证券交易托管系统

交易员应用程序接口

2016 年 1 月

1. 文件属性

文件属性	内容
文件名称	证券交易托管系统_ TradeAPI 接口
文件编号	
文件版本号	V0.3
文件状态	草稿
作　　者	上海期货信息技术有限公司
文档编写日期	2016-1-27
文档发布日期	

2. 文件变更历史清单

文件版本号	修正日期	修正人	备　　注
V0.1	2015-4-8	桂荣盛	创建
V0.2	2015-5-21	桂荣盛	增加个股期权接口说明
V0.3	2016-1-27	包路跃	增加分级基金、融资融券接口说明

3. 本次修改变更说明

序号	变更内容简述
1.	增加分级基金、融资融券接口说明
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	

目 录

1. 介绍

交易托管系统 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相关交易功能，包括报单与报价的录入、报单与报价的撤销、报单与报价的挂起、报单与报价的激活、报单与报价的修改、报单与报价的查询、成交单查询、投资者查询、投资者持仓查询、合约查询、交易日获取等。该类库包含以下 5 个文件：

文件名	版本	文件大小	文件描述
ZQThostFtdcTraderApi.h	V1.0	26kb	交易接口头文件
ZQThostFtdcUserApiStruct.h	V1.0	94kb	定义了 API 所需的一系列数据类型的头文件
ZQThostFtdcUserApiDataType.h	V1.0	80kb	定义了一系列业务相关的数据结构的头文件
zqthosttraderapi.dll	V1.0	1225kb	动态链接库二进制文件
zqthosttraderapi.lib	V1.0	4kb	导入库文件
zqthostmduserapi.dll	V1.0	1225kb	动态链接库二进制文件
zqthostmduserapi.lib	V1.0	4kb	导入库文件

支持 MS VC 6.0，MS VC.NET 2003 编译器。需要打开多线程编译选项/MT。

2. 体系结构

交易员 API 使用建立在 TCP 协议之上 FTD 协议与交易托管系统进行通讯，交易托管系统负责投资者的交易业务处理。

2.1. 通讯模式

FTD 协议中的所有通讯都基于某个通讯模式。通讯模式实际上就是通讯双方协同工作的方式。

FTD 涉及的通讯模式共有三种：

- 对话通讯模式
- 私有通讯模式
- 广播通讯模式

对话通讯模式是指由会员端主动发起的通讯请求。该请求被交易所端接收和处理，并给予响应。例如报单、查询等。这种通讯模式与普通的客户/服务器模式相同。

私有通讯模式是指交易所端主动，向某个特定的会员发出的信息。例如成交回报等。

广播通讯模式是指交易所端主动，向市场中的所有会员都发出相同的信息。例如公告、市场公共信息等。

通讯模式和网络的连接不一定存在简单的一对一的关系。也就是说，一个网络连接中可能传送多种不同通讯模式的报文，一种通讯模式的报文也可以在多个不同的连接中传送。

无论哪种通讯模式，其通讯过程都如图 1 所示：

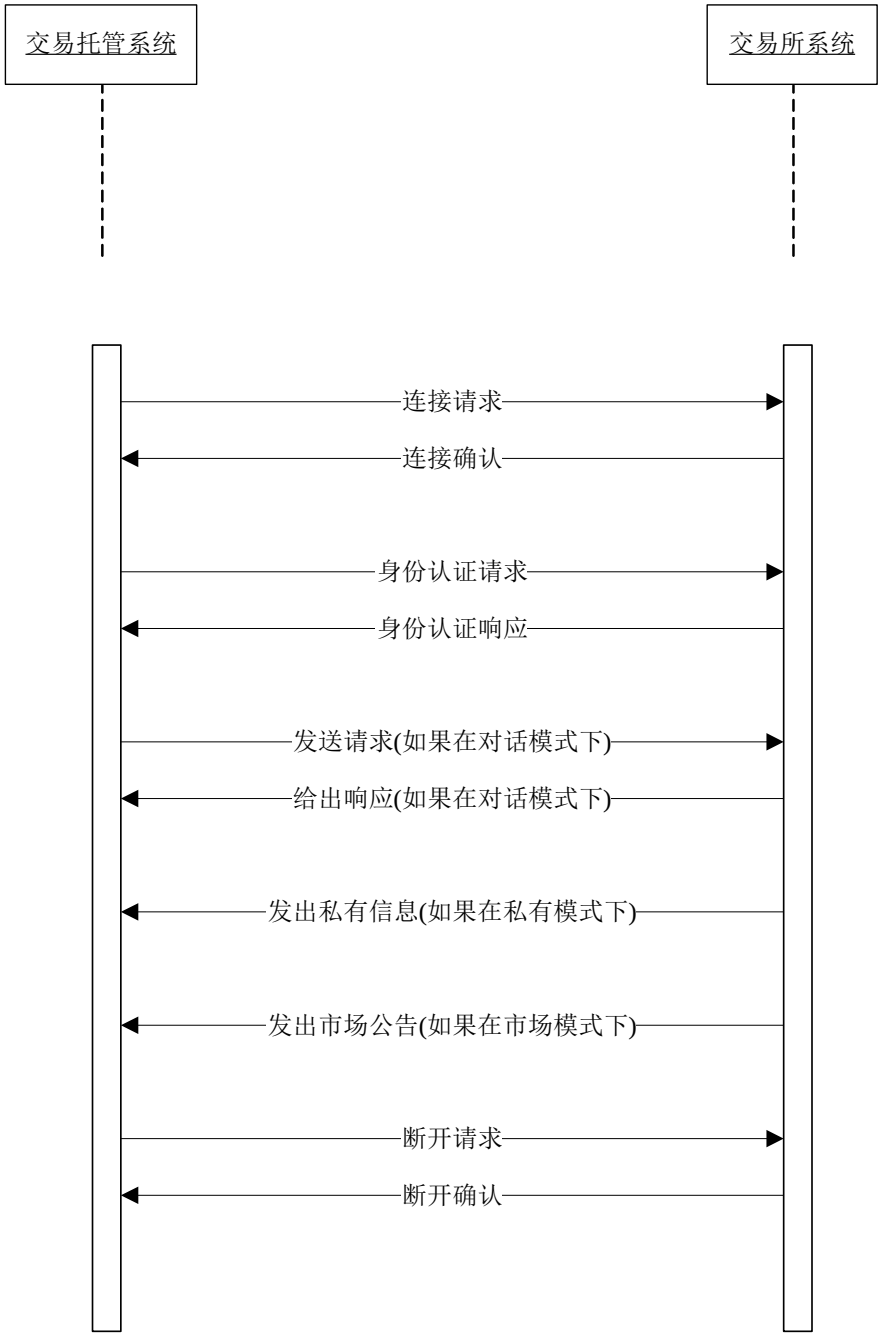


图1) 各通讯模式的工作过程

本接口暂时没有使用广播通信方式。

2.2. 数据流

交易托管系统支持对话通讯模式、私有通讯模式、广播通讯模式：

对话通讯模式下支持对话数据流和查询数据流：

对话数据流是一个双向数据流，交易托管系统发送交易请求，交易系统反馈应答。交易系统不维护对话流的状态。系统故障时，对话数据流会重置，通讯途中的数据可能会丢失。

查询数据流是一个双向数据流，交易托管系统发送查询请求，交易系统反馈应答。交易系统不维护查询流的状态。系统故障时，查询数据流会重置，通讯途中的数据可能会丢失。

私有通讯模式下支持私有数据流：

私有流是一个单向数据流，由交易系统发向交易托管系统，用于传送交易员私有的通知和回报信息。私有流是一个可靠的数据流，交易系统维护每个交易托管系统的私有流，在一个交易日内，交易托管系统断线后恢复连接时，可以请求交易系统发送指定序号之后的私有流数据。私有数据流向交易托管系统提供报单状态报告、成交回报更等信息。

广播通讯模式下支持公共数据流：

公共数据流是一个单向数据流，由交易系统发向交易托管系统，用于发送市场公共信息；公共数据流也是一个可靠的数据流，交易系统维护整个系统的公共数据流，在一个交易日内，交易托管系统断线恢复连接时，可以请求交易系统发送指定序号之后的公共数据流数据。

3. 接口模式

证券交易员 API 提供了二个接口，分别为 CZQThostFtdcTraderApi 和 CZQThostFtdcTraderSpi。这两个接口对 FTD 协议进行了封装，方便客户端应用程序的开发。

客户端应用程序可以通过 CZQThostFtdcTraderApi 发出操作请求，通继承 CZQThostFtdcTraderSpi 并重载回调函数来处理后台服务的响应。

3.1. 对话流和查询流编程接口

通过对话流进行通讯的编程接口通常如下：

请求：int CZQThostFtdcTraderApi::ReqXXX(CZQThostFtdcXXXField *pReqXXX, int nRequestID)

响应：

void CZQThostFtdcTraderSpi::OnRspXXX(CZQThostFtdcXXXField *pRspXXX, CZQThostFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)

其中请求接口第一个参数为请求的内容，不能为空。

第二个参数为请求号。请求号由客户端应用程序负责维护，正常情况下每个请求的请求号不要重复。在接收交易托管系统的响应时，可以得到当时发出请求时填写的请求号，从而可以将响应与请求对应起来。

当收到后台服务应答时，CZQThostFtdcTraderSpi 的回调函数会被调用。如果响应数据不止一个，则回调函数会被多次调用。

回调函数的第一个参数为响应的具体数据，如果出错或没有结果有可能为 NULL。

第二个参数为处理结果，表明本次请求的处理结果是成功还是失败。在发生多次回调时，除了第一次回调，其它的回调该参数都可能为 NULL。

第三个参数为请求号，即原来发出请求时填写的请求号。

第四个参数为响应结束标志，表明是否是本次响应的最后一次回调。

3.2. 私有流编程接口

私有流中的数据中会员的私有信息，包括报单回报、成交回报等。

通过私有流接收回报的编程接口通常如下：

void CZQThostFtdcTraderSpi::OnRtnXXX(CZQThostFtdcXXXField *pXXX)

或

void CZQThostFtdcTraderSpi::OnErrRtnXXX(CZQThostFtdcXXXField *pXXX, CZQThostFtdcRspInfoField *pRspInfo)

当收到交易托管系统通过私有流发布的回报数据时，CZQThostFtdcTraderSpi 的回调函数会被调用。回调函数的参数为回报的具体内容。

4. 运行模式

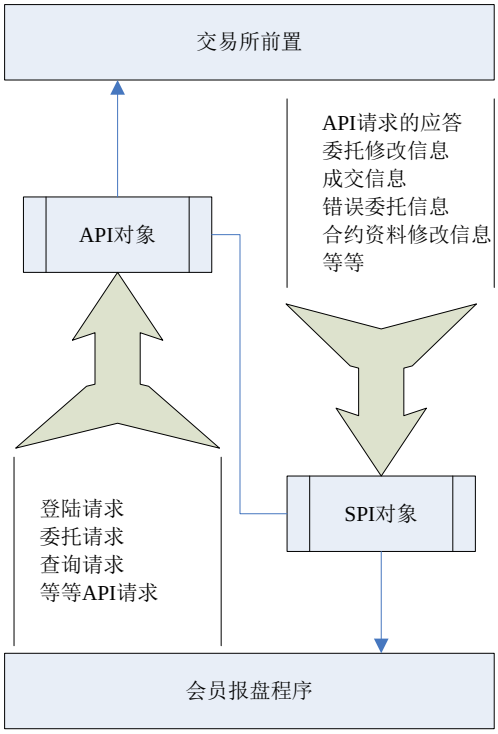
4.1. 工作线程

交易员客户端应用程序至少由两个线程组成，一个是应用程序主线程，一个是交易员 API 工作线程。应用程序与交易系统的通讯是由 API 工作线程驱动的。

CZQThostFtdcTraderApi 提供的接口是线程安全的，可以有多个应用程序线程同时发出请求。

CZQThostFtdcTraderSpi 提供的接口回调是由 API 工作线程驱动，通过实现 SPI 中的接口方法，可以从交易托管系统收取所需数据。

如果重载的某个回调函数阻塞，则等于阻塞了 API 工作线程，API 与交易系统的通讯会停止。因此，在 CZQThostFtdcTraderSpi 派生类的回调函数中，通常应迅速返回，可以利用将数据放入缓冲区或通过 Windows 的消息机制来实现。



4.2. 本地文件

交易员 API 在运行过程中，会将一些数据写入本地文件中。调用 CreateFtdcTraderApi 函数，可以传递一个参数，指明存贮本地文件的路径。该路径必须在运行前已创建好。本地文件的扩展名都是".con"。

5. 业务与接口对照

业务类型	业务	请求接口	响应接口	数据流
初始化	创建 API 实例	CreateApi	N/A	
	初始化执行	Init	N/A	
	线程等待	Join	N/A	
	注册 Spi	RegisterSpi	N/A	
	注册前置	RegisterFront	N/A	
信息订阅	订阅私有流	SubscribePrivateTopic	N/A	
	订阅公有流	SubscribePublicTopic	N/A	
信息通知	前置连接通知	N/A	OnFrontConnected	
	前置断开通知	N/A	OnFrontDisconnected	
	心跳超时警告	N/A	OnHeartBeatWarning	
	错误应答	N/A	OnRspError	公有流
版本信息	获取 API 版本号	GetApiVersion	N/A	

5.1. TraderAPI

提供给交易员使用，用于交易及进行相关查询。

业务类型	业务	请求接口	响应接口	数据流
登录	客户端认证	ReqAuthenticate	OnRspAuthenticate	对话流
	登录	ReqUserLogin	OnRspUserLogin	对话流
	登出	ReqUserLogout	OnRspUserLogout	对话流
	用户口令更新	ReqUserPasswordUpdate	OnRspUserPasswordUpdate	对话流
报单	报单录入	ReqOrderInsert	OnRspOrderInsert	对话流
	报单通知	N/A	OnRtnOrder	私有流
	报单录入错误通知	N/A	OnErrRtnOrderInsert	私有流
	成交通知	N/A	OnRtnTrade	私有流
报单操作	报单操作	ReqOrderAction	OnRspOrderAction	对话流
	报单操作错误通知	N/A	OnErrRtnOrderAction	私有流
资金转移	资金转移通知	N/A	OnRtnTransferFund	私有流
仓位转移	仓位转移通知	N/A	OnRtnTransferPosition	私有流
现券转移	现券转移	ReqSecurityTransfer	OnRspSecurityTransfer	对话流
	现券转移通知	N/A	OnRtnSecurityTransfer	私有流
	转移错误通知	N/A	OnErrRtnSecurityTransfer	私有流
证券锁定	锁定录入	ReqLockInsert	OnRspLockInsert	对话流
	锁定通知	N/A	OnRtnLock	私有流

	锁定录入错误通知	N/A	OnErrRtnLockInsert	私有流
执行宣告	执行宣告录入	ReqExecOrderInsert	OnRspExecOrderInsert	对话流
	执行宣告通知	N/A	OnRtnExecOrder	私有流
	执行宣告录入错误通知	N/A	OnErrRtnExecOrderInsert	私有流
执行宣告操作	执行宣告操作	ReqExecOrderAction	OnRspExecOrderAction	对话流
	执行宣告操作错误通知	N/A	OnErrRtnExecOrderAction	私有流
直接还款	直接还款录入	ReqCashRepayInsert	OnRspCashRepayInsert	对话流
	直接还款通知	N/A	OnRtnCashRepay	私有流
	直接还款录入错误通知	N/A	OnErrRtnCashRepayInsert	私有流
查询	交易所查询	ReqQryExchange	OnRspQryExchange	查询流
	行情查询	ReqQryDepthMarketData	OnRspQryDepthMarketData	查询流
	证券信息查询	ReqQrySecurity	OnRspQrySecurity	查询流
	合约信息查询	ReqQryInstrument	OnRspQryInstrument	查询流
	上市开放式基金查询	ReqQryLOFInfo	OnRspQryLOFInfo	查询流
	上市开放式基金子基金查询	ReqQryLOFSubInfo	OnRspQryLOFSubInfo	查询流
	市场信息查询	ReqQryMarket	OnRspQryMarket	查询流
	证券品种查询	ReqQrySecurityProduct	OnRspQrySecurityProduct	查询流
	证券类别查询	ReqQrySecurityClass	OnRspQrySecurityClass	查询流
	报单查询	ReqQryOrder	OnRspQryOrder	查询流
	锁定查询	ReqQryLock	OnRspQryLock	查询流
	执行宣告查询	ReqQryExecOrder	OnRspQryExecOrder	查询流
	报单资金明细查询	ReqQryOrderFundDetail	OnRspQryOrderFundDetail	查询流
	成交查询	ReqQryTrade	OnRspQryTrade	查询流
	投资者查询	ReqQryInvestor	OnRspQryInvestor	查询流
	投资者交易账户查询	ReqQryTradingCode	OnRspQryTradingCode	查询流
	投资单元查询	ReqQryInvestUnit	OnRspQryInvestUnit	查询流
	投资者资金查询	ReqQryTradingAccount	OnRspQryTradingAccount	查询流
	投资者证券持仓查询	ReqQryInvestorSecurityPosition	OnRspQryInvestorSecurityPosition	查询流
	投资者合约持仓查询	ReqQryInvestorPosition	OnRspQryInvestorPosition	查询流
	锁定持仓查询	ReqQryLockPosition	OnRspQryLockPosition	查询流
	投资者佣金查询	ReqQryBrokerage	OnRspQryBrokerage	查询流

	新股申购额度查询	ReqQrySubscribingSharesQuota	OnRspQrySubscribingSharesQuota	查询流
	新股信息查询	ReqQrySubscribingShares	OnRspQrySubscribingShares	查询流
	质押申报代码查询	ReqQryPledgeOrderInfo	OnRspQryPledgeOrderInfo	查询流
	质押持仓查询	ReqQryInvestorPledgePosition	OnRspQryInvestorPledgePosition	查询流
	质押回购信息查询	ReqQryInvestorPledgeInfo	OnRspQryInvestorPledgeInfo	查询流
	标准券使用率查询	ReqQryRepoRatio	OnRspQryRepoRatio	查询流
	回购放大倍数查询	ReqQryRepurchaseMaxTimes	OnRspQryRepurchaseMaxTimes	查询流
	ETF 清单查询	ReqQryETFFile	OnRspQryETFFile	查询流
	ETF 成分证券查询	ReqQryETFBasket	OnRspQryETFBasket	查询流
	资金转移查询	ReqQryTransferFund	OnRspQryTransferFund	查询流
	仓位转移查询	ReqQryTransferPosition	OnRspQryTransferPosition	查询流
	现券转移查询	ReqQrySecurityTransfer	OnRspQrySecurityTransfer	查询流
	投资单元与交易资金账户关系查询	ReqQryInvestUnitAndTradingAcct	OnRspQryInvestUnitAndTradingAcct	查询流
	投资单元与交易用户关系查询	ReqQryInvestUnitAndUser	OnRspQryInvestUnitAndUser	查询流
	投资者直接还款查询	ReqQryCashRepay	OnRspQryCashRepay	查询流
	投资者融资融券信息查询	ReqQryCreditInfo	OnRspQryCreditInfo	查询流
	投资者融资融券标的证券查询	ReqQryCreditUnderlying	OnRspQryCreditUnderlying	查询流
	投资者融资融券担保证券查询	ReqQryCreditCollateral	OnRspQryCreditCollateral	查询流
	投资者融资融券可用证券头寸查询	ReqQryCreditSecurityPosition	OnRspQryCreditSecurityPosition	查询流
	投资者融资融券可用资金头寸查询	ReqQryCreditFundPosition	OnRspQryCreditFundPosition	查询流
	投资者信用合约查询	ReqQryCreditContract	OnRspQryCreditContract	查询流

5.2. MarketDataAPI

MarketDataAPI 用于为投资者提供行情信息。

业务类型	业务	请求接口	响应接口	数据流
登录	登录	ReqUserLogin	OnRspUserLogin	对话流
	登出	ReqUserLogout	OnRspUserLogout	对话流
行情管理	订阅行情	SubscribeMarketData	OnRspSubMarketData	对话流
	取消订阅行情	UnsubscribeMarketData	OnRspUnSubMarketData	对话流
	深度行情通知	N/A	OnRtnDepthMarketData	N/A

注：行情通知的返回方式比较特殊。它不属于通常意义上的流方式，而是从订阅 Session 直接下传数据。类似于查询流的工作方式，但又不像查询流那样即时返回。
鉴于此，行情 API 的使用方式也与其它 API 有所区别。它无需订阅私有流或公有流，在注册完 Spi 和 Front 后可以直接 Init 初始化，然后根据需要 Login 并 SubscribeMarketData 即可。

6. 开发接口

6.1. 通用规则

客户端和交易托管系统的通讯过程分为 2 个阶段：初始化阶段和功能调用阶段。

在初始化阶段，程序必须完成如下步骤（具体代码请参考开发实例）：

1, 产生一个 CZQThostFtdcTraderApi 实例
2, 产生一个事件处理的实例
3, 注册一个事件处理的实例
4, 订阅私有流
5, 订阅公共流
6, 设置交易托管服务的地址

在功能调用阶段，程序可以任意调用交易接口中的请求方法，如 ReqOrderInsert 等。同时按照需要响应回调接口中的。

其他注意事项：

1、 API 请求的返回参数，0 表示正确，其他表示错误，详细错误编码请查表。
2、 输入参数的字段初始化，可以使用 ZQThostFtdcUserApiDataType.h 中的 set_null 。
3、 对于返回结果的处理，使用 ZQThostFtdcUserApiDataType.h 中的 is_null 函数来判断是否是空值。

6.2. 托管服务地址设置要求

客户端需要注册地址列表中的所有地址，ＡＰＩ会根据具体情况自动选择一个合适的主机进行连接。

6.3. CZQThostFtdcTraderSpi 接口

CZQThostFtdcTraderSpi 实现了事件通知接口。用户必需派生 CZQThostFtdcTraderSpi 接口，编写事件处理方法来处理感兴趣的事件。

6.3.1. OnFrontConnected 方法

当客户端与交易托管系统建立起通信连接时（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。

6.3.2. OnFrontDisconnected 方法

当客户端与交易托管系统通信连接断开时，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

0x1001 网络读失败

0x1002 网络写失败

0x2001 接收心跳超时

0x2002 发送心跳失败

0x2003 收到错误报文

6.3.3. 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

6.3.4. OnRspAuthenticate 方法

当客户端发出认证请求之后，交易托管系统返回响应时，该方法会被调用，通知客户端认证是否成功。

函数原形：


```
void OnRspAuthenticate(
                        CZQThostFtdcRspAuthenticateField *pRspAuthenticateField,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast);
```

参数：

pRspAuthenticateField：返回客户端认证信息的地址。

客户端认证信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
UserID	用户代码	
UserProductInfo	用户端产品信息	

pRspInfo：返回响应信息的地址。

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回客户端认证请求的 ID，该 ID 由用户在登录时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.5. OnRspUserLogin 方法

当客户端发出登录请求之后，交易托管系统返回响应时，该方法会被调用，通知客户端登录是否成功。

函数原形：

```
void OnRspUserLogin(
                        CZQThostFtdcRspUserLoginField *pRspUserLogin,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast);
```


参数：

pRspUserLogin：返回用户登录信息的地址。

用户登录信息结构：

字段名称	属性	描述
LoginTime	登录成功时间	
BrokerID	经纪公司代码	
UserID	用户代码	
SystemName	交易系统名称	
FrontID	前置编号	
SessionID	会话编号	
MaxOrderRef	最大报单引用	

pRspInfo：返回用户响应信息的地址。特别注意在有连续的成功的响应数据时，中间有可能返回 NULL，但第一次不会，以下同。错误代码为 0 时，表示操作成功，以下同。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回用户登录请求的 ID，该 ID 由用户在登录时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.6. OnRspUserLogout 方法

当客户端发出退出请求之后，交易托管系统返回响应时，该方法会被调用，通知客户端退出是否成功。

函数原形：

```
void OnRspUserLogout(
                        CZQThostFtdcUserLogoutField *pUserLogout,
```

CZQThostFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast);

参数：

pRspUserLogout：返回用户退出信息的地址。

用户登出信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
UserID	用户代码	

pRspInfo：返回用户响应信息的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回用户登出请求的 ID，该 ID 由用户在登出时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.7. OnRspUserPasswordUpdate 方法

用户密码修改应答。当客户端发出用户密码修改指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspUserPasswordUpdate(
                    CZQThostFtdcUserPasswordUpdateField
                    *pUserPasswordUpdate,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast);
```

参数：

pUserPasswordUpdate：指向用户密码修改结构的地址，包含了用户密码修

改请求的输入数据。

用户密码修改结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
UserID	用户代码	
OldPassword	原来的口令	
NewPassword	新的口令	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回用户密码修改请求的 ID，该 ID 由用户在密码修改时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.8. OnRspQryMarginRate 方法

投资者保证金率查询应答。当客户端发出查询投资者保证金率指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryMarginRate(
                        CZQThostFtdcMarginRateField *pMarginRate,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast) ;
```

参数：

pMarginRate：指向投资者保证金率查询结构的地址，包含了用户密码修改请求的输入数据。

投资者保证金率查询响应

字段名称	属性	描述

BrokerID	经纪公司代码	
MarginTemplateNo	保证金模板代码	
ExchangeID	交易所代码	
InstrumentID	合约代码	
LongMarginRatioByMoney	多头保证金率	暂不使用
LongMarginRatioByVolume	多头保证金费	暂不使用
ShortMarginRatioByMoney	空头保证金率	暂不使用
ShortMarginRatioByVolume	空头保证金费	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回用户密码修改请求的 ID，该 ID 由用户在密码修改时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.9. OnRspError 方法

针对用户请求的出错通知。

函数原形：

```
void OnRspError(
                CZQThostFtdcRspInfoField *pRspInfo,
                int nRequestID,
                bool bIsLast)
```

参数：

pRspInfo：返回用户响应信息的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
```

```
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回用户操作请求的 ID，该 ID 由用户在操作请求时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.10. OnRspOrderInsert 方法

报单录入应答。当客户端发出过报单录入指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspOrderInsert(
                        CZQThostFtdcInputOrderField *pInputOrder,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast);
```

参数：

pInputOrder：指向报单录入结构的地址，包含了提交报单录入时的输入数据，和后台返回的报单编号。

输入报单结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
OrderRef	报单引用	
UserID	用户代码	证券暂不使用
OrderPriceType	报单价格条件	1=任意价 2=限价 3=最优价 4=最新价 5=最新价浮动上浮 1 个 ticks …… F=买一价浮动上浮 3 个 ticks G=五档价 a=本方最优
Direction	买卖方向	0=买入

		1=卖出 2=ETF 申购 3=ETF 赎回 4=新股申购 5=正回购 6=逆回购 7=转托管 8=开放式基金申购 9=开放式基金赎回 d=质押入库 e=质押出库 f=配股配债 g=开放式基金拆分 h=开放式基金合并
CombOffsetFlag	组合开平标志	第一位是开平标志： 0=开仓 1=平仓 2=强平 3=平今 4=平昨 5=强减 6=本地强平
CombHedgeFlag	组合投机套保标志	第一位是投机套保标志：（除备兑外都写投机） 1=投机 2=套利 3=套保 4=备兑
LimitPrice	价格	
VolumeTotalOriginal	数量	
TimeCondition	有效期类型	1=立即完成，否则撤销 2=本节有效 3=当日有效 4=指定日期前有效 5=撤销前有效 6=集合竞价有效
GTDDate	GTD 日期	证券暂不使用
VolumeCondition	成交量类型	1=任何数量 2=最小数量 3=全部数量
MinVolume	最小成交量	证券暂不使用
ContingentCondition	触发条件	证券暂不使用
StopPrice	止损价	证券暂不使用
ForceCloseReason	强平原因	证券暂不使用

IsAutoSuspend	自动挂起标志	证券暂不使用
BusinessUnit	业务单元	证券暂不使用
RequestID	请求编号	
UserForceClose	用户强平标志	
IsSwapOrder	互换单标志	证券暂不使用
ExchangeID	交易所代码	
ClientID	客户代码	对应“交易账户代码”
InvestUnitID	投资单元代码	
AccountID	资金帐号代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	
CreditPositionType	头寸类型	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回报单录入操作请求的 ID，该 ID 由用户在报单录入时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.11. OnRspOrderAction 方法

报单操作应答。报单操作包括报单的撤销、报单的挂起、报单的激活、报单的修改。当客户端发出过报单操作指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspOrderAction(
                        CZQThostFtdcInputOrderActionField *pInputOrderAction,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast);
```

参数：

pOrderAction：指向报单操作结构的地址，包含了提交报单操作的输入数据，

和后台返回的报单编号。

报单操作结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
OrderActionRef	报单操作引用	
OrderRef	报单引用	
RequestID	请求编号	
FrontID	前置编号	
SessionID	会话编号	
ExchangeID	交易所代码	
OrderSysID	报单编号	
ActionFlag	操作标志	0=删除 3=修改
LimitPrice	价格	证券暂时不用
VolumeChange	数量变化	证券暂时不用
UserID	用户代码	证券暂时不用
InstrumentID	合约代码	证券暂时不用
IPAddress	IP 地址	
MacAddress	Mac 地址	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回用户报单操作请求的 ID，该 ID 由用户在报单操作时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.12. OnRspQryOrder 方法

报单查询请求。当客户端发出报单查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryOrder(
```

CZQThostFtdcOrderField *pOrder,
CZQThostFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)；

参数：

pOrder：指向报单信息结构的地址。

报单信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
OrderRef	报单引用	
UserID	用户代码	
OrderPriceType	报单价格条件	
Direction	买卖方向	
CombOffsetFlag	组合开平标志	
CombHedgeFlag	组合投机套保标志	
LimitPrice	价格	
VolumeTotalOriginal	数量（申报数量）	
TimeCondition	有效期类型	
GTDDate	GTD 日期	证券暂不使用
VolumeCondition	成交量类型	
MinVolume	最小成交量	证券暂不使用
ContingentCondition	触发条件	证券暂不使用
StopPrice	止损价	证券暂不使用
ForceCloseReason	强平原因	证券暂不使用
IsAutoSuspend	自动挂起标志	证券暂不使用
BusinessUnit	业务单元	证券暂不使用
RequestID	请求编号	
OrderLocalID	本地报单编号	
ExchangeID	交易所代码	
ParticipantID	会员代码	证券暂不使用
ClientID	客户代码	
ExchangeInstID	合约在交易所的代码	
TraderID	交易所交易员代码	
InstallID	安装编号	证券暂不使用
OrderSubmitStatus	报单提交状态	0=已经提交 1=撤单已经提交 2=修改已经提交 3=已经接受

		4=报单已经被拒绝 5=撤单已经被拒绝 6=改单已经被拒绝
NotifySequence	报单提示序号	证券暂不使用
TradingDay	交易日	
SettlementID	结算编号	证券暂不使用
OrderSysID	报单编号	
OrderSource	报单来源	证券暂不使用 0=来自参与者 1=来自管理员
OrderStatus	报单状态	0=全部成交 1=部分成交还在队列中 2=部分成交不在队列中 3=未成交还在队列中 4=未成交不在队列中 5=撤单 a=未知 b=尚未触发 c=已触发
OrderType	报单类型	证券始终返回 0。 0=正常 1=报价衍生 2=组合衍生 3=组合报单 4=条件单 5=互换单
VolumeTraded	今成交数量	
VolumeTotal	剩余数量	剩余数量 = 数量（申报数量） - 今成交数量 - 撤单数量
InsertDate	报单日期	
InsertTime	委托时间	
ActiveTime	激活时间	证券暂不使用
SuspendTime	挂起时间	证券暂不使用
UpdateTime	最后修改时间	证券暂不使用
CancelTime	撤销时间	
ActiveTraderID	最后修改交易所交易员代码	
ClearingPartID	结算会员编号	证券暂不使用
SequenceNo	序号	证券暂不使用
FrontID	前置编号	
SessionID	会话编号	
UserProductInfo	用户端产品信息	
StatusMsg	状态信息	存放交易所错误代码、系统错误代码

UserForceClose	用户强平标志	
ActiveUserID	操作用户代码	
BrokerOrderSeq	经纪公司报单编号	证券暂不使用
RelativeOrderSysID	相关报单	证券暂不使用
IsSwapOrder	互换单标志	证券暂不使用
InvestUnitID	投资单元代码	
AccountID	资金账户代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	
CreditPositionType	头寸类型	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回用户报单查询请求的 ID，该 ID 由用户在报单查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.13. OnRspQryTrade 方法

成交单查询应答。当客户端发出成交单查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryTrade(
                    CZQThostFtdcTradeField *pTrade,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast);
```

参数：

pTrade：指向成交信息结构的地址。

成交信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
OrderRef	报单引用	
UserID	用户代码	
ExchangeID	交易所代码	
TradeID	成交编号	
Direction	买卖方向	
OrderSysID	报单编号	
ParticipantID	会员代码	证券暂不使用
ClientID	客户代码	
TradingRole	交易角色	证券暂不使用 1=代理 2=自营 3=做市商
ExchangeInstID	合约在交易所的代码	
OffsetFlag	开平标志	0=开仓 1=平仓 2=强平 3=平今 4=平昨 5=强减 6=本地强平
HedgeFlag	投机套保标志	1=投机 2=套利 3=套保
Price	价格	
Volume	数量	
TradeDate	成交日期	
TradeTime	成交时间	
TradeType	成交类型	证券暂不使用 #=组合持仓拆分为单一持仓,初始化不应包含该类型的持仓 0=普通成交 1=期权执行 2=OTC 成交 3=期转现衍生成交 4=组合衍生成交
PriceSource	成交价来源	证券暂不使用 0=前成交价 1=买委托价 2=卖委托价

TraderID	交易所交易员代码	
OrderLocalID	本地报单编号	
ClearingPartID	结算会员编号	证券暂不使用
BusinessUnit	业务单元	证券暂不使用
SequenceNo	序号	证券暂不使用
TradingDay	交易日	
SettlementID	结算编号	证券暂不使用
BrokerOrderSeq	经纪公司报单编号	证券暂不使用
TradeSource	成交来源	证券暂不使用 0=来自交易所普通回报 1=来自查询
AccountID	资金账户代码	
InvestUnitID	投资单元代码	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回用户成交单请求的 ID，该 ID 由用户在成交单查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.14. OnRspQryInvestor 方法

会员客户查询应答。当客户端发出会员客户查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestor (
                    CZQThostFtdcInvestorField *pInvestor,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast);
```

参数：

pInvestor：指向投资者信息结构的地址。

投资者信息结构：

字段名称	属性	描述
InvestorID	投资者代码	
BrokerID	经纪公司代码	
InvestorName	投资者名称	
IdentifiedCardType	证件类型	0=组织机构代码 1=中国公民身份证 2=军官证 3=警官证 4=士兵证 5=户口簿 6=护照 7=台胞证 8=回乡证 9=营业执照号 A=税务登记号/当地纳税 ID B=港澳居民来往内地通行证 C=台湾居民来往大陆通行证 D=驾照 F=当地社保 ID G=当地身份证 H=商业登记证 I=港澳永久性居民身份证 J=人行开户许可证 x=其他证件
IdentifiedCardNo	证件号码	
IsActive	是否活跃	
Telephone	联系电话	
Address	通讯地址	
OpenDate	开户日期	
Mobile	手机	
BrokerageTemplateNo	佣金模板代码	
SecurityOptionTemplateNo	个股期权佣金模板代码	
MarginTemplateNo	保证金模板代码	
CreditTemplateNo	融资融券佣金模板代码	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.15. OnRspQryInvestorSecurityPosition 方法

投资者持仓查询应答。当客户端发出投资者持仓查询指令后，后交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestorSecurityPosition(
                    CZQThostFtdcInvestorSecurityPositionField    *
                    pInvestorSecurityPosition,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast);
```

参数：

pInvestorSecurityPosition：指向投资者持仓应答结构的地址。

///投资者证券持仓

字段名称	属性	描述
InstrumentID	合约代码	
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
ExchangeID	交易所代码	
ClientID	交易账户代码	

TradingDay	交易日	
PosiDirection	持仓多空方向	证券固定返回 1。 1=净 2=多头 3=空头
HistoryPosition	昨日持仓	
HistoryFrozen	昨日持仓冻结	
TodayBSPosition	今日买卖持仓	
TodayBSFrozen	今日买卖持仓冻结	
TodayPRPosition	今日申赎持仓	
TodayPRFrozen	今日申赎持仓冻结	
PositionCost	持仓成本	
TodaySMPosition	今日拆分合并持仓	
TodaySMFrozen	今日拆分合并持仓冻结	
MarginBuyVolume	融资买入数量	
ShortSellVolume	融券卖出数量	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回会员持仓查询请求的 ID，该 ID 由用户在会员持仓查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.16. OnRspQryTradingAccount 方法

请求查询资金账户响应。当客户端发出请求查询资金账户指令后，交易托

管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryTradingAccount(
        CZQThostFtdcTradingAccountField *pTradingAccount,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

参数：

pTradingAccount：指向资金账户结构的地址。

资金账户结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
AccountID	资金帐号	
Available	可用资金	
WithdrawQuota	可取资金	
CurrencyID	币种代码	
Deposit	入金金额	
Withdraw	出金金额	
FrozenMargin	冻结的保证金	
FrozenCash	冻结的资金	
FrozenCommission	冻结的手续费	
CurrMargin	当前保证金总额	
Commission	手续费	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
```


};

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.17. OnRspQryTradingCode 方法

请求查询交易编码响应。当客户端发出请求查询交易编码指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryTradingCode(
        CZQThostFtdcTradingCodeField *pTradingCode,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast) ；
```

参数：

pTradingCode：指向交易编码结构的地址。

交易编码结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
ExchangeID	交易所代码	
ClientID	客户代码	
IsActive	是否活跃	始终填写 1，活跃
ClientIDType	交易编码类型	
MarketID	市场代码	
BranchID	营业部代码	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
```

```
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.18. OnRspQryExchange 方法

请求查询交易所响应。当客户端发出请求查询交易所指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryExchange(
                    CZQThostFtdcExchangeField *pExchange,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast) ；
```

参数：

pExchange：指向交易所结构的地址。

交易所结构：

字段名称	属性	描述
ExchangeID	交易所代码	
ExchangeName	交易所名称	
ExchangeProperty	交易所属性	证券暂不使用。统一返回 0。 0=正常 1=根据成交生成报单
TradingDay	当前交易日	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
```

```
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.19. OnRspQrySecurity 方法

请求查询证券响应。当客户端发出请求查询证券指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQrySecurity(
                    CZQThostFtdcSecurityField *pSecurity,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast) ；
```

参数：

pSecurity：指向证券结构的地址。

证券结构：

字段名称	属性	描述
InstrumentID	合约代码	
ExchangeID	交易所代码	
InstrumentName	合约名称	
ExchangeInstID	合约在交易所的代码	
MarketID	市场代码	
ProductID	产品代码	
SecurityClassID	证券类别代码	
OrderUnit	申报单位	
BuyTradingUnit	买入交易单位	

SellTradingUnit	卖出交易单位	
MaxMarketOrderBuyVolume	市价单买最大下单量	
MinMarketOrderBuyVolume	市价单买最小下单量	
MaxLimitOrderBuyVolume	限价单买最大下单量	
MinLimitOrderBuyVolume	限价单买最小下单量	
MaxMarketOrderSellVolume	市价单卖最大下单量	
MinMarketOrderSellVolume	市价单卖最小下单量	
MaxLimitOrderSellVolume	限价单卖最大下单量	
MinLimitOrderSellVolume	限价单卖最小下单量	
VolumeMultiple	合约乘数	
PriceTick	最小变动价位	
OpenDate	上市日	
PositionType	持仓类型	1=净持仓 2-综合持仓
SecurityValue	面值	
SecurityStatusFlag	证券状态	用一个 32 位整数来表示，每一位代表一种状态。 0x00000001=停牌; 0x00000002=除权; 0x00000004=除息; 0x00000008=ST; 0x00000010=*ST; 0x00000020=上市首日; 0x00000040=长期停牌; 0x00000100=投资者适当性要求债券; 0x00000200=退市整理期; 0x00000400=退市转让; 0x00000800=风险警示债券; 0x00001000=暂停上市转让债券; 0x00002000=当前可融资（保留，暂未使用） 0x00004000=当前可融券（保留，暂未使用） 0x00008000=不接受新订单（目前仅在指数熔断时设置） 其它值目前未定义。
BondInterest	债券应计利息	
ConversionRate	折算率	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.20. OnRspQryDepthMarketData 方法

请求查询行情响应。当客户端发出请求查询行情指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryDepthMarketData(
                    CZQThostFtdcDepthMarketDataField *pDepthMarketData,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast) ;
```

参数：

pDepthMarketData：指向深度行情结构的地址。

深度行情结构：

字段名称	属性	描述
TradingDay	交易日	
InstrumentID	合约代码	
ExchangeID	交易所代码	
ExchangeInstID	交易所合约代码	
LastPrice	最新价	

PreSettlementPrice	昨结算价	
PreClosePrice	昨收盘	
PreOpenInterest	昨持仓量	
OpenPrice	今开盘	
HighestPrice	最高价	
LowestPrice	最新价	
Volume	数量	
Turnover	成交金额	
OpenInterest	持仓量	
ClosePrice	今收盘	
SettlementPrice	本次结算价	
UpperLimitPrice	涨停板价	
LowerLimitPrice	跌停板价	
PreDelta	昨虚实度	
CurrDelta	今虚实度	
UpdateTime	最后修改时间	
UpdateMillisec	最后修改毫秒	
BidPrice1	申买价一	
BidVolume1	申买量一	
AskPrice1	申卖价一	
AskVolume1	申卖量一	
BidPrice2	申买价二	
BidVolume2	申买量二	
AskPrice2	申卖价二	
AskVolume2	申卖量二	
BidPrice3	申买价三	
BidVolume3	申买量三	
AskPrice3	申卖价三	
AskVolume3	申卖量三	

BidPrice4	申买价四	
BidVolume4	申买量四	
AskPrice4	申卖价四	
AskVolume4	申卖量四	
BidPrice5	申买价五	
BidVolume5	申买量五	
AskPrice5	申卖价五	
AskVolume5	申卖量五	
AveragePrice	当日均价	
ActionDay	业务日期	
InstrumentName	合约名称	
TradingCount	成交笔数	
PERatio1	市盈率 1	
PERatio2	市盈率 2	
PriceUpDown1	价格升跌 1	
PriceUpDown2	价格升跌 2	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.21. OnRspQryOrderFundDetail 方法

请求查询报单资金明细响应响应。当客户端发出请求请求查询请求查询报单资金明细指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryOrderFundDetail(
                    CZQThostFtdcOrderFundDetailField *pOrderFundDetail,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast);
```

参数：

pOrderFundDetail：指向报单资金明细结构的地址。

报单资金明细结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
ExchangeID	交易所代码	
TradingDay	交易日	
OrderSysID	报单编号	
InvestUnitID	投资单元代码	
AccountID	资金账户代码	
TotalFrozen	初始冻结金额	报单发起时的冻结金额合计。目前计算公式： 初始冻结金额 = 报单金额 + 初始冻结保证金 + 冻结预估现金 + 初始冻结费用合计
TotalFee	总费用	报单结束时扣减的实际费用合计。目前计算公式： 总费用 = 印花税 + 过户费 + 经手费 + 证管费 + 结算费 + 佣金
Turnover	成交金额	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.22. OnRspQryMarket 方法

市场查询应答。当客户端发出市场查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryMarket (
                    CZQThostFtdcMarketField *pMarket,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast);
```

参数：

pMarket：指向市场结构的地址。

市场结构：

字段名称	属性	描述
ExchangeID	交易所代码	
MarketID	市场代码	
MarketName	市场名称	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
```



```
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.23. OnRtnTrade 方法

成交回报。当发生成交时交易托管系统会通知客户端，该方法会被调用。

函数原形：

void OnRtnTrade(CZQThostFtdcTradeField *pTrade);

参数：

pTrade：指向成交信息结构的地址。

成交信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
OrderRef	报单引用	
UserID	用户代码	
ExchangeID	交易所代码	
TradeID	成交编号	
Direction	买卖方向	
OrderSysID	报单编号	
ParticipantID	会员代码	证券暂不使用
ClientID	客户代码	
TradingRole	交易角色	证券暂不使用
ExchangeInstID	合约在交易所的代码	
OffsetFlag	开平标志	0=开仓 1=平仓 2=强平 3=平今 4=平昨 5=强减 6=本地强平

HedgeFlag	投机套保标志	1=投机 2=套利 3=套保
Price	价格	
Volume	数量	
TradeDate	成交日期	
TradeTime	成交时间	
TradeType	成交类型	证券暂不使用
PriceSource	成交价来源	证券暂不使用
TraderID	交易所交易员代码	
OrderLocalID	本地报单编号	
ClearingPartID	结算会员编号	证券暂不使用
BusinessUnit	业务单元	证券暂不使用
SequenceNo	序号	证券暂不使用
TradingDay	交易日	
SettlementID	结算编号	证券暂不使用
BrokerOrderSeq	经纪公司报单编号	证券暂不使用
TradeSource	成交来源	证券暂不使用
AccountID	资金账户代码	
InvestUnitID	投资单元代码	

6.3.24. OnRtnOrder 方法

报单回报。当客户端进行报单录入、报单操作及其它原因（如部分成交）导致报单状态发生变化时，交易托管系统会主动通知客户端，该方法会被调用。

函数原形：

void OnRtnOrder(CZQThostFtdcOrderField *pOrder);

参数：

pOrder：指向报单信息结构的地址。

报单信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
OrderRef	报单引用	
UserID	用户代码	

OrderPriceType	报单价格条件	
Direction	买卖方向	
CombOffsetFlag	组合开平标志	
CombHedgeFlag	组合投机套保标志	
LimitPrice	价格	
VolumeTotalOriginal	数量（申报数量）	
TimeCondition	有效期类型	
GTDDate	GTD 日期	证券暂不使用
VolumeCondition	成交量类型	
MinVolume	最小成交量	证券暂不使用
ContingentCondition	触发条件	证券暂不使用
StopPrice	止损价	证券暂不使用
ForceCloseReason	强平原因	证券暂不使用
IsAutoSuspend	自动挂起标志	证券暂不使用
BusinessUnit	业务单元	证券暂不使用
RequestID	请求编号	
OrderLocalID	本地报单编号	
ExchangeID	交易所代码	
ParticipantID	会员代码	证券暂不使用
ClientID	客户代码	
ExchangeInstID	合约在交易所的代码	
TraderID	交易所交易员代码	
InstallID	安装编号	证券暂不使用
OrderSubmitStatus	报单提交状态	0=已经提交 1=撤单已经提交 2=修改已经提交 3=已经接受 4=报单已经被拒绝 5=撤单已经被拒绝 6=改单已经被拒绝
NotifySequence	报单提示序号	证券暂不使用
TradingDay	交易日	
SettlementID	结算编号	证券暂不使用
OrderSysID	报单编号	
OrderSource	报单来源	证券暂不使用
OrderStatus	报单状态	0=全部成交 1=部分成交还在队列中 2=部分成交不在队列中 3=未成交还在队列中 4=未成交不在队列中 5=撤单 a=未知 b=尚未触发

		c=已触发
OrderType	报单类型	始终返回 0（正常）。
VolumeTraded	今成交数量	
VolumeTotal	剩余数量	
InsertDate	报单日期	
InsertTime	委托时间	
ActiveTime	激活时间	证券暂不使用
SuspendTime	挂起时间	证券暂不使用
UpdateTime	最后修改时间	证券暂不使用
CancelTime	撤销时间	
ActiveTraderID	最后修改交易所交易员代码	
ClearingPartID	结算会员编号	证券暂不使用
SequenceNo	序号	证券暂不使用
FrontID	前置编号	
SessionID	会话编号	
UserProductInfo	用户端产品信息	
StatusMsg	状态信息	存放交易所错误代码、系统错误代码
UserForceClose	用户强平标志	
ActiveUserID	操作用户代码	
BrokerOrderSeq	经纪公司报单编号	证券暂不使用
RelativeOrderSysID	相关报单	证券暂不使用
IsSwapOrder	互换单标志	证券暂不使用
InvestUnitID	投资单元代码	
AccountID	资金账户代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	
CreditPositionType	头寸类型	

6.3.25. OnErrRtnOrderInsert 方法

报单录入错误回报。由交易托管系统主动通知客户端，该方法会被调用。

函数原形：

```
void OnErrRtnOrderInsert(
                        CZQThostFtdcInputOrderField *pInputOrder,
                        CZQThostFtdcRspInfoField *pRspInfo)；
```

参数：

pInputOrder：指向报单录入结构的地址，包含了提交报单录入时的输入数据，和后台返回的报单编号。

输入报单结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
OrderRef	报单引用	
UserID	用户代码	证券暂不使用
OrderPriceType	报单价格条件	1=任意价 2=限价 3=最优价 4=最新价 5=最新价浮动上浮 1 个 ticks …… F=买一价浮动上浮 3 个 ticks G=五档价 a=本方最优
Direction	买卖方向	0=买入 1=卖出 2=ETF 申购 3=ETF 赎回 4=新股申购 5=正回购 6=逆回购 7=转托管 8=开放式基金申购 9=开放式基金赎回 d=质押入库 e=质押出库 f=配股配债 g=开放式基金拆分 h=开放式基金合并
CombOffsetFlag	组合开平标志	第一位是开平标志： 0=开仓 1=平仓 2=强平 3=平今 4=平昨 5=强减 6=本地强平
CombHedgeFlag	组合投机套保标志	第一位是投机套保标志：（除备兑外都

		写投机） 1=投机 2=套利 3=套保 4=备兑
LimitPrice	价格	
VolumeTotalOriginal	数量	
TimeCondition	有效期类型	1=立即完成，否则撤销 2=本节有效 3=当日有效 4=指定日期前有效 5=撤销前有效 6=集合竞价有效
GTDDate	GTD 日期	证券暂不使用
VolumeCondition	成交量类型	1=任何数量 2=最小数量 3=全部数量
MinVolume	最小成交量	证券暂不使用
ContingentCondition	触发条件	证券暂不使用
StopPrice	止损价	证券暂不使用
ForceCloseReason	强平原因	证券暂不使用
IsAutoSuspend	自动挂起标志	证券暂不使用
BusinessUnit	业务单元	证券暂不使用
RequestID	请求编号	
UserForceClose	用户强平标志	
IsSwapOrder	互换单标志	证券暂不使用
ExchangeID	交易所代码	
ClientID	客户代码	对应“交易账户代码”
InvestUnitID	投资单元代码	
AccountID	资金帐号代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	
CreditPositionType	头寸类型	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

6.3.26. OnErrRtnOrderAction 方法

报价操作错误回报。由交易托管系统主动通知客户端，该方法会被调用。

函数原形：

```
void OnErrRtnOrderAction (
                    CZQThostFtdcOrderActionField *pOrderAction,
                    CZQThostFtdcRspInfoField *pRspInfo);
```

参数：

pOrderAction：指向报价操作结构的地址，包含了报价操作请求的输入数据，和后台返回的报价编号。

报价操作结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
OrderActionRef	报单操作引用	
OrderRef	报单引用	
RequestID	请求编号	
FrontID	前置编号	
SessionID	会话编号	
ExchangeID	交易所代码	
OrderSysID	报单编号	
ActionFlag	操作标志	
LimitPrice	价格	证券暂时不用
VolumeChange	数量变化	证券暂时不用
ActionDate	操作日期	
ActionTime	操作时间	
TraderID	交易所交易员代码	
InstallID	安装编号	证券暂时不用
OrderLocalID	本地报单编号	
ActionLocalID	操作本地编号	
ParticipantID	会员代码	证券暂时不用
ClientID	客户代码	
BusinessUnit	业务单元	证券暂时不用
OrderActionStatus	报单操作状态	a=已经提交 b=已经接受’ c=已经被拒绝
UserID	用户代码	

StatusMsg	状态信息	存放交易所错误代码、系统错误代码
InstrumentID	合约代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

6.3.27. OnRspQrySecurityProduct 方法

请求证券品种响应。当客户端发出请求证券品种查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQrySecurityProduct(
                                CZQThostFtdcSecurityProductField *pSecurityProduct,
                                CZQThostFtdcRspInfoField *pRspInfo,
                                int nRequestID,
 bool bIsLast)
```

参数：

pSecurityProduct：指向证券品种的地址。

证券品种结构：

字段名称	属性	描述
ExchangeID	交易所代码	
ProductID	产品代码	
ProductName	产品名称	
CurrencyID	币种	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.28. OnRspQrySecurityClass 方法

请求查询证券类别响应。当客户端发出请求查询证券类别指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQrySecurityClass(
                    CZQThostFtdcSecurityClassField      *pSecurityClass,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast)
```

参数：

pSecurityClass：指向证券类别的地址。

证券类别结构：

字段名称	属性	描述
ExchangeID	交易所代码	
ProductID	产品代码	
SecurityClassID	证券类别代码	
SecurityClassName	证券类别名称	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.29. OnRspQryInvestUnit 方法

请求查询投资单元响应。当客户端发出请求投资单元查询别指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
Void OnRspQryInvestUnit(
                        CZQThostFtdcInvestUnitField *pInvestUnit,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast);
```

参数：

pInvestUnit：指向投资单元的地址。

投资单元结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
InvestUnitName	投资单元名称	

pRspInfo：指向响应信息结构的地址。

响应信息结构：


```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.30. OnRspQryBrokerage 方法

请求查询投资者佣金响应查询应答。当客户端发出请求投资者佣金指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryBrokerage(
                        CZQThostFtdcBrokerageField *pBrokerage,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast)
```

参数：

pBrokerage：指向投资者佣金结构的地址。

投资者佣金结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
BrokerageTemplateNo	佣金模板代码	
ExchangeID	交易所代码	
ProductID	品种代码	
SecurityClassID	证券类别代码	
InstrumentID	合约代码	

BizClass	业务类别	
BrokerageType	佣金类型	0=毛拥金。佣金中涵盖了经手费和证管费，对上述两项费用不再单独计算; 1=净佣金。佣金中不含经手费和证管费，上述两项费用需要单独计算，最低佣金收取金额判断时同样不包含经手费与证管费
RatioByAmt	佣金按金额收取比例	
RatioByPar	佣金按面值收取比例	
FeePerOrder	佣金按笔收取金额	
FeeMin	佣金最低收取金额	
FeeMax	佣金最高收取金额	
FeeByVolume	佣金按数量收取金额	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.31. OnRspQrySubscribingSharesQuota 方法

请求查询新股申购额度应答。当客户端发出请求新股申购额度指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQrySubscribingSharesQuota(
        CZQThostFtdcSubscribingSharesQuotaField  *pSubscribingSharesQuota,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)
```

参数：

pSubscribingSharesQuota：指向新股申购额度结构的地址。

新股申购额度接口：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
ExchangeID	交易所代码	
ClientID	客户代码	
MaxVolume	新股可申购额度	每个投资者可申购单一新股的最大额度

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.32. OnRspQrySubscribingShares 方法

请求查询新股信息应答。当客户端发出请求新股信息指令后，交易托管系

统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQrySubscribingShares(
                CZQThostFtdcSubscribingSharesField      *pSubscribingShares,
                CZQThostFtdcRspInfoField *pRspInfo,
                int nRequestID,
               bool bIsLast)
```

参数：

pSubscribingShares：指向新股信息结构的地址。

新股信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
ExchangeID	交易所代码	
InstrumentID	申购代码	新股的申购代码，上海的申购代码 730*** 对应 600***、780*** 对应 601***、732***对应 603***。
MarketID	市场代码	
ProductID	品种代码	
SecurityClassID	证券类别代码	
OnlineLimit	网上申购上限	投资者参与网上申购新股的上限，单位：股。
Price	发行价格	
CurrencyID	币种	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.33. OnRspQryPledgeOrderInfo 方法

请求查询质押申报代码应答。当客户端发出请求质押申报代码指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryPledgeOrderInfo(
                    CZQThostFtdcPledgeOrderInfoField    *pPledgeOrderInfo,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast)
```

参数：

pPledgeOrderInfo：指向质押申报代码结构的地址。

质押申报代码结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
ExchangeID	交易所代码	
InstrumentID	合约代码	
PledgeOrderID	质押申报代码	证券质押入库时的申报代码。
StandardBondID	标准券代码	入库后返回的标准券代码

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
    struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```


nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.34. OnRspQryInvestorPledgePosition 方法

请求查询质押持仓应答。当客户端发出请求质押持仓指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void  OnRspQryInvestorPledgePosition(
        CZQThostFtdcInvestorPledgePositionField  *pInvestorPledgePosition,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)
```

参数：

pInvestorPledgePosition：指向质押持仓结构的地址。

质押持仓结构：

字段名称	属性	描述
TradingDay	交易日	
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
ClientID	客户代码	
ExchangeID	交易所代码	
InstrumentID	合约代码	
PledgeOrderID	质押代码	
StandardBondID	标准券代码	
HisPledgePos	昨日入库持仓	
HisPledgePosFrozen	昨日入库持仓冻结	
TodayPledgePos	今日入库持仓	
TodayPledgePosFrozen	今日入库持仓冻结	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.35. OnRspQryInvestorPledgeInfo 方法

请求查询质回购应答。当客户端发出请求质押回购指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void  OnRspQryInvestorPledgeInfo(
                CZQThostFtdcInvestorPledgeInfoField    *pInvestorPledgeInfo,
                CZQThostFtdcRspInfoField *pRspInfo,
                int nRequestID,
                bool bIsLast)
```

参数：

pInvestorPledgeInfo：指向质押回购结构的地址。

投资者质押回购结构：

字段名称	属性	描述
TradingDay	交易日	
BrokerID	经纪公司代码	
InvestorID	投资者代码	

InvestUnitID	投资单元代码	
ClientID	客户代码	
ExchangeID	交易所代码	
StandardBondID	标准券代码	
PurchaseVol	融资数量	
PurchaseAmt	融资金额	
ReversePurVol	融券数量	
ReversePurAmt	融券金额	
NoTradePurchaseVol	未成交融资数量	
NoTradePurchaseAmt	未成交融资金额	
NoTradeReversePurVol	未成交融券数量	
NoTradeReversePurAmt	未成交融券金额	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.36. OnRspQryRepoRatio 方法

请求查询标准券使用率应答。当客户端发出请求标准券使用率指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryRepoRatio(
                        CZQThostFtdcRepoRatioField *pRepoRatio,
                       CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast)
```

参数：

pRepoRatio：指向标准券使用率结构的地址。

标准券使用率结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
ExchangeID	交易所代码	
StandardBondID	标准券代码	
StandardUseRatio	标准券使用率	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.37. OnRspQryRepurchaseMaxTimes 方法

请求查询回购放大倍数应答。当客户端发出请求回购放大倍数指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryRepurchaseMaxTimes(
            CZQThostFtdcRepurchaseMaxTimesField *pRepurchaseMaxTimes,
            CZQThostFtdcRspInfoField *pRspInfo,
```


int nRequestID,
bool bIsLast)

回购放大倍数结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
ExchangeID	交易所代码	
RepurchaseMaxTimes	回购放大倍数	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.38. OnRspQryETFFILE 方法

请求查询 ETF 清单应答。当客户端发出请求 ETF 清单指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryETFFile(
                    CZQThostFtdcETFFileField *pETFFile,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast)
```

参数：

pETFFile：指向标 ETF 清单结构的地址。

ETF 清单结构：

字段名称	属性	描述

TradingDay	交易日	T 日日期，格式 YYYYMMDD
ExchangeID	交易所代码	
ETFInstrumentID	ETF 合约代码	
ETFCreRedInstrumentID	ETF 申赎代码	
CreationRedemptionUnit	最小申购赎回单位份数	每个篮子（最小申购赎回单位）对应的 ETF 份数，目前为 1000000 份 取值范围：[0，99999999]
Maxcashratio	最大现金替代比例	申购时允许的最大现金替代比例，总长为 7 位（包括小数点），小数点后 5 位，例如：5.551%在文件中用 0.05551 表示 取值范围：[0.00001，1]
CreationStatus	是否允许申购	0=否;1=是
RedemptionStatus	是否允许赎回	0=否;1=是
EstimateCashComponent	预估现金差额	T 日每个篮子的现金差额，总长为 11 位（包括小数点），2 位小数 取值范围：[-9999999.99，99999999.99]
CashComponent	前一交易日现金差额	T－1 日申购赎回基准单位的现金余额 ，总长为 11 位（包括小数点），2 位小数 取值范围：[-9999999.99，99999999.99]
NAV	前一交易日基金单位净值	T－1 日基金的单位净值。总长为 8 位，包括小数点，其中小数部分为 4 位。 取值范围：[0，999.9999]
NAVperCU	前一交易日申赎基准单位净值	T－1 日申购赎回基准单位净值，总长为 12 位，包括小数点，其中小数部分为 2 位。 取值范围：[0，999999999.99]
DividendPerCU	当日申购赎回基准单位的红利金额	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```


nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.39. OnRspQryETFBasket 方法

请求查询 ETFBasket 成分证券应答。当客户端发出请求 ETF 成分证券指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryETFBasket(
                        CZQThostFtdcETFBasketField *pETFBasket,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast)
```

参数：

pETFBasket：指向 ETF 成分证券结构的地址。

ETF 成分证券结构：

字段名称	属性	描述
TradingDay	交易日	T 日日期，格式 YYYYMMDD
ExchangeID	交易所代码	
ETFInstrumentID	ETF 合约代码	
InstrumentID	成分合约代码	成分合约代码
InstrumentName	合约名称	四位中文，左对齐，右补空，对正确性不校验
Volume	合约数量	每个申购篮子中该成分合约的数量
ETFCurrenceReplaceStatus	替代标志	表示该证券是否允许用现金进行替代。 上证所:0 表示沪市证券禁止现金替代（必须有证券） 1 表示沪市股票可以进行现金替代(先用证券，证券不足的话用现金替代） 2 表示沪市证券必须用现金替代。 对于跨市场 ETF，3 表示深市证券退补现金替代，此时对应的溢价比例和

		现金金额部分必填。 对于跨市场 ETF，4 表示深市证券必须现金替代，对应的现金金额部分不为空。 （沪市成分证券该字段取值只能为 0，1，2。） 深圳：0 表示禁止现金替代（必须有证券），1 表示可以进行现金替代（先用证券，证券不足的话用现金替代），2 表示 必须用现金替代
Premium	溢价比例	证券用现金进行替代的时候，计算价格时增加或减少的比例（不含 100%）。 总长为 7 位（包括小数点），小数点后 5 位，数据右对齐，左补空，小数必须为 5 位，例如：2.551％在文件中用 0.02551 表示；2.1% 在文件中用 0.02100 表示。 此字段只有当替代标志为‘1’或‘3’时才使用。
CreationReplaceAmount	申购替代金额	当某只证券必须用现金替代或者退补现金替代的时候，申购该证券所需总金额。 总长为 12 位（包括小数点），小数点后 3 位 数据右对齐，左补空，小数必须为 3 位，例如：2000 在文件中用 2000.000 表示 此字段只有当替代标志为‘2’、‘3’或‘4’时才使用。
RedemptionReplaceAmount	赎回替代金额	当某只证券必须用现金替代或者退补现金替代的时候，赎回该证券得到的总金额。 总长为 12 位（包括小数点），小数点后 3 位 数据右对齐，左补空，小数必须为 3 位，例如：2000 在文件中用 2000.000 表示 此字段只有当替代标志为‘2’、‘3’或‘4’时才使用。
Market	挂牌市场	成分证券挂牌市场，市场代码采取 ISO10383 标准，XSHE- 深圳市场，XSHG-上证所市场，XHKG-香港市场

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
```

```
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.40. OnRspQryTransferFund 方法

请求查询资金转移应答。当客户端发出请求资金转移指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryTransferFund(
                          CZQThostFtdcTransferFundField  *pTransferFund,
                          CZQThostFtdcRspInfoField *pRspInfo,
                          int nRequestID,
                           bool bIsLast)
```

参数：

pTransferFund：指向资金转移结构的地址。

资金转移回报结构：

字段名称	属性	描述
CTPFundSerial	CTP 资金流水号	
ApplySerial	申请流水号	
FrontID	前置编号	
SessionID	会话编号	
BrokerID	经纪公司代码	
AccoutID	资金账户代码	CTP 系统的资金账户
CurrencyID	币种	
TransferDirection	转账方向	0：转入 CTP， 1：转出 CTP
Amount	出入金额	

TransferStatus	转移状态	0：正常；1：已冲正
OperatorID	操作人员	
OperateDate	操作日期	
OperateTime	操作时间	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.41. OnRspQryTransferPosition 方法

请求查询仓位转移应答。当客户端发出请求仓位转移指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryTransferPosition(
                    CZQThostFtdcTransferPositionField     *pTransferPosition,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                     bool bIsLast)
```

参数：

pTransferPosition：指向仓位转移结构的地址。

仓位转移回报结构：

字段名称	属性	描述
CTPPositionSerial	CTP 仓位转移流水号	

ApplySerial	申请流水号	
FrontID	前置编号	
SessionID	会话编号	
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
ExchangeID	交易所代码	
ClientID	客户代码	
InstrumentID	合约代码	
TradingDay	交易日	
TransferDirection	转账方向	0：转入 CTP；1：转出 CTP
TransferPositionType	转移持仓类型	0：任意仓 1：昨仓 2：今买卖仓 3：今申赎仓
HistoryVolume	昨日仓位数量	昨日仓位转入转出数量
TodayBSVolume	今日买卖仓位数量	今日买卖仓位当日转入转出数量
TodayPRVolume	今日申赎仓位数量	今日申赎仓位当日转入转出数量
TransferStatus	转移状态	0：正常；1：已冲正
OperatorID	操作人员	
OperateDate	操作日期	
OperateTime	操作时间	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
```

```
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.42. OnRspQryInvestUnitAndTradingAcct 方法

请求查询投资单元与交易资金账户关系应答。当客户端发出请求投资单元与交易资金账户关系查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestUnitAndTradingAcct(
CZQThostFtdcInvestUnitAndTradingAcctField      *pInvestUnitAndTradingAcct,
CZQThostFtdcRspInfoField *pRspInfo,
 int nRequestID,
 bool bIsLast)
```

参数：

pInvestUnitAndTradingAcct：指向投资单元与交易资金账户关系结构的地址。

投资单元与交易资金账户关系结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestUnitID	投资单元代码	
ExchangeID	交易所代码	
ClientID	客户代码	
ProductID	产品代码	
AccountID	资金账户代码	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
```

```
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.43. OnRspQryInvestUnitAndUser 方法

请求查询投资单元与用户关系应答。当客户端发出请求投资单元与用户查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestUnitAndUser(
                CZQThostFtdcInvestUnitAndUserField    *pInvestUnitAndUser,
                CZQThostFtdcRspInfoField *pRspInfo,
                int nRequestID,
                bool bIsLast)
```

参数：

pInvestUnitAndUser：指向投资单元与用户关系结构的地址。

投资者用户与投资单元关系结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
UserID	用户代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	

pRspInfo：指向响应信息结构的地址。

响应信息结构：


```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.44. OnRspQryInstrument 方法

请求个股期权合约查询响应应答。当客户端发出请求投个股期权合约查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInstrument(
                        CZQThostFtdcInstrumentField    *pInstrument,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast) ;
```

参数：

pInstrument：指向个股期权合约结构的地址。

个股期权合约查询响应结构：

字段名称	属性	描述
InstrumentID	合约代码	
ExchangeID	交易所代码	
InstrumentName	合约名称	
ExchangeInstID	合约在交易所的代码	与合约代码相同
ProductID	产品代码	证券定义
ProductClass	产品类型	证券暂不使用
DeliveryYear	交割年份	根据行权交割日拆分。
DeliveryMonth	交割月份	根据行权交割日拆分。
MaxMarketOrderBuyVolume	市价单买最大下单量	
MinMarketOrderBuyVolume	市价单买最小下单量	

MaxLimitOrderBuyVolume	限价单买最大下单量	
MinLimitOrderBuyVolume	限价单买最小下单量	
MaxMarketOrderSellVolume	市价单卖最大下单量	
MinMarketOrderSellVolume	市价单卖最小下单量	
MaxLimitOrderSellVolume	限价单卖最大下单量	
MinLimitOrderSellVolume	限价单卖最小下单量	
VolumeMultiple	合约乘数	
PriceTick	最小变动价位	
CreateDate	创建日	证券暂不使用
OpenDate	上市日	
ExpireDate	到期日	
StartDelivDate	开始交割日	
EndDelivDate	结束交割日	
InstLifePhase	合约生命周期状态	证券暂不使用 0=未上市 1=上市 2=停牌 3=到期
IsTrading	当前是否交易	交易所接口 SecurityStatusFlag[1]=='0'（未连续停牌）
PositionType	持仓类型	固定为 2。 1-净持仓 2-综合持仓
PositionDateType	持仓日期类型	固定为 2。 1=使用历史持仓 2=不使用历史持仓
LongMarginRatio	多头保证金率	证券暂不使用
ShortMarginRatio	空头保证金率	证券暂不使用
MaxMarginSideAlgorithm	是否使用大额单边保证金算法	证券暂不使用
UnderlyingInstrID	基础商品代码	标的证券代码
StrikePrice	执行价	
OptionsType	期权类型	1=看涨 2=看跌
UnderlyingMultiple	合约基础商品乘数	
CombinationType	组合类型	证券暂不使用。 0=期货组合 1=垂直价差 BUL 2=垂直价差 BER 3=跨式组合 4=宽跨式组合 5=备兑组合 6=时间价差组合
BuyTradingUnit	买入交易单位	

SellTradingUnit	卖出交易单位	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.45. OnRspQryInvestorPosition 方法

请求查询投资者合约持仓查询响应应答。当客户端发出投资者合约持仓查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
virtual void OnRspQryInvestorPosition(
                    CZQThostFtdcInvestorPositionField *pInvestorPosition,
                    CZQThostFtdcRspInfoField *pRspInfo,
  int nRequestID, bool bIsLast) {};
```

参数：

pInvestorPosition：指向投资者合约持仓的地址。

投资者合约持仓查询响应结构：

字段名称	属性	描述
InstrumentID	合约代码	
BrokerID	经纪公司代码	
InvestorID	投资者代码	

PosiDirection	持仓多空方向	
HedgeFlag	投机套保标志	
PositionDate	持仓日期	固定为 1。 1=今日持仓 2=历史持仓
YdPosition	上日持仓	同昨日持仓
Position	今日持仓	总持仓（含冻结）
LongFrozen	多头冻结	
ShortFrozen	空头冻结	
LongFrozenAmount	多头冻结金额	
ShortFrozenAmount	空头冻结金额	
OpenVolume	开仓量	当日成交开仓数量
CloseVolume	平仓量	当日成交平仓数量
OpenAmount	开仓金额	当日成交权利金
CloseAmount	平仓金额	当日成交权利金
PositionCost	持仓成本	
PreMargin	上次占用的保证金	证券暂不使用
UseMargin	占用的保证金	总持仓*合约乘数*单位保证金（与期货略有不同）
FrozenMargin	冻结的保证金	
FrozenCash	冻结的资金	
FrozenCommission	冻结的手续费	
CashIn	资金差额	
Commission	手续费	上场时为 0
CloseProfit	平仓盈亏	证券暂不使用
PositionProfit	持仓盈亏	证券暂不使用
PreSettlementPrice	上次结算价	从行情信息获取
SettlementPrice	本次结算价	最新价。从行情信息获取
TradingDay	交易日	
SettlementID	结算编号	证券暂不使用
OpenCost	开仓成本	证券暂不使用

ExchangeMargin	交易所保证金	证券暂不使用
CombPosition	组合成交形成的持仓	
CombLongFrozen	组合多头冻结	
CombShortFrozen	组合空头冻结	
CloseProfitByDate	逐日盯市平仓盈亏	证券暂不使用
CloseProfitByTrade	逐笔对冲平仓盈亏	证券暂不使用
TodayPosition	今日持仓	
MarginRateByMoney	保证金率	证券暂不使用
MarginRateByVolume	保证金率(按手数)	证券暂不使用
StrikeFrozen	执行冻结	
StrikeFrozenAmount	执行冻结金额	
AbandonFrozen	放弃执行冻结	证券暂不使用
ExchangeID	交易所代码	
ClientID	客户代码	期货无
InvestUnitID	投资单元代码	期货无

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.46. OnRspQryLockPosition 方法

请求查询投资者锁定持仓应答。当客户端发出请求锁定持仓查询指令后，交

易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryLockPosition(
                        CZQThostFtdcLockPositionField *pLockPosition,
                        CZQThostFtdcRspInfoField *pRspInfo, int
                        nRequestID,
                       bool bIsLast) ;
```

参数：

pLockPosition：指向投资者锁定持仓关系结构的地址。

锁定持仓查询相应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
ExchangeID	交易所代码	
Volume	数量	
FrozenVolume	冻结数量	
ClientID	客户代码	
InvestUnitID	投资单元代码	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
    struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```


nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.47. OnRspQryLock 方法

请求查询投资者锁定解锁应答。当客户端发出请求锁定解锁查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryLock(
                    CZQThostFtdcLockField *pLock,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast) ;
```

参数：

pLock：指向投资者锁定解锁关系结构的地址。

投资者锁定解锁响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
LockRef	锁定引用	
UserID	用户代码	
Volume	数量	
RequestID	请求编号	
BusinessUnit	业务单元	证券暂不使用
LockType	锁定类型	
LockLocalID	本地锁定编号	
ExchangeID	交易所代码	
ParticipantID	会员代码	
ClientID	客户代码	
ExchangeInstID	合约在交易所的代码	
TraderID	交易所交易员代码	

InstallID	安装编号	
OrderSubmitStatus	报单提交状态	
NotifySequence	报单提示序号	
TradingDay	交易日	
SettlementID	结算编号	
LockSysID	锁定编号	
InsertDate	报单日期	
InsertTime	插入时间	
CancelTime	撤销时间	
OrderStatus	报单状态	
ClearingPartID	结算会员编号	
SequenceNo	序号	
FrontID	前置编号	
SessionID	会话编号	
UserProductInfo	用户端产品信息	
StatusMsg	状态信息	
ActiveUserID	操作用户代码	
BrokerLockSeq	经纪公司报单编号	
BranchID	营业部编号	
InvestUnitID	投资单元代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.48. OnRspQryExecOrder 方法

请求查询执行宣告应答。当客户端发出请求执行宣告查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryExecOrder(
                        CZQThostFtdcExecOrderField    *pExecOrder,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast) ;
```

参数：

pExecOrder：指向执行宣告结构的地址。

执行宣告查询应答结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
ExecOrderRef	执行宣告引用	
UserID	用户代码	
Volume	数量	
RequestID	请求编号	
BusinessUnit	业务单元	证券暂不使用
OffsetFlag	开平标志	
HedgeFlag	投机套保标志	
ActionType	执行类型	
PosiDirection	保留头寸申请的持仓方向	
ReservePositionFlag	期权行权后是否保留期货头寸的标记	
CloseFlag	期权行权后生成的头寸是否自动平仓	
ExecOrderLocalID	本地执行宣告编号	
ExchangeID	交易所代码	
ParticipantID	会员代码	证券暂不使用
ClientID	客户代码	

ExchangeInstID	合约在交易所的代码	
TraderID	交易所交易员代码	
InstallID	安装编号	证券暂不使用
OrderSubmitStatus	执行宣告提交状态	
NotifySequence	报单提示序号	证券暂不使用
TradingDay	交易日	
SettlementID	结算编号	证券暂不使用
ExecOrderSysID	执行宣告编号	
InsertDate	报单日期	
InsertTime	插入时间	
CancelTime	撤销时间	
ExecResult	执行结果	n=没有执行 c=已经取消 0=执行成功 1=期权持仓不够 2=资金不够 3=会员不存在 4=客户不存在 6=合约不存在 7=没有执行权限 8=不合理的数量 9=没有足够的历史成交 a=未知
ClearingPartID	结算会员编号	证券暂不使用
SequenceNo	序号	证券暂不使用
FrontID	前置编号	
SessionID	会话编号	
UserProductInfo	用户端产品信息	
StatusMsg	状态信息	
ActiveUserID	操作用户代码	
BrokerExecOrderSeq	经纪公司报单编号	证券暂不使用
BranchID	营业部编号	
InvestUnitID	投资单元代码	
AccountID	资金帐号代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

注：关于 ExecResult 字段，个股期权仅使用“没有执行”、“已经取消”、“未知”这 3 个值。其中报单完成但还没有返回结果是“未知”，交易所返回申报成功是“没有执行”，交易所拒绝是“已经取消”。

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.49. OnRspQryLimitAmount 方法

请求投资者买入额度查询应答。当客户端发出请求买入额度查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryLimitAmount(
                    CZQThostFtdcLimitAmountField *pLimitAmount,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast) ;
```

参数：

pLimitAmount：指向投资者额度结构的地址。

投资者买入额度查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
ExchangeID	交易所代码	
LongAmount	多头持仓金额限制	

LongAmountFrozen	多头持仓金额冻结	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.50. OnRspQryLimitPosition 方法

请求投资者持仓限制查询应答。当客户端发出投资者持仓限制查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryLimitPosition(
                    CZQThostFtdcLimitPositionField *pLimitPosition,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast);
```

参数：

pLimitPosition：指向投资者持仓限制结构的地址。

投资者持仓限制查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	

InvestorID	投资者代码	
InstrumentID	合约代码	
ExchangeID	交易所代码	
TotalVolume	总数量限制	
LongVolume	多头数量限制	
OpenVolume	当日开仓数量限制	
TotalVolumeFrozen	总数量冻结	
LongVolumeFrozen	多头数量冻结	
OpenVolumeFrozen	当日开仓数量冻结	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.51. OnRspLockInsert 方法

锁定录入应答。当客户端发出过锁定录入指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspLockInsert(
                                CZQThostFtdcInputLockField    *pInputLock,
```

CZQThostFtdcRspInfoField *pRspInfo,

int nRequestID,

bool bIsLast) ;

参数：

pInputLock：指向锁定/解锁结构的地址。

锁定录入应答结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
LockRef	锁定引用	
UserID	用户代码	证券暂不使用
Volume	数量	
RequestID	请求编号	
BusinessUnit	业务单元	证券暂不使用
LockType	锁定类型	1=锁定 2=解锁
ExchangeID	交易所代码	
ClientID	客户代码	对应“交易账户代码”
InvestUnitID	投资单元代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.52. OnRtnLock 方法

锁定回报。当客户端进行锁定录入、锁定操作及其它原因导致报单状态发生变化时，交易托管系统会主动通知客户端，该方法会被调用。

函数原形：

void OnRtnLock (CZQThostFtdcLockField *pLock);

参数：

pLock：指向锁定录入结构的地址，包含了提行权录入时的输入数据，和后台返回的锁定系统编号。

锁定通知结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
LockRef	锁定引用	
UserID	用户代码	
Volume	数量	
RequestID	请求编号	
BusinessUnit	业务单元	证券暂不使用
LockType	锁定类型	
LockLocalID	本地锁定编号	
ExchangeID	交易所代码	
ParticipantID	会员代码	
ClientID	客户代码	
ExchangeInstID	合约在交易所的代码	
TraderID	交易所交易员代码	
InstallID	安装编号	
OrderSubmitStatus	报单提交状态	
NotifySequence	报单提示序号	
TradingDay	交易日	
SettlementID	结算编号	
LockSysID	锁定编号	
InsertDate	报单日期	
InsertTime	插入时间	
CancelTime	撤销时间	

OrderStatus	报单状态	
ClearingPartID	结算会员编号	
SequenceNo	序号	
FrontID	前置编号	
SessionID	会话编号	
UserProductInfo	用户端产品信息	
StatusMsg	状态信息	
ActiveUserID	操作用户代码	
BrokerLockSeq	经纪公司报单编号	
BranchID	营业部编号	
InvestUnitID	投资单元代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

6.3.53. OnErrRtnLockInsert 方法

锁定录入错误回报。由交易托管系统主动通知客户端，该方法会被调用。

函数原形：

```
void OnErrRtnLockInsert(
                        CZQThostFtdcInputLockField *pInputLock,
                        CZQThostFtdcRspInfoField    *pRspInfo) ；
```

参数：

pInputLock：指向锁定录入结构的地址，包含了提交锁定录入时的输入数据，和后台返回的锁定编号。

锁定录入错误回报结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
LockRef	锁定引用	
UserID	用户代码	证券暂不使用
Volume	数量	
RequestID	请求编号	
BusinessUnit	业务单元	证券暂不使用
LockType	锁定类型	1=锁定

		2=解锁
ExchangeID	交易所代码	
ClientID	客户代码	对应“交易账户代码”
InvestUnitID	投资单元代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

6.3.54. OnRspExecOrderInsert 方法

行权录入应答。当客户端发出过行权录入指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspExecOrderInsert(
                        CZQThostFtdcInputExecOrderField  *pInputExecOrder,
                        CZQThostFtdcRspInfoField *pRspInfo,
                         int nRequestID,
                         bool bIsLast) ;
```

参数：

pInputExecOrder：指向行权录入结构的地址，包含了提行权录入时的输入数据，和后台返回的行权系统编号。

行权输入应答结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
ExecOrderRef	执行宣告引用	
UserID	用户代码	证券暂不使用
Volume	数量	
RequestID	请求编号	
BusinessUnit	业务单元	证券暂不使用
OffsetFlag	开平标志	证券暂不使用
HedgeFlag	投机套保标志	证券暂不使用。 1=投机 2=套利 3=套保 4=备兑
ActionType	执行类型	固定为 1 1=执行 2=放弃
PosiDirection	保留头寸申请的持仓方向	证券暂不使用。 1=净 2=多头 3=空头
ReservePositionFlag	期权行权后是否保留期货头寸的标记	证券暂不使用。 0=保留 1=不保留
CloseFlag	期权行权后生成的头寸是否自动平仓	证券暂不使用。 0=自动平仓 1=免于自动平仓
ExchangeID	交易所代码	
ClientID	客户代码	对应“交易账户代码”
InvestUnitID	投资单元代码	
AccountID	资金帐号代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
```

```
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.55. OnRtnExecOrder

行权录入通知。交易托管系统返回响应时，该方法会被调用

函数原形：

```
void OnRtnExecOrder(CZQThostFtdcExecOrderField *pExecOrder) ;
```

参数：

pExecOrder：指向行权录入结构的地址，包含了提行权录入时的输入数据，和后台返回的行权系统编号。

行权录入通知结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
ExecOrderRef	执行宣告引用	
UserID	用户代码	
Volume	数量	
RequestID	请求编号	
BusinessUnit	业务单元	证券暂不使用
OffsetFlag	开平标志	
HedgeFlag	投机套保标志	
ActionType	执行类型	
PosiDirection	保留头寸申请的持仓方向	
ReservePositionFlag	期权行权后是否保留期货头寸的标记	
CloseFlag	期权行权后生成的头寸是否自动平仓	
ExecOrderLocalID	本地执行宣告编号	

ExchangeID	交易所代码	
ParticipantID	会员代码	证券暂不使用
ClientID	客户代码	
ExchangeInstID	合约在交易所的代码	
TraderID	交易所交易员代码	
InstallID	安装编号	证券暂不使用
OrderSubmitStatus	执行宣告提交状态	
NotifySequence	报单提示序号	证券暂不使用
TradingDay	交易日	
SettlementID	结算编号	证券暂不使用
ExecOrderSysID	执行宣告编号	
InsertDate	报单日期	
InsertTime	插入时间	
CancelTime	撤销时间	
ExecResult	执行结果	n=没有执行 c=已经取消 0=执行成功 1=期权持仓不够 2=资金不够 3=会员不存在 4=客户不存在 6=合约不存在 7=没有执行权限 8=不合理的数量 9=没有足够的历史成交 a=未知
ClearingPartID	结算会员编号	证券暂不使用
SequenceNo	序号	证券暂不使用
FrontID	前置编号	
SessionID	会话编号	
UserProductInfo	用户端产品信息	
StatusMsg	状态信息	
ActiveUserID	操作用户代码	
BrokerExecOrderSeq	经纪公司报单编号	证券暂不使用
BranchID	营业部编号	
InvestUnitID	投资单元代码	
AccountID	资金帐号代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

注：关于 ExecResult 字段，个股期权仅使用“没有执行”、“已经取消”、“未知”这 3 个值。其中报单完成但还没有返回结果是“未知”，交易所返回申报成功是“没有执行”，交易所拒绝是“已经取消”。

6.3.56. OnErrRtnExecOrderInsert 方法

行权录入错误回报。由交易托管系统主动通知客户端，该方法会被调用。

函数原形：

void OnErrRtnExecOrderInsert (

CZQThostFtdcInputExecOrderField *pInputExecOrder,

CZQThostFtdcRspInfoField *pRspInfo) ；

参数：

pInputExecOrder：指向行权录入结构的地址，包含了提交行权录入时的输入数据，和后台返回的行权系统编号。

行权录入错误回报结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InstrumentID	合约代码	
ExecOrderRef	执行宣告引用	
UserID	用户代码	证券暂不使用
Volume	数量	
RequestID	请求编号	
BusinessUnit	业务单元	证券暂不使用
OffsetFlag	开平标志	证券暂不使用
HedgeFlag	投机套保标志	证券暂不使用。 1=投机 2=套利 3=套保 4=备兑
ActionType	执行类型	固定为 1 1=执行 2=放弃
PosiDirection	保留头寸申请的持仓方向	证券暂不使用。 1=净 2=多头 3=空头
ReservePositionFlag	期权行权后是否保留期货	证券暂不使用。

	头寸的标记	0=保留 1=不保留
CloseFlag	期权行权后生成的头寸是否自动平仓	证券暂不使用。 0=自动平仓 1=免于自动平仓
ExchangeID	交易所代码	
ClientID	客户代码	对应“交易账户代码”
InvestUnitID	投资单元代码	
AccountID	资金帐号代码	
IPAddress	IP 地址	
MacAddress	Mac 地址	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

6.3.57. OnRspExecOrderAction 方法

行权操作录入应答。当客户端发出行权操作录入指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspExecOrderAction(
        CZQThostFtdcInputExecOrderActionField      *pInputExecOrderAction,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast) ;
```

参数：

pInputExecOrderAction：指向行权操作录入结构的地址，包含了提交行权操作录入时的输入数据，和后台返回的行权操作系统编号。

行权操作输入应答结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
ExecOrderActionRef	执行宣告操作引用	
ExecOrderRef	执行宣告引用	执行宣告引用、前置编号、会话编号，3 个字段需一起填写，否则就都不填
RequestID	请求编号	
FrontID	前置编号	
SessionID	会话编号	
ExchangeID	交易所代码	
ExecOrderSysID	执行宣告编号	
ActionFlag	操作标志	0=删除 3=修改
UserID	用户代码	证券暂时不用
InstrumentID	合约代码	证券暂时不用
IPAddress	IP 地址	
MacAddress	Mac 地址	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.58. OnErrRtnExecOrderAction 方法

行权操作录入错误通知。交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnErrRtnExecOrderAction(
                    CZQThostFtdcExecOrderActionField   *pExecOrderAction,
                    CZQThostFtdcRspInfoField *pRspInfo) ;
;
```

参数：

pExecOrderAction：指向行权操作录入结构的地址，包含了提交行权操作录入时的输入数据，和后台返回的行权操作编号。

行权操作录入错误通知结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
ExecOrderActionRef	执行宣告操作引用	
ExecOrderRef	执行宣告引用	
RequestID	请求编号	
FrontID	前置编号	
SessionID	会话编号	
ExchangeID	交易所代码	
ExecOrderSysID	执行宣告编号	
ActionFlag	操作标志	
ActionDate	操作日期	
ActionTime	操作时间	
TraderID	交易所交易员代码	
InstallID	安装编号	证券暂时不用
ExecOrderLocalID	本地执行宣告编号	
ActionLocalID	操作本地编号	
ParticipantID	会员代码	证券暂时不用
ClientID	客户代码	
BusinessUnit	业务单元	证券暂时不用
OrderActionStatus	报单操作状态	a=已经提交 b=已经接受’ c=已经被拒绝
UserID	用户代码	

ActionType	执行类型	
StatusMsg	状态信息	存放交易所错误代码、系统错误代码
InstrumentID	合约代码	
BranchID	营业部编号	
IPAddress	IP 地址	
MacAddress	Mac 地址	

6.3.59. OnRspSecurityTransfer 方法

现券转移录入应答。当客户端发出过现券转移录入指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspSecurityTransfer(
        CZQThostFtdcInputSecurityTransferField  *pInputSecurityTransfer,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

参数：

pInputSecurityTransfer：指向现券转移录入结构的地址，包含了提交现券转移录入时的输入数据，和后台返回的现券转移系统编号。

现券转移输入应答结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
SecurityInvestorID	现券系统投资者代码	
SecurityInvestUnitID	现券系统投资单元代码	
ExchangeID	交易所代码	
ClientID	客户代码	
InstrumentID	合约代码	
ApplySerial	申请流水号	
SecurityTransferDirection	现券转移方向	0：转入现券系统；1：转出现券系统
Volume	数量	
TransferPositionType	转移持仓类型	0：任意仓

		1：昨仓 2：今买卖仓 3：今申赎仓 转出时支持所有类型；转入不支持任意仓

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.60. OnRtnSecurityTransfer 方法

现券转移录入通知。交易托管系统返回响应时，该方法会被调用

函数原形：

```
void OnRtnSecurityTransfer(
                    CZQThostFtdcSecurityTransferField *pSecurityTransfer) ;
```

参数：

pSecurityTransfer：指向现券转移结构的地址。

现券转移录入通知结构：

字段名称	属性	描述
SecurityTransferSerial	现券转移流水号	
ApplySerial	申请流水号	

SecurityPositionSerial	证券系统仓位转移流水号	
FrontID	前置编号	
SessionID	会话编号	
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
SecurityInvestorID	现券系统投资者代码	
SecurityInvestUnitID	现券系统投资单元代码	
ExchangeID	交易所代码	
ClientID	客户代码	
InstrumentID	合约代码	
SecurityTransferDirection	转移方向	0：转入现券系统；1：转出现券系统
TransferPositionType	转移持仓类型	0：任意仓 1：昨仓 2：今买卖仓 3：今申赎仓
HistoryVolume	昨日仓位数量	昨日仓位转入转出数量
TodayBSVolume	今日买卖仓位数量	今日买卖仓位当日转入转出数量
TodayPRVolume	今日申赎仓位数量	今日申赎仓位当日转入转出数量
SecurityTransferStatus	转移状态	显示报单目前的处理状态。1=初始状态; 2=转移成功; 3=转移失败; 4=报盘失败
StatusMsg	状态信息	
OperatorID	操作人员	
OperateDate	操作日期	
OperateTime	操作时间	

6.3.61. OnErrRtnSecurityTransfer 方法

现券转移录入错误通知。交易托管系统返回响应时，该方法会被调用。

函数原形：

void OnErrRtnSecurityTransfer(

CZQThostFtdcInputSecurityTransferField *pInputSecurityTransfer,

CZQThostFtdcRspInfoField *pRspInfo) ;

参数：

pInputSecurityTransfer：指向现券转移录入结构的地址，包含了提交现券转移录入时的输入数据，和后台返回的现券转移系统编号。

现券转移输入错误通知结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
SecurityInvestorID	现券系统投资者代码	
SecurityInvestUnitID	现券系统投资单元代码	
ExchangeID	交易所代码	
ClientID	客户代码	
InstrumentID	合约代码	
ApplySerial	申请流水号	
SecurityTransferDirection	现券转移方向	0：转入现券系统；1：转出现券系统
Volume	数量	
TransferPositionType	转移持仓类型	0：任意仓 1：昨仓 2：今买卖仓 3：今申赎仓 转出时支持所有类型；转入不支持任意仓

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
```



```
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

6.3.62. OnRtnTransferFund 方法

资金转移通知。交易托管系统返回响应时，该方法会被调用

函数原形：

```
void OnRtnTransferFund(
                    CZQThostFtdcTransferFundField *pTransferFund) ;
```

参数：

pTransferFund：指向资金转移通知结构的地址。

资金转移回报结构：

字段名称	属性	描述
CTPFundSerial	CTP 资金流水号	
ApplySerial	申请流水号	
FrontID	前置编号	
SessionID	会话编号	
BrokerID	经纪公司代码	
AccoutID	资金账户代码	CTP 系统的资金账户
CurrencyID	币种	
TransferDirection	转账方向	0：转入 CTP， 1：转出 CTP
Amount	出入金额	
TransferStatus	转移状态	0：正常；1：已冲正
OperatorID	操作人员	
OperateDate	操作日期	
OperateTime	操作时间	

注：只返回操作来源为“钱券同步”的操作记录

6.3.63. OnRtnTransferPosition 方法

仓位转移回报通知。交易托管系统返回响应时，该方法会被调用

函数原形：

```
void OnRtnTransferPosition(
        CZQThostFtdcTransferPositionField *pTransferPosition) ;
```

参数：

pTransferPosition：指向仓位转移通知结构的地址。

仓位转移回报结构：

字段名称	属性	描述
CTPPositionSerial	CTP 仓位转移流水号	
ApplySerial	申请流水号	
FrontID	前置编号	
SessionID	会话编号	
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
ExchangeID	交易所代码	
ClientID	客户代码	
InstrumentID	合约代码	
TradingDay	交易日	
TransferDirection	转账方向	0：转入 CTP；1：转出 CTP
TransferPositionType	转移持仓类型	0：任意仓 1：昨仓 2：今买卖仓 3：今申赎仓
HistoryVolume	昨日仓位数量	昨日仓位转入转出数量
TodayBSVolume	今日买卖仓位数量	今日买卖仓位当日转入转出数量
TodayPRVolume	今日申赎仓位数量	今日申赎仓位当日转入转出数量

TransferStatus	转移状态	0：正常；1：已冲正
OperatorID	操作人员	
OperateDate	操作日期	
OperateTime	操作时间	

6.3.64. OnRspQryExerciseAssignment 方法

行权指派查询响应。当客户端发出行权指派查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryExerciseAssignment(
        CZQThostFtdcExerciseAssignmentField *pExerciseAssignment,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast) ;
```

参数：

pExerciseAssignment：指向行权指派结构的地址。

行权指派查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
ClientID	客户代码	
TradingDay	交易日	
ExchangeID	交易所代码	
InstrumentID	合约代码	
AssignmentVolume	行权指派数量	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CZQThostFtdcRspInfoField
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.65. OnRspQrySecurityTransfer 方法

现券转移查询响应。当客户端发出现券转移查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQrySecurityTransfer(
                CZQThostFtdcSecurityTransferField    *pSecurityTransfer,
                CZQThostFtdcRspInfoField *pRspInfo, int nRequestID,
                bool bIsLast);
```

参数：

pSecurityTransfer：指向现券转移结构的地址。

现券转移查询响应结构：

字段名称	属性	描述
SecurityTransferSerial	现券转移流水号	
ApplySerial	申请流水号	

SecurityPositionSerial	证券系统仓位转移流水号	
FrontID	前置编号	
SessionID	会话编号	
BrokerID	经纪公司代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
SecurityInvestorID	现券系统投资者代码	
SecurityInvestUnitID	现券系统投资单元代码	
ExchangeID	交易所代码	
ClientID	客户代码	
InstrumentID	合约代码	
SecurityTransferDirection	转移方向	0：转入现券系统；1：转出现券系统
TransferPositionType	转移持仓类型	0：任意仓 1：昨仓 2：今买卖仓 3：今申赎仓
HistoryVolume	昨日仓位数量	昨日仓位转入转出数量
TodayBSVolume	今日买卖仓位数量	今日买卖仓位当日转入转出数量
TodayPRVolume	今日申赎仓位数量	今日申赎仓位当日转入转出数量
SecurityTransferStatus	转移状态	显示报单目前的处理状态。1=初始状态; 2=转移成功; 3=转移失败; 4=报盘失败
StatusMsg	状态信息	
OperatorID	操作人员	
OperateDate	操作日期	
OperateTime	操作时间	

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CZQThostFtdcRspInfoField

```
{
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。

bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

6.3.66. OnRspQryLOFInfo 方法

上市开放式基金信息查询响应。当客户端发出上市开放式基金信息查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryLOFInfo(
        CZQThostFtdcLOFInfoField *pLOFInfo,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pLOFInfo：指向上市开放式基金信息结构的地址。

上市开放式基金信息结构：

字段名称	属性	描述
TradingDay	交易日	
ExchangeID	交易所代码	
FundID	基金代码	
FundTypeID	基金类型	
SMTradingUnit	拆分合并交易单位	
MasterRatio	母基金转换系数	

SplitVolMin	拆分申报最小数量	
MergeVolMin	合并申报最小数量	
IsSupportPur	是否允许申购	
IsSupportRed	是否允许赎回	
IsSupportSplit	是否允许拆分	
IsSupportMerge	是否允许合并	

6.3.67. OnRspQryLOFSubInfo 方法

上市开放式基金子基金信息查询响应。当客户端发出上市开放式基金子基金信息查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryLOFSubInfo(
        CZQThostFtdcLOFSubInfoField *pLOFSubInfo,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pLOFSubInfo：指向上市开放式基金子基金信息结构的地址。

上市开放式基金子基金信息结构：

字段名称	属性	描述
TradingDay	交易日	
ExchangeID	交易所代码	
MasterFundID	母基金代码	
SubFundID	子基金代码	
SubRatio	子份额转换系数	

6.3.68. OnRspCashRepayInsert 方法

直接还款录入响应。当客户端发出直接还款录入后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspCashRepayInsert(
        CZQThostFtdcInputCashRepayField *pInputCashRepay,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pInputCashRepay：指向直接还款录入响应结构的地址。

直接还款录入响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
CROrderRef	直接还款引用	
AccountID	资金账户代码	
Amount	申报还款金额	
InvestUnitID	投资单元代码	
RequestID	请求编号	

6.3.69. OnRtnCashRepay 方法

直接还款通知。交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRtnCashRepay(CZQThostFtdcCashRepayField *pCashRepay);
```

参数：

pCashRepay：指向直接还款通知结构的地址。

直接还款通知结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
CROrderRef	直接还款引用	
AccountID	资金账户代码	
Amount	申报还款金额	
InvestUnitID	投资单元代码	
RequestID	请求编号	
RepaidAmount	实际还款金额	
UserID	用户代码	
CROrderSysID	直接还款报单编号	
FrontID	前置编号	
SessionID	会话编号	
InsertDate	申报日期	
InsertTime	申报时间	
UserProductInfo	用户端产品信息	

6.3.70. OnErrRtnCashRepayInsert 方法

直接还款录入错误通知。交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnErrRtnCashRepayInsert(
        CZQThostFtdcInputCashRepayField *pInputCashRepay,
        CZQThostFtdcRspInfoField *pRspInfo);
```

参数：

pInputCashRepay：指向直接还款录入错误通知结构的地址。

直接还款录入错误通知结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
CROrderRef	直接还款引用	
AccountID	资金账户代码	
Amount	申报还款金额	
InvestUnitID	投资单元代码	
RequestID	请求编号	

6.3.71. OnRspQryCashRepay 方法

融资融券直接还款信息查询响应。当客户端发出融资融券直接还款信息查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryCashRepay(
        CZQThostFtdcCashRepayField *pCashRepay,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pCashRepay：指向融资融券直接还款信息结构的地址。

融资融券直接还款信息查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
CROrderRef	直接还款引用	
AccountID	资金账户代码	
Amount	申报还款金额	
InvestUnitID	投资单元代码	

RequestID	请求编号	
RepaidAmount	实际还款金额	
UserID	用户代码	
CROrderSysID	直接还款报单编号	
FrontID	前置编号	
SessionID	会话编号	
InsertDate	申报日期	
InsertTime	申报时间	
UserProductInfo	用户端产品信息	

6.3.72. OnRspQryCreditInfo 方法

投资者融资融券信息查询响应。当客户端发出投资者融资融券信息查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryCreditInfo(
        CZQThostFtdcCreditInfoField *pCreditInfo,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pCreditInfo：指向投资者融资融券信息结构的地址。

投资者融资融券信息查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
CreditAmount	授信总额度	
MarginBuyLimitAmount	融资额度上限	
MarginBuyAmount	融资买入金额	

NoTradeMarginBuyAmount	融资买入未成交金额	
MarginBuyFrozenMargin	融资冻结保证金	
MarginBuyInterest	融资买入利息	
ShortSellLimitAmount	融券额度上限	
ShortSellAmount	融券卖出金额	
NoTradeShortSellAmount	融券卖出未成交金额	
ShortSellFrozenMargin	融券冻结保证金	
ShortSellInterest	融券卖出利息	

6.3.73. OnRspQryCreditSecurityPosition 方法

投资者融资融券可用证券头寸查询响应。当客户端发出投资者融资融券可用证券头寸查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryCreditSecurityPosition(
    CZQThostFtdcCreditSecurityPositionField *pCreditSecurityPosition,
    CZQThostFtdcRspInfoField *pRspInfo,
    int nRequestID,    bool bIsLast);
```

参数：

pCreditSecurityPosition：指向投资者融资融券可用证券头寸结构的地址。

投资者融资融券可用证券头寸查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
PositionGroupID	头寸编号	
InvestorID	投资者代码	
ExchangeID	交易所代码	
InstrumentID	合约代码	
CreditPositionType	头寸类型	

Volume	可用数量	

6.3.74. OnRspQryCreditFundPosition 方法

投资者融资融券可用资金头寸查询响应。当客户端发出投资者融资融券可用资金头寸查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryCreditFundPosition(
        CZQThostFtdcCreditFundPositionField *pCreditFundPosition,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pCreditFundPosition：指向投资者融资融券可用资金头寸结构的地址。

投资者融资融券可用资金头寸查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
PositionGroupID	头寸编号	
InvestorID	投资者代码	
CreditPositionType	头寸类型	
Amount	可用资金	

6.3.75. OnRspQryCreditContract 方法

投资者信用合约查询响应。当客户端发出投资者信用合约查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryCreditContract(
```

CZQThostFtdcCreditContractField *pCreditContract,

CZQThostFtdcRspInfoField *pRspInfo,

int nRequestID, bool bIsLast);

参数：

pCreditContract：指向投资者信用合约结构的地址。

投资者信用合约查询响应结构：

字段名称	属性	描述
TradingDay	交易日	
BrokerID	经纪公司代码	
CreditContractID	信用合约代码	
InvestorID	投资者代码	
InvestUnitID	投资单元代码	
ClientID	客户代码	
ExchangeID	交易所代码	
InstrumentID	合约代码	
OpenDate	开仓日期	
ExpireDate	到期日	
CreditContractType	信用合约类型	
AveragePrice	成交均价	
Volume	数量	
Amount	金额	
LeavesVolume	未偿还数量	
LeavesAmount	未偿还金额	
LeavesInterest	未结利息	
LeavesFee	未结交易费用	
CreditContractStatus	信用合约状态	
CreditPositionType	头寸类型	

6.3.76. OnRspQryCreditRepayment 方法

投资者融资融券偿还明细查询响应。当客户端发出投资者融资融券偿还明细查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryCreditRepayment(
        CZQThostFtdcCreditRepaymentField *pCreditRepayment,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pCreditRepayment：指向投资者融资融券偿还明细结构的地址。

投资者融资融券偿还明细查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
CreditContractID	信用合约代码	
RepaymentID	偿还流水号	
OrderSysID	报单编号	
RepaidAmount	偿还金额	
RepaidVolume	偿还数量	
RepaidFee	偿还交易费用	
RepaidInterest	偿还利息	

6.3.77. OnRspQryCreditUnderlying 方法

投资者融资融券标的证券信息查询响应。当客户端发出投资者融资融券标的证券信息查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryCreditUnderlying(
        CZQThostFtdcCreditUnderlyingField *pCreditUnderlying,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pCreditUnderlying：指向投资者融资融券标的证券信息结构的地址。

投资者融资融券标的证券信息查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
ExchangeID	交易所代码	
InstrumentID	合约代码	
IsMarginBuy	可否融资	
IsShortSell	可否融券	
MarginBuyRate	融资保证金率	
ShortSellRate	融券保证金率	

6.3.78. OnRspQryCreditCollateral 方法

投资者融资融券担保证券信息查询响应。当客户端发出投资者融资融券担保证券信息查询后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryCreditCollateral(
        CZQThostFtdcCreditCollateralField *pCreditCollateral,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,    bool bIsLast);
```

参数：

pCreditCollateral：指向投资者融资融券担保证券信息结构的地址。

投资者融资融券担保证券信息查询响应结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
ExchangeID	交易所代码	
InstrumentID	合约代码	
IsCollateral	是否担保品	
CollateralConversionRate	担保品折算率	

6.3.79. OnRtnTradingNotice 方法

交易通知。交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRtnTradingNotice(
        CZQThostFtdcTradingNoticeInfoField *pTradingNoticeInfo);
```

参数：

pTradingNoticeInfo：指向交易通知结构的地址。

交易通知结构：

字段名称	属性	描述
BrokerID	经纪公司代码	
InvestorID	投资者代码	
SendTime	发送时间	
Content	消息正文	
SequenceSeries	序列系列号	
SequenceNo	序列号	

6.4. CZQThostFtdcTraderApi 接口

CZQThostFtdcTraderApi 接口提供给用户的功能包括，报单与报价的录入、报单与报价的撤销、报单与报价的挂起、报单与报价的激活、报单与报价的修改、

报单与报价的查询、成交单查询、会员客户查询、会员持仓查询、客户持仓查询、合约查询、出入金查询、通知等功能。

6.4.1. CreateFtdcTraderApi 方法

产生一个 CZQThostFtdcTradeApi 的一个实例，不能通过 new 来产生。

函数原形：

static CZQThostFtdcTradeApi *CreateFtdcTradeApi(const char *pszFlowPath = "");

参数：

pszFlowPath：常量字符指针，用于指定一个文件目录来存贮交易托管系统发布消息的状态。 默认值代表当前目录。

返回值：

返回一个指向 CZQThostFtdcTradeApi 实例的指针。

6.4.2. Release 方法

释放一个 CZQThostFtdcTradeApi 实例。不能使用 delete 方法

函数原形：

void Release();

6.4.3. Init 方法

使客户端开始与交易托管系统建立连接，连接成功后可以进行登陆。

函数原形：

void Init();

6.4.4. Join 方法

客户端等待一个接口实例线程的结束。

函数原形：

void Join();

6.4.5. RegisterSpi 方法

注册一个派生自 CZQThostFtdcTraderSpi 接口类的实例，该实例将完成事件处理。

函数原形：

void RegisterSpi(CZQThostFtdcTraderSpi *pSpi) ;

参数：

pSpi：实现了 CZQThostFtdcTraderSpi 接口的实例指针。

6.4.6. RegisterFront 方法

设置交易托管系统的网络通讯地址，交易托管系统拥有多个通信地址，但用户只需要选择一个通信地址。

函数原形：

void RegisterFront(char *pszFrontAddress);

参数：

pszFrontAddress：指向后台服务器地址的指针。服务器地址的格式为：“protocol://ipaddress:port”，如：”tcp://127.0.0.1:17001”。 “tcp”代表传输协议，“127.0.0.1”代表服务器地址。”17001”代表服务器端口号。

6.4.7. SubscribePrivateTopic 方法

订阅私有流。该方法要在 Init 方法前调用。若不调用则不会收到私有流的数据。

函数原形：

void SubscribePrivateTopic(TE_RESUME_TYPE nResumeType);

参数：

nResumeType： 私有流重传方式

TERT_RESTART:从本交易日开始重传

TERT_RESUME:从上次收到的续传

TERT_QUICK:只传送登录后私有流的内容

6.4.8. SubscribePublicTopic 方法

订阅公共流。该方法要在 Init 方法前调用。若不调用则不会收到公共流的数据。

函数原形：

void SubscribePublicTopic(TE_RESUME_TYPE nResumeType);

参数：

nResumeType： 公共流重传方式

TERT_RESTART:从本交易日开始重传

TERT_RESUME:从上次收到的续传

TERT_QUICK:只传送登录后公共流的内容

6.4.9. GetApiVersion 方法

获取 API 版本号。

函数原形：

static const char* GetApiVersion();

参数：

无

返回值：

返回 API 版本号。

6.4.10. ReqAuthenticate 方法

用户发出客户端认证请求。

函数原形：

int ReqAuthenticate(CZQThostFtdcReqAuthenticateField *pReqAuthenticateField,
int nRequestID);

参数：

pReqAuthenticateField：指向客户端认证请求结构的地址。

客户端认证请求结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
UserID	用户代码	必填
UserProductInfo	用户端产品信息	必填
AuthCode	认证码	必填

nRequestID：用户客户端认证请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcReqAuthenticateField reqAuthenticate;

//必须填写字段

strcpy(reqAuthenticate.BrokerID, "2011");
strcpy(reqAuthenticate.UserID, "2019001001");
strcpy(reqAuthenticate.UserProductInfo, "SFITTrader");
strcpy(reqAuthenticate.AuthCode, "12345");


int nResult =g_pTradeStockApi->ReqAuthenticate(&reqAuthenticate,0);
```

6.4.11. ReqUserLogin 方法

用户发出登陆请求。

函数原形：

int ReqUserLogin(
CZQThostFtdcReqUserLoginField *pReqUserLoginField,

int nRequestID)；

参数：

pReqUserLoginField：指向用户登录请求结构的地址。

用户登录请求结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
UserID	用户代码	必填
Password	密码	必填
UserProductInfo	用户端产品信息	必填
InterfaceProductInfo	接口端产品信息	无需填写，由 API 进行填充
ProtocolInfo	协议信息	无需填写，由 API 进行填充
MacAddress	Mac 地址	无需填写，由 API 进行填充
OneTimePassword	动态密码	备用。目前无需填写
ClientIPAddress	IP 地址	无需填写，由 API 进行填充
Lang	接口语言	目前可以接受的是“zh”（中文）和“en”（英文）。如果不填，则默认为 zh。这个值会影响返回结果中的文字信息，包括但不限于：RspInfo 中的 ErrorMsg。

nRequestID：用户登录请求的 ID，该 ID 由用户指定，管理。

用户需要填写 UserProductInfo 字段，即客户端的产品信息，如软件开发商、版本号等，例如：SFITTrader。

InterfaceProductInfo 和 ProtocolInfo 只须占位，不必有效赋值。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcReqUserLoginField qryField;
//必须填写字段
```

```
strcpy(qryField.BrokerID, "8000 ");
strcpy(qryField.UserID, "8000_admin ");
strcpy(qryField.Password, "1 ");
//其余字段选填
int nResult =g_pTradeStockApi->ReqUserLogin(&qryField,0);
```

6.4.12. ReqUserLogout 方法

用户发出登出请求。

函数原形：

```
int ReqUserLogout(
                    CZQThostFtdcUserLogoutField *pUserLogout,
                    int nRequestID);
```

参数：

pReqUserLogout：指向用户登出请求结构的地址。

用户登出请求结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
UserID	用户代码	必填

nRequestID：用户登出请求的 ID，该 ID 由用户指定，管理。

返回值：

0,代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcUserLogoutField qryField;
//必须填写字段
strcpy(qryField.BrokerID, "8000 ");
strcpy(qryField.UserID, "8000_admin ");
int nResult =g_pTradeStockApi-> ReqUserLogout(&qryField,0);
```


6.4.13. ReqUserPasswordUpdate 方法

用户密码修改请求。

函数原形：

```
int ReqUserPasswordUpdate(
                        CZQThostFtdcUserPasswordUpdateField
                        *pUserPasswordUpdate,
                        int nRequestID);
```

参数：

pUserPasswordUpdate：指向用户口令修改结构的地址。

用户口令修改结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
UserID	用户代码	必填
OldPassword	原来的口令	如果修改的是登陆用户的口令，则必填，否则不填
NewPassword	新的口令	必填

nRequestID：用户操作请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcUserPasswordUpdateField qryField;
strcpy(qryField.BrokerID, "8000 ");
strcpy(qryField.UserID, "8000_admin ");
strcpy(qryField.OldPassword, "1 ");
strcpy(qryField.NewPassword, "2 ");
int nResult =  =g_pTradeStockApi ->ReqUserPasswordUpdate(&qryField,0);
```


6.4.14. ReqQryMarginRate 方法

投资者保证金率查询。

函数原形：

```
int ReqQryMarginRate(
                    CZQThostFtdcQryMarginRateField *pQryMarginRate,
                    int nRequestID) ;
```

参数：

pQryMarginRate：指向投资者保证金率查询结构的地址。

投资者保证金率查询

字段名称	属性	描述
BrokerID	经纪公司代码	必填
MarginTemplateNo	保证金模板代码	可选
ExchangeID	交易所代码	可选
InstrumentID	合约代码	可选

nRequestID：用户操作请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcQryMarginRateField qryField;


strcpy(qryField.BrokerID, "8000 ");

strcpy(qryField. ExchangeID, "SSE");

strcpy(qryField. InstrumentID, "000001");

int iResult = =g_pTradeStockApi->ReqQryMarginRate (&qryField,0));
```

6.4.15. ReqOrderInsert 方法

客户端发出报单录入请求。

函数原形：

```
int ReqOrderInsert(
                    CThostFtdcInputOrderField *pInputOrder,
                    int nRequestID);
```

参数：

pInputOrder：指向输入报单结构的地址。

输入报单结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
InstrumentID	合约代码	必填
OrderRef	报单引用	可选
UserID	用户代码	不填（暂不使用）
OrderPriceType	报单价格条件	可选。 1=任意价 2=限价 3=最优价 4=最新价 5=最新价浮动上浮 1 个 ticks …… F=买一价浮动上浮 3 个 ticks G=五档价 a=本方最优
Direction	买卖方向	必填。 0=买入 1=卖出 2=ETF 申购 3=ETF 赎回 4=新股申购 5=正回购 6=逆回购 7=转托管 8=开放式基金申购 9=开放式基金赎回 a=担保品划入 b=担保品划出

		d=质押入库 e=质押出库 f=配股配债 g=开放式基金拆分 h=开放式基金合并 i=融资买入 j=融券卖出 k=卖券还款 l=买券还券 m=还券划转 n=余券划转
CombOffsetFlag	组合开平标志	第一位是开平标志： 0=开仓 1=平仓 2=强平 3=平今 4=平昨 5=强减 6=本地强平
CombHedgeFlag	组合投机套保标志	第一位是投机套保标志：（除备兑外都写投机） 1=投机 2=套利 3=套保 4=备兑
LimitPrice	价格	可选（对应“申报价格”）
VolumeTotalOriginal	数量	必填
TimeCondition	有效期类型	可选。 1=立即完成，否则撤销 2=本节有效 3=当日有效 4=指定日期前有效 5=撤销前有效 6=集合竞价有效
GTDDate	GTD 日期	不填（暂不使用）
VolumeCondition	成交量类型	可选。 1=任何数量 2=最小数量 3=全部数量
MinVolume	最小成交量	不填（暂不使用）
ContingentCondition	触发条件	不填（暂不使用）
StopPrice	止损价	不填（暂不使用）
ForceCloseReason	强平原因	不填（暂不使用）
IsAutoSuspend	自动挂起标志	不填（暂不使用）

BusinessUnit	业务单元	不填（暂不使用）
RequestID	请求编号	可选
UserForceClose	用户强平标志	经纪公司发起强平时使用。投资者用户固定输入 false。
IsSwapOrder	互换单标志	不填（暂不使用）
ExchangeID	交易所代码	必填
ClientID	客户代码	必填。对应“交易账户代码”
InvestUnitID	投资单元代码	可选
AccountID	资金帐号代码	可选
IPAddress	IP 地址	可选
MacAddress	Mac 地址	可选
CreditPositionType	头寸类型	可选。 0=普通 1=专项

nRequestID：用户报单请求的 ID，该 ID 由用户指定，管理。在一次会话中，该 ID 不能重复。

OrderRef：报单引用，只能单调递增。每次登入成功后，可以从 OnRspUserLogin 的输出参数 CThostFtdcRspUserLoginField 中获得上次登入用过的最大 OrderRef, MaxOrderRef。

其中，交易所的价格申报类型与 CTP 报单接口字段的对应关系如下：

1、现货

价格申报类型	报单价格类型 OrderPriceType	有效期类型 TimeCondition	成交量类型 VolumeCondition
限价（沪、深）	限价	当日有效	任何数量
本方最优（深）	本方最优	当日有效	任何数量
对手方最优（深）	最优价	当日有效	任何数量
立即成交剩余转撤销（深）	任意价	立即成交，否则撤销	任何数量
全额成交或撤销（深）	任意价	立即成交，否则撤销	全部数量
最优五档成交剩余转撤销（沪、深）	五档价	立即成交，否则撤销	任何数量
最优五档成交剩余转限价（沪）	五档价	当日有效	任何数量

2、个股期权

价格申报类型	报单价格类型 OrderPriceType	有效期类型 TimeCondition	成交量类型 VolumeCondition
限价（沪、深）	限价	当日有效	任何数量
本方最优（深）	本方最优	当日有效	任何数量
对手方最优（深）	最优价	当日有效	任何数量

最优五档成交剩余转撤销（深）	五档价	立即成交，否则撤销	任何数量
市价剩余转限价（沪）	任意价	当日有效	任何数量
市价剩余转撤销（沪、深）	任意价	立即成交，否则撤销	任何数量
限价立即全部成交否则自动撤销（沪、深）	限价	立即成交，否则撤销	全部数量
市价立即全部成交否则自动撤销（沪、深）	任意价	立即成交，否则撤销	全部数量

返回值：

0，代表成功；

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

请参考交易 API 开发示例。

6.4.16. ReqOrderAction 方法

客户端发出报单操作请求，包括报单的撤销、报单的挂起、报单的激活、报单的修改。

函数原形：

```
int ReqOrderAction(
                CZQThostFtdcOrderActionField *pInputOrderAction,
                int nRequestID);
```

参数：

pInputOrderAction：指向报单操作结构的地址。

报单操作结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
OrderActionRef	报单操作引用	可选
OrderRef	报单引用	报单引用、前置编号、会话编号，3个字段需一起填写，否则就都不填
RequestID	请求编号	可选
FrontID	前置编号	
SessionID	会话编号	
ExchangeID	交易所代码	必填
OrderSysID	报单编号	
ActionFlag	操作标志	撤单时填'0' 0=删除 3=修改
LimitPrice	价格	证券暂时不用
VolumeChange	数量变化	证券暂时不用
UserID	用户代码	证券暂时不用
InstrumentID	合约代码	证券暂时不用
IPAddress	IP 地址	可选
MacAddress	Mac 地址	可选

nRequestID：用户报单操作请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//从委托复制字段到撤单中，必须从可以撤单的委托中复制,fldOrder 为可撤报单
CZQThostFtdcInputOrderActionField fldAction;
// 身份参数
strcpy(fldAction.BrokerID, "8000 ");
strcpy(fldAction.UserID, "8000_admin ");
// 关键字
fldAction.FrontID = fldOrder.FrontID;
```



```
fldAction.SessionID = fldOrder.SessionID;
strcpy(fldAction.OrderRef, fldOrder.OrderRef);
strcpy(fldAction.ExchangeID, fldOrder.ExchangeID);
// 其他
strcpy(fldAction.BrokerID, fldOrder.BrokerID);
strcpy(fldAction.UserID, fldOrder.UserID);
strcpy(fldAction.InvestorID, fldOrder.InvestorID);
strcpy(fldAction.InstrumentID, fldOrder.InstrumentID);
//证券中必须填 OrderLocalID，TradeId
strcpy(fldAction.OrderLocalID,fldOrder.OrderLocalID);
strcpy(fldAction.TraderID,fldOrder.TraderID);
fldAction.ActionFlag = ZQTHOST_FTDC_AF_Delete;
int iReq = g_pTradeStockApi->ReqOrderAction(&fldAction,0);
```

6.4.17. ReqQryOrder 方法

报单查询请求。

函数原形：

```
int ReqQryOrder(
        CZQThostFtdcQryOrderField *pQryOrder,
        int nRequestID);
```

参数：

pQryOrder：指向报单查询结构的地址。

报单查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选

OrderSysID	报单编号	可选
InsertTimeStart	开始时间	可选
InsertTimeEnd	结束时间	可选
InvestUnitID	投资单元代码	可选

nRequestID：用户报单查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 查询全部委托
    CZQThostFtdcQryOrderField qryField;
    int iReq = g_pTradeStockApi->ReqQryOrder(&qryField, 0);
//查询投资者 000001 的委托
    CZQThostFtdcQryOrderField qryField;
    strcpy(qryField.BrokerID, "8000 ");
    strcpy(qryField.InvestorId, "000001");
    int iReq = g_pTradeStockApi->ReqQryOrder(&qryField, 0);
```

6.4.18. ReqQryTrade 方法

成交单查询请求。

函数原形：

```
int ReqQryTrade(
                CZQThostFtdcQryTradeField *pQryTrade,
                int nRequestID);
```

参数：

pQryTrade：指向成交查询结构的地址。

成交查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
TradeID	成交编号	可选
TradeTimeStart	开始时间	可选
TradeTimeEnd	结束时间	可选
InvestUnitID	投资单元代码	可选

nRequestID：用户成交单查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

```
// 查询全部成交

    CZQThostFtdcQryTradeField qryField;

    int iReq = g_pTradeStockApi-> ReqQryTrade (&qryField, 0);

//查询投资者 000001 的成交

    CZQThostFtdcQryTradeField qryField;

    strcpy(qryField.BrokerID, "8000 ");

    strcpy(qryField.InvestorId, "000001");

    int iReq = g_pTradeStockApi-> ReqQryTrade (&qryField, 0);
```

6.4.19. ReqQryInvestor 方法

会员客户查询请求。

函数原形：

int ReqQryInvestor (

CZQThostFtdcQryInvestorField *pQryInvestor,
int nRequestID);

参数：

pQryInvestor：指向客户查询结构的地址。

客户查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选

nRequestID：用户客户查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 查询所有投资者
    CZQThostFtdcQryInvestorField qryField;
    int iReq = g_pTradeStockApi-> ReqQryInvestor (&qryField, 0);
//查询投资者 000001
    CZQThostFtdcQryTradeField qryField;
    strcpy(qryField.BrokerID, "8000 ");
    strcpy(qryField.InvestorId, "000001");
    int iReq = g_pTradeStockApi-> ReqQryInvestor (&qryField, 0);
```

6.4.20. ReqQryInvestorSecurityPosition 方法

会员持仓查询请求。

函数原形：

```
int ReqQryInvestorSecurityPosition(
        CZQThostFtdcQryInvestorSecurityPositionField    *pQryInvestorSecurityPosition,
```

int nRequestID);

参数：

pQryInvestorSecurityPosition：指向会员持仓查询结构的地址。

查询投资者证券持仓

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
InvestUnitID	投资单元代码	可选

nRequestID：会员持仓查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 查询所有投资者持仓
    CZQThostFtdcQryInvestorSecurityPositionField qryField;
    int iReq = g_pTradeStockApi-> ReqQryInvestorSecurityPosition(&qryField, 0);
//查询投资者 000001 的持仓
    CZQThostFtdcQryInvestorSecurityPositionField qryField;
    strcpy(qryField.BrokerID, "8000 ");
    strcpy(qryField.InvestorId, "000001");
    int iReq = g_pTradeStockApi-> ReqQryInvestorSecurityPosition(&qryField, 0);
```

6.4.21. ReqQryTradingAccount 方法

请求查询资金账户。

函数原形：

```
int ReqQryTradingAccount(
                CZQThostFtdcQryTradingAccountField *pQryTradingAccount,
                int nRequestID);
```

参数：

pQryTradingAccount：指向查询资金账户结构的地址。

查询资金账户结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
CurrencyID	币种代码	可选
AccountID	资金账户代码	可选

nRequestID：会员持仓查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 查询所有资金账户
    CZQThostFtdcQryTradingAccountField qryField;
    int iReq = g_pTradeStockApi-> ReqQryTradingAccount(&qryField, 0);
//查询投资者 000001 的资金账户
    CZQThostFtdcQryTradingAccountField qryField;
    strcpy(qryField.BrokerID, "8000 ");
    strcpy(qryField.InvestorId, "000001");
    int iReq = g_pTradeStockApi-> ReqQryTradingAccount(&qryField, 0);
```


6.4.22. ReqQryTradingCode 方法

请求查询交易编码。

函数原形：

```
int ReqQryTradingCode(
                CZQThostFtdcQryTradingCodeField *pQryTradingCode,
                int nRequestID);
```

参数：

pQryTradingCode：指向查询交易编码结构的地址。

查询交易编码结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
ExchangeID	交易所代码	可选
ClientID	客户代码	可选
ClientIDType	交易编码类型	可选。 1=投机 2=套利 3=套保 a=普通 b=信用 c=衍生品

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 查询所有投资者交易编码

    CThostFtdcQryTradingCodeField qryField;

    int iReq = g_pTradeStockApi-> ReqQryTradingCode(&qryField, 0);
```



```
//查询投资者 000001 在 SSE 的交易编码
    CThostFtdcQryTradingCodeField qryField;
    strcpy(qryField.BrokerID, "8000 ");
    strcpy(qryField.InvestorId, "000001");
    strcpy(qryField.ExchangeId, "SSE");
    set_null (qryField. ClientID);
   int iReq = g_pTradeStockApi-> ReqQryTradingCode(&qryField, 0);
```

6.4.23. ReqQryExchange 方法

请求查询交易所。

函数原形：

```
int ReqQryExchange(
        CZQThostFtdcQryExchangeField *pQryExchange,
        int nRequestID);
```

参数：

pQryExchange：指向查询交易编码结构的地址。

查询交易所编码结构：

字段名称	属性	描述
ExchangeID	交易所代码	可选

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 查询所有交易所
    CZQThostFtdcQryExchangeField qryField;
    int iReq = g_pTradeStockApi-> ReqQryExchange(&qryField, 0);
// 查询 SSE 的信息
    CZQThostFtdcQryExchangeField qryField;
    strcpy(qryField.ExchangId, "SSE");
   int iReq = g_pTradeStockApi-> ReqQryExchange(&qryField, 0);
```

6.4.24. ReqQrySecurity 方法

请求查询证券。

函数原形：

```
int ReqQrySecurity(
                CZQThostFtdcQryInstrumentField *pQrySecurity,
                int nRequestID);
```

参数：

pQrySecurity：指向查询证券结构的地址。

查询证券结构：

字段名称	属性	描述
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
ExchangeInstID	合约在交易所的代码	证券暂不使用
ProductID	产品代码	可选

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 查询所有合约
    CZQThostFtdcQryInstrumentField qryField;
    int iReq = g_pTradeStockApi-> ReqQrySecurity(&qryField, 0);
// 查询 SSE 的所有合约信息
    CZQThostFtdcQryInstrumentField qryField;
    strcpy(qryField.ExchangId, "SSE");
    int iReq = g_pTradeStockApi-> ReqQrySecurity (&qryField, 0);
```

6.4.25. ReqQryDepthMarketData 方法

请求查询行情。

函数原形：

```
int ReqQryDepthMarketData(
        CZQThostFtdcQryDepthMarketDataField *pQryDepthMarketData,
        int nRequestID);
```

参数：

pQryDepthMarketData：指向查询行情结构的地址。

查询行情结构：

字段名称	属性	描述
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//适用查询单个合约，不要查询所有合约的行情
//所有合约的行情，请订阅行情
    CZQThostFtdcQryDepthMarketDataField qryField;
    strcpy(qryField.InstrumentID, "000001");
    int iReq = g_pTradeStockApi-> ReqQryDepthMarketData(&qryField, 0);
```

6.4.26. ReqQryOrderFundDetail 方法

请求查询报单资金明细。

函数原形：

```
int ReqQryOrderFundDetail(
        CZQThostFtdcQryOrderFundDetailField * pQryOrderFundDetail,
        int nRequestID);
```

参数：

pQryOrderFundDetail：指向查询报单资金明细结构的地址。

查询报单资金明细结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
OrderSysID	报单编号	可选
InsertTimeStart	开始时间	可选
InsertTimeEnd	结束时间	可选
InvestUnitID	投资单元代码	可选

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 所有报单明细
    CZQThostFtdcQryOrderFundDetailField qryField;
    int iReq = g_pTradeStockApi-> ReqQryOrderFundDetail (&qryField, 0);
// 查询投资者 000001 的报单明细
    CZQThostFtdcQryOrderFundDetailField qryField;
    strcpy(qryField.BrokerID, "8000 ");
    strcpy(qryField.InvestorId, "000001");
    strcpy(qryField.ExchangId, "SSE");
    int iReq = g_pTradeStockApi-> ReqQryInvestorPositionDetail(&qryField, 0);
```

6.4.27. ReqQryMarket 方法

请求查询市场信息。

函数原形：

```
int   ReqQryMarket(
                    CZQThostFtdcQryMarketField *pQryMarket,
                    int nRequestID) ;
```

参数：

pQryMarket：指向查询市场信息结构的地址。

查询市场结构：

字段名称	属性	描述
ExchangeID	交易所代码	可选

MarketID	市场代码	可选

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 所有市场信息
    CZQThostFtdcQryBondInterestField qryField;
    int iReq = g_pTradeStockApi-> ReqQryBondInterest (&qryField, 0);
// 查询 010107 的债券利息
    CZQThostFtdcQryBondInterestField qryField;
    strcpy(qryField.ExchangId, "SSE");
    strcpy(qryField. InstrumentID, " 010107");
    int iReq = g_pTradeStockApi-> ReqQryBondInterest (&qryField, 0);
```

6.4.28. ReqQrySecurityProduct 方法

请求查询证券品种。

函数原形：

int ReqQrySecurityProduct(

CZQThostFtdcQrySecurityProductField *pQrySecurityProduct,

int nRequestID) ;

参数：

pQrySecurityProduct：指向证券品种请求的地址。

查询证券品种结构：

字段名称	属性	描述
ExchangeID	交易所代码	可选
ProductID	产品代码	可选

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 所有证券品种信息
    CZQThostFtdcQrySecurityProductField qryField;
    int iReq = g_pTradeStockApi-> ReqQrySecurityProduct (&qryField, 0);
// 查询 SSE 的证券品种利息
CZQThostFtdcQrySecurityProductField qryField;
strcpy(qryField.ExchangId, "SSE");
int iReq = g_pTradeStockApi-> ReqQrySecurityProduct (&qryField, 0);
```

6.4.29. ReqQrySecurityClass 方法

请求查询证券类别。

函数原形：

int ReqQrySecurityClass(

CZQThostFtdcQrySecurityClassField *pQrySecurityClass,

int nRequestID) ;

参数：

pQrySecurityClass：指向查询证券类别的地址。

查询证券类别结构：

字段名称	属性	描述
ExchangeID	交易所代码	可选
ProductID	产品代码	可选
SecurityClassID	证券类别代码	可选

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 所有证券类别记录
    CZQThostFtdcQrySecurityClassField qryField;
    int iReq = g_pTradeStockApi->ReqQrySecurityClass(&qryField, 0);
//// 查询 SSE 的证券类别信息
    CZQThostFtdcQryFundIOCTPAccountField qryField;
    strcpy(qryField.ExchangeId, "SSE");
    int iReq = g_pTradeStockApi-> ReqQryFundIOCTPAccount (&qryField, 0);
```

6.4.30. ReqQryInvestUnit 方法

请求查询投资单元。

函数原形：

```
int ReqQryInvestUnit(
                    CZQThostFtdcQryInvestUnitField    *pQryInvestUnit,
```

int nRequestID) ;

参数：

pQryInvestUnit：指向查询投资单元的地址。

查询投资单元结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 所有投资单元记录

    CZQThostFtdcQryInvestUnitField qryField;

    int iReq = g_pTradeStockApi-> ReqQryInvestUnit (&qryField, 0);
```

6.4.31. ReqQryBrokerage 方法

请求查询投资者佣金。

函数原形：

int ReqQryBrokerage(

CZQThostFtdcQryBrokerageField *pQryBrokerage,

int nRequestID) ;

参数：

pQryBrokerage：指向查询投资者佣金的地址。

查询投资者佣金结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
BrokerageTemplateNo	佣金模板代码	可选
ExchangeID	交易所代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 所有投资者佣金记录
    CZQThostFtdcQryBrokerageField qryField;
    int iReq = g_pTradeStockApi-> ReqQryBrokerage(&qryField, 0);
//// 查询 SSE 的投资者佣金信息
    CZQThostFtdcQryBrokerageField qryField;
    strcpy(qryField.BrokerId, "8000");
    strcpy(qryField.ExchangeId, "SSE");
    int iReq = g_pTradeStockApi-> ReqQryBrokerage (&qryField, 0);
```

6.4.32. ReqQrySubscribingSharesQuota 方法

请求查询新股申购额度

函数原形：

int ReqQrySubscribingSharesQuota(

CZQThostFtdcQrySubscribingSharesQuotaField *pQrySubscribingSharesQuota,

int nRequestID) ;

参数：

pQrySubscribingSharesQuota：指向查询新股额度的地址。

查询新股申购额度结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
ExchangeID	交易所代码	可选
ClientID	客户代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
// 所有新股额度记录
    CZQThostFtdcQrySubscribingSharesQuotaField qryField;
    int iReq = g_pTradeStockApi-> ReqQrySubscribingSharesQuota (&qryField, 0);
//// 查询投资者 0000001 的申购额度
    CZQThostFtdcQrySubscribingSharesQuotaField qryField;
    strcpy(qryField.BrokerId, "8000");
    strcpy(qryField.ExchangeId, "SSE");
    strcpy(qryField.InvestorId, "0000001");
    int iReq = g_pTradeStockApi-> ReqQrySubscribingSharesQuota (&qryField, 0);
```

6.4.33. ReqQrySubscribingShares 方法

请求查询新股信息

函数原形：

```
int ReqQrySubscribingShares(
        CZQThostFtdcQrySubscribingSharesField *pQrySubscribingShares,
        int nRequestID) ;
```

参数：

pQrySubscribingShares：指向查询新股信息的地址。

查询新股信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
ExchangeID	交易所代码	可选
InstrumentID	申购代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有新股信息
CZQThostFtdcQrySubscribingSharesField qryField;
int iReq = g_pTradeStockApi-> ReqQrySubscribingShares (&qryField, 0);
```

6.4.34. ReqQryPledgeOrderInfo 方法

请求查询质押申报代码

函数原形：

```
int ReqQryPledgeOrderInfo(
        CZQThostFtdcQryPledgeOrderInfoField *pQryPledgeOrderInfo,
        int nRequestID) ;
```

参数：

pQryPledgeOrderInfo：指向查询申报代码的地址。

查询质押申报代码信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
ExchangeID	交易所代码	可选
InstrumentID	合约代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有质押申报代码信息
CZQThostFtdcQryPledgeOrderInfoField qryField;
int iReq = g_pTradeStockApi-> ReqQryPledgeOrderInfo (&qryField, 0);
```

6.4.35. ReqQryInvestorPledgePosition 方法

请求查询质押持仓

函数原形：

int ReqQryInvestorPledgePosition(

CZQThostFtdcQryInvestorPledgePositionField *pQryInvestorPledgePosition,

int nRequestID);

参数：

pQryInvestorPledgePosition：指向查询质押持仓的地址。

查询投资者质押持仓结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
ClientID	客户代码	可选
InvestUnitID	投资单元代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有质押持仓信息

CZQThostFtdcQryInvestorPledgePositionField qryField;

int iReq = g_pTradeStockApi-> ReqQryInvestorPledgePosition (&qryField, 0);
```

6.4.36. ReqQryInvestorPledgeInfo 方法

请求查询质押回购信息。

函数原形：

int ReqQryInvestorPledgeInfo(

CZQThostFtdcQryInvestorPledgeInfoField *pQryInvestorPledgeInfo, int nRequestID);

参数：

pQryInvestorPledgeInfo：指向查询质押回购信息的地址。

查询投资者质押回购结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
StandardBondID	标准券代码	可选
ExchangeID	交易所代码	可选
ClientID	客户代码	可选
InvestUnitID	投资单元代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有质押回购信息

CZQThostFtdcQryInvestorPledgeInfoField qryField;

int iReq = g_pTradeStockApi-> ReqQryInvestorPledgeInfo (&qryField, 0);
```

6.4.37. ReqQryRepoRatio 方法

请求查询标准券使用率信息。

函数原形：

int ReqQryRepoRatio(

CZQThostFtdcQryRepoRatioField *pQryRepoRatio,

int nRequestID);

参数：

pQryRepoRatio：指向查询标准券使用率的地址。

查询标准券使用率结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
StandardBondID	标准券代码	可选
ExchangeID	交易所代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有标准券使用率信息
CZQThostFtdcQryRepoRatioField qryField;
int iReq = g_pTradeStockApi-> ReqQryRepoRatio (&qryField, 0);
```

6.4.38. ReqQryRepurchaseMaxTimes 方法

请求查询回购放大倍数信息。

函数原形：

int ReqQryRepurchaseMaxTimes(

CZQThostFtdcQryRepurchaseMaxTimesField * pQryRepurchaseMaxTimes,

int nRequestID);

参数：

pQryRepurchaseMaxTimes：指向查询回购放大倍数的地址。

查询回购放大倍数结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
ExchangeID	交易所代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有回购放大倍数信息

CZQThostFtdcQryRepurchaseMaxTimesField qryField;

int iReq = g_pTradeStockApi-> ReqQryRepurchaseMaxTimes (&qryField, 0)
```

6.4.39. ReqQryETFFILE 方法

请求查询 ETF 清单信息。

函数原形：

```
int ReqQryETFFile(
                    CZQThostFtdcQryETFFileField *pQryETFFile,
                    int nRequestID);
```

参数：

pQryETFFile：指向查询 ETF 清单的地址。

查询 ETF 清单结构：

字段名称	属性	描述

ExchangeID	交易所代码	可选
ETFInstrumentID	ETF 合约代码	可选
ETFCreRedInstrumentID	ETF 申赎代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有 ETF 清单信息

CZQThostFtdcQryETFFileField qryField;

int iReq = g_pTradeStockApi-> ReqQryETFFile (&qryField, 0);
```

6.4.40. ReqQryETFBasket 方法

请求查询 ETF 成分证券信息。

函数原形：

```
int ReqQryETFBasket(
                    CZQThostFtdcQryETFBasketField   *pQryETFBasket,
                    int nRequestID) ;
```

参数：

pQryETFBasket：指向查询 ETF 成分证券的地址。

查询 ETF 成分证券结构：

字段名称	属性	描述
ExchangeID	交易所代码	可选
ETFInstrumentID	ETF 合约代码	可选
InstrumentID	成分合约代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有 ETF 成分证券信息

CZQThostFtdcQryETFBasketField qryField;

int iReq = g_pTradeStockApi-> ReqQryETFBasket (&qryField, 0);
```

6.4.41. ReqQryTransferFund 方法

请求资金转移信息。

函数原形：

int ReqQryTransferFund(

CZQThostFtdcQryTransferFundField *pQryTransferFund,

int nRequestID);

参数：

pQryTransferFund：指向查询资金转移的地址。

查询资金转移结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
资金账户代码	资金账户代码	可选
CTPFundSerial	CTP 转账流水号	可选
FrontID	前置编号	可选
SessionID	会话编号	可选
ApplySerial	申请流水号	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有资金转移信息

CZQThostFtdcQryTransferFundField qryField;

int iReq = g_pTradeStockApi-> ReqQryTransferFund (&qryField, 0);
```

6.4.42. ReqQryTransferPosition 方法

请求查询证券转移信息。

函数原形：

```
int ReqQryTransferPosition(
                CZQThostFtdcQryTransferPositionField *pQryTransferPosition,
                int nRequestID);
```

参数：

pQryTransferPosition：指向查询证券转移的地址。

查询证券转移结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InvestUnitID	投资单元代码	可选
ExchangeID	交易所代码	可选
ClientID	客户代码	可选
InstrumentID	合约代码	可选
CTPPositionSerial	CTP 仓位转移流水号	可选

FrontID	前置编号	可选
SessionID	会话编号	可选
ApplySerial	申请流水号	可选

}返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有证券转移信息

CZQThostFtdcQryTransferPositionField qryField;

int iReq = g_pTradeStockApi-> ReqQryTransferPosition (&qryField, 0);
```

6.4.43. ReqQryInvestUnitAndTradingAcct 方法

请求查询投资单元与交易资金账户关系信息。

函数原形：

int ReqQryInvestUnitAndTradingAcct(

CZQThostFtdcQryInvestUnitAndTradingAcctField *pQryInvestUnitAndTradingAcct,

int nRequestID) ;

参数：

pQryInvestUnitAndTradingAcct：指向查询投资单元与交易资金账户关系的地址。

查询投资单元与交易资金账户关系结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填

InvestUnitID	投资单元代码	可选
ExchangeID	交易所代码	可选
ProductID	产品代码	可选
AccountID	资金账户代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有投资单元与交易资金账户关系信息
CZQThostFtdcQryInvestUnitAndTradingAcctField  qryField;
int iReq = g_pTradeStockApi-> ReqQryInvestUnitAndTradingAcct (&qryField, 0);
```

6.4.44. ReqQryInvestunitAndUser 方法

请求查询投资单元与交易用户关系信息。

函数原形：

int ReqQryInvestUnitAndUser(

CZQThostFtdcQryInvestUnitAndUserField *pQryInvestUnitAndUser,

int nRequestID) ;

参数：

pQryInvestUnitAndUser：指向查询投资单元与交易用户的地址。

查询投资者用户与投资单元关系结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填

UserID	用户代码	可选
InvestUnitID	投资单元代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有投资单元与交易用户关系信息

CZQThostFtdcQryInvestUnitAndUserField qryField;

int iReq = g_pTradeStockApi-> ReqQryInvestUnitAndUser (&qryField, 0);
```

6.4.45. ReqQryInstrument 方法

请求查询个股期权合约信息。

函数原形：

int ReqQryInstrument(

CZQThostFtdcQryInstrumentField *pQryInstrument,

int nRequestID) ;

参数：

pQryInstrument：指向查询投资单元与交易用户的地址。

个股期权合约查询结构：

字段名称	属性	描述
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
ExchangeInstID	合约在交易所的代码	证券暂不使用

ProductID	产品代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有个股期权合约信息
CZQThostFtdcQryInstrumentField qryField;
int iReq = g_pTradeStockApi-> ReqQryInstrument (&qryField, 0);
//查询上海证券交易所的合约信息
CZQThostFtdcQryInstrumentField qryField;
strcpy(qryField.ExchangeID,"SSE");
int iReq = g_pTradeStockApi-> ReqQryInstrument (&qryField, 0);
```

6.4.46. ReqQryInvestorPosition 方法

请求查询投资者合约持仓信息。

函数原形：

int ReqQryInvestorPosition(

CZQThostFtdcQryInvestorPositionField *pQryInvestorPosition,

int nRequestID) = 0;

参数：

pQryInvestorPosition：指向合约持仓条件的地址。

投资者合约持仓查询结构：

字段名称	属性	描述

BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
ClientID	客户代码	可选
InvestUnitID	投资单元代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有投资者合约持仓信息
CZQThostFtdcQryInvestorPositionField  qryField;
int iReq = g_pTradeStockApi-> ReqQryInvestorPosition (&qryField, 0);
//查询投资者上海证券交易所的合约信息
CZQThostFtdcQryInvestorPositionField qryField;
strcpy(qryField.ExchangeID,"SSE");
int iReq = g_pTradeStockApi-> ReqQryInvestorPosition (&qryField, 0);
```

6.4.47. ReqQryLockPosition 方法

请求查询投资者锁定持仓信息。

函数原形：

```
int ReqQryLockPosition(
                CZQThostFtdcQryLockPositionField *pQryLockPosition,
                int nRequestID) ;
```

参数：

pQryLockPosition：指向查询投资单元锁定持仓条件地址。

锁定持仓查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
ClientID	客户代码	可选
InvestUnitID	投资单元代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有锁定持仓信息

CZQThostFtdcQryLockPositionField qryField;

int iReq = g_pTradeStockApi-> ReqQryLockPosition (&qryField, 0);

//查询投资者 0000010 上海证券交易所的合约信息

CZQThostFtdcQryLockPositionField qryField;

strcpy(qryField. InvestorID," 0000010");

strcpy(qryField.ExchangeID,"SSE");

int iReq = g_pTradeStockApi-> ReqQryLockPosition (&qryField, 0);
```


6.4.48. ReqQryLock 方法

请求查询锁定解锁信息。

函数原形：

int ReqQryLock(

CZQThostFtdcQryLockField *pQryLock,

int nRequestID) ;

参数：

pQryLock：指向查询投资者锁定条件的地址。

锁定查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
LockSysID	锁定编号	可选
InsertTimeStart	开始时间	可选
InsertTimeEnd	结束时间	可选
InvestUnitID	投资单元代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

//查询所有投资者锁定信息


```
CZQThostFtdcQryLockField qryField;
int iReq = g_pTradeStockApi-> ReqQryLock (&qryField, 0);
```

6.4.49. ReqQryExecOrder 方法

请求查询执行宣告信息。

函数原形：

```
int ReqQryExecOrder(
                    CZQThostFtdcQryExecOrderField    *pQryExecOrder,
                    int nRequestID);
```

参数：

pQryExecOrder：指向查询执行宣告条件地址。

执行宣告查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InstrumentID	合约代码	可选
ExchangeID	交易所代码	可选
ExecOrderSysID	执行宣告编号	可选
InsertTimeStart	开始时间	可选
InsertTimeEnd	结束时间	可选
InvestUnitID	投资单元代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询所有执行宣告信息
CZQThostFtdcQryExecOrderField qryField;
int iReq = g_pTradeStockApi-> ReqQryExecOrder (&qryField, 0);
//查询上海证券交易所的执行宣告信息
CZQThostFtdcQryExecOrderField qryField;
strcpy(qryField.ExchangeID,"SSE");
int iReq = g_pTradeStockApi-> ReqQryExecOrder (&qryField, 0);
```

6.4.50. ReqQryLimitAmount 方法

请求查询投资者买入额度信息。

函数原形：

```
int ReqQryLimitAmount(
        CZQThostFtdcQryLimitAmountField *pQryLimitAmount,
        int nRequestID) ;
```

参数：

pQryLimitAmount：指向查询投资者买入额度的地址。

投资者买入额度查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
ExchangeID	交易所代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//投资者买入额度查询信息

CZQThostFtdcQryLimitAmountField qryField;

int iReq = g_pTradeStockApi-> ReqQryLimitAmount (&qryField, 0);

//查询上海证券交易所的投资者买入额度信息

CZQThostFtdcQryLimitAmountField qryField;

strcpy(qryField.ExchangeID,"SSE");

int iReq = g_pTradeStockApi-> ReqQryLimitAmount (&qryField, 0);
```

6.4.51. ReqQryLimitPosition 方法

请求查询投资者持仓限制信息。

函数原形：

```
int ReqQryLimitPosition (
        CZQThostFtdcQryLimitPositionField *pQryLimitPosition,
        int nRequestID) ;
```

参数：

pQryLimitPosition：指向查询投资者持仓限制的地址。

投资者买入额度查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
ExchangeID	交易所代码	可选
InstrumentID	合约代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询投资者持仓限制信息
CZQThostFtdcQryLimitPositionField qryField;
int iReq = g_pTradeStockApi-> ReqQryLimitPosition (&qryField, 0);
//查询上海证券交易所的投资者持仓限制信息
CZQThostFtdcQryLimitPositionField qryField;
strcpy(qryField.ExchangeID,"SSE");
int iReq = g_pTradeStockApi-> ReqQryLimitPosition (&qryField, 0);
```

6.4.52. ReqLockInsert 方法

请求锁定/解锁录入。

函数原形：

```
int ReqLockInsert(
                    CZQThostFtdcInputLockField *pInputLock,
                    int nRequestID) ;
```

参数：

pInputLock：指向锁定/解锁结构地址。

锁定输入结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
InstrumentID	合约代码	必填
LockRef	锁定引用	可选
UserID	用户代码	不填（证券暂不使用）
Volume	数量	必填
RequestID	请求编号	可选

BusinessUnit	业务单元	证券暂不使用
LockType	锁定类型	必填。 1=锁定 2=解锁
ExchangeID	交易所代码	必填
ClientID	客户代码	必填。对应“交易账户代码”（期货无）
InvestUnitID	投资单元代码	可选
IPAddress	IP 地址	可选
MacAddress	Mac 地址	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcInputLockField fldLock;
///经纪公司代码
strcpy(fldLock.BrokerID,"8000");
///投资者代码
strcpy(fldLock.InvestorID,"000001");
///用户代码
strcpy(fldLock.UserID,"0000001");
strcpy(fldLock.InstrumentID,"510050");
///交易所代码
strcpy(fldLock.ExchangeID,"SSE")
///数量
fldLock.Volume =10;
strcpy(fldLock.InvestUnitID,"2001001");
strcpy(fldLock.ClientID,"123456");
fldLock.LockType = ZQTHOST_FTDC_LF_Lock;
```

```
int iReq = g_pTradeStockApi-> ReqLockInsert (&fldLock, 0);
```

6.4.53. ReqQryExerciseAssignment 方法

请求查询行权指派信息。

函数原形：

int ReqQryExerciseAssignment(

CZQThostFtdcQryExerciseAssignmentField *pQryExerciseAssignment,

int nRequestID) ;

参数：

pQryExerciseAssignment：指向行权指派结构地址。

行权指派查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcQryExerciseAssignmentField qryField;

int iReq = g_pTradeStockApi-> ReqQryExerciseAssignment (&qryField, 0);
```

6.4.54. ReqExecOrderInsert 方法

请求行权录入。

函数原形：

int ReqExecOrderInsert(

CZQThostFtdcInputExecOrderField *pInputExecOrder,

int nRequestID) ;

参数：

pInputExecOrder：指向行权结构地址。

行权输入结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
InstrumentID	合约代码	必填
ExecOrderRef	执行宣告引用	可选
UserID	用户代码	不填（证券暂不使用）
Volume	数量	必填
RequestID	请求编号	可选
BusinessUnit	业务单元	证券暂不使用
OffsetFlag	开平标志	证券暂不使用
HedgeFlag	投机套保标志	证券暂不使用。 1=投机 2=套利 3=套保 4=备兑
ActionType	执行类型	必填。固定为 1 1=执行 2=放弃
PosiDirection	保留头寸申请的持仓方向	证券暂不使用。 1=净 2=多头 3=空头
ReservePositionFlag	期权行权后是否保留期货头寸的标记	证券暂不使用。 0=保留 1=不保留
CloseFlag	期权行权后生成的头寸是否自动平仓	证券暂不使用。 0=自动平仓 1=免于自动平仓
ExchangeID	交易所代码	必填
ClientID	客户代码	必填。对应“交易账户代码”（期货无）
InvestUnitID	投资单元代码	可选
AccountID	资金帐号代码	可选

IPAddress	IP 地址	可选
MacAddress	Mac 地址	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcInputExecOrderField fldExecOrder;
///经纪公司代码
strcpy(fldExecOrder.BrokerID," 8000");
///投资者代码
strcpy(fldExecOrder.InvestorID, "000001");
///用户代码
strcpy(fldExecOrder.UserID," 000001");
///合约交易代码(8 位,90000001)
strcpy(fldExecOrder.InstrumentID," 90000001");
///交易所代码
strcpy(fldExecOrder.ExchangeID,"SSE")
///数量
fldExecOrder.Volume =10;
///投机套保标志
fldExecOrder.HedgeFlag =ZQTHOST_FTDC_HF_Speculation;
///执行类型
fldExecOrder.ActionType = ZQTHOST_FTDC_EOAT_Execute;
///保留头寸申请的持仓方向
fldExecOrder.PosiDirection = ZQTHOST_FTDC_PD_Long;
///期权行权后是否保留期货头寸的标记
fldExecOrder.ReservePositionFlag =ZQTHOST_FTDC_RPF_Reserved ;
```

```
///期权行权后生成的头寸是否自动平仓
fldExecOrder.CloseFlag = ZQTHOST_FTDC_CF_CloseAuto ;
strcpy(fldExecOrder.InvestUnitID, "2001001");
strcpy(fldExecOrder.ClientID, "123456");
strcpy(fldExecOrder.AccountID, "123456-1");
int iReq = g_pTradeStockApi-> ReqExecOrderInsert (&fldExecOrder, 0);
```

6.4.55. ReqExecOrderAction 方法

请求行权操作录入。

函数原形：

int ReqExecOrderAction (

CZQThostFtdcInputExecOrderActionField *pInputExecOrderAction,

int nRequestID) ;

参数：

pInputExecOrderAction：指向行权操作结构地址。

行权操作输入结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
ExecOrderActionRef	执行宣告操作引用	可选
ExecOrderRef	执行宣告引用	执行宣告引用、前置编号、会话编号，3 个字段需一起填写，否则就都不填
RequestID	请求编号	可选
FrontID	前置编号	
SessionID	会话编号	
ExchangeID	交易所代码	必填
ExecOrderSysID	执行宣告编号	
ActionFlag	操作标志	撤单时填'0' 0=删除 3=修改
UserID	用户代码	证券暂时不用
InstrumentID	合约代码	证券暂时不用

IPAddress	IP 地址	可选
MacAddress	Mac 地址	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcExecOrderField* pExecOrder;
///经纪公司代码
strcpy(fldExerOrderAction.BrokerID, pExecOrder ->BrokerID);
///投资者代码
strcpy(fldExerOrderAction.InvestorID, pExecOrder->InvestorID);
///用户代码
strcpy(fldExerOrderAction.UserID, pExecOrder ->UserID);
///合约交易代码(8 位,90000001)
strcpy(fldExerOrderAction.InstrumentID, pExecOrder ->InstrumentID);
///交易所代码
strcpy(fldExerOrderAction.ExchangeID, pExecOrder -> ExchangeID);
strcpy(fldExerOrderAction.ExecOrderRef pExecOrder -> ExecOrderRef);
strcpy(fldExerOrderAction.ExecOrderSysID pExecOrder -> ExecOrderSysID);
fldExerOrderAction.FrontID = pExecOrder -> FrontID;
fldExerOrderAction.SessionID = pExecOrder -> SessionID;
fldExerOrderAction.ActionFlag = ZQTHOST_FTDC_AF_Delete;
int iReq = g_pTradeStockApi-> ReqExecOrderAction (&fldExerOrderAction, 0);
```

6.4.56. ReqSecurityTransfer 方法

请求现券转移录入。

函数原形：

int ReqSecurityTransfer(

CZQThostFtdcInputSecurityTransferField *pInputSecurityTransfer,

int nRequestID);

参数：

pInputSecurityTransfer：指向现券转移录入结构地址。

现券转移输入结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
InvestUnitID	投资单元代码	可选
SecurityInvestorID	现券系统投资者代码	必填
SecurityInvestUnitID	现券系统投资单元代码	可选
ExchangeID	交易所代码	必填
ClientID	客户代码	必填
InstrumentID	合约代码	必填
ApplySerial	申请流水号	可选
SecurityTransferDirection	现券转移方向	必填。 0：转入现券系统；1：转出现券系统
Volume	数量	必填
TransferPositionType	转移持仓类型	必填。 0：任意仓 1：昨仓 2：今买卖仓 3：今申赎仓 转出时支持所有类型；转入不支持任意仓

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
CZQThostFtdcInputSecurityTransferField fldSecurityTransfer;
///经纪公司代码
strcpy(fldSecurityTransfer.BrokerID,"8000");
///投资者代码
strcpy(fldSecurityTransfer.InvestorID,"00001");
 ///投资单元代码
strcpy(fldSecurityTransfer.InvestUnitID,"123456");
    ///现券系统投资者代码
strcpy(fldSecurityTransfer.SecurityInvestorID,"00002");
    ///现券系统投资单元代码
strcpy(fldSecurityTransfer.SecurityInvestUnitID,"123456");
    ///交易所代码
strcpy(fldSecurityTransfer.ExchangeID,"SSE");
    ///客户代码
strcpy(fldSecurityTransfer.ClientID,"000001");
    ///合约代码
strcpy(fldSecurityTransfer.InstrumentID,"900001");
    ///现券转移方向
fldSecurityTransfer.SecurityTransferDirection = ZQTHOST_FTDC_STD_In;
    ///数量
fldSecurityTransfer.Volume = 100;
    ///转移持仓类型
fldSecurityTransfer.TransferPositionType = ZQTHOST_FTDC_TPT_ALL;
int ret = g_pTradeStockApi-> ReqSecurityTransfer (&fldSecurityTransfer,0);
```

6.4.57. ReqQrySecurityTransfer 方法

请求查询现券转移信息。

函数原形：

int ReqQrySecurityTransfer(

CZQThostFtdcQrySecurityTransferField *pQrySecurityTransfer,

int nRequestID) ;

参数：

pQrySecurityTransfer：指向现券录入结构地址。

现券转移查询结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
InvestUnitID	投资单元代码	可选
ExchangeID	交易所代码	可选
ClientID	客户代码	可选
InstrumentID	合约代码	可选
SecurityTransferSerial	现券转移流水号	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

示例：

```
//查询现券转移信息

CZQThostFtdcQrySecurityTransferField qryField;

int iReq = g_pTradeStockApi-> ReqQrySecurityTransfer (&qryField, 0);
```



```
//查询上海证券交易所的投资者现券转移信息
CZQThostFtdcQrySecurityTransferField qryField;
strcpy(qryField.ExchangeID,"SSE");
int iReq = g_pTradeStockApi-> ReqQrySecurityTransfer (&qryField, 0);
```

6.4.58. ReqQryLOFInfo 方法

请求查询上市开放式基金信息。

函数原形：

int ReqQryLOFInfo(

CZQThostFtdcQryLOFInfoField *pQryLOFInfo,

int nRequestID) ;

参数：

pQryLOFInfo：指向上市开放式基金结构地址。

上市开放式基金结构：

字段名称	属性	描述
ExchangeID	交易所代码	可选
FundID	基金代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.59. ReqQryLOFSubInfo 方法

请求查询上市开放式基金子基金信息。

函数原形：

int ReqQryLOFSubInfo(

CZQThostFtdcQryLOFSubInfoField *pQryLOFSubInfo,

int nRequestID) ;

参数：

pQryLOFSubInfo：指向上市开放式基金子基金结构地址。

上市开放式基金子基金信息结构：

字段名称	属性	描述
ExchangeID	交易所代码	可选
MasterFundID	母基金代码	可选
SubFundID	子基金代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.60. ReqCashRepayInsert 方法

请求直接还款录入。

函数原形：

int ReqCashRepayInsert(

CZQThostFtdcInputCashRepayField *pInputCashRepay,

int nRequestID) ;

参数：

pInputCashRepay：指向直接还款录入结构地址。

直接还款录入结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
CROrderRef	直接还款引用	可选
AccountID	资金账户代码	必填
Amount	申报还款金额	必填
InvestUnitID	投资单元代码	可选
RequestID	请求编号	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.61. ReqQryCashRepay 方法

请求查询融资融券直接还款信息。

函数原形：

int ReqQryCashRepay(

CZQThostFtdcQryCashRepayField *pQryCashRepay,

int nRequestID) ;

参数：

pQryCashRepay：指向融资融券直接还款信息结构地址。

融资融券直接还款信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填

InvestorID	投资者代码	可选
CROrderSysID	直接还款报单编号	可选
InsertTimeStart	开始时间	可选
InsertTimeEnd	结束时间	可选
InvestUnitID	投资单元代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.62. ReqQryCreditInfo 方法

请求查询投资者融资融券信息。

函数原形：

int ReqQryCreditInfo(

CZQThostFtdcQryCreditInfoField *pQryCreditInfo,

int nRequestID) ;

参数：

pQryCreditInfo：指向投资者融资融券信息结构地址。

投资者融资融券信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.63. ReqQryCreditSecurityPosition 方法

请求查询投资者融资融券可用证券头寸。

函数原形：

int ReqQryCreditSecurityPosition(

CZQThostFtdcQryCreditSecurityPositionField *pQryCreditSecurityPosition,

int nRequestID) ;

参数：

pQryCreditSecurityPosition：指向投资者融资融券可用证券头寸结构地址。

投资者融资融券可用证券头寸结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
ExchangeID	交易所代码	可选
InstrumentID	合约代码	可选
CreditPositionType	头寸类型	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.64. ReqQryCreditFundPosition 方法

请求查询投资者融资融券可用资金头寸。

函数原形：

int ReqQryCreditFundPosition(

CZQThostFtdcQryCreditFundPositionField *pQryCreditFundPosition,

int nRequestID) ;

参数：

pQryCreditFundPosition：指向投资者融资融券可用资金头寸结构地址。

投资者融资融券可用资金头寸结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选
CreditPositionType	头寸类型	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.65. ReqQryCreditContract 方法

请求查询投资者信用合约。

函数原形：

int ReqQryCreditContract(

CZQThostFtdcQryCreditContractField *pQryCreditContract,

int nRequestID) ;

参数：

pQryCreditContract：指向投资者信用合约结构地址。

投资者信用合约结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.66. ReqQryCreditRepayment 方法

请求查询投资者融资融券偿还明细。

函数原形：

int ReqQryCreditRepayment(

CZQThostFtdcQryCreditRepaymentField *pQryCreditRepayment,

int nRequestID) ;

参数：

pQryCreditRepayment：指向投资者融资融券偿还明细结构地址。

投资者融资融券偿还明细结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	可选

CreditContractID	信用合约代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.67. ReqQryCreditUnderlying 方法

请求查询投资者融资融券标的证券信息。

函数原形：

int ReqQryCreditUnderlying(

CZQThostFtdcQryCreditUnderlyingField *pQryCreditUnderlying,

int nRequestID) ;

参数：

pQryCreditUnderlying：指向投资者融资融券标的证券信息结构地址。

投资者融资融券标的证券信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
ExchangeID	交易所代码	可选
InstrumentID	合约代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

6.4.68. ReqQryCreditCollateral 方法

请求查询投资者融资融券担保证券信息。

函数原形：

int ReqQryCreditCollateral(

CZQThostFtdcQryCreditCollateralField *pQryCreditCollateral,

int nRequestID) ;

参数：

pQryCreditCollateral：指向投资者融资融券担保证券信息结构地址。

投资者融资融券担保证券信息结构：

字段名称	属性	描述
BrokerID	经纪公司代码	必填
InvestorID	投资者代码	必填
ExchangeID	交易所代码	可选
InstrumentID	合约代码	可选

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

7. 开发示例

7.1. 交易 API 开发示例

```
// tradeapitest.cpp :
// 一个简单的例子，介绍CThostFtdcTraderApi和CThostFtdcTraderSpi接口的使用。
// 本例将演示一个报单录入操作的过程
#include <stdio.h>
#include <windows.h>
#include "ThostFtdcUserApiSSE.h"
// 报单录入操作是否完成的标志
// Create a manual reset event with no signal
HANDLE g_hEvent = CreateEvent(NULL, true, false, NULL);
// 会员代码
TZQThostFtdcBrokerIDType g_chBrokerID;
// 交易用户代码
TZQThostFtdcUserIDType g_chUserID;
Void InsertOrder()
{
}
class CSimpleHandler : public CZQThostFtdcTraderSpi
{
public:
    // 构造函数，需要一个有效的指向CThostFtdcMduserApi实例的指针
```

```
    CSimpleHandler(CZQThostFtdcTraderApi *pTradeApi) : m_pTradeApi(pTradeApi) {};
    ~CSimpleHandler() {};
// 当客户端与交易托管系统建立起通信连接，客户端需要进行登录
    virtual void OnFrontConnected()
    {
            CZQThostFtdcReqUserLoginField reqUserLogin;
            // get BrokerID
            printf("BrokerID:");
            scanf("%s", &g_chBrokerID);
            strcpy(reqUserLogin. BrokerID, g_chBrokerID);
            // get userid
            printf("userid:");
            scanf("%s", &g_chUserID);
            strcpy(reqUserLogin.UserID, g_chUserID);
            // get password
            printf("password:");
            scanf("%s", &reqUserLogin.Password);
            // 发出登陆请求
            m_pTradeApi->ReqUserLogin(&reqUserLogin, 0);
        }
        // 当客户端与交易托管系统通信连接断开时，该方法被调用
        virtual void OnFrontDisconnected(int nReason)
        {
```

```
            // 当发生这个情况后，API会自动重新连接，客户端可不做处理
            printf("OnFrontDisconnected.\n");
        }
        // 当客户端发出登录请求之后，该方法会被调用，通知客户端登录是否成功
        virtual void OnRspUserLogin(CZQThostFtdcRspUserLoginField
*pRspUserLogin, CZQThostFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
        {
            printf("OnRspUserLogin:\n");
           printf("ErrorCode=[%d], ErrorMsg=[%s]\n", pRspInfo->ErrorID,
           pRspInfo->ErrorMsg);
            printf("RequestID=[%d], Chain=[%d]\n", nRequestID, bIsLast);
            if (pRspInfo->ErrorID != 0)
          {
                // 端登失败，客户端需进行错误处理
                   printf("Failed to login, errorcode=%d errormsg=%s requestid=%d
                   chain=%d", pRspInfo->ErrorID, pRspInfo->ErrorMsg, nRequestID,
                   bIsLast);
               exit(-1);
            }
    //经纪公司代码
    strcpy_s(order.BrokerID, g_chBrokerID);
    //投资者代码
    strcpy_s(order.InvestorID,“0000001”);
    // 合约代码
    strcpy_s(order.InstrumentID, "600000");

    // 用户代码
    strcpy_s(order.UserID, g_chUserID);
    // 报单价格条件
```

```
    order.OrderPriceType = ZQTHOST_FTDC_OPT_LimitPrice;
    // 买卖方向
    order.Direction = GetDirection(Dirction);

    // 价格
    order.LimitPrice = Price;
    // 数量
    order.VolumeTotalOriginal = Volume;
    // 有效期类型
    order.TimeCondition = ZQTHOST_FTDC_TC_GFD;
    // GTD日期
    strcpy_s(order.GTDDate, "");
    // 成交量类型
    order.VolumeCondition = ZQTHOST_FTDC_VC_AV;
    // 最小成交量
    //order.MinVolume = 0;
    // 触发条件
    order.ContingentCondition = ZQTHOST_FTDC_CC_Immediately;
   strcpy_s(order.InvestUnitID,“12345678”);
    strcpy_s(order.AccountID,"0000001");
    strcpy_s(order.ClientID,“0000001_1”);

    strcpy_s(order.ExchangeID,"SSE");
    order.UserForceClose = 0;
    order.IsAutoSuspend = 0;


    int nRet = m_pTradeApi ->ReqOrderInsert(&ordField,1);

    }

// 报单录入应答

virtual void OnRspOrderInsert(CZQThostFtdcInputOrderField *pInputOrder,

CZQThostFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)

{

// 输出报单录入结果

    printf("ErrorCode=[%d], ErrorMsg=[%s]\n", pRspInfo->ErrorID,

            pRspInfo->ErrorMsg);
```

```
            // 通知报单录入完成
    SetEvent(g_hEvent);
 };
///报单回报
virtual void OnRtnOrder(CThostFtdcOrderField *pOrder)
{
   printf("OnRtnOrder:\n");
   printf("OrderSysID=[%s]\n", pOrder->OrderSysID);
}
// 针对用户请求的出错通知
virtual void OnRspError(CThostFtdcRspInfoField *pRspInfo, int nRequestID, bool
bIsLast)
{
    printf("OnRspError:\n");
    printf("ErrorCode=[%d], ErrorMsg=[%s]\n", pRspInfo->ErrorID,
pRspInfo->ErrorMsg);
      printf("RequestID=[%d], Chain=[%d]\n", nRequestID, bIsLast);
      // 客户端需进行错误处理
      //……
    }
    private:
    CZQThostFtdcTraderApi *m_pTradeApi;
    };
    int main()
```

```
{
    // 产生一个CZQThostFtdcTraderApi实例
CZQThostFtdcTraderApi *pTradeApi =
CZQThostFtdcTraderApi::CreateFtdcTraderApi();
    // 产生一个事件处理的实例
    CSimpleHandler sh(pTradeApi);
    // 注册一事件处理的实例
    pTradeApi->RegisterSpi(&sh);
    // 订阅私有流
    //         TERT_RESTART:从本交易日开始重传
    //         TERT_RESUME:从上次收到的续传
    //         TERT_QUICK:只传送登录后私有流的内容
    pTradeApi->SubscribePrivateTopic(TERT_RESUME);
    // 订阅公共流
    //         TERT_RESTART:从本交易日开始重传
    //         TERT_RESUME:从上次收到的续传
    //         TERT_QUICK:只传送登录后公共流的内容
    pTradeApi->SubscribePublicTopic(TERT_RESUME);

    // 设置交易托管系统服务的地址，可以注册多个地址备用
    pTradeApi->RegisterFront("tcp://172.16.0.31:57205");
    // 使客户端开始与后台服务建立连接
    pTradeApi->Init();
```

```
    // 客户端等待报单操作完成
    WaitForSingleObject(g_hEvent, INFINITE);
    // 释放API实例
    pTradeApi->Release();
    return 0;
}
```

7.2.行情 API 开发示例

```
// tradeapitest.cpp :
// 一个简单的例子，介绍CZQThostFtdcMdApi和CZQThostFtdcMdSpi接口的使用。
// 本例将演示一个报单录入操作的过程
#include <stdio.h>
#include <windows.h>
#include "ThostFtdcMdApiSSE.h"
// 报单录入操作是否完成的标志
// Create a manual reset event with no signal
HANDLE g_hEvent = CreateEvent(NULL, true, false, NULL);
// 会员代码
TZQThostFtdcBrokerIDType g_chBrokerID;
// 交易用户代码
TZQThostFtdcUserIDType g_chUserID;
class CSimpleHandler : public CZQThostFtdcMdSpi
{
public:
    // 构造函数，需要一个有效的指向CThostFtdcMdApi实例的指针
    CSimpleHandler(CZQThostFtdcMdApi *pMdApi)
                : m_ pMdApi(pMdApi)
    {};
    ~CSimpleHandler()
    {};

    // 当客户端与交易托管系统建立起通信连接，客户端需要进行登录
    virtual void OnFrontConnected()
    {
```

```
        CZQThostFtdcReqUserLoginField reqUserLogin;
        // get BrokerID
        printf("BrokerID:");
        scanf("%s", &g_chBrokerID);
        strcpy(reqUserLogin. BrokerID, g_chBrokerID);
        // get userid
        printf("userid:");
        scanf("%s", &g_chUserID);
        strcpy(reqUserLogin.UserID, g_chUserID);
        // get password
        printf("password:");
        scanf("%s", &reqUserLogin.Password);
        // 发出登陆请求
        m_ pMdApi->ReqUserLogin(&reqUserLogin, 0);
    }

    // 当客户端与交易托管系统通信连接断开时，该方法被调用
    virtual void OnFrontDisconnected(int nReason)
    {
        // 当发生这个情况后，API会自动重新连接，客户端可不做处理
        printf("OnFrontDisconnected.\n");
    }
    // 当客户端发出登录请求之后，该方法会被调用，通知客户端登录是否成功
    virtual void OnRspUserLogin(CZQThostFtdcRspUserLoginField *pRspUserLogin,
    CZQThostFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
    {
        printf("OnRspUserLogin:\n");
        printf("ErrorCode=[%d], ErrorMsg=[%s]\n", pRspInfo->ErrorID,
        pRspInfo->ErrorMsg);
        printf("RequestID=[%d], Chain=[%d]\n", nRequestID, bIsLast);
        if (pRspInfo->ErrorID != 0)
        {
            // 端登失败，客户端需进行错误处理
                printf("Failed to login, errorcode=%d errormsg=%s requestid=%d
                chain=%d", pRspInfo->ErrorID, pRspInfo->ErrorMsg, nRequestID,
                bIsLast);
            exit(-1);
        }
        // 端登成功
        // 订阅行情
       char  * Instrumnet[]={“000001”,”000002”};
        m_ pMdApi->SubscribeMarketData (Instrumnet,2,”SZE”);
    }
    // 行情应答
```

```
    virtual     void     OnRtnDepthMarketData(CThostFtdcDepthMarketDataField
            *pDepthMarketData)
    {
                printf("ErrorCode=[%d], ErrorMsg=[%s]\n", pRspInfo->ErrorID,
                pRspInfo->ErrorMsg);
                //收到深度行情
                SetEvent(g_hEvent);
    };

    // 针对用户请求的出错通知
    virtual void OnRspError(CThostFtdcRspInfoField *pRspInfo, int nRequestID,
    bool bIsLast)
    {
        printf("OnRspError:\n");
        printf("ErrorCode=[%d],     ErrorMsg=[%s]\n",     pRspInfo->ErrorID,
             pRspInfo->ErrorMsg);
        printf("RequestID=[%d], Chain=[%d]\n", nRequestID, bIsLast);
        // 客户端需进行错误处理
    }
private:
    // 指向CThostFtdcMdApi实例的指针
    CZQThostFtdcMdApi *m_pMdApi;
};

int main()
{
    // 产生一个CThostFtdcMdApi实例
    CZQThostFtdcMdApi *pMdApi = CZQThostFtdcMdApi::CreateFtdcMdApi();
    // 产生一个事件处理的实例
    CSimpleHandler sh(pMdApi);
    // 注册一事件处理的实例
    pMdApi->RegisterSpi(&sh);

    // 设置交易托管系统服务的地址，可以注册多个地址备用
    pMdApi->RegisterFront("tcp://172.16.0.31:57205");
    // 使客户端开始与后台服务建立连接
    pMdApi->Init();

    WaitForSingleObject(g_hEvent, INFINITE);
    // 释放API实例
    pMdApi->Release();
    return 0;
```

```
CString InvestorID,InstrumentID,Dirction,InvestUnit,TradingCode,AccountId,strExchangeId;
        double Price;
        int Volume;

        InvestorID     =pOrderNode->GetAttribute(_T("InvestorID"));
        InstrumentID =pOrderNode->GetAttribute(_T("InstrumentID"));
        Dirction =pOrderNode->GetAttribute(_T("Direction"));
        Price       =pOrderNode->GetAttribute(_T("LimitPrice"),0.0);
        Volume =pOrderNode->GetAttribute(_T("Volume"),0);

        InvestUnit = pOrderNode->GetAttribute(_T("InvestUnit"));
        TradingCode = pOrderNode->GetAttribute(_T("TradingCode"));
        AccountId = pOrderNode->GetAttribute(_T("AccountId"));
       strExchangeId = pOrderNode->GetAttribute(_T("ExchangeId"));
        //bool IsCounterparty = nodes[i]->GetAttribute(_T("IsCounterparty"),false);

        //经纪公司代码
        strcpy_s(order.BrokerID, pRspLogin->BrokerID);
        //投资者代码
        strcpy_s(order.InvestorID, InvestorID);
        // 合约代码
        strcpy_s(order.InstrumentID, InstrumentID);
        ///报单引用,由交易系统返回，不设置
        //strcpy(ord.OrderRef, "000000000001");
        // 用户代码
        strcpy_s(order.UserID, pRspLogin->UserID);
        // 报单价格条件
        order.OrderPriceType = ZQTHOST_FTDC_OPT_LimitPrice;
        // 买卖方向
        order.Direction = GetDirection(Dirction);

        // 价格
        order.LimitPrice = Price;
        // 数量
        order.VolumeTotalOriginal = Volume;
        // 有效期类型
        order.TimeCondition = ZQTHOST_FTDC_TC_GFD;
        // GTD日期
        strcpy_s(order.GTDDate, "");
        // 成交量类型
        order.VolumeCondition = ZQTHOST_FTDC_VC_AV;
        // 最小成交量
        //order.MinVolume = 0;
```



```
// 触发条件
order.ContingentCondition = ZQTHOST_FTDC_CC_Immediately;
strcpy_s(order.InvestUnitID,InvestUnit);
strcpy_s(order.AccountID,AccountId);
strcpy_s(order.ClientID,TradingCode);

strcpy_s(order.ExchangeID,strExchangeId);
order.UserForceClose = 0;
order.IsAutoSuspend = 0;
```

