**国际期货股票API接口开发规范**

**V2.1**

**(2023.03)**

目录

[一、 变更履历 3](#_Toc132104853)

[一、 概览 4](#_Toc132104854)

[1.1. 背景 4](#_Toc132104855)

[1.2. API接口概要及架构 5](#_Toc132104856)

[1.3. 全局的错误代码 6](#_Toc132104857)

[二、 期货股票通用行情API接口参考 8](#_Toc132104858)

[1. 接口工作过程 8](#_Toc132104859)

[2. 接口模式 9](#_Toc132104860)

[3. 函数接口说明 10](#_Toc132104861)

[3.1. API实例对象创建 10](#_Toc132104867)

[3.2. API函数请求与回调方法 10](#_Toc132104868)

[1）OnFrontConnected 方法 10](#_Toc132104869)

[2）OnFrontDisconnected方法 11](#_Toc132104870)

[3）OnHeartBeatWarning方法 11](#_Toc132104871)

[4）设置心跳超时时间SetHeartBeatTimeout 11](#_Toc132104872)

[5）登录ReqUserLogin/OnRspUserLogin 12](#_Toc132104873)

[6）行情请求/响应ReqMarketData/OnRspMarketData 12](#_Toc132104874)

[7）行情推送OnRtnMarketData 15](#_Toc132104875)

[8）经纪商获取请求ReqBrokerData/OnRtnBrokerData 18](#_Toc132104876)

[9）经纪商推送OnRtnBrokerData 18](#_Toc132104877)

[10）交易日推送OnRtnTradeDate 19](#_Toc132104878)

[11）通用错误OnRspError 19](#_Toc132104879)

[三、 常见问题-行情相关 20](#_Toc132104880)

# 变更履历

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **变更内容** | **作者** | **日期** |
| V2.0.0.1 | 行情独立版本 | Kan.li | 2022/01/18 |
| V2.1.0.1 | 1. 登录调用时，增加行情broker ID   Chapter二、→3.2→ 2) CMarketReqUserLoginField结构体增加字段   1. 增加网络底层错误码说明   Chapter一、→1.3→ 2 | Rainer | 2022/03/03 |

# 概览

## 背景

本API可用来交易中国境外期货和股票，并且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C++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。

与ctp的主要不同：订阅和取消订阅请求合并为一个接口，根据入参类型来区分是订阅或取消；；由于外盘行情和内盘行情的不同，所以采用string作为主要类型。

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。

## 全局的错误代码

1、业务函数中通用错误

//行情中的错误信息结构体

struct CMarketRspInfoField

{

TDAIntType ErrorID; // 错误码

TDAStringType ErrorMsg; // 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

void 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

行情API 的DAMarketApi.h头文件中提供了两个接口，分别为CDAMarketDataApi和CDAMarketDataSpi，方便客户端应用程序的开发。客户端应用程序可以通过CDAMarketDataApi实例发出操作请求，通过继承CDAMarketDataSpi并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. CDAMarketDataApi::CreateMarketApi方法

创建API对象实例。

**函数原形：**

CDAMarketDataApi\* CDAMarketDataApi::CreateMarketApi

(bool bRecordLog, const char \*pszFlowPath )

**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

## 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：**

void 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：**

void OnFrontDisconnected (int nReason)；

**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：**

void OnHeartBeatWarning(int nTimeLapse)；

**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：

|  |
| --- |
| bool SetHeartBeatTimeout (int iTimeout, bool bIsStrictCheck = true) ; |

参数：iTimeout : 心跳超时时间，单位为毫秒。

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。

注意：推荐设置为true，设置为false主要为适配兼容行情前置的老版本，因为有些行情前置不应答心跳。

### 5）登录ReqUserLogin/OnRspUserLogin

函数原型：

|  |
| --- |
| bool ReqUserLogin( CMarketReqUserLoginField \*pReqUserLoginField, int iRequestID) ; |

参数：CMarketReqUserLoginField

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

函数原型：

|  |
| --- |
| void OnRspUserLogin( CMarketRspInfoField \*pRspInfo, int iRequestID, bool bIsLast) {}; |

参数：CMarketRspInfoField

|  |
| --- |
| // 错误码  TDAIntType ErrorID;  // 错误描述  TDAStringType ErrorMsg; |

错误码 参考1.3

### 6）行情请求/响应ReqMarketData/OnRspMarketData

函数原型：

|  |
| --- |
| bool ReqMarketData(CMarketReqMarketDataField\* pReqMarketDataField, int iRequestID, bool bIsLast) = 0 |

参数：CMarketReqMarketDataField

|  |
| --- |
| // 金融类型  TDACharType MarketType;  // 请求类型  TDACharType SubscMode;  // 交易所  TDAStringType ExchangeCode;  // 单次最大订阅数  TDAIntType MarketCount;  //单次订阅合约数组  TDAStringType MarketTrcode[MAX\_SUB\_COUNT];  // 错误信息  TDAStringType ErrorDescription; |
|  |

说明：

1. 按单个合约订阅行情。 交易所字段必输。比如

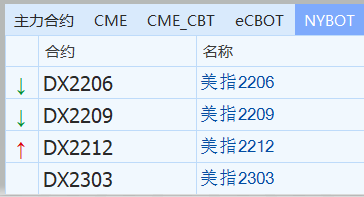
safe\_cpy(qry.ExchangeCode, "HKEX");

qry.MarketCount = 1;

const char cont0[] = "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. 订阅行情时，使用的交易所代码，合约代码以直达客户端界面上展示的代码为准。

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：

|  |
| --- |
| void OnRspMarketData(CMarketRspMarketDataField\* pRspMarketDataField, CMarketRspInfoField \*pRspInfo, int iRequestID, bool bIsLast) {} |

参数：CMarketRspMarketDataField

|  |
| --- |
| // 金融类型  TDACharType MarketType;  // 请求类型  TDACharType SubscMode;  // 交易所  TDAStringType ExchangeCode; |

参数：CMarketRspInfoField

|  |
| --- |
| // 错误码  TDAIntType ErrorID;  // 错误描述  TDAStringType ErrorMsg; |

### 7）行情推送OnRtnMarketData

函数原型：

|  |
| --- |
| bool RtnMarketData(CMarketRtnMarketDataField \* pReqMarketDataField) = 0 |

参数：CMarketRtnMarketDataField

|  |
| --- |
| // 交易所代码  TDAStringType ExchangeCode;  // 合约代码  TDAStringType TreatyCode;  // 买价  TDAStringType BuyPrice;  // 买量  TDAStringType BuyNumber;  // 卖价  TDAStringType SalePrice;  // 卖量  TDAStringType SaleNumber;  // 最新价  TDAStringType CurrPrice;  // 现量  TDAStringType CurrNumber;  // 当天最高价  TDAStringType High;  // 当天最低价  TDAStringType Low;  // 开盘价  TDAStringType Open;  //盘中结算价(股票：收盘价)，CME交易所在交易盘中会推出当前交易日结算价，亚洲的交易所是在收盘 (T session) 后推出结算价(通过最后几分钟竞价得到)  TDAStringType ~~OldClose~~ 🡪 IntradaySettlePrice;  //当前交易日的前结算（股票：昨收盘价）  TDAStringType PreSettlementPrice;  // 废弃字段  TDAStringType Close;  // 行情时间  TDAStringType Time;  // 成交量： 如果是股票行情，此字段为成交量  TDAStringType FilledNum;  // 持仓量:如果是股票行情，此字段是成交额  TDAStringType HoldNum;  // 买价2  TDAStringType BuyPrice2;  // 买价3  TDAStringType BuyPrice3;  // 买价4  TDAStringType BuyPrice4;  // 买价5  TDAStringType BuyPrice5;  // 买量2  TDAStringType BuyNumber2;  // 买量3  TDAStringType BuyNumber3;  // 买量4  TDAStringType BuyNumber4;  // 买量5  TDAStringType BuyNumber5;  // 卖价2  TDAStringType SalePrice2;  // 卖价3  TDAStringType SalePrice3;  // 卖价4  TDAStringType SalePrice4;  // 卖价5  TDAStringType SalePrice5;  // 卖量2  TDAStringType SaleNumber2;  // 卖量3  TDAStringType SaleNumber3;  // 卖量4  TDAStringType SaleNumber4;  // 卖量5  TDAStringType SaleNumber5;  // 隐藏买价  TDAStringType HideBuyPrice;  // 隐藏买量  TDAStringType HideBuyNumber;  // 隐藏卖价  TDAStringType HideSalePrice;  // 隐藏卖量  TDAStringType HideSaleNumber;  // 跌停价  TDAStringType LimitDownPrice;  // 涨停价  TDAStringType LimitUpPrice;  // 交易日  TDAStringType TradeDay;  // 买价6  TDAStringType BuyPrice6;  // 买价7  TDAStringType BuyPrice7;  // 买价8  TDAStringType BuyPrice8;  // 买价9  TDAStringType BuyPrice9;  // 买价10  TDAStringType BuyPrice10;  // 买量6  TDAStringType BuyNumber6;  // 买量7  TDAStringType BuyNumber7;  // 买量8  TDAStringType BuyNumber8;  // 买量9  TDAStringType BuyNumber9;  // 买量10  TDAStringType BuyNumber10;  // 卖价6  TDAStringType SalePrice6;  // 卖价7  TDAStringType SalePrice7;  // 卖价8  TDAStringType SalePrice8;  // 卖价9  TDAStringType SalePrice9;  // 卖价10  TDAStringType SalePrice10;  // 卖量6  TDAStringType SaleNumber6;  // 卖量7  TDAStringType SaleNumber7;  // 卖量8  TDAStringType SaleNumber8;  // 卖量9  TDAStringType SaleNumber9;  // 卖量10  TDAStringType SaleNumber10;  // 港交所股票行情：成交类型  TDAStringType TradeFlag;  // 交易所数据时间戳  TDAStringType DataTimestamp;  // 数据来源  TDAStringType DataSourceId;  // 可卖空股数（美股行情用）  TDAStringType CanSellVol;  // 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  TDAStringType QuoteType; |

### 8）经纪商获取请求ReqBrokerData/OnRtnBrokerData

函数原型：

|  |
| --- |
| bool ReqBrokerData(CMarketReqBrokerDataField\* pReqBrokerDataField, int iRequestID); |

参数：CMarketReqBrokerDataField

|  |
| --- |
| // 合约代码  TDAStringType HeyuCode;  // 错误信息  TDAStringType ErrorDescription; |

经纪商响应和回报都放在接口OnR**tn**BrokerData中

### 9）经纪商推送OnRtnBrokerData

函数原型：

|  |
| --- |
| void OnRtnBrokerData(CMarketRtnBrokerDataField \*pRtnBrokerData, CMarketRspInfoField \*pRspInfo, int iRequestID, bool bIsLast) |

参数：CMarketRspBrokerDataField

|  |
| --- |
| // 经济商数据  TDABrokerType BrokerData; |

参数：CMarketRspInfoField

|  |
| --- |
| // 错误码  TDAIntType ErrorID;  // 错误描述  TDAStringType ErrorMsg; |

### 10）交易日推送OnRtnTradeDate

函数原型：

|  |
| --- |
| void OnRtnTradeDate(CMarketRtnTradeDateField \*pRtnTradeDate, CMarketRspInfoField \*pRspInfo, int iRequestID, bool bIsLast) |

参数：CMarketRtnTradeDateField

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

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：CMarketRspInfoField

|  |
| --- |
| // 错误码  TDAIntType ErrorID;  // 错误描述  TDAStringType ErrorMsg; |

### 11）通用错误OnRspError

函数原型：

|  |
| --- |
| void OnRspError(CMarketRspInfoField \*pRspInfo, int nRequestID, bool bIsLast) |

参数：CMarketRspInfoField

|  |
| --- |
| // 错误码  TDAIntType ErrorID;  // 错误描述  TDAStringType ErrorMsg; |

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-行情相关

1、Question：

有个zd market的问题请教一下。在返回的CMarketRspMarketDataField的数据结构中，

有两个时间field,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

TDAStringType QuoteType;