**Python行情API接口开发规范**

目录

[一、 概览 3](#_Toc184912737)

[1.1. 背景 3](#_Toc184912738)

[1.2. API接口概要及架构 4](#_Toc184912739)

[1.3. 全局的错误代码 5](#_Toc184912740)

[二、 期货股票通用行情API接口参考 7](#_Toc184912741)

[1. 接口工作过程 7](#_Toc184912742)

[2. 接口模式 8](#_Toc184912743)

[3. 函数接口说明 8](#_Toc184912744)

[3.1. API实例对象创建 8](#_Toc184912750)

[3.2. API函数请求与回调方法 8](#_Toc184912751)

[1）onFrontConnected 方法 9](#_Toc184912752)

[2）onFrontDisconnected方法 9](#_Toc184912753)

[3）onHeartBeatWarning方法 9](#_Toc184912754)

[4）设置心跳超时时间setHeartBeatTimeout 9](#_Toc184912755)

[5）登录reqUserLogin/onRspUserLogin 11](#_Toc184912756)

[6）行情订阅请求/响应reqMarketData/onRspMarketData 11](#_Toc184912757)

[7）期货盘口行情推送onRtnSnapData 14](#_Toc184912758)

[8）期货成交数据推送onRtnDealData 17](#_Toc184912759)

[9）期货统计数据数据推送onRtnStatisticsData 17](#_Toc184912760)

[10）股票行情推送onRtnMarketData 18](#_Toc184912761)

[11）港股经纪商获取请求reqBrokerData/OnRtnBrokerData 21](#_Toc184912762)

[12）港股经纪商推送onRtnBrokerData 22](#_Toc184912763)

[13）交易日推送onRtnTradeDate 23](#_Toc184912764)

[14）通用错误onRspError 23](#_Toc184912765)

[6) 常见问题-股票行情相关 24](#_Toc184912766)

# 概览

## 背景

本API可用来获取相应市场的行情数据。

目前支持的交易所/市场有以下(但不限于)这些：

期货类

|  |  |  |
| --- | --- | --- |
| No | 交易所 | 说明 |
| 1 | CME | 包括CME group旗下的CME,COMEX,NYMEX,CBOT  官方网站：  https://www.cmegroup.com/ |
| 2 | HKEX | 香港交易所  官方网站：  http://www.hkex.com.hk |
| 3 | SGX | 新加坡交易所  官方网站：  http://www.sgx.com |
| 4 | ICE | 洲际交易所  官方网站：  https://www.theice.com/index |
| 5 | LME | 伦敦金属交易所  官方网站：  https://www.lme.com/ |

股票类

|  |  |  |
| --- | --- | --- |
| No | 市场 | 说明 |
| 1 | NASD | 美国市场的股票。  需要注意的是，2008年金融危机后，美国通过法案，允许上市公司在美国所有的交易所都可以交易。比如，在NYSE挂牌的股票可以在NASDQ交易所交易，反之亦然。 |
| 2 | HKEX | 香港交易所的股票 |
| 3 | KRX | 韩国的股票 |

## API接口概要及架构

本API是一个基于Python的类库，通过使用和扩展类库提供的接口来实现交易所需的功能。鉴于国内期货界CTP接口的广泛使用，本API接口仿照CTP接口的方式，使得有CTP接口使用经验得用户能够快速上手开发。

本API中无论任何地方，对于以下的错误码，值为0时表示成功。

**接口函数分类：**

期货市场数据回调函数(期货行情需连接并登录期货的行情前置)

|  |  |
| --- | --- |
| onRtnSnapData() | 盘口(Order Book)快照数据推送 |
| onRtnDealData() | 成交数据推送 |
| onRtnStatisticsData() | 交易日开/高/低/结算价等数据推送 |
| onRtnTradeDate() | 未实现，保留 |

股票市场数据回调函数(股票行情需连接并登录股票的行情前置)

|  |  |
| --- | --- |
| onRtnMarketData() | 股票行情数据推送 |
| onRtnBrokerData() | 港股特有，经纪商队列推送 |

行情订阅函数及应答

|  |  |
| --- | --- |
| reqMarketData() | 订阅合约的函数 |
| onRspMarketData() | 订阅应答(只有期货市场有应答) |

## 全局的错误代码

1、业务函数中通用错误

//行情中的错误信息字典

dict error

{

ErrorID[int]:错误码

ErrorMsg[str]:错误描述

};

|  |  |  |
| --- | --- | --- |
| 错误代码  (ErrorID) | 错误描述 | 说明 |
| 0 | 成功 | 各种函数返回结果成功 |
| 00000 | 处理成功 |  |
| 10001 | 用户名不正确 |  |
| 10002 | 登录密码错误 |  |
| 10003 | 密码错误次数超限，用户已冻结，请联系客服解冻 |  |
| 10004 | 用户已被冻结，请联系客服解冻 |  |
| 10011 | 非法登录 |  |
| 10012 | 无登录权限 |  |
| 10013 | API认证错误 |  |
| 10014 | 没有这家股票交易所权限 |  |
| 10015 | 没有这家期货交易所权限 |  |
| 10016 | 不在白名单内 |  |
| 10017 | 登录次数受限 |  |
| 10018 | MarketType为空 |  |
| 10019 | 请等待上次订阅指令结束 |  |
| 10020 | 尚未登录 |  |
| 10021 | 发送-1错误 |  |
| 10022 | 发送0错误 |  |
| 10023 | 订阅类型和服务不匹配 |  |
| 10024 | 订阅格式不正确 |  |
| 10025 | 权限校验应答为空 |  |

1. 连接断开通用错误

onFrontDisconnected (int nReason)

|  |  |  |
| --- | --- | --- |
| 错误代码  (nReason) | 错误描述 | 说明 |
| 4001 | 设置阻塞失败 | Socket底层错误 |
| 4002 | 设置非阻塞失败 | Socket底层错误 |
| 4003 | 接收数据失败 | Socket底层错误 |
| 4004 | 接收0长度数据，已断开 | Socket底层错误 |
| 4005 | 发送数据失败 | Socket底层错误 |
| 4006 | 发送0长度数据，已断开 | Socket底层错误 |
| 4007 | 选择失败 | Socket底层错误 |
| 4008 | 服务器无响应 | Socket底层错误 |
|  |  |  |
|  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 业务类型 | 请求接口 | 请求应答响应接口 | | 回报接口 | |
| 连接 | init |  |  | onFrontConnected | 连接回报 |
| 断开 |  |  |  | onFrontDisconnected | 连接断开回报 |
| 心跳 | setHeartBeatTimeout |  |  | onHeartBeatWarning | 心跳警告 |
| 登录 | reqUserLogin | onRspUserLogin | 登录应答 |  |  |
| 行情订阅 | reqMarketData | onRspMarketData | 订阅应答 |  |  |
|  |  |  | onRtnMarketData | 行情推送 |
| 经纪商获取  (港股专有) | reqBrokerData |  |  | onRtnBrokerData | 经纪商队列回报 |
| 交易日推送 |  |  |  | onRtnTradeDate | 交易日推送 |

# 期货股票通用行情API接口参考

## 接口工作过程



## 接口模式

market\_demo.py文件中提供了一个MarketApi类，通过继承MarketApi类可以发出操作请求，也可以重载回调函数来处理后台服务的响应。

## 函数接口说明



## API实例对象创建

1. createMarketApi方法

创建API对象实例。

**函数原形：**

createMarketApi(bRecordLog: bool, pszFlowPath: str = “”,isFutureMode: str = “Y”)

**参数：**

bRecordLog：是否记录行情数据(行情数据文件会很大，通常建议关闭记录)

true:记录行情数据；false:不记录

pszFlowPath：日志数据文件路径

isFutureMode: Y:期货模式(缺省值);N: 股票模式

注：期货行情和股票行情，需连接并登录不通的行情前置。

当连接期货行情前置时，isFutureMode设置为Y；

当连接股票行情前置时，isFutureMode设置为N

## API函数请求与回调方法

### 1）onFrontConnected 方法

当客户端与前置建立起通信连接时（还未登录前），该方法被调用。

**函数原形：**

onFrontConnected()

本方法在完成初始化后调用，可以在其中完成用户登录任务。

### 2）onFrontDisconnected方法

当客户端与前置通信连接断开时，该方法被调用。当发生这个情况

后，API会自动重新连接，客户端可不做处理。自动重连地址，可能是原来注册

的地址，也可能是系统支持的其它可用的通信地址，它由程序自动选择。

**函数原形：**

onFrontDisconnected (nReason: int)

**参数：**

nReason：连接断开原因

### 3）onHeartBeatWarning方法

心跳超时警告。当长时间未收到报文时，该方法被调用。

**函数原形：**

onHeartBeatWarning(nTimeLapse: int)

**参数：**

nTimeLapse：距离上次接收报文的时间

SetHeartBeatTimeout

### 4）设置心跳超时时间setHeartBeatTimeout

函数原型：

|  |
| --- |
| setHeartBeatTimeout (iTimeout: int, bIsStrictCheck: bool = true) |

参数：iTimeout : 心跳超时时间，单位为秒。请设置心跳时间至少为30秒。

bIsStrictCheck ：是否检查心跳超时。默认为true，检查心跳；如果为false不检查心跳。

### 5）登录reqUserLogin/onRspUserLogin

函数原型：

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

参数：req

|  |
| --- |
| // 用户ID  UserId[str]  // 用户密码  UserPwd[str]  // 用户类型  UserType[str]  // 软件名称  SoftwareName[str]  // 软件版本号  SoftwareVersion[str]  // 授权码  AuthorCode[str]  // 错误信息  ErrorDescription[str]  //行情分发商ID(since v2.1.0.1)  BrokerIDForMarketPrice[str] |

函数原型：

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

参数：error

|  |
| --- |
| // 错误码  ErrorID[int]  // 错误描述  ErrorMsg[str] |

错误码 参考1.3

### 6）行情订阅请求/响应reqMarketData/onRspMarketData

函数原型：

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

参数：req

|  |
| --- |
| // 金融类型  MarketType[str]  // 请求类型  SubscMode[str]  // 交易所  ExchangeCode[str]  // 单次最大订阅数  MarketCount[int]  //单次订阅合约数组  MarketTrcode[list[str]]  // 错误信息  ErrorDescription[str] |
|  |

说明：

1. 按单个合约订阅行情。 交易所字段必输。比如

req[“ExchangeCode”] = "HKEX"

req[“MarketCount”] = 1

req[“MarketTrcodes”] = ["00001.HK"]

1. 第N次订阅调用响应结果(无论成功或失败)回来之后，才可以进行第N+1次订阅调用，
2. 第N次订阅调用在指定的最大时间1秒没有响应时，允许用户进行第N+1次订阅调用。
3. 订阅请求中的**ExchangeCode**设置特别说明
   1. 非CME group旗下交易所，请参照【直达软件会员行情API商品合约取得接口文档.docx】文档中”交易所列表查询”返回的交易所代码，或者直达客户端界面上看到的交易所代码。
   2. CME group旗下交易所，有4个，CME\_CBT，CME\_COMEX，CME\_NYMEX，CME，订阅时按品种所属的交易所代码进行填写。

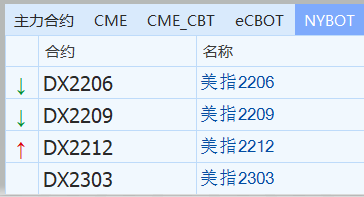
(例CME旗下NYMEX的原油CL品种，订阅时交易所代码使用CME\_NYMEX)

注意：订阅成功后推送的行情数据中的交易所代码，只有CME(包括CME,CME\_COMEX,CME\_NYMEX) 和 CME\_CBT。

1. 订阅请求中的**MarketTrcode**说明

可以参照【直达软件会员行情API商品合约取得接口文档.docx】文档中”合约列表查询”返回的合约代码，或者直达客户端界面上看到的交易所代码。

下图中第一行为直达系统中的交易所代码，第一列为合约代码



函数原型：

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

参数：req

|  |
| --- |
| // 金融类型  MarketType[str]  // 请求类型  SubscMode[str]  // 交易所  ExchangeCode[str] |

参数：error

|  |
| --- |
| // 错误码  ErrorID[int]  // 错误描述  TDAStringType [str] |

注：

用户每次调用**reqMarketData**订阅时，如果设置reqid的值为递增的序号，可以通过回调函数onRspMarketData的reqid确定订阅结果。

### 7）期货盘口行情推送onRtnSnapData

函数原型：

|  |
| --- |
| onRtnSnapData (data: dict) |

参数：data

|  |
| --- |
| // 交易所代码  ExchangeCode[str]  // 合约代码  TreatyCode[str]  // 买价  BuyPrice[float]  // 买量  BuyNumber[int]  // 卖价  SalePrice[float]  // 卖量  SaleNumber[int]  // 最新价  CurrPrice[float]  // 现量  CurrNumber[int]  // 行情时间unixepoch(精度到毫秒)  Time[int]  // 成交量  FilledNum[int]  // 持仓量  HoldNum[int]  // 买价2  BuyPrice2[float]  // 买价3  BuyPrice3[float]  // 买价4  BuyPrice4[float]  // 买价5  BuyPrice5[float]  // 买量2  BuyNumber2[int]  // 买量3  BuyNumber3[int]  // 买量4  BuyNumber4[int]  // 买量5  BuyNumber5[int]  // 卖价2  SalePrice2[float]  // 卖价3  SalePrice3[float]  // 卖价4  SalePrice4[float]  // 卖价5  SalePrice5[float]  // 卖量2  SaleNumber2[int]  // 卖量3  SaleNumber3[int]  // 卖量4  SaleNumber4[int]  // 卖量5  SaleNumber5[int]  // 隐藏买价  HideBuyPrice[float]  // 隐藏买量  HideBuyNumber[int]  // 隐藏卖价  HideSalePrice[float]  // 隐藏卖量  HideSaleNumber[int]  // 交易日  TradeDay[str]  // 买价6  BuyPrice6[float]  // 买价7  BuyPrice7[float]  // 买价8  BuyPrice8[float]  // 买价9  BuyPrice9[float]  // 买价10  BuyPrice10[float]  // 买量6  BuyNumber6[int]  // 买量7  BuyNumber7[int]  // 买量8  BuyNumber8[int]  // 买量9  BuyNumber9[int]  // 买量10  BuyNumber10[int]  // 卖价6  SalePrice6[float]  // 卖价7  SalePrice7[float]  // 卖价8  SalePrice8[float]  // 卖价9  SalePrice9[float]  // 卖价10  SalePrice10[float]  // 卖量6  SaleNumber6[int]  // 卖量7  SaleNumber7[int]  // 卖量8  SaleNumber8[int]  // 卖量9  SaleNumber9[int]  // 卖量10  SaleNumber10[int]  // 数据来源  DataSourceId[str] |

### 8）期货成交数据推送onRtnDealData

函数原型：

|  |
| --- |
| onRtnDealData (data: dict) |

参数：data

|  |
| --- |
| // 交易所代码  ExchangeCode[str]  // 合约代码  TreatyCode[str]  // 成交价  TradePrice[float]  // 成交量  TradeVol[int]  // 主动买或主动卖(未实现，保留)  AggressorSide[str]  // (未使用，保留)  TradeFlag[str]  // 时间戳unixepoch(精度到毫秒)  DateTimestamp[str]  // (未使用，保留)  Extends[str] |

### 9）期货统计数据数据推送onRtnStatisticsData

函数原型：

|  |
| --- |
| onRtnStatisticsData(data: dict) |

参数：data

|  |
| --- |
| // 交易所代码  ExchangeCode[str]  // 合约代码  TreatyCode[str]  // 最高价  HighPrice[float]  // 最低价  LowPrice[float]  // 开盘价  OpenPrice[float]  // 收盘价(未实现)  ClosePrice[float]  // 昨结算价  TySettlePrice[float]  // 实时结算价(交易所推送的结算价，实时反映)  IntradaySettlePrice[float]  // 涨停价(未实现，保留)  HighLimit[float]  // 跌停价(未实现，保留)  [float]LowLimit[float]  // 持仓  Position[int]  // 昨持仓  PrePosition[int]  // 开盘价Flag(未实现，保留)  OpenPrxBitFlag[str]  // 实时结算价Flag(未实现，保留)  SetlPrxBitFlag[str]  // 时间戳unixepoch(精度到毫秒)  DateTimestamp[int]  // 数据来源  DataSourceId[str] |
|  |
|  |

### 10）股票行情推送onRtnMarketData

函数原型：

|  |
| --- |
| onRtnMarketData(data: dict) |

参数：data

|  |
| --- |
| // 交易所代码  ExchangeCode[str]  // 合约代码  TreatyCode[str]  // 买价  BuyPrice[str]  // 买量  BuyNumber[str]  // 卖价  SalePrice[str]  // 卖量  SaleNumber[str]    // 最新价  CurrPrice[str]  // 现量  CurrNumber[str]  // 当天最高价  High[str]  // 当天最低价  Low[str]  // 开盘价  Open[str]  //收盘价  IntradaySettlePrice[str]  //股票：昨收盘价  Close[str]  // 行情时间  // 行情时间  Time[str]  // 成交量： 如果是股票行情，此字段为成交量  FilledNum[str]  // 持仓量:如果是股票行情，此字段是成交额  HoldNum[str]  // 买价2  BuyPrice2[str]  // 买价3  BuyPrice3[str]  // 买价4  BuyPrice4[str]  // 买价5  BuyPrice5[str]  // 买量2  BuyNumber2[str]  // 买量3  BuyNumber3[str]  // 买量4  BuyNumber4[str]  // 买量5  BuyNumber5[str]  // 卖价2  SalePrice2[str]  // 卖价3  SalePrice3[str]  // 卖价4  SalePrice4[str]  // 卖价5  SalePrice5[str]  // 卖量2  SaleNumber2[str]  // 卖量3  SaleNumber3[str]  // 卖量4  SaleNumber4[str]  // 卖量5  SaleNumber5[str]  // 隐藏买价  HideBuyPrice[str]  // 隐藏买量  HideBuyNumber[str]  // 隐藏卖价  HideSalePrice[str]  // 隐藏卖量  HideSaleNumber[str]  // 跌停价  LimitDownPrice[str]  // 涨停价  LimitUpPrice[str]  // 交易日  TradeDay[str]  // 买价6  BuyPrice6[str]  // 买价7  BuyPrice7[str]  // 买价8  BuyPrice8[str]  // 买价9  BuyPrice9[str]  // 买价10  BuyPrice10[str]  // 买量6  BuyNumber6[str]  // 买量7  BuyNumber7[str]  // 买量8  BuyNumber8[str]  // 买量9  BuyNumber9[str]  // 买量10  BuyNumber10[str]    // 卖价6  SalePrice6[str]  // 卖价7  SalePrice7[str]  // 卖价8  SalePrice8[str]  // 卖价9  SalePrice9[str]  // 卖价10  SalePrice10[str]  // 卖量6  SaleNumber6[str]  // 卖量7  SaleNumber7[str]  // 卖量8  SaleNumber8[str]  // 卖量9  SaleNumber9[str]  // 卖量10  SaleNumber10[str]  // 港交所股票行情：成交类型  TradeFlag[str]  // 交易所数据时间戳  DataTimestamp[str]  // 数据来源  DataSourceId[str]  // 可卖空股数（美股行情用）  CanSellVol[str]  // 行情区分，分两种情况(意思是Y和2中的成交量可以统计到分钟数据里，Z的不可以)  // 直连交易所的行情 -- Y: 当前回调数据是最新成交数据；Z: 当前回调数据是快照数据  // 非直连交易所的行情 -- 2：当前回调数据包括最新成交数据和盘口变化 Z: 快照数据中成交量// 不能用于统计  QuoteType[str] |

### 11）港股经纪商获取请求reqBrokerData/OnRtnBrokerData

函数原型：

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

参数：req

|  |
| --- |
| // 合约代码  HeyuCode[str]  // 错误信息  ErrorDescription[str] |

经纪商响应和回报都放在接口onR**tn**BrokerData中

### 12）港股经纪商推送onRtnBrokerData

函数原型：

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

参数：data

|  |
| --- |
| // 经济商数据  BrokerData[str] |

参数：error

|  |
| --- |
| // 错误码  ErrorID[int]  // 错误描述  ErrorMsg[str] |

备注：经纪商队列暂不提供

### 13）交易日推送onRtnTradeDate

函数原型：

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

参数：data

|  |
| --- |
| // 交易日期(yyyy-MM-dd)  TradeDate[str]  // 交易品种列表，格式为”GC,SI,HG”  TradeProduct[str] |

说明:

1. 只有期货有这个推送交易日功能。
2. 交易日推送功能，碰到节假日需要API用户自己调整，比如:

香港交易所2019/4/5日假日，恒指HSI在2019/ 4/4 17:00推送的交易日 是 2019-04-05,实际应该是2019-04-08

1. 每个品种按开盘时间提前15分钟推送。

参数：error

|  |
| --- |
| // 错误码  ErrorID[int]  // 错误描述  ErrorMsg[str] |

### 14）通用错误onRspError

函数原型：

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

参数：error

|  |
| --- |
| // 错误码  ErrorID[int]  // 错误描述  ErrorMsg[str] |

这个一般在行情数据回报的解析处理中有错误的话，会推送到此接口。

# 常见问题-股票行情相关

1、Question：

两个时间字段,一个是time,一个是DataTimestamp,这两个有什么区别？有的产品datatimestamp字段有数据，有的没有，这是什么情况？

-->Answer:

time是转换成中国本地时间

DataTimestamp是交易所发出来的数据时间戳，Unix epoch格式

直连交易所的品种，DataTimestamp会有数据，非直连的为空

2、Question：

收到的价格中，bidprice askprice 有时候为空

-->Answer:

请参照行情开发文档中的数据结构体中QuoteType字段的说明

// 行情区分，分两种情况(意思是Y和2中的成交量可以统计到分钟数据里，Z的不可以)

// 直连交易所的行情 -- Y: 当前回调数据是最新成交数据；Z: 当前回调数据是快照数据

// 非直连交易所的行情 -- 2：当前回调数据包括最新成交数据和盘口变化 Z: 快照数据中成交量// 不能用于统计