00:00  205.54  203.19  207.49  201.25  103419006
1  US.AAPL   蘋果  2025-04-04 00:00:00  193.89  188.38  199.88  187.34  125910913
0.00689
[0.00689, 0.00838]
```

介面限制

- 此介面為獲取實時 K 線介面，最多能獲取最近的 1000 根。如需獲取歷史 K 線，請參考 獲取歷史 K 線
- 市盈率和換手率欄位，只有日 K 及以上週期的正股才有數據
- **期權**，僅提供日K, 1分K，5分K，15分K，60分K。

提示

- 此介面提供了一次性獲取實時數據的功能，如需持續獲取推送數據，請參考 實時 K 線回呼 介面
- 獲取實時數據 和 實時數據回呼 的差別，請參考 如何通過訂閱介面獲取實時行情？

獲取即時分時

`get_rt_data(code)`

- **介紹**

  獲取已訂閱股票的即時分時資料，必須要先訂閱。

- **參數**

參數	類型	說明
code	str	股票

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回分時資料
	str	當 ret != RET_OK，返回錯誤描述

  - 分時資料格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
time	str	時間
is_blank	bool	資料狀態
opened_mins	int	零點到當前多少分鐘

欄位	類型	說明
cur_price	float	當前價格
last_close	float	昨天收盤的價格
avg_price	float	平均價格
volume	float	成交量
turnover	float	成交金額

- 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
# 先訂閱分時資料類型。訂閱成功後 OpenD 將持續收到伺服器的推送，False 代表暫時不需要推送
if ret_sub == RET_OK:   # 訂閱成功
    ret, data = quote_ctx.get_rt_data('US.AAPL')   # 獲取一次分時資料
    if ret == RET_OK:
        print(data)
    else:
        print('error:', data)
else:
    print('subscription failed', err_message)
quote_ctx.close()   # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

```
code  name                time  is_blank  opened_mins  cur_price  last_close   a
0    US.AAPL   蘋果  2025-04-06 20:01:00     False         1201     183.00      1
..       ...   ...                  ...       ...          ...        ...
586  US.AAPL   蘋果  2025-04-07 05:47:00     False          347     181.26      1

[587 rows x 10 columns]
```

> 提示

- 此介面提供了一次性獲取即時資料的功能，如需持續獲取推送資料，請參考 即時分時回呼 介面
- 獲取即時資料 和 即時資料回呼 的差別，請參考 如何通過訂閱介面獲取即時行情？

獲取實時逐筆

`get_rt_ticker(code, num=500)`

- **介紹**

  獲取已訂閱股票的實時逐筆數據，必須要先訂閱。

- **參數**

參數	類型	說明
code	str	股票代碼
num	int	最近逐筆個數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回逐筆數據
	str	當 ret != RET_OK，返回錯誤描述

  - 逐筆數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
sequence	int	逐筆序號
time	str	成交時間

欄位	類型	說明
price	float	成交價格
volume	int	成交數量
turnover	float	成交金額
ticker_direction	TickerDirect	逐筆方向
type	TickerType	逐筆類型

- 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
# 先訂閱逐筆類型。訂閱成功後 OpenD 將持續收到伺服器的推送，False 代表暫時不需要推送給腳
if ret_sub == RET_OK:  # 訂閱成功
    ret, data = quote_ctx.get_rt_ticker('US.AAPL', 2)  # 獲取美股AAPL最近2個逐筆
    if ret == RET_OK:
        print(data)
        print(data['turnover'][0])   # 取第一條的成交金額
        print(data['turnover'].values.tolist())   # 轉為 list
    else:
        print('error:', data)
else:
    print('subscription failed', err_message)
quote_ctx.close()  # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

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

介面限制

- 最多能獲取最近 1000 個逐筆數據，更多歷史逐筆數據暫未提供
- 港股期權期貨在 LV1 權限下，不支援獲取逐筆

提示

- 此介面提供了一次性獲取實時數據的功能，如需持續獲取推送數據，請參考 實時逐筆回呼 介面
- 獲取實時數據 和 實時數據回呼 的差別，請參考 如何通過訂閱介面獲取實時行情？

獲取實時經紀佇列

`get_broker_queue(code)`

- **介紹**

  獲取已訂閱股票的實時經紀佇列數據，必須要先訂閱。

- **參數**

參數	類型	說明
code	str	股票代號

- **傳回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
bid_frame_table	pd.DataFrame	當 ret == RET_OK，bid_frame_table 傳回買盤經紀佇列數據
	str	當 ret != RET_OK，bid_frame_table 傳回錯誤描述
ask_frame_table	pd.DataFrame	當 ret == RET_OK，ask_frame_table 傳回賣盤經紀佇列數據
	str	當 ret != RET_OK，ask_frame_table 傳回錯誤描述

  - 買盤經紀佇列格式如下：

欄位	類型	說明
code	str	股票代號

欄位	類型	說明
name	str	股票名稱
bid_broker_id	int	經紀買盤 ID
bid_broker_name	str	經紀買盤名稱
bid_broker_pos	int	經紀檔位
order_id	int	交易所訂單 ID
order_volume	int	單筆委託數量

- 賣盤經紀佇列格式如下：

欄位	類型	說明
code	str	股票代號
name	str	股票名稱
ask_broker_id	int	經紀賣盤 ID
ask_broker_name	str	經紀賣盤名稱
ask_broker_pos	int	經紀檔位
order_id	int	交易所訂單 ID
order_volume	int	單筆委託數量

- **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
# 先訂閱經紀佇列類型。訂閱成功後 OpenD 將持續收到伺服器的推送，False 代表暫時不需要推送
if ret_sub == RET_OK:   # 訂閱成功
    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()   # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

```
        code  name  bid_broker_id bid_broker_name  bid_broker_pos order_id order_
0   HK.00700  騰訊控股           5338          J.P.摩根               1      N/A
..       ...   ...             ...             ...             ...      ...
36  HK.00700  騰訊控股           8305  富途證券國際(香港)有限公司                4

[37 rows x 7 columns]
```

> **提示**
>
> - 此介面提供了一次性獲取實時數據的功能，如需持續獲取推送數據，請參考 實時經紀佇列回呼 介面
> - 獲取實時數據 和 實時數據回呼 的差別，請參考 如何通過訂閱介面獲取實時行情？
> - 港股 LV1 權限下，不支援獲取經紀佇列數據

獲取標的市場狀態

`get_market_state(code_list)`

- **介紹**

  獲取指定標的的市場狀態

- **參數**

參數	類型	說明
code_list	list	需要查詢市場狀態的股票代碼列表

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回市場狀態數據
	str	當 ret != RET_OK，返回錯誤描述

  - 市場狀態數據

欄位	類型	說明
code	str	股票代碼
stock_name	str	股票名稱
market_state	MarketState	市場狀態

- **Example**

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

```python

ret, data = quote_ctx.get_market_state(['SZ.000001', 'HK.00700'])
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **Output**

```
    code          stock_name   market_state
0  SZ.000001     平安銀行      AFTERNOON
1  HK.00700      騰訊控股      AFTERNOON
```

> **介面限制**
>
> - 每 30 秒內最多請求 10 次獲取標的市場狀態介面。
> - 每次請求的股票代碼個數上限為 400 個。

獲取資金流向

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

- **介紹**

  獲取個股資金流向

- **參數**

參數	類型	說明
stock_code	str	股票代號
period_type	PeriodType	週期類型
start	str	開始時間
end	str	結束時間

  - start 和 end 的組合如下

start 類型	end 類型	說明
str	str	start 和 end 分別為指定的日期
None	str	start 為 end 往前 365 天
str	None	end 為 start 往後 365 天
None	None	end 為 當前日期，start 往前 365 天

- **傳回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果

data	pd.DataFrame	當 ret == RET_OK，傳回資金流向數據
	str	當 ret != RET_OK，傳回錯誤描述

- 資金流向數據格式如下：

欄位	類型	說明
in_flow	float	整體淨流入
main_in_flow	float	主力大單淨流入
super_in_flow	float	特大單淨流入
big_in_flow	float	大單淨流入
mid_in_flow	float	中單淨流入
sml_in_flow	float	小單淨流入
capital_flow_item_time	str	開始時間
last_valid_time	str	數據最後有效時間

- **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])    # 取第一條的淨流入的資金額度
    print(data['in_flow'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **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
```

介面限制

- 每 30 秒內最多請求 30 次獲取資金流向介面。
- 僅支援正股、窩輪和基金。
- 歷史週期（日、月、年）僅提供最近 1 年數據；實時週期僅提供最新一天的數據。
- 傳回數據只包括盤中數據，不包含盤前盤後數據。

獲取資金分佈

`get_capital_distribution(stock_code)`

- **介紹**

  獲取資金分佈

- **參數**

參數	類型	說明
stock_code	str	股票代號

- **傳回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，傳回股票資金分佈數據
	str	當 ret != RET_OK，傳回錯誤描述

  - 資金分佈數據格式如下：

欄位	類型	說明
capital_in_super	float	流入資金額度，特大單
capital_in_big	float	流入資金額度，大單
capital_in_mid	float	流入資金額度，中單
capital_in_small	float	流入資金額度，小單
capital_out_super	float	流出資金額度，特大單

欄位	類型	說明
capital_out_big	float	流出資金額度，大單
capital_out_mid	float	流出資金額度，中單
capital_out_small	float	流出資金額度，小單
update_time	str	更新時間字串

- 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])    # 取第一條的流入資金額度，大單
    print(data['capital_in_big'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- 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]
```

介面限制

- 每 30 秒內最多請求 30 次獲取資金分佈介面。
- 僅支援正股、窩輪和基金。
- 更多資金分佈介紹，請參考 這裡。

- 傳回數據只包括盤中數據，不包含盤前盤後數據。

獲取股票所屬板塊

`get_owner_plate(code_list)`

- **介紹**

  獲取單支或多支股票的所屬板塊資訊列表

- **參數**

參數	類型	說明
code_list	list	股票代碼列表

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回所屬板塊數據
	str	當 ret != RET_OK，返回錯誤描述

  - 所屬板塊數據格式如下：

欄位	類型	說明
code	str	證券代碼
name	str	股票名稱
plate_code	str	板塊代碼
plate_name	str	板塊名字
plate_type	Plate	板塊類型

- 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])    # 取第一條的股票代碼
    print(data['plate_code'].values.tolist())   # 板塊代碼轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
        code name          plate_code plate_name plate_type
0   HK.00001   長和  HK.HSI Constituent      恒指成份股      OTHER
..        ...  ...                 ...         ...        ...
8   HK.00001   長和           HK.BK1983     香港股票ADR      OTHER

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

介面限制

- 每 30 秒內最多請求 10 次獲取股票所屬板塊介面
- 每次請求的股票列表中，股票個數上限為 200 個
- 僅支援正股和指數

獲取歷史 K 線

`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)`

- **介紹**

  獲取歷史 K 線

- **參數**

參數	類型	說明
code	str	股票代碼
start	str	開始時間
end	str	結束時間
ktype	KLType	K 線類型
autype	AuType	復權類型
fields	KLFields	需返回的欄位列表
max_count	int	本次請求最大返回的 K 線根數
page_req_key	bytes	分頁請求
extended_time	bool	是否允許美股盤前盤後數據

  - start 和 end 的組合如下

Start 類型	End 類型	說明
str	str	start 和 end 分別為指定的日期

Start 類型	End 類型	說明
None	str	start 為 end 往前 365 天
str	None	end 為 start 往後 365 天
None	None	end 為當前日期，start 往前 365 天

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回歷史 K 線數據
	str	當 ret != RET_OK，返回錯誤描述
page_req_key	bytes	下一頁請求的 key

  - 歷史 K 線數據格式如下:

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
time_key	str	K 線時間
open	float	開盤價
close	float	收盤價
high	float	最高價
low	float	最低價
pe_ratio	float	市盈率

欄位	類型	說明
turnover_rate	float	換手率
volume	int	成交量
turnover	float	成交額
change_rate	float	漲跌幅
last_close	float	昨收價

- 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])    # 取第一條的股票代碼
    print(data['close'].values.tolist())   # 第一頁收盤價轉為 list
else:
    print('error:', data)
while page_req_key != None:  # 請求後面的所有結果
    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() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
code  name            time_key       open      close       high        low  pe_
0  US.AAPL   蘋果  2019-09-11 00:00:00  52.631194  53.963447  53.992409  52.54913
..       ...   ...                  ...        ...        ...        ...
4  US.AAPL   蘋果  2019-09-17 00:00:00  53.087346  53.265945  53.294907  52.88461

[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   蘋果  2019-09-18 00:00:00  53.352831  53.76554  53.784847  52.961844
All pages are finished!
```

介面限制

- 分 K 提供最近 8 年數據，日 K 提供最近 20 年的數據，日 K 以上不限制。
- 我們會根據您賬戶的資產和交易的情況，下發歷史 K 線額度。因此，30 天內您只能獲取有限只股票的歷史 K 線數據。具體規則參見 訂閱額度 & 歷史 K 線額度。您當日消耗的歷史 K 線額度，會在 30 天后自動釋放。
- 每 30 秒內最多請求 60 次歷史 K 線介面。注意：如果您是分頁獲取數據，此限頻規則僅適用於每隻股票的首頁，後續頁請求不受限頻規則的限制。
- **換手率**，僅提供日 K 及以上級別。
- **期權**，僅提供日K, 1分K，5分K，15分K，60分K。
- 美股 **盤前、盤後、夜盤 K 線**，僅支援 60 分鐘及以下級別。由於美股盤前盤後和夜盤時段為非常規交易時段，此時段的 K 線數據可能不足 2 年。
- 美股的 **成交額**，僅提供 2015-10-12 之後的數據。

獲取復權因子

`get_rehab(code)`

- **介紹**

  獲取股票的復權因子

- **參數**

參數	類型	說明
code	str	股票代碼

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回復權數據
	str	當 ret != RET_OK，返回錯誤描述

  - 復權數據格式如下：

欄位	類型	說明
ex_div_date	str	除權除息日
split_base	float	拆股分子
split_ert	float	拆股分母
join_base	float	合股分子
join_ert	float	合股分母

欄位	類型	說明
split_ratio	float	拆合股比例
per_cash_div	float	每股派現
bonus_base	float	送股分子
bonus_ert	float	送股分母
per_share_div_ratio	float	送股比例
transfer_base	float	轉增股分子
transfer_ert	float	轉增股分母
per_share_trans_ratio	float	轉增股比例
allot_base	float	配股分子
allot_ert	float	配股分母
allotment_ratio	float	配股比例
allotment_price	float	配股價
add_base	float	增發股分子
add_ert	float	增發股分母
stk_spo_ratio	float	增發比例
stk_spo_price	float	增發價格
spin_off_base	float	分立分子
spin_off_ert	float	分立分母
spin_off_ratio	float	分立比例
forward_adj_factorA	float	前復權因子 A

欄位	類型	說明
forward_adj_factorB	float	前復權因子 B
backward_adj_factorA	float	後復權因子 A
backward_adj_factorB	float	後復權因子 B

前復權價格 = 不復權價格 × 前復權因子 A + 前復權因子 B
後復權價格 = 不復權價格 × 後復權因子 A + 後復權因子 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])    # 取第一條的除權除息日
    print(data['ex_div_date'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- 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-
```

> **介面限制**
>
> - 每 30 秒內最多請求 60 次獲取復權因子介面。

獲取期權鏈到期日

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

- **介紹**

  透過標的股票，查詢期權鏈的所有到期日。如需獲取完整期權鏈，請配合 獲取期權鏈 介面使用。

- **參數**

參數	類型	說明
code	str	標的股票代碼
index_option_type	IndexOptionType	指數期權類型

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回期權鏈到期日相關數據
	str	當 ret != RET_OK，返回錯誤描述

  - 期權鏈到期日數據格式如下：

欄位	類型	說明
strike_time	str	期權鏈行權日
option_expiry_date_distance	int	距離到期日天數
expiration_cycle	ExpirationCycle	交割週期

- **Example**

```python
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())  # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- 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-
```

> **介面限制**
>
> - 每 30 秒內最多請求 60 次獲取期權鏈到期日介面

獲取期權鏈

```
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)
```

- **介紹**

  透過標的股票查詢期權鏈。此介面僅返回期權鏈的靜態資訊，如需獲取報價或擺盤等動態資訊，請用此介面返回的股票代碼，自行 訂閱 所需要的類型。

- **參數**

參數	類型	說明
code	str	標的股票代碼
index_option_type	IndexOptionType	指數期權類型
start	str	開始日期，該日期指到期日
end	str	結束日期（包括這一天），該日期指到期日
option_type	OptionType	期權看漲看跌類型
option_cond_type	OptionCondType	期權價內外類型
data_filter	OptionDataFilter	數據篩選條件

  - start 和 end 的組合如下：

Start 類型	End 類型	說明
str	str	start 和 end 分別為指定的日期
None	str	start 為 end 往前 30 天

Start 類型	End 類型	說明
str	None	end 為 start 往後30天
None	None	start 為當前日期，end 往後 30 天

- OptionDataFilter 欄位如下

欄位	類型	說明
implied_volatility_min	float	隱含波動率過濾起點
implied_volatility_max	float	隱含波動率過濾終點
delta_min	float	希臘值 Delta 過濾起點
delta_max	float	希臘值 Delta 過濾終點
gamma_min	float	希臘值 Gamma 過濾起點
gamma_max	float	希臘值 Gamma 過濾終點
vega_min	float	希臘值 Vega 過濾起點
vega_max	float	希臘值 Vega 過濾終點
theta_min	float	希臘值 Theta 過濾起點
theta_max	float	希臘值 Theta 過濾終點
rho_min	float	希臘值 Rho 過濾起點
rho_max	float	希臘值 Rho 過濾終點
net_open_interest_min	float	淨未平倉合約數過濾起點
net_open_interest_max	float	淨未平倉合約數過濾終點
open_interest_min	float	未平倉合約數過濾起點
open_interest_max	float	未平倉合約數過濾終點

欄位	類型	說明
vol_min	float	成交量過濾起點
vol_max	float	成交量過濾終點

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回期權鏈數據
	str	當 ret != RET_OK，返回錯誤描述

  - 期權鏈數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	名字
lot_size	int	每手股數，期權表示每份合約股數
stock_type	SecurityType	股票類型
option_type	OptionType	期權類型
stock_owner	str	標的股
strike_time	str	行權日
strike_price	float	行權價
suspension	bool	是否停牌
stock_id	int	股票 ID

欄位	類型	說明
index_option_type	IndexOptionType	指數期權類型
expiration_cycle	ExpirationCycle	交割週期
option_standard_type	OptionStandardType	期權標準類型
option_settlement_mode	OptionSettlementMode	期權結算方式

- Example

```
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])  # 取第一條的股票代碼
            print(data2['code'].values.tolist())  # 轉為 list
        else:
            print('error:', data2)
        time.sleep(3)
else:
    print('error:', data1)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
                    code                 name  lot_size stock_type option_type s
0     HK.TCH210429C350000   騰訊 210429 350.00 購        100       DRVT        CAL
1     HK.TCH210429P350000   騰訊 210429 350.00 沽        100       DRVT         PU
```

```
2     HK.TCH210429C360000   騰訊 210429 360.00 購       100        DRVT         CAL
3     HK.TCH210429P360000   騰訊 210429 360.00 沽       100        DRVT          PU
4     HK.TCH210429C370000   騰訊 210429 370.00 購       100        DRVT         CAL
5     HK.TCH210429P370000   騰訊 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  騰訊 220330 490.00 購       100        DRVT        CALL
1   HK.TCH220330P490000  騰訊 220330 490.00 沽       100        DRVT         PUT
2   HK.TCH220330C500000  騰訊 220330 500.00 購       100        DRVT        CALL
3   HK.TCH220330P500000  騰訊 220330 500.00 沽       100        DRVT         PUT
4   HK.TCH220330C510000  騰訊 220330 510.00 購       100        DRVT        CALL
5   HK.TCH220330P510000  騰訊 220330 510.00 沽       100        DRVT         PUT
HK.TCH220330C490000
['HK.TCH220330C490000', 'HK.TCH220330P490000', 'HK.TCH220330C500000', 'HK.TCH2203
```

介面限制

- 每 30 秒內最多請求 10 次獲取期權鏈介面
- 傳入的時間跨度上限為 30 天

提示

- 此介面不支援查詢已過期的期權鏈，**結束日期** 參數請輸入今天或未來的日期
- Open interest (OI) 數據每日更新，更新時點取決於具體交易所。美股期權在盤前時段更新，港股期權在盤後更新。

篩選窩輪

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

- **介紹**

  篩選窩輪（僅用於篩選香港市場的窩輪、牛熊證、界內證）

- **參數**

參數	類型	說明
stock_owner	str	所屬正股的股票代碼
req	WarrantRequest	篩選參數組合

  - WarrantRequest 類型欄位說明如下：

欄位	類型	說明
begin	int	數據起始點
num	int	請求數據個數
sort_field	SortField	根據哪個欄位排序
ascend	bool	排序方向
type_list	list	窩輪類型過濾列表
issuer_list	list	發行人過濾列表
maturity_time_min	str	到期日過濾範圍的開始時間
maturity_time_max	str	到期日過濾範圍的結束時間
ipo_period	IpoPeriod	上市時段

欄位	類型	說明
price_type	PriceType	價內/價外
status	WarrantStatus	窩輪狀態
cur_price_min	float	最新價的過濾下限
cur_price_max	float	最新價的過濾上限
strike_price_min	float	行使價的過濾下限
strike_price_max	float	行使價的過濾上限
street_min	float	街貨佔比的過濾下限
street_max	float	街貨佔比的過濾上限
conversion_min	float	換股比率的過濾下限
conversion_max	float	換股比率的過濾上限
vol_min	int	成交量的過濾下限
vol_max	int	成交量的過濾上限
premium_min	float	溢價的過濾下限
premium_max	float	溢價的過濾上限
leverage_ratio_min	float	槓桿比率的過濾下限
leverage_ratio_max	float	槓桿比率的過濾上限
delta_min	float	對沖值的過濾下限
delta_max	float	對沖值的過濾上限
implied_min	float	引伸波幅的過濾下限
implied_max	float	引伸波幅的過濾上限

欄位	類型	說明
recovery_price_min	float	收回價的過濾下限
recovery_price_max	float	收回價的過濾上限
price_recovery_ratio_min	float	正股距收回價的過濾下限
price_recovery_ratio_max	float	正股距收回價的過濾上限

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	tuple	當 ret == RET_OK，返回窩輪數據
	str	當 ret != RET_OK，返回錯誤描述

  - 窩輪數據組成如下：

欄位	類型	說明
warrant_data_list	pd.DataFrame	篩選後的窩輪數據
last_page	bool	是否是最後一頁
all_count	int	篩選結果中的窩輪總數量

    - warrant_data_list 返回的 pd dataframe 數據格式：

欄位	類型	說明
stock	str	窩輪代碼
stock_owner	str	所屬正股
type	WrtType	窩輪類型

欄位	類型	說明
issuer	Issuer	發行人
maturity_time	str	到期日
list_time	str	上市時間
last_trade_time	str	最後交易日
recovery_price	float	收回價
conversion_ratio	float	換股比率
lot_size	int	每手數量
strike_price	float	行使價
last_close_price	float	昨收價
name	str	名稱
cur_price	float	當前價
price_change_val	float	漲跌額
status	WarrantStatus	窩輪狀態
bid_price	float	買入價
ask_price	float	賣出價
bid_vol	int	買量
ask_vol	int	賣量
volume	int	成交量
turnover	float	成交額
score	float	綜合評分

欄位	類型	說明
premium	float	溢價
break_even_point	float	打和點
leverage	float	槓桿比率
ipop	float	價內/價外
price_recovery_ratio	float	正股距收回價
conversion_price	float	換股價
street_rate	float	街貨佔比
street_vol	int	街貨量
amplitude	float	振幅
issue_size	int	發行量
high_price	float	最高價
low_price	float	最低價
implied_volatility	float	引伸波幅
delta	float	對沖值
effective_leverage	float	有效槓桿
upper_strike_price	float	上限價
lower_strike_price	float	下限價
inline_price_status	PriceType	界內界外

- **Example**

```python
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:  # 先判斷介面返回是否正常，再取數據
    warrant_data_list, last_page, all_count = ls
    print(len(warrant_data_list), all_count, warrant_data_list)
    print(warrant_data_list['stock'][0])    # 取第一條的窩輪代碼
    print(warrant_data_list['stock'].values.tolist())   # 轉為 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()  # 所有介面結尾加上這條 close，防止連線條數用盡
```

- Output

```
2 2
    stock        name stock_owner  type issuer maturity_time   list_time last_tra
0   HK.20306  騰訊麥銀零乙購A.C    HK.00700  CALL     MB    2020-12-01  2019-06-27
1   HK.16545  騰訊法興一二購B.C    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  平安瑞銀零乙購A.C    HK.02318    CALL     UB    2020-12-31  2017-12
1    HK.20084  平安中銀零乙購A.C    HK.02318    CALL     BI    2020-12-31  2017-12
```

```
12  ......
13  198  HK.56886  恒指瑞銀三一牛F.C   HK.800000    BULL     UB    2023-01-30  2020-03
14  199  HK.56895  小米瑞銀零乙牛D.C    HK.01810     BULL     UB    2020-12-30  2020-03
15
```

介面限制

- 港股 BMP 權限不支援呼叫此介面
- 每 30 秒內最多請求 60 次篩選窩輪介面
- 每次請求的數據個數上限為 200 個

獲取窩輪和期貨列表

`get_referencestock_list(code, reference_type)`

- **介紹**

  獲取證券的關聯數據，如：獲取正股相關窩輪、獲取期貨相關合約

- **參數**

參數	類型	說明
code	str	證券代碼
reference_type	SecurityReferenceType	要獲得的相關數據

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回證券的關聯數據
	str	當 ret != RET_OK，返回錯誤描述

  - 證券的關聯數據格式如下：

欄位	類型	說明
code	str	證券代碼
lot_size	int	每手股數，期貨表示合約乘數
stock_type	SecurityType	證券類型
stock_name	str	證券名字

欄位	類型	說明
list_time	str	上市時間
wrt_valid	bool	是否是窩輪
wrt_type	WrtType	窩輪類型
wrt_code	str	所屬正股
future_valid	bool	是否是期貨
future_main_contract	bool	是否主連合約
future_last_trade_time	str	最後交易時間

- Example

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

# 獲取正股相關的窩輪
ret, data = quote_ctx.get_referencestock_list('HK.00700', SecurityReferenceType.W
if ret == RET_OK:
    print(data)
    print(data['code'][0])    # 取第一條的股票代碼
    print(data['code'].values.tolist())   # 轉為 list
else:
    print('error:', data)
print('**************************************')
# 港期相關合約
ret, data = quote_ctx.get_referencestock_list('HK.A50main', SecurityReferenceType
if ret == RET_OK:
    print(data)
    print(data['code'][0])    # 取第一條的股票代碼
    print(data['code'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
        code  lot_size stock_type stock_name   list_time  wrt_valid wrt_type  wr
0     HK.24719      1000    WARRANT    騰訊東亞九四沽A  2018-07-20       True
..         ...       ...        ...                ...       ...        ...
1617  HK.63402     10000    WARRANT    騰訊高盛一八牛Y  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      安碩富時 A50 ETF主連(2012)
..         ...       ...        ...                ...       ...        ...
5  HK.A502106      5000     FUTURE      安碩富時 A50 ETF2106               False

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

介面限制

- 每 30 秒內最多請求 10 次獲取證券關聯數據介面
- 當獲取正股相關窩輪時，不受上述限頻限制

獲取期貨合約資料

`get_future_info(code_list)`

- **介紹**

  獲取期貨合約資料

- **參數**

參數	類型	說明
code_list	list	股票代號列表

- **傳回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，傳回期貨合約資料數據
	str	當 ret != RET_OK，傳回錯誤描述

  - 期貨合約資料數據格式如下：

欄位	類型	說明
code	str	股票代號
name	str	股票名稱
owner	str	標的
exchange	str	交易所
type	str	合約類型

欄位	類型	說明
size	float	合約規模
size_unit	str	合約規模單位
price_currency	str	報價貨幣
price_unit	str	報價單位
min_change	float	最小變動
min_change_unit	str	最小變動的單位
trade_time	str	交易時間
time_zone	str	時區
last_trade_time	str	最後交易時間
exchange_format_url	str	交易所規格連結 url
origin_code	str	實際合約代號

- **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])    # 取第一條的股票代號
    print(data['code'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **Output**

```
    code      name        owner exchange  type     size size_unit price_currency p
0  HK.MPImain   內房期貨主連   恒生中國內地地產指數       港交所   股指期貨     50.0
1  HK.HAImain   海通證券期貨主連     HK.06837       港交所   股票期貨  10000.0
HK.MPImain
['HK.MPImain', 'HK.HAImain']
```

介面限制

- 每 30 秒內最多請求 30 次獲取期貨合約資料介面
- 每次請求的代號列表中，期貨個數上限為 200 個

條件選股

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

- **介紹**

  條件選股

- **參數**

參數	類型	說明
market	Market	市場標識
filter_list	list	篩選條件的列表
plate_code	str	板塊代碼
begin	int	數據起始點
num	int	請求數據個數

  - SimpleFilter 對象相關參數如下：

欄位	類型	說明
stock_field	StockField	簡單屬性
filter_min	float	區間下限
filter_max	float	區間上限
is_no_filter	bool	該欄位是否不需要篩選
sort	SortDir	排序方向

  - AccumulateFilter 對象相關參數如下：

欄位	類型	說明
stock_field	StockField	累積屬性
filter_min	float	區間下限
filter_max	float	區間上限
is_no_filter	bool	該欄位是否不需要篩選
sort	SortDir	排序方向
days	int	所篩選的數據的累計天數

- FinancialFilter 對象相關參數如下：

欄位	類型	說明
stock_field	StockField	財務屬性
filter_min	float	區間下限
filter_max	float	區間上限
is_no_filter	bool	該欄位是否不需要篩選
sort	SortDir	排序方向
quarter	FinancialQuarter	財報累積時間

- CustomIndicatorFilter 對象相關參數如下：

欄位	類型	說明
stock_field1	StockField	自定義技術指標屬性
stock_field1_para	list	自定義技術指標屬性參數
relative_position	RelativePosition	相對位置

欄位	類型	說明
stock_field2	StockField	自定義技術指標屬性
stock_field2_para	list	自定義技術指標屬性參數
value	float	自定義數值
ktype	KLType	K線類型 KLType
consecutive_period	int	篩選連續週期（consecutive_period）都符合條件的數據
is_no_filter	bool	該欄位是否不需要篩選

- PatternFilter 對象相關參數如下：

欄位	類型	說明
stock_field	StockField	形態技術指標屬性
ktype	KLType	K線類型 KLType （僅支援K_60M，K_DAY，K_WEEK，K_MON 四種時間週期）
consecutive_period	int	篩選連續週期（consecutive_period）都符合條件的數據
is_no_filter	bool	該欄位是否不需要篩選

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	tuple	當 ret == RET_OK，返回選股數據
	str	當 ret != RET_OK，返回錯誤描述

- 選股數據元組組成如下：

欄位	類型	說明
last_page	bool	是否是最後一頁
all_count	int	列表總數量
stock_list	list	選股數據

- FilterStockData 類型的欄位格式：

欄位	類型	說明
stock_code	str	股票代碼
stock_name	str	股票名字
cur_price	float	最新價
cur_price_to_highest_52weeks_ratio	float	(現價 - 52周最高)/52周最高
cur_price_to_lowest_52weeks_ratio	float	(現價 - 52周最低)/52周最低
high_price_to_highest_52weeks_ratio	float	(今日最高 - 52周最高)/52周最高
low_price_to_lowest_52weeks_ratio	float	(今日最低 - 52周最低)/52周最低
volume_ratio	float	量比
bid_ask_ratio	float	委比
lot_price	float	每手價格
market_val	float	市值
pe_annual	float	市盈率
pe_ttm	float	市盈率 TTM

欄位	類型	說明
pb_rate	float	市淨率
change_rate_5min	float	五分鐘價格漲跌幅
change_rate_begin_year	float	年初至今價格漲跌幅
ps_ttm	float	市銷率 TTM
pcf_ttm	float	市現率 TTM
total_share	float	總股數
float_share	float	流通股數
float_market_val	float	流通市值
change_rate	float	漲跌幅
amplitude	float	振幅
volume	float	日均成交量
turnover	float	日均成交額
turnover_rate	float	換手率
net_profit	float	淨利潤
net_profix_growth	float	淨利潤增長率
sum_of_business	float	營業收入
sum_of_business_growth	float	營業同比增長率
net_profit_rate	float	淨利率
gross_profit_rate	float	毛利率

欄位	類型	說明
debt_asset_rate	float	資產負債率
return_on_equity_rate	float	淨資產收益率
roic	float	投入資本回報率
roa_ttm	float	資產回報率 TTM
ebit_ttm	float	息稅前利潤 TTM
ebitda	float	稅息折舊及攤銷前利潤
operating_margin_ttm	float	營業利潤率 TTM
ebit_margin	float	EBIT 利潤率
ebitda_margin	float	EBITDA 利潤率
financial_cost_rate	float	財務成本率
operating_profit_ttm	float	營業利潤 TTM
shareholder_net_profit_ttm	float	歸屬於母公司的淨利潤
net_profit_cash_cover_ttm	float	盈利中的現金收入比例
current_ratio	float	流動比率
quick_ratio	float	速動比率
current_asset_ratio	float	流動資產率
current_debt_ratio	float	流動負債率
equity_multiplier	float	權益乘數
property_ratio	float	產權比率

欄位	類型	說明
cash_and_cash_equivalents	float	現金和現金等價
total_asset_turnover	float	總資產週轉率
fixed_asset_turnover	float	固定資產週轉率
inventory_turnover	float	存貨週轉率
operating_cash_flow_ttm	float	經營活動現金流 TTM
accounts_receivable	float	應收賬款淨額
ebit_growth_rate	float	EBIT 同比增長率
operating_profit_growth_rate	float	營業利潤同比增長率
total_assets_growth_rate	float	總資產同比增長率
profit_to_shareholders_growth_rate	float	歸母淨利潤同比增長率
profit_before_tax_growth_rate	float	總利潤同比增長率
eps_growth_rate	float	EPS 同比增長率
roe_growth_rate	float	ROE 同比增長率
roic_growth_rate	float	ROIC 同比增長率
nocf_growth_rate	float	經營現金流同比增長率
nocf_per_share_growth_rate	float	每股經營現金流同比增長率
operating_revenue_cash_cover	float	經營現金收入比
operating_profit_to_total_profit	float	營業利潤佔比
basic_eps	float	基本每股收益

欄位	類型	說明
diluted_eps	float	稀釋每股收益
nocf_per_share	float	每股經營現金淨流量
price	float	最新價格
ma	float	簡單均線
ma5	float	5日簡單均線
ma10	float	10日簡單均線
ma20	float	20日簡單均線
ma30	float	30日簡單均線
ma60	float	60日簡單均線
ma120	float	120日簡單均線
ma250	float	250日簡單均線
rsi	float	RSI的值
ema	float	指數移動均線
ema5	float	5日指數移動均線
ema10	float	10日指數移動均線
ema20	float	20日指數移動均線
ema30	float	30日指數移動均線
ema60	float	60日指數移動均線
ema120	float	120日指數移動均線

欄位	類型	說明
ema250	float	250日指數移動均線
kdj_k	float	KDJ 指標的 K 值
kdj_d	float	KDJ 指標的 D 值
kdj_j	float	KDJ 指標的 J 值
macd_diff	float	MACD 指標的 DIFF 值
macd_dea	float	MACD 指標的 DEA 值
macd	float	MACD 指標的 MACD 值
boll_upper	float	BOLL 指標的 UPPER 值
boll_middler	float	BOLL 指標的 MIDDLER 值
boll_lower	float	BOLL 指標的 LOWER 值

- **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)  # 取股票代碼
            print(item.stock_name)  # 取股票名稱
            print(item[simple_filter])   # 取 simple_filter 對應的變量值
            print(item[financial_filter])   # 取 financial_filter 對應的變量值
            print(item[custom_filter])  # 獲取 custom_filter 的數值
    else:
        print('error: ', ls)
    time.sleep(3)  # 加入時間間隔，避免觸發限頻

quote_ctx.close()  # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
39 39 [ stock_code:HK.08103  stock_name:HMVOD視頻  cur_price:2.69  current_ratio(
HK.08103
HMVOD視頻
2.69
2.69
4.413
...
HK.00306
冠忠巴士集團
```

```
2.29
2.29
49.769
```

提示

- 利用獲取子板塊列表函數 獲取子板塊代碼，條件選股支援的板塊分別為
  1. 港股的行業板塊和概念板塊。
  2. 美股的行業板塊
  3. 滬深的行業板塊，概念板塊和地域板塊
- 支援的板塊指數代碼

代碼	說明
HK.Motherboard	港股主板
HK.GEM	港股創業板
HK.BK1911	H 股主板
HK.BK1912	H 股創業板
US.NYSE	紐交所
US.AMEX	美交所
US.NASDAQ	納斯達克
SH.3000000	上海主板
SZ.3000001	深證主板
SZ.3000004	深證創業板

介面限制

- 每 30 秒內最多請求 10 次條件選股介面
- 每頁返回的篩選結果最多 200 個
- 建議篩選條件不超過 250 個，否則可能會出現“業務處理超時沒返回”

- 累積屬性的同一篩選條件數量上限 10 個
- 如果使用“最新價”等動態數據作為排序字段，在多頁獲取的間隙，數據的排序有可能發生變化
- 非同類指標不支援比較，僅限於同類指標之間建立比較關係，跨不同類型的指標比較會報錯。例如：MA5 和 MA10 可以建立關係。MA5和EMA10不能建立關係。
- 自定義指標屬性的同一類篩選條件超出數量上限10個
- 簡單屬性，財務屬性，形態屬性不支援對同一字段重複指定篩選條件
- 條件選股暫不支援美股盤前盤後、夜盤，篩選結果均按照盤中數據返回

獲取板塊內股票列表

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

- **介紹**

  獲取指定板塊內的股票列表，獲取股指的成分股

- **參數**

參數	類型	說明
plate_code	str	板塊代碼
sort_field	SortField	排序欄位
ascend	bool	排序方向

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回板塊股票數據
	str	當 ret != RET_OK，返回錯誤描述

  - 板塊股票數據

欄位	類型	說明
code	str	股票代碼
lot_size	int	每手股數，期貨表示合約乘數
stock_name	str	股票名稱

欄位	類型	說明
stock_type	SecurityType	股票類型
list_time	str	上市時間
stock_id	int	股票 ID
main_contract	bool	是否主連合約
last_trade_time	str	最後交易時間

- 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])    # 取第一條的股票名稱
    print(data['stock_name'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
    code  lot_size stock_name  stock_owner  stock_child_type stock_type   list_ti
0   HK.00462      4000       天然乳品          NaN               NaN      STOCK
..       ...       ...        ...          ...               ...        ...
9   HK.06186      1000       中國飛鶴          NaN               NaN      STOCK

[10 rows x 10 columns]
天然乳品
['天然乳品', '現代牧業', '雅士利國際', '原生態牧業', '中國聖牧', '中地乳業', '莊園牧場
```

> 介面限制

- 每 30 秒內最多請求 10 次獲取板塊內股票列表介面

▶ 常用的板塊、指數代碼

獲取板塊列表

`get_plate_list(market, plate_class)`

- **介紹**

  獲取板塊列表

- **參數**

參數	類型	說明
market	Market	市場標識
plate_class	Plate	板塊分類

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回板塊列表數據
	str	當 ret != RET_OK，返回錯誤描述

  - 板塊列表數據格式如下：

欄位	類型	說明
code	str	板塊代碼
plate_name	str	板塊名字
plate_id	str	板塊 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])    # 取第一條的板塊名稱
    print(data['plate_name'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
    code plate_name plate_id
0   HK.BK1000      做空集合股   BK1000
..        ...         ...      ...
77  HK.BK1999       殯葬概念   BK1999

[78 rows x 3 columns]
做空集合股
['做空集合股', '阿里概念股', '雄安概念股', '蘋果概念', '一帶一路', '5G概念', '夜店股',
```

> **介面限制**
>
> - 每 30 秒內最多請求 10 次獲取板塊列表介面

獲取靜態數據

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

- **介紹**

  獲取靜態數據

- **參數**

參數	類型	說明
market	Market	市場類型
stock_type	SecurityType	股票類型，但不支援傳入 SecurityType.DRVT
code_list	list	股票列表

  注：當 market 和 code_list 同時存在時，會忽略 market，僅對 code_list 進行查詢。

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回股票靜態數據
	str	當 ret != RET_OK，返回錯誤描述

  - 股票靜態數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱

欄位	類型	說明
lot_size	int	每手股數，期權表示每份合約股數，期貨表示合約乘數
stock_type	SecurityType	股票類型
stock_child_type	WrtType	窩輪子類型
stock_owner	str	窩輪所屬正股的代碼，或期權標的股的代碼
option_type	OptionType	期權類型
strike_time	str	期權行權日
strike_price	float	期權行權價
suspension	bool	期權是否停牌
listing_date	str	上市時間
stock_id	int	股票 ID
delisting	bool	是否退市
index_option_type	str	指數期權類型
main_contract	bool	是否主連合約
last_trade_time	str	最後交易時間
exchange_type	ExchType	所屬交易所

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
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])  # 取第一條的股票名稱
    print(data['name'].values.tolist())  # 轉為 list
else:
    print('error:', data)
quote_ctx.close()  # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
         code              name  lot_size stock_type stock_child_type stock_owner c
0      HK.00001              長和       500      STOCK              N/A
...         ...               ...       ...        ...               ...         .
2592   HK.09979      綠城管理控股      1000      STOCK              N/A

[2593 rows x 16 columns]
**************************************
         code            name  lot_size stock_type stock_child_type stock_owner op
0  HK.06998      嘉和生物-B        500      STOCK              N/A
1  HK.00700      騰訊控股          100      STOCK              N/A
嘉和生物-B
['嘉和生物-B', '騰訊控股']
```

提示

- 當傳入程式無法識別的股票時（包括很久之前退市的股票和不存在的股票），此介面仍然返回股票資訊，用"是否退市"欄位來標識該股票不存在。統一處理為：代碼正常顯示，股票名顯示為"未知股票"，其他欄位均為預設值（整型預設是0，字串預設是空字串）。
- 此介面與其他的行情介面不同，其他介面遇到程式無法識別的股票時，會拒絕請求並返回錯誤描述"未知股票"。

獲取 IPO 資訊

`get_ipo_list(market)`

- **介紹**

  獲取指定市場的 IPO 資訊

- **參數**

參數	類型	說明
market	Market	市場標識

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回 IPO 數據
	str	當 ret != RET_OK，返回錯誤描述

  - IPO 數據

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
list_time	str	上市日期，美股是預計上市日期
list_timestamp	float	上市日期時間戳記，美股是預計上市日期時間戳記
apply_code	str	申購代碼（A 股適用）

欄位	類型	說明
issue_size	int	發行總數（A 股適用）；發行量（美股適用）
online_issue_size	int	網上發行量（A 股適用）
apply_upper_limit	int	申購上限（A 股適用）
apply_limit_market_value	int	頂格申購需配市值（A 股適用）
is_estimate_ipo_price	bool	是否預估發行價（A 股適用）
ipo_price	float	發行價 ℹ（A 股適用）
industry_pe_rate	float	行業市盈率（A 股適用）
is_estimate_winning_ratio	bool	是否預估中籤率（A 股適用）
winning_ratio	float	中籤率 ℹ（A 股適用）
issue_pe_rate	float	發行市盈率（A 股適用）
apply_time	str	申購日期字串ℹ（A 股適用）
apply_timestamp	float	申購日期時間戳記（A 股適用）
winning_time	str	公佈中籤日期字串ℹ（A 股適用）
winning_timestamp	float	公佈中籤日期時間戳記（A 股適用）
is_has_won	bool	是否已經公佈中籤號（A 股適用）
winning_num_data	str	中籤號（A 股適用） ℹ
ipo_price_min	float	最低發售價（港股適用）；最低發行價（美股適用）
ipo_price_max	float	最高發售價（港股適用）；最高發行價（美股適用）
list_price	float	上市價（港股適用）

欄位	類型	說明
lot_size	int	每手股數
entrance_price	float	入場費（港股適用）
is_subscribe_status	bool	是否為認購狀態
apply_end_time	str	截止認購日期字串（港股適用）
apply_end_timestamp	float	截止認購日期時間戳記

- **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])    # 取第一條的股票代碼
    print(data['code'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **Output**

```
    code      name   list_time  list_timestamp apply_code issue_size online_issue
0  HK.06666  恒大物業  2020-12-02    1.606838e+09        N/A        N/A
1  HK.02110  裕勤控股  2020-12-07    1.607270e+09        N/A        N/A
HK.06666
['HK.06666', 'HK.02110']
```

> **介面限制**
>
> - 每 30 秒內最多請求 10 次獲取 IPO 資訊介面

獲取全域市場狀態

`get_global_state()`

- **介紹**

  獲取全域狀態

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	dict	當 ret == RET_OK 時，返回全域狀態
	str	當 ret != RET_OK，返回錯誤描述

  - 全域狀態字典格式如下：

欄位	類型	說明
market_sz	MarketState	深圳市場狀態
market_sh	MarketState	上海市場狀態
market_hk	MarketState	香港市場狀態
market_hkfuture	MarketState	香港期貨市場狀態
market_usfuture	MarketState	美國期貨市場狀態
market_us	MarketState	美國市場狀態
market_sgfuture	MarketState	新加坡期貨市場狀態
market_jpfuture	MarketState	日本期貨市場狀態

欄位	類型	說明
server_ver	str	OpenD 版本號
trd_logined	bool	True：已登入交易伺服器，False：未登入交易伺服器
qot_logined	bool	True：已登入行情伺服器，False：未登入行情伺服器
timestamp	str	當前格林威治時間戳記
local_timestamp	float	OpenD 運行機器的當前時間戳記
program_status_type	ProgramStatusType	當前狀態
program_status_desc	str	額外描述

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
print(quote_ctx.get_global_state())
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **Output**

```
(0, {'market_sz': 'MORNING', 'market_us': 'AFTER_HOURS_END', 'market_sh': 'MORNI
```

獲取交易日曆

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

- **介紹**

  請求指定市場 / 指定標的的交易日曆。
  注意：該交易日是透過自然日剔除週末和節假日得到，未剔除臨時休市數據。

- **參數**

參數	類型	說明
market	TradeDateMarket	市場類型
start	str	起始日期
end	str	結束日期
code	str	股票代碼

  注：當 market 和 code 同時存在時，會忽略 market，僅對 code 進行查詢。

  - start 和 end 的組合如下

Start 類型	End 類型	說明
str	str	start 和 end 分別為指定的日期
None	str	start 為 end 往前 365 天
str	None	end 為 start 往後 365 天
None	None	start 為往前 365 天，end 當前日期

- **返回**

參數	類型	說明

ret	RET_CODE	介面呼叫結果
data	list	當 ret == RET_OK 時，返回交易日數據。list 中元素類型為 dict
	str	當 ret != RET_OK 時，返回錯誤描述

- 交易日數據格式如下：

欄位	類型	說明
time	str	時間
trade_date_type	TradeDateType	交易日類型

- **Example**

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

ret, data = quote_ctx.request_trading_days(market=TradeDateMarket.HK, start='2020
if ret == RET_OK:
    print('HK market calendar:', data)
else:
    print('error:', data)
print('**************************************')
ret, data = quote_ctx.request_trading_days(start='2020-04-01', end='2020-04-10',
if ret == RET_OK:
    print('HK.00700 calendar:', data)
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **Output**

```
HK market calendar: [{'time': '2020-04-01', 'trade_date_type': 'WHOLE'}, {'time'
**************************************
HK.00700 calendar: [{'time': '2020-04-01', 'trade_date_type': 'WHOLE'}, {'time':
```

> **介面限制**

- 每 30 秒內最多請求 30 次獲取交易日介面。
- 歷史交易日曆提供過去 10 年的數據，未來交易日曆提供到今年 12 月 31 日ℹ️。

獲取歷史 K 線額度使用明細

`get_history_kl_quota(get_detail=False)`

- **介紹**

  獲取歷史 K 線額度使用明細

- **參數**

參數	類型	說明
get_detail	bool	是否返回下載 / 讀取歷史 K 線的詳細紀錄

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	tuple	當 ret == RET_OK，返回歷史 K 線額度數據
	str	當 ret != RET_OK，返回錯誤描述

  - 歷史 K 線額度數據格式如下：

欄位	類型	說明
used_quota	int	已用額度
remain_quota	int	剩餘額度
detail_list	list	下載 / 讀取歷史 K 線的詳細紀錄，含股票代碼和下載 / 讀取時間

    - detail_list 數據列格式如下

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
request_time	str	最後一次下載 / 讀取的時間字串

- 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)  # 設定 true 代表需要
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
(2, 98, {'code': 'HK.00123', 'name': '越秀地產', 'request_time': '2023-06-20 19:5
```

> **介面限制**
>
> - 我們會根據您帳戶的資產和交易的情況，發放歷史 K 線額度。因此，30 天內您只能獲取有限只股票的歷史 K 線數據。具體規則參見 訂閱額度 & 歷史 K 線額度。
> - 您當日消耗的歷史 K 線額度，會在 30 天后自動釋放。

設定到價提醒

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

- **介紹**

  新增、刪除、修改、啟用、禁用指定股票的到價提醒

- **參數**

參數	類型	說明
code	str	股票代碼
op	SetPriceReminderOp	操作類型
key	int	標識，新增和刪除全部的情況不需要填
reminder_type	PriceReminderType	到價提醒的類型，刪除、啟用、禁用的情況下會忽略該入參
reminder_freq	PriceReminderFreq	到價提醒的頻率，刪除、啟用、禁用的情況下會忽略該入參
value	float	提醒值，刪除、啟用、禁用的情況下會忽略該入參
note	str	用戶設定的備註，僅支援 20 個以內的中文字元，刪除、啟用、禁用的情況下會忽略該入參
reminder_session_list	list	美股到價提醒的時段列表，刪除、啟用、禁用的情況下會忽略該入參

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
key	int	當 ret == RET_OK 時，返回操作的到價提醒 key
	str	當 ret != RET_OK，返回錯誤描述

- 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) # PriceReminderTest 自己的處理邏輯
        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]  # 獲取實時買一價
    ask_price = data['ask_price'][0]  # 獲取實時賣一價
    # 設定當AAPL全時段賣一價低於（ask_price-1）時提醒
    ret_ask, ask_data = quote_ctx.set_price_reminder(code='US.AAPL', op=SetPriceR
    if ret_ask == RET_OK:
        print('賣一價低於（ask_price-1）時提醒設定成功：', ask_data)
    else:
        print('error:', ask_data)
    # 設定當AAPL全時段買一價高於（bid_price+1）時提醒
    ret_bid, bid_data = quote_ctx.set_price_reminder(code='US.AAPL', op=SetPriceR
    if ret_bid == RET_OK:
        print('買一價高於（bid_price+1）時提醒設定成功：', bid_data)
    else:
        print('error:', bid_data)
time.sleep(15)
quote_ctx.close()
```

- Output

```
賣一價低於（ask_price-1）時提醒設定成功： 1744022257023211123
買一價高於（bid_price+1）時提醒設定成功： 1744022257052794489
```

提示

- API 中成交量設定統一以股為單位。但是 moomoo 用戶端中，A 股是以手為單位展示
- 到價提醒類型，存在最小精度，如下：

  TURNOVER_UP：成交額最小精度為 10 元（人民幣元，港元，美元）。傳入的數值會自動向下取整到最小精度的整數倍。如果設定【00700成交額102元提醒】，設定後會得到【00700成交額100元提醒】；如果設定【00700 成交額 8 元提醒】，設定後會得到【00700 成交額 0 元提醒】。

  VOLUME_UP：A 股成交量最小精度為 1000 股，其他市場股票成交量最小精度為 10 股。傳入的數值會自動向下取整到最小精度的整數倍。

  BID_VOL_UP、ASK_VOL_UP：A 股的買一賣一量最小精度為 100 股。傳入的數值會自動向下取整到最小精度的整數倍。

  其餘到價提醒類型精度支援到小數點後 3 位

介面限制

- 每 30 秒內最多請求 60 次設定到價提醒介面
- 每隻股票每種類型可設定的提醒上限是 10 個

獲取到價提醒列表

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

- **介紹**

  獲取對指定股票 / 指定市場設定的到價提醒列表

- **參數**

參數	類型	說明
code	str	股票代碼
market	Market	市場類型

  注：code 和 market 都存在的情況下，code 優先。

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回到價提醒數據
	str	當 ret != RET_OK，返回錯誤描述

  - 到價提醒數據格式如下：

欄位	類型	說明
code	str	股票代碼
key	int	標識，用於修改到價提醒
reminder_type	PriceReminderType	到價提醒的類型

欄位	類型	說明
reminder_freq	PriceReminderFreq	到價提醒的頻率
value	float	提醒值
enable	bool	是否啓用
note	str	備註
reminder_session_list	list	美股到價提醒時段列表

- Example

```python
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())   # 轉為 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:  # 如果到價提醒列表不為空
        print(data['code'][0])    # 取第一條的股票代碼
        print(data['code'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- Output

```
code name                 key    reminder_type reminder_freq   value  enable note
0  US.AAPL   蘋果  1744021708234288125     BID_PRICE_UP        ALWAYS  184.37    T
1  US.AAPL   蘋果  1744022257052794489     BID_PRICE_UP        ALWAYS  185.50    T
2  US.AAPL   蘋果  1744021708211891867  ASK_PRICE_DOWN        ALWAYS  182.54    T
3  US.AAPL   蘋果  1744022257023211123  ASK_PRICE_DOWN        ALWAYS  183.70    T
```

```
[1744021708234288125, 1744022257052794489, 1744021708211891867, 1744022257023211
**************************************
      code name                 key   reminder_type reminder_freq   value  enabl
0  US.AAPL   蘋果  1744021708234288125    BID_PRICE_UP        ALWAYS  184.37     T
1  US.AAPL   蘋果  1744022257052794489    BID_PRICE_UP        ALWAYS  185.50     T
2  US.AAPL   蘋果  1744021708211891867  ASK_PRICE_DOWN        ALWAYS  182.54     T
3  US.AAPL   蘋果  1744022257023211123  ASK_PRICE_DOWN        ALWAYS  183.70     T
4  US.NVDA  英偉達  1739697581665326308      PRICE_DOWN         ALWAYS  102.00
US.AAPL
['US.AAPL', 'US.AAPL', 'US.AAPL', 'US.AAPL', 'US.NVDA']
```

介面限制

- 每 30 秒內最多請求 10 次獲取到價提醒列表介面

獲取自選股列表

`get_user_security(group_name)`

- **介紹**

  獲取指定分組的自選股列表

- **參數**

參數	類型	說明
group_name	str	需要查詢的自選股分組名稱

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回自選股數據
	str	當 ret != RET_OK，返回錯誤描述

  - 自選股數據格式如下：

欄位	類型	說明
code	str	股票代碼
name	str	名字
lot_size	int	每手股數，期權表示每份合約股數，期貨表示合約乘數
stock_type	SecurityType	股票類型
stock_child_type	WrtType	窩輪子類型

欄位	類型	說明
stock_owner	str	窩輪所屬正股的代碼，或期權標的股的代碼
option_type	OptionType	期權類型
strike_time	str	期權行權日
strike_price	float	期權行權價
suspension	bool	期權是否停牌
listing_date	str	上市時間
stock_id	int	股票 ID
delisting	bool	是否退市
main_contract	bool	是否主連合約
last_trade_time	str	最後交易時間

- **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:  # 如果自選股列表不為空
        print(data['code'][0])    # 取第一條的股票代碼
        print(data['code'].values.tolist())   # 轉為 list
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **Output**

```
    code    name  lot_size stock_type stock_child_type stock_owner option_type st
0  HK.HSImain  恒指期貨主連        50     FUTURE              N/A
1  HK.00700    騰訊控股       100      STOCK              N/A
HK.HSImain
['HK.HSImain', 'HK.00700']
```

提示

系統分組的中英文對應名稱如下

中文	英文
全部	All
滬深	CN
港股	HK
美股	US
期權	Options
港股期權	HK options
美股期權	US options
特別關注	Starred
期貨	Futures

介面限制

- 每 30 秒內最多請求 10 次獲取自選股列表介面
- 不支援持倉（Positions），基金寶（Mutual Fund），外匯（Forex）分組的查詢

獲取自選股分組

`get_user_security_group(group_type = UserSecurityGroupType.ALL)`

- **介紹**

  獲取自選股分組列表

- **參數**

參數	類型	說明
group_type	UserSecurityGroupType	分組類型

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	pd.DataFrame	當 ret == RET_OK，返回自選股分組數據
	str	當 ret != RET_OK，返回錯誤描述

  - 自選股分組數據格式如下：

欄位	類型	說明
group_name	str	分組名
group_type	UserSecurityGroupType	分組類型

- **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
```

```python
if ret == RET_OK:
    print(data)
else:
    print('error:', data)
quote_ctx.close() # 結束後記得關閉當條連接，防止連接條數用盡
```

- **Output**

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

[13 rows x 2 columns]
```

> **介面限制**
>
> - 每 30 秒內最多請求 10 次獲取自選股分組介面

修改自選股列表

`modify_user_security(group_name, op, code_list)`

- **介紹**

  修改指定分組的自選股列表（系統分組不支援修改）

- **參數**

參數	類型	說明
group_name	str	需要修改的自選股分組名稱
op	ModifyUserSecurityOp	操作類型
code_list	list	股票列表

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
msg	str	當 ret == RET_OK，返回“success”
		當 ret != RET_OK，msg 返回錯誤描述

- **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) # 返回 success
else:
```

```
        print('error:', data)
    quote_ctx.close() # 結束後記得關閉當條連線，防止連線條數用盡
```

- **Output**

```
success
```

> **介面限制**
>
> - 每 30 秒內最多請求 10 次修改自選股列表介面
> - 僅支援修改自定義分組，不支援修改系統分組
> - “全部”自選股列表的數量存在上限：無交易客戶 500 個，有交易客戶 2000 個（向其他分組增加自選股時，“全部”列表中也會同步增加）
> - 如果有同名的分組，會操作排序在第一個的分組

到價提醒回呼

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  到價提醒通知回呼，非同步處理已設定到價提醒的通知推送。
  在收到實時到價提醒通知推送後會回呼到該函數，您需要在衍生類別中覆寫 on_recv_rsp。

- **參數**

參數	類型	說明
rsp_pb	Qot_UpdatePriceReminder_pb2.Response	衍生類別中不需要直接處理該參數

- **返回**

參數	類型	說明
ret	RET_CODE	介面呼叫結果
data	dict	當 ret == RET_OK，返回到價提醒
	str	當 ret != RET_OK，返回錯誤描述

  - 到價提醒

欄位	類型	說明
code	str	股票代碼
name	str	股票名稱
price	float	當前價格
change_rate	str	當前漲跌幅

欄位	類型	說明
market_status	PriceReminderMarketStatus	觸發的時間段
content	str	到價提醒文字內容
note	str	備註
key	int	到價提醒標識
reminder_type	PriceReminderType	到價提醒的類型
set_value	float	用戶設定的提醒值
cur_value	float	提醒觸發時的值

- 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 自己的處理邏輯
        return RET_OK, content
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = PriceReminderTest()
quote_ctx.set_handler(handler)  # 設定到價提醒通知回呼
time.sleep(15)  # 設定腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()   # 關閉當條連線，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

- Output

```
PriceReminderTest  {'code': 'US.AAPL', 'name': '蘋果', 'price': 185.750, 'change_
```

提示

- 此介面提供了持續獲取推送數據的功能，如需一次性獲取實時數據，請參考 獲取到價提醒 介面
- 獲取實時數據 和 實時數據回呼 的差別，請參考 如何透過訂閱介面獲取實時行情？

行情定義

累積過濾屬性

> StockField

- `NONE`

  未知

- `CHANGE_RATE`

  漲跌幅

- `AMPLITUDE`

  振幅

- `VOLUME`

  日均成交量

- `TURNOVER`

  日均成交額

- `TURNOVER_RATE`

  換手率

資產類別

> AssetClass

- `UNKNOW`

  未知

- `STOCK`

  股票

- `BOND`

  債券

- `COMMODITY`

  商品

- `CURRENCY_MARKET`

  貨幣市場

- `FUTURE`

  期貨

- `SWAP`

  掉期（互換）

公司行動

暗盤狀態

> DarkStatus

- `NONE`

  無暗盤交易

- `TRADING`

  暗盤交易中

- `END`

  暗盤交易結束

財務過濾屬性

> StockField

- `NONE`

  未知

- `NET_PROFIT`

  淨利潤 ℹ

- `NET_PROFIX_GROWTH`

  淨利潤增長率 ℹ

- `SUM_OF_BUSINESS`

  營業收入 ℹ

- `SUM_OF_BUSINESS_GROWTH`

  營收按年增長率 ℹ

- `NET_PROFIT_RATE`

  淨利率 ℹ

- `GROSS_PROFIT_RATE`

  毛利率 ℹ

- `DEBT_ASSET_RATE`

  資產負債率 ℹ

- `RETURN_ON_EQUITY_RATE`

  淨資產收益率 ℹ

- `ROIC`

投入資本回報率

- `ROA_TTM`

  資產回報率 TTM

- `EBIT_TTM`

  息稅前利潤 TTM

- `EBITDA`

  稅息折舊及攤銷前利潤

- `OPERATING_MARGIN_TTM`

  營業利潤率 TTM

- `EBIT_MARGIN`

  EBIT 利潤率

- `EBITDA_MARGIN`

  EBITDA 利潤率

- `FINANCIAL_COST_RATE`

  財務成本率

- `OPERATING_PROFIT_TTM`

  營業利潤 TTM

- `SHAREHOLDER_NET_PROFIT_TTM`

  歸屬於母公司的淨利潤

- `NET_PROFIT_CASH_COVER_TTM`

  盈利中的現金收入比例

- `CURRENT_RATIO`

  流動比率

- `QUICK_RATIO`

  速動比率 ℹ

- `CURRENT_ASSET_RATIO`

  流動資產率 ℹ

- `CURRENT_DEBT_RATIO`

  流動負債率 ℹ

- `EQUITY_MULTIPLIER`

  權益乘數 ℹ

- `PROPERTY_RATIO`

  產權比率 ℹ

- `CASH_AND_CASH_EQUIVALENTS`

  現金和現金等價物 ℹ

- `TOTAL_ASSET_TURNOVER`

  總資產週轉率 ℹ

- `FIXED_ASSET_TURNOVER`

  固定資產週轉率 ℹ

- `INVENTORY_TURNOVER`

  存貨週轉率 ℹ

- `OPERATING_CASH_FLOW_TTM`

  經營活動現金流 TTM ℹ

- `ACCOUNTS_RECEIVABLE`

  應收帳款淨額 ℹ

- `EBIT_GROWTH_RATE`

EBIT 按年增長率

- `OPERATING_PROFIT_GROWTH_RATE`

  營業利潤按年增長率

- `TOTAL_ASSETS_GROWTH_RATE`

  總資產按年增長率

- `PROFIT_TO_SHAREHOLDERS_GROWTH_RATE`

  歸母淨利潤按年增長率

- `PROFIT_BEFORE_TAX_GROWTH_RATE`

  總利潤按年增長率

- `EPS_GROWTH_RATE`

  EPS 按年增長率

- `ROE_GROWTH_RATE`

  ROE 按年增長率

- `ROIC_GROWTH_RATE`

  ROIC 按年增長率

- `NOCF_GROWTH_RATE`

  經營現金流按年增長率

- `NOCF_PER_SHARE_GROWTH_RATE`

  每股經營現金流按年增長率

- `OPERATING_REVENUE_CASH_COVER`

  經營現金收入比

- `OPERATING_PROFIT_TO_TOTAL_PROFIT`

  營業利潤佔比

- `BASIC_EPS`

  基本每股收益 ℹ

- `DILUTED_EPS`

  稀釋每股收益 ℹ

- `NOCF_PER_SHARE`

  每股經營現金淨流量 ℹ

財務過濾屬性週期

> FinancialQuarter

- `NONE`

  未知

- `ANNUAL`

  年報

- `FIRST_QUARTER`

  一季報

- `INTERIM`

  中報

- `THIRD_QUARTER`

  三季報

- `MOST_RECENT_QUARTER`

  最近季報

自定義技術指標屬性

> StockField

- `NONE`

  未知

- `PRICE`

  最新價格

- `MA`

  簡單均線

- `MA5`

  5日簡單均線（不建議使用）

- `MA10`

  10日簡單均線（不建議使用）

- `MA20`

  20日簡單均線（不建議使用）

- `MA30`

  30日簡單均線（不建議使用）

- `MA60`

  60日簡單均線（不建議使用）

- `MA120`

  120日簡單均線（不建議使用）

- `MA250`

  250日簡單均線（不建議使用）

- `RSI`

  RSI ℹ

- `EMA`

  指數移動均線

- `EMA5`

  5日指數移動均線（不建議使用）

- `EMA10`

  10日指數移動均線（不建議使用）

- `EMA20`

  20日指數移動均線（不建議使用）

- `EMA30`

  30日指數移動均線（不建議使用）

- `EMA60`

  60日指數移動均線（不建議使用）

- `EMA120`

  120日指數移動均線（不建議使用）

- `EMA250`

  250日指數移動均線（不建議使用）

- `KDJ_K`

  KDJ 指標的 K 值

- `KDJ_D`

  KDJ 指標的 D 值

- `KDJ_J`

  KDJ 指標的 J 值

- `MACD_DIFF`

MACD 指標的 DIFF 值

- `MACD_DEA`

  MACD 指標的 DEA 值

- `MACD`

  MACD

- `BOLL_UPPER`

  BOLL 指標的 UPPER 值

- `BOLL_MIDDLER`

  BOLL 指標的 MIDDLER 值

- `BOLL_LOWER`

  BOLL 指標的 LOWER 值

- `VALUE`

  自定義數值（stock_field1 不支援此欄位）

相對位置

> **RelativePosition**

- `NONE`

  未知

- `MORE`

  大於，stock_field1 位於stock_field2 的上方

- `LESS`

  小於，stock_field1 位於stock_field2 的下方

- `CROSS_UP`

升穿，stock_field1 從下往上穿stock_field2

- `CROSS_DOWN`

  跌穿，stock_field1 從上往下穿stock_field2

形態技術指標屬性

> PatternField

- `NONE`

  未知

- `MA_ALIGNMENT_LONG`

  MA多頭排列（連續兩天MA5>MA10>MA20>MA30>MA60，且當日收盤價大於前一天收盤價）

- `MA_ALIGNMENT_SHORT`

  MA空頭排列（連續兩天MA5<MA10<MA20<MA30<MA60，且當日收盤價小於前一天收盤價）

- `EMA_ALIGNMENT_LONG`

  EMA多頭排列（連續兩天EMA5>EMA10>EMA20>EMA30>EMA60，且當日收盤價大於前一天收盤價）

- `EMA_ALIGNMENT_SHORT`

  EMA空頭排列（連續兩天EMA5<EMA10<EMA20<EMA30<EMA60，且當日收盤價小於前一天收盤價）

- `RSI_GOLD_CROSS_LOW`

  RSI低位金叉（50以下，短線RSI上穿長線RSI（前一日短線RSI小於長線RSI，當日短線RSI大於長線RSI））

- `RSI_DEATH_CROSS_HIGH`

  RSI高位死叉（50以上，短線RSI下穿長線RSI（前一日短線RSI大於長線RSI，當日短線RSI小於長線RSI））

- `RSI_TOP_DIVERGENCE`

  RSI頂背離（相鄰的兩個K線波峯，後面的波峯對應的CLOSE>前面的波峯對應的CLOSE，後面波峯的RSI12值<前面波峯的RSI12值）

- `RSI_BOTTOM_DIVERGENCE`

  RSI底背離（相鄰的兩個K線波谷，後面的波谷對應的CLOSE<前面的波谷對應的CLOSE，後面波谷的RSI12值>前面波谷的RSI12值）

- `KDJ_GOLD_CROSS_LOW`

  KDJ低位金叉（D值小於或等於30，且前一日K值小於D值，當日K值大於D值）

- `KDJ_DEATH_CROSS_HIGH`

  KDJ高位死叉（D值大於或等於70，且前一日K值大於D值，當日K值小於D值）

- `KDJ_TOP_DIVERGENCE`

  KDJ頂背離（相鄰的兩個K線波峯，後面的波峯對應的CLOSE>前面的波峯對應的CLOSE，後面波峯的J值<前面波峯的J值）

- `KDJ_BOTTOM_DIVERGENCE`

  KDJ底背離（相鄰的兩個K線波谷，後面的波谷對應的CLOSE<前面的波谷對應的CLOSE，後面波谷的J值>前面波谷的J值）

- `MACD_GOLD_CROSS_LOW`

  MACD低位金叉（DIFF上穿DEA（前一日DIFF小於DEA，當日DIFF大於DEA））

- `MACD_DEATH_CROSS_HIGH`

  MACD高位死叉（DIFF下穿DEA（前一日DIFF大於DEA，當日DIFF小於DEA））

- `MACD_TOP_DIVERGENCE`

  MACD頂背離（相鄰的兩個K線波峯，後面的波峯對應的CLOSE>前面的波峯對應的CLOSE，後面波峯的macd值<前面波峯的macd值）

- `MACD_BOTTOM_DIVERGENCE`

MACD底背離（相鄰的兩個K線波谷，後面的波谷對應的CLOSE<前面的波谷對應的CLOSE，後面波谷的macd值>前面波谷的macd值）

- `BOLL_BREAK_UPPER`

  BOLL突破上軌（前一日股價低於上軌值，當日股價大於上軌值）

- `BOLL_BREAK_LOWER`

  BOLL突破下軌（前一日股價高於下軌值，當日股價小於下軌值）

- `BOLL_CROSS_MIDDLE_UP`

  BOLL向上破中軌（前一日股價低於中軌值，當日股價大於中軌值）

- `BOLL_CROSS_MIDDLE_DOWN`

  BOLL向下破中軌（前一日股價大於中軌值，當日股價小於中軌值）

自選股分組類型

> UserSecurityGroupType

- `NONE`

  未知

- `CUSTOM`

  自定義分組

- `SYSTEM`

  系統分組

- `ALL`

  全部分組

指數期權類別

> IndexOptionType

- `NONE`

  未知

- `NORMAL`

  普通的指數期權

- `SMALL`

  小型指數期權

上市時段

> IpoPeriod

- `NONE`

  未知

- `TODAY`

  今日上市

- `TOMORROW`

  明日上市

- `NEXTWEEK`

  未來一週上市

- `LASTWEEK`

  過去一週上市

- `LASTMONTH`

  過去一月上市

窩輪發行商

> Issuer

- `UNKNOW`

  未知

- `SG`

  法興

- `BP`

  法巴

- `CS`

  瑞信

- `CT`

  花旗

- `EA`

  東亞

- `GS`

  高盛

- `HS`

  滙豐

- `JP`

  摩通

- `MB`

  麥銀

- `SC`

  渣打

- `UB`

  瑞銀
- `BI`

  中銀
- `DB`

  德銀
- `DC`

  大和
- `ML`

  美林
- `NM`

  野村
- `RB`

  荷合
- `RS`

  蘇皇
- `BC`

  巴克萊
- `HT`

  海通
- `VT`

  瑞通
- `KC`

比聯

- `MS`

  摩利

- `GJ`

  國君

- `XZ`

  星展

- `HU`

  華泰

- `KS`

  韓投

- `CI`

  信證

K 線欄位

> KL_FIELD

- `ALL`

  所有

- `DATE_TIME`

  時間

- `HIGH`

  最高價

- `OPEN`

開盤價

- `LOW`

  最低價

- `CLOSE`

  收盤價

- `LAST_CLOSE`

  昨收價

- `TRADE_VOL`

  成交量

- `TRADE_VAL`

  成交額

- `TURNOVER_RATE`

  換手率

- `PE_RATIO`

  市盈率

- `CHANGE_RATE`

  漲跌幅

K 線類型

> KLType

- `NONE`

  未知

- `K_1M`

1分 K

- `K_DAY`

  日 K

- `K_WEEK`

  周 K

- `K_MON`

  月 K

- `K_YEAR`

  年 K

- `K_5M`

  5分 K

- `K_15M`

  15分 K

- `K_30M`

  30分 K

- `K_60M`

  60分 K

- `K_3M`

  3分 K

- `K_QUARTER`

  季 K

週期類型

> PeriodType

- `INTRADAY`

  實時

- `DAY`

  日

- `WEEK`

  周

- `MONTH`

  月

到價提醒市場狀態

> PriceReminderMarketStatus

- `NONE`

  未知

- `OPEN`

  盤中

- `US_PRE`

  美股盤前

- `US_AFTER`

  美股盤後

- `US_OVERNIGHT`

  美股夜盤

自選股操作

> ModifyUserSecurityOp

- `NONE`

  未知

- `ADD`

  新增

- `DEL`

  刪除自選

- `MOVE_OUT`

  移出分組

期權類型（按行權時間）

> OptionAreaType

- `NONE`

  未知

- `AMERICAN`

  美式

- `EUROPEAN`

  歐式

- `BERMUDA`

  百慕大

期權價內/外

> OptionCondType

- `ALL`

  所有

- `WITHIN`

  價內

- `OUTSIDE`

  價外

期權類型（按方向）

> OptionType

- `ALL`

  所有

- `CALL`

  認購期權

- `PUT`

  認沽期權

板塊集合類型

> Plate

- `ALL`

所有板塊

- `INDUSTRY`

  行業板塊

- `REGION`

  地域板塊

- `CONCEPT`

  概念板塊

- `OTHER`

  其他板塊

到價提醒頻率

> PriceReminderFreq

- `NONE`

  未知

- `ALWAYS`

  持續提醒

- `ONCE_A_DAY`

  每日一次

- `ONCE`

  僅提醒一次

到價提醒類型

> PriceReminderType

- `NONE`

  未知

- `PRICE_UP`

  價格漲到

- `PRICE_DOWN`

  價格跌到

- `CHANGE_RATE_UP`

  日漲幅超 ℹ

- `CHANGE_RATE_DOWN`

  日跌幅超 ℹ

- `FIVE_MIN_CHANGE_RATE_UP`

  5 分鐘漲幅超 ℹ

- `FIVE_MIN_CHANGE_RATE_DOWN`

  5 分鐘跌幅超 ℹ

- `VOLUME_UP`

  成交量超過

- `TURNOVER_UP`

  成交額超過

- `TURNOVER_RATE_UP`

  換手率超過 ℹ

- `BID_PRICE_UP`

  買一價高於

- `ASK_PRICE_DOWN`

賣一價低於

- `BID_VOL_UP`

  買一量高於

- `ASK_VOL_UP`

  賣一量高於

- `THREE_MIN_CHANGE_RATE_UP`

  3 分鐘漲幅超 ℹ

- `THREE_MIN_CHANGE_RATE_DOWN`

  3 分鐘跌幅超 ℹ

窩輪價內/外

> PriceType

- `UNKNOW`

  未知

- `OUTSIDE`

  價外，界內證表示界外

- `WITH_IN`

  價內，界內證表示界內

逐筆推送類型

> PushDataType

- `UNKNOW`

  未知

- `REALTIME`

  實時推送的資料

- `BYDISCONN`

  與富途伺服器連線斷開期間，拉取補充的資料

- `CACHE`

  非實時非連線斷開補充資料

行情市場

> Market

- `NONE`

  未知市場

- `HK`

  香港市場

- `US`

  美國市場

- `SH`

  滬股市場

- `SZ`

  深股市場

- `SG`

  新加坡市場

- `JP`

  日本市場

- `AU`

  澳大利亞市場

- `CA`

  加拿大市場

- `MY`

  馬來西亞市場

- `FX`

  外匯市場

市場狀態

> MarketState

各市場狀態的對應時段：點擊這裡瞭解更多

- `NONE`

  無交易

- `AUCTION`

  盤前競價

- `WAITING_OPEN`

  等待開盤

- `MORNING`

  早盤

- `REST`

  午間休市

- `AFTERNOON`

午盤 / 美股持續交易時段

- `CLOSED`

  收盤

- `PRE_MARKET_BEGIN`

  美股盤前交易時段

- `PRE_MARKET_END`

  美股盤前交易結束

- `AFTER_HOURS_BEGIN`

  美股盤後交易時段

- `AFTER_HOURS_END`

  美股盤後結束

- `OVERNIGHT`

  美股夜盤交易時段

- `NIGHT_OPEN`

  夜市交易時段

- `NIGHT_END`

  夜市收盤

- `NIGHT`

  美指期權夜市交易時段

- `TRADE_AT_LAST`

  美指期權盤尾交易時段

- `FUTURE_DAY_OPEN`

  日市交易時段

- `FUTURE_DAY_BREAK`

  日市休市
- `FUTURE_DAY_CLOSE`

  日市收盤
- `FUTURE_DAY_WAIT_OPEN`

  期貨待開盤
- `HK_CAS`

  港股盤後競價
- `FUTURE_NIGHT_WAIT`

  夜市等待開盤（已廢棄）
- `FUTURE_AFTERNOON`

  期貨下午開盤（已廢棄）
- `FUTURE_SWITCH_DATE`

  美期待開盤
- `FUTURE_OPEN`

  美期交易時段
- `FUTURE_BREAK`

  美期中盤休息
- `FUTURE_BREAK_OVER`

  美期休息後交易時段
- `FUTURE_CLOSE`

  美期收盤
- `STIB_AFTER_HOURS_WAIT`

科創板的盤後撮合時段（已廢棄）

- `STIB_AFTER_HOURS_BEGIN`

  科創板的盤後交易開始（已廢棄）

- `STIB_AFTER_HOURS_END`

  科創板的盤後交易結束（已廢棄）

美股時段

> Session

- `NONE`

  未知

- `RTH`

  美股盤中時段

- `ETH`

  美股盤中+盤前盤後

- `OVERNIGHT`

  美股夜盤時段 (僅用於交易介面)

- `ALL`

  美股全時段 (用於行情&交易介面)

行情權限

> QotRight

- `UNKNOW`

  未知

- `BMP`

  BMP（此權限不支援訂閱）
- `LEVEL1`

  Level1
- `LEVEL2`

  Level2
- `SF`

  港股 SF 高級全盤行情
- `NO`

  無權限

關聯資料類型

> SecurityReferenceType

- `UNKNOW`

  未知
- `WARRANT`

  正股相關的窩輪
- `FUTURE`

  期貨主連的相關合約

K 線復權類型

> AuType

- `NONE`

不復權

- `QFQ`

  前復權

- `HFQ`

  後復權

股票狀態

> SecurityStatus

- `NONE`

  未知

- `NORMAL`

  正常狀態

- `LISTING`

  待上市

- `PURCHASING`

  申購中

- `SUBSCRIBING`

  認購中

- `BEFORE_DRAK_TRADE_OPENING`

  暗盤開盤前

- `DRAK_TRADING`

  暗盤交易中

- `DRAK_TRADE_END`

暗盤已收盤

- `TO_BE_OPEN`

  待開盤

- `SUSPENDED`

  停牌

- `CALLED`

  已收回

- `EXPIRED_LAST_TRADING_DATE`

  已過最後交易日

- `EXPIRED`

  已過期

- `DELISTED`

  已退市

- `CHANGE_TO_TEMPORARY_CODE`

  公司行動中，交易關閉，轉至臨時代碼交易

- `TEMPORARY_CODE_TRADE_END`

  臨時買賣結束，交易關閉

- `CHANGED_PLATE_TRADE_END`

  已轉板，舊代碼交易關閉

- `CHANGED_CODE_TRADE_END`

  已換代碼，舊代碼交易關閉

- `RECOVERABLE_CIRCUIT_BREAKER`

  可恢復性熔斷

- `UN_RECOVERABLE_CIRCUIT_BREAKER`

  不可恢復性熔斷

- `AFTER_COMBINATION`

  盤後撮合

- `AFTER_TRANSATION`

  盤後交易

股票類型

> SecurityType

- `NONE`

  未知

- `BOND`

  債券

- `BWRT`

  一攬子權證

- `STOCK`

  正股

- `ETF`

  信託,基金

- `WARRANT`

  窩輪

- `IDX`

  指數

- `PLATE`

  板塊

- `DRVT`

  期權

- `PLATESET`

  板塊集

- `FUTURE`

  期貨

設置到價提醒操作類型

> SetPriceReminderOp

- `NONE`

  未知

- `ADD`

  新增

- `DEL`

  刪除

- `ENABLE`

  啟用

- `DISABLE`

  禁用

- `MODIFY`

  修改

- `DEL_ALL`

  刪除全部（刪除指定股票下的所有到價提醒）

排序方向

> SortDir

- `NONE`

  不排序

- `ASCEND`

  升序

- `DESCEND`

  降序

排序欄位

> SortField

- `NONE`

  未知

- `CODE`

  代碼

- `CUR_PRICE`

  最新價

- `PRICE_CHANGE_VAL`

  漲跌額

- `CHANGE_RATE`

漲跌幅 %

- `STATUS`

  狀態

- `BID_PRICE`

  買入價

- `ASK_PRICE`

  賣出價

- `BID_VOL`

  買量

- `ASK_VOL`

  賣量

- `VOLUME`

  成交量

- `TURNOVER`

  成交額

- `AMPLITUDE`

  振幅 %

- `SCORE`

  綜合評分

- `PREMIUM`

  溢價 %

- `EFFECTIVE_LEVERAGE`

  有效槓桿

- `DELTA`

  對沖值 ℹ

- `IMPLIED_VOLATILITY`

  引伸波幅 ℹ

- `TYPE`

  類型

- `STRIKE_PRICE`

  行權價

- `BREAK_EVEN_POINT`

  打和點

- `MATURITY_TIME`

  到期日

- `LIST_TIME`

  上市日期

- `LAST_TRADE_TIME`

  最後交易日

- `LEVERAGE`

  槓桿比率

- `IN_OUT_MONEY`

  價內/價外 %

- `RECOVERY_PRICE`

  收回價 ℹ

- `CHANGE_PRICE`

換股價

- `CHANGE`

  換股比率

- `STREET_RATE`

  街貨比 %

- `STREET_VOL`

  街貨量

- `WARRANT_NAME`

  窩輪名稱

- `ISSUER`

  發行人

- `LOT_SIZE`

  每手

- `ISSUE_SIZE`

  發行量

- `UPPER_STRIKE_PRICE`

  上限價

- `LOWER_STRIKE_PRICE`

  下限價

- `INLINE_PRICE_STATUS`

  界內界外

- `PRE_CUR_PRICE`

  盤前最新價

- `AFTER_CUR_PRICE`

  盤後最新價
- `PRE_PRICE_CHANGE_VAL`

  盤前漲跌額
- `AFTER_PRICE_CHANGE_VAL`

  盤後漲跌額
- `PRE_CHANGE_RATE`

  盤前漲跌幅 %
- `AFTER_CHANGE_RATE`

  盤後漲跌幅 %
- `PRE_AMPLITUDE`

  盤前振幅 %
- `AFTER_AMPLITUDE`

  盤後振幅 %
- `PRE_TURNOVER`

  盤前成交額
- `AFTER_TURNOVER`

  盤後成交額
- `LAST_SETTLE_PRICE`

  昨結
- `POSITION`

  持倉量
- `POSITION_CHANGE`

日增倉

簡單過濾屬性

> StockField

- `NONE`

  未知

- `STOCK_CODE`

  股票代碼，不能填區間上下限值。

- `STOCK_NAME`

  股票名稱，不能填區間上下限值。

- `CUR_PRICE`

  最新價 ℹ

- `CUR_PRICE_TO_HIGHEST52_WEEKS_RATIO`

  (CP - WH52) / WH52
  CP：現價
  WH52：52 周最高
  對應 PC 端"離 52 周高點百分比" ℹ

- `CUR_PRICE_TO_LOWEST52_WEEKS_RATIO`

  (CP - WL52) / WL52
  CP：現價
  WL52：52 周最低
  對應 PC 端"離 52 周低點百分比" ℹ

- `HIGH_PRICE_TO_HIGHEST52_WEEKS_RATIO`

  (TH - WH52) / WH52
  TH：今日最高
  WH52：52 周最高
  ℹ

- `LOW_PRICE_TO_LOWEST52_WEEKS_RATIO`

  (TL - WL52) / WL52
  TL：今日最低
  WL52：52 周最低

- `VOLUME_RATIO`

  量比

- `BID_ASK_RATIO`

  委比

- `LOT_PRICE`

  每手價格

- `MARKET_VAL`

  市值

- `PE_ANNUAL`

  市盈率(靜態)

- `PE_TTM`

  市盈率 TTM

- `PB_RATE`

  市淨率

- `CHANGE_RATE_5MIN`

  五分鐘價格漲跌幅

- `CHANGE_RATE_BEGIN_YEAR`

  年初至今價格漲跌幅

- `PS_TTM`

市銷率 TTM

- `PCF_TTM`

  市現率 TTM

- `TOTAL_SHARE`

  總股數

- `FLOAT_SHARE`

  流通股數

- `FLOAT_MARKET_VAL`

  流通市值

訂閱類型

> SubType

- `NONE`

  未知

- `QUOTE`

  基礎報價

- `ORDER_BOOK`

  擺盤

- `TICKER`

  逐筆

- `RT_DATA`

  分時

- `K_DAY`

日 K

- `K_5M`

  5 分 K

- `K_15M`

  15 分 K

- `K_30M`

  30 分 K

- `K_60M`

  60 分 K

- `K_1M`

  1 分 K

- `K_WEEK`

  周 K

- `K_MON`

  月 K

- `BROKER`

  經紀隊列

- `K_QURATER`

  季 K

- `K_YEAR`

  年 K

- `K_3M`

  3 分 K

逐筆成交方向

> TickerDirect

- `NONE`

  未知

- `BUY`

  外盤

- `SELL`

  內盤

- `NEUTRAL`

  中性盤

逐筆成交類型

> TickerType

- `UNKNOWN`

  未知

- `AUTO_MATCH`

  自動對盤

- `LATE`

  開市前成交盤

- `NON_AUTO_MATCH`

  非自動對盤

- `INTER_AUTO_MATCH`

  同一證券商自動對盤

- `INTER_NON_AUTO_MATCH`

  同一證券商非自動對盤

- `ODD_LOT`

  碎股交易

- `AUCTION`

  競價交易

- `BULK`

  批量交易

- `CRASH`

  現金交易

- `CROSS_MARKET`

  跨市場交易

- `BULK_SOLD`

  批量賣出

- `FREE_ON_BOARD`

  離價交易

- `RULE127_OR155`

  第 127 條交易（紐交所規則）或第 155 條交易

- `DELAY`

  延遲交易

- `MARKET_CENTER_CLOSE_PRICE`

中央收市價

- `NEXT_DAY`

  隔日交易

- `MARKET_CENTER_OPENING`

  中央開盤價交易

- `PRIOR_REFERENCE_PRICE`

  前參考價

- `MARKET_CENTER_OPEN_PRICE`

  中央開盤價

- `SELLER`

  賣方

- `T`

  T 類交易（盤前和盤後交易）

- `EXTENDED_TRADING_HOURS`

  延長交易時段

- `CONTINGENT`

  合單交易

- `AVERAGE_PRICE`

  平均價成交

- `OTC_SOLD`

  場外售出

- `ODD_LOT_CROSS_MARKET`

  碎股跨市場交易

- `DERIVATIVELY_PRICED`

  衍生工具定價

- `REOPENINGP_RICED`

  再開盤定價

- `CLOSING_PRICED`

  收盤定價

- `COMPREHENSIVE_DELAY_PRICE`

  綜合延遲價格

- `OVERSEAS`

  交易的一方不是香港交易所的成員，屬於場外交易

交易日查詢市場

> TradeDateMarket

- `NONE`

  未知

- `HK`

  香港市場

- `US`

  美國市場

- `CN`

  A 股市場

- `NT`

  深（滬）股通

- `ST`

  港股通（深、滬）

- `JP_FUTURE`

  日本期貨

- `SG_FUTURE`

  新加坡期貨

交易日類型

> TradeDateType

- `WHOLE`

  全天交易

- `MORNING`

  上午交易，下午休市

- `AFTERNOON`

  下午交易，上午休市

窩輪狀態

> WarrantStatus

- `NONE`

  未知

- `NORMAL`

  正常狀態

- `SUSPEND`

停牌

- `STOP_TRADE`

  終止交易

- `PENDING_LISTING`

  等待上市

窩輪類型

> WrtType

- `NONE`

  未知

- `CALL`

  認購窩輪

- `PUT`

  認沽窩輪

- `BULL`

  牛證

- `BEAR`

  熊證

- `INLINE`

  界內證

所屬交易所

> ExchType

- `NONE`

  未知
- `HK_MAINBOARD`

  港交所·主板
- `HK_GEMBOARD`

  港交所·創業板
- `HK_HKEX`

  港交所
- `US_NYSE`

  紐交所
- `US_NASDAQ`

  納斯達克
- `US_PINK`

  OTC市場
- `US_AMEX`

  美交所
- `US_OPTION`

  美國
- `US_NYMEX`

  NYMEX
- `US_COMEX`

  COMEX
- `US_CBOT`

CBOT

- `US_CME`

  CME

- `US_CBOE`

  CBOE

- `CN_SH`

  上交所

- `CN_SZ`

  深交所

- `CN_STIB`

  科創板

- `SG_SGX`

  新交所

- `JP_OSE`

  大阪交易所

證券標識

Security

```
message Security
{
    required int32 market = 1; //QotMarket，行情市場
    required string code = 2; //代碼
}
```

K 線資料

KLine

```
message KLine
{
    required string time = 1; //時間戳字符串（格式：yyyy-MM-dd HH:mm:ss）
    required bool isBlank = 2; //是否是空內容的點,若為 true 則只有時間資訊
    optional double highPrice = 3; //最高價
    optional double openPrice = 4; //開盤價
    optional double lowPrice = 5; //最低價
    optional double closePrice = 6; //收盤價
    optional double lastClosePrice = 7; //昨收價
    optional int64 volume = 8; //成交量
    optional double turnover = 9; //成交額
    optional double turnoverRate = 10; //換手率（該欄位為百分比欄位，展示為小數表示）
    optional double pe = 11; //市盈率
    optional double changeRate = 12; //漲跌幅（該欄位為百分比欄位，預設不展示 %，如
    optional double timestamp = 13; //時間戳
}
```

基礎報價的期權特有欄位

OptionBasicQotExData

```
message OptionBasicQotExData
{
    required double strikePrice = 1; //行權價
    required int32 contractSize = 2; //每份合約數(整型資料)
    optional double contractSizeFloat = 17; //每份合約數（浮點型資料）
    required int32 openInterest = 3; //未平倉合約數
    required double impliedVolatility = 4; //隱含波動率（該欄位為百分比欄位，預設不
    required double premium = 5; //溢價（該欄位為百分比欄位，預設不展示 %，如 20 實際
    required double delta = 6; //希臘值 Delta
    required double gamma = 7; //希臘值 Gamma
    required double vega = 8; //希臘值 Vega
    required double theta = 9; //希臘值 Theta
    required double rho = 10; //希臘值 Rho
    optional int32 netOpenInterest = 11; //淨未平倉合約數，僅港股期權適用
    optional int32 expiryDateDistance = 12; //距離到期日天數，負數表示已過期
    optional double contractNominalValue = 13; //合約名義金額，僅港股期權適用
    optional double ownerLotMultiplier = 14; //相等正股手數，指數期權無該欄位，僅港
```

```
    optional int32 optionAreaType = 15; //OptionAreaType，期權類型（按行權時間）
    optional double contractMultiplier = 16; //合約乘數
    optional int32 indexOptionType = 18; //IndexOptionType，指數期權類型
}
```

基礎報價的期貨特有欄位

FutureBasicQotExData

```
message FutureBasicQotExData
{
    required double lastSettlePrice = 1; //昨結
    required int32 position = 2; //持倉量
    required int32 positionChange = 3; //日增倉
    optional int32 expiryDateDistance = 4; //距離到期日天數
}
```

基礎報價

BasicQot

```
message BasicQot
{
    required Security security = 1; //股票
    optional string name = 24; // 股票名稱
    required bool isSuspended = 2; //是否停牌
    required string listTime = 3; //上市日期字符串（此欄位停止維護，不建議使用，格式
    required double priceSpread = 4; //價差
    required string updateTime = 5; //最新價的更新時間字符串（格式：yyyy-MM-dd HH:m
    required double highPrice = 6; //最高價
    required double openPrice = 7; //開盤價
    required double lowPrice = 8; //最低價
    required double curPrice = 9; //最新價
    required double lastClosePrice = 10; //昨收價
    required int64 volume = 11; //成交量
    required double turnover = 12; //成交額
    required double turnoverRate = 13; //換手率（該欄位為百分比欄位，預設不展示 %，如
    required double amplitude = 14; //振幅（該欄位為百分比欄位，預設不展示 %，如 20
```

```
    optional int32 darkStatus = 15; //DarkStatus, 暗盤交易狀態
    optional OptionBasicQotExData optionExData = 16; //期權特有欄位
    optional double listTimestamp = 17; //上市日期時間戳（此欄位停止維護，不建議使用
    optional double updateTimestamp = 18; //最新價的更新時間戳，對其他欄位不適用
    optional PreAfterMarketData preMarket = 19; //盤前資料
    optional PreAfterMarketData afterMarket = 20; //盤後資料
    optional int32 secStatus = 21; //SecurityStatus, 股票狀態
    optional FutureBasicQotExData futureExData = 22; //期貨特有欄位
}
```

盤前盤後資料

PreAfterMarketData

```
//美股支援盤前盤後資料
//科創板僅支援盤後資料：成交量，成交額
message PreAfterMarketData
{
    optional double price = 1;  // 盤前或盤後## 價格
    optional double highPrice = 2;  // 盤前或盤後## 最高價
    optional double lowPrice = 3;  // 盤前或盤後## 最低價
    optional int64 volume = 4;  // 盤前或盤後## 成交量
    optional double turnover = 5;  // 盤前或盤後## 成交額
    optional double changeVal = 6;  // 盤前或盤後## 漲跌額
    optional double changeRate = 7;  // 盤前或盤後## 漲跌幅（該欄位為百分比欄位，預
    optional double amplitude = 8;  // 盤前或盤後## 振幅（該欄位為百分比欄位，預設不
}
```

分時資料

TimeShare

```
message TimeShare
{
    required string time = 1; //時間字串（格式：yyyy-MM-dd HH:mm:ss）
    required int32 minute = 2; //距離0點過了多少分鐘
    required bool isBlank = 3; //是否是空內容的點,若為 true 則只有時間資訊
    optional double price = 4; //當前價
```

```
    optional double lastClosePrice = 5; //昨收價
    optional double avgPrice = 6; //均價
    optional int64 volume = 7; //成交量
    optional double turnover = 8; //成交額
    optional double timestamp = 9; //時間戳
}
```

證券基本靜態資訊

SecurityStaticBasic

```

message SecurityStaticBasic
{
    required Qot_Common.Security security = 1; //股票
    required int64 id = 2; //股票 ID
    required int32 lotSize = 3; //每手數量,期權類型表示一份合約的股數
    required int32 secType = 4; //Qot_Common.SecurityType,股票類型
    required string name = 5; //股票名字
    required string listTime = 6; //上市時間字串（此欄位停止維護，不建議使用，格式：y
    optional bool delisting = 7; //是否退市
    optional double listTimestamp = 8; //上市時間戳（此欄位停止維護，不建議使用）
    optional int32 exchType = 9; //Qot_Common.ExchType,所屬交易所
}
```

窩輪額外靜態資訊

WarrantStaticExData

```
message WarrantStaticExData
{
    required int32 type = 1; //Qot_Common.WarrantType,窩輪類型
    required Qot_Common.Security owner = 2; //所屬正股
}
```

期權額外靜態資訊

OptionStaticExData

```
message OptionStaticExData
{
    required int32 type = 1; //Qot_Common.OptionType,期權
    required Qot_Common.Security owner = 2; //標的股
    required string strikeTime = 3; //行權日（格式：yyyy-MM-dd）
    required double strikePrice = 4; //行權價
    required bool suspend = 5; //是否停牌
    required string market = 6; //發行市場名字
    optional double strikeTimestamp = 7; //行權日時間戳
    optional int32 indexOptionType = 8; //Qot_Common.IndexOptionType, 指數期權的類
     optional int32 expirationCycle = 9; // ExpirationCycle，交割週期
    optional int32 optionStandardType = 10; // OptionStandardType，標準期權
    optional int32 optionSettlementMode = 11; // OptionSettlementMode，結算方式
}
```

期貨額外靜態資訊

FutureStaticExData

```
message FutureStaticExData
{
    required string lastTradeTime = 1; //最後交易日，只有非主連期貨合約才有該欄位
    optional double lastTradeTimestamp = 2; //最後交易日時間戳，只有非主連期貨合約才
    required bool isMainContract = 3; //是否主連合約
}
```

證券靜態資訊

SecurityStaticInfo

```
message SecurityStaticInfo
{
    required SecurityStaticBasic basic = 1; //證券基本靜態資訊
    optional WarrantStaticExData warrantExData = 2; //窩輪額外靜態資訊
    optional OptionStaticExData optionExData = 3; //期權額外靜態資訊
    optional FutureStaticExData futureExData = 4; //期貨額外靜態資訊
}
```

買賣經紀

Broker

```
message Broker
{
    required int64 id = 1; //經紀 ID
    required string name = 2; //經紀名稱
    required int32 pos = 3; //經紀檔位

    //以下為港股 SF 行情特有欄位
    optional int64 orderID = 4; //交易所訂單 ID，與交易介面返回的訂單 ID 並不一樣
    optional int64 volume = 5; //訂單股數
}
```

逐筆成交

Ticker

```
message Ticker
{
    required string time = 1; //時間字串（格式：yyyy-MM-dd HH:mm:ss）
    required int64 sequence = 2; // 唯一標識
    required int32 dir = 3; //TickerDirection, 買賣方向
    required double price = 4; //價格
    required int64 volume = 5; //成交量
    required double turnover = 6; //成交額
    optional double recvTime = 7; //收到推送資料的本地時間戳，用於定位延遲
```

```
    optional int32 type = 8; //TickerType, 逐筆類型
    optional int32 typeSign = 9; //逐筆類型符號
    optional int32 pushDataType = 10; //用於區分推送情況，僅推送時有該欄位
    optional double timestamp = 11; //時間戳
}
```

買賣檔明細

OrderBookDetail

```
message OrderBookDetail
{
    required int64 orderID = 1; //交易所訂單 ID，與交易介面返回的訂單 ID 並不一樣
    required int64 volume = 2; //訂單股數
}
```

買賣檔

OrderBook

```
message OrderBook
{
    required double price = 1; //委託價格
    required int64 volume = 2; //委託數量
    required int32 orederCount = 3; //委託訂單個數
    repeated OrderBookDetail detailList = 4; //訂單資訊，港股 SF，美股深度擺盤特有
}
```

持股變動

ShareHoldingChange

```
message ShareHoldingChange
{
```

```
    required string holderName = 1; //持有者名稱（機構名稱 或 基金名稱 或 高管姓名）
    required double holdingQty = 2; //當前持股數量
    required double holdingRatio = 3; //當前持股百分比（該欄位為百分比欄位，預設不展
    required double changeQty = 4; //較上一次變動數量
    required double changeRatio = 5; //較上一次變動百分比（該欄位為百分比欄位，預設不
    required string time = 6; //發佈時間（格式：yyyy-MM-dd HH:mm:ss）
    optional double timestamp = 7; //時間戳
}
```

單個訂閱類型資訊

SubInfo

```
message SubInfo
{
    required int32 subType = 1;  //Qot_Common.SubType,訂閱類型
    repeated Qot_Common.Security securityList = 2;  //訂閱該類型行情的證券
}
```

單條連線訂閱資訊

ConnSubInfo

```
message ConnSubInfo
{
    repeated SubInfo subInfoList = 1; //該連線訂閱資訊
    required int32 usedQuota = 2; //該連線已經使用的訂閱額度
    required bool isOwnConnData = 3; //用於區分是否是自己連線的資料
}
```

板塊資訊

PlateInfo

```
message PlateInfo
{
    required Qot_Common.Security plate = 1; //板塊
    required string name = 2; //板塊名字
    optional int32 plateType = 3; //PlateSetType 板塊類型, 僅3207（獲取股票所屬板塊
}
```

復權資訊

Rehab

```
message Rehab
{
    required string time = 1; //時間字串（格式：yyyy-MM-dd）
    required int64 companyActFlag = 2; //公司行動(CompanyAct)組合標誌位,指定某些欄位
    required double fwdFactorA = 3; //前復權因子 A
    required double fwdFactorB = 4; //前復權因子 B
    required double bwdFactorA = 5; //後復權因子 A
    required double bwdFactorB = 6; //後復權因子 B
    optional int32 splitBase = 7; //拆股(例如，1拆5，Base 為1，Ert 為5)
    optional int32 splitErt = 8;
    optional int32 joinBase = 9; //合股(例如，50合1，Base 為50，Ert 為1)
    optional int32 joinErt = 10;
    optional int32 bonusBase = 11; //送股(例如，10送3, Base 為10,Ert 為3)
    optional int32 bonusErt = 12;
    optional int32 transferBase = 13; //轉贈股(例如，10轉3, Base 為10,Ert 為3)
    optional int32 transferErt = 14;
    optional int32 allotBase = 15; //配股(例如，10送2, 配股價為6.3元, Base 為10, Er
    optional int32 allotErt = 16;
    optional double allotPrice = 17;
    optional int32 addBase = 18; //增發股(例如，10送2, 增發股價為6.3元, Base 為10,
    optional int32 addErt = 19;
    optional double addPrice = 20;
    optional double dividend = 21; //現金分紅(例如，每10股派現0.5元,則該欄位值為0.05
    optional double spDividend = 22; //特別股息(例如，每10股派特別股息0.5元,則該欄位
    optional double timestamp = 23; //時間戳
}
```

> - 公司行動組合標誌位參見 CompanyAct

交割週期

> ExpirationCycle

- `NONE`

  未知

- `WEEK`

  週期權

- `MONTH`

  月期權

- `END_OF_MONTH`

  月末期權

- `QUARTERLY`

  季期權

- `WEEKMON`

  週期權-週一

- `WEEKTUE`

  週期權-週二

- `WEEKWED`

  週期權-週三

- `WEEKTHU`

  週期權-週四

- `WEEKFRI`

  週期權-週五

期權標準類型

> OptionStandardType

- `NONE`

  未知

- `STANDARD`

  標準期權

- `NON_STANDARD`

  非標準期權

期權結算方式

> OptionSettlementMode

- `NONE`

  未知

- `AM`

  亞式期權

- `PM`

  路徑依賴型

股票持有者（已廢棄）

> StockHolder

- `NONE`

  未知
- `INSTITUTE`

  機構
- `FUND`

  基金
- `EXECUTIVE`

  高管

交易介面總覽

模組	介面名	功能簡介
帳戶	Get Account List	讀取交易業務帳戶列表
	Unlock Trading	解鎖交易
資產持倉	Get Account Financial Information	讀取帳戶資金數據
	Get Maximum Tradable Quantity	查詢帳戶最大可買賣數量
	Get Positions List	讀取持倉列表
	Get Margin Trading Data	讀取融資融券數據
	Get Cash Flow Summary	查詢帳戶現金流紀錄
訂單	Place Order	下單
	Modify or Cancel Order	改單撤單
	Get Order list	查詢未完成訂單
	Get Order Fees	查詢訂單費用
	Get Historical Order List	查詢歷史訂單
	Order Callback	訂單回呼
	Trade Data Callback	訂閱交易推送
成交	Get Today's Executed Trades	查詢當日成交
	Get Historical Executed Trades	查詢歷史成交
	Trade Execution Callback	成交回呼

交易物件

建立連接

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

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

- **介紹**

  根據交易品類，選擇帳戶，並創建對應的交易物件。

實例	帳戶
OpenSecTradeContext	證券帳戶
OpenFutureTradeContext	期貨帳戶

- **參數**

參數	類型	說明
filter_trdmarket	TrdMarket	篩選對應交易市場權限的帳戶
host	str	OpenD 監聽的 IP 地址
port	int	OpenD 監聽的 IP 端口
is_encrypt	bool	是否啟用加密
security_firm	SecurityFirm	所屬券商

- **Example**

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
trd_ctx.close() # 結束後記得關閉當條連接，防止連接條數用盡
```

關閉連接

`close()`

- 介紹

  關閉交易物件。預設情況下，moomoo API 內部創建的執行緒會阻止程序退出，只有當所有 Context 都 close 後，程序才能正常退出。但通過 set_all_thread_daemon 可以設置所有內部執行緒為 daemon 執行緒，這時即使沒有調用 Context 的 close，程序也可以正常退出。

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
trd_ctx.close()  # 結束後記得關閉當條連接，防止連接條數用盡
```

讀取交易業務賬户列表

`get_acc_list()`

- **介紹**

  讀取交易業務賬户列表。
  要執行其他交易介面前，請先讀取此列表，確認要操作的交易業務賬户無誤。

- **參數**

- **返回**

參數	類型	說明
ret	RET_CODE	接口調用結果
data	pd.DataFrame	當 ret == RET_OK 時，返回交易業務賬户列表
	str	當 ret != RET_OK 時，返回錯誤描述

  - 交易業務賬户列表格式如下：

欄位	類型	說明
acc_id	int	交易業務賬户
trd_env	TrdEnv	交易環境
acc_type	TrdAccType	賬户類型
uni_card_num	str	綜合賬户卡號，同流動裝置內的展示
card_num	str	業務賬户卡號
security_firm	SecurityFirm	所屬券商
sim_acc_type	SimAccType	模擬賬户類型

欄位	類型	說明
trdmarket_auth	list	交易市場權限
acc_status	TrdAccStatus	賬户狀態
acc_role	TrdAccRole	賬户結構
jp_acc_type	list	日本賬户類型

- **說明**

  當開通了港/美股期權模擬交易後，此接口在讀取港/美交易賬號列表時，會返回2個模擬交易賬號。其中第1個為原先的賬號，第2個是期權模擬交易賬號。 目前OpenAPI拉取的美股模擬交易賬號和流動裝置不是同一個賬户，點擊這裏瞭解更多。

- **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])  # 取第一個賬號
    print(data['acc_id'].values.tolist())  # 轉為 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]
```

解鎖交易

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

- **介紹**

  解鎖或鎖定交易

- **參數**

參數	類型	說明
password	str	交易密碼
password_md5	str	交易密碼的 32 位 MD5 加密（全小寫）
is_unlock	bool	解鎖或鎖定

- **回傳**

參數	類型	說明
ret	RET_CODE	介面執行結果
msg	NoneType	當 ret == RET_OK 時，回傳 None
	str	當 ret != RET_OK 時，回傳錯誤描述

- **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:
```

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

- **Output**

```
    unlock success!
```

> **提示**
>
> - 真實帳戶調用 下單 或 改單撤單 介面，需要先解鎖交易；模擬帳戶無需解鎖。
> - 解鎖或鎖定交易，是針對 OpenD 的操作，只要有一個連線解鎖，其他連線都可以調用交易介面。
> - 強烈建議，通過外網連線 OpenD 進行實盤交易的客戶，使用加密通道，參見 啟用協議加密。
> - OpenAPI 不支援富途令牌，如果開通了富途令牌，則會解鎖失敗，需要關閉令牌功能後再使用 OpenAPI 解鎖。

> **介面限制**
>
> - 單用戶ID 每 30 秒內最多請求 10 次解鎖交易介面

查詢帳戶資金

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

- **介紹**

  查詢交易業務帳戶的資產淨值、證券市值、現金、購買力等資金數據。

- **參數**

參數	類型	說明
trd_env	TrdEnv	交易環境
acc_id	int	交易業務帳戶 ID
acc_index	int	交易業務帳戶列表中的帳戶序號
refresh_cache	bool	是否更新快取
currency	Currency	資金的展示貨幣
asset_category	AssetCategory	資產類別

- **返回**

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，返回資金數據
	str	當 ret != RET_OK 時，返回錯誤描述

  - 資金數據格式如下：

字段	類型	說明
power	float	最大購買力
max_power_short	float	賣空購買力
net_cash_power	float	現金購買力
total_assets	float	總資產淨值
securities_assets	float	證券資產淨值
fund_assets	float	基金資產淨值
bond_assets	float	債券資產淨值
cash	float	現金
market_val	float	證券市值
long_mv	float	長倉市值
short_mv	float	短倉市值
pending_asset	float	未交收資產
interest_charged_amount	float	計息金額
frozen_cash	float	凍結資金
avl_withdrawal_cash	float	現金可提
max_withdrawal	float	最大可提
currency	Currency	計價貨幣
available_funds	float	可用資金
unrealized_pl	float	未實現盈虧
realized_pl	float	已實現盈虧

字段	類型	說明
risk_level	CltRiskLevel	風險管理狀態
risk_status	CltRiskStatus	風險狀態
initial_margin	float	初始保證金
margin_call_margin	float	Margin Call 保證金
maintenance_margin	float	維持保證金
hk_cash	float	港元現金
hk_avl_withdrawal_cash	float	港元可提
hkd_net_cash_power	float	港元現金購買力
hkd_assets	float	港股資產淨值
us_cash	float	美元現金
us_avl_withdrawal_cash	float	美元可提
usd_net_cash_power	float	美元現金購買力
usd_assets	float	美股資產淨值
cn_cash	float	人民幣現金
cn_avl_withdrawal_cash	float	人民幣可提
cnh_net_cash_power	float	人民幣現金購買力
cnh_assets	float	A股資產淨值
jp_cash	float	日元現金
jp_avl_withdrawal_cash	float	日元可提
jpy_net_cash_power	float	日元現金購買力

字段	類型	說明
jpy_assets	float	日股資產淨值
sg_cash	float	新元現金
sg_avl_withdrawal_cash	float	新元可提
sgd_net_cash_power	float	新元現金購買力
sgd_assets	float	新股資產淨值
au_cash	float	澳元現金
au_avl_withdrawal_cash	float	澳元可提
aud_net_cash_power	float	澳元現金購買力
aud_assets	float	澳股資產淨值
ca_cash	float	加元現金
ca_avl_withdrawal_cash	float	加元可提
cad_net_cash_power	float	加元現金購買力
cad_assets	float	加元資產淨值
my_cash	float	令吉現金
my_avl_withdrawal_cash	float	令吉可提
myr_net_cash_power	float	令吉現金購買力
myr_assets	float	令吉資產淨值
is_pdt	bool	是否為 PDT 帳戶
pdt_seq	string	剩餘日內交易次數
beginning_dtbp	float	初始日內交易購買力

字段	類型	說明
remaining_dtbp	float	剩餘日內交易購買力
dt_call_amount	float	日內交易待繳金額
dt_status	DtStatus	日內交易限制情況

- 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])  # 取第一行的購買力
    print(data['power'].values.tolist())  # 轉為 list
else:
    print('accinfo_query error: ', data)
trd_ctx.close()  # 關閉當條連接
```

- 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]
```

接口限制

- 同一帳戶ID(acc_id) 每 30 秒內最多請求 10 次查詢帳戶資金接口
- 調用此接口，只有在刷新緩存時，才受到限頻限制

查詢最大可買可賣

```
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)
```

- **介紹**

  查詢指定交易業務帳戶下的最大可買賣數量，亦可查詢指定交易業務帳戶下指定訂單的最大可改成的數量。

  現金帳戶請求期權不適用。

- **參數**

參數	類型	說明
order_type	OrderType	訂單類型
code	str	證券代碼
price	float	報價
order_id	str	訂單號
adjust_limit	float	價格微調幅度
trd_env	TrdEnv	交易環境
acc_id	int	交易業務帳戶 ID
acc_index	int	交易業務帳戶列表中的帳戶序號
session	Session	美股交易時段
jp_acc_type	SubAccType	日本帳戶類型
position_id	int	持倉ID

- **回傳**

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，回傳賬號列表
	str	當 ret != RET_OK 時，回傳錯誤描述

  - 賬號列表格式如下：

欄位	類型	說明
max_cash_buy	float	現金可買
max_cash_and_margin_buy	float	最大可買
max_position_sell	float	持倉可賣
max_sell_short	float	可賣空
max_buy_back	float	平倉需買入
long_required_im	float	買 1 張合約所帶來的初始保證金變動。
short_required_im	float	賣 1 張合約所帶來的初始保證金變動。
session	Session	交易訂單時段（僅用於美股）

- **Example**

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
ret, data = trd_ctx.acctradinginfo_query(order_type=OrderType.NORMAL, code='US.A
if ret == RET_OK:
    print(data)
    print(data['max_cash_and_margin_buy'][0])  # 最大融資可買數量
else:
```

```
    print('acctradinginfo_query error: ', data)
trd_ctx.close()  # 關閉當條連接
```

- **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
```

> **介面限制**
>
> - 同一帳戶ID(acc_id) 每 30 秒內最多請求 10 次查詢最大可買可賣介面

> **提示**
>
> - 現金業務帳戶不支援交易衍生品，因此不支援通過現金業務帳戶查詢期權的最大可買可賣。
> - 期貨的最大可買，需自行計算，公式：floor(最大購買力/買 1 張合約所帶來的初始保證金變動)。其中，最大購買力來自查詢帳戶資金，買 1 張合約所帶來的初始保證金變動來自本介面。

查詢持倉

```
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)
```

- **介紹**

  查詢交易業務賬户的持倉列表

- **參數**

參數	類型	說明
code	str	代碼過濾
position_market	TrdMarket	持倉所屬市場過濾
pl_ratio_min	float	當前盈虧比例下限過濾，僅回傳高於此比例的持倉
pl_ratio_max	float	當前盈虧比例上限過濾，低於此比例的會回傳
trd_env	TrdEnv	交易環境
acc_id	int	交易業務賬户 ID
acc_index	int	交易業務賬户列表中的賬户序號
refresh_cache	bool	是否更新快取
asset_category	AssetCategory	資產類別

- **回傳**

參數	類型	說明

ret	RET_CODE	接口執行結果
data	pd.DataFrame	當 ret == RET_OK 時，回傳持倉列表
	str	當 ret != RET_OK 時，回傳錯誤描述

- 持倉列表

欄位	類型	說明
position_side	PositionSide	持倉方向
code	str	股票編號
stock_name	str	股票名稱
position_market	TrdMarket	持倉所屬市場
qty	float	持有數量
can_sell_qty	float	可用數量
currency	Currency	交易貨幣
nominal_price	float	市價
cost_price	float	攤薄成本價（證券賬户），平均開倉價（期貨賬户）
cost_price_valid	bool	成本價是否有效
average_cost	float	平均成本價
diluted_cost	float	攤薄成本價
market_val	float	市值
pl_ratio	float	盈虧比例（攤薄成本價模式）
pl_ratio_valid	bool	盈虧比例是否有效

欄位	類型	說明
pl_ratio_avg_cost	float	盈虧比例（平均成本價模式）
pl_val	float	盈虧金額
pl_val_valid	bool	盈虧金額是否有效
today_pl_val	float	今日盈虧金額
today_trd_val	float	今日交易金額
today_buy_qty	float	今日買入總量
today_buy_val	float	今日買入總額
today_sell_qty	float	今日賣出總量
today_sell_val	float	今日賣出總額
unrealized_pl	float	未實現盈虧
realized_pl	float	已實現盈虧
position_id	int	持倉ID

- **Example**

```
from futu import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
ret, data = trd_ctx.position_list_query()
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # 如果持倉列表不為空
        print(data['stock_name'][0])  # 獲取持倉第一個股票名稱
        print(data['stock_name'].values.tolist())  # 轉為 list
else:
    print('position_list_query error: ', data)
trd_ctx.close()  # 關閉當條連接
```

- **Output**

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

接口限制

- 同一賬户ID(acc_id) 每 30 秒內最多請求 10 次查詢持倉接口
- 執行此接口，只有在更新快取時，才受到限頻限制

讀取融資融券數據

`get_margin_ratio(code_list)`

- **介紹**

  查詢股票的融資融券數據。

- **參數**

參數	類型	說明
code_list	list	股票代碼列表

- **傳回**

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，傳回融資融券數據
	str	當 ret != RET_OK 時，傳回錯誤描述

  - 融資融券數據格式如下：

欄位	類型	說明
code	str	股票代碼
is_long_permit	bool	是否允許融資
is_short_permit	bool	是否允許融券
short_pool_remain	float	賣空池剩餘
short_fee_rate	float	融券參考利率

欄位	類型	說明
alert_long_ratio	float	融資預警比率
alert_short_ratio	float	融券預警比率
im_long_ratio	float	融資初始保證金率
im_short_ratio	float	融券初始保證金率
mcm_long_ratio	float	融資 margin call 保證金率
mcm_short_ratio	float	融券 margin call 保證金率
mm_long_ratio	float	融資維持保證金率
mm_short_ratio	float	融券維持保證金率

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
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])  # 取第一條的是否允許融資
    print(data['im_short_ratio'].values.tolist())  # 轉為 list
else:
    print('error:', data)
trd_ctx.close()  # 結束後記得關閉當條連接，防止連接條數用盡
```

- 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]
```

接口限制

- 單用戶ID 每 30 秒內最多請求 10 次讀取融資融券數據接口。
- 每次請求，接口參數股票代碼列表，支援傳入的標的數量上限是 100 個。
- 支援美國、香港、A股市場的股票和ETF。

查詢帳戶現金流紀錄

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

- **介紹**

  查詢交易業務帳戶在指定日期的現金流紀錄數據。數據覆蓋存款及提款、調撥、貨幣兌換、買賣金融資產、融資融券利息等所有導致現金變動的事項。

- **參數**

參數	類型	說明
clearing_date	str	清算日期
trd_env	TrdEnv	交易環境
acc_id	int	交易業務帳戶 ID
acc_index	int	交易業務帳戶列表中的帳戶序號
cashflow_direction	CashFlowDirection	篩選現金流方向

- **返回**

參數	類型	說明
ret	RET_CODE	API 執行結果
data	pd.DataFrame	當 ret == RET_OK 時，返回交易業務帳戶現金流紀錄列表格式
	str	當 ret != RET_OK 時，傳回錯誤描述

  - 交易業務帳戶現金流紀錄列表格式如下：

欄位	類型	說明
cashflow_id	int	現金流唯一標識
clearing_date	str	清算日期
settlement_date	str	交收日期
currency	Currency	貨幣種類
cashflow_type	str	現金流類型
cashflow_direction	CashFlowDirection	現金流方向
cashflow_amount	float	金額（正數表示流入，負數表示流出）
cashflow_remark	str	備註

- 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:  # 如果現金流紀錄列表不為空
        print(data['cashflow_type'][0])  # 獲取第一條流水的現金流類型
        print(data['cashflow_amount'].values.tolist())  # 轉為 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            其他
1  16357           2025-02-27        2025-03-03          HKD            其他
2  16360           2025-02-27        2025-02-27          USD            基金贖回
```

```
3  16384          2025-02-27       2025-02-27          HKD          基金贖回
其他
[0.00, -104000.00, 23000.00, 104108.96]
```

接口限制

- 同一帳戶ID(acc_id) 每 30 秒內最多請求 20 次現金流紀錄接口。
- 現金流紀錄，按照時間的“順序”進行排列。
- 模擬交易和 moomoo US 帳戶暫不支援查詢現金流紀錄。

下單

```
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)
```

- **介紹**

  下單

  > **提示**
  >
  > Python API 是同步的，但網絡收發是非同步的。當 place_order 對應的應答數據包與 響應成交推送回呼 或 響應訂單推送回呼 間隔很短時，就可能出現 place_order 的數據包先返回，但回呼函數先被執行的情況。例如：可能先執行了 響應訂單推送回呼，然後 place_order 這個介面才返回。

- **參數**

參數	類型	說明
price	float	訂單價格
qty	float	訂單數量
code	str	標的代碼
trd_side	TrdSide	交易方向
order_type	OrderType	訂單類型
adjust_limit	float	價格微調幅度
trd_env	TrdEnv	交易環境

參數	類型	說明
acc_id	int	交易業務賬户 ID
acc_index	int	交易業務賬户列表中的賬户序號
remark	str	備註
time_in_force	TimeInForce	有效期限
fill_outside_rth	bool	是否允許盤前盤後
aux_price	float	觸發價格
trail_type	TrailType	跟蹤類型
trail_value	float	跟蹤金額/百分比
trail_spread	float	指定價差
session	Session	美股交易時段
jp_acc_type	SubAccType	日本賬户類型
position_id	int	持倉ID

- **返回**

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，返回訂單列表
	str	當 ret != RET_OK 時，返回錯誤描述

  - 訂單列表格式如下：

欄位	類型	說明
trd_side	TrdSide	交易方向
order_type	OrderType	訂單類型
order_status	OrderStatus	訂單狀態
order_id	str	訂單號
code	str	股票代碼
stock_name	str	股票名稱
qty	float	訂單數量
price	float	訂單價格
create_time	str	創建時間
updated_time	str	最後更新時間
dealt_qty	float	成交數量
dealt_avg_price	float	成交均價
last_err_msg	str	最後的錯誤描述
remark	str	下單時備註的標記
time_in_force	TimeInForce	有效期限
fill_outside_rth	bool	是否允許盤前盤後（用於港股盤前競價與美股盤前盤後）
aux_price	float	觸發價格
trail_type	TrailType	跟蹤類型
trail_value	float	跟蹤金額/百分比

欄位	類型	說明
trail_spread	float	指定價差
session	Session	交易訂單時段（僅用於美股）

- 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)  # 若使用真實賬户下單，需先對賬户進行解
if ret == RET_OK:
    ret, data = trd_ctx.place_order(price=510.0, qty=100, code="US.AAPL", trd_sid
    if ret == RET_OK:
        print(data)
        print(data['order_id'][0])  # 獲取下單的訂單號
        print(data['order_id'].values.tolist())  # 轉為 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        蘋果        BUY     NORMAL   SUBMITTING  381960065487095000  100
381960065487095000
['381960065487095000']
```

> **介面限制**
>
> - 同一賬户ID(acc_id) 每 30 秒內最多請求 15 次下單介面，且連續兩次請求的間隔不可小於 0.02 秒。
> - 真實賬户執行下單介面前，需要先進行 解鎖；模擬賬户無需解鎖。

提示

- 各訂單類型對應的必傳遞參數數：點擊這裏 瞭解更多
- 各券商針對不同交易品種，對單筆訂單股數有所限制，超出限制會導致下單失敗：點擊這裏 瞭解更多
- 對於 **可做空標的**，暫不支援鎖倉功能，故無法同時持有相同產品的多頭頭寸和空頭頭寸。
- 如果希望對 **可做空標的** 進行 **平倉** 操作，需要自行判斷持倉頭寸的方向，然後提交一筆反向的相同數量的訂單完成平倉操作。
- 如果希望對 **可做空標的** 進行 **反手** 操作，需要兩步：1. 先判斷持倉頭寸的方向，並提交一筆反向的相同數量的訂單完成平倉操作；2. 提交一筆反向的訂單，完成反向訂單的提交。

  舉例：A 當前持有 1 手 HK.HSI2012 期貨合約的多單，如果希望反手，必須先 賣出 1 手 HK.HSI2012 完成平倉，再賣出 1 手 HK.HSI2012 完成空單的建立。
- 美股全時段交易，僅支援限價單，訂單期限可以選擇當日有效或撤單前有效。選擇全時段，交易者可以一次掛單參與多個時段（夜盤、盤前、盤中、盤後時段）的交易，全時段交易時間是星期日到星期四 20:00 - 次日20:00（美東時間）
- 美股模擬交易不支援盤前盤後與夜盤。

改單撤單

```
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)
```

- **介紹**

  修改訂單的價格和數量、撤單、操作訂單的失效和生效、刪除訂單等。
  如果是 A 股通市場，將不支援改單。可撤單。刪除訂單是 OpenD 本地操作。

- **參數**

參數	類型	說明
modify_order_op	ModifyOrderOp	改單操作類型
order_id	str	訂單號
qty	float	訂單改單後的數量
price	float	訂單改單後的價格
adjust_limit	float	價格微調幅度
trd_env	TrdEnv	交易環境
acc_id	int	交易業務帳戶 ID
acc_index	int	交易業務帳戶列表中的帳戶序號
aux_price	float	觸發價格
trail_type	TrailType	跟蹤類型
trail_value	float	跟蹤金額/百分比
trail_spread	float	指定價差

- 返回

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，返回改單資訊
	str	當 ret != RET_OK 時，返回錯誤描述

  - 改單資訊格式如下：

欄位	類型	說明
trd_env	TrdEnv	交易環境
order_id	str	訂單號

- 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)  # 若使用真實帳戶改單/撤單，需先對帳戶
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['order_id'][0])  # 獲取改單的訂單號
        print(data['order_id'].values.tolist())  # 轉為 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)`

- **介紹**

  撤消全部訂單。模擬交易以及 A 股通帳戶暫不支援全部撤單。

- **參數**

參數	類型	說明
trd_env	TrdEnv	交易環境
acc_id	int	交易業務帳戶 ID
acc_index	int	交易業務帳戶列表中的帳戶序號
trdmarket	TrdMarket	指定交易市場

- **返回**

參數	類型	說明
ret	str	介面執行結果。ret == RET_OK 代表介面執行正常，ret != RET_OK 代表介面執行失敗
data	str	當 ret == RET_OK，返回"success"
		當 ret != RET_OK，返回錯誤描述

  - 全部撤單資訊格式如下：

欄位	類型	說明
trd_env	TrdEnv	交易環境
order_id	str	訂單號

- 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)  # 若使用真實帳戶改單/撤單，需先對帳戶
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
```

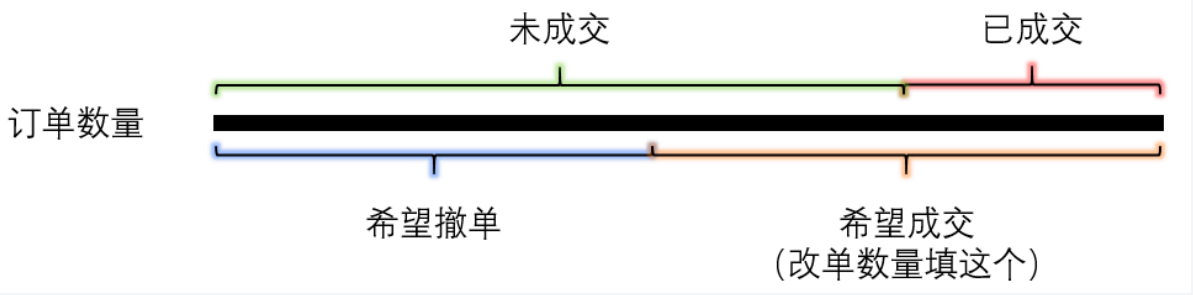
介面限制

- 同一帳戶ID(acc_id) 每 30 秒內最多請求 20 次改單撤單介面，且連續兩次請求的間隔不可小於 0.04 秒。
- 真實帳戶執行改單撤單介面前，需要先進行 解鎖；模擬帳戶無需解鎖。

提示

- 若執行 **修改訂單** 操作，各類訂單類型對應的必傳參數，可 點擊這裏 瞭解更多。
- 如果希望執行 **改單操作** 去 **修改訂單數量**，此介面入參的訂單數量 **qty**，應該等於期望成交的總數量。
  舉例： 一筆訂單數量是 N 股，已部分成交 n 股。對於暫未成交的 (N-n) 股，如果您

希望撤掉其中的 x 股，**modify_order_op** 應選擇 NORMAL，**qty** 應傳 (N-x)。



- 如果希望執行 **撤單操作**，此介面入參的 **modify_order_op** 應該選擇 CANCEL。
  舉例： 一筆訂單數量是 N 股，已部分成交 n 股。如果希望將未成交的 (N-n) 股全部撤掉，modify_order_op 應選擇 CANCEL，此時 qty 和 price 的入參會被忽略。

查詢未完成訂單

```
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)
```

- **介紹**

  查詢指定交易業務賬户的未完成訂單列表

- **參數**

參數	類型	說明
order_id	str	訂單號過濾
order_market	TrdMarket	訂單標的所屬市場過濾
status_filter_list	list	訂單狀態過濾
code	str	代碼過濾
start	str	開始時間
end	str	結束時間
trd_env	TrdEnv	交易環境
acc_id	int	交易業務賬户 ID
acc_index	int	交易業務賬户列表中的賬户序號
refresh_cache	bool	是否更新快取

- **回傳**

參數	類型	說明

ret	RET_CODE	介面調用結果
data	pd.DataFrame	當 ret == RET_OK 時，回傳訂單列表
	str	當 ret != RET_OK 時，回傳錯誤描述

- 訂單列表格式如下：

欄位	類型	說明
trd_side	TrdSide	交易方向
order_type	OrderType	訂單類型
order_status	OrderStatus	訂單狀態
order_id	str	訂單號
code	str	股票編號
stock_name	str	股票名稱
order_market	TrdMarket	訂單標的所屬市場
qty	float	訂單數量
price	float	訂單價格
currency	Currency	交易貨幣
create_time	str	創建時間
updated_time	str	最後更新時間
dealt_qty	float	成交數量
dealt_avg_price	float	成交均價
last_err_msg	str	最後的錯誤描述
remark	str	下單時備註的標識

欄位	類型	說明
time_in_force	TimeInForce	有效期限
fill_outside_rth	bool	是否允許盤前盤後（用於港股盤前競價與美股盤前盤後）
session	Session	交易訂單時段（僅用於美股）
aux_price	float	觸發價格
trail_type	TrailType	跟蹤類型
trail_value	float	跟蹤金額/百分比
trail_spread	float	指定價差
jp_acc_type	SubAccType	日本賬户類型

- 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:  # 如果訂單列表不為空
        print(data['order_id'][0])  # 獲取未完成訂單的第一個訂單號
        print(data['order_id'].values.tolist())  # 轉為 list
else:
    print('order_list_query error: ', data)
trd_ctx.close()
```

- Output

```
        code stock_name   order_amrket      trd_side           order_type   order
0   US.AAPL         US          BUY           NORMAL  CANCELLED_ALL  664446861527
6644468615272262086
['6644468615272262086']
```

介面限制

- 同一賬户ID(acc_id) 每 30 秒內最多請求 10 次查詢未完成訂單介面
- 調用此介面，只有在更新快取時，才受到限頻限制

提示

- 未完成訂單，按照時間的“順序”進行排列，即：先提交的訂單在前，後提交的訂單在後

查詢歷史訂單

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

- **介紹**

  查詢指定交易業務帳戶的歷史訂單列表

- **參數**

參數	類型	說明
status_filter_list	list	訂單狀態過濾
code	str	代碼過濾
order_market	TrdMarket	訂單標的所屬市場過濾
start	str	開始時間
end	str	結束時間
trd_env	TrdEnv	交易環境
acc_id	int	交易業務帳戶 ID
acc_index	int	交易業務帳戶列表中的帳戶序號

  - start 和 end 的組合如下

Start 類型	End 類型	說明
str	str	start 和 end 分別為指定的日期
None	str	start 為 end 往前 90 天

Start 類型	End 類型	說明
str	None	end 為 start 往後 90 天
None	None	start 為往前 90 天，end 當前日期

- **返回**

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，返回訂單列表
	str	當 ret != RET_OK 時，返回錯誤描述

  - 訂單列表格式如下：

欄位	類型	說明
trd_side	TrdSide	交易方向
order_type	OrderType	訂單類型
order_status	OrderStatus	訂單狀態
order_id	str	訂單號
code	str	股票編號
stock_name	str	股票名稱
order_market	TrdMarket	訂單標的所屬市場
qty	float	訂單數量
price	float	訂單價格
currency	Currency	交易貨幣

欄位	類型	說明
create_time	str	建立時間
updated_time	str	最後更新時間
dealt_qty	float	成交數量
dealt_avg_price	float	成交均價
last_err_msg	str	最後的錯誤描述
remark	str	下單時備註的標記
time_in_force	TimeInForce	有效期限
fill_outside_rth	bool	是否允許盤前盤後（用於港股盤前競價與美股盤前盤後）
session	Session	交易訂單時段（僅用於美股）
aux_price	float	觸發價格
trail_type	TrailType	跟蹤類型
trail_value	float	跟蹤金額/百分比
trail_spread	float	指定價差
jp_acc_type	SubAccType	日本帳戶類型

- **Example**

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, 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:  # 如果訂單列表不為空
        print(data['order_id'][0])  # 獲取持倉第一個訂單號
        print(data['order_id'].values.tolist())  # 轉為 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   HK.00700        HK           BUY           NORMAL  CANCELLED_ALL  664446861527
6644468615272262086
['6644468615272262086']
```

> **接口限制**
>
> - 同一帳戶ID(acc_id) 每 30 秒內最多請求 10 次查詢歷史訂單接口

> **提示**
>
> - 歷史訂單，按照時間的“倒序”進行排列，即：後提交的訂單在前，先提交的訂單在後

回應訂單推送回呼

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  回應訂單推送，非同步處理 OpenD 推送過來的訂單狀態資訊。
  在收到 OpenD 推送過來的訂單狀態資訊後會回呼到該函數，您需要在衍生類別中覆寫 on_recv_rsp。

- **參數**

參數	類型	說明
rsp_pb	Trd_UpdateOrder_pb2.Response	衍生類別中不需要直接處理該參數

- **傳回**

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，傳回訂單列表
	str	當 ret != RET_OK 時，傳回錯誤描述

  - 訂單列表格式如下：

欄位	類型	說明
trd_side	TrdSide	交易方向
order_type	OrderType	訂單類型
order_status	OrderStatus	訂單狀態
order_id	str	訂單號

欄位	類型	說明
code	str	股票編號
stock_name	str	股票名稱
qty	float	訂單數量
price	float	訂單價格
currency	Currency	交易貨幣
create_time	str	建立時間
updated_time	str	最後更新時間
dealt_qty	float	成交數量
dealt_avg_price	float	成交均價
last_err_msg	str	最後的錯誤描述
remark	str	下單時備註的標識
time_in_force	TimeInForce	有效期限
fill_outside_rth	bool	是否允許盤前盤後（僅用於美股）
session	Session	交易訂單時段（僅用於美股）
aux_price	float	觸發價格
trail_type	TrailType	跟蹤類型
trail_value	float	跟蹤金額/百分比
trail_spread	float	指定價差

- **Example**

```python
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', po
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       蘋果                0.0        0.0  100.0  7262526370867
```

查詢訂單費用

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

- **介紹**

  查詢指定訂單的收費明細（最低版本要求：8.2.4218）

- **參數**

參數	類型	說明
order_id_list	list	訂單號列表
trd_env	TrdEnv	交易環境
acc_id	int	交易業務賬戶 ID
acc_index	int	交易業務賬戶列表中的賬戶序號

- **傳回**

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，傳回訂單費用列表
	str	當 ret != RET_OK 時，傳回錯誤描述

  - 訂單列表格式如下：

欄位	類型	說明
order_id	str	訂單號
fee_amount	float	總費用

欄位	類型	說明
fee_details	list	收費明細

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', po
ret1, data1 = trd_ctx.history_order_list_query(status_filter_list=[OrderStatus.Fi
if ret1 == RET_OK:
    if data1.shape[0] > 0:  # 如果訂單列表不為空
        ret2, data2 = trd_ctx.order_fee_query(data1['order_id'].values.tolist())
        if ret2 == RET_OK:
            print(data2)
            print(data2['fee_details'][0])  # 打印第一筆訂單的收費明細
        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  [(佣金, 5.85), (平
1  v3_20240318_12345678_MTM5Nzc5MDYxNDY1NDM1MDI1M        2.25  [(佣金, 0.99), (平
[('佣金', 5.85), ('平台使用費', 2.7), ('期權監管費', 0.11), ('期權清算費', 0.18), (
```

接口限制

- 同一賬戶ID(acc_id) 每 30 秒內最多請求 10 次查詢訂單費用接口。
- 僅支援查詢 2018-01-01 之後的訂單。
- 模擬賬戶不支援查詢訂單費用。
- 加拿大證券商賬戶不支援查詢訂單費用。

訂閱交易推送

Python 不需要訂閱交易推送

查詢當日成交

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

- **介紹**

  查詢指定交易業務賬户的當日成交列表。
  該介面只支援實盤交易，不支援模擬交易。

- **參數**

參數	類型	說明
code	str	代碼過濾
deal_market	TrdMarket	成交標的所屬市場過濾
trd_env	TrdEnv	交易環境
acc_id	int	交易業務賬户 ID
acc_index	int	交易業務賬户列表中的賬户序號
refresh_cache	bool	是否更新快取

- **返回**

參數	類型	說明
ret	RET_CODE	介面調用結果
data	pd.DataFrame	當 ret == RET_OK 時，返回交易成交列表
	str	當 ret != RET_OK 時，返回錯誤描述

  - 交易成交列表格式如下：

欄位	類型	說明
trd_side	TrdSide	交易方向
deal_id	str	成交號
order_id	str	訂單號
code	str	股票代碼
stock_name	str	股票名稱
deal_market	TrdMarket	成交標的所屬市場
qty	float	成交數量
price	float	成交價格
create_time	str	創建時間
counter_broker_id	int	對手經紀號
counter_broker_name	str	對手經紀名稱
status	DealStatus	成交狀態
jp_acc_type	SubAccType	日本賬户類型

- Example

```
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US, host='127.0.0.1', p
ret, data = trd_ctx.deal_list_query()
if ret == RET_OK:
    print(data)
    if data.shape[0] > 0:  # 如果成交列表不為空
        print(data['order_id'][0])  # 獲取當日成交的第一個訂單號
        print(data['order_id'].values.tolist())  # 轉為 list
else:
    print('deal_list_query error: ', data)
trd_ctx.close()
```

- Output

```
    code stock_name        deal_market        deal_id             order_id    qty
0  US.AAPL      蘋果         US    5056208452274069375  4665291631090960915  100.0
4665291631090960915
['4665291631090960915']
```

> **介面限制**
>
> - 同一賬户ID(acc_id) 每 30 秒內最多請求 10 次查詢當日成交介面
> - 調用此介面，只有在更新快取時，才受到限頻限制

> **提示**
>
> - 當日成交，按照時間的“順序”進行排列，即：先成交的記錄在前，後成交的記錄在後

查詢歷史成交

`history_deal_list_query(code='', deal_market=TrdMarket.NONE, start='', end='', trd_env=TrdEnv.REAL, acc_id=0, acc_index=0)`

- **介紹**

  查詢指定交易業務帳戶的歷史成交列表。
  該介面只支援實盤交易，不支援模擬交易。

- **參數**

參數	類型	說明
code	str	代碼過濾
deal_market	TrdMarket	成交標的所屬市場過濾
start	str	開始時間
end	str	結束時間
trd_env	TrdEnv	交易環境
acc_id	int	交易業務帳戶 ID
acc_index	int	交易業務帳戶列表中的帳戶序號

  - start 和 end 的組合如下

Start 類型	End 類型	說明
str	str	start 和 end 分別為指定的日期
None	str	start 為 end 往前 90 天
str	None	end 為 start 往後 90 天

Start 類型	End 類型	說明
None	None	start 為往前 90 天，end 當前日期

- **回傳**

參數	類型	說明
ret	RET_CODE	介面調用結果
data	pd.DataFrame	當 ret == RET_OK 時，回傳交易成交列表
	str	當 ret != RET_OK 時，回傳錯誤描述

  - 交易成交列表格式如下：

欄位	類型	說明
trd_side	TrdSide	交易方向
deal_id	str	成交號
order_id	str	訂單號
code	str	股票編號
stock_name	str	股票名稱
deal_market	TrdMarket	成交標的所屬市場
qty	float	成交數量
price	float	成交價格
create_time	str	建立時間
counter_broker_id	int	對手經紀號
counter_broker_name	str	對手經紀名稱

欄位	類型	說明
status	DealStatus	成交狀態
jp_acc_type	SubAccType	日本帳戶類型

- 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:  # 如果成交列表不為空
        print(data['deal_id'][0])  # 獲取歷史成交的第一個成交號
        print(data['deal_id'].values.tolist())  # 轉為 list
else:
    print('history_deal_list_query error: ', data)
trd_ctx.close()
```

- Output

```
    code stock_name         deal_market        deal_id             order_id    q
0  US.AAPL       蘋果      US   5056208452274069375  4665291631090960915  100.0
5056208452274069375
['5056208452274069375']
```

> **介面限制**
>
> - 同一帳戶ID(acc_id) 每 30 秒內最多請求 10 次查詢歷史成交介面

> **提示**
>
> - 歷史成交，按照時間的“倒序”進行排列，即：後成交的記錄在前，先成交的記錄在後

回應成交推送回呼

`on_recv_rsp(self, rsp_pb)`

- **介紹**

  回應成交推送，非同步處理 OpenD 推送過來的成交狀態資訊。
  在收到 OpenD 推送過來的成交狀態資訊後會回呼到該函數，您需要在衍生類別中覆寫 on_recv_rsp。
  該接口只支援實盤交易，不支援模擬交易。

- **參數**

參數	類型	說明
rsp_pb	Trd_UpdateOrderFill_pb2.Response	衍生類別中不需要直接處理該參數

- **返回**

參數	類型	說明
ret	RET_CODE	介面執行結果
data	pd.DataFrame	當 ret == RET_OK 時，返回交易成交列表
	str	當 ret != RET_OK 時，返回錯誤描述

  - 交易成交列表格式如下：

欄位	類型	說明
trd_side	TrdSide	交易方向
deal_id	str	成交號
order_id	str	訂單號

欄位	類型	說明
code	str	股票編號
stock_name	str	股票名稱
qty	float	成交數量
price	float	成交價格
create_time	str	建立時間
counter_broker_id	int	對手經紀號
counter_broker_name	str	對手經紀名稱
status	DealStatus	成交狀態

- 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', po
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
```

```
2    0    REAL  US.AAPL        蘋果  2511067564122483295  8561504228375901919  100.0
```

交易定義

賬戶風控狀態

> CltRiskLevel

- `NONE`

  未知

- `SAFE`

  安全

- `WARNING`

  預警

- `DANGER`

  危險

- `ABSOLUTE_SAFE`

  絕對安全

- `OPT_DANGER`

  危險

> **提示**
>
> - 查詢期貨賬戶的風險狀態，建議使用 risk_status 欄位， 返回結果詳見 CltRiskStatus

貨幣類型

> Currency

- `NONE`

  未知貨幣

- `HKD`

  港元

- `USD`

  美元

- `CNH`

  離岸人民幣

- `JPY`

  日元

- `SGD`

  新元

- `AUD`

  澳元

- `CAD`

  加拿大元

- `MYR`

  馬幣

追蹤類型

TrailType

- `NONE`

  未知

- `RATIO`

比例

- `AMOUNT`

  金額

修改訂單操作

> ModifyOrderOp

- `NONE`

  未知操作

- `NORMAL`

  修改訂單

- `CANCEL`

  撤單

- `DISABLE`

  使失效

- `ENABLE`

  使生效

- `DELETE`

  刪除

成交狀態

> DealStatus

- `OK`

  正常

- `CANCELLED`

  成交被取消

- `CHANGED`

  成交被更改

訂單狀態

> OrderStatus

- `NONE`

  未知狀態

- `WAITING_SUBMIT`

  待提交

- `SUBMITTING`

  提交中

- `SUBMITTED`

  已提交，等待成交

- `FILLED_PART`

  部分成交

- `FILLED_ALL`

  全部已成交

- `CANCELLED_PART`

  部分成交，剩餘部分已撤單

- `CANCELLED_ALL`

  全部已撤單，無成交

- `FAILED`

  下單失敗，服務拒絕

- `DISABLED`

  已失效

- `DELETED`

  已刪除，無成交的訂單才能刪除

訂單類型

> **提示**
>
> - 實盤交易中，各個品類支援的訂單類型
> - 模擬交易中，僅支援限價單(NORMAL)和市價單(MARKET)。

> OrderType

- `NONE`

  未知類型

- `NORMAL`

  限價單

- `MARKET`

  市價單

- `ABSOLUTE_LIMIT`

  絕對限價訂單

- `AUCTION`

  競價市價單

- `AUCTION_LIMIT`

競價限價單

- `SPECIAL_LIMIT`

  特別限價單

- `SPECIAL_LIMIT_ALL`

  特別限價且要求全部成交訂單

- `STOP`

  止損市價單

- `STOP_LIMIT`

  止損限價單

- `MARKET_IF_TOUCHED`

  觸及市價單（止盈）

- `LIMIT_IF_TOUCHED`

  觸及限價單（止盈）

- `TRAILING_STOP`

  追蹤止損市價單

- `TRAILING_STOP_LIMIT`

  追蹤止損限價單

- `TWAP_LIMIT`

  時間加權限價算法單（港股和美股）

- `TWAP`

  時間加權市價算法單（僅美股）

- `VWAP_LIMIT`

  成交量加權限價算法單（港股和美股）

- `VWAP`

  成交量加權市價算法單（僅美股）

持倉方向

> PositionSide

- `NONE`

  未知方向

- `LONG`

  長倉

- `SHORT`

  短倉

賬戶類型

> TrdAccType

- `NONE`

  未知類型

- `CASH`

  現金賬戶

- `MARGIN`

  保證金賬戶

- `TFSA`

  加拿大免稅賬戶

- `RRSP`

加拿大註冊退休賬戶

- `SRRSP`

  加拿大配偶退休賬戶

- `DERIVATIVE`

  日本衍生品賬戶

交易環境

> TrdEnv

- `SIMULATE`

  模擬環境

- `REAL`

  真實環境

交易市場

> TrdMarket

- `NONE`

  未知市場

- `HK`

  香港市場

- `US`

  美國市場

- `CN`

  A 股市場

- `HKCC`

  香港 A 股通市場

- `FUTURES`

  期貨市場

- `FUTURES_SIMULATE_US`

  美國期貨模擬市場

- `FUTURES_SIMULATE_HK`

  香港期貨模擬市場

- `FUTURES_SIMULATE_SG`

  新加坡期貨模擬市場

- `FUTURES_SIMULATE_JP`

  日本期貨模擬市場

- `HKFUND`

  香港基金市場

- `USFUND`

  美國基金市場

- `SG`

  新加坡市場

- `JP`

  日本市場

- `AU`

  澳大利亞市場

- `MY`

馬來西亞市場

- `CA`

  加拿大市場

賬戶狀態

> TrdAccStatus

- `ACTIVE`

  生效賬戶

- `DISABLED`

  失效賬戶

賬戶結構

> TrdAccRole

- `NONE`

  未知

- `MASTER`

  主賬戶

- `NORMAL`

  普通賬戶

- `IPO`

  馬來西亞IPO賬戶

交易證券市場

交易方向

> TrdSide

- `NONE`

  未知方向

- `BUY`

  買入

- `SELL`

  賣出

- `SELL_SHORT`

  賣空

- `BUY_BACK`

  買回

> **提示**
>
> **下單** 接口的交易方向，建議僅使用 `買入` 和 `賣出` 兩個方向作為入參。`賣空` 和 `買回` 僅適用於日本券商，其他券商僅用於 **查詢今日訂單**，**查詢歷史訂單**，**響應訂單推送回調**，**查詢當日成交**，**查詢歷史成交**，**響應成交推送回調** 接口的返回欄位展示。

訂單有效期

> TimeInForce

- `DAY`

當日有效

- `GTC`

  撤單前有效

賬戶所屬券商

> SecurityFirm

- `NONE`

  未知

- `FUTUSECURITIES`

  富途證券（香港）

- `FUTUINC`

  moomoo證券(美國)

- `FUTUSG`
  moomoo證券(新加坡)

- `FUTUAU`
  moomoo證券(澳大利亞)

- `FUTUCA`
  moomoo證券(加拿大)

- `FUTUMY`
  moomoo證券(馬來西亞)

- `FUTUJP`
  moomoo證券(日本)

模擬交易賬戶類型

SimAccType

- `NONE`

  未知

- `STOCK`

  股票模擬賬戶

- `OPTION`

  期權模擬賬戶

- `FUTURES`

  期貨模擬賬戶

風險狀態

> CltRiskStatus

- `NONE`

  未知

- `LEVEL1`

  非常安全

- `LEVEL2`

  安全

- `LEVEL3`

  較安全

- `LEVEL4`

  較低風險

- `LEVEL5`

  中等風險

- `LEVEL6`

  偏高風險
- `LEVEL7`

  預警
- `LEVEL8`

  危險
- `LEVEL9`

  危險

日內交易限制情況

> DtStatus

- `NONE`

  未知
- `Unlimited`

  無限次
- `EM_Call`

  EM-Call
- `DT_Call`

  DT-Call

現金流方向

> CashFlowDirection

- `NONE`

未知

- `IN`

  現金流入

- `OUT`

  現金流出

日本子賬戶類型

> SubAccType

- `NONE`

  未知

- `JP_GENERAL`

  一般-Long

- `JP_TOKUTEI`

  特定-Long

- `JP_NISA_GENERAL`

  一般NISA

- `JP_NISA_TSUMITATE`

  累計NISA

- `JP_GENERAL_SHORT`

  一般-short

- `JP_TOKUTEI_SHORT`

  特定-short

- `JP_HONPO_GENERAL`

本國信用交易抵押品-一般

- `JP_GAIKOKU_GENERAL`

  外國信用交易抵押品-一般

- `JP_HONPO_TOKUTEI`

  本國信用交易抵押品-特定

- `JP_GAIKOKU_TOKUTEI`

  外國信用交易抵押品-特定

- `JP_DERIVATIVE_LONG`

  衍生品子賬戶-Long

- `JP_DERIVATIVE_SHORT`

  衍生品子賬戶-Short

- `JP_HONPO_DERIVATIVE_GENERAL`

  本國衍生品證據金子賬戶-一般

- `JP_GAIKOKU_DERIVATIVE_GENERAL`

  外國衍生品證據金子賬戶-一般

- `JP_HONPO_DERIVATIVE_TOKUTEI`

  本國衍生品證據金子賬戶-特定

- `JP_GAIKOKU_DERIVATIVE_TOKUTEI`

  外國衍生品證據金子賬戶-特定

資產類別

> AssetCategory

- `NONE`

未知

- `JP`

  本國

- `US`

  外國

交易品類

TrdCategory

```
enum TrdCategory
{
    TrdCategory_Unknown = 0; //未知品類
    TrdCategory_Security = 1; //證券
    TrdCategory_Future = 2; //期貨
}
```

賬戶現金資訊

AccCashInfo

```
message AccCashInfo
{
    optional int32 currency = 1;         // 貨幣類型，取值參考 Currency
    optional double cash = 2;            // 現金結餘
    optional double availableBalance = 3;    // 現金可提金額
    optional double netCashPower = 4;      // 現金購買力
}
```

分市場資產資訊

AccMarketInfo

```
message AccCashInfo
{
    optional int32 trdMarket = 1;         // 交易市場，參見TrdMarket的枚舉定義
    optional double assets = 2;           // 分市場資產資訊
}
```

交易協議公共參數頭

TrdHeader

```
message TrdHeader
{
  required int32 trdEnv = 1; //交易環境，參見 TrdEnv 的枚舉定義
  required uint64 accID = 2; //業務賬號，業務賬號與交易環境、市場權限需要匹配，否則會
  required int32 trdMarket = 3; //交易市場，參見 TrdMarket 的枚舉定義
  optional int32 jpAccType = 4; //JP子賬戶類型，取值見 TrdSubAccType
}
```

交易業務賬戶

TrdAcc

```
message TrdAcc
{
  required int32 trdEnv = 1; //交易環境，參見 TrdEnv 的枚舉定義
  required uint64 accID = 2; //業務賬號
  repeated int32 trdMarketAuthList = 3; //業務賬戶支援的交易市場權限，即此賬戶能交易
  optional int32 accType = 4;   //賬戶類型，取值見 TrdAccType
  optional string cardNum = 5;  //卡號
  optional int32 securityFirm = 6; //所屬券商，取值見SecurityFirm
  optional int32 simAccType = 7; //模擬交易賬號類型，取值見SimAccType
  optional string uniCardNum = 8;  //所屬綜合賬戶卡號
  optional int32 accStatus = 9; //賬號狀態，取值見TrdAccStatus
  optional int32 accRole = 10; //賬號分類，是不是主賬號，取值見TrdAccRole
  repeated int32 jpAccType = 11; //JP子賬戶類型，取值見 TrdSubAccType
}
```

```
13
14
```

賬戶資金

Funds

```
message Funds
{
  required double power = 1; //最大購買力（此欄位是按照 50% 的融資初始保證金率計算得到
  required double totalAssets = 2; //資產淨值
  required double cash = 3; //現金（僅單幣種賬戶使用此欄位，綜合賬戶請使用 cashInfoLi
  required double marketVal = 4; //證券市值，僅證券賬戶適用
  required double frozenCash = 5; //凍結資金
  required double debtCash = 6; //計息金額
  required double avlWithdrawalCash = 7; //現金可提（僅單幣種賬戶使用此欄位，綜合賬戶

  optional int32 currency = 8;            //幣種，本結構體資金相關的貨幣類型，取值參
  optional double availableFunds = 9;     //可用資金，期貨適用
  optional double unrealizedPL = 10;      //未實現盈虧，期貨適用
  optional double realizedPL = 11;        //已實現盈虧，期貨適用
  optional int32 riskLevel = 12;           //風控狀態，參見 CltRiskLevel，期貨適用
  optional double initialMargin = 13;      //初始保證金
  optional double maintenanceMargin = 14;  //維持保證金
  repeated AccCashInfo cashInfoList = 15;  //分幣種的現金、現金可提和現金購買力（僅給
  optional double maxPowerShort = 16; //賣空購買力（此欄位是按照 60% 的融券保證金率計
  optional double netCashPower = 17;  //現金購買力（僅單幣種賬戶使用此欄位，綜合賬戶計
  optional double longMv = 18;        //多頭市值
  optional double shortMv = 19;       //空頭市值
  optional double pendingAsset = 20;  //在途資產
  optional double maxWithdrawal = 21;          //融資可提，僅證券賬戶適用
  optional int32 riskStatus = 22;              //風險狀態，參見 CltRiskStatus，共分
  optional double marginCallMargin = 23;       //   Margin Call 保證金

  optional bool isPdt = 24;               //是否PDT賬戶，僅moomoo證券(美國)賬戶適用
  optional string pdtSeq = 25;            //剩餘日內交易次數，僅被標記為 PDT 的moom
  optional double beginningDTBP = 26;     //初始日內交易購買力，僅被標記為 PDT 的mo
  optional double remainingDTBP = 27;     //剩餘日內交易購買力，僅被標記為 PDT 的mo
  optional double dtCallAmount = 28;      //日內交易待繳金額，僅被標記為 PDT 的moom
  optional int32 dtStatus = 29;                //日內交易限制情況，取值見 DTStatus

  optional double securitiesAssets = 30; // 證券資產淨值
  optional double fundAssets = 31; // 基金資產淨值
```

```
  optional double bondAssets = 32; // 債券資產淨值

  repeated AccMarketInfo marketInfoList = 33; //分市場資產資訊
}
```

賬戶持倉

Position

```
message Position
{
    required uint64 positionID = 1;      //持倉 ID，一條持倉的唯一標識
    required int32 positionSide = 2;     //持倉方向，參見 PositionSide 的枚舉定義
    required string code = 3;            //代碼
    required string name = 4;            //名稱
    required double qty = 5;             //持有數量，2位精度，期權單位是"張"，下同
    required double canSellQty = 6;      //可用數量，是指持有的可平倉的數量。可用數量
    required double price = 7;           //市價，3位精度，期貨為2位精度
    optional double costPrice = 8;       //攤薄成本價（證券賬戶），平均開倉價（期貨賬
    required double val = 9;             //市值，3位精度，期貨此欄位值為0
    required double plVal = 10;          //盈虧金額，3位精度，期貨為2位精度
    optional double plRatio = 11;        //盈虧百分比(平均成本價模式)，無精度限制，如
    optional int32 secMarket = 12;       //證券所屬市場，參見 TrdSecMarket 的枚舉定義

     //以下是此持倉今日統計
    optional double td_plVal = 21;       //今日盈虧金額，3位精度，下同，期貨為2位精度
    optional double td_trdVal = 22;      //今日交易額，期貨不適用
    optional double td_buyVal = 23;      //今日買入總額，期貨不適用
    optional double td_buyQty = 24;      //今日買入總量，期貨不適用
    optional double td_sellVal = 25;     //今日賣出總額，期貨不適用
    optional double td_sellQty = 26;     //今日賣出總量，期貨不適用

    optional double unrealizedPL = 28;        //未實現盈虧（僅期貨賬戶適用）
    optional double realizedPL = 29;          //已實現盈虧（僅期貨賬戶適用）
    optional int32 currency = 30;          // 貨幣類型，取值參考 Currency
    optional int32 trdMarket = 31;  //交易市場，參見 TrdMarket 的枚舉定義

    optional double dilutedCostPrice = 32;       //攤薄成本價，僅支援證券賬戶使用
    optional double averageCostPrice = 33;       //平均成本價，模擬交易證券賬戶不適用
    optional double averagePlRatio = 34;         //盈虧百分比(平均成本價模式)，無精度
}
```

訂單

Order

```
message Order
{
    required int32 trdSide = 1; //交易方向, 參見 TrdSide 的枚舉定義
    required int32 orderType = 2; //訂單類型, 參見 OrderType 的枚舉定義
    required int32 orderStatus = 3; //訂單狀態, 參見 OrderStatus 的枚舉定義
    required uint64 orderID = 4; //訂單號
    required string orderIDEx = 5; //擴展訂單號(僅查問題時備用)
    required string code = 6; //代碼
    required string name = 7; //名稱
    required double qty = 8; //訂單數量，2位精度，期權單位是"張"
    optional double price = 9; //訂單價格，3位精度
    required string createTime = 10; //建立時間，嚴格按 YYYY-MM-DD HH:MM:SS 或 YYY
    required string updateTime = 11; //最後更新時間，嚴格按 YYYY-MM-DD HH:MM:SS 或
    optional double fillQty = 12; //成交數量，2位精度，期權單位是"張"
    optional double fillAvgPrice = 13; //成交均價，無精度限制
    optional string lastErrMsg = 14; //最後的錯誤描述，如果有錯誤，會有此描述最後一次
    optional int32 secMarket = 15; //證券所屬市場，參見 TrdSecMarket 的枚舉定義
    optional double createTimestamp = 16; //建立時間戳
    optional double updateTimestamp = 17; //最後更新時間戳
    optional string remark = 18; //用戶備註字串，最大長度64字節
    optional double auxPrice = 21; //觸發價格
    optional int32 trailType = 22; //追蹤類型, 參見Trd_Common.TrailType的枚舉定義
    optional double trailValue = 23; //追蹤金額/百分比
    optional double trailSpread = 24; //指定價差
    optional int32 currency = 25;         // 貨幣類型，取值參考 Currency
    optional int32 trdMarket = 26;  //交易市場, 參見TrdMarket的枚舉定義
    optional int32 session = 27; //美股訂單時段, 參見Common.Session的枚舉定義
    optional int32 jpAccType = 28; //JP子賬戶類型，取值見 TrdSubAccType
}
```

訂單費用條目

OrderFeeItem

```
message OrderFeeItem
{
    optional string title = 1; //費用名字
    optional double value = 2; //費用金額
}
```

訂單費用

OrderFee

```
message OrderFee
{
    required string orderIDEx = 1; //擴展訂單號
    optional double feeAmount = 2; //費用總額
    repeated OrderFeeItem feeList = 3; //費用明細
}
```

成交

OrderFill

```
message OrderFill
{
     required int32 trdSide = 1; //交易方向, 參見 TrdSide 的枚舉定義
    required uint64 fillID = 2; //成交號
    required string fillIDEx = 3; //擴展成交號(僅查問題時備用)
    optional uint64 orderID = 4; //訂單號
    optional string orderIDEx = 5; //擴展訂單號(僅查問題時備用)
    required string code = 6; //代碼
    required string name = 7; //名稱
    required double qty = 8; //成交數量，2位精度，期權單位是"張"
    required double price = 9; //成交價格，3位精度
    required string createTime = 10; //建立時間（成交時間），嚴格按 YYYY-MM-DD HH:M
    optional int32 counterBrokerID = 11; //對手經紀號，港股有效
    optional string counterBrokerName = 12; //對手經紀名稱，港股有效
    optional int32 secMarket = 13; //證券所屬市場，參見 TrdSecMarket 的枚舉定義
```

```
    optional double createTimestamp = 14; //建立時間戳
    optional double updateTimestamp = 15; //最後更新時間戳
    optional int32 status = 16; //成交狀態, 參見 OrderFillStatus 的枚舉定義
    optional int32 trdMarket = 17;  //交易市場, 參見TrdMarket的枚舉定義
    optional int32 jpAccType = 18; //JP子賬戶類型，取值見 TrdSubAccType
}
```

最大可交易數量

MaxTrdQtys

```
message MaxTrdQtys
{
    //因目前伺服器實現的問題，賣空需要先賣掉多頭持倉才能再賣空，是分開兩步賣的，買回來同
    required double maxCashBuy = 1;              //現金可買（期權的單位是“張”，期貨
    optional double maxCashAndMarginBuy = 2;     //最大可買（期權的單位是“張”，期貨賬
    required double maxPositionSell = 3;         //持倉可賣（期權的單位是“張”）
    optional double maxSellShort = 4;            //可賣空（期權的單位是“張”，期貨賬戶
    optional double maxBuyBack = 5;              //平倉需買入（當持有淨短倉時，必須先
    optional double longRequiredIM = 6;          //買 1 張合約所帶來的初始保證金變動
    optional double shortRequiredIM = 7;         //賣 1 張合約所帶來的初始保證金變動
}
```

現金流水數據

FlowSummaryInfo

```
message FlowSummaryInfo
{
    optional string clearingDate = 1; //清算日期
    optional string settlementDate = 2; //結算日期
    optional int32 currency = 3; //幣種
    optional string cashFlowType = 4; //現金流類型
    optional int32 cashFlowDirection = 5; //現金流方向 TrdCashFlowDirection
    optional double cashFlowAmount = 6; //金額
    optional string cashFlowRemark = 7; //備註
    optional uint64 cashFlowID = 8; //現金流 ID
}
```

```
10
11
```

過濾條件

TrdFilterConditions

```
message TrdFilterConditions
{
  repeated string codeList = 1; //代碼過濾，只返回包含這些代碼的數據，沒傳不過濾
  repeated uint64 idList = 2; //ID 主鍵過濾，只返回包含這些 ID 的數據，沒傳不過濾，訂
  optional string beginTime = 3; //開始時間，嚴格按 YYYY-MM-DD HH:MM:SS 或 YYYY-MM
  optional string endTime = 4; //結束時間，嚴格按 YYYY-MM-DD HH:MM:SS 或 YYYY-MM-D
  repeated string orderIDExList = 5; // 伺服器訂單ID列表，可以用來替代orderID列表，
  optional int32 filterMarket = 6; //指定交易市場, 參見TrdMarket的枚舉定義
}
```

基礎功能

設定 API 資訊

`set_client_info(client_id, client_ver)`

- **介紹**

  設定呼叫 API 的資訊（非必須）。

- **參數**

  - client_id: client 的標識
  - client_ver: client 的版本號

- **Example**

```
from moomoo import *
SysConfig.set_client_info("MymoomooAPI", 0)
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close()
```

設置協議格式

`set_proto_fmt(proto_fmt)`

- **介紹**

  設定通訊協議 Body 格式（非必須）　目前支援 Protobuf 或 Json 兩種格式，預設為 Protobuf。

- **參數**

  - proto_fmt: 協議格式，參見ProtoFMT

```
from moomoo import *
SysConfig.set_proto_fmt(ProtoFMT.Protobuf)
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close()
```

- Example

對所有連接設置協議加密

`enable_proto_encrypt(is_encrypt)`

- 介紹

  數據加密　對所有連線的請求及回傳內容進行加密。如需了解協議的加密流程，請詳見這裡。

- 參數

參數	類型	說明
is_encrypt	bool	是否啟用加密

- Example

```
from moomoo import *
SysConfig.enable_proto_encrypt(is_encrypt = True)
SysConfig.set_init_rsa_file("conn_key.txt")   # rsa 私鑰文件路徑
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close()
```

設置私鑰路徑

`set_init_rsa_file(file)`

- 介紹

  設置 RSA 私鑰文件路徑。如需瞭解協議加密流程，詳見 這裏。

- 參數

參數	類型	說明
file	str	私鑰文件路徑

- Example

```
from moomoo import *
SysConfig.enable_proto_encrypt(is_encrypt = True)
SysConfig.set_init_rsa_file("conn_key.txt")   # rsa 私鑰文件路徑
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.close()
```

設置線程模式

`set_all_thread_daemon(all_daemon)`

- 介紹

  執行緒 設定說明是否將所有內部建立的執行緒設定為背景執行緒：

  1. 若設定為背景執行緒：主執行緒結束後，整個程式也會隨之結束。 例如：使用實時回調 API 時，需自行確保主執行緒持續運作
  2. 若設定為非背景執行緒：主執行緒結束後，程式不會終止。 例如：建立行情或交易物件後，若未呼叫 close() 關閉連線...

- 參數

參數	類型	說明
all_daemon	bool	是否設置為 daemon 線程

- Example

```
from moomoo import *
SysConfig.set_all_thread_daemon(True)
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
# 不調用 quote_ctx.close()，進程也會退出
```

設置回調

`set_handler(handler)`

- **介紹**

  設定非同步回調處理物件

- **參數**

  - handler: 回調處理對象

類	說明
SysNotifyHandlerBase	OpenD 通知處理
StockQuoteHandlerBase	報價處理
OrderBookHandlerBase	買賣盤處理
CurKlineHandlerBase	實時 K 線處理
TickerHandlerBase	逐筆交易處理
RTDataHandlerBase	分時數據處理
BrokerHandlerBase	經紀排位處理
PriceReminderHandlerBase	到價提醒處理
TradeOrderHandlerBase	訂單處理
TradeDealHandlerBase	成交處理

```
import time
from moomoo import *
class OrderBookTest(OrderBookHandlerBase):
    def on_recv_rsp(self, rsp_str):
```

```
        ret_code, data = super(OrderBookTest,self).on_recv_rsp(rsp_str)
        if ret_code != RET_OK:
            print("OrderBookTest: error, msg: %s" % data)
            return RET_ERROR, data
        print("OrderBookTest ", data) # OrderBookTest 自己的處理邏輯
        return RET_OK, data
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = OrderBookTest()
quote_ctx.set_handler(handler)  # 設置實時擺盤迴調
quote_ctx.subscribe(['HK.00700'], [SubType.ORDER_BOOK])  # 訂閱買賣擺盤類型，OpenD
time.sleep(15)  #  設置腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()  # 關閉當條連接，OpenD 會在1分鐘後自動取消相應股票相應類型的訂閱
```

獲取連接 ID

`get_sync_conn_id()`

- **介紹**

  取得連線 ID，連線成功初始化後才會產生數值

- **返回**

  - conn_id: 連接 ID

- **Example**

```
from moomoo import *
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
quote_ctx.get_sync_conn_id()
quote_ctx.close() # 結束後記得關閉當條連接，防止連接條數用盡
```

事件通知回調

`SysNotifyHandlerBase`

- **介紹**

  系統通知 通知 OpenD 重要訊息（例如連線中斷等）。

- **協議 ID**

1003

- **返回**

參數	類型	說明
ret	RET_CODE	接口調用結果
data	tuple	當 ret == RET_OK 時，返回 **事件通知數據**
	str	當 ret != RET_OK，返回錯誤描述

  - **事件通知數據** 的格式如下：

參數	類型	說明
notify_type	SysNotifyType	通知類型
sub_type	ProgramStatusType	子類型。當 notify_type == SysNotifyType.PROGRAM_STATUS 時，sub_type 返回程式狀態類型
	GtwEventType	子類型。當 notify_type == SysNotifyType.GTW_EVENT 時，sub_type 返回 OpenD 事件通知類型
	0	當 notify_type != SysNotifyType.PROGRAM_STATUS 且 notify_type != SysNotifyType.GTW_EVENT 時，sub_type 返回 0
msg	dict	事件資訊。當 notify_type == SysNotifyType.CONN_STATUS 時，msg 返回 **連接狀態事件資訊** 字典
		事件資訊。當 notify_type == SysNotifyType.QOT_RIGHT 時，msg 返回 **行情權限事件資訊** 字典

- **連接狀態事件資訊** 字典結構如下（連接狀態類型為 bool，True 表示連接正常，False 表示連接斷開）：

```
{
    'qot_logined': bool1,
    'trd_logined': bool2,
}
```

- **行情權限事件資訊** 字典結構如下（點擊瞭解 行情權限）：

```
{
    '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,  // 已廢棄
    '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 事件通知
            print("GTW_EVENT, type: {} msg: {}".format(sub_type, msg))
        elif notify_type == SysNotifyType.PROGRAM_STATUS:  # 程序狀態變化通知
            print("PROGRAM_STATUS, type: {} msg: {}".format(sub_type, msg))
        elif notify_type == SysNotifyType.CONN_STATUS:  ## 連接狀態變化通知
            print("CONN_STATUS, qot: {}".format(msg['qot_logined']))
            print("CONN_STATUS, trd: {}".format(msg['trd_logined']))
        elif notify_type == SysNotifyType.QOT_RIGHT:  # 行情權限變化通知
            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, 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_
            print("QOT_RIGHT, us_future_cbot: {}".format(msg['us_future_qot_right
            print("QOT_RIGHT, us_future_nymex: {}".format(msg['us_future_qot_righ
            print("QOT_RIGHT, us_future_comex: {}".format(msg['us_future_qot_righ
            print("QOT_RIGHT, us_future_cboe: {}".format(msg['us_future_qot_right
        return RET_OK, data


quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111)
handler = SysNotifyTest()
quote_ctx.set_handler(handler)  # 設置回調
time.sleep(15)  # 設置腳本接收 OpenD 的推送持續時間為15秒
quote_ctx.close()  # 結束後記得關閉當條連接，防止連接條數用盡`
```

通用定義

API 調用結果

> RET_CODE

- `RET_OK`

  成功

- `RET_ERROR`

  失敗

協議格式

> ProtoFMT

- `Protobuf`

  Google Protobuf 格式

- `Json`

  Json 格式

封包加密演算法

程式狀態類型

> ProgramStatusType

- `NONE`

  未知

- `LOADED`

  已完成必要模塊加載

- `LOGING`

  登錄中

- `NEED_PIC_VERIFY_CODE`

  需要圖形驗證碼

- `NEED_PHONE_VERIFY_CODE`

  需要手機驗證碼

- `LOGIN_FAILED`

  登錄失敗

- `FORCE_UPDATE`

  客端戶版本過舊

- `NESSARY_DATA_PREPARING`

  正在獲取必要資訊

- `NESSARY_DATA_MISSING`

  缺少必要資訊

- `UN_AGREE_DISCLAIMER`

  未同意免責聲明

- `READY`

  正常可用狀態

- `FORCE_LOGOUT`

  OpenD 登入後被強制登出

網關事件通知類型

GtwEventType

- **`LocalCfgLoadFailed`**

  本地設定檔載入失敗

- **`APISvrRunFailed`**

  網關監聽服務運行失敗

- **`ForceUpdate`**

  強制更新網關

- **`LoginFailed`**

  登入富途伺服器失敗

- **`UnAgreeDisclaimer`**

  未同意免責聲明，無法執行

- **`LOGIN_FAILED`**

  登錄失敗

- **`NetCfgMissing`**

  缺少網絡連線設定

- **`KickedOut`**

  帳戶被強制下線

- **`LoginPwdChanged`**

  登入密碼已更改

- **`BanLogin`**

  牛牛後台不允許此帳戶登入

- **`NeedPicVerifyCode`**

  登錄需要輸入圖形驗證碼

- `NeedPhoneVerifyCode`

  登錄需要輸入手機驗證碼

- `AppDataNotExist`

  程式封包資料遺失

- `NessaryDataMissing`

  必要資料同步失敗

- `TradePwdChanged`

  交易密碼變更通知

- `EnableDeviceLock`

  需啟用裝置鎖

系統通知類型

> SysNotifyType

- `GTW_EVENT`

  網關事件

- `PROGRAM_STATUS`

  程式狀態變化

- `CONN_STATUS`

  與後台伺服器的連線狀態變化

- `QOT_RIGHT`

  行情權限變化

封包唯一識別碼

PacketID

```
message PacketID
{
	required uint64 connID = 1; //當前 TCP 連接的連接 ID，一條連接的唯一標識，Ini
	required uint32 serialNo = 2; //自增序列號
}
```

程式狀態

ProgramStatus

```
message ProgramStatus
{
	required ProgramStatusType type = 1; //當前狀態
	optional string strExtDesc = 2; // 額外描述
}
```

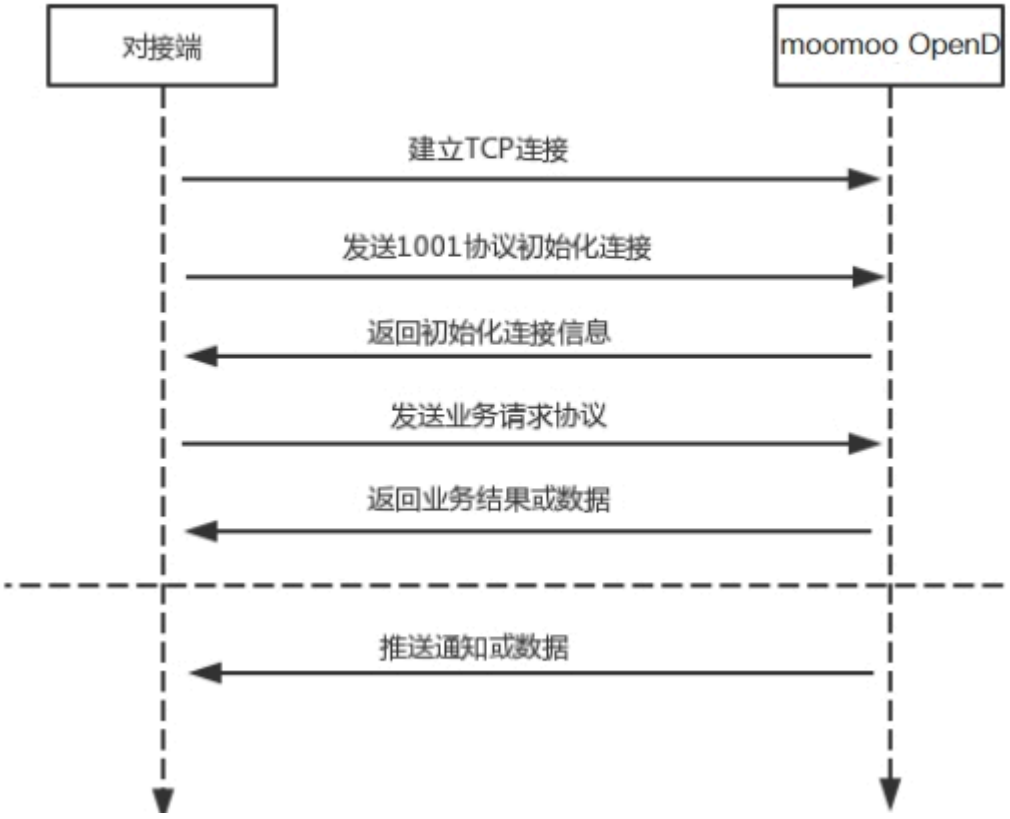
底層協議介紹

moomoo API 是 moomoo 為主流的程式語言（Python、Java、C#、C++、JavaScript）封裝的 API SDK，以方便您調用，降低策略開發難度。
這部分主要介紹策略腳本與 OpenD 服務之間通訊的底層協議，適用於非上述 5 種程式語言用戶，自行對接實現底層裸協議。

> **提示**
>
> - 如果您使用的程式語言在上述的 5 種主流程式語言之內，可以直接跳過這部分內容。

協議請求流程

- 建立連接
- 初始化連接
- 請求數據或接收推送數據
- 定時發送 KeepAlive 保持連接



協議設計

協議數據包括協議頭以及協議體，協議頭固定欄位，協議體根據具體協議決定。

協議頭

```
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];
};
```

欄位	說明
szHeaderFlag	包頭起始標誌，固定為“FT”
nProtoID	協議 ID
nProtoFmtType	協議格式類型，0 為 Protobuf 格式，1 為 Json 格式
nProtoVer	協議版本，用於迭代兼容，目前填 0
nSerialNo	包序列號，用於對應請求包和回傳封包 / 回傳資料，要求遞增
nBodyLen	包體長度
arrBodySHA1	包體原始數據(解密後)的 SHA1 雜湊值 (Hash)
arrReserved	保留 8 位元組 (Byte)擴展

> **提示**

- u8_t 表示 8 位無符號整數，u32_t 表示 32 位無符號整數
- OpenD 內部處理使用 Protobuf，因此協議格式建議使用 Protobuf，減少 Json 轉換開銷
- nProtoFmtType 欄位指定了包體的數據類型，回傳封包 / 回傳資料會回對應類型的數據；推送協議數據類型由 OpenD 配置檔案指定
- **arrBodySHA1 用於驗證請求數據在網絡傳輸前後的一致性，必須正確填入**
- **協議頭的二進制流使用的是小端位元組 (Byte)序，即一般不需要使用 ntohl 等相關函數轉換數據**

協議體

Protobuf 協議請求包體結構

```
message C2S
{
    required int64 req = 1;
}

message Request
{
    required C2S c2s = 1;
}
```

Protobuf 協議回應包體結構

```
message S2C
{
    required int64 data = 1;
}

message Response
{
    required int32 retType = 1 [default = -400]; //RetType，返回結果
    optional string retMsg = 2;
    optional int32 errCode = 3;
    optional S2C s2c = 4;
}
```

欄位	說明
c2s	請求參數結構
req	請求參數，實際根據協議定義
retType	請求結果
retMsg	若請求失敗，說明失敗原因
errCode	若請求失敗對應錯誤碼
s2c	回應數據結構，部分協議不返回數據則無該欄位
data	回應數據，實際根據協議定義

提示

- 包體格式類型請求包由協議頭 nProtoFmtType 指定，OpenD 主動推送格式在 InitConnect 設置。
- 原始協議檔案格式是以 Protobuf 格式定義，若需要 json 格式傳輸，建議使用 protobuf3 的介面 / API直接轉換成 json。
- 枚舉值欄位定義使用有符號整形，註解指明對應枚舉，枚舉一般定義於 Common.proto，Qot_Common.proto，Trd_Common.proto 檔案中。
- 協議中價格、百分比等數據用浮點類型來傳輸，直接使用會有精度問題，需要根據精度（如協議中未指明，預設小數點後三位）做四捨五入之後再使用。

心跳保活

```
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
{
```

```
10      required int64 time = 1; //客户端發包時的格林威治時間戳，單位秒
11  }
12
13  message S2C
14  {
15      required int64 time = 1; //伺服器回傳封包 / 回傳資料時的格林威治時間戳，單位秒
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,返回結果
26      optional string retMsg = 2;
27      optional int32 errCode = 3;
28
29      optional S2C s2c = 4;
30  }
```

- **介紹**

  心跳保活

- **協議 ID**

  1004

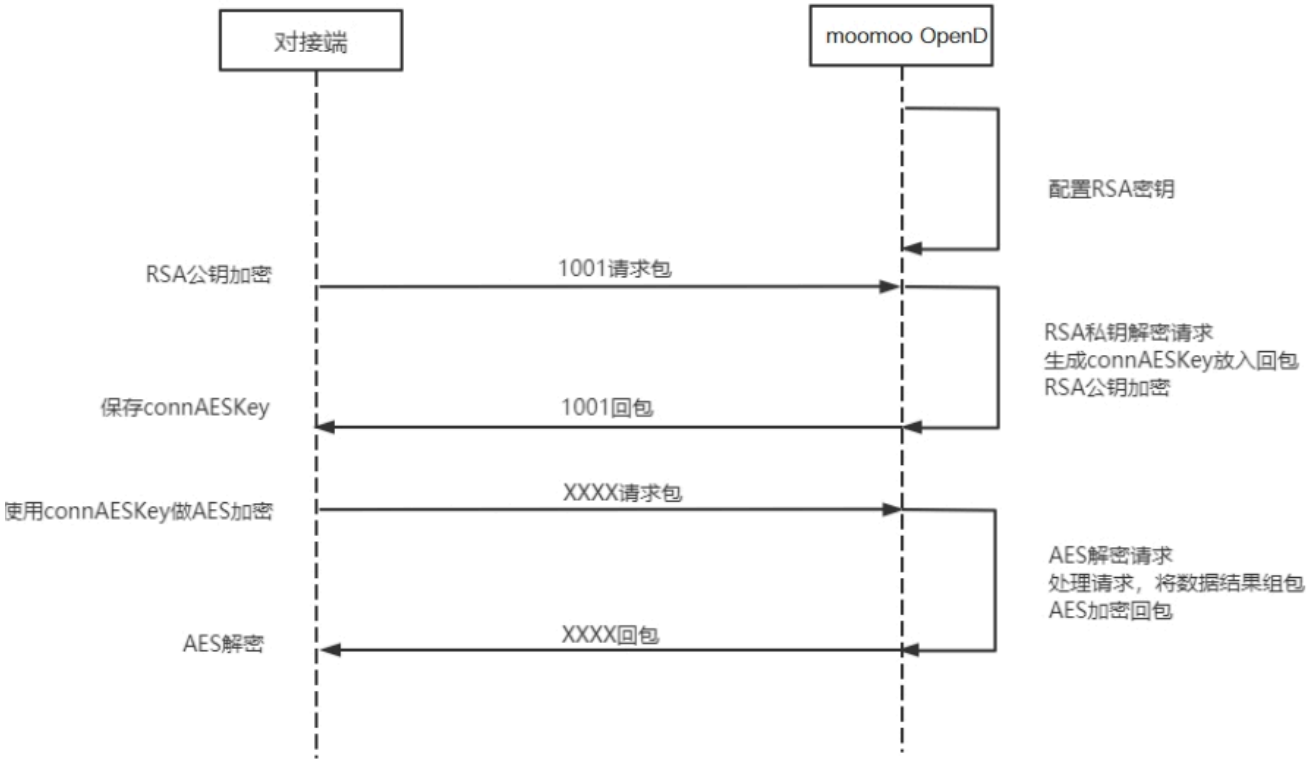
- **使用**

  根據初始化連結返回的心跳保活間隔時間，向 OpenD 發送保活協議

加密通訊流程

- 若 OpenD 配置了加密，**InitConnect** 初始化連接協議必須使用 **RSA** 公鑰加密，後續其他協議使用 InitConnect 返回的隨機密鑰進行 AES 加密通訊。
- OpenD 的加密流程借鑑了 SSL 協議，但考慮到一般是本地部署服務和應用，簡化了相關流程，OpenD 與接入 Client 共用了同一個 **RSA** 私鑰檔案，請妥善保存和分發私鑰檔案。

- 可到這個 網址 在線生成隨機 **RSA** 密鑰對，密鑰格式必須為 PCKS#1，密鑰長度 512，1024 都可以，不要設置密碼，將生成的私鑰複製保存到檔案中，然後將私鑰檔案路徑配置到 **OpenD** 配置 約定的 **rsa_private_key** 配置項中。
- **建議有實盤交易的用戶配置加密，避免賬户和交易資訊泄露。**



RSA 加解密

- **OpenD** 配置 約定 **rsa_private_key** 為私鑰檔案路徑
- OpenD 與接入客户端共用相同的私鑰檔案
- RSA 加解密僅用於 InitConnect 請求，用於安全獲取其它請求協議的對稱加密 Key
- OpenD 的 **RSA** 密鑰為 1024 位，填充方式 PKCS1，公鑰加密，私鑰解密，公鑰可通過私鑰生成
- Python API 參考實現：**RsaCrypt** 類的 encrypt / decrypt 介面 / API

發送數據加密

- RSA 加密規則:若密鑰位數是 key_size，單次加密串的最大長度為 (key_size)/8 - 11，目前位數 1024，一次加密長度可定為 100。
- 將明文數據分成一個或數個最長 100 位元組 (Byte)的小段進行加密，拼接分段加密數據即為最終的 Body 加密數據。

接收數據解密

- RSA 解密同樣遵循分段規則，對於 1024 位密鑰，每小段待解密數據長度為 128 位元組 (Byte)。
- 將密文數據分成一個或數個 128 位元組 (Byte)長的小段進行解密，拼接分段解密數據即為最終的 Body 解密數據。

AES 加解密

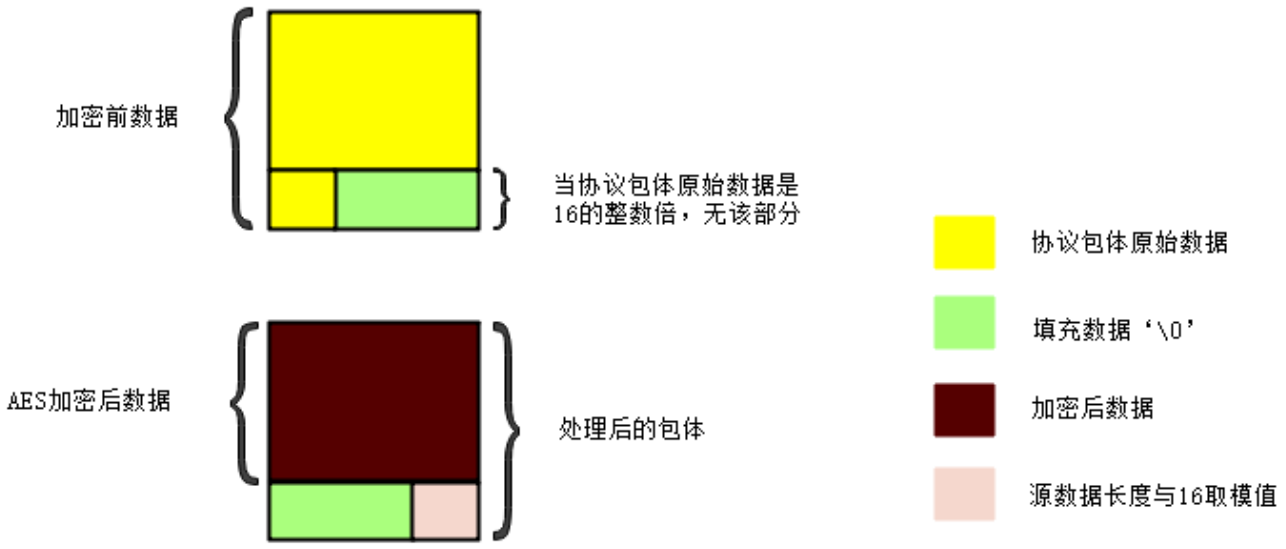
- 加密 key 由 InitConnect 協議返回
- 預設使用的是 AES 的 ecb 加密模式。
- Python API 參考實現: ConnMng 類的 encrypt_conn_data / decrypt_conn_data 介面 / API

發送數據加密

- AES 加密要求源數據長度必須是 16 的整數倍，故需補'0'對齊後再加密，記錄 mod_len 為源數據長度與 16 取模值。
- 因加密前有可能對源數據作修改，故需在加密後的數據尾再增加一個 16 位元組 (Byte)的填充數據塊，其最後一個位元組 (Byte)賦值 mod_len，其餘位元組 (Byte)賦值'0'，將加密數據和額外的填充數據塊拼接作為最終要發送協議的 body 數據。

接收數據解密

- 協議 body 數據，先將最後一個位元組 (Byte)取出，記為 mod_len，然後將 body 截掉尾部 16 位元組 (Byte)填充數據塊後再解密（與加密填充額外數據塊邏輯對應）。
- mod_len 為 0 時，上述解密後的數據即為協議返回的 body 數據，否則需截掉尾部(16 - mod_len)長度的用於填充對齊的數據。


加密前数据
当协议包体原始数据是
16的整数倍，无该部分
AES加密后数据
处理后的包体
协议包体原始数据
填充数据‘\0’
加密后数据
源数据长度与16取模值

OpenD 相關

Q1：OpenD 因未完成“問卷評估及協議確認”自動退出

A: 您需要進行相關問卷評估及協議確認，才可以使用 OpenD，請先 前往完成。

Q2：OpenD 因”程式內置數據不存在“退出

A: 一般因權限問題導致自帶數據拷貝失敗，可以嘗試將程式目錄下 *Appdata.dat* 解壓後的檔案複製到程式資料目錄下。

- windows 程式資料目錄: `%appdata%/com.moomoo.OpenD/F3CNN`
- 非 windows 程式資料目錄: `~/.com.moomoo.OpenD/F3CNN`

Q3：OpenD 服務啟動失敗

A: 請檢查：

1. 是否有其他程式佔用所配置的端口；
2. 是否已經有配置了相同端口的 OpenD 在運行。

Q4：如何驗證手機驗證碼？

A: 在 OpenD 界面上或遠程連接至 Telnet 端口，輸入命令 `input_phone_verify_code -code=123456`。

> **提示**
>
> - 123456 是收到的手機驗證碼
> - -code=123456 前有空格

Q5：是否支持其他編程語言？

A: OpenD 有對外提供基於 socket 的協議，目前我們提供並維護 Python，C++，Java，C# 和 JavaScript 接口，下載入口。

如果上述語言仍不能滿足您的需求，您可以自行整合 Protobuf 協議。

Q6：在同一設備多次驗證設備鎖

A: 設備標識隨機生成並存放於

windows: %appdata%/com.moomoo.OpenD/F3CNN/Device.dat 檔案中。 非windows: ~/.com.moomoo.OpenD/F3CNN/Device.dat

> **提示**
>
> 1. 如果檔案被刪除或損壞，OpenD 會重新生成新設備標識，然後驗證設備鎖。
> 2. 另外映像檔複製部署的用戶需要注意，如果多台機器的 Device.dat 內容相同，也會導致這些機器多次驗證設備鎖。刪除 Device.dat 檔案即可解決。

Q7：OpenD 是否有提供 Docker 鏡像？

A: 目前沒有提供。

Q8：一個賬號可以登入多個 OpenD 嗎？

A: 一個賬號可以在多台機器上登入 OpenD 或者其他客戶端，最多允許 10 個 OpenD 終端同時登入。同時有"行情互踢"的限制，只能有一個 OpenD 獲得最高權限行情。例如：兩個終端登入同一個賬號，只能有一個港股 LV2 行情，另一個是港股 BMP 行情。

Q9：如何控制 OpenD 和其他客户端（桌面版和流動版）的行情權限？

A: 應交易所的規定，多個終端同時在線會有"行情互踢"的限制，只能有一個終端獲得最高權限行情。OpenD 命令行版本的啟動參數中，內置了 **auto_hold_quote_right** 參數，用於靈活配置行情權限。當該參數選項開啟時，OpenD 在行情權限被搶後，會自動搶回。如果 10 秒內再次被搶，則其他終端獲得最高行情權限（OpenD 不會再搶）。

Q10：如何優先保證 OpenD 行情權限？

A:

1. 將 OpenD 啟動參數 auto_hold_quote_right 配置為 1；
2. 保證不要在流動版或桌面版富途牛牛上在 10 秒內連續兩次搶最高權限（登入算一次，點擊“重啟行情”算第二次）。

Q11：如何優先保證流動版（或桌面版）的行情權限？

A: OpenD 啟動參數 auto_hold_quote_right 設置為 0，流動版或桌面版富途牛牛在 OpenD 之後登入即可。

Q12：使用可視化 OpenD 記住密碼登入，長時間掛機後提示連接斷開，需要重新登入？

A: 使用可視化 OpenD，如果選擇記住密碼登入，用的是記錄在本地的令牌。由於令牌有時間限制，當令牌過期後，如果出現網絡波動或富途後台發佈，就可能導致與後台斷開連接後無法自動連接上的情況。因此，可視化 OpenD 如果希望長時間掛機，建議手動輸入密碼登入，由 OpenD 自動處理該情況。

Q13：遇到產品缺陷，如何請富途的研發工程師排查日誌？

A:

1. 與客服溝通問題表現，詳述：發生錯誤的時間、OpenD 版本號、 API 版本號、程式語言名稱、接口名或通訊協定編號、含詳細輸入參數和回傳的短代碼或截圖。
2. 客服確認是產品缺陷後，如需進一步日誌排查，研發工程師會主動聯繫。
3. 部分問題須提供 OpenD 日誌，方便定位確認問題。交易類問題需要 info 日誌級別，行情類問題需要 debug 日誌級別。日誌級別 log_level 可以在 ***OpenD.xml*** 中 配置 ，配置後需要重啟 OpenD 方能生效，待問題重現後，將該段日誌打包發給富途研發工程師。

> **提示**
>
> 日誌路徑如下：
> windows：`%appdata%/com.moomoo.OpenD/Log`
>
> 非 windows：`~/.com.moomoo.OpenD/Log`

Q14：程式連接不上 OpenD

A: 請先嘗試檢查：

1. 程式連接的端口與 OpenD 配置的端口是否一致。
2. 由於 OpenD 連接上限為 128，是否有無用連接未關閉。
3. 檢查監聽地址是否正確，如果腳本和 OpenD 不在同一機器，OpenD 監聽地址需要設置成 0.0.0.0 。

Q15：連接上一段時間後斷開

A: 如果是自己整合協議，檢查下是否有定時發送心跳維持連接。

Q16：Linux 下通過 multiprocessing 模塊以多進程方式運行 Python 腳本，連不上 OpenD？

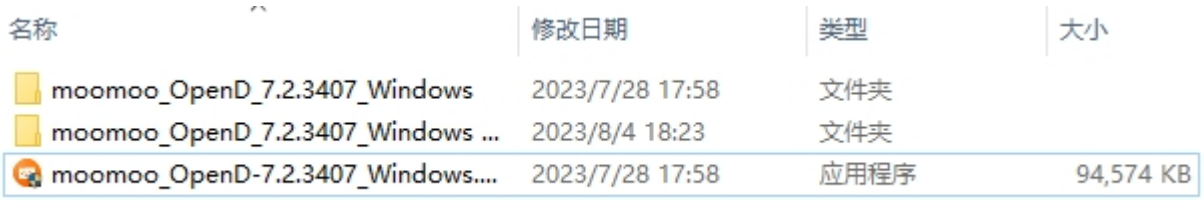
A: Linux/Mac 環境下以預設方式創建進程後，父進程中 py-moomoo-api 內部創建的線程將會在子進程中消失，導致程式內部狀態錯誤。
可以用 spawn 方式來啟動進程：

```py
import multiprocessing as mp
mp.set_start_method('spawn')
p = mp.Process(target=func)
```

Q17：如何在一台電腦同時登入兩個 OpenD?

A: 可視化 OpenD 不支持，命令行 OpenD 支持。

1. 解壓從官網下載的檔案，複製整個命令行 OpenD 資料夾（如 OpenD_5.2.1408_Windows）得到副本（此處以 Windows 為例，其他系統可採取相同操作）。

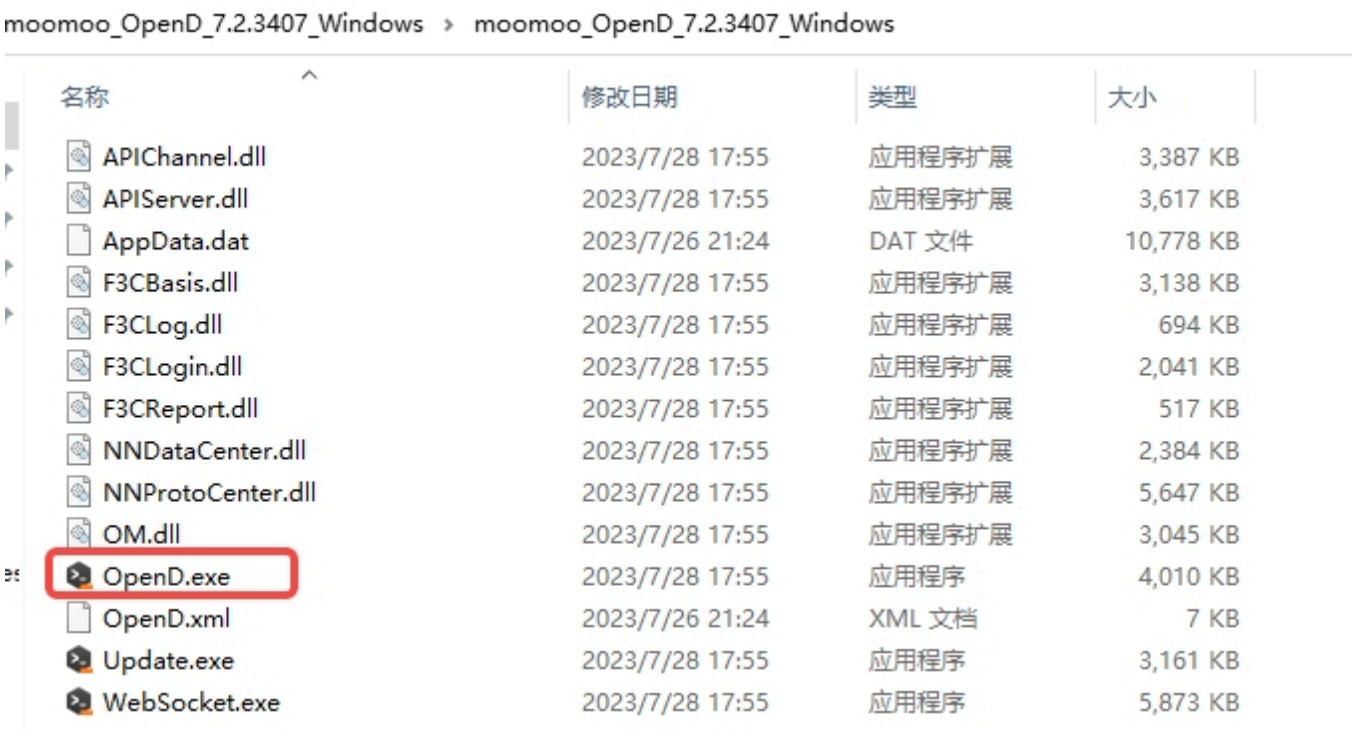


2. 分別打開兩個命令行 OpenD 資料夾配置好兩份 OpenD.xml 檔案。

第一份配置檔案參數：api_port = 11111，login_account = 登入賬號1，login_pwd = 登入密碼1

第二份配置檔案參數：api_port = 11112，login_account = 登入賬號2，login_pwd = 登入密碼2

3. 配置完成後，分別打開兩個 OpenD 程式運行。



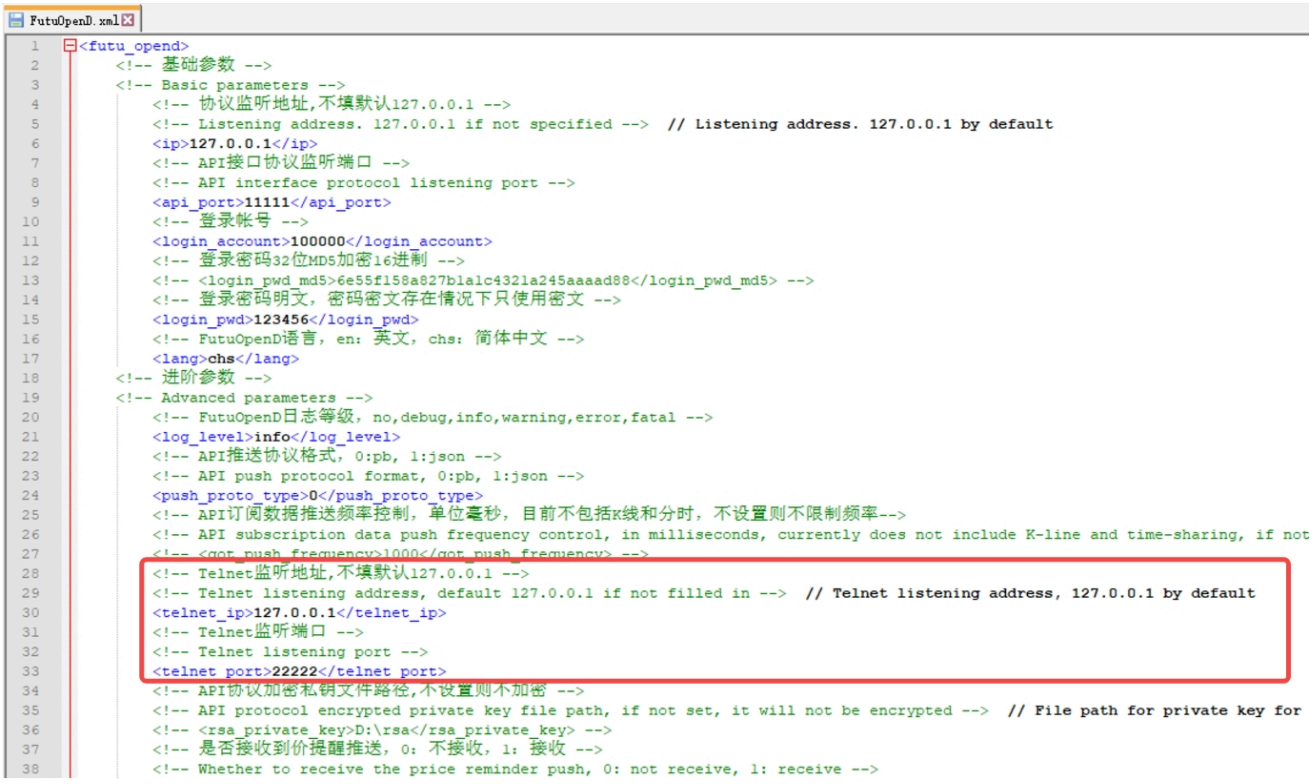
4. 調用接口時，注意接口的參數 **`port`**（OpenD 監聽端口）與 OpenD.xml 檔案中的參數 **`api_port`** 為對應關係
   例如：

```py
from moomoo import *

# 向賬號1登入的 OpenD 進行請求
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11111, is_encrypt=False)
quote_ctx.close() # 結束後記得關閉當條連接，防止連接條數用盡

# 向賬號2登入的 OpenD 進行請求
quote_ctx = OpenQuoteContext(host='127.0.0.1', port=11112, is_encrypt=False)
quote_ctx.close() # 結束後記得關閉當條連接，防止連接條數用盡
```

Q18：行情權限被其他客户端踢掉，如何通過腳本執行搶權限的運維命令？

A：

1. 在OpenD啟動參數中，配置好 Telnet 地址和 Telnet 端口。



2. 啟動 OpenD（會同時啟動 Telnet）。
3. 當發現行情權限被搶之後，您可以參考如下代碼示例，通過 Telnet，向 OpenD 發送 `request_highest_quote_right` 命令。

```py
from telnetlib import Telnet
with Telnet('127.0.0.1', 22222) as tn:  # Telnet 地址為：127.0.0.1，Telnet 端口為
    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 自動升級失敗

A： 通過 **update** 命令執行 OpenD 自動更新失敗，可能的原因：

- 檔案被其他進程佔用：可以嘗試關閉其他 OpenD 進程，或者重啟系統後，再次執行 **update** 如果以上仍無法解決，可以通過官網自行下載更新。

Q20：ubuntu22無法啟動可視化 OpenD？

A： 在有些Linux發行版（例如Ubuntu 22.04）運行可視化OpenD時，可能會提示：**dlopen(): error loading libfuse.so.2**。 這是因為這些系統沒有預設安裝libfuse。通常可以手動安裝來解決，例如對於Ubuntu22.04，可以在命令行運行：

```
sudo apt update
sudo apt install -y libfuse2
```

安裝成功後就可以正常運行可視化OpenD了。詳細資訊請參考：https://docs.appimage.org/user-guide/troubleshooting/fuse.html。

Q21：Linux上如何在後台運行命令行OpenD？

A：先切到 OpenD 所在目錄，配置好 OpenD.xml 之後，執行如下命令

```
nohup ./moomoo OpenD &
```

行情相關

Q1：訂閱失敗

A: 訂閱接口返回錯誤，有以下兩類常見情況：

- 訂閱額度不足：

  訂閱額度規則參見 訂閱額度 & 歷史 K 線額度

- 訂閱權限不足：

  支持訂閱的行情權限見下表

市場	品種	支持訂閱的行情權限
香港市場	股票	LV1, LV2, SF
	期權	LV1, LV2
	期貨	LV1, LV2
美國市場	股票	LV1, LV2
	期權	LV1
	期貨	LV1, LV2
A 股市場	股票	LV1

  獲取行情權限的方式參見 行情權限

  注意：若賬號擁有上述權限，但仍訂閱失敗，可能存在被其他終端 踢掉行情權限 的情況。

Q2：取消訂閱失敗

A: 訂閱至少一分鐘後才能取消訂閱。

Q3：取消訂閱成功但沒釋放額度

A: 所有連接都對該行情取消訂閱，才會釋放額度。

舉例：A 連接和 B 連接都在訂閱 HK.00700 的掛盤，當 A 連接取消訂閱後，由於 B 連接仍在調用騰訊的掛盤數據，因此 OpenD 的額度不會釋放，直至所有連接都取消訂閱 HK.00700 的掛盤。

Q4：訂閱不足一分鐘關閉程式連接，會釋放額度嗎？

A: 不會。連接關閉後，訂閱時長不足一分鐘的標的類型，會在達到一分鐘後才自動取消訂閱，並釋放相應的訂閱額度。

Q5：請求頻率限制的具體限制邏輯是怎樣？

A: 30 秒內最多 n 次，是指第 1 次和第 n+1 次請求間隔需要大於 30 秒。

Q6：自選股添加不上是什麼原因？

A: 請先檢查是否有超出上限，或者刪除一部分自選。

Q7：為什麼 OpenAPI 端的美股報價和牛牛顯示端的全美綜合報價有不同？

A: 由於美股交易分散在很多家交易所，富途有提供兩種美股基本報價行情，一種是 Nasdaq Basic（Nasdaq 交易所的報價），另一種是全美綜合報價（全美13家交易所的報價）。而 OpenAPI 的美股正股行情目前僅支持通過行情卡購買的方式獲取 Nasdaq Basic，不支持全美綜合報價。因此，如果您同時購買了顯示端的全美綜合報價行情卡，和僅用於 OpenAPI 的 Nasdaq Basic 行情卡，確實有可能出現牛牛顯示端和 OpenAPI 端的報價差異。
因此，如果您發現美股當天開市價與客戶端顯示不一致，這是因為OpenAPI實時上游行情僅會獲取 Nasdaq Basic 數據。

Q8：OpenAPI 行情卡在哪裏購買？

A:

- 港股市場
  - 港股 LV2 高級行情（僅港澳台及海外 IP）
  - 港股 LV2 + 期權期貨 LV2 行情（僅港澳台及海外 IP）
- 美股市場
  - Nasdaq Basic
  - Nasdaq Basic+TotalView (Non-Pro)
  - Nasdaq Basic+TotalView (Pro)
  - 期權 OPRA 實時行情

Q9：為什麼有時候，獲取實時數據的 get 接口響應比較慢？

A: 因為獲取實時數據的 get 接口需要先訂閱，並依賴後台給 OpenD 的推送。如果用戶剛訂閱就立刻用 get 接口請求，OpenD 有可能尚未收到後台推送。為了防止這種情況的發生，get 接口內置了等待邏輯，3 秒內收到推送會立刻返回給程式，超過 3 秒仍未收到後台推送，才會給程式返回空數據。
涉及的 get 接口包括：get_rt_ticker、get_rt_data、get_cur_kline、get_order_book、get_broker_queue、get_stock_quote。因此，當發現獲取實時數據的 get 接口響應比較慢時，可以先檢查一下是否是無成交數據的原因。

Q10：購買 OpenAPI 美股 Nasdaq Basic 行情卡後，可以獲取哪些數據？

A: Nasdaq Basic 行情卡購買啟用後，可以獲取的品類涵蓋 Nasdaq、NYSE、NYSE MKT 交易所上市證券（包括美股正股和 ETF，不包括美股期貨和美股期權）。
支持的數據接口包括：快照，歷史 K 線，實時逐筆訂閱，實時一檔掛盤訂閱，實時 K 線訂閱，實時報價訂閱，實時分時訂閱，到價提醒。

Q11：各個行情品類的掛盤支持多少檔？

A:

行情品類	LV1	LV2	SF
港股（含正股、窩輪、牛熊、界內證）	/	10	全盤+千筆明細
港股期權期貨	1	10	/
美股（含 ETF）	1	60檔	/
美股期權	1	/	/
美股期貨	/	40檔	/
A 股	5	/	/

Q12：為什麼我購買啟用了行情卡之後，OpenD 仍然沒有行情權限？

A:

1. 由於 OpenAPI 的行情權限跟 APP 的行情權限不完全一樣，部分行情卡僅適用於 APP 端（例如：OpenAPI美股行情卡需單獨購買）。請先確認您所購買的行情卡是否是 OpenD 適用的。我們已將 OpenAPI 適用的 **所有** 行情卡列在《權限與限制》一節，請點擊 這裏 查看。
2. 行情卡購買啟用成功後，是立即生效的。請 **重新啟動 OpenD** 後，再次查看權限狀態。

Q13：如何通過訂閱接口獲取實時行情？

第一步：訂閱

將標的的代碼和數據類型傳入 訂閱接口，完成訂閱。

訂閱接口支持了實時報價、實時掛盤、實時逐筆、實時分時、實時 K 線、實時經紀隊列數據的獲取。訂閱成功後，OpenD 會持續收到富途服務器的實時數據推送。

注意：訂閱額度會根據您的總資產、交易筆數和交易量，來進行分配，具體規則參見 訂閱額度 & 歷史 K 線額度。所以，如果您的訂閱額度不足，可以先檢查一下是否有無用的訂閱在佔用額度，及時 取消訂閱 即可釋放已佔用的訂閱額度。

第二步：取得數據

如何將訂閱推送的數據從 OpenD 取回程式呢？我們提供瞭如下兩種方式：

方式 1：實時數據回調
設置相應的回調函數，來異步處理 OpenD 收到的數據推送。

設置好回調函數後，OpenD 會將收到的實時數據，立即推給程式的回調函數進行處理。

如果所訂閱的標的比較活躍，此時的推送數據可能數據量較大且頻率較高。如果您希望適當降低 OpenD 給程式的推送頻率，建議在 OpenD 啟動參數 中設定 API 推送頻率（ `qot_push_frequency` ）。

方式 1 涉及的接口包括：實時報價回調、實時掛盤迴調、實時 K 線回調、實時分時回調、實時逐筆回調、實時經紀隊列回調。

方式 2：獲取實時數據
通過獲取實時數據接口，可以將 OpenD 收到的最新的數據，取回程式。這種方式更加靈活，程式不需要處理海量的推送。只要 OpenD 在持續接收富途服務器的推送，程式可以隨用隨取，不用不取。

由於是從 OpenD 接收的推送數據中取，所以這類接口沒有頻率限制。

方式 2 涉及的接口包括：獲取實時報價、獲取實時掛盤、獲取實時 K 線、獲取實時分時、獲取實時逐筆、獲取實時經紀隊列。

Q14：各個市場狀態對應什麼時間段？

A:

市場	品類	市場狀態	時間段（當地時間）
香港市場	證券類產品（含股票、	* NONE：無交易	CST 08:55 - 09:00

	ETFs、窩輪、牛熊、界內證）	* AUCTION：盤前競價	CST 09:00 - 09:20
		* WAITING_OPEN：等待開市	CST 09:20 - 09:30
		* MORNING：早盤	CST 09:30 - 12:00
		* REST: 午間休市	CST 12:00 - 13:00
		* AFTERNOON：午盤	CST 13:00 - 16:00
		* HK_CAS：港股盤後競價（港股市場增加 CAS 機制對應的市場狀態）	CST 16:00 - 16:08
		* CLOSED：收市	CST 16:08 - 08:55（T+1）
	期權、期貨（僅日市）	* NONE：期權待開市	CST 08:55 - 09:30
		* MORNING：早盤	CST 09:30 - 12:00
		* REST: 午間休市	CST 12:00 - 13:00
		* AFTERNOON：午盤	CST 13:00 - 16:00
		* CLOSED：收市	CST 16:00 - 08:55（T+1）
	期貨（日夜市）	* FUTURE_DAY_WAIT_FOR_OPEN：期貨待開市	不同品種交易時間不同

		* NIGHT_OPEN: 夜市交易時段	
		* NIGHT_END：夜市收市	
		* FUTURE_DAY_WAIT_FOR_OPEN：期貨待開市	
		* FUTURE_DAY_OPEN：日市交易時段	
		* FUTURE_DAY_CLOSE：日市收市	
美國市場	證券類產品（含股票、ETFs）	* PRE_MARKET_BEGIN：美股盤前交易時段	EST 04:00 - 09:30
		* AFTERNOON：美股持續交易時段	EST 09:30 - 16:00
		* AFTER_HOURS_BEGIN：美股盤後交易時段	EST 16:00 - 20:00
		* AFTER_HOURS_END：美股盤後收市	EST 20:00 - 04:00（T+1）
		* OVERNIGHT：美股夜盤交易時段	EST 20:00 - 04:00（T+1）
	期權	* NONE：期權待開市	不同品種交易時間不同
		* REST：美指期權午間休市	
		* AFTERNOON：美股持續交易時段	
		* TRADE_AT_LAST：美指期權盤尾交易時段	
		* NIGHT：美指期權夜市交易時段	
		* CLOSED：收市	
	期貨	* FUTURE_SWITCH_DATE：美期待開市	不同品種交易時間不同

		* FUTURE_OPEN：美期交易時段	
		* FUTURE_BREAK：美期中盤休息	
		* FUTRUE_BREAK_OVER：美期休息後交易時段	
		* FUTURE_CLOSE：美期收市	
A股市場	證券類產品（含股票、ETFs）	* NONE：無交易	CST 08:55 - 09:15
		* Auction：盤前競價	CST 09:15 - 09:25
		* WAITING_OPEN：等待開市	CST 09:25 - 09:30
		* MORNING：早盤	CST 09:30 - 11:30
		* REST：午間休市	CST 11:30 - 13:00
		* AFTERNOON：午盤	CST 13:00 - 15:00
		* CLOSED：收市	CST 15:00 - 08:55（T+1）
新加坡市場	期貨	* FUTURE_DAY_WAIT_FOR_OPEN：期貨待開市	不同品種交易時間不同
		* NIGHT_OPEN：夜市交易時段	
		* NIGHT_END：夜市收市	
		* FUTURE_DAY_OPEN：日市交易時段	
		* FUTURE_DAY_CLOSE：日市收市	

日本市場	期貨	* FUTURE_DAY_WAIT_FOR_OPEN：期貨待開市	JST 16:25（T-1）- 16:30（T-1）
		* NIGHT_OPEN：夜市交易時段	JST 16:30（T-1） - 05:30
		* NIGHT_END：夜市收市	JST 05:30 - 08:45
		* FUTURE_DAY_OPEN：日市交易時段	JST 08:45 - 15:15
		* FUTURE_DAY_CLOSE：日市收市	JST 15:15 - 16:25

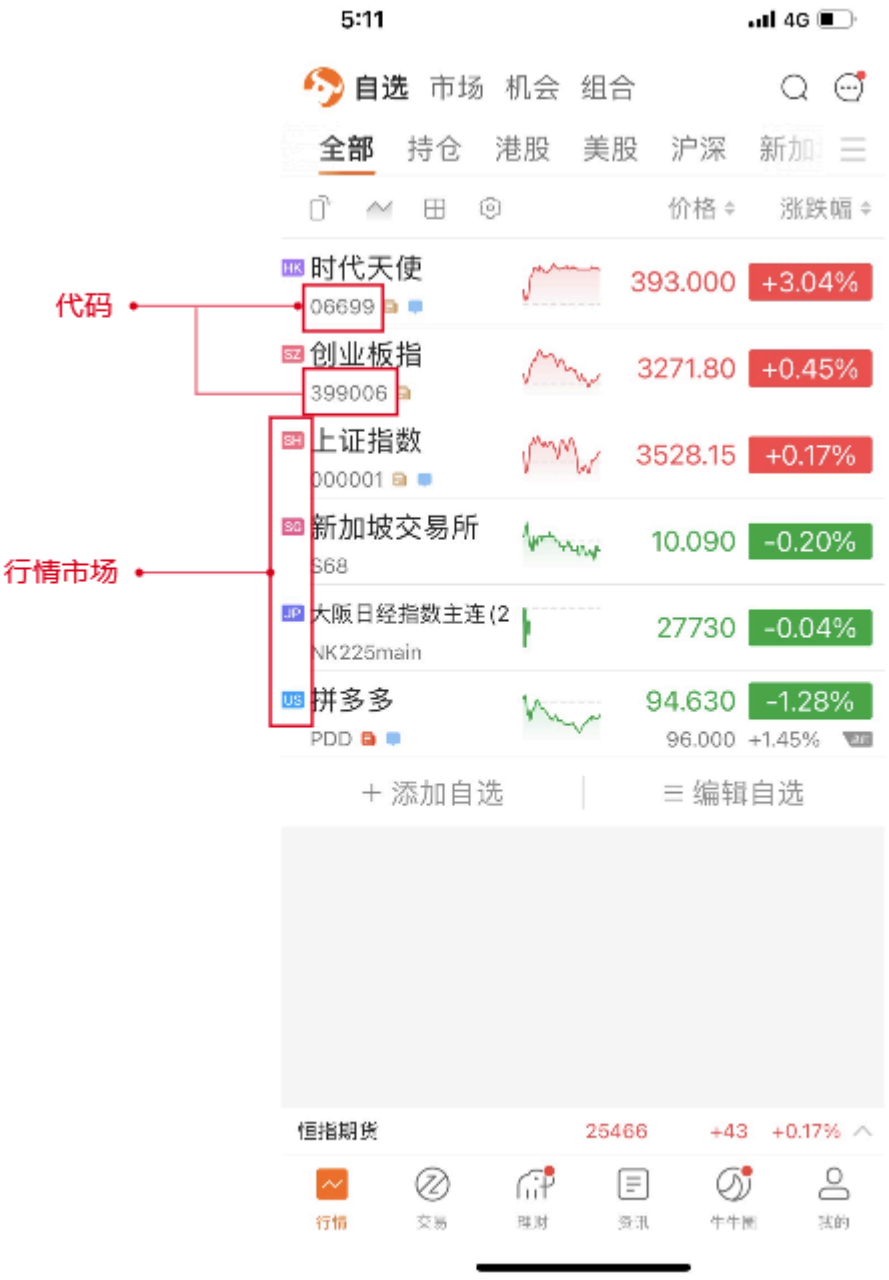
* CST, EST, JST 分別表示中國時間，美東時間，日本時間

Q15：接口參數股票代碼的格式

A：

- 使用不同程式語言的用戶，需要的股票代碼的格式不同：
  - **Python 用戶**
    股票代碼 code 格式：`行情市場.代碼`。
    例如：騰訊控股，參數 code 傳入'HK.00700'。
  - **非 Python 用戶**
    股票結構參見 Security。
    例如：騰訊控股，參數 market 傳入 QotMarket_HK_Security，參數 code 傳入'00700'。
- 查詢方式：
  通過 APP 查看代碼和行情市場：行情 > 自選 > 全部。

行情市場定義，請參考 這裏。



Q16：復權因子相關

A：

概述

所謂 復權 就是對股價和成交量進行權息修復，按照股票的實際漲跌繪製股價走勢圖，並把成交量調整為相同的股本口徑。

公司行動（如：拆股、合股、送股、轉增股、配股、增發股、分紅）均可能對股價產生影響，而復權計算可對量價進行調整，剔除公司行動的影響，保持股價走勢的連續性。

名詞解釋

- 公司行動：上市公司進行一些股權、股票等影響公司股價和股東持倉變化的行為。
- 前復權：保持現有的股價不變，以當前的股價為基準，對以前的股價進行復權計算。
- 後復權：保持先前的股價不變，以過去的股價為基準，對以後的股價進行復權計算。
- 復權因子：即權息修複比例，用於計算復權後的價格及持倉數量。
- 除權除息日：即股權登記日下一個交易日。在股票的除權除息日，證券交易所都要計算出股票的除權除息價，以作為股民在除權除息日開市的參考。其意義是股票股利分配給股東的日期。

復權方法

主流的復權計算方法分為兩種：事件法和連乘法；而 OpenAPI 針對不同市場使用不同的計算方法。

- 事件復權法：通過還原除權除息的各類事件進行復權；存在兩個復權因子（復權因子 A 和 復權因子 B），復權因子 B 主要調整現金分紅對股價的影響，而復權因子 A 調整其他公司行動對股價的影響。
- 連乘復權法：通過復權因子連乘的方式進行復權，只保留 復權因子 A（或將 復權因子 B 置為 0），復權因子 A 為 除權除息日前收市價/該日經權息調整後的前收市價。

> **提示**
>
> - OpenAPI 對美股前復權使用連乘法，即將 復權因子 B 置為0。
> - OpenAPI 對除美股以外的標的（A股、港股、新加坡股票等）及美股後復權使用事件法。

計算公式

單次復權

- 前復權：
  前復權價格 = 不復權價格 × 前復權因子 A + 前復權因子 B

- 後復權：
  後復權價格 = 不復權價格 × 後復權因子 A + 後復權因子 B

多次復權

- 前復權：按照時間順序，篩選出大於計算日期的復權因子，優先使用時間較早的復權因子進行復權計算。以兩次復權為例：

$$Price_n^{adjusted} = (Price_n * FactorA_{n+1}^{adjusted} + FactorB_{n+1}^{adjusted}) * FactorA_{n+2}^{adjusted} + FactorB_{n+2}^{adjusted}$$

$Price_n^{adjusted}$：被计算当天的前复权价格

$Price_n$：当天的不复权价格

$FactorA_{n+1}^{adjusted}$：后一天的前复权因子 A

$FactorB_{n+1}^{adjusted}$：后一天的前复权因子 B

$FactorA_{n+2}^{adjusted}$：后两天的前复权因子 A

$FactorB_{n+2}^{adjusted}$：后两天的前复权因子 B

- 後復權：按照時間倒序，篩選出小於等於計算日期的復權因子，優先使用時間較晚的復權因子進行復權計算。以兩次復權為例：

$$Price_n^{cumulative} = (Price_n * FactorA_n^{cumulative} + FactorB_n^{backward}) * FactorA_{n-1}^{cumulative} + FactorB_{n-1}^{cumulative}$$

$Price_n^{cumulative}$：被计算当天的后复权价格

$Price_n$：当天的不复权价格

$FactorA_n^{cumulative}$：当天的后复权因子 A

$FactorB_n^{cumulative}$：当天的后复权因子 B

$FactorA_{n-1}^{cumulative}$：前一天的后复权因子 A

$FactorB_{n-1}^{cumulative}$：前一天的后复权因子 B

示例

單次前復權示例

以牧原股份為例：

- 篩選復權因子如下：

除權除息日	股票代碼	方案說明	前復權因子 A	前復權因子 B
2021/06/03	SZ.002714	10轉4.0股派14.61元（含税）	0.71429	-1.04357

- 不復權數據如下：

日期	股票代碼	不復權收市價
2021/06/02	SZ.002714	93.11
2021/06/03	SZ.002714	66.25

- 前復權數據如下：

日期	股票代碼	前復權收市價
2021/06/02	SZ.002714	65.4639719
2021/06/03	SZ.002714	66.25

- 前復權數據計算方法：
  牧原股份在 2021/06/03 進行拆股及現金分紅行動（10轉4.0股派14.61元），根據前復權計算公式對 2021/06/02 的收市價進行調整計算，則：前復權價格（65.4639719） = 不復權價格（93.11） × 前復權因子 A（0.71429） + 前復權因子 B（-1.04357）

多次後復權示例

接上一個例子，計算牧原股份在 2021/06/02 的後復權價格：

- 篩選復權因子如下：

除權除息日	股票代碼	方案說明	後復權因子 A	後復權因子 B
2014/07/04	SZ.002714	10派2.34元（含税）	1	0.234
2015-06-10	SZ.002714	10轉10.0股派0.61元（含税）	2	0.061
2016-07-08	SZ.002714	10轉10.0股派3.53元（含税）	2	0.353
2017-07-11	SZ.002714	10轉8.0股派6.9元（含税）	1.8	0.69
2018-07-03	SZ.002714	10派6.91元（含税）	1	0.691
2019-07-04	SZ.002714	10派0.5元（含税）	1	0.05
2020-06-04	SZ.002714	10轉7.0股派5.5元（含税）	1.7	0.55

- 不復權數據如下：

日期	股票代碼	不復權收市價
2021/06/02	SZ.002714	93.11

- 後復權數據如下：

日期	股票代碼	後復權收市價
2021/06/02	SZ.002714	1152.7226

- 後復權數據計算方法：
  為了計算牧原股份在 2021/06/02 的後復權價格，需要將早於 2021/06/02 的復權事件進行一一復權，得到最後的後復權價格，具體計算如下：

不复权价格

日期	股票代码	不复权收盘价
2021/06/02	SZ.002714	93.11

后复权价格

日期	股票代码	不复权收盘价
2021/06/02	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

复权因子

除权除息日	股票代码	方案说明	后复权因子 A	后复权因子 B
2014/07/04	SZ.002714	10派2.34元（含税）	1	0.234
2015-06-10	SZ.002714	10转10.0股派0.61元（含税）	2	0.061
2016-07-08	SZ.002714	10转10.0股派3.53元（含税）	2	0.353
2017-07-11	SZ.002714	10转8.0股派6.9元（含税）	1.8	0.69
2018-07-03	SZ.002714	10派6.91元（含税）	1	0.691
2019-07-04	SZ.002714	10派0.5元（含税）	1	0.05
2020-06-04	SZ.002714	10转7.0股派5.5元（含税）	1.7	0.55

交易相關

Q1：模擬交易相關

A:

概述

模擬交易是在真實的市場環境中，用虛擬資金做交易，不會對您的真實帳戶的資產造成影響。

交易時間

模擬交易僅支持在常規交易時段交易，不支持在非交易時段、美股盤前盤後時段、A股港股盤前盤後競價時段交易。詳情可點擊 模擬交易規則。

支持品類

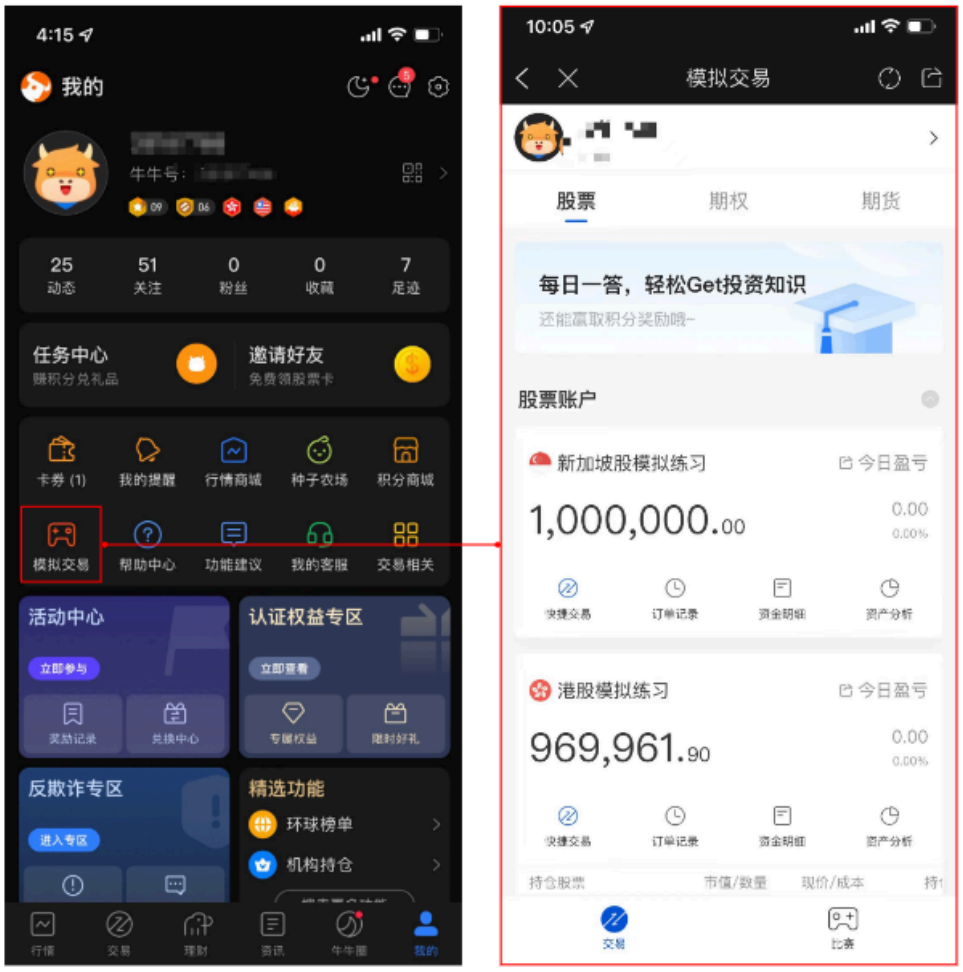
OpenAPI 支持模擬交易的品類請參考 這裏。

訂單

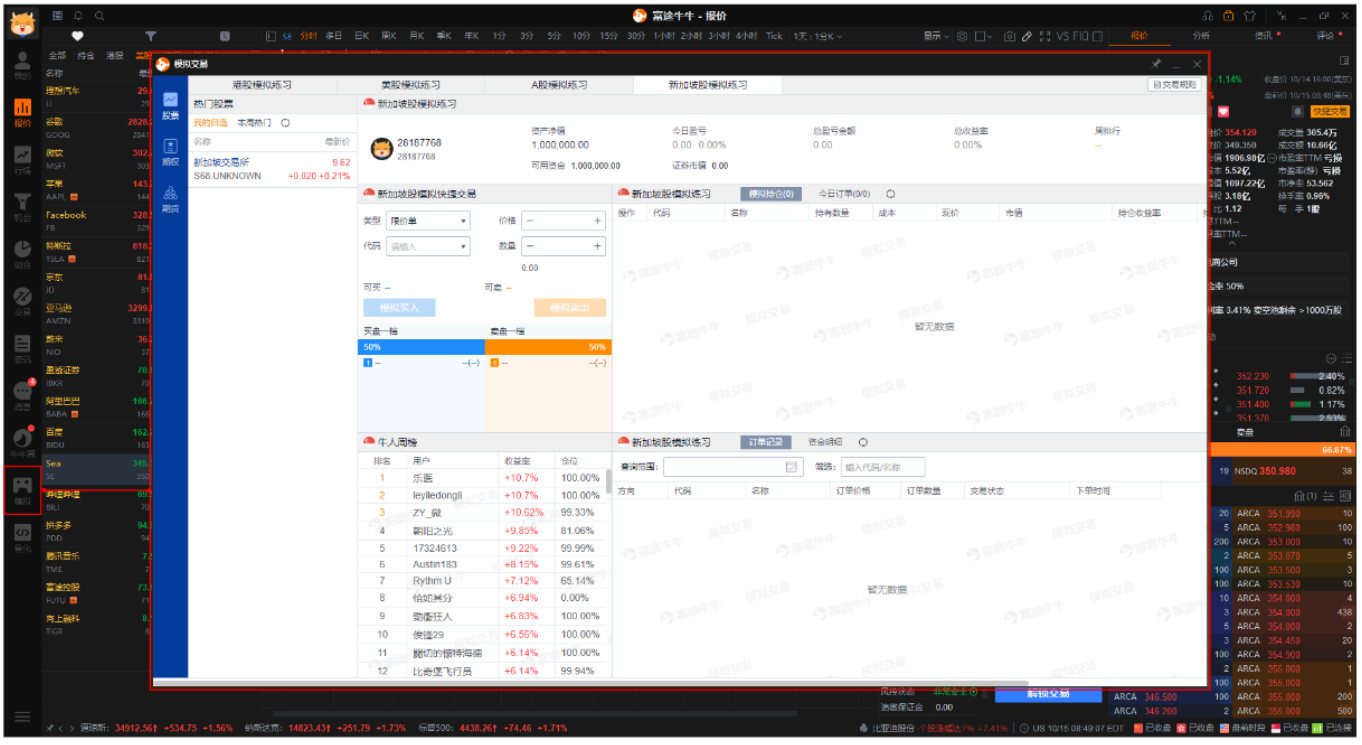
1. 訂單類型：限價單和市價單。
2. 修改訂單操作類型：模擬交易不支持使生效、使失效、刪除，僅支持支持修改訂單、 取消訂單。
3. 成交：模擬交易不支持成交相關操作，包括 查詢今日成交、查詢歷史成交、響應成交推送回調。
4. 有效期限：模擬交易有效期限僅支持當日有效。
5. 賣空：期權和期貨支持賣空。股票僅美股支持賣空。

操作平台

1. 手機版：我的 — 模擬交易



2. 桌面版：左側模擬 tab



3. 網頁版：模擬交易界面

4. OpenAPI：在調用接口時，設置參數交易環境為模擬環境即可。詳見 如何使用 **OpenAPI** 進行模擬交易。

> **提示**
>
> - 以上四種方式只是操作平台不同，四種方式操作的模擬帳戶是共通的。

如何使用 OpenAPI 進行模擬交易？

創建連接

先根據交易品種 創建相應的連接 。當交易品種是股票或期權時，請使用 `OpenSecTradeContext` 。當交易品種是期貨時，請使用 `OpenFutureTradeContext` 。

獲取交易業務帳戶列表

使用 獲取交易業務帳戶列表 查看交易帳戶（包括模擬帳戶、真實帳戶）。以 Python 為例：返回欄位交易環境 `trd_env` 為 `SIMULATE` ，表示模擬帳戶。

- Example: Stocks and Options

```py
from moomoo import *
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, host='127.0.0.1', port=11111, security_firm
#trd_ctx = OpenFutureTradeContext(host='127.0.0.1', port=11111, is_encrypt=None, security_firm=SecurityF
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]
```

> **提示**
>
> - 模擬交易中，區分股票帳戶和期權帳戶，股票帳戶只能交易股票，期權帳戶只能交易期權；以 Python 為例：返回欄位中模擬帳戶類型 `sim_acc_type` 為 `STOCK` ，表示股票帳戶；為 `OPTION` ，表示期權帳戶。

- Example: Futures

```py
from moomoo import *
#trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.HK, host='127.0.0.1', port=11111, security_fir
trd_ctx = OpenFutureTradeContext(host='127.0.0.1', port=11111, is_encrypt=None, security_firm=SecurityFi
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 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]
```

下單

使用 下單接口 時，設置交易環境為模擬環境即可。以 Python 為例：`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
ret, data = trd_ctx.place_order(price=510.0, qty=100, code="HK.00700", trd_side=TrdSide.BUY, trd_env=Trd
if ret == RET_OK:
    print(data)
else:
    print('place_order error: ', data)
trd_ctx.close()
```

- Output

```py
     code stock_name    trd_side order_type    order_status  order_id qty  price     create_time    updat
0    HK.00700 騰訊控股   BUY  NORMAL    SUBMITTING    4642000476506964749    100.0    510.0    2021-10-09
```

取消訂單修改訂單

使用 取消訂單接口 時，設置交易環境為模擬環境即可。以 Python 為例：`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
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

```py
    trd_env             order_id
0  SIMULATE  4642000476506964749
```

查詢歷史訂單

使用 查詢歷史訂單接口 時，設置交易環境為模擬環境即可。以 Python 為例：`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
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

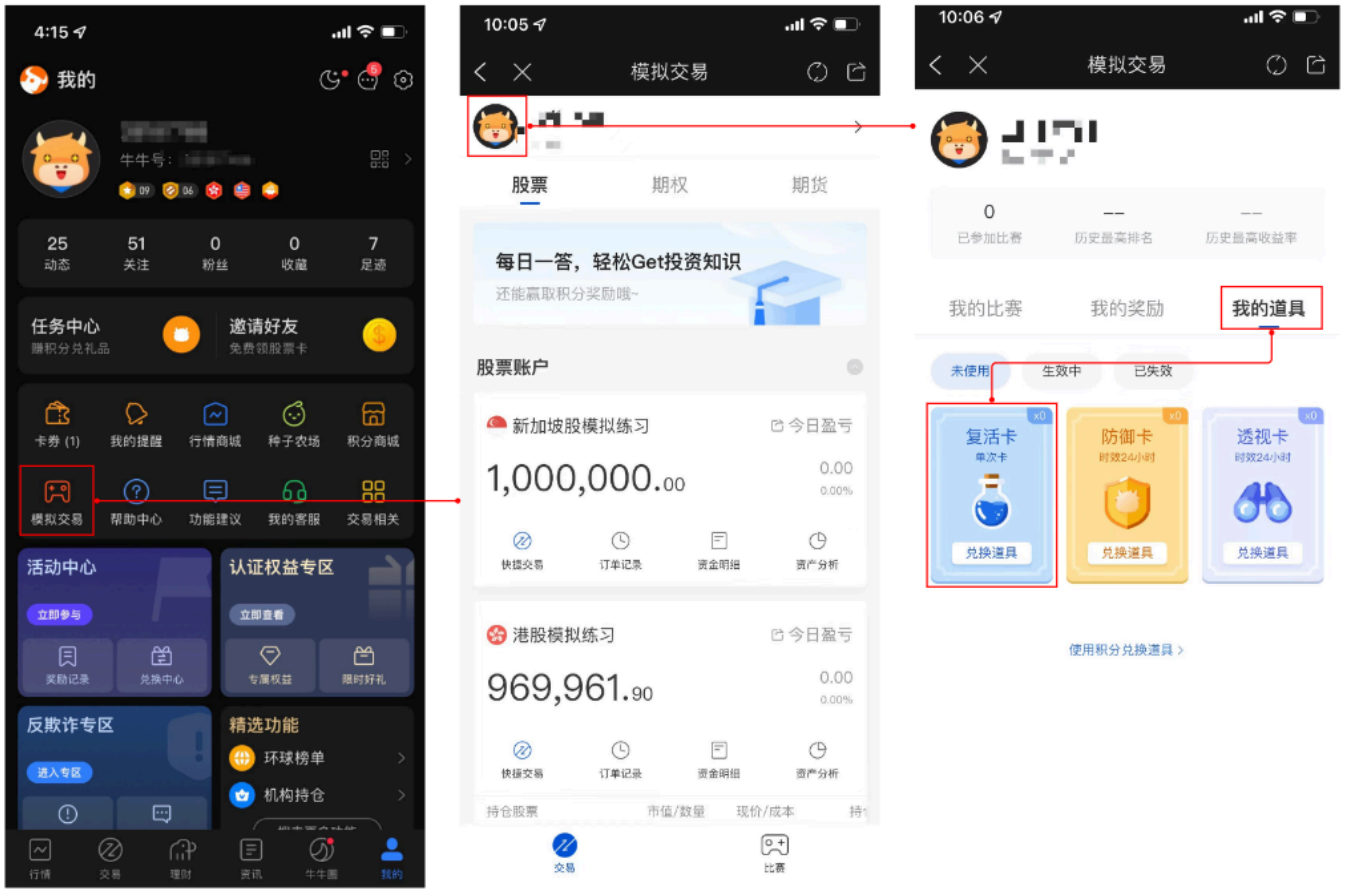
```py
     code stock_name    trd_side order_type    order_status  order_id qty  price    create_time   updat
0  HK.00700  騰訊控股  BUY  ABSOLUTE_LIMIT    CANCELLED_ALL 4642000476506964749    100.0    510.0
```

如何重置模擬帳戶？

目前 OpenAPI 不支持重置模擬帳戶，您可在手機版使用復活卡重置指定模擬帳戶，重置後帳戶資金將恢復至初始值，歷史訂單將會被清空。

具體操作

手機版：我的 — 模擬交易 — 我的頭像 — 我的道具 — 復活卡。



Q2：是否支持 A 股交易？

A: 模擬交易支持 A 股交易。但真實交易僅可通過 A 股通交易部分 A 股，具體詳見 A 股通名單。

Q3：各市場支持的交易方向

A: 除了期貨，其他股票都只支持傳入 BUY 和 SELL 兩個交易方向。在空倉情況下傳入 SELL，產生的訂單交易方向是賣空。

Q4：真實交易中，各市場支持的訂單類型

A:

市場	品種	限價單	市價單	競價限價單	競價市價單	絕對限價單	特別限價單	特別限價且要求全部	止損市價單	止損限價單	觸及市價單（止盈）	觸及限價單（止盈）	跟蹤止損市價單	跟蹤止損限價單

								成交訂單					
香港市場	證券類產品（含股票、ETFs、窩輪、牛熊、界內證）	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	期權	✓	X	-	-	-	-	-	X	✓	X	✓	X	✓
	期貨	✓	✓	-	✓	-	-	-	✓	✓	✓	✓	✓	✓
美國市場	證券類產品（含股票、ETFs）	✓	✓	-	-	-	-	-	✓	✓	✓	✓	✓	✓
	期權	✓	✓	-	-	-	-	-	✓	✓	✓	✓	✓	✓
	期貨	✓	✓	-	-	-	-	-	✓	✓	✓	✓	✓	✓
A股通市場	證券類產品（含股票、ETFs）	✓	X	-	-	-	-	-	X	✓	X	✓	X	✓
新加坡市場	期貨	✓	✓	-	-	-	-	-	✓	✓	✓	✓	✓	✓
日本市場	期貨	✓	✓	-	-	-	-	-	✓	✓	✓	✓	✓	✓

Q5：各市場支持的訂單操作

A:

- 港股支持修改訂單、取消訂單、生效、失效、刪除
- 美股僅支持修改訂單和取消訂單
- A 股通僅支持取消訂單
- 期貨支持修改訂單、取消訂單、刪除

Q6：OpenD 啟動參數 future_trade_api_time_zone 如何使用？

A：由於期貨帳戶支持交易的品種分佈在全球多個交易所，交易所的所屬時區各有不同，因此期貨交易 API 的時間顯示就成為了一個問題。
OpenD 啟動參數中新增了 future_trade_api_time_zone 這一參數，供全球不同地區的期貨交易者靈活指定時區。預設時區為 UTC+8，如果您更習慣美東時間，只需將此參數設定為 UTC-5 即可。

> **提示**
>
> - 此參數僅會對期貨交易接口類對象生效。港股交易、美股交易、A 股通交易接口類對象的時區，仍然按照交易所所在的時區進行顯示。
> - 此參數會影響的接口包括：響應訂單推送回調，響應成交推送回調，查詢今日訂單，查詢歷史訂單，查詢當日成交，查詢歷史成交，下單。

Q7：通過 OpenAPI 下的訂單，能在 APP 上面看到嗎？

A：可以看到。
通過 OpenAPI 成功發出下單指示後，您可以在 APP 的 **交易** 頁面，查看今日訂單、訂單狀態、成交情況等等，也可以在 **消息—訂單消息** 中收到成交提醒的通知。

Q8：哪些品類支持在非交易時段下單？

A：所有的訂單，都需要在開市期間才能夠成交。
OpenAPI 僅對一部分品類，支持了 **非交易時段下單** 的功能（APP 上支持更多品類的非交易時段下單功能）。具體請參考下表：

市場	標的類型	模擬交易	真實交易						
			Futu HK	Moomoo US	Moomoo SG	Moomoo AU	Moomoo MY	Moomoo CA	Moomoo JP
香港市場	股票、ETFs、窩輪、牛熊、界內證	✓	✓	✓	✓	✓	✓	X	X
	期權	✓	✓	X	X	X	X	X	X
	期貨	✓	✓	X	X	X	X	X	X

美國市場	股票、ETFs	✓	✓	✓	✓	✓	✓	✓	✓
	期權	✓	✓	✓	✓	✓	✓	✓	✓
	期貨	✓	✓	X	✓	X	✓	X	X
A股市場	A 股通股票	✓	✓	✓	✓	X	X	X	X
	非 A 股通股票	✓	X	X	X	X	X	X	X
新加坡市場	股票、ETFs、窩輪、REITs、DLCs	X	X	X	X	X	X	X	X
	期貨	✓	✓	X	✓	X	X	X	X
日本市場	股票、ETFs、REITs	X	X	X	X	X	X	X	X
	期貨	✓	✓	X	X	X	X	X	X
澳大利亞市場	股票、ETFs	X	X	X	X	X	X	X	X
加拿大市場	股票	X	X	X	X	X	X	X	X

> **提示**
>
> - ✓：支持非交易時段下單
> - X：暫不支持非交易時段下單（或暫不支持交易）

Q9：對於下單接口，各訂單類型對應的必要參數以及券商對單筆訂單的

下單限制

A1: 各訂單類型對應的必要參數

參數	限價單	市價單	競價限價單	競價市價單	絕對限價單	特別限價單	特別限價且要求全部成交訂單	止損市價單	止損限價單	觸及市價單（止盈）	觸及限價單（止盈）	跟蹤止損市價單	跟蹤止損限價單
price	✓		✓		✓	✓	✓		✓		✓		
qty	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
code	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
trd_side	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
order_type	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
trd_env	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
aux_price								✓	✓	✓	✓		
trail_type												✓	✓
trail_value												✓	✓
trail_spread													✓

Python 用户 注意，place_order 並未對 price 設置預設值，對於上述五類訂單類型，仍需對 price 傳參，price 可以傳入任意值。

A2：各券商對單筆訂單的股數及金額限制

券商	品類	單筆訂單股數上限	單筆訂單金額上限
FUTU HK	A股通	1,000,000 股	￥5,000,000
	美股	500,000 股	$5,000,000
	香港股票期貨/期權	3,000 手	無限制
moomoo US	美股	500,000 股	$10,000,000
moomoo SG	美股	500,000 股	$5,000,000
moomoo AU	美股	無限制	無限制

Q10：對於修改訂單接口，修改訂單時，各訂單類型對應的必要參數

A:

參數	限價單	市價單	競價限	競價市	絕對限	特別限	特別限	止損市	止損限	觸及市價單	觸及限價單	跟蹤止	跟蹤止

			價單	價單	價單	價單	價且要求全部成交訂單	價單	價單	（止盈）	（止盈）	損市價單	損限價單
modify_order_op	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
order_id	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
price	✓		✓		✓	✓	✓		✓		✓		
qty	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
trd_env	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
aux_price								✓	✓	✓	✓		
trail_type												✓	✓
trail_value												✓	✓
trail_spread													✓

Python 用户 注意，modify_order 並未對 price 設置預設值，對於上述五類訂單類型，仍需對 price 傳參，price 可以傳入任意值。

Q11：交易接口返回“當前證券業務帳戶尚未同意免責協議”？

A：
點擊下方連結完成協議確認，重啟 OpenD 即可正常使用交易功能。

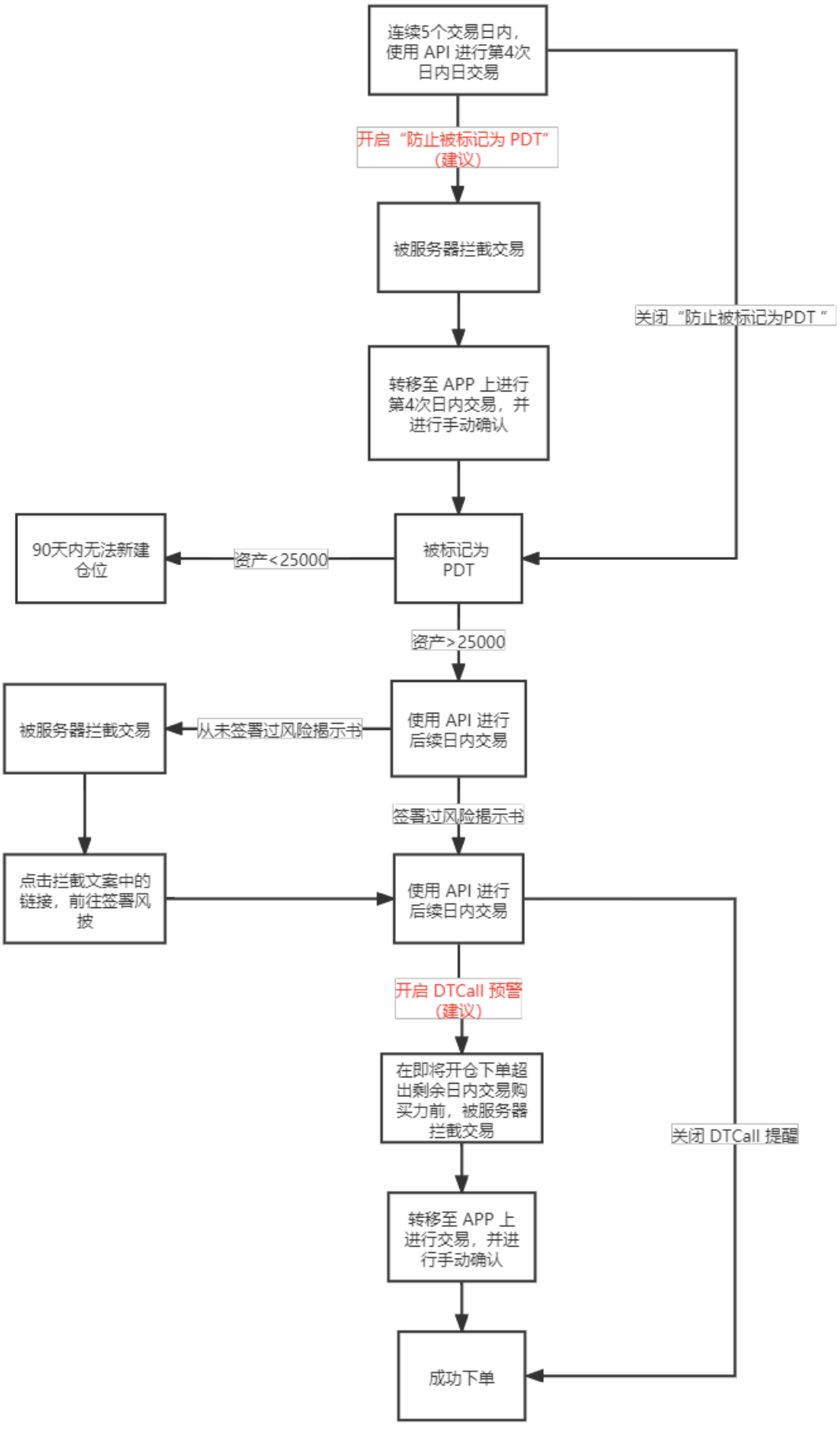
所屬券商	協議確認
FUTU HK	點擊這裏
Moomoo US	點擊這裏
Moomoo SG	點擊這裏
Moomoo AU	點擊這裏
Moomoo CA	點擊這裏
Moomoo MY	點擊這裏
Moomoo JP	點擊這裏

Q12：典型日內交易者（PDT）相關

概述

客户使用moomoo證券(美國) 帳戶進行日內交易時，會受到美國 FINRA 的監管限制（此為美國券商受到的監管要求，與交易股票的所屬市場無關。其他國家或地區的券商 ⓘ 的交易帳戶則不受此限制）。若用户在任意連續的5個交易日內，進行日內交易 3 次以上，則會被標記為典型日內交易者（PDT）。
更多詳情，點擊這裏

進行日內交易的流程圖



我願意被標記為 PDT，且不希望程式交易被打斷，如何關閉“防止被標記為 PDT”？

A：
當您在連續的 5 個交易日內，進行第 4 次日內交易時，為了防止您被無意識地標記為 PDT，服務器會對此交易進行攔截。若您主動想被標記為 PDT，並且不希望服務器攔截，可以採取以下措施：
在 命令行 OpenD 中設定參數，將啟動參數 **pdt_protection** 的值修改為 0，以關閉"防止被標記為日內交易者"的功能。

```
<!-- FUTU US 专用参数 -->
<!-- Specific parameters for FUTU 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$25000时，您将无法开仓。-->
    <!-- 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 Trader(PDT). The Protection c
    <pdt_protection>1</pdt_protection>
```

注意：若您被標記 PDT，當您的帳戶權益小於$25000時，您將無法開倉。

如何關閉 DTCall 預警提醒？

A：
您被標記為 PDT 後，需要留意帳戶的日內交易購買力（DTBP），日內交易超出 DTBP 時將收到日內交易保證金追繳（DTCall）。服務器會在您即將開倉下單超出剩餘日內交易購買力前，阻止您的下單。若您仍然希望進行下單，並且不希望服務器攔截，可以採取以下措施：
在 命令行 OpenD 中設定參數，將啟動參數 **dtcall_confirmation** 的值修改為 0，以關閉"日內交易保證金追繳預警"的功能。

```
<!-- 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>
```

注意：若您開倉訂單的市值大於您的剩餘日內交易購買力，並且在今日平倉當前標的，您將會收到日內交易保證金追繳通知（Day-Trading Call），只能通過存入資金才能解除。

如何查看 DTBP 的值？

A：
通過 查詢帳戶資金 接口，可以獲取日內交易相關的返回值，如：剩餘日內交易次數、初始日內交易購買力、剩餘日內交易購買力等。

Q13：如何跟蹤訂單成交狀態

A: 下單後，可使用以下接口跟蹤訂單成交狀態：

交易環境	接口
真實交易	響應訂單推送回調，響應成交推送回調
模擬交易	響應訂單推送回調

注意：對於非 python 語言用户，在使用上述兩個接口之前，需要先進行 訂閱交易推送

響應訂單推送回調 的特點：

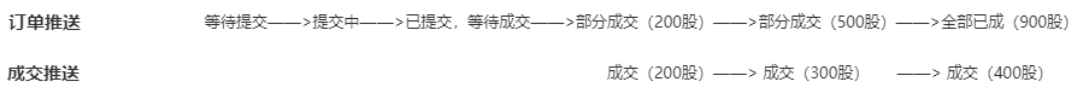
反饋 整個訂單 的資訊變動。當以下 8 個字段發生變化時，會觸發訂單推送：
訂單狀態 · **訂單價格** · **訂單數量** · **成交數量** · **觸發價格** · **跟蹤類型** · **跟蹤金額/百分比** · **指定價差**

因此，當您進行下單、修改訂單，取消訂單、使生效、使失效操作，或者訂單在市場中發生了高級訂單被觸發、有成交變動的情況，都會觸發訂單推送。您只需要調用 響應成交推送回調，即可監聽這些資訊。

響應成交推送回調 的特點：

只反饋 單筆成交 的資訊。當以下 1 個字段發生變化時，會觸發訂單推送：
成交狀態

舉例：假設一筆限價單訂單 900 股，分成了 3 次才完全成交，每次成交分別是：200、300、400 股。



Q14：下單接口返回“此產品最小單位為 xxx，請調整至最小單位的整數倍後再提交”？

A:
對於不同市場的標的，交易所有着不同的最小變動單位要求。如果提交的訂單價格不符合要求，訂單將會被拒絕。各市場價位規則如下：

價位規則

香港市場

以港交所官方說明為準，點擊 這裏。

A 股市場

股票價位：0.01。

美國市場

股票價位：

合約價格	價位
$1 以下	$0.0001
$1 以上	$0.01

期權價位：

合約價格	價位
$0.10 - $3.00	$0.01 或者 $0.05

$3.00 以上	$0.05 或者 $0.10

期貨價位：不同合約價位規則不同。可以通過 獲取期貨合約資料 接口的返回欄位 **最小變動的單位** 查看。

怎麼避免訂單價格不在價位上？

- 方法一：通過 獲取實時擺盤 接口，獲取合法的交易價格。交易所擺盤上的價位一定是合法的價位。
- 方法二：通過 下單 接口的參數 **價格微調幅度** ，將傳入價格自動調整到合法的交易價格上。

  例如：假設騰訊控股當前市價為 359.600，根據價位規則，對應的最小變動價位為 0.200。

  假設您的下單傳入訂單價格為 359.678，價格微調幅度為 0.0015，代表接受 OpenD 對傳入價格自動向上調整到最近的合法價位，且不能超過 0.15%。此情景下，向上最近的合法價格為 359.800，價格實際需要調整的幅度為 0.034%，符合價格微調幅度的要求，因此最終提交的訂單價格為 359.800。

  若價格微調幅度設置數值小於實際需要調整的幅度，OpenD 自動調整價位失敗，訂單仍會返回報錯"訂單價格不在價位上"。

Q15：我的購買力足夠，為什麼下市價單會返回"購買力不足"？

A：

為什麼市價單會提示購買力不足

- 出於風控考量，系統給了市價單較高的購買力系數。在所有訂單參數都相同的情況下，選擇市價單會比限價單佔用更多的購買力。
- 而且對於不同的品種，和不同的市場情況，風控系統會對市價單的購買力系數做動態調整。所以在下市價單時，若您通過最大購買力去計算最大可買數量，計算的結果很可能是不準確的。

如何計算正確的可買數量

不建議自己計算，您可以通過 查詢最大可買可賣 接口獲取正確的可買數量。

如何儘可能買更多

您可以用價格為對價的限價單，替代市價單進行交易。
其中，對價：買1價（下賣單時）或 賣1價（下買單時）

Q16：API模擬交易下單，為什麼手機版看不到？

A：
手機版、桌面版、網頁版，美股模擬交易帳戶，已經從【美股模擬帳戶】升級成為功能更豐富的【美股融資融券帳戶】。
OpenAPI 暫未升級（規劃中），目前只能使用舊的【美股模擬帳戶】，且舊的【美股模擬帳戶】無法在其他客户端上展示，請謹慎使用。

Q17：交易接口參數使用説明

1. 什麼是交易對象？

您的平台賬號下一般會開設一個保證金綜合帳戶，其中有多個交易子帳戶（正常有兩個，一個綜合證券帳戶，一個綜合期貨帳戶；根據需要還可能有綜合外匯帳戶等其他子帳戶）。一些特殊用户或機構客户可能會在多個券商下開設多個綜合帳戶。
創建交易對象，是初步篩選子帳戶的過程。

- 使用 OpenSecTradeContext 創建的交易對象，調用 get_acc_list 時只會返回**證券交易帳戶**
- 使用 OpenFutureTradeContext 創建的交易對象，調用 get_acc_list 時只會返回**期貨交易帳戶**

參數 security_firm 用來篩選對應歸屬券商的帳戶，參數 filter_trdmarket 用來篩選對應交易市場權限的帳戶。

1.1 security_firm 券商參數

OpenAPI 目前支持的券商有 這些。
創建的交易對象，在調用 get_acc_list 時，會返回 security_firm 對應券商的真實帳戶和所有模擬交易帳戶（這是因為模擬交易沒有券商的概念，所以無論 security_firm 傳什麼，都會返回所有的模擬帳戶）。
security_firm 的預設值是 FUTUSECURITIES，FUTU HK 券商帳戶可以不填此參數，但需要獲取其他券商的帳戶時，需要修改券商參數。

- 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_typ
0  281756478396547854       REAL   MARGIN  1001200163530138  1001369091153722  FUTUSECURITIES          N/
1             3450309  SIMULATE     CASH               N/A               N/A             N/A        STOC
2             3548731  SIMULATE   MARGIN               N/A               N/A             N/A       OPTIC
3  281756455998014447       REAL   MARGIN               N/A  1001100320482767  FUTUSECURITIES          N/
```

- Example 2

```py
trd_ctx = OpenSecTradeContext(security_firm=SecurityFirm.FUTUSG)
ret, data = trd_ctx.get_acc_list()
print(data)
```

- Output

```
    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 交易市場參數

OpenAPI 目前支持的交易市場有 這些。
創建的交易對象，在調用 get_acc_list 時，會返回所有擁有 filter_trdmarket 市場交易權限的帳戶；當 filter_trdmarket 入參傳 NONE 時，不過濾市場，返回所有的帳戶。
filter_trdmarket 的預設參數是 HK，在綜合帳戶體系下，這個參數用來篩選不同市場下的模擬交易帳戶。

- Example 1

```py
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.US)
ret, data = trd_ctx.get_acc_list()
print(data)
```

- Output

```
              acc_id   trd_env acc_type      uni_card_num          card_num   security_firm sim_acc_typ
0  281756478396547854      REAL   MARGIN  1001200163530138  1001369091153722  FUTUSECURITIES          N/
1            3450310  SIMULATE   MARGIN              N/A               N/A             N/A        STOC
2            3548732  SIMULATE   MARGIN              N/A               N/A             N/A       OPTIC
3  281756460292981743      REAL   MARGIN              N/A  1001100520714263  FUTUSECURITIES          N/
```

- Example 2

```py
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.NONE)
ret, data = trd_ctx.get_acc_list()
print(data)
```

- Output

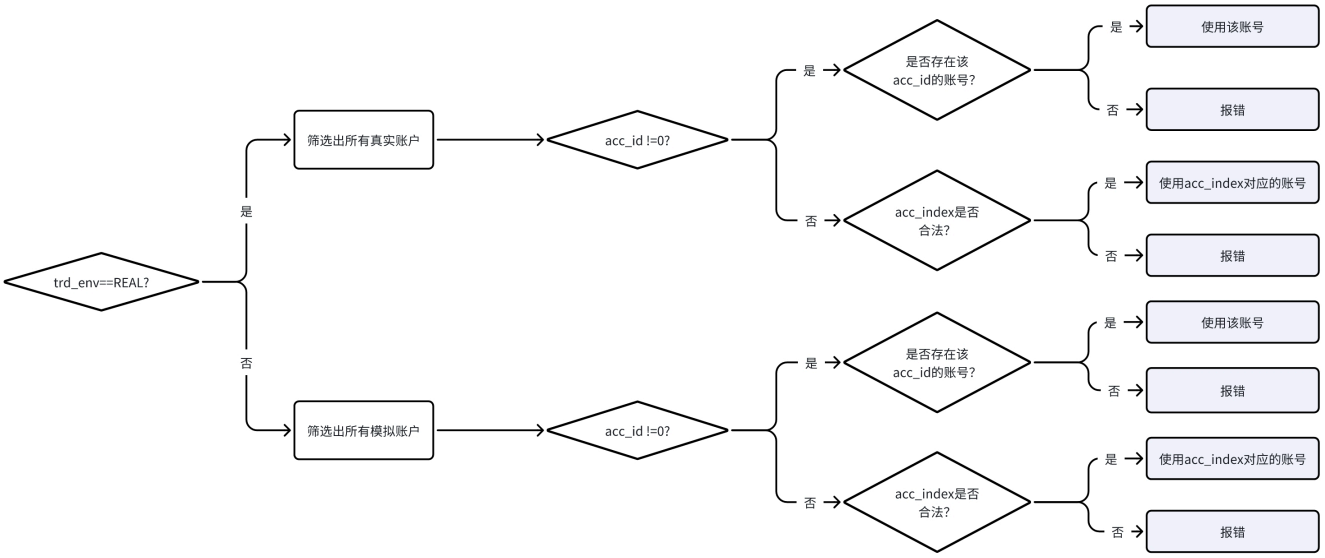
```
               acc_id   trd_env acc_type      uni_card_num          card_num   security_firm sim_acc_ty
0  281756478396547854      REAL   MARGIN  1001200163530138  1001369091153722  FUTUSECURITIES          N
1            3450309  SIMULATE     CASH              N/A               N/A             N/A        STO
2            3450310  SIMULATE   MARGIN              N/A               N/A             N/A        STO
3            3450311  SIMULATE     CASH              N/A               N/A             N/A        STO
4            3548732  SIMULATE   MARGIN              N/A               N/A             N/A       OPTI
5            3548731  SIMULATE   MARGIN              N/A               N/A             N/A       OPTI
6  281756455998014447      REAL   MARGIN              N/A  1001100320482767  FUTUSECURITIES          N
7  281756460292981743      REAL   MARGIN              N/A  1001100520714263  FUTUSECURITIES          N
```

```
8   281756468882916335    REAL   MARGIN          N/A  1001100610464507  FUTUSECURITIES
9   281756507537621999    REAL     CASH          N/A  1001100910390035  FUTUSECURITIES
10  281756550487294959    REAL     CASH          N/A  1001101010406844  FUTUSECURITIES
```

> **提示**
>
> 當 filter_trdmarket 入參NONE時，可以返回所有的交易帳戶。其中第0行是真實帳戶，1~5行均為模擬交易帳戶，6~10行是已失效的真實帳戶。這些失效帳戶都是單市場帳戶，現已被綜合帳戶替代。但歷史訂單和歷史成交還在這些已失效的帳戶中，可以通過這些帳戶來查詢。
>
> OpenFutureTradeContext 對象中沒有 filter_trdmarket 參數，只有 security_firm 參數，功能與 OpenSecTradeContext 一樣。

2. 交易接口參數

在使用具體的交易接口（如下單、查詢訂單列表）時，接口中的 `trd_env`，`acc_index` 和 `acc_id` 參數，會先篩選確認一個唯一的帳戶，對此帳戶實施對應的接口行為。



> **總結**
>
> 1. 根據 trd_env 篩選出真實帳戶還是模擬帳戶
> 2. 在篩選結果中，優先選擇 acc_id 指定的帳戶
> 3. 如果 acc_id 為0，則通過 acc_index選取對應賬號
> 4. 報錯場景：指定的 acc_id 不存在，或 acc_index 超出範圍

3. 應用舉例

3.1 綜合證券帳戶實盤下單

```py
trd_ctx = OpenSecTradeContext(filter_trdmarket=TrdMarket.NONE, security_firm=SecurityFirm.FUTUSECURITIES
ret, data = trd_ctx.unlock_trade("123123")
if ret == RET_OK:
```

```
    print("解鎖成功")
    ret, data = trd_ctx.place_order(45, 200, 'HK.00700', TrdSide.BUY,
                                    order_type=OrderType.NORMAL,
                                    trd_env=TrdEnv.REAL,  # 和預設參數一樣，可以不填
                                    acc_id=0)  # 和預設參數一樣，可以不填
    print(data)
```

3.2 綜合期貨帳戶查詢實盤訂單列表

```py
trd_ctx = OpenFutureTradeContext(security_firm=SecurityFirm.FUTUSECURITIES)

ret, data = trd_ctx.order_list_query(trd_env=TrdEnv.REAL,   # 和預設參數一樣，可以不填
                                     acc_id=0)  # 和預設參數一樣，可以不填
print(data)
```

3.3 港股模擬現金帳戶查詢帳戶資金

```py
# filter_trdmarket 填 TrdMarket.HK
# trd_env 填 TrdEnv.SIMULATE
# acc_index 填 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 美股模擬保證金帳戶下單期權

```py
# 通過 filter_trdmarket 和 trd_env 篩選完之後只剩兩個帳戶
# 第0個是美股現金帳戶（交易股票）,第1個是美股保證金帳戶（交易期權）
# acc_index 填 1 指定美股保證金帳戶
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 日本期貨模擬帳戶查詢最大可買賣

```py
# 將 get_acc_list 的結果打印出來，可以看到日本期貨模擬帳戶的 acc_id 是 6271199
# 請求最大可買賣接口時傳入這個 acc_id
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. OpenAPI 中的帳戶如何與 APP/桌面版對應

APP 上的帳戶僅顯示卡號後 4 位數字，我們將 get_acc_list 的返回結果打印出來後，有 uni_card_num 列和 card_num 列，分別對應綜合帳戶的卡號，和單幣種帳戶（已廢棄）的卡號。通過卡號後 4 位數就能把 API 中獲取到的賬號與 APP 上對應起來了。

其他

Q1：如何編譯C++ API？

A: moomoo api c++ SDK支持Windows/MacOS/Linux，每個系統提供了以下編譯環境生成的程式庫檔案：

操作系統	編譯工具
Windows	Visual Studio 2013
Centos 7	g++ 4.8.5
Ubuntu 16.04	g++ 5.4.0
MacOS	XCode 11

如果編譯器版本不同，或相依性項的protobuf版本不同，則可能需要自己使用原始碼重新編譯MMAPI和protobuf，原始碼位置見下圖目錄：

```
MMAPI目錄結構：
+---Bin                                存放各個系統預設編譯環境編譯出的相依性項庫
+---Include                            存放公共標頭檔，以及proto協議生成的.h/.cc文件
+---Sample                             示例專案
\---Src
    +---MMAPI                          MMAPI原始碼
    +---protobuf-all-3.5.1.tar.gz      protobuf原始碼
```

編譯步驟：

1. 重新編譯protobuf：生成libprotobuf靜態程式庫
2. 從協議proto檔案中生成C++檔案
3. 重新編譯MMAPI: 原始碼在Src/MMAPI，生成libMMAPI靜態程式庫

步驟1： 重新編譯protobuf：

- Windows：

    - 安裝CMake
    - 打開VS命令行工具，cd到protobuf/cmake目錄
    - 執行：cmake -G "Visual Studio 12 2019" -DCMAKE_INSTALL_PREFIX=install -Dprotobuf_BUILD_TESTS=OFF 這樣會生成Visual Studio 2019的項目檔案，其它版本Visual Studio請修改-G參數
    - 打開生成的Visual Studio項目檔案，平台工具組設置為v120_xp，編譯即可
- Linux（參考protobuf/src/README）
    - 執行 ./autogen.sh
    - 執行 CXXFLAGS="-std=gnu++11" ./configure --disable-shared
    - 執行 make
    - 將生成的libprotobuf.a放入Bin/Linux目錄
- MacOS（參考protobuf/src/README）
    - 使用brew安裝這些相依性項庫：autoconf automake libtool
    - 執行./configure CC=clang CXX="clang++ -std=gnu++11 -stdlib=libc++" --disable-shared

步驟2: 重新生成proto代碼

- 上面編譯Protobuf後會同時生成可執行檔案protoc。用protoc將Include/Proto下面的.proto檔案生成對應的.h和.cc檔案。例如命令以下命令會從Common.proto生成對應的Common.pb.h和Common.pb.cc
    - protoc -I="MMAPI路徑/Include/Proto" --cpp_out="." MMAPI路徑/Include/Proto/Common.proto
- 將生成的.h和.cc檔案放到Include/Proto下面

步驟3: 重新編譯MMAPI

- Windows：新建Visual Studio C++靜態程式庫專案，將Src/MMAPI和Include下的原始碼加入專案中，平台工具組設置為v120_xp，然後編譯
- Mac：新建XCode C++靜態程式庫專案，將Src/MMAPI和Include下的原始碼加入專案中，然後編譯
- Linux：使用CMake編譯MMAPI靜態程式庫，在MMAPI路徑/Src目錄下執行：
    - cmake -DTARGET_OS=Linux

Q2：有沒有更完整的策略範例可以參考？

A:

- Python 策略範例在 /moomoo/examples/ 資料夾下。您可以通過執行如下命令，找到 Python API 的安裝路徑：

```
import moomoo
print(moomoo.__file__)
```

- C# 策略範例在 /MMAPI4NET/Sample/ 資料夾下
- Java 策略範例在 /MMAPI4J/sample/ 資料夾下
- C++ 策略範例在 /MMAPI4CPP/Sample/ 資料夾下
- JavaScript 策略範例在 /MMAPI4JS/sample/ 資料夾下

Q3：使用 python API 匯入異常

場景一：已經在 Python 環境中安裝了 moomoo 模組，仍然提示 No module named 'moomoo'？
很可能是因為當前 IDE 所使用的 interpreter 並不是你裝過 moomoo 模組的 interpreter。也就是說，您的電腦可能裝了兩個以上的 Python 環境。 您可以操作如下兩步：

1. 在 Python 中運行如下代碼，得到當前 interpreter 的路徑：

```
import sys
print(sys.executable)
```

示例圖：

```
In[3]: import sys
  ...: print(sys.executable)
D:\software\anaconda3\python.exe
```

2. 在命令行中，執行 `$ D:\software\anaconda3\python.exe -m pip install moomoo-api`（其中前半部分的檔案路徑來自第 1 步打印的路徑）。 這樣就可以在當前的 interpreter 中也安裝一份 moomoo 模組。

Q4： import 成功了，仍然調用不了相關接口？

A：通常遇到這種情況，需要確認一下：成功匯入的 moomoo，是不是真正的 moomoo API 模組。以下幾種場景也可能 import 成功。

場景一：存在與“moomoo”重名的檔案

1. 當前檔案名是 moomoo.py
2. 當前檔案所在目錄下存在另一個名為 moomoo.py 的檔案
3. 當前檔案所在目錄下存在名為 `/moomoo` 的資料夾

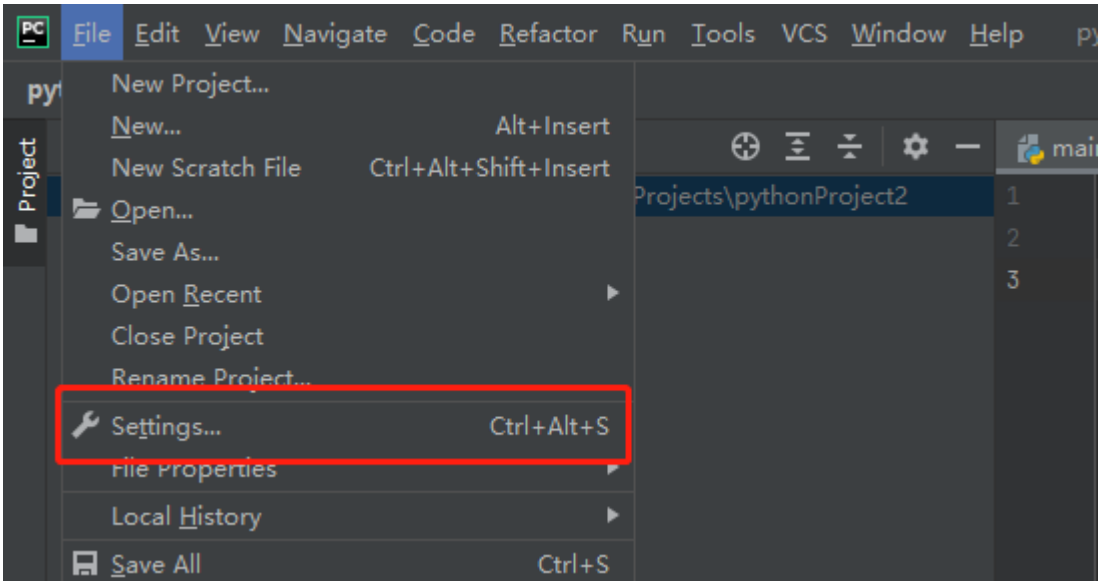
因此，我們強烈建議您，在給檔案 / 資料夾 / 專案起名的時候，不要起名叫“moomoo”。重名一時爽，查 bug 兩行淚。

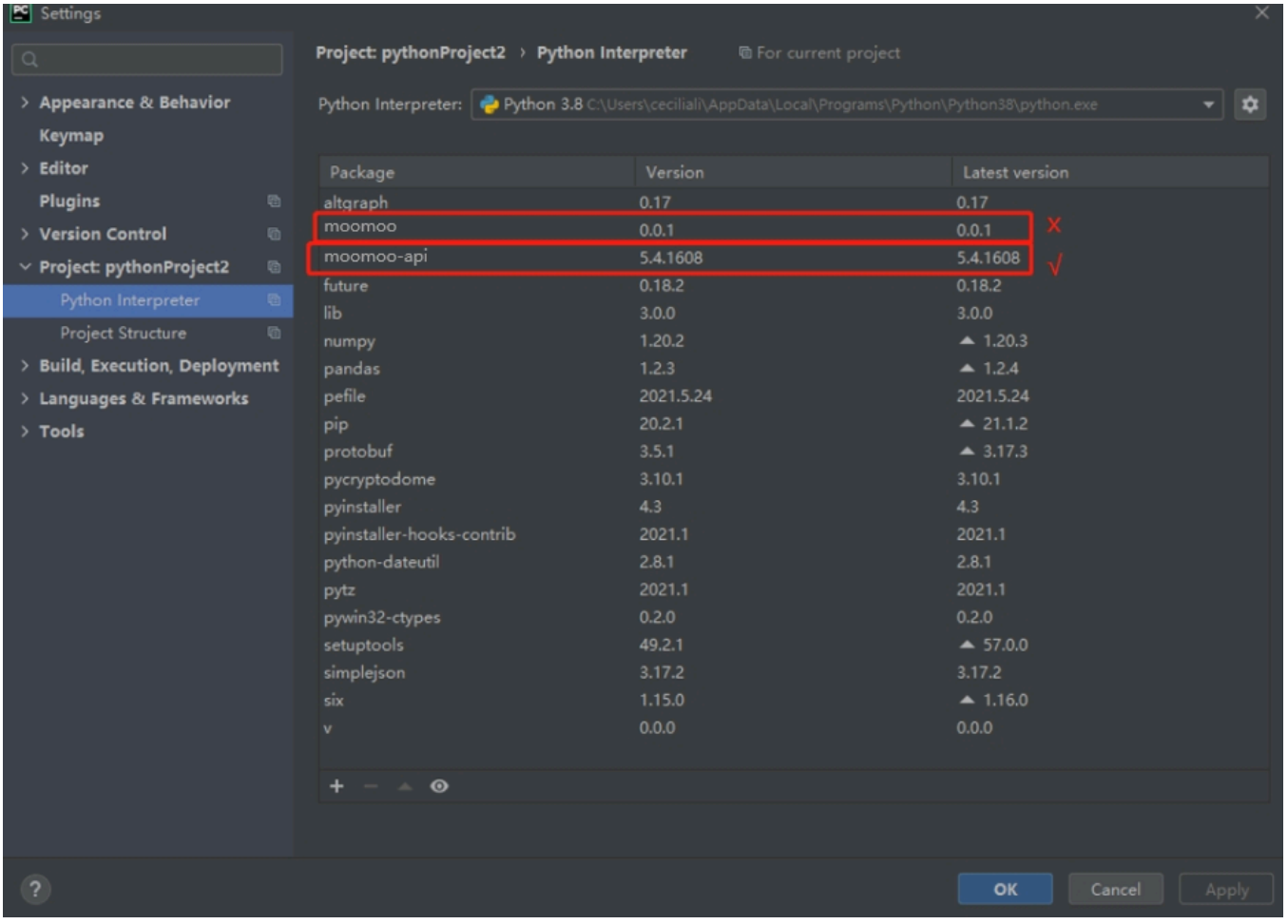
場景二：誤安裝了一個名為“moomoo”的第三方程式庫

moomoo API 的正確名稱為 `moomoo-api` ，而非“moomoo”。

如果您安裝過名為“moomoo”的第三方程式庫，請將其卸載，並 下載 **moomoo-api**。

以 PyCharm 為例：查看第三方程式庫的安裝情況。





Q5：協議加密相關

A:

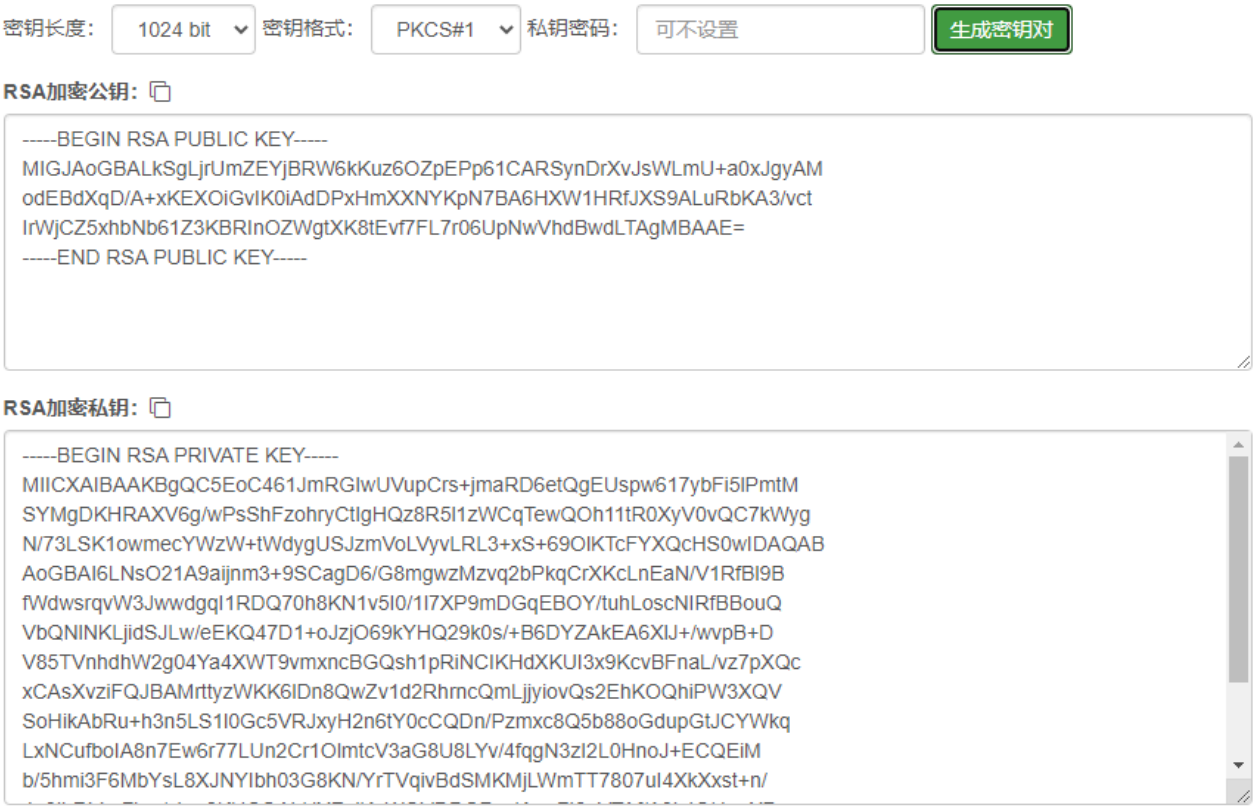
概述

您可以使用非對稱加密演算法 RSA，對策略程式（moomoo API）與 OpenD 之間的請求和返回內容進行加密，以保證通信安全。
如果您的策略程式（moomoo API）與 OpenD 在同一台電腦上，則通常無需加密。

協議加密流程

您可以嘗試通過以下步驟解決此問題：

1. 通過第三方 web 平台自動生成密鑰檔案。
   - 具體方法：在 baidu 或 google 上搜索"RSA 在線生成"，**密鑰格式**設置為 PKCS#1，**密鑰長度**設置為 1024 bit，不需要設置私鑰密碼，點擊**生成密鑰對**。



2. 將生成的 **RSA 加密私鑰** 複製粘貼至 txt 記事本，並保存至 OpenD 所在電腦的指定路徑。

3. 在 OpenD 所在的電腦中，指定 **RSA 加密私鑰** 的路徑。
   - 方式一：在 可視化 **OpenD** 啟動界面右側的"加密私鑰"一欄，指定上一步驟中放置 **RSA 加密私鑰** 的路徑。如下圖所示：

- 方式二：在 命令行 OpenD 啟動檔案 OpenD.xml 中，找到參數 `rsa_private_key` ，將其設定為第 2 步中 **RSA 加密私鑰** 的路徑。如下圖所示：

```
<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. 將第 2 步中 txt 檔案另存至策略程式（moomoo API）所在電腦的指定路徑， 並在策略程式中將此路徑 設置為私鑰路徑。

5. 在策略程式（moomoo API）中啟用協議加密。 啟用協議加密的方式有兩種，其中方式二的優先級更高。

   - 方式一：對單條的連接加密（通用）。在對 行情對象 或 交易對象 創建連接時，通過 **是否啟用加密** 參數設置加密。
   - 方式二：對所有的連接加密（僅 Python）。通過 `enable_proto_encrypt` 接口設置加密，詳見 這裏。

> **提示**
>
> - 在 OpenD 或策略程式（moomoo API）中指定 **RSA 加密私鑰** 路徑時，需指定至 txt 檔案本身。
> - RSA 加密公鑰無需保存，可通過私鑰計算得到。

Q6：為什麼我獲取的 DataFrame 數據，只能展示一部分？

A：打印 pandas.DataFrame 數據的時候，如果行列數過多，pandas 預設會將數據摺疊，導致看起來顯示不全。
因此，並不是接口返回數據真的不全。您只需要在 Python 程式前面加上如下代碼即可解決。

```
import pandas as pd
pd.options.display.max_rows=5000
pd.options.display.max_columns=5000
pd.options.display.width=1000
```

Q7：Mac 機器使用 C++ 語言的 API，遇到 “無法打開 libFTAPIChannel.dylib” 的問題

A：在對應程式庫目錄中執行以下命令即可解決: `$ xattr -r -d com.apple.quarantine libAPIChannel.dylib` 。

Q8：Python 用戶，為什麼在 OpenD 設定檔中設置了日誌級別為 no 後，log 資料夾下仍然持續產生超大容量的日誌檔案？

A：OpenD 設定檔中的日誌級別參數，只用來控制 OpenD 產生的日誌。而 Python API 預設也會產生日誌，如果您不希望希望 Python API 產生日誌，可以在 Python 程式加上如下語句：

```
logger.file_level = logging.FATAL  # 用於關閉 Python API 日誌
logger.console_level = logging.FATAL  # 用於關閉 Python 運行時的控制台日誌
```

Q9：對於 5.4 及以上的版本，Java API 的程式庫名稱和設定方式的變更

A: * 如果您是 Java API 5.3 及以下版本的用戶，在更新版本時，請注意以下變更：

設定流程的變更：

1. 通過 moomoo 官網 下載 moomoo API。
2. 解壓下載好的 mmAPI 檔案，`/MMAPI4J` 是 Java API 的目錄，將目錄結構中的 `/lib/moomoo-api-.x.y.z.jar` 添加到您的專案設置中。創建 moomoo-api 專案請參考這裏。

目錄結構的變更：

1. moomoo API 的 Java 版本，程式庫名稱由之前的 mmapi4j.jar 變更為 `moomoo-api-x.y.z.jar`，其中 “x.y.z” 表示版本編號。
2. 第三方程式庫的引用中，去掉了 /lib/jna.jar 和 /lib/jna-platform.jar 相依性項，增加了 `/lib/bcprov-jdk15on-1.68.jar` 和 `/lib/bcpkix-jdk15on-1.68.jar` 相依性項。

```
```
+---mmapi4j   moomoo-api 原始碼，如果所用 JDK 版本不兼容可以用這裏的專案
+---lib   存放公共庫文件
| moomoo-api-x.y.z.jar   moomoo API 的 Java 版本
| bcprov-jdk15on-1.68.jar   第三方程式庫，用於加解密
```
```

```
|    bcpkix-jdk15on-1.68.jar      第三方程式庫，用於加解密
|    protobuf-java-3.5.1.jar      第三方程式庫，用於解析 protobuf 數據
+---sample                        示例專案
+---resources                     maven 專案預設生成的目錄
```

- 如果您第一次接觸 moomoo API，我們提供了更便捷的通過 maven 倉庫設定 Java API 的方式。設定流程請參考 這裏。

Q10：Python 用戶，使用 pyinstaller 打包程式時報錯：找不到 Common_pb2 模組

A：你可以嘗試通過以下步驟解決此問題：

1. 假設你需要對 main.py 進行打包。使用命令行語句，運行代碼：pyinstaller main.py，不要加參數 "- F"（path 為 main.py 的所在路徑）

```
pyinstaller path\main.py
```

打包成功後，main.py 所在目錄下的 /dist 中，會生成 /main 資料夾，main.exe 就在這個資料夾中。

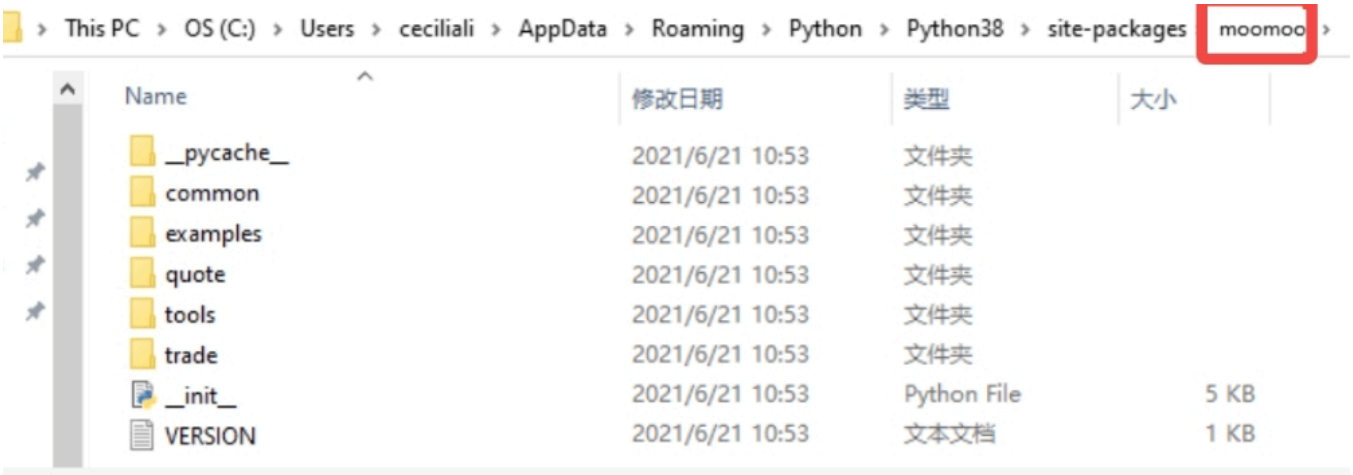
2. 運行以下代碼，找到 moomoo-api 的安裝目錄。

```
import moomoo
print(moomoo.__file__)
```

```
```

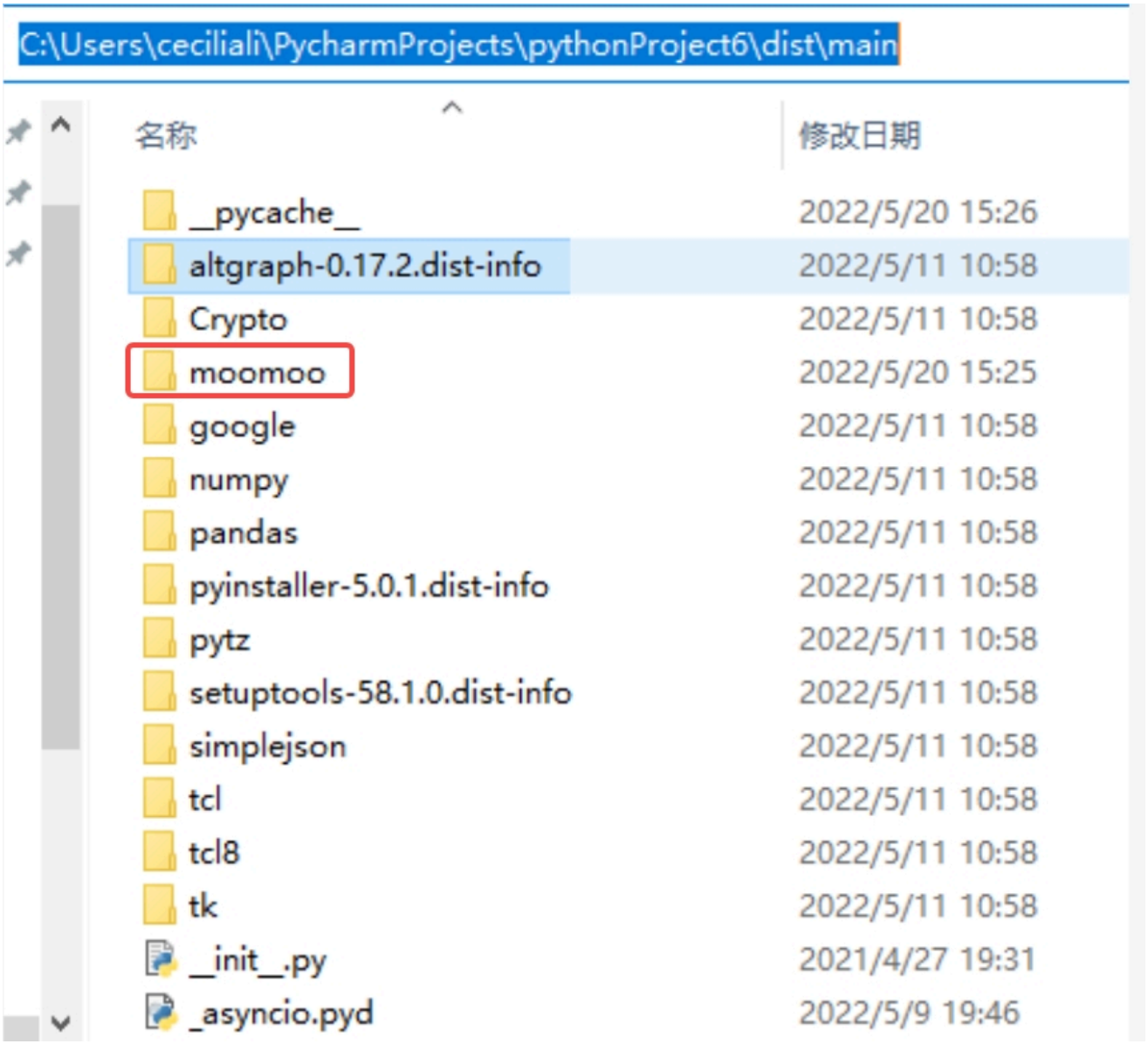
運行結果:

```
C:\Users\ceciliali\Anaconda3\lib\site-packages\moomoo\__init__.py
```



3. 打開上圖資料夾中的 /common/pb，將所有檔案全部複製到 /main 中。

4. 在 /main 中創建資料夾，命名為 moomoo，將上圖資料夾中的 `VERSION.txt` 檔案複製到 /main/moomoo 中。



5. 再次嘗試運行 main.exe

Q11：接口調用結果正常，但其返回表現不符合預期？

A:

- 接口調用結果正常，表示富途已經成功收到並響應了您的請求，但接口返回表現可能與您的預期不符。

  例如：若您在非交易時段調用 訂閱 接口，雖然您的請求可以被成功響應，並且接口調用結果正常，但在非交易時段下，交易所無行情數據變動，所以您將暫時無法收到行情數據推送，直至市場重新回到交易時段。

- 接口調用結果可以通過返回欄位（定義參見：接口調用結果）查看，返回欄位為 0 代表接口調用正常，非 0 代表接口調用失敗。

對於 Python 用戶，下面兩種寫法等價：

```
if ret_code == RET_OK:
```

```
if ret_code == 0:
```

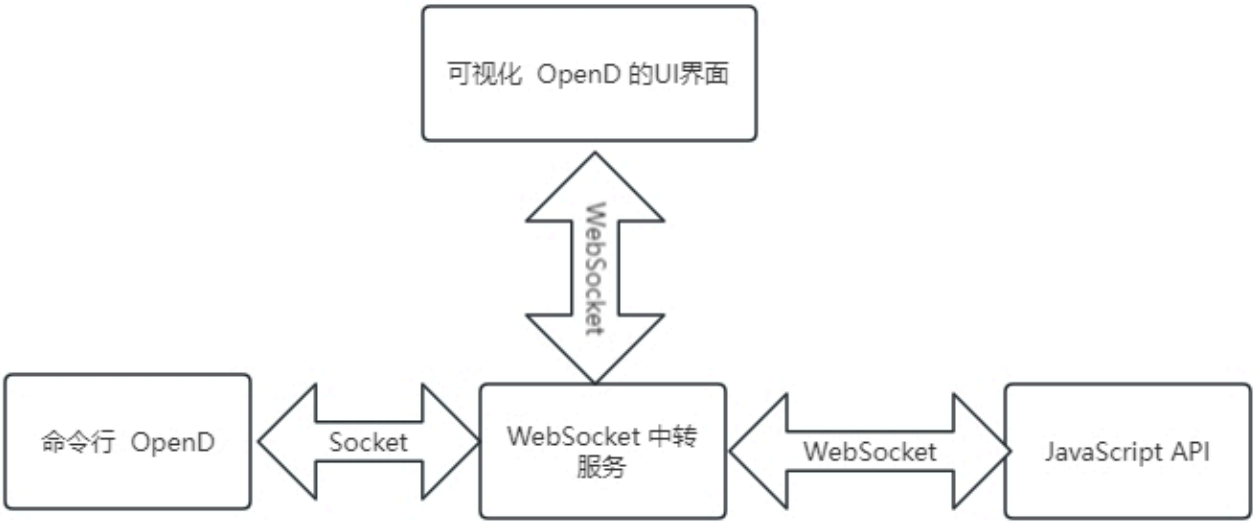
Q12：WebSocket相關

A：

概述

OpenAPI 中，WebSocket 主要用於以下兩方面：

- 可視化 OpenD 中，UI 界面跟底層的命令行 OpenD 的通信使用 WebSocket 方式。
- JavaScript API 跟 OpenD 之間的通信使用 WebSocket 方式。



- 當 WebSocket 啟動時，命令行 OpenD 會與 **MMWebSocket 中轉服務** 建立 Socket 連接（TCP），這一連接會用到預設的 **監聽地址** 和 **API 協議監聽連接埠**。
- 同時，JavaScript API 會與 **MMWebSocket 中轉服務** 建立 WebSocket 連接（HTTP），這一連接會用到 **WebSocket 監聽地址** 和 **WebSocket 連接埠**。

使用

為保證帳戶安全，當 WebSocket 監聽來自非本地請求時，我們強烈建議您啟用 SSL 並設定 **WebSocket 鑑權密鑰**。

SSL 通過在設定 **WebSocket 證書** 以及 **WebSocket 私鑰** 來啟用。
命令行 OpenD 可通過設定 OpenD.xml 或設定命令行參數來設置檔案路徑。可視化 OpenD 點擊【更多選項】下拉菜單，可以看到設置項。

> **提示**
>
> 如果證書是自籤的，則需要在調用 JavaScript 接口所在機器上安裝該證書，或者設置不驗證證書。

生成自簽證書

自簽證書生成詳細資料不便在此文檔展開，請自行查閱。
在此提供較簡單可用的生成步驟：

1. 安裝 openssl。

2. 修改 openssl.cnf，在 alt_names 節點下加上 OpenD 所在機器 IP 地址或域名。
   例如：IP.2 = xxx.xxx.xxx.xxx, DNS.2 = www.xxx.com
3. 生成私鑰以及證書（PEM）。

證書生成參數參考如下：

```
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
```

> **提示**
>
> - openssl.cnf 需要放到系統路徑下，或在生成參數中指定絕對路徑。
> - 注意生成私鑰需要指定不設置密碼（-nodes）。

附上本地自簽證書以及生成證書的設定檔供測試：

- openssl.cnf
- moomoo.cer
- moomoo.key

Q13：OpenAPI 的行情和交易服務分別部署在哪裏？

A：

- 行情：

平台賬號	行情伺服器所在地
牛牛號	騰訊雲廣州和香港
moomoo 號	騰訊雲美國弗吉尼亞和新加坡

- 交易：

所屬券商	交易伺服器所在地
富途證券(香港)	香港
moomoo證券(美國)	騰訊雲美國弗吉尼亞
moomoo證券(新加坡)	騰訊雲新加坡

所屬券商	交易伺服器所在地
moomoo證券(澳大利亞)	騰訊雲新加坡
moomoo證券(馬來西亞)	阿里雲馬來西亞
moomoo證券(加拿大)	AWS加拿大
moomoo證券(日本)	騰訊雲日本