**Python Trading API Development Specification**

Table of contents

[I.](#_Toc185002813)  [Overview](#_Toc185002813)  [4](#_Toc185002813)

[1.1.](#_Toc185002814)  [Background](#_Toc185002814)  [4](#_Toc185002814)

[1.2.](#_Toc185002815)  [API interface overview and architecture](#_Toc185002815)  [5](#_Toc185002815)

[1.3.](#_Toc185002816)  [Global error code](#_Toc185002816)  [6](#_Toc185002816)

[2.](#_Toc185002817)  [Stock Trading API Interface Reference](#_Toc185002817)  [8](#_Toc185002817)

[1. Interface working process](#_Toc185002818)  [9](#_Toc185002818)

[2. Interface mode](#_Toc185002819)  [9](#_Toc185002819)

[3. Function interface description](#_Toc185002820)  [9](#_Toc185002820)

[1)](#_Toc185002821)  [createStockApi method](#_Toc185002821)  [9](#_Toc185002821)

[2)](#_Toc185002822)  [onFrontConnected method](#_Toc185002822)  [10](#_Toc185002822)

[3)](#_Toc185002823)  [onFrontDisconnected method](#_Toc185002823)  [10](#_Toc185002823)

[4)](#_Toc185002824)  [onHeartBeatWarning method](#_Toc185002824)  [11](#_Toc185002824)

[5)](#_Toc185002825)  [Login reqUserLogin / onRspUserLogin / onRspNeedVerify / onRspAccount](#_Toc185002825)  [12](#_Toc185002825)

[6)](#_Toc185002826)  [Logout reqUserLogout/onRspUserLogout method](#_Toc185002826)  [14](#_Toc185002826)

[7)](#_Toc185002827)  [Change password reqPasswordUpdate/onRspPasswordUpdate](#_Toc185002827)  [15](#_Toc185002827)

[8)](#_Toc185002828)  [Order reqOrderInsert/onRspOrderInsert](#_Toc185002828)  [16](#_Toc185002828)

[9)](#_Toc185002829)  [Order push onRtnOrder](#_Toc185002829)  [20](#_Toc185002829)

[10)](#_Toc185002830)  [Transaction push onRtnTrade](#_Toc185002830)  [22](#_Toc185002830)

[11)](#_Toc185002831)  [Funding information push onRtnCapital](#_Toc185002831)  [24](#_Toc185002831)

[12)](#_Toc185002832)  [Order modification request / return reqOrderModify/onRspOrderModify method](#_Toc185002832)  [27](#_Toc185002832)

[13)](#_Toc185002833)  [Cancel order request / return reqOrderCancel/onRspOrderCancel](#_Toc185002833)  [29](#_Toc185002833)

[14)](#_Toc185002834)  [Order query / return reqQryOrder/onRspQryOrder](#_Toc185002834)  [31](#_Toc185002834)

[15)](#_Toc185002835)  [Transaction query request / return reqQryTrade/onRspQryTrade](#_Toc185002835)  [33](#_Toc185002835)

[16)](#_Toc185002836)  [Position details query request / return reqQryPosition/onRspQryPosition](#_Toc185002836)  [34](#_Toc185002836)

[17)](#_Toc185002837)  [Funds query request / return reqQryCapital/onRspQryCapital](#_Toc185002837)  [36](#_Toc185002837)

[18)](#_Toc185002838)  [Broker query request / return reqQryBroker/onRspQryBroker](#_Toc185002838)  [39](#_Toc185002838)

[19)](#_Toc185002839)  [Version number query request / return reqQryVersion/onRspQryQryVersion](#_Toc185002839)  [40](#_Toc185002839)

[20)](#_Toc185002840)  [Currency query request / return reqQryCurrency/ onRspQryCurrency](#_Toc185002840)  [41](#_Toc185002840)

[21)](#_Toc185002841)  [Trading time query request / return reqQryTradeTime/ onRspQryTradeTime](#_Toc185002841)  [42](#_Toc185002841)

[22)](#_Toc185002842)  [Exchange query request / return reqQryExchange /onRspQryExchange](#_Toc185002842)  [43](#_Toc185002842)

[23)](#_Toc185002843)  [Contract query request / return reqQryInstrument/onRspQryInstrument](#_Toc185002843)  [44](#_Toc185002843)

[24)](#_Toc185002844)  [Jump point query request / return reqQryTick/ onRspQryTick](#_Toc185002844)  [47](#_Toc185002844)

[25)](#_Toc185002845)  [Order type query request / return reqQryOrderType/ onRspQryOrderType](#_Toc185002845)  [48](#_Toc185002845)

[26)](#_Toc185002846)  [Set the two-factor authentication password / return to reqSetVerifyQA/ onRspSetVerifyQA](#_Toc185002846)  [49](#_Toc185002846)

[27) Bind the](#_Toc185002847)  [MAC address required for dual authentication / return reqSafeVerify / onRspSafeVerify](#_Toc185002847)  [50](#_Toc185002847)

[3.](#_Toc185002848)  [Futures Trading API Interface Reference](#_Toc185002848)  [51](#_Toc185002848)

[1. Interface working process](#_Toc185002849)  [53](#_Toc185002849)

[2. Interface mode](#_Toc185002850)  [53](#_Toc185002850)

[3. Function interface description](#_Toc185002851)  [54](#_Toc185002851)

[1)](#_Toc185002852)  [CreateFutureApi method](#_Toc185002852)  [54](#_Toc185002852)

[2)](#_Toc185002853)  [OnFrontConnected method](#_Toc185002853)  [54](#_Toc185002853)

[3)](#_Toc185002854)  [OnFrontDisconnected method](#_Toc185002854)  [54](#_Toc185002854)

[4)](#_Toc185002855)  [OnHeartBeatWarning method](#_Toc185002855)  [55](#_Toc185002855)

[5)](#_Toc185002856)  [LoginReqUserLogin / OnRspUserLogin /OnRspNeedVerify/OnRspAccount](#_Toc185002856)  [56](#_Toc185002856)

[6)](#_Toc185002857)  [Logout ReqUserLogout/OnRspUserLogout method](#_Toc185002857)  [58](#_Toc185002857)

[7)](#_Toc185002858)  [Change password ReqPasswordUpdate/OnRspPasswordUpdate](#_Toc185002858)  [59](#_Toc185002858)

[8)](#_Toc185002859)  [Order ReqOrderInsert/OnRspOrderInsert](#_Toc185002859)  [60](#_Toc185002859)

[9)](#_Toc185002860)  [Order push OnRtnOrder](#_Toc185002860)  [66](#_Toc185002860)

[10)](#_Toc185002861)  [Transaction push OnRtnTrade](#_Toc185002861)  [68](#_Toc185002861)

[11)](#_Toc185002862)  [Push position information OnRtnPosition](#_Toc185002862)  [70](#_Toc185002862)

[12)](#_Toc185002863)  [Fund information push](#_Toc185002863) ~~[OnRtnAccount](#_Toc185002863)~~ [OnRtnCapital](#_Toc185002863)  [71](#_Toc185002863)

[13)](#_Toc185002864)  [Order modification request / return ReqOrderModify/OnRspOrderModify method](#_Toc185002864)  [74](#_Toc185002864)

[14)](#_Toc185002865)  [Cancel order request / return ReqOrderCancel/OnRspOrderCancel](#_Toc185002865)  [76](#_Toc185002865)

[15)](#_Toc185002866)  [Order query / return ReqQryOrder/OnRspQryOrder](#_Toc185002866)  [79](#_Toc185002866)

[16)](#_Toc185002867)  [Transaction query request / return ReqQryTrade/OnRspQryTrade](#_Toc185002867)  [82](#_Toc185002867)

[17)](#_Toc185002868)  [Position details query request / return ReqQryPosition/OnRspQryPosition](#_Toc185002868)  [84](#_Toc185002868)

[18)](#_Toc185002869)  [Funds query request / return ReqQryCapital/OnRspQryCapital](#_Toc185002869)  [86](#_Toc185002869)

[19)](#_Toc185002870)  [Version number query request / return ReqQryVersion/OnRspQryVersion](#_Toc185002870)  [88](#_Toc185002870)

[20)](#_Toc185002871)  [Currency query request / return ReqQryCurrency/ OnRspQryCurrency](#_Toc185002871)  [89](#_Toc185002871)

[21)](#_Toc185002872)  [Trading time query request / return ReqQryTradeTime/ OnRspQryTradeTime](#_Toc185002872)  [90](#_Toc185002872)

[22)](#_Toc185002873)  [Exchange query request / return ReqQryExchange /OnRspQryExchange](#_Toc185002873)  [91](#_Toc185002873)

[23)](#_Toc185002874)  [Contract query request / return ReqQryInstrument/OnRspQryInstrument](#_Toc185002874)  [92](#_Toc185002874)

[24)](#_Toc185002875)  [Strategy query request / return ReqQryStrategy/ OnRspQryStrategy](#_Toc185002875)  [95](#_Toc185002875)

[25)](#_Toc185002876)  [Strategy detail query request / return ReqQryStrategyDetail/ OnRspQryStrategyDetail](#_Toc185002876)  [98](#_Toc185002876)

[25)](#_Toc185002877)  [Set two-factor authentication password / Return to ReqSetVerifyQA/ OnRspSetVerifyQA](#_Toc185002877)  [100](#_Toc185002877)

[26) Bind the](#_Toc185002878)  [MAC address required for dual authentication / return ReqSafeVerify / OnRspSafeVerify](#_Toc185002878)  [101](#_Toc185002878)

[27)](#_Toc185002879)  [Query the commodity list request / return ReqQryCommodity / OnRspQryCommodity](#_Toc185002879)  [102](#_Toc185002879)

[28)](#_Toc185002880)  [Position summary query request / return ReqQryTotalPosition / OnRspQryTotalPosition](#_Toc185002880)  [104](#_Toc185002880)

[29)](#_Toc185002881)  [Query request for the opening time of the exchange in winter and summer / return ReqQryExchangeTime / OnRspQryExchangeTime](#_Toc185002881)  [106](#_Toc185002881)

[30)](#_Toc185002882)  [Query request for opening and closing time of products / return ReqQryExchangeTime / OnRspQryCommodityTime](#_Toc185002882)  [107](#_Toc185002882)

[IV.](#_Toc185002883)  [FAQ - Futures Trading Related](#_Toc185002883)  [108](#_Toc185002883)

[V.](#_Toc185002884)  [FAQ - Stock Trading Related](#_Toc185002884)  [110](#_Toc185002884)

# Overview

## background

This API can be used to trade futures and stocks outside of China, and to obtain market data for the corresponding markets.

Currently supported exchanges/markets include (but not limited to ) the following:

Futures

|  |  |  |
| --- | --- | --- |
| No | Exchanges | illustrate |
| 1 | CME | , COMEX, NYMEX , CBOT under CME group  Official Website:  https://www.cmegroup.com/ |
| 2 | HKEX | Hong Kong Exchanges and Clearing Limited  Official Website:  http://www.hkex.com.hk |
| 3 | SGX | Singapore Exchange  Official Website:  http://www.sgx.com |
| 4 | I CE | Intercontinental Exchange  Official Website:  https://www.theice.com/index |
| 5 | LME | London Metal Exchange  Official Website:  https://www.lme.com/ |

Stocks

|  |  |  |
| --- | --- | --- |
| No | market | illustrate |
| 1 | NASD | Stocks in the US market.  It is important to note that after the 2008 financial crisis, the United States passed a bill allowing listed companies to be traded on all U.S. exchanges. For example, stocks listed on the NYSE can be traded on the NASDQ exchange and vice versa. |
| 2 | HKEX | Stocks on the Hong Kong Stock Exchange |
| 3 | KRX | South Korean stocks |

## API interface overview and architecture

This API is a Python-based class library that implements the functions required for trading by using and extending the interfaces provided by the class library. In view of the widespread use of CTP interface in the domestic futures industry, this API interface imitates the CTP interface so that users with experience in using CTP interface can quickly get started with development.

At the same time, due to some differences between stocks and futures, the trading interface of stocks and futures is separated, but the API library is the same library, please pay attention to it. The following two subsections explain futures and stocks respectively.

**How to use the futures interface**

Future Trade Client

(call DAApi.dll)

Future Trade front

Market Client

(call DAApi.dll)

Future market front

**How to use the stock interface**

Stock Trade Client

(call DAApi.dll)

Stock Trade front

Stock Market Client

(call DAApi.dll)

Stock Market front

For the following error codes, a value of 0 indicates success, regardless of where in this API.

// Error code

ErrorID [int]

## Global error code

1. Common errors in business functions

// Error information structure in futures interface

dict error

{

ErrorID [int]: Error code

ErrorMsg [str]: Error description

};

// Error information structure in the stock interface

dict error

{

ErrorID [int]: Error code

ErrorMsg [str]: Error description

};

|  |  |  |
| --- | --- | --- |
| Error Code  ( ErrorID ) | Error description | illustrate |
| 0 | success | Various functions return results successfully |
| 90001 | Failed to parse the protocol header | Direct system internal error |
| 90002 | Failed to decompress data | Direct system internal error |
| 90003 | The uncompressed data body is empty | Directly to the internal system error; usually refers to the query result is empty, the server has a response, but there is no business data |
| 90004 | The compressed data body is empty | Directly to the internal system error; usually refers to the query result is empty, the server has a response, but there is no business data |
| 90005 | Decryption failed | Direct system internal error |
| 90006 | The server returns an empty array. | Directly to the internal system error; usually refers to the query result is empty, the server has a response, but there is no business data |
|  |  |  |
|  |  |  |

2. General error of disconnection

onFrontDisconnected ( nReason : int )

|  |  |  |
| --- | --- | --- |
| Error Code  ( nReason ) | Error description | illustrate |
| 4 001 | Setting blocking failed | Socket underlying error |
| 4 002 | Setting non-blocking failure | Socket underlying error |
| 4 003 | Failed to receive data | Socket underlying error |
| 4 004 | Received 0 length data, disconnected | Socket underlying error |
| 4 005 | Failed to send data | Socket underlying error |
| 4 006 | Sending 0 length data, disconnected | Socket underlying error |
| 4 007 | Failure of selection | Socket underlying error |
| 4 008 | Server Not Responding | Socket underlying error |
|  |  |  |
|  |  |  |

# Stock Trading API Reference

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Business Type | business | Request interface | illustrate | Response interface |
| connect | connect | i nit | Connection Return | onFrontConnected |
| disconnect |  |  | Disconnection report | onFrontDisconnected |
| Heartbeat |  |  | Heartbeat warning | oHeartBeatWarning |
| Log in | Log in | reqUserLogin | Login Report | onRspUserLogin / onRspNeedVerify |
| Sign out | reqUserLogout | Logout Report | onRspUserLogout |
| Change Password | reqPasswordUpdate | Change password report | onRspPasswordUpdate |
| Two-factor authentication step1 | reqSetVerifyQA | Set up a two-factor authentication password | onRspSetVerifyQA |
| Two-factor authentication step2 | reqSafeVerify | Bind MAC address | onRspSafeVerify |
| trade | declaration | reqOrderInsert | Order return | onRspOrderInsert |
| Change order | reqOrder Modify | Change order return | onRspOrder Modify |
| Cancel Order | reqOrderCancel | Cancellation return | onRspOrderCancel |
| Push |  |  | Order status push | onRtnOrder |
|  |  | Transaction information push | onRtnTrade |
|  |  | *Position information push* | *onRtnPosition* |
|  |  | Funding information push | onRtnAccount |
| Query | Order Query | reqQryOrder | Order query return | onRspQryOrder |
| Transaction Inquiry | reqQryTrade | Transaction query report | onRspQryTrade |
| Position Details Query | reqQryPosition | Position Details Query Return | o nRspQryPosition |
| Funding Inquiry | reqQryCapital | Funds Inquiry Return | onRspQryCapital |
| Broker Inquiry  (Hong Kong stocks only ) | reqQryBroker | Broker Inquiry Return | onRspQryBroker |
| Version number query | reqQryVersion | Version number query report | onRspQryQryVersion |
| Currency Query | reqQryCurrency | Currency query return | onRspQryCurrency |
| Trading time query | reqQryTradeTime | Transaction time query report | onRspQryTradeTime |
| Exchange Query | reqQryExchange | Exchange Query Return | onRspQryExchange |
| Contract Query | reqQryInstrument | Contract query report | onRspQryInstrument |
| Jump point query | reqQryTick | Jump point query report | onRspQryTick |
| *Order Type Query* | *ReqQryOrderType* | *Order Type Query Return* | *onRspQryOrderType* |

Note: The above italic interfaces are not implemented in the current API version.

## Interface working process



## Interface Mode

StockApi class is provided in the stock\_demo.py file . By inheriting the StockApi class, you can issue operation requests and overload callback functions to handle the responses of background services.

## Function interface description

### createStockApi​ method

Create instance methods.

**Function prototype** :

createStockApi ( bRecordLog : bool , lpszLogFileName : str = "Stock.log" , lpszLogFileDir : str = "" );

bRecordLog : true means recording logs, false means not recording logs.

lpszLogFileName : Log file.

lpszLogFileDir : Log file folder path.

### onFrontConnected method

This method is called when the client establishes a communication connection with the frontend (before logging in).

**Function prototype:**

o nFrontConnected ( )

This method is called after initialization is completed, in which the user login task can be completed.

### onFrontDisconnected method

This method is called when the client loses the connection with the frontend communication. When this happens

After that, the API will automatically reconnect and the client does not need to do anything. Automatic reconnection address, which may be the original registration address

It may also be other available communication addresses supported by the system, which are automatically selected by the program.

**Function prototype:**

onFrontDisconnected ( nReason : int )

**parameter:**

nReason : Reason for disconnection

### onHeartBeatWarning method

Heartbeat timeout warning. This method is called when no message is received for a long time.

**Function prototype:**

onHeartBeatWarning ( nTimeLapse : int )

**parameter:**

nTimeLapse : The time since the last message was received

### Login r eqUserLogin / onRspUserLogin / onRspNeedVerify / onRspAccount

from API v1.5 . For the specific interaction process, please refer to **" International Futures and Stocks API\_Two-factor Authentication Interaction Instructions.xlsx "**

Function prototype:

|  |
| --- |
| reqUserLogin ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // User ID  UserId [str]  // User password  UserPwd [str]  // User type  UserType [str]  // Software Name  SoftwareName [str]  // Software version number  SoftwareVersion [str]  // Authorization code  AuthorCode [str]  // Login machine network card MAC address ( since API v1.5, used for dual authentication )  MacAddress [str]  // error message  ErrorDescription [str] |

1. Function prototype:

|  |
| --- |
| onRspUserLogin ( error: dict , reqid : int, last: bool ) |

Parameters: error

Error code:

|  |  |
| --- | --- |
| **Login error code** | **Error content** |
| 10001 | Incorrect username |
| 10002 | Wrong login password |
| 10003 | The number of incorrect passwords has exceeded the limit and the user has been frozen. Please contact customer service to unfreeze the account! |
| 10004 | The user has been frozen, please contact customer service to unfreeze! |
| 1 0013 | authorization code information is incorrect. Please confirm the values of UserType , SoftwareName , SoftwareVersion , and AuthorCode. |

1. Function prototype:

When the server determines that two-factor authentication is required, it will instruct the API to call back this function.

|  |
| --- |
| o nRspNeedVerify ( bFirstLogin : bool , bHasSetQA : bool ) ; |

Parameter: bFirstLogin

|  |
| --- |
| identified by the MacAddress field in the login message ) logs in for the first time before a transaction, o nRspNeedVerify function is called back, and bFirstLogin The value of is true, and the user needs to call the reqSetVerifyQA function in the callback function to set the dual authentication information.  MacAddress field in the login message ) logs in for the first time before a transaction, the onRspNeedVerify () function will not be called back. |

1. Function prototype:

After onRspUserLogin indicates a successful login callback, this function is called back and the fund account corresponding to each currency is returned in the data dictionary.

|  |
| --- |
| o nRspAccount ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Account currency  CurrencyNo [str]  // Funding account  AccountNo [str] |

### Logout reqUserLogout / onRspUserLogout method

Function prototype:

|  |
| --- |
| reqUserLogout ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // User ID  UserId [str]  // Funding account  AccountNo [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspUserLogout ( error: dict , reqid : int, last: bool ) |

Parameters: error

### Change password reqPasswordUpdate / onRspPasswordUpdate

Function prototype:

|  |
| --- |
| reqPasswordUpdate ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // User code  UserId [str]  // User's original password  OldPassword [str]  // User's new password  NewPassword [str]  // error message  ErrorDescription [ str] |

Function prototype:

|  |
| --- |
| onRspPasswordUpdate ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User code  UserId [str]  // User's original password  OldPassword [str]  // User's new password  NewPassword [str]  // error message  ErrorDescription [ str] |

Parameters: error

### Order reqOrderInsert / onRspOrderInsert​

illustrate:

SystemNo [str] : system number

LocalNo [str]: local number

OrderNo [str]: order number

LocalNo is the order number given by the program using the API, which is equivalent to the orderref in CTP . It is best not to repeat it within the same day.

SystemNo is the order number given by the direct system;

OrderNo is the order number given by the exchange, which is equivalent to OrderSysID in CTP

Function prototype:

|  |
| --- |
| reqOrderInsert ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // User ID  UserId [str]  // Funding account (a user can have funds in multiple currencies, so there will be multiple funding accounts, which need to be specified when placing an order )  // Please refer to the description of the callback function OnRspAccount after successful login  AccountNo [str]  // Local number  LocalNo [str]  // Exchange code  ExchangeCode [str]  // Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  // Used when short selling, 1=open position ; 2=close position, and opposite to the buy and sell direction of the order. When not short selling, 0 or no value is set  OpenCloseFlag [str]  // Number of orders  OrderQty [str]  // Order price  OrderPrice [str]  // Order type: (from v1.12)  // US stocks: 1 = limit order  // Hong Kong stocks: 5 = Auction limit order; 6 = Auction order; 7 = Enhanced limit order; 8 = Special limit order  OrderType [str]  // Trigger price ( not used in stocks )  TriggerPrice [str]  // Effective date (1=valid today)  TIF [ str]  // Strategy ID ( not used in stocks )  StrategyId [str]  // Show the entrustment amount, must be less than the entrustment amount ( not used in stocks )  MaxShow [str]  // Minimum trading volume ( not used in stocks )  MinQty [str]  // error message  ErrorDescription [str] |

illustrate:

After the API user places an order, the onRspOrderInsert function will be called back twice:

1. The first callback returns the system number SystemNo assigned by the direct system to this order , indicating that the direct system has received and confirmed this order.
2. OrderNo of the exchange ( if everything is normal ). Usually when insufficient funds occur, there will be no order number because the order instruction has not been sent to the exchange.

Function prototype:

|  |
| --- |
| onRspOrderInsert ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Funding account  AccountNo [str]  // System number  SystemNo [str]  // Local number  LocalNo [str]  //Order number  OrderNo [str]  // Exchange code  ExchangeCode [str]  //Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  //Number of orders  OrderQty [str]  //Order price  OrderPrice [str]  // Order type: ( from v1.12)  // US stocks: 1 = limit order  // Hong Kong stocks: 5 = auction limit order; 6=Auction order; 7 = Enhanced limit price; 8 = Special limit price  OrderType [str]  //Order date  OrderDate [str]  // Order time  OrderTime [str]  // Error code (reserved, not used)  ErrorCode [str]  // Order status (2: queued; 3: partially executed; 4: fully executed; 5: remaining orders canceled; 6: canceled; 7: order failed; 8: to be sent; 9: to be changed; A: to be canceled)  OrderState [str]  // User ID who canceled the order  CancelUserId [str]  // Trigger price  TriggerPrice [str]  // Valid date (1=valid today)  TIF [ str]  // Used when short selling, 1=open position ; 2=close position, and opposite to the buy and sell direction of the order. When not short selling, 0 or no value is set  OpenCloseFlag [str]  // Strategy ID ( not used in stocks )  StrategyId [str]  // Display the order amount, which must be less than the order amount (not used in stocks)  MaxShow [str]  // Minimum trading volume (not used in stocks)  MinQty [str]  // Exchange return time  ExchangeTime [str] |

Parameters: error

|  |
| --- |
| // Error code (used to determine whether the order is successful or failed)  ErrorID [int]  // Error description  ErrorMsg [str] |

Error Code

|  |  |
| --- | --- |
| **System number error code** | **Error content** |
| 20005 | System number generation failed |

Order error code

|  |  |
| --- | --- |
| **Order error code** | **Error content** |
| 20000 | Order failed |
| 20001 | Insufficient funds |
| 20002 | Trading server not connected |
| 20003 | You have been banned from trading, please contact customer service to unblock |
| 20004 | Order rejected |
| 20005 | System number generation failed |
| 20006 | Your fund account does not have the currency required for this contract transaction |
| 20007 | The contract has expired and cannot be traded. |
| 20008 | The trading market is not open and cannot be traded |
| 20009 | The trading market is closed and cannot be traded. |
| 20010 | The order price exceeds the limit |
| 20011 | The order quantity is too large |
| 20012 | You are prohibited from trading this contract, please contact customer service |
| 20013 | Your account risk rate is too high and trading has been prohibited |
| 20015 | Your account transaction data is abnormal, and the transaction is suspended |
| 20019 | The order quantity exceeds the customer's position limit |
| 20020 | The order quantity exceeds the customer's purchase quantity |
| 20021 | The order quantity exceeds the customer's available order quantity |
| 20022 | The order quantity exceeds the company's position limit |
| 20023 | The order quantity exceeds the company's purchase quantity |
| 20024 | The order quantity exceeds the company's available order quantity |
| 20025 | The dealer's order quantity exceeds the maximum order quantity |
| 20026 | When the contract is about to be delivered, you can only close the position, not open a position |
| 20027 | The last trading day of this contract has passed, and electronic trading is prohibited |
| 20028 | The product is not open, trading is prohibited |
| 20029 | The single order quantity exceeds the maximum number of orders that can be placed for this product |
| 20032 | Insufficient positions, position closing failed |
| 20033 | You cannot trade this product, please contact customer service to open |

### Order push onRtnOrder

Function prototype:

|  |
| --- |
| onRtnOrder ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Exchange code      ExchangeNo [str]    //Contract code      TreatyCode [str]      //Order number      OrderNo [str]      // Order quantity      OrderNumber [int]      // Quantity of transactions completed      FilledNumber [int]    // Average transaction price      FilledAdvPrice [float]    // Buy quantity      BuyHoldNumber [int]      // Average price for opening long positions      BuyHoldOpenPrice [float]  // Holding average price      BuyHoldPrice [float]    // Sell quantity      SaleHoldNumber [int]    // Average opening price of short positions      SaleHoldOpenPrice [float]  // Average selling price      SaleHoldPrice [float]    // Has the order been cancelled? (0: No; 1: The order has been cancelled)      IsCanceled [str]      // Total transaction fee      FilledTotalFee [float]      // Sequence number      Status[int]      // Funding account      AccountNo [str]      // Position type (0: yesterday's position; 1: today's position)      HoldType [str]      // Buy margin      HoldMarginBuy [float]      // Sell margin  HoldMarginSale [float]  // Latest price  CurrPrice [float]  // Floating profit and loss  FloatProfit [float] |

Parameters: error

### Transaction push onRtnTrade

Function prototype:

|  |
| --- |
| onRtnTrade ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Funding account  AccountNo [str]  // Transaction number (including the 7-digit order number, a total of 11 digits)  FilledNo [str]  //Order number  OrderNo [str]  // System number  SystemNo [str]  // Local number  LocalNo [str]  // Exchange code  ExchangeCode [str]  //Contract code  TreatyCode [str]  // Buy or sell: 1=buy 2=sell  BuySale [str]  // Number of transactions  FilledNumber [str]  // Transaction price  FilledPrice [str]  // Transaction date ( yyyy -MM-dd)  FilledDate [str]  // Transaction time ( hh:mm:ss )  FilledTime [str]  // Transaction fee  Commsion [str]  // Order quantity  HTSCode [str]  //Commission price  ErrorCode [str]  // Contract delivery date ( yyyyMMdd )  DeliveryDate [str]  // Transaction type (N: ordinary order transaction; C, T: simulated transaction of swap)  FilledType [str]  // Order type (1=limit order, 2=market order, 3=stop to limit, 4=stop to market) OrderType [str]  // Valid date (1 = valid today, 2 = valid forever)  ValidDate [str]  // Open or close position: 1=Open 2=Close, 3=Close today, 4= Close yesterday  AddReduce [str]  // error message  ErrorDescription [str] |

Parameters: error

### Funding information push onRtnCapital

Function prototype:

|  |
| --- |
| onRtnCapital ( data: dict , error: dict , reqid : int, last: bool ) |

Parameters: data

|  |
| --- |
| // Customer Number  ClientNo [str]  // Funding account  AccountNo [str]  //Currency  CurrencyNo [str]  // Available now  Available[float]  // Available yesterday  YAvailable [float]  // Available today  CanCashOut [float]  //Today's balance  Money[float]  //Profit and loss of closing position  ExpiredProfit [float]  //Freeze funds  FrozenDeposit [float]  //Handling Fee  Fee[float]  //Security deposit  Deposit[float]  //Maintenance Margin  KeepDeposit [float]  //state  Status [int]  // Deposit  InMoney [float]  // Withdrawal  OutMoney [float]  // Unexpired profit  UnexpiredProfit [float]  // Today's equity  TodayTotal [float]  // Unresolved profit  UnaccountProfit [float]  // Option premium  Royalty [float]  // Exchange code  ExchangeNo [str]  // Contract code  TreatyCode [str]  // Order number  OrderNo [str]  // Order quantity  OrderNumber [int]  // Quantity of transactions completed  FilledNumber [int]  // Average transaction price  FilledAdvPrice [float]  // Buy quantity  BuyHoldNumber [int]  // Average price for opening long positions  BuyHoldOpenPrice [float]  // Holding average price  BuyHoldPrice [float]  // Sell quantity  SaleHoldNumber [int]  // Average price of opening a short position  SaleHoldOpenPrice [float]  // Average selling price  SaleHoldPrice [float]  // Whether the order has been cancelled ( 0 : No; 1 : The order has been cancelled)  IsCanceled [str]  // Total transaction fee  FilledTotalFee [float]  // Trust limit  Credit [float]  // Margin margin limit 20150727 added for Hong Kong stocks  MarginLimit [float]  // Loan value 20150727 added for Hong Kong stocks  BorrowValue [float]  // Mortgage value 20150727 added for Hong Kong stocks  FMortgageMoney [float]  // T1 20160219 added for Hong Kong stocks  T1 [float]  // T2 20160219 added for Hong Kong stocks  T2 [float]  //T3 20160219 added for Hong Kong stocks  T3 [float]  // Tn 20160219 added for Hong Kong stocks  TN [ float]  // Transaction limit  TradeLimit [float]  // Cross-market funding limit  FCrossMax [float]  // Short selling frozen funds  SellFreezenMoney [float]  // Short interest  SellInterest [float]  // Required deposit  SellNeedAddMargin [float] |

### Order modification request /return reqOrderModify / onRspOrderModify method

Function prototype:

|  |
| --- |
| reqOrderModify ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // System number  SystemNo [str]  // User ID  UserId [str]  // Local number  LocalNo [str]  // Funding account  AccountNo [str]  //Order number  OrderNo [str]  // Exchange code  ExchangeCode [str]  //Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  //Number of orders  OrderQty [str]  //Order price  OrderPrice [str]  //Change the number  ModifyQty [str]  // Change order price  ModifyPrice [str]  // US stocks: 1 = limit order  // Hong Kong stocks: 5 = Auction limit order; 6 = Auction order; 7 = Enhanced limit order; 8 = Special limit order  OrderType [str]  // Trigger price (not used in stocks)  TriggerPrice [str]  // Change order trigger price (not used in stocks)  ModifyTriggerPrice [str]  // Valid date (1: valid today)  TIF[ str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspOrderModify ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: error

|  |
| --- |
| // Error code (used to determine whether the order modification is successful or failed )  ErrorID [int]  // Error description  ErrorMsg [str] |

### Cancel order request / return reqOrderCancel / onRspOrderCancel

Function prototype:

|  |
| --- |
| reqOrderCancel ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // User ID      UserId [str]  // Local number      LocalNo [str]  // Funding account      AccountNo [str]  // System number      SystemNo [str]  //Order number      OrderNo [str]  // Exchange code      ExchangeCode [str]  //Contract code      ContractCode [str]  // Buy or sell: 1=buy 2=sell      BidAskFlag [str]  //Number of orders      OrderQty [str]  //Order price      OrderPrice [str]  // Order type      OrderType [str]  // error message      ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspOrderCancel ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID      UserId [str]      // Funding account      AccountNo [str]    // System number      SystemNo [str]  // Local number  LocalNo [str]  // Order number      OrderNo [str]      //Cancellation order number      CancelNo [str]      // Exchange code      ExchangeCode [str]  // Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  // Number of orders  OrderQty [str]  // Order price  OrderPrice [str]  // Number of transactions completed  FilledQty [str]  // Number of cancelled orders  CancelledQty [str]  // Order Type  OrderType [str]  // Cancellation date  CancelledDate [str]  // Cancellation time  CancelledTime [str]  // Error code (reserved, not used)  ErrorCode [str] |
|  |

Parameters: error

|  |
| --- |
| // Error code (used to determine whether the order cancellation is successful or failed )  ErrorID [int]  // Error description  ErrorMsg [str] |

### Order query /return reqQryOrder / onRspQryOrder

The minimum interval between two order queries is 1 second, otherwise the query will fail. ( Since API v1.5)

Function prototype:

|  |
| --- |
| reqQryOrder ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // User ID      UserId [str]      // Funding account      AccountNo [str]      // Whether to simulate the user: 1: Yes; 0 or other: No      IsSimulation [str]    // Get orders after the specified order number      OrderNo [str]    // Get orders after the specified order time (format: yyyy -MM-dd hh:mm:ss )      OrderDateTime [str]      // error message      ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryOrder ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID      UserId [str]      // Funding account      AccountNo [str]      // System number      SystemNo [str]      // Local number      LocalNo [str]      //Order number      OrderNo [str]    // Exchange code      ExchangeCode [str]  // Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  // Number of orders  OrderQty [str]  // Order price  OrderPrice [str]  // Number of transactions completed  FilledQty [str]  // Average transaction price  FilledPrice [str]  // Order Type  OrderType [str]  // Order date  OrderDate [str]  // Order time  OrderTime [str]  // Error code  ErrorCode [str]  // Order status (1: requested; 2: queued; 3: partially executed; 4: fully executed; 5: remaining orders canceled; 6: canceled; 7: order failed; 8: to be sent; 9: to be changed; A: to be canceled)  OrderState [str]  // User ID for canceling order  CancelUserId [str]  // Trigger price ( not used for stocks )  TriggerPrice [str]  // Effective date (1=valid today)  TIF [ str]  // For short selling  OpenCloseFlag [str]  // Strategy ID ( not used by stocks )  StrategyId [str]  // Show the entrustment amount, must be less than the entrustment amount ( not used for stocks )  MaxShow [str]  // Minimum trading volume ( not used for stocks )  MinQty [str]  // Exchange return time  ExchangeTime [str]  // Cancellation time  CancelTime [str] |

### Transaction query request /return reqQryTrade / onRspQryTrade

The minimum interval between two transaction queries is 1 second, otherwise the query will fail. ( Since API v1.5)

Function prototype:

|  |
| --- |
| reqQryTrade ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // user  UserId [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryTrade ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| Please refer to o nRtnTrade data return fields |

### Position details query request /return reqQryPosition / onRspQryPosition

The minimum interval between two position queries is 1 second, otherwise the query will fail. ( Since API v1.5)

Function prototype:

|  |
| --- |
| reqQryPosition ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryPosition ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  ClientNo [str]  // Exchange  ExchangeNo [str]  // Securities code  ProductCode [str]  // Position direction (1: Buy; 2: Sell)  LongShortPosFlag [str]  // Holding cost price  PosCostPrice [str]  // Quantity available for sale  CanSellShares [str]  // Today's purchase quantity  TodayBuyShares [str]  // Freeze quantity  FrosenShares [str]  // Total purchase amount during the holding period  TotalBuyMoney [float]  // Total amount sold during the holding period  TotalSellMoney [float]  // Quantity purchased during the holding period  TotalBuyShares [int]  // Quantity sold during the holding period  TotalSellShares [int]  // First opening date ( yyyy -MM-dd)  FirstPosDate [str]  // Profit and loss of closing position  ClosePosPL [float]  // Number of shares delivered on T1 day, positive in and negative out  T1DeliveryShares [int]  // Number of shares delivered on T2 day, positive in and negative out  T2DeliveryShares [int]  // Number of shares delivered on T3, positive in and negative out  T3DeliveryShares [int]  // Number of undelivered shares = T1+T2  NotDeliveryShares [int]  // Number of shares delivered  DeliveredShares [int]  // Number of shares sold short  SellShortShares [int]  // Short selling frozen funds  SellShortMoney [float]  // Average opening price of short positions ( average opening price )  SSPosAvgCostPrice [float] |

### Funds query request / return reqQryCapital / onRspQryCapital

The minimum interval between two fund queries is 1 second, otherwise the query will fail. ( Since API v1.5)

Function prototype:

|  |
| --- |
| reqQryCapital ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // Not used  Unused [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRsqQryCapital ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Deposit  Deposit [str]  // Withdrawal  Withdraw [str]  // The amount of funds available for trading today ( available today )  TodayTradableFund [str]  // Equity at the beginning of the day ( current balance)  TodayInitialBalance [str]  // Real-time floating equity of the day ( today's equity )  TodayRealtimeBalance [str]  // Freeze funds  FrozenFund [str]  // commission  Commission[str]  // Initial Margin  InitialMargin [str]  // The amount of funds available for trading yesterday ( available yesterday)  YdTradableFund [str]  // Yesterday's beginning equity ( yesterday's balance)  YdInitialBalance [str]  // Yesterday's ending equity ( yesterday's equity )  YdFinalBalance [str]  // Floating profit and loss  ProfitLoss [str]  // Currency number  CurrencyNo [str]  // Currency and base exchange rate  CurrencyRate [float]  // LME unexpired profit ( unexpired profit ) ( not used in stocks)  LMEUnexpiredPL [float]  // LME unrealized profit ( unrealized profit ) ( not used in stocks)  LMEUnaccountPL [float]  //Maintenance Margin  MaintenanceMargin [float]  // Option premium  Premium [float]  // Trust limit  CreditAmount [float]  // Initial funds  IntialFund [float]  // Funding account 1  FundAccountNo [str]  // Mortgage value  MortgageInMoney [str]  // Margin margin limit  MarginLimit [float]  // Borrow value  BorrowInMoney [float]  // Funds delivered on T1 day, positive in and negative out  T1DeliveryMoney [float]  // Funds delivered on T2, positive in and negative out  T2DeliveryMoney [float]  // Funds settled on T3, positive in and negative out  T3DeliveryMoney [float]  // Funds delivered on TN day, positive in and negative out  TNDeliveryMoney [float]  // Transaction limit  TradeLimit [float]  // Available funds == min(T0, T0+T1, T0+T1+T2) - frozen funds (pending order)  CanCashOutMoneyAmount [float]  // Monthly deposit interest  DepositInterest [float]  // Monthly arrears interest  LoanInterest [float]  // Cross-market funding limit  CrossCurrencyMaxMoneyAmt [float]  // Short selling frozen funds  SellShortFrozenMoney [float]  // Short interest  SellShortInterest [float]  // Required margin (when short selling )  ShortPosAddtionalMargin [float]  // error message  ErrorDescription [str] |

### Broker query request /return reqQryBroker / onRspQryBroker

|  |
| --- |
|  |

### Version number query request /return reqQryVersion / onRspQryQryVersion

Function prototype:

|  |
| --- |
| reqQryVersion ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User ID      UserId [str]  // User password      UserPwd [str]  // error message      ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryVersion ( data: dict , error: dict , reqid : int, last: bool) |

Parameter: dat

|  |
| --- |
| // Version number  Version[str]  // Whether it is necessary to update to the new version to use it normally (0: no need to update; 1: must update)  MustUpdate [str]  // The version number that must be updated  MustVersion [str]  // Version content Chinese  VersionContent\_ CN [ str ]  // Version content English  VersionContent\_ US [ str ] |

Parameters: error

### Currency query request / return reqQryCurrency / onRspQryCurrency

Function prototype:

|  |
| --- |
| reqQryCurrency ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // Not used  Unused [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryCurrency ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Currency code  CurrencyNo [str]  // Base currency number  IsBase [int]  // Conversion rate with base currency  ChangeRate [float]  // Currency name  CurrencyName [str]  // Currency name (English)  CurrencyNameEN [str] |

### Trading time query request /return reqQryTradeTime / onRspQryTradeTime

Function prototype:

|  |
| --- |
| reqQryTradeTime ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // Not used  Unused [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryTradeTime ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Current transaction date ( yyyyMMdd )      TradeDate [str]    // Opening time ( HH:mm:ss )      OpenTime [str]    // Closing time ( HH: mm:ss      CloseTime [str] |

### Exchange query request /return reqQryExchange / onRsp QryExchange

Function prototype:

|  |
| --- |
| reqQryExchange ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| ProductGroupID [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryExchange ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Exchange ID  ExchangeNo [str]  // Exchange name  ExchangeName [str]  // Exchange type (HK: Hong Kong stocks; US: US stocks)  SettleType [str]  // Exchange name (English)  NameEN [str] |

Parameters: error

### Contract query request /return reqQryInstrument / onRspQryInstrument

Function prototype:

|  |
| --- |
| reqQryInstrument ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // Not used  Unused [str]  // Number of queries  Index [int]  // Modify time, incremental query  ModifyDay [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryInstrument ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Exchange number      ExchangeNo [str]    // Exchange name      ExchangeName [str]    //Contract NO      CommodityNo [str]    // Product name      CommodityName [str]      // Commodity category, 0: Stocks, 1: Warrants and CBBCs, 2: Stock Options, 3: ETFs, 4: Bonds, 5: Trusts, 6: Others, 8: Indexes      CommodityType [str]      // Currency code      CurrencyNo [str]    // Currency name      CurrencyName [str]  // Point value ( the value of a minimum jump point )  ProductDot [float]  // Minimum change unit  UpperTick [float]  // Previous day's settlement price  SettlePrice [float]  // Transaction month ( yyyyMM )/Delivery date ( yyyyMMdd )  TradeMonth [str]  // Decimal places of market price  DotNum [int]  // Advanced units  LowerTick [int]  // Adjust the decimal places  DotNum\_Carry [int]  // Minimum change unit for adjustment  UpperTick\_Carry [float]  // First notification date ( yyyyMMdd )  FirstNoticeDay [str]  // Freeze margin percentage  FreezenPercent [float]  // Freeze the fixed value of the deposit  FreezenMoney [float]  // Fixed fee  FeeMoney [float]  // Percentage fee  FeePercent [float]  // Spot commodity settlement price yesterday  PriceStrike [float]  // Spot commodity point value  ProductDotStrike [float]  // The minimum change unit of spot goods  UpperTickStrike [float]  // Last trading day ( yyyyMMdd ) (Futures only)  LastTradeDay [str]  // Last updated date ( yyyyMMdd ) (Futures only)  LastUpdateDay [str]  // Option critical price (for futures only)  CriticalPrice [float]  // Minimum jump point below the option critical price (for futures only)  CriticalMinChangedPrice [float]  // Actual exchange (CME is divided into 3 exchanges: CME, CME\_COMEX, CME\_NYMEX) (Futures only)  Exchange2[str]    // Option type (R: call; F: put) (Futures only)      OptionType [str]      // Option year and month ( yyyyMM ) (for futures only)      OptionMonth [str]    // Option execution price (for futures only)      OptionStrikePrice [str]    // The futures product number corresponding to the option (for futures only)      OptionCommodityNo [str]  // Option corresponding futures contract number (for futures only)  OptionContractNo [str]  // Mortgage value percentage      MortgagePercent [str]    //Minimum change unit code      UpperTickCode [str]    // Minimum batch size      LotSize [str]    // Closing time flag (control T+1: 0 represents T+0, you can sell and close the position on the same day; 1 represents T+1, you must sell and close the position the next day; N represents T+N)      FlatTime [str]    // Product name (English)      CommodityFNameEN [str]    // Whether to support short selling (0 or empty: no; 1: yes)      CanSell [str]  // Margin ratio for short selling (e.g. 0.5)  SellRate [float]  // Short position limit  SellMax [float]  // Conversion ratio, which indicates how many shares a CBBC is equivalent to  StrikeRate [float]  // Strike price  StrikePrice [float]  // Callback price  ReceivePrice [float]  // Expiration date ( yyyy -MM-dd)  ExpireDate [str]  // Maintenance margin ratio for short selling (e.g. 0.5)  SellRateKeep [float]  // Turbo CBBC corresponding stock number  StrikeCommodityNo [str]  // The rise and fall symbol of turbo CBBC (C: rise; P: fall)  CallPutFlag [str]  // Issuer of Turbo CBBC  Pu blisher [str] |

### Jump point query request / return reqQryTick / onRspQryTick

Function prototype:

|  |
| --- |
| r eqQryTick ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // Not used  Unused [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryTick ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Minimum change unit code      UpperTickCode [str]    // The minimum change unit starting price      PriceFrom [str]      // The minimum unit of market change      UpperTick [str]      // The point value corresponding to the minimum change unit      ProductDot [str]    // Decimal digits of market price      DotNum [str]    // Purchase price unit      LowerTick [ str] |

### Order type query request /return reqQryOrderType / onRspQryOrderType

|  |
| --- |
|  |

### Set two-factor authentication password / return reqSetVerifyQA / onRspSetVerifyQA

Function prototype:

|  |
| --- |
| r eqSetVerifyQA ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User account  UserId [str]  // Transaction password  UserPwd [str]  // Fixed value, "1"  Type[str]  // Fixed value, "1"  Question[str]  // Two-factor authentication password, alphanumeric  Answer[str]  // Fixed value, "1"  SaveMac [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| o nRspSetVerifyQA ( error: dict , reqid : int, last: bool ) |

Parameters: error

When the callback function error code indicates success, call the reqSafeVerify ( ) function to bind the network card address.

### the MAC address required for two-factor authentication /return reqSafeVerify / onRspSafeVerify

Function prototype:

|  |
| --- |
| r eqSafeVerify ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User account  UserId [str]  // Transaction password  UserPwd [str]  // Fixed value, "1"  Type[str]  // Fixed value, "1"  Question[str]  // Two-factor authentication password, alphanumeric  Answer[str]  // Fixed value, "1"  SaveMac [str]  // MAC address  MacAddress [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| o nRspSafeVerify ( error: dict , reqid : int, last: bool) |

Parameters: error

# Futures Trading API Reference

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Business Type | business | Request interface |  | Response interface |
| connect | connect | i nit | Connection Return | onFrontConnected |
| disconnect |  |  | Disconnection report | onFrontDisconnected |
| Heartbeat |  |  | Heartbeat warning | onHeartBeatWarning |
| Log in | Log in | reqUserLogin | Login Report | onRspUserLogin / OnRspNeedVerify |
| Sign out | reqUserLogout | Logout Report | onRspUserLogout |
| Change Password | reqUserPasswordUpdate | Change password report | onRspUserPasswordUpdate |
| Two-factor authentication step1 | reqSetVerifyQA | Set up a two-factor authentication password | onRspSetVerifyQA |
| Two-factor authentication step2 | reqSafeVerify | Bind MAC address | onRspSafeVerify |
| trade | declaration | reqOrderInsert | Order return | onRspOrderInsert |
| Change order | reqOrder Modify | Change order return | onRspOrder Modify |
| Cancel Order | reqOrderCancel | Cancellation return | onRspOrderCancel |
| Push |  |  | Order status push | onRtnOrder |
|  |  | Transaction information push | onRtnTrade |
|  |  | *Position information push* | *onRtnPosition* |
|  |  | Funding information push | onRtnAccount |
| Query | Order Query | reqQryOrder | Order query return | onRspQryOrder |
| Transaction Inquiry | reqQryTrade | Transaction query report | onRspQryTrade |
| Position Details Query | reqQryPosition | Position Details Query Return | onRspQryPosition |
| Position Summary Query | r eqQryTotalPosition | Position summary query return | onRspQryTotalPosition |
| Funding Inquiry | req QryCapital | Funds Inquiry Return | onRspQryCapital |
| Version number query | reqQryVersion | Version number query report | onRspQryQryVersion |
| Currency Query | reqQryCurrency | Currency query return | onRspQryCurrency |
| Trading time query | reqQryTradeTime | Transaction time query report | onRspQryTradeTime |
| Strategy Query | reqQryStrategy | Strategy query return | onRspQryStrategy |
| Exchange Query | reqQryExchange | Exchange Query Return | onRspQryExchange |
| Contract Query | reqQryInstrument | Contract query report | onRspQryInstrument |
| Strategy details query | reqQryStrategyDetail | Strategy Details Query Return | onRspQryStrategyDetail |
|  | Exchange winter and summer opening time query ( from v1.10) | reqQryExchangeTime | Query Report | onRspQryExchangeTime |
|  | Trading product opening and closing time query ( from v1.10) | reqQryCommodityTime | Query Report | onRspQryCommodityTime |
|  | Query the variety list ( from v1.10) | reqQryCommodityTime | Query Report | onRspQryCommodity |

Note: The above italic interfaces are not implemented in the current API version.

## Interface working process



## Interface Mode

FutureApi class is provided in the future\_demo.py file . By inheriting the FutureApi class, you can issue operation requests and overload callback functions to handle the response of background services.

.

## Function interface description

### createFutureApi method​

Create instance methods.

**Function prototype** :

createFutureApi ( bRecordLog : bool , lpszLogFileName : str = " Future.log" , lpszLogFileDir : str = "" );

bRecordLog : true means recording logs, false means not recording logs.

lpszLogFileName : Log file.

lpszLogFileDir : Log file folder path.

### onFrontConnected method

This method is called when the client establishes a communication connection with the frontend (before logging in).

**Function prototype:**

onFrontConnected ( )

This method is called after initialization is completed, in which the user login task can be completed.

### onFrontDisconnected method

This method is called when the client loses the connection with the frontend communication. When this happens

After that, the API will automatically reconnect and the client does not need to do anything. Automatic reconnection address, which may be the original registration address

It may also be other available communication addresses supported by the system, which are automatically selected by the program.

**Function prototype:**

onFrontDisconnected (int nReason )

**parameter:**

nReason : Reason for disconnection

### onHeartBeatWarning method

Heartbeat timeout warning. This method is called when no message is received for a long time.

**Function prototype:**

onHeartBeatWarning ( nTimeLapse : int )

**parameter:**

nTimeLapse : The time since the last message was received

### Login r eqUserLogin / onRspUserLogin / onRspNeedVerify / onRspAccount

from API v1.5 . For the specific interaction process, please refer to **" International Futures and Stocks API\_Two-factor Authentication Interaction Instructions.xlsx "**

Function prototype:

|  |
| --- |
| reqUserLogin ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User ID  UserId [str]  // User password  UserPwd [str]  // User type  UserType [str]  // Software Name  SoftwareName [str]  // Software version number  SoftwareVersion [str]  // Authorization code  AuthorCode [str]  // Login machine network card MAC address ( since API v1.5, used for dual authentication )  MacAddress [str]  // error message  TDAStringType [str] |

1 ) Function prototype:

|  |
| --- |
| onRspUserLogin ( error: dict , reqid : int, last: bool ) |

Parameters: error

Error code:

|  |  |
| --- | --- |
| **Login error code** | **Error content** |
| 10001 | Incorrect username |
| 10002 | Wrong login password |
| 10003 | The number of incorrect passwords has exceeded the limit and the user has been frozen. Please contact customer service to unfreeze the account! |
| 10004 | The user has been frozen, please contact customer service to unfreeze! |
| 1 0005 | authorization code information is incorrect. Please confirm the values of UserType , SoftwareName , SoftwareVersion , and AuthorCode. |

2 ) Function prototype:

When the server determines that two-factor authentication is required, it will instruct the API to call back this function.

|  |
| --- |
| o nRspNeedVerify ( bFirstLogin : bool , bHasSetQA : bool ) ; |

Parameter: bFirstLogin

|  |
| --- |
| identified by the MacAddress field in the login message ) logs in for the first time before a transaction, o nRspNeedVerify () function is called back, and bFirstLogin The value of is true, and the user needs to call the reqSetVerifyQA ( ) function in the callback function to set the dual authentication information.  MacAddress field in the login message ) logs in for the first time before a transaction, the onRspNeedVerify () function will not be called back. |

1. Function prototype:

After onRspUserLogin indicates a successful login callback, this function is called back and the fund account corresponding to each currency is returned in the data dictionary.

|  |
| --- |
| onRspAccount ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Account currency  CurrencyNo [str]  // Funding account  AccountNo [str] |

### Sign out r eqUserLogout / onRspUserLogout Method

Function prototype:

|  |
| --- |
| reqUserLogout ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User ID  UserId [str]  // Fund account  AccountNo [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspUserLogout ( error: dict , reqid: int , last: bool ) |

Parameters: error

### Change password reqPasswordUpdate / onRspPasswordUpdate

Function prototype:

|  |
| --- |
| reqPasswordUpdate (req: dict , reqid : int) |

Parameters: req

|  |
| --- |
| // User code  UserId [str]  // User's original password  OldPassword [str]  // User's new password  NewPassword [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspPasswordUpdate (data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User code  UserId [str]  // User's original password  OldPassword [str]  // User's new password  NewPassword [str] |

Parameters: error

### Order reqOrderInsert / onRspOrderInsert​

illustrate:

SystemNo [str]: system number

LocalNo [str] : local number

OrderNo [str]: order number

LocalNo is the order number given by the program using the API, which is equivalent to the orderref in CTP . It is best not to repeat it within the same day.

SystemNo is the order number given by the direct system;

OrderNo is the order number given by the exchange, which is equivalent to OrderSysID in CTP

Function prototype:

|  |
| --- |
| reqOrderInsert ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // User ID  UserId [str]  // Fund account (a user can have funds in multiple currencies, so there will be multiple fund accounts, which need to be specified when placing an order )  // Please refer to the description of callback function o nRspAccount after successful login  AccountNo [str]  // Local number  LocalNo [str]  // Exchange code  ExchangeCode [str]  // Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  // Open or close a position: 1=Open, 2=Close, 3=Close today  OpenCloseFlag [str]  // Number of orders  OrderQty [str]  // Order price  OrderPrice [str]  // Order type: 1=limit order, 2=market order, 3=stop to limit, 4=stop to market  OrderType [str]  // Trigger price  TriggerPrice [str]  // Effective date (1=valid today, 2=valid forever (GTC), 4=IOC)  TIF [ str]  // Strategy ID  StrategyId [str]  // Use of iceberg order: the displayed entrustment amount must be less than the entrustment amount  MaxShow [str]  //IOC order usage: minimum transaction volume  // CME Exchange: When the effective date = 4IOC, MinQty >= 1 and less than the order amount, it is FAK, and when MinQty = order amount, it is FOK  // HKEX , SGX: MinQty = 1 for FAK orders, MinQty = commissioned amount for FOK orders, cannot be other values  MinQty [str]  // Tag50 ( order beneficiary/owner )  Tag50 [str]  // error message  ErrorDescription [str] |
| //Programmed order (0) or manual order (1)  IsProgram [int]  // (\*4) Two-letter country code as defined in the ISO (3166-1) standard  OrgOrderLocationID [str]  // Contract type ( Required for cross-period contract: '1'-DERIVATIVE\_CONTRACT\_CATEGORY\_SPREAD, other types are optional by default)  ContractCategory [str] |

illustrate:

1 : After the API user places an order, the onRspOrderInsert function will be called back twice:

1. The first callback returns the system number SystemNo assigned by the direct system to this order , indicating that the direct system has received and confirmed this order.
2. OrderNo of the exchange ( if everything is normal ). Usually when insufficient funds occur, there will be no order number because the order instruction has not been sent to the exchange.

2: Tag 50 (order operator/owner) field description, tag 50 in order modification and cancellation is similar

1. The tag50 field is applicable to CME- affiliated exchanges. Orders placed on non-CME exchanges do not need to set tag50. (If you have any questions about the tag50 order rules, please contact the broker's business personnel for consultation.)
2. If the customer trades for a long time, for example more than 12 hours, multiple rounds of management, operation and monitoring will be required (at least 2 shifts ). When the personnel responsible for management, operation and monitoring change, the new shift personnel need to be assigned a Tag50 different from the previous shift and transmitted in the order .
3. Tag50 (Order Operator/Owner) In the production environment, there are two situations:
   1. If the API is not set or the value is empty, the customer number is used as Tag50 by default (the customer number does not need to be reported to the brokerage company when used as Tag50).
   2. a ) above , you must report Tag50 information to your brokerage firm in advance . Once your brokerage firm sets Tag50, it can be used in the API. Any API order using any unreported Tag50 will be rejected. Customers need to ensure that the Tag50 information transmitted when placing an actual order is consistent with the Tag50 information declared when applying for the API. The logic for assigning   
      Tag50 values is customer number + ' \_001 ' , customer number + ' \_002 ' , and so on;

3 : IsProgram : Distinguish between manual orders and programmed orders.

The default in the API is programmatic order. Customers need to set 0 (programmatic order) or 1 (manual order) according to actual conditions.

4: OrgOrderLocationID

field is only required when there are multiple different customer transactions under the API account (the API program is used as an order routing gateway ) . There are two ways to fill it in:

the country/ region code is Canada, it is followed by a comma and then the province code, for a total of 5 ASCII characters .

Example: Quebec, Canada corresponds to CA, QC

the country/ region code is other than Canada, enter only the country/ region code, a total of 2 ASCII characters .

Example: Japan corresponds to JP, China corresponds to CN

Function prototype:

|  |
| --- |
| onRspOrderInsert ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Funding account  AccountNo [str]  // System number  SystemNo [str]  // Local number  LocalNo [str]  //Order number  OrderNo [str]  // Exchange code  ExchangeCode [str]  //Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  //Number of orders  OrderQty [str]  //Order price  OrderPrice [str]  // Order type: 1=limit order, 2=market order, 3=stop to limit  OrderType [str]  //Order date  OrderDate [str]  // Order time  OrderTime [str]  // Error code (reserved, not used )  ErrorCode [str]  // Order status (2: queued; 3: partially executed; 4: fully executed; 5: remaining orders cancelled; 6: order cancelled; 7: order failed)  OrderState [str]  // Trigger price  TriggerPrice [str]  // Validity date (1=valid today, 2=valid forever (GTC), 4=IOC)  TIF [ str]  // Open or close position: 1=Open 2=Close, 3=Close today, 4=Close yesterday  OpenCloseFlag [str]  // Strategy ID  StrategyId [str]  // Iceberg order: the displayed order amount must be less than the order amount  MaxShow [str]  //Minimum transaction volume  MinQty [str]  // Exchange return time  ExchangeTime [str]  // Terminal type (CME exchange specific)  OrdSourceType [str]  // T ag50 (order beneficiary/owner ), the format is "Organization ID\_Sub - account"  Tag50 [str] |

Parameters: error

|  |
| --- |
| // Error code (used to determine whether the order is successful or failed )  ErrorID [int]  // Error description  ErrorMsg [str] |

Error Code

|  |  |
| --- | --- |
| **System number error code** | **Error content** |
| 20005 | System number generation failed |

Order error code

|  |  |
| --- | --- |
| **Order error code** | **Error content** |
| 20000 | Order failed |
| 20001 | Insufficient funds |
| 20002 | Trading server not connected |
| 20003 | You have been banned from trading, please contact customer service to unblock |
| 20004 | Order rejected |
| 20005 | System number generation failed |
| 20006 | Your fund account does not have the currency required for this contract transaction |
| 20007 | The contract has expired and cannot be traded. |
| 20008 | The trading market is not open and cannot be traded |
| 20009 | The trading market is closed and cannot be traded. |
| 20010 | The order price exceeds the limit |
| 20011 | The order quantity is too large |
| 20012 | You are prohibited from trading this contract, please contact customer service |
| 20013 | Your account risk rate is too high and trading has been prohibited |
| 20015 | Your account transaction data is abnormal, and the transaction is suspended |
| 20019 | The order quantity exceeds the customer's position limit |
| 20020 | The order quantity exceeds the customer's purchase quantity |
| 20021 | The order quantity exceeds the customer's available order quantity |
| 20022 | The order quantity exceeds the company's position limit |
| 20023 | The order quantity exceeds the company's purchase quantity |
| 20024 | The order quantity exceeds the company's available order quantity |
| 20025 | The dealer's order quantity exceeds the maximum order quantity |
| 20026 | When the contract is about to be delivered, you can only close the position, not open a position |
| 20027 | The last trading day of this contract has passed, and electronic trading is prohibited |
| 20028 | The product is not open, trading is prohibited |
| 20029 | The single order quantity exceeds the maximum number of orders that can be placed for this product |
| 20032 | Insufficient positions, position closing failed |
| 20033 | You cannot trade this product, please contact customer service to open |

### Order push o nRtnOrder

Function prototype:

|  |
| --- |
| onRtnOrder ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Exchange code  ExchangeNo [str]  // Contract code  TreatyCode [str]  // Order number  OrderNo [str]  // Order quantity  OrderNumber [int]  // Quantity of transactions completed  FilledNumber [int]  // Average transaction price  FilledAdvPrice [float]  // Buy quantity  BuyHoldNumber [int]  // Average price for opening long positions  BuyHoldOpenPrice [float]  // Holding average price  BuyHoldPrice [float]  // Sell quantity  SaleHoldNumber [int]  // Average price of opening a short position  SaleHoldOpenPrice [float]  // Average selling price  SaleHoldPrice [float]  // Has the order been cancelled? (0: No; 1: The order has been cancelled)  TDAStringType IsCanceled [float]  // Total transaction fee  FilledTotalFee [float]  // Sequence number  Status [int]  // Fund account  AccountNo [str]  // Position type (0: yesterday's position; 1: today's position)  HoldType [str]  // Buy margin  HoldMarginBuy [float]  // Sell margin  HoldMarginSale [float]  // Latest price  CurrPrice [float]  // Floating profit and loss  FloatProfit [float] |

Parameters: error

### Transaction push onRtnTrade

Function prototype:

|  |
| --- |
| onRtnTrade ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Funding account  AccountNo [str]  // Transaction number (including the 7-digit order number, a total of 11 digits)  FilledNo [str]  //Order number  OrderNo [str]  // System number  SystemNo [str]  // Local number  LocalNo [str]  // Exchange code  ExchangeCode [str]  //Contract code  TreatyCode [str]  // Buy or sell: 1=buy 2=sell  BuySale [str]  // Number of transactions  FilledNumber [str]  // Transaction price  FilledPrice [str]  // Transaction date ( yyyy -MM-dd)  FilledDate [str]  // Transaction time ( hh:mm:ss )  FilledTime [str]  // Transaction fee  Commsion [str]  // Order quantity  OrderNumber [str]  //Commission price  OrderPrice [str]  // Contract delivery date ( yyyyMMdd )  DeliveryDate [str]  // Transaction type (N: ordinary order transaction; C, T: simulated transaction of swap)  FilledType [str]  // Order type (1=limit order, 2=market order, 3=stop to limit, 4=stop to market)  OrderType [str]  // Valid date (1 = valid today, 2 = valid forever)  ValidDate [str]  // Open or close position: 1=Open 2=Close, 3=Close today, 4= Close yesterday  AddReduce [str] |

### Position information push onRtnPosition

### Funding information push onRtnCapital

Function prototype:

|  |
| --- |
| onRtnCapital ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Customer Number  ClientNo [str]  // Fund account  AccountNo [str]  // Currency  CurrencyNo [str]  // Available now  Available [float]  // Available yesterday  YAvailable [float]  // Available today  CanCashOut [float]  // Today's balance  Money [float]  // Profit and loss of closing position  ExpiredProfit [float]  // Freeze funds  FrozenDeposit [float]  // Handling Fee  Fee [float]  // Security deposit  Deposit [float]  // Maintenance Margin  KeepDeposit [float]  //state  Status [int]  // Deposit  InMoney [float]  // Withdrawal  OutMoney [float]  // Unexpired profit  UnexpiredProfit [float]  // Today's equity  TodayTotal [float]  // Unresolved profit  UnaccountProfit [float]  // Option premium  Royalty [float]  // Exchange code  ExchangeNo [str]  // Contract code  TreatyCode [str]  // Order number  OrderNo [str]  // Order quantity  OrderNumber [int]  // Quantity of transactions completed  FilledNumber [int]  // Average transaction price  FilledAdvPrice [float]  // Buy quantity  BuyHoldNumber [int]  // Average price for opening long positions  BuyHoldOpenPrice [float]  // Holding average price  BuyHoldPrice [float]  // Sell quantity  SaleHoldNumber [int]  // Average price of opening a short position  SaleHoldOpenPrice [float]  // Average selling price  SaleHoldPrice [float]  // Whether the order has been cancelled ( 0 : No; 1 : The order has been cancelled)  IsCanceled [str]  // Total transaction fee  FilledTotalFee [float]  // Trust limit  Credit [float]  // Margin margin limit for Hong Kong stocks  MarginLimit [float]  // Loan value for Hong Kong stocks  BorrowValue [float]  // Mortgage value for Hong Kong stocks  FMortgageMoney [float]  // T1 for Hong Kong stocks  T1 [float]  // T2 for Hong Kong stocks  T2 [float]  //T3 for Hong Kong stocks  T3 [float]  // Tn for Hong Kong stocks  TN [ float]  // Transaction limit  TradeLimit [float]  // Cross-market funding limit  FCrossMax [float]  // Short selling frozen funds  SellFreezenMoney [float]  // Short interest  SellInterest [float]  // Required deposit  SellNeedAddMargin [float] |

Note: There are stock-related fields in the callback data dictionary, please ignore them when using them.

### Order modification request /return reqOrderModify / onRspOrderModify method

Function prototype:

|  |
| --- |
| reqOrderModify ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // System number  SystemNo [str]  // User ID  UserId [str]  // Local number  LocalNo [str]  // Funding account  AccountNo [str]  //Order number  OrderNo [str]  // Exchange code  ExchangeCode [str]  //Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  //Number of orders  OrderQty [str]  //Order price  OrderPrice [str]  //Change the number  ModifyQty [str]  // Change order price  ModifyPrice [str]  // Price type: 1=limit order, 2=market order, 3=stop to limit, 4=stop to market  OrderType [str]  // Trigger price  TriggerPrice [str]  // Change order trigger price  ModifyTriggerPrice [str]  // Valid date (1: valid today; 2: valid forever)  TIF [ str]  // Order ID  Tag50[str]  // error message  ErrorDescription [str]  // Two-letter country/region code defined in the ISO (3166-1) standard  OrgOrderLocationID [str] |

Function prototype:

|  |
| --- |
| onRspOrderModify ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: error

|  |
| --- |
| // Error code (used to determine whether the order modification is successful or failed )  ErrorID [int]  // Error description  ErrorMsg [str] |

### Cancel order request / return reqOrderCancel / onRspOrderCancel

Function prototype:

|  |
| --- |
| reqOrderCancel ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // User ID  UserId [str]  // User type: 1: general user; 2: institutional user;  UserType [str]  // Local number  LocalNo [str]  // Fund account  AccountNo [str]  // System number  SystemNo [str]  // Order number  OrderNo [str]  // Exchange code  ExchangeCode [str]  // Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  // Number of orders  OrderQty [str]  // Order price  OrderPrice [str]  // Price type: 1=limit order, 2=market order  OrderType [str]  // error message  ErrorDescription [str]  // Two-letter country/region code defined in the ISO (3166-1) standard  OrgOrderLocationID [str]  // Contract type ( Required for cross-period contract: '1'-DERIVATIVE\_CONTRACT\_CATEGORY\_SPREAD, other types are optional by default)  ContractCategory [str] |

Function prototype:

|  |
| --- |
| onRspOrderCancel data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Fund account  AccountNo [str]  // System number  SystemNo [str]  // Local Number  LocalNo [str]  //Order number  OrderNo [str]  // Cancellation order number  CancelNo [str]  // Exchange code  ExchangeCode [str]  // Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  // Number of orders  OrderQty [str]  // Order price  OrderPrice [str]  // Number of transactions completed  FilledQty [str]  // Number of cancelled orders  CancelledQty [str]  // Price type: 1=limit order, 2=market order  OrderType [str]  // Cancellation date  CancelledDate [str]  // Cancellation time  CancelledTime [str]  // Error code (reserved, not used)  ErrorCode [str] |

Parameters: error

|  |
| --- |
| // Error code (used to determine whether the order cancellation is successful or failed )  ErrorID [int]  // Error description  ErrorMsg [str] |

### Order query /return reqQryOrder / onRspQryOrder

The minimum interval between two order queries is 1 second, otherwise the query will fail. ( Since API v1.5)

Function prototype:

|  |
| --- |
| reqQryOrder ( req : dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User ID  UserId [str]  // Fund account  AccountNo [str]  // Whether to simulate the user: 1: Yes; 0 or other: No  IsSimulation [str]  // Get the orders after the specified order number  OrderNo [str]  // Get the orders after the specified order time (format: yyyy -MM-dd hh:mm:ss )  OrderDateTime [str]  // error message  ErrorDescription [str]  // Order filter conditions ( Since API v1.18.2.2)  // Value : DERIVATIVE\_ORD\_FILTER\_L, returns live orders ( uncancelled orders waiting to be filled ) // DERIVATIVE\_ORD\_FILTER\_A ( default value ), returns all orders, including fully filled and cancelled orders  FilterFlag [str] |
|  |

Function prototype:

|  |
| --- |
| onRspQryOrder ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Fund account  AccountNo [str]  // System number  SystemNo [str]  // Local number  LocalNo [str]  // Order number  OrderNo [str]  // Exchange code  ExchangeCode [str]  // Contract code  ContractCode [str]  // Buy or sell: 1=buy 2=sell  BidAskFlag [str]  // Number of orders  OrderQty [str]  // Order price  OrderPrice [str]  // Number of transactions completed  FilledQty [str]  // Average transaction price  FilledPrice [str]  // Order type: 1=limit order, 2=market order, 3=stop to limit, 4=stop to market  OrderType [str]  // Order date  OrderDate [str]  // Order time  OrderTime [str]  // Error code  ErrorCode [str]  // Order status (1: requested; 2: queued; 3: partially executed; 4: fully executed; 5: remaining orders canceled; 6: order canceled; 7: instruction failed;)  OrderState [str]  // User ID for canceling order  CancelUserId [str]  // Trigger price  TriggerPrice [str]  // Effective date (1=valid today, 2=valid forever (GTC), 4=IOC)  TIF [ str]  // Open or close position: 1=Open, 2=Close, 3=Close today, 4=Close yesterday  OpenCloseFlag [str]  // Strategy ID  StrategyId [str]  // Display the entrustment amount must be less than the entrustment amount  MaxShow [str]  // Minimum transaction volume  MinQty [str]  // Exchange return time  ExchangeTime [str]  // Cancellation time  CancelTime [str] |

Parameters: error

### Transaction query request /return reqQryTrade / onRsp QryTrade

The minimum interval between two transaction queries is 1 second, otherwise the query will fail. ( Since API v1.5)

Function prototype:

|  |
| --- |
| reqQryTrade ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // user  UserId [str]  // error message  ErrorDescription [str] |

Since there may be a lot of transaction data, add paging query method. Please refer to the code in demo for details .

( Since API v1.15)

Function prototype:

|  |
| --- |
| onRspQryTrade ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Funding account  AccountNo [str]  // Transaction number (including the 7-digit order number, a total of 11 digits)  FilledNo [str]  //Order number  OrderNo [str]  // System number  SystemNo [str]  // Local number  LocalNo [str]  // Exchange code  ExchangeCode [str]  //Contract code  TreatyCode [str]  // Buy or sell: 1=buy 2=sell  BuySale [str]  // Number of transactions  FilledNumber [str]  // Transaction price  FilledPrice [str]  // Transaction date ( yyyy -MM-dd)  FilledDate [str]  // Transaction time ( hh:mm:ss )  FilledTime [str]  // Transaction fee  Commsion [str]  // Order quantity  OrderNumber [str]  //Commission price  OrderPrice [str]  // Contract delivery date ( yyyyMMdd )  DeliveryDate [str]  // Transaction type (N: ordinary order transaction; C, T: simulated transaction of swap)  FilledType [str]  // Order type (1=limit order, 2=market order, 3=stop to limit, 4=stop to market)  OrderType [str]  // Valid date (1 = valid today, 2 = valid forever)  ValidDate [str]  // Open or close position: 1=Open 2=Close, 3=Close today, 4= Close yesterday  AddReduce [str]  // error message  ErrorDescription [str] |

Parameters: error

|  |
| --- |
| // Error code  ErrorID [int]  // Error description  ErrorMsg [str] |

### Position details query request/return reqQryPosition / onRspQryPosition

The minimum interval between two position detail queries is 1 second, otherwise the query will fail. ( Since API v1.5)

Function prototype:

|  |
| --- |
| reqQryPosition ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryPosition ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  ClientNo [str]  // Exchange  ExchangeNo [str]  // Securities code  CommodityNo [str]  // Position direction (1: Buy; 2: Sell)  Direct[str]  // Holding cost price  HoldPrice [float]  // Quantity available for sale  CanTradeVol [int]  // Today's purchase quantity  TodayBuyVol [int]  // Freeze quantity  FrozenVol [int]  // Total purchase amount during the holding period  TotalBuyMoney [float]  // Total amount sold during the holding period  TotalSellMoney [float]  // Quantity purchased during the holding period  TotalBuyVol [int]  // Quantity sold during the holding period  TotalSellVol [int]  // First opening date ( yyyy -MM-dd)  OpenDate [str]  // Close the position to make a profit  FlatProfit [float]  // Hong Kong Stock T+1 Quantity  HkexT1[int]  // Hong Kong Stock T+2 Quantity  HkexT2[int]  // US stock T+3 quantity  HkexT3[int]  // Number of unsettled Hong Kong stocks = T1+T2  UnsettleVol [int]  // Number of Hong Kong stocks delivered  SettledVol [int]  // Number of positions held  HoldVol [int]  // Today's sales quantity  TodaySaleVol [int]  // Short selling frozen funds  SellFrozenMoney [float]  // Average opening price  OpenPrice [float] |

Parameters: error

### Funds query request / return reqQryCapital /

### onRspQryCapital

The minimum interval between two fund queries is 1 second, otherwise the query will fail. ( Since API v1.5)

Function prototype:

|  |
| --- |
| reqQryCapital ( req : dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // Not used  Unused [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryCapital ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID  UserId [str]  // Deposit  Deposit [str]  // Withdrawal  Withdraw [str]  // The amount of funds available for trading today ( available today )  TodayTradableFund [str]  // Equity at the beginning of the day ( current balance)  TodayInitialBalance [str]  // Real-time floating equity of the day ( today's equity )  TodayRealtimeBalance [str]  // Freeze funds  FrozenFund [str]  // commission  Commission[str]  // Initial Margin  InitialMargin [str]  // The amount of funds available for trading yesterday ( available yesterday)  YdTradableFund [str]  // Yesterday's beginning equity ( yesterday's balance)  YdInitialBalance [str]  // Yesterday's ending equity ( yesterday's equity )  YdFinalBalance [str]  // Floating profit and loss  ProfitLoss [str]  //Currency code  CurrencyNo [str]  // Currency and base exchange rate  CurrencyRate [float]  // LME unexpired profit ( unexpired profit )  LMEUnexpiredPL [float]  // LME unrealized profit ( unrealized profit )  LMEUnaccountPL [float]  //Maintenance Margin  MaintenanceMargin [float]  // Option premium  Premium [float]  // Trust limit  CreditAmount [float]  // Initial funds  IntialFund [float]  // Funding account 1  FundAccountNo [str]  // Transaction limit  TradeLimit [float]  // Available funds ==min(T+0,T+0+T+1,T+0+T+1+T+2)-frozen funds (pending order)  CanCashOutMoneyAmount [float]  // Monthly deposit interest  DepositInterest [float]  // Monthly arrears interest  LoanInterest [float]  // error message  ErrorDescription [str] |

Parameters: error

### Version number query request/return reqQryVersion / onRspQryVersion

Function prototype:

|  |
| --- |
| reqQryVersion ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User ID  UserId [str]  // User password  UserPwd [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryVersion ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Version number  Version[str]  // Whether it is necessary to update to the new version to use it normally (0: no need to update; 1: must update)  MustUpdate [str]  // The version number that must be updated  MustVersion [str]  // Version content Chinese  VersionContent\_ CN [ str ]  // Version content English  VersionContent\_ US [ str ] |

Parameters: error

### Currency query request / return reqQryCurrency / onRspQryCurrency

Function prototype:

|  |
| --- |
| reqQryCurrency ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // Not used  Unused [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryCurrency ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Currency code  CurrencyNo [str]  // Base currency number  IsBase [int]  // Conversion rate with base currency  ChangeRate [float]  // Currency name  CurrencyName [str]  // Currency name (English)  CurrencyNameEN [str] |

Parameters: error

### Trading time query request/return reqQryTradeTime / onRspQryTradeTime

Function prototype:

|  |
| --- |
| reqQryTradeTime ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // Not used  Unused [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryTradeTime ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Current transaction date ( yyyyMMdd )  TradeDate [str]  // Opening time ( HH:mm:ss )  OpenTime [str]  // Closing time ( HH: mm:ss  CloseTime [str] |

Parameters: error

### Exchange query request/return reqQryExchange / onRspQryExchange

Function prototype:

|  |
| --- |
| reqQryExchange ( req: dict , reqid : int ) |

Parameters: data

|  |
| --- |
| ProductGroupID [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryExchange ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Exchange ID  ExchangeNo [str]  // Exchange name  ExchangeName [str]  // Exchange type (HK: Hong Kong stocks; US: US stocks)  SettleType [str]  // Exchange name (English)  NameEN [str] |

Parameters: error

### Contract query request /return reqQryInstrument / onRspQryInstrument

Function prototype:

|  |
| --- |
| reqQryInstrument ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // How many records to query, and how many records to return each time  PageIndex [int]  // Exchange code, if filled with a value, query the contract of an exchange  ExchangeNo [str]  // Contract code. Query a single contract  CommodityNo [str]  // Product category  CommodityType [str]  //Contract NO  ContractNo [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryInstrument ( data: dict , error: dict , reqid : int, last: bool) |

Parameter: CFutureRspInstrumentField

|  |
| --- |
| // Exchange number  ExchangeNo [str]  // Exchange name  ExchangeName [str]  // Contract NO  CommodityNo [str]  // Product name  CommodityName [str]  // Commodity category, O: options, F: futures, S: inter-period arbitrage of the same commodity  CommodityType [str]  // Currency code  CurrencyNo [str]  // Currency name  CurrencyName [str]  // Point value ( the value of a minimum jump point )  ProductDot [float]  // Minimum change unit  UpperTick [float]  // Previous day's settlement price  SettlePrice [float]  // Transaction month ( yyyyMM )/Delivery date ( yyyyMMdd )  TradeMonth [str]  // Decimal places of market price  DotNum [int]  // Advanced units  LowerTick [int]  // Adjust the decimal places  DotNum\_Carry [int]  // Minimum change unit for adjustment  UpperTick\_Carry [float]  // First notification date ( yyyyMMdd )  FirstNoticeDay [str]  // Freeze margin percentage  FreezenPercent [float]  // Freeze the fixed value of the deposit  FreezenMoney [float]  // Fixed fee  FeeMoney [float]  // Percentage fee  FeePercent [float]  // Spot commodity settlement price yesterday  PriceStrike [float]  // Spot commodity point value  ProductDotStrike [float]  // The minimum change unit of spot goods  UpperTickStrike [float]  // Last trading day ( yyyyMMdd ) (Futures only)  LastTradeDay [str]  // Last updated date ( yyyyMMdd ) (Futures only)  LastUpdateDay [str]  // Option critical price (for futures only)  CriticalPrice [float]  // Minimum jump point below the option critical price (for futures only)  CriticalMinChangedPrice [float]  // Actual exchange (CME is divided into 3 exchanges: CME, CME\_COMEX, CME\_NYMEX) (Futures only)  Exchange2 [str]  // Option type (R: call; F: put) (for futures only)  OptionType [str]  // Option year and month ( yyyyMM ) (for futures only)  OptionMonth [str]  // Option execution price (for futures only)  OptionStrikePrice [str]  // Option corresponding to the futures product number (futures only)  OptionCommodityNo [str]  // Option corresponding futures contract number (for futures only)  OptionContractNo [str]  // Mortgage value percentage  MortgagePercent [str]  // Minimum change unit code  UpperTickCode [str]  // Minimum batch size  LotSize [str]  // Close time flag (control T+1: 0 represents T+0, you can close the position by selling on the same day; 1 represents T+1, you must close the position by selling on the next day; N represents T+N)  FlatTime [str]  // Product Name (English)  CommodityFNameEN [str]  // Whether to support short selling (0 or empty: no; 1: yes)  CanSell [str]  // Margin ratio for short selling (e.g. 0.5)  SellRate [float]  // Short position limit  SellMax [float]  // Conversion ratio, which indicates how many shares a CBBC is equivalent to  StrikeRate [float]  // Strike price  StrikePrice [float]  // Callback price  ReceivePrice [float]  // Expiration date ( yyyy -MM-dd)  ExpireDate [str]  // Maintenance margin ratio for short selling (e.g. 0.5)  SellRateKeep [float]  // Turbo CBBC corresponding stock number  StrikeCommodityNo [str]  // The rise and fall symbol of turbo CBBC (C: rise; P: fall)  CallPutFlag [str]  // Issuer of Turbo CBBC  Publisher[str] |

### Strategy query request / return reqQryStrategy / onRspQryStrategy

Function prototype:

|  |
| --- |
| reqQryStrategy ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // Contract code. Query a single contract  InstrumentID[str]  // Exchange code, if filled with a value, query the contract of an exchange  ExchangeID [str]  //Contract code on the exchange  ExchangeInstID [str]  // Product code, if you fill in a value, query the contract of a product  ProductID [str]  // Start time, if filled in, it will be added after this time  InsertTimeStart [str]  // How many records to query, and how many records to return each time  Index[int]  // Query the contract category directly  ProductType [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryStrategy ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // User ID to which the policy belongs  UserId [str]  // Strategy Id  FKeyId [str]  // Strategy name  FName[str]  // Strategy code  FCode [str]  // Strategy price type  PriceType [int]  // Strategy price details type  PriceTypeDetailType [int]  // Strategy price formula  PriceFormula [str]  // Trigger mode  TriggerMethod [int]  // Whether to enable internal disk protection  InnerProtect [int]  // Whether to immediately close the corresponding active leg if the passive leg order fails  PassiveFailCloseMainLeg [int]  // Trigger mode is 2: Slippage number set when transaction priority is given  SlipPoint [int]  // Add protection strategy  RecoverPriceMethod [int]  // How many seconds after the passive leg order is placed, follow up the order at the market price  RecoverPriceSeconds [int]  // Policy setting type  SetType [str]  // The minimum price change unit of the strategy market  MinChangePrice [float]  // Maximum single order quantity for active leg  MaxNum [int]  //Minimum pending order quantity for passive leg contracts  SuportQuantity [int]  // Safe depth  SafeDeep [int]  // Active area  MainRange [int]  // Single-leg manual pursuit passive setting  ManualZhuiDanBeyondNum [str]  // Single-leg manual level active setting  ManualPingZhuDongBeyondNum [str]  // Single-leg automatic timing setting, number of seconds for pending orders  AutoGuaDanSeconds [int]  // Single leg automatic timing pursuit passive setting  AutoZhuiBeiDongDots [int]  // Single leg automatic timing level active setting  AutoPingZhuDongDots [int]  // Single leg automatic stop loss setting  AutoZhiSunDot [int]  // Single-leg automatic stop loss pursuit passive settings, pursuit price slippage points :  AutoZhiSunZhuiBeiDongDots[int]  // Single-leg automatic stop loss level active setting, price pursuit slippage :  AutoZhiSunPingZhuDongDots [int]  // The number of decimal places for the strategy quote price  DotLen [int]  // Set the trading time period  TradeTime [str] |

Parameters: error

### Strategy detail query request /return reqQryStrategyDetail / onRspQryStrategyDetail

Function prototype:

|  |
| --- |
| reqQryStrategyDetail ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // strategy commodity no  StrategyCommodityNo [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| void onRspQryStrategyDetail ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Strategy product number  StartegyCommodityNo [str]  // Strategy contract combination number  StartegyContractNo [str]  // price  Price[str]  //Number of legs (2 contracts arbitrage is 2 legs)  LegNum [int]  // Last trading day ( yyyy -MM-dd)  LastTradeDate [str]  //Contract sequence number in the strategy  SeqId [int]  // The product number of the contract in the strategy  CommodityNo [str]  // Contract number of the contract in the strategy  ContractNo [str]  // Buy and sell direction (1 means buy, 2 means sell)  Direct[str]  //Contract order quantity  Quantity[int]  //Contract order quantity  Model[str]  // Last updated date ( yyyy -MM-dd)  ModifyDate [str]  // Order type (limit price, market price)  OrderMethod [str]  // Price tolerance  PriceTolerance [float]  // Price multiplier parameters  PriceParam [float]  // Minimum order quantity for the contract  SuportQuantity [int]  // Minimum contract movement  MinMove [int]  //Contract timing follow-up  TimerOrder [int]  //Slippage of scheduled order tracking  TimerOrderBeyondNum [str]  //Contract timing price tracking  TimerOrderPrice [int]  //Timed price tracking slippage  TimerOrderPriceBeyondNum [str]  //Timed price chasing trigger times  TimerOrderPriceTriggerNum [str]  // Stop loss order  ZhiSunOrder [int]  // Stop loss order slippage  ZhiSunOrderBeyondNum [str]  //Contract timed closing  TimerPingCang [int]  // Timed closing slippage  TimerPingCangBeyondNum [str]  // Safe depth  SafeDeep [int]  // Active area  MainRange [ int] |

Parameters: error

### Set two-factor authentication password / return reqSetVerifyQA / onRspSetVerifyQA

Function prototype:

|  |
| --- |
| r eqSetVerifyQA ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User account  UserId [str]  // Transaction password  UserPwd [str]  // Fixed value, "1"  Type[str]  // Fixed value, "1"  Question[str]  // Two-factor authentication password, alphanumeric  Answer[str]  // Fixed value, "1"  SaveMac [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspSetVerifyQA ( error: dict , reqid : int, last: bool ) |

Parameters: error

When the callback function error code indicates success, call the reqSafeVerify function to bind the network card address.

### the MAC address required for dual authentication Request/Return r eqSafeVerify / o nRspSafeVerify

Function prototype:

|  |
| --- |
| reqSafeVerify ( req: dict , reqid : int ) |

Parameter: req

|  |
| --- |
| // User account  UserId [str]  // Transaction password  UserPwd [str]  // Fixed value, "1"  Type[str]  // Fixed value, "1"  Question[str]  // Two-factor authentication password, alphanumeric  Answer[str]  // Fixed value, "1"  SaveMac [str]  // MAC address  MacAddress [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspSafeVerify ( error: dict , reqid : int, last: bool) |

Parameters: error

### Query the commodity list request / return reqQryCommodity / onRspQryCommodity

Function prototype:

|  |
| --- |
| reqQryCommodity ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // Update date  UpdateDate [str]  // Exchange number  ExchangeNo [str]  // error message  ErrorDescription [str] |

Note: When UpdateDate is not filled in, all supported varieties are returned. When a date is entered, the types that have changed since that date are returned.

Function prototype:

|  |
| --- |
| onRspQryCommodity ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Product number, such as GC, CL  CommodityNo [str]  // Exchange code, such as CME, HKEX  ExchangeNo [str]  // Product type  CommodityType [str]  // Product name  Name[str]  // Transaction status  Enabled[int]  // Update date  RegDate [int]  // Currency number  CurrencyNo [float]  // Book jump point value  ProductDot [float]  // Advanced units  LowerTick [float]  // Minimum change unit  UpperTick [float]  // Decimal places of market price  DotNum [int]  // Strategy contract ID  StrikeCommodityId [int]  // Option type  OptionStyle [str]  // Exchange number 2 ( Exchanges under ExchangeNo such as CME\_COMEX, CME\_ NYMEX CME\_CBT, CME)  ExchangeNo2[str]  // Is it a stock futures?  IsSFuture [float] |

illustrate:

ExchangeNo2/ Exchange No. 2, this field is used for market information payment. Taking CME as an example, CME allows investors to choose one or more of CME\_COMEX , CME\_ NYMEX , CME\_CBT , CME to pay. Correspondingly, when API users subscribe to market information by exchange, they should fill in this exchange code.

### Position summary query request /return reqQryTotalPosition / onRspQryTotalPosition

Function prototype:

|  |
| --- |
| r eqQryTotalPosition ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // Fund account  AccountNo [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryTotalPosition ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Local order number  LocalOrderNo [str]  // Exchange code  ExchangeNo [str]  // Contract code  ContractCode [str]  // Order number  OrderNo [str]  // Order quantity  OrderQty [int]  // Quantity of transactions completed  FilledQty [int]  // Average transaction price  FilledAvgPrice [float]  // Long position/buy quantity  LongPositionQty [int]  // Average opening price of long position/holding position  LongPosAveragePrx [float]  // Average price ( holding and buying average price ) for Chinese varieties (such as INE/ sc )  CNLongPosAveragePrx [float]  // Short position/ holding quantity  ShortPositionQty [int]  // Average opening price of short position/short position  ShortPosAveragePrx [float]  // Average price ( holding and selling price ) for Chinese products (such as INE/ sc )  CNShortPosAveragePrx [float]  // Has the order been cancelled? (0: No; 1: The order has been cancelled)  IsCanceled [str]  // Total transaction fee  FilledTotalFee [float]  // Sequence number  SequenceNo [int]  // Fund account  AccountNo [str]  // Position type (0: yesterday's position; 1: today's position)  PositionType [str]  // Long Position/Buy Margin  LongPosMargin [float]  // Short position/ sell margin  ShortPosMargin [float]  // Latest price  CurrPrice [float]  // Floating profit and loss  ProfitLoss [float] |

### Exchange winter and summer opening time query request /return reqQryExchangeTime / onRspQryExchangeTime

Function prototype:

|  |
| --- |
| reqQryExchangeTime ( req: dict , reqid : int ) |

Parameters: req

|  |
| --- |
| // Not used  Unused [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryExchangeTime ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Current year  Year[str]  // Daylight saving time start time  SummerBegin [str]  // Winter time start time  WinterBegin [str]  // Exchange code  ExchangeNo [str]  // Exchange name  Name [int] |

### Query the opening and closing time of a product / return r eqQryExchangeTime / o nRspQryCommodityTime

Function prototype:

|  |
| --- |
| reqQryCommodityTime ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| // Exchange code  ExchangeNo [str]  //Variety code  CommodityNo [str]  // error message  ErrorDescription [str] |

Function prototype:

|  |
| --- |
| onRspQryCommodityTime ( data: dict , error: dict , reqid : int, last: bool) |

Parameters: data

|  |
| --- |
| //Whether it crosses the day  CrossTrade [str]  //Transaction segment  Stage[str]  //2 winter and 1 summer  Summer[str]  //Product opening time  Opendate [str]  //Product closing time  Closingdate [str]  //Variety code  CommodityNo [str]  //Variety name  ComName [str]  //Exchange code  ExchangeNo [str]  //Exchange name  ExName [str] |

# FAQ - Futures Trading Related

1. Contract multiplier

Contract multiplier = (point value/minimum change unit)\*( advanced unit /10 to the power of (number of decimal places in the market price))

illustrate:

If you don't do U.S. debt

You can only view the point value/minimum change unit

Because most varieties are in decimal and the result is 1

2. Margin

Margin is the margin verified when placing an order, and maintenance margin is the margin calculated at other times.

The point value of a large point = contract point value \* purchase price unit, divided by 10 to the power of the number of decimal points, and then divided by the minimum jump point

3. Profit and Loss

Floating profit of holdings = number of positions \* (opening price - latest price) \* point value of one large point

Both the opening price and the last price must be converted into decimal prices

Decimal price = price integer + decimal \* 10 to the power of the decimal place divided by the purchase price unit

Contract point value = 336 Purchase price unit = 532 Decimal point power = 531 Minimum jump point = 337

4. Positions and Equity

Positions: Calculated based on the requested position details, and positions are closed directly in chronological order.

Mark-to-Market Profit and Loss: (yesterday's settlement price - latest market price) \* number of positions \* contract point value;

Current equity: current balance + floating profit + unrealized profit + unexpired profit

Today's equity = Tags.todayAccount + floating profit + Tags. LMEUnaccountPL + Tags . LMEUnexpiredPL ;

Available now: Current equity - Margin - Frozen funds

Available today = current equity - Tags. Deposit - Tags. frozenDeposit ;

For overnight positions, the opening price is yesterday's settlement price, and the floating profit is calculated using yesterday's settlement price.

5. Options

Options:

Premium: option contract opening price \* minimum tick value / minimum tick \* lot size;

Option value: the latest price of the option contract \* the value of the minimum jump point / the minimum jump point \* the number of lots;

Net profit or loss of option: option value - premium;

6. Cancellation of order

You need to fill in the system number, order number, commodity exchange and order direction

7. Tag 50 field description, tag 50 is the same in the order change and cancellation

1. You must use the Operator ID (formerly known as Tag50) registered with the brokerage company to place orders, otherwise the order will be rejected. Customers who apply to open API can register and declare Tag50 through the "Due Diligence and Operator Authorization Letter";
2. 2. If the customer trades for a long time, for example, more than 12 hours, multiple rounds of management, operation and monitoring are required (at least 2 shifts ). When the personnel responsible for management, operation and monitoring change, the new shift personnel need to be assigned a Tag50 different from the previous shift and transmitted in the order .

# FAQ - Stock Trading Related

1. get cash balance (USD)

Check fund balance

Available funds : CanCashOutMoneyAmount

Available funds: FBalance (DB) structure ( TodayTradableFund )

Available funds include money that is loaned directly to customers.

Available funds are the client's own money.

So available funds may be greater than available funds, divided into two fields.

2. get total asset value (USD)

Check total assets , which is available funds + holding market value

Market value->No direct result, customers need to calculate it themselves.

(Latest Price \* Total Holdings ( CanSellShares + FrosenShares ))

The total holdings consist of two parts : CanSellShares + FrosenShares

Frozen quantity: When a customer has unfulfilled orders in the queue, the number of queued orders is the frozen quantity.

3. get positions (should return list of symbols and its holdings)

Check holdings (the result should include stock code and corresponding number of shares)

CanSellShares: The amount of shares that can be sold, taking into account the frozen amount of the queued orders ( FrosenShares )

4. Place mkt order (buy and sell)

Place market buy and market sell orders

5. Cancellation of order

You need to fill in the system number, order number, commodity exchange and order direction