证券交易托管系统

交易员应用程序接口

2013 年 8 月

1. 文件属性

文件属性	内容
文件名称	交易托管系统_ TradeAPI 接口
文件编号	
文件版本号	V0.1
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	2013-8-12
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	2013-08-12	桂荣盛	创建
			1、

3. 本次修改变更说明

序号	变更内容简述
1.	创建（文中的接口结构以实际的 API 接口结构为准）
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	

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
ThostFtdcTraderApiSSE.h	V1.0	11kb	交易接口头文件
ThostFtdcUserApiStructSSE.h	V1.0	144kb	定义了 API 所需的一系列数据类型的头文件
ThostFtdcUserApiDataTypeSSE.h	V1.0	179kb	定义了一系列业务相关的数据结构的头文件
thosttraderapiSSE.dll	V1.0	1169kb	动态链接库二进制文件
thosttraderapiSSE.lib	V1.0	4kb	导入库文件
thostmduserapiSSE.dll	V1.0	625kb	动态链接库二进制文件
thostmduserapiSSE.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：

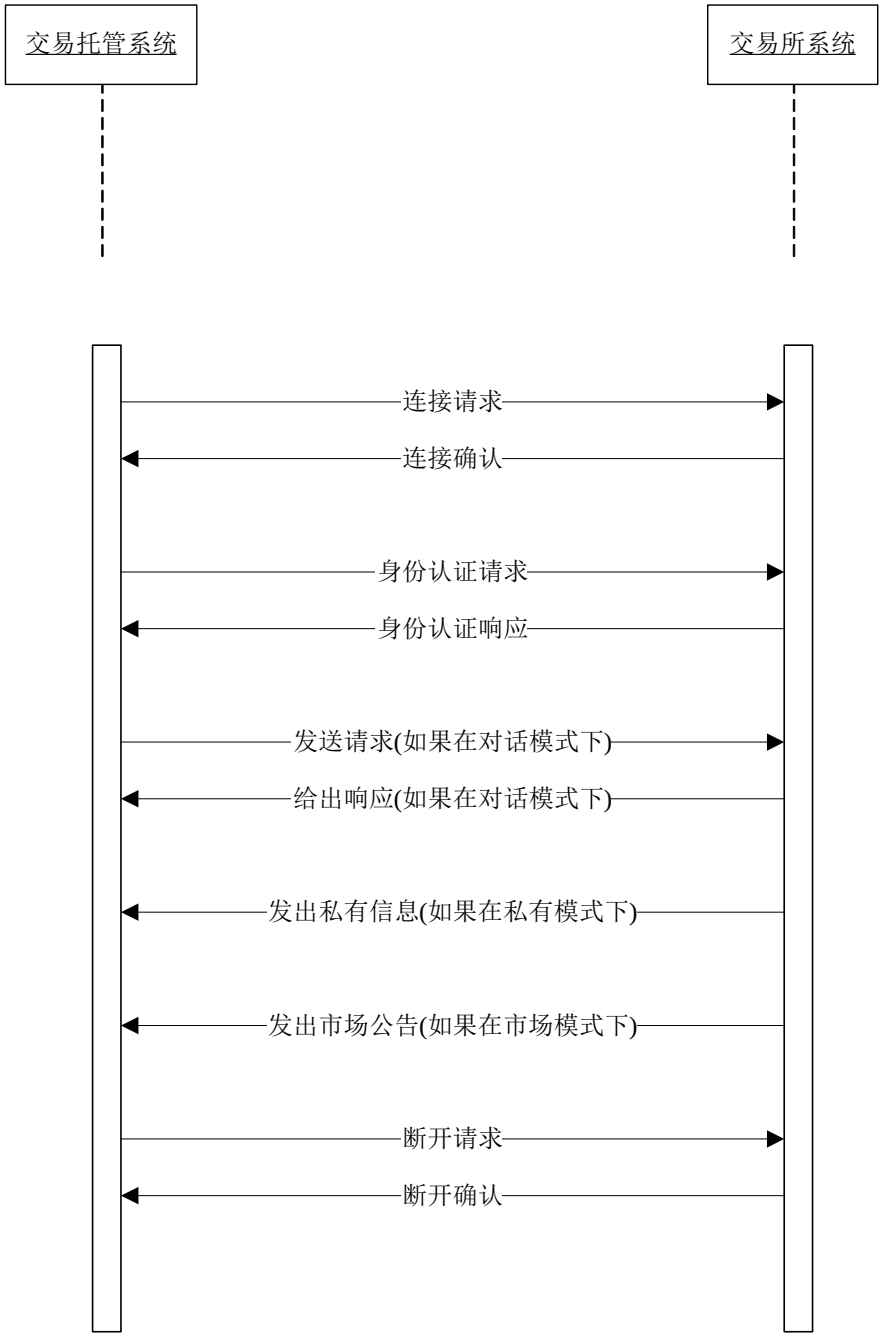


图1) 各通讯模式的工作过程

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

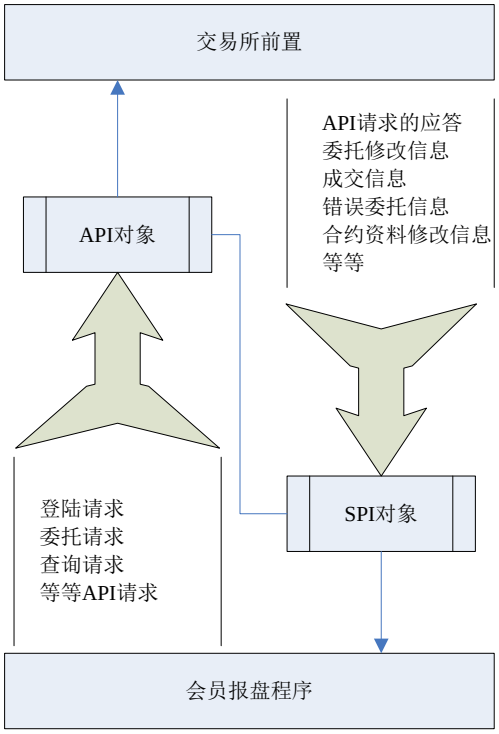
4.1. 工作线程

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
登录	登录	CZQThostFtdcTraderApi:: ReqUserLogin	CZQThostFtdcTraderSpi::OnRspUserLogin	对话流
	登出	CZQThostFtdcTraderApi::ReqUserLogout	CZQThostFtdcTraderSpi::OnRspUserLogout	对话流
	修改用户口令	CZQThostFtdcTraderApi::ReqUserPasswordUpdate	CZQThostFtdcTraderSpi::OnRspUserPasswordUpdate	对话流
交易	报单录入	CZQThostFtdcTraderApi::ReqOrderInsert	CZQThostFtdcTraderSpi::OnRspOrderInsert	对话流
	报单操作	CZQThostFtdcTraderApi::ReqOrderAction	CZQThostFtdcTraderSpi::OnRspOrderAction	对话流
	出金	CZQThostFtdcTraderApi::ReqFundOutCTPAccount	CZQThostFtdcTraderSpi::OnRspFundOutCTPAccount	对话流
私有回报	成交回报	N/A	CZQThostFtdcTraderSpi::OnRtnTrade	私有流
	报单回报	N/A	CZQThostFtdcTraderSpi::OnRtnOrder	私有流
	报单录入错误回报	N/A	CZQThostFtdcTraderSpi::OnErrRtnOrderInsert	私有流
	报单操作错误回报	N/A	CThostFtdcTraderSpi::OnErrRtnOrderAction	私有流
	资金转入 CTP 通知	N/A	CThostFtdcTraderSpi::OnRtnFundIntoCTPAccount	私有流
	资金转出 CTP 通知	N/A	CThostFtdcTraderSpi::OnRtnFundOutCTPAccount	私有流
查询	报单查询	CZQThostFtdcTraderApi::ReqQryOrder	CZQThostFtdcTraderSpi::OnRspQryOrder	查询流
	成交查询	CZQThostFtdcTraderApi::ReqQryTrade	CZQThostFtdcTraderSpi::OnRspQryTrade	查询流
	投资者查询	CZQThostFtdcTraderApi::ReqQry Investor	CZQThostFtdcTraderSpi::OnRspQry Investor	查询流
	投资者持仓查询	CZQThostFtdcTraderApi::ReqQry Investor Position	CZQThostFtdcTraderSpi::OnRspQry Investor Position	查询流
	合约查询	CZQThostFtdcTraderApi::ReqQryInstrument	CZQThostFtdcTraderSpi::OnRspQryInstrument	查询流
	债券利息查询	CZQThostFtdcTraderApi ::ReqQryBondInterest	CZQThostFtdcTraderSpi::OnRspQryBondInterest	查询流
	资金转入转出 CTP 查询	CZQThostFtdcTraderApi ::ReqQryFundIOCTPAccount	CZQThostFtdcTraderSpi::OnRspQryFundIOCTPAccount	查询流

交易接口和私有流接口会有相互关联，如用户报单录入 ReqOrderInsert，马上会收到报单响应 OnRspOrderInsert，说明交易系统已经收到报单。报单进入交易系统后，如果报单的交易状态发生变化，就会收到报单回报 OnRtnOrder。如果报单被撮合(部分)成交，就会收到成交回报 OnRtnTrade。其中，一个用户的报单回报和成交回报也会被所属会员下其他的用户接受到。

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 请求的输入参数不能为 NULL。
2，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。

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. 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：

```
struct CZQThostFtdcRspUserLoginField
{
    ///交易日
    TZQThostFtdcDateType TradingDay;
    ///登录成功时间
    TZQThostFtdcTimeType    LoginTime;
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///交易系统名称
    TZQThostFtdcSystemNameType   SystemName;
};
```

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。

6.3.5. 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：

```
struct CZQThostFtdcUserLogoutField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
};
```

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。

6.3.6. 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：

```
struct CZQThostFtdcUserPasswordUpdateField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///原来的口令
    TZQThostFtdcPasswordType OldPassword;
    ///新的口令
    TZQThostFtdcPasswordType NewPassword;
};
```

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。

6.3.7. OnRspTradingAccountPasswordUpdate 方法

资金账户口令更新应答。当客户端发出资金账户口令更新指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspTradingAccountPasswordUpdate(
        CZQThostFtdcTradingAccountPasswordUpdateField
        *pTradingAccountPasswordUpdate,
        CZQThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
```


bool bIsLast)；

参数：

pTradingAccountPasswordUpdate：指向资金账户口令变更域结构的地址，包含了用户密码修改请求的输入数据。

资金账户口令变更域结构：

```
struct CZQThostFtdcTradingAccountPasswordUpdateField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者帐号
    TThostFtdcAccountIDType    AccountID;
    ///原来的口令
    TZQThostFtdcPasswordType OldPassword;
    ///新的口令
    TZQThostFtdcPasswordType NewPassword;
};
```

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. 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。

6.3.9. 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：

```
struct CZQThostFtdcInputOrderField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///用户代码
```

```
    TZQThostFtdcUserIDType    UserID;
    ///报单价格条件
    TZQThostFtdcOrderPriceTypeType      OrderPriceType;
    ///买卖方向
    TZQThostFtdcDirectionType Direction;
    ///组合开平标志
    TZQThostFtdcCombOffsetFlagType     CombOffsetFlag;
    ///组合投机套保标志
    TZQThostFtdcCombHedgeFlagType     CombHedgeFlag;
    ///价格
    TZQThostFtdcPriceType       LimitPrice;
    ///数量
    TZQThostFtdcVolumeType    VolumeTotalOriginal;
    ///有效期类型
    TZQThostFtdcTimeConditionType       TimeCondition;
    ///GTD 日期
    TZQThostFtdcDateType GTDDate;
    ///成交量类型
    TZQThostFtdcVolumeConditionType    VolumeCondition;
    ///最小成交量
    TZQThostFtdcVolumeType    MinVolume;
    ///触发条件
    TZQThostFtdcContingentConditionType      ContingentCondition;
    ///止损价
    TZQThostFtdcPriceType       StopPrice;
    ///强平原因
    TZQThostFtdcForceCloseReasonType  ForceCloseReason;
    ///自动挂起标志
    TZQThostFtdcBoolType IsAutoSuspend;
    ///业务单元
    TZQThostFtdcBusinessUnitType   BusinessUnit;
    ///请求编号
    TZQThostFtdcRequestIDType      RequestID;
};
```

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。

6.3.10. 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(
                        CZQThostFtdcOrderActionField *pOrderAction,
                        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：

```
struct CZQThostFtdcOrderActionField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///报单操作引用
    TZQThostFtdcOrderActionRefType      OrderActionRef;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///请求编号
    TZQThostFtdcRequestIDType       RequestID;
    ///前置编号
    TZQThostFtdcFrontIDType    FrontID;
    ///会话编号
    TZQThostFtdcSessionIDType       SessionID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///报单编号
    TZQThostFtdcOrderSysIDType    OrderSysID;
```

```
    ///操作标志
    TZQThostFtdcActionFlagType      ActionFlag;
    ///价格
    TZQThostFtdcPriceType       LimitPrice;
    ///数量变化
    TZQThostFtdcVolumeType    VolumeChange;
    ///操作日期
    TZQThostFtdcDateType ActionDate;
    ///操作时间
    TZQThostFtdcTimeType       ActionTime;
    ///交易所交易员代码
    TZQThostFtdcTraderIDType  TraderID;
    ///安装编号
    TZQThostFtdcInstallIDType  InstallID;
    ///本地报单编号
    TZQThostFtdcOrderLocalIDType  OrderLocalID;
    ///操作本地编号
    TZQThostFtdcOrderLocalIDType  ActionLocalID;
    ///会员代码
    TZQThostFtdcParticipantIDType   ParticipantID;
    ///客户代码
    TZQThostFtdcClientIDType  ClientID;
    ///业务单元
    TZQThostFtdcBusinessUnitType   BusinessUnit;
    ///报单操作状态
    TZQThostFtdcOrderActionStatusType  OrderActionStatus;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///状态信息
    TZQThostFtdcErrorMsgType StatusMsg;
};
```

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。

6.3.11. 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：

```
struct CZQThostFtdcOrderField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///报单价格条件
    TZQThostFtdcOrderPriceTypeType      OrderPriceType;
    ///买卖方向
    TZQThostFtdcDirectionType Direction;
    ///组合开平标志
    TZQThostFtdcCombOffsetFlagType     CombOffsetFlag;
    ///组合投机套保标志
    TZQThostFtdcCombHedgeFlagType     CombHedgeFlag;
    ///价格
    TZQThostFtdcPriceType       LimitPrice;
    ///数量
    TZQThostFtdcVolumeType    VolumeTotalOriginal;
    ///有效期类型
    TZQThostFtdcTimeConditionType       TimeCondition;
```

```
///GTD 日期
TZQThostFtdcDateType GTDDate;
///成交量类型
TZQThostFtdcVolumeConditionType    VolumeCondition;
///最小成交量
TZQThostFtdcVolumeType    MinVolume;
///触发条件
TZQThostFtdcContingentConditionType      ContingentCondition;
///止损价
TZQThostFtdcPriceType       StopPrice;
///强平原因
TZQThostFtdcForceCloseReasonType  ForceCloseReason;
///自动挂起标志
TZQThostFtdcBoolType IsAutoSuspend;
///业务单元
TZQThostFtdcBusinessUnitType   BusinessUnit;
///请求编号
TZQThostFtdcRequestIDType      RequestID;
///本地报单编号
TZQThostFtdcOrderLocalIDType  OrderLocalID;
///交易所代码
TZQThostFtdcExchangeIDType    ExchangeID;
///会员代码
TZQThostFtdcParticipantIDType   ParticipantID;
///客户代码
TZQThostFtdcClientIDType  ClientID;
///合约在交易所的代码
TZQThostFtdcExchangeInstIDType      ExchangeInstID;
///交易所交易员代码
TZQThostFtdcTraderIDType  TraderID;
///安装编号
TZQThostFtdcInstallIDType  InstallID;
///报单提交状态
TZQThostFtdcOrderSubmitStatusType  OrderSubmitStatus;
///报单提示序号
TZQThostFtdcSequenceNoType    NotifySequence;
///交易日
TZQThostFtdcDateType TradingDay;
///结算编号
TZQThostFtdcSettlementIDType   SettlementID;
///报单编号
TZQThostFtdcOrderSysIDType    OrderSysID;
///报单来源
TZQThostFtdcOrderSourceType    OrderSource;
```



```
    ///报单状态
    TZQThostFtdcOrderStatusType     OrderStatus;
    ///报单类型
    TZQThostFtdcOrderTypeType       OrderType;
    ///今成交数量
    TZQThostFtdcVolumeType    VolumeTraded;
    ///剩余数量
    TZQThostFtdcVolumeType    VolumeTotal;
    ///报单日期
    TZQThostFtdcDateType InsertDate;
    ///插入时间
    TZQThostFtdcTimeType       InsertTime;
    ///激活时间
    TZQThostFtdcTimeType       ActiveTime;
    ///挂起时间
    TZQThostFtdcTimeType       SuspendTime;
    ///最后修改时间
    TZQThostFtdcTimeType       UpdateTime;
    ///撤销时间
    TZQThostFtdcTimeType       CancelTime;
    ///最后修改交易所交易员代码
    TZQThostFtdcTraderIDType  ActiveTraderID;
    ///结算会员编号
    TZQThostFtdcParticipantIDType   ClearingPartID;
    ///序号
    TZQThostFtdcSequenceNoType    SequenceNo;
    ///前置编号
    TZQThostFtdcFrontIDType    FrontID;
    ///会话编号
    TZQThostFtdcSessionIDType       SessionID;
    ///用户端产品信息
    TZQThostFtdcProductInfoType     UserProductInfo;
    ///状态信息
    TZQThostFtdcErrorMsgType StatusMsg;
};
```

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。

6.3.12. 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：

```
struct CZQThostFtdcTradeField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///成交编号
    TZQThostFtdcTradeIDType   TradeID;
    ///买卖方向
    TZQThostFtdcDirectionType Direction;
    ///报单编号
    TZQThostFtdcOrderSysIDType    OrderSysID;
    ///会员代码
    TZQThostFtdcParticipantIDType   ParticipantID;
```

```
    ///客户代码
    TZQThostFtdcClientIDType  ClientID;
    ///交易角色
    TZQThostFtdcTradingRoleType    TradingRole;
    ///合约在交易所的代码
    TZQThostFtdcExchangeInstIDType      ExchangeInstID;
    ///开平标志
    TZQThostFtdcOffsetFlagType      OffsetFlag;
    ///投机套保标志
    TZQThostFtdcHedgeFlagType      HedgeFlag;
    ///价格
    TZQThostFtdcPriceType       Price;
    ///数量
    TZQThostFtdcVolumeType    Volume;
    ///成交时期
    TZQThostFtdcDateType TradeDate;
    ///成交时间
    TZQThostFtdcTimeType       TradeTime;
    ///成交类型
    TZQThostFtdcTradeTypeType      TradeType;
    ///成交价来源
    TZQThostFtdcPriceSourceType     PriceSource;
    ///交易所交易员代码
    TZQThostFtdcTraderIDType  TraderID;
    ///本地报单编号
    TZQThostFtdcOrderLocalIDType  OrderLocalID;
    ///结算会员编号
    TZQThostFtdcParticipantIDType   ClearingPartID;
    ///业务单元
    TZQThostFtdcBusinessUnitType   BusinessUnit;
    ///序号
    TZQThostFtdcSequenceNoType    SequenceNo;
    ///交易日
    TZQThostFtdcDateType TradingDay;
    ///结算编号
    TZQThostFtdcSettlementIDType   SettlementID;
};
```

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。

6.3.13. 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 OnRspQry Investor (
                    CZQThostFtdcInvestorField *pInvestor,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast);
```

参数：

p Investor：指向投资者信息结构的地址。

投资者信息结构：

```
struct CZQThostFtdcInvestorField
{
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者分组代码
    TZQThostFtdcInvestorIDType      InvestorGroupID;
    ///投资者名称
    TZQThostFtdcPartyNameType      InvestorName;
    ///证件类型
    TZQThostFtdcIdCardTypeType     IdentifiedCardType;
    ///证件号码
```



```
    TZQThostFtdcIdentifiedCardNoType    IdentifiedCardNo;
    ///是否活跃
    TZQThostFtdcBoolType IsActive;
};
```

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。

6.3.14. OnRspQryInvestorPosition 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 OnRspQry InvestorPosition(
                        CZQThostFtdcInvestorPositionField *pInvestorPosition,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast);
```

参数：

pInvestorPosition：指向投资者持仓应答结构的地址。

投资者持仓应答结构：

```
struct CZQThostFtdcInvestorPositionField
{
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///经纪公司代码
```

```
TZQThostFtdcBrokerIDType BrokerID;
///投资者代码
TZQThostFtdcInvestorIDType      InvestorID;
///持仓多空方向
TZQThostFtdcPosiDirectionType  PosiDirection;
///投机套保标志
TZQThostFtdcHedgeFlagType      HedgeFlag;
///持仓日期
TZQThostFtdcPositionDateType   PositionDate;
///上日持仓
TZQThostFtdcVolumeType   YdPosition;
///今日持仓
TZQThostFtdcVolumeType   Position;
///多头冻结
TZQThostFtdcVolumeType   LongFrozen;
///空头冻结
TZQThostFtdcVolumeType   ShortFrozen;
///开仓冻结金额
TZQThostFtdcMoneyType    LongFrozenAmount;
///开仓冻结金额
TZQThostFtdcMoneyType    ShortFrozenAmount;
///开仓量
TZQThostFtdcVolumeType   OpenVolume;
///平仓量
TZQThostFtdcVolumeType   CloseVolume;
///开仓金额
TZQThostFtdcMoneyType    OpenAmount;
///平仓金额
TZQThostFtdcMoneyType    CloseAmount;
///持仓成本
TZQThostFtdcMoneyType    PositionCost;
///上次占用的保证金
TZQThostFtdcMoneyType    PreMargin;
///占用的保证金
TZQThostFtdcMoneyType    UseMargin;
///冻结的保证金
TZQThostFtdcMoneyType    FrozenMargin;
///冻结的资金
TZQThostFtdcMoneyType    FrozenCash;
///冻结的手续费
TZQThostFtdcMoneyType    FrozenCommission;
///资金差额
TZQThostFtdcMoneyType    CashIn;
///手续费
```



```
    TZQThostFtdcMoneyType    Commission;
    ///平仓盈亏
    TZQThostFtdcMoneyType    CloseProfit;
    ///持仓盈亏
    TZQThostFtdcMoneyType    PositionProfit;
    ///上次结算价
    TZQThostFtdcPriceType    PreSettlementPrice;
    ///本次结算价
    TZQThostFtdcPriceType    SettlementPrice;
    ///交易日
    TZQThostFtdcDateType TradingDay;
    ///结算编号
    TZQThostFtdcSettlementIDType   SettlementID;
};
```

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。

6.3.15. 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：

```
struct CZQThostFtdcTradingAccountField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者帐号
    TZQThostFtdcAccountIDType    AccountID;
    ///上次质押金额
    TZQThostFtdcMoneyType    PreMortgage;
    ///上次信用额度
    TZQThostFtdcMoneyType    PreCredit;
    ///上次存款额
    TZQThostFtdcMoneyType    PreDeposit;
    ///上次结算准备金
    TZQThostFtdcMoneyType    PreBalance;
    ///上次占用的保证金
    TZQThostFtdcMoneyType    PreMargin;
    ///利息基数
    TZQThostFtdcMoneyType    InterestBase;
    ///利息收入
    TZQThostFtdcMoneyType    Interest;
    ///入金金额
    TZQThostFtdcMoneyType    Deposit;
    ///出金金额
    TZQThostFtdcMoneyType    Withdraw;
    ///冻结的保证金
    TZQThostFtdcMoneyType    FrozenMargin;
    ///冻结的资金
    TZQThostFtdcMoneyType    FrozenCash;
```



```
///冻结的手续费
TZQThostFtdcMoneyType    FrozenCommission;
///当前保证金总额
TZQThostFtdcMoneyType    CurrMargin;
///资金差额
TZQThostFtdcMoneyType    CashIn;
///手续费
TZQThostFtdcMoneyType    Commission;
///平仓盈亏
TZQThostFtdcMoneyType    CloseProfit;
///持仓盈亏
TZQThostFtdcMoneyType    PositionProfit;
///期货结算准备金
TZQThostFtdcMoneyType    Balance;
///可用资金
TZQThostFtdcMoneyType    Available;
///可取资金
TZQThostFtdcMoneyType    WithdrawQuota;
///基本准备金
TZQThostFtdcMoneyType    Reserve;
///交易日
TZQThostFtdcDateType TradingDay;
///结算编号
TZQThostFtdcSettlementIDType   SettlementID;
///信用额度
TZQThostFtdcMoneyType    Credit;
///质押金额
TZQThostFtdcMoneyType    Mortgage;
///交易所保证金
TZQThostFtdcMoneyType    ExchangeMargin;
```

```
};
```

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。

6.3.16. 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：

```
struct CZQThostFtdcTradingCodeField
{
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///经纪公司代码
    TZQThostFtdcBrokerIDType        BrokerID;
```

```
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///交易编码
    TZQThostFtdcClientIDType ClientID;
    ///是否活跃
    TZQThostFtdcBoolType       IsActive;
};
```

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. 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：

```
struct CZQThostFtdcExchangeField
{
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///交易所名称
    TZQThostFtdcExchangeNameType      ExchangeName;
    ///交易所属性
    TZQThostFtdcExchangePropertyType  ExchangeProperty;
};
```

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. OnRspQryInstrument 方法

请求查询合约响应。当客户端发出请求查询合约指令后，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：指向合约结构的地址。

合约结构：

```
struct CZQThostFtdcInstrumentField
{
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///合约名称
    TZQThostFtdcInstrumentNameType     InstrumentName;
    ///合约在交易所的代码
    TZQThostFtdcExchangeInstIDType     ExchangeInstID;
    ///产品代码
    TZQThostFtdcInstrumentIDType   ProductID;
    ///产品类型
    TZQThostFtdcProductClassType   ProductClass;
    ///交割年份
    TZQThostFtdcYearType DeliveryYear;
    ///交割月
    TZQThostFtdcMonthType     DeliveryMonth;
    ///市价单最大下单量
    TZQThostFtdcVolumeType    MaxMarketOrderVolume;
    ///市价单最小下单量
    TZQThostFtdcVolumeType    MinMarketOrderVolume;
    ///限价单最大下单量
    TZQThostFtdcVolumeType    MaxLimitOrderVolume;
    ///限价单最小下单量
    TZQThostFtdcVolumeType    MinLimitOrderVolume;
    ///合约数量乘数
    TZQThostFtdcVolumeMultipleType     VolumeMultiple;
```



```
    ///最小变动价位
    TZQThostFtdcPriceType       PriceTick;
    ///创建日
    TZQThostFtdcDateType CreateDate;
    ///上市日
    TZQThostFtdcDateType OpenDate;
    ///到期日
    TZQThostFtdcDateType ExpireDate;
    ///开始交割日
    TZQThostFtdcDateType StartDelivDate;
    ///结束交割日
    TZQThostFtdcDateType EndDelivDate;
    ///合约生命周期状态
    TZQThostFtdcInstLifePhaseType   InstLifePhase;
    ///当前是否交易
    TZQThostFtdcBoolType IsTrading;
    ///持仓类型
    TZQThostFtdcPositionTypeType    PositionType;
    ///持仓日期类型
    TZQThostFtdcPositionDateTypeType    PositionDateType;
    ///多头保证金率
    TZQThostFtdcRatioType       LongMarginRatio;
    ///空头保证金率
    TZQThostFtdcRatioType       ShortMarginRatio;
};
```

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. 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：

```
struct CZQThostFtdcDepthMarketDataField
{
    ///交易日
    TZQThostFtdcDateType TradingDay;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///合约在交易所的代码
    TZQThostFtdcExchangeInstIDType      ExchangeInstID;
    ///最新价
    TZQThostFtdcPriceType       LastPrice;
```

```
///上次结算价
TZQThostFtdcPriceType	PreSettlementPrice;
///昨收盘
TZQThostFtdcPriceType	PreClosePrice;
///昨持仓量
TZQThostFtdcLargeVolumeType	PreOpenInterest;
///今开盘
TZQThostFtdcPriceType	OpenPrice;
///最高价
TZQThostFtdcPriceType	HighestPrice;
///最低价
TZQThostFtdcPriceType	LowestPrice;
///数量
TZQThostFtdcVolumeType	Volume;
///成交金额
TZQThostFtdcMoneyType	Turnover;
///持仓量
TZQThostFtdcLargeVolumeType	OpenInterest;
///今收盘
TZQThostFtdcPriceType	ClosePrice;
///本次结算价
TZQThostFtdcPriceType	SettlementPrice;
///涨停板价
TZQThostFtdcPriceType	UpperLimitPrice;
///跌停板价
TZQThostFtdcPriceType	LowerLimitPrice;
///昨虚实度
TZQThostFtdcRatioType	PreDelta;
///今虚实度
TZQThostFtdcRatioType	CurrDelta;
```



```
///最后修改时间
TZQThostFtdcTimeType	UpdateTime;
///最后修改毫秒
TZQThostFtdcMillisecType	UpdateMillisec;
///申买价一
TZQThostFtdcPriceType	BidPrice1;
///申买量一
TZQThostFtdcVolumeType	BidVolume1;
///申卖价一
TZQThostFtdcPriceType	AskPrice1;
///申卖量一
TZQThostFtdcVolumeType	AskVolume1;
///申买价二
TZQThostFtdcPriceType	BidPrice2;
///申买量二
TZQThostFtdcVolumeType	BidVolume2;
///申卖价二
TZQThostFtdcPriceType	AskPrice2;
///申卖量二
TZQThostFtdcVolumeType	AskVolume2;
///申买价三
TZQThostFtdcPriceType	BidPrice3;
///申买量三
TZQThostFtdcVolumeType	BidVolume3;
///申卖价三
TZQThostFtdcPriceType	AskPrice3;
///申卖量三
TZQThostFtdcVolumeType	AskVolume3;
///申买价四
TZQThostFtdcPriceType	BidPrice4;
```



```
    ///申买量四
    TZQThostFtdcVolumeType    BidVolume4;
    ///申卖价四
    TZQThostFtdcPriceType     AskPrice4;
    ///申卖量四
    TZQThostFtdcVolumeType    AskVolume4;
    ///申买价五
    TZQThostFtdcPriceType     BidPrice5;
    ///申买量五
    TZQThostFtdcVolumeType    BidVolume5;
    ///申卖价五
    TZQThostFtdcPriceType     AskPrice5;
    ///申卖量五
    TZQThostFtdcVolumeType    AskVolume5;
};
```

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. OnRspQryInvestorPositionDetail 方法

请求查询投资者持仓明细响应。当客户端发出请求请求查询投资者持仓明细指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestorPositionDetail(
                    CZQThostFtdcInvestorPositionDetailField *pInvestorPositionDetail,
                    CZQThostFtdcRspInfoField *pRspInfo,
                    int nRequestID,
                    bool bIsLast) ；
```

参数：

pInvestorPositionDetail：指向投资者持仓明细结构的地址。

投资者持仓明细结构：

```
struct CZQThostFtdcInvestorPositionDetailField
{
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///投机套保标志
    TZQThostFtdcHedgeFlagType      HedgeFlag;
    ///买卖方向
    TZQThostFtdcDirectionType Direction;
    ///开仓日期
    TZQThostFtdcDateType OpenDate;
    ///成交编号
    TZQThostFtdcTradeIDType   TradeID;
    ///数量
    TZQThostFtdcVolumeType    Volume;
    ///开仓价
    TZQThostFtdcPriceType       OpenPrice;
    ///交易日
    TZQThostFtdcDateType TradingDay;
```



```
    ///结算编号
    TZQThostFtdcSettlementIDType   SettlementID;
    ///成交类型
    TZQThostFtdcTradeTypeType      TradeType;
    ///组合合约代码
    TZQThostFtdcInstrumentIDType   CombInstrumentID;
};
```

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. OnRspQryInstrument 方法

合约查询应答。当客户端发出合约查询指令后，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：

pRspInstrument：指向合约结构的地址。

合约结构：

```
struct CZQThostFtdcInstrumentField
```



```
{
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///合约名称
    TZQThostFtdcInstrumentNameType     InstrumentName;
    ///合约在交易所的代码
    TZQThostFtdcExchangeInstIDType      ExchangeInstID;
    ///产品代码
    TZQThostFtdcInstrumentIDType   ProductID;
    ///产品类型
    TZQThostFtdcProductClassType   ProductClass;
    ///交割年份
    TZQThostFtdcYearType DeliveryYear;
    ///交割月
    TZQThostFtdcMonthType     DeliveryMonth;
    ///市价单最大下单量
    TZQThostFtdcVolumeType    MaxMarketOrderVolume;
    ///市价单最小下单量
    TZQThostFtdcVolumeType    MinMarketOrderVolume;
    ///限价单最大下单量
    TZQThostFtdcVolumeType    MaxLimitOrderVolume;
    ///限价单最小下单量
    TZQThostFtdcVolumeType    MinLimitOrderVolume;
    ///合约数量乘数
    TZQThostFtdcVolumeMultipleType      VolumeMultiple;
    ///最小变动价位
    TZQThostFtdcPriceType       PriceTick;
    ///创建日
    TZQThostFtdcDateType CreateDate;
    ///上市日
    TZQThostFtdcDateType OpenDate;
    ///到期日
    TZQThostFtdcDateType ExpireDate;
    ///开始交割日
    TZQThostFtdcDateType StartDelivDate;
    ///结束交割日
    TZQThostFtdcDateType EndDelivDate;
    ///合约生命周期状态
    TZQThostFtdcInstLifePhaseType   InstLifePhase;
    ///当前是否交易
    TZQThostFtdcBoolType IsTrading;
    ///持仓类型
```

```
    TZQThostFtdcPositionTypeType    PositionType;
    ///持仓日期类型
    TZQThostFtdcPositionDateTypeType    PositionDateType;
};
```

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。

6.3.22. 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：

```
struct CZQThostFtdcTradeField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
```

```
///成交编号
TZQThostFtdcTradeIDType   TradeID;
///买卖方向
TZQThostFtdcDirectionType Direction;
///报单编号
TZQThostFtdcOrderSysIDType    OrderSysID;
///会员代码
TZQThostFtdcParticipantIDType   ParticipantID;
///客户代码
TZQThostFtdcClientIDType  ClientID;
///交易角色
TZQThostFtdcTradingRoleType    TradingRole;
///合约在交易所的代码
TZQThostFtdcExchangeInstIDType      ExchangeInstID;
///开平标志
TZQThostFtdcOffsetFlagType      OffsetFlag;
///投机套保标志
TZQThostFtdcHedgeFlagType      HedgeFlag;
///价格
TZQThostFtdcPriceType       Price;
///数量
TZQThostFtdcVolumeType    Volume;
///成交时期
TZQThostFtdcDateType TradeDate;
///成交时间
TZQThostFtdcTimeType       TradeTime;
///成交类型
TZQThostFtdcTradeTypeType      TradeType;
///成交价来源
TZQThostFtdcPriceSourceType    PriceSource;
///交易所交易员代码
TZQThostFtdcTraderIDType  TraderID;
///本地报单编号
TZQThostFtdcOrderLocalIDType  OrderLocalID;
///结算会员编号
TZQThostFtdcParticipantIDType   ClearingPartID;
///业务单元
TZQThostFtdcBusinessUnitType   BusinessUnit;
///序号
TZQThostFtdcSequenceNoType    SequenceNo;
///交易日
TZQThostFtdcDateType TradingDay;
///结算编号
TZQThostFtdcSettlementIDType   SettlementID;
```

};

6.3.23. 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：

```
struct CZQThostFtdcOrderField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///报单价格条件
    TZQThostFtdcOrderPriceTypeType      OrderPriceType;
    ///买卖方向
    TZQThostFtdcDirectionType Direction;
    ///组合开平标志
    TZQThostFtdcCombOffsetFlagType     CombOffsetFlag;
    ///组合投机套保标志
    TZQThostFtdcCombHedgeFlagType     CombHedgeFlag;
    ///价格
    TZQThostFtdcPriceType       LimitPrice;
    ///数量
    TZQThostFtdcVolumeType    VolumeTotalOriginal;
    ///有效期类型
    TZQThostFtdcTimeConditionType       TimeCondition;
    ///GTD 日期
    TZQThostFtdcDateType GTDDate;
    ///成交量类型
```

```
TZQThostFtdcVolumeConditionType    VolumeCondition;
///最小成交量
TZQThostFtdcVolumeType    MinVolume;
///触发条件
TZQThostFtdcContingentConditionType      ContingentCondition;
///止损价
TZQThostFtdcPriceType       StopPrice;
///强平原因
TZQThostFtdcForceCloseReasonType  ForceCloseReason;
///自动挂起标志
TZQThostFtdcBoolType IsAutoSuspend;
///业务单元
TZQThostFtdcBusinessUnitType   BusinessUnit;
///请求编号
TZQThostFtdcRequestIDType      RequestID;
///本地报单编号
TZQThostFtdcOrderLocalIDType  OrderLocalID;
///交易所代码
TZQThostFtdcExchangeIDType    ExchangeID;
///会员代码
TZQThostFtdcParticipantIDType   ParticipantID;
///客户代码
TZQThostFtdcClientIDType  ClientID;
///合约在交易所的代码
TZQThostFtdcExchangeInstIDType      ExchangeInstID;
///交易所交易员代码
TZQThostFtdcTraderIDType  TraderID;
///安装编号
TZQThostFtdcInstallIDType  InstallID;
///报单提交状态
TZQThostFtdcOrderSubmitStatusType  OrderSubmitStatus;
///报单提示序号
TZQThostFtdcSequenceNoType    NotifySequence;
///交易日
TZQThostFtdcDateType TradingDay;
///结算编号
TZQThostFtdcSettlementIDType   SettlementID;
///报单编号
TZQThostFtdcOrderSysIDType    OrderSysID;
///报单来源
TZQThostFtdcOrderSourceType    OrderSource;
///报单状态
TZQThostFtdcOrderStatusType    OrderStatus;
///报单类型
```



```
    TZQThostFtdcOrderTypeType       OrderType;
    ///今成交数量
    TZQThostFtdcVolumeType    VolumeTraded;
    ///剩余数量
    TZQThostFtdcVolumeType    VolumeTotal;
    ///报单日期
    TZQThostFtdcDateType InsertDate;
    ///插入时间
    TZQThostFtdcTimeType       InsertTime;
    ///激活时间
    TZQThostFtdcTimeType       ActiveTime;
    ///挂起时间
    TZQThostFtdcTimeType       SuspendTime;
    ///最后修改时间
    TZQThostFtdcTimeType       UpdateTime;
    ///撤销时间
    TZQThostFtdcTimeType       CancelTime;
    ///最后修改交易所交易员代码
    TZQThostFtdcTraderIDType  ActiveTraderID;
    ///结算会员编号
    TZQThostFtdcParticipantIDType   ClearingPartID;
    ///序号
    TZQThostFtdcSequenceNoType    SequenceNo;
    ///前置编号
    TZQThostFtdcFrontIDType    FrontID;
    ///会话编号
    TZQThostFtdcSessionIDType       SessionID;
    ///用户端产品信息
    TZQThostFtdcProductInfoType     UserProductInfo;
    ///状态信息
    TZQThostFtdcErrorMsgType StatusMsg;
};
```

6.3.24. 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：

```
struct CZQThostFtdcInputOrderField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///报单价格条件
    TZQThostFtdcOrderPriceTypeType      OrderPriceType;
    ///买卖方向
    TZQThostFtdcDirectionType Direction;
    ///组合开平标志
    TZQThostFtdcCombOffsetFlagType     CombOffsetFlag;
    ///组合投机套保标志
    TZQThostFtdcCombHedgeFlagType     CombHedgeFlag;
    ///价格
    TZQThostFtdcPriceType       LimitPrice;
    ///数量
    TZQThostFtdcVolumeType    VolumeTotalOriginal;
    ///有效期类型
    TZQThostFtdcTimeConditionType       TimeCondition;
    ///GTD 日期
    TZQThostFtdcDateType GTDDate;
    ///成交量类型
    TZQThostFtdcVolumeConditionType    VolumeCondition;
    ///最小成交量
    TZQThostFtdcVolumeType    MinVolume;
    ///触发条件
    TZQThostFtdcContingentConditionType      ContingentCondition;
    ///止损价
    TZQThostFtdcPriceType       StopPrice;
    ///强平原因
    TZQThostFtdcForceCloseReasonType  ForceCloseReason;
    ///自动挂起标志
    TZQThostFtdcBoolType IsAutoSuspend;
```

```
    ///业务单元
    TZQThostFtdcBusinessUnitType   BusinessUnit;
    ///请求编号
    TZQThostFtdcRequestIDType      RequestID;
};
```

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;
};
```

6.3.25. 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：

```
struct CZQThostFtdcOrderActionField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///报单操作引用
    TZQThostFtdcOrderActionRefType      OrderActionRef;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///请求编号
    TZQThostFtdcRequestIDType      RequestID;
    ///前置编号
```

```
    TZQThostFtdcFrontIDType    FrontID;
    ///会话编号
    TZQThostFtdcSessionIDType      SessionID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///报单编号
    TZQThostFtdcOrderSysIDType    OrderSysID;
    ///操作标志
    TZQThostFtdcActionFlagType     ActionFlag;
    ///价格
    TZQThostFtdcPriceType      LimitPrice;
    ///数量变化
    TZQThostFtdcVolumeType    VolumeChange;
    ///操作日期
    TZQThostFtdcDateType ActionDate;
    ///操作时间
    TZQThostFtdcTimeType       ActionTime;
    ///交易所交易员代码
    TZQThostFtdcTraderIDType  TraderID;
    ///安装编号
    TZQThostFtdcInstallIDType  InstallID;
    ///本地报单编号
    TZQThostFtdcOrderLocalIDType  OrderLocalID;
    ///操作本地编号
    TZQThostFtdcOrderLocalIDType  ActionLocalID;
    ///会员代码
    TZQThostFtdcParticipantIDType   ParticipantID;
    ///客户代码
    TZQThostFtdcClientIDType  ClientID;
    ///业务单元
    TZQThostFtdcBusinessUnitType   BusinessUnit;
    ///报单操作状态
    TZQThostFtdcOrderActionStatusType  OrderActionStatus;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///状态信息
    TZQThostFtdcErrorMsgType StatusMsg;
};
```

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. OnRtnFundIntoCTPAccount 方法

资金转入 CTP 通知。

函数原形：

```
Void    OnRtnFundIntoCTPAccount(CZQThostFtdcFundIOCTPAccountField
*pFundIOCTPAccount) ;
```

参数：

pFundIOCTPAccount：指向资金转入通知的地址，包含了资金转入通知的金额、方向、转账序号等。

```
///资金转入转出 CTP
struct CZQThostFtdcFundIOCTPAccountField
{
    ///证券公司代码
    TZQThostFtdcBrokerIDType  BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///投资者资金帐号
    TZQThostFtdcAccountIDType      AccountID;
    ///资金帐户密码
    TZQThostFtdcPasswordType  Password;
    ///用户代码
    TZQThostFtdcUserIDType     UserID;
    ///会话编号
    TZQThostFtdcSessionIDType SessionID;
    ///CTP 核心流水号
    TZQThostFtdcFutureSerialType    CTPSerial;
    ///转账平台流水号
```

```
    TZQThostFtdcPlateSerialType      PlateSerial;
    ///第三方流水号
    TZQThostFtdcBankSerialType      SettlementSerial;
    ///交易金额
    TZQThostFtdcTradeAmountType  TradeAmount;
    ///交易日
    TZQThostFtdcDateType   TradingDay;
    ///转账时间
    TZQThostFtdcTimeType  TradeTime;
    ///摘要
    TZQThostFtdcDigestType      Digest;
    ///出入金方向
    TZQThostFtdcFundDirectionType FundDirection;
    ///错误代码
    TZQThostFtdcErrorIDType     ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

6.3.27. OnRspFundOutCTPAccount 方法

资金转出 CTP 请求应答。当客户端发出资金转出 CTP 请求指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnFundOutCTPAccount (
                        CZQThostFtdcRspFundIOCTPAccountField
                        *pRspFundIOCTPAccount,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast);
```

参数：

pRspFundIOCTPAccount：指向资金转出应答的地址。

```
///资金转入转出应答
struct CZQThostFtdcRspFundIOCTPAccountField
{
    ///证券公司代码
    TZQThostFtdcBrokerIDType  BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///投资者资金帐号
    TZQThostFtdcAccountIDType      AccountID;
    ///资金帐户密码
    TZQThostFtdcPasswordType  Password;
    ///用户代码
    TZQThostFtdcUserIDType     UserID;
    ///会话编号
    TZQThostFtdcSessionIDType SessionID;
    ///CTP 核心流水号
    TZQThostFtdcFutureSerialType    CTPSerial;
    ///转账平台流水号
    TZQThostFtdcPlateSerialType      PlateSerial;
    ///第三方流水号
    TZQThostFtdcBankSerialType      SettlementSerial;
    ///交易金额
    TZQThostFtdcTradeAmountType  TradeAmount;
    ///交易日
    TZQThostFtdcDateType   TradingDay;
    ///转账时间
    TZQThostFtdcTimeType  TradeTime;
    ///摘要
    TZQThostFtdcDigestType      Digest;
```

```
    ///出入金方向
    TZQThostFtdcFundDirectionType FundDirection;
};
```

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. OnRtnFundOutCTPAccount 方法

资金转出 CTP 通知。

函数原形：

```
Void    OnRtnFundOutCTPAccount(CZQThostFtdcFundIOCTPAccountField
*pFundIOCTPAccount) ;
```

参数：

pFundIOCTPAccount：指向资金转入通知的地址，包含了资金转入通知的金额、方向、转账序号等。

```
///资金转入转出 CTP
struct CZQThostFtdcFundIOCTPAccountField
{
    ///证券公司代码
    TZQThostFtdcBrokerIDType  BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///投资者资金帐号
    TZQThostFtdcAccountIDType      AccountID;
```



```
    ///资金帐户密码
    TZQThostFtdcPasswordType  Password;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///会话编号
    TZQThostFtdcSessionIDType SessionID;
    ///CTP 核心流水号
    TZQThostFtdcFutureSerialType    CTPSerial;
    ///转账平台流水号
    TZQThostFtdcPlateSerialType     PlateSerial;
    ///第三方流水号
    TZQThostFtdcBankSerialType      SettlementSerial;
    ///交易金额
    TZQThostFtdcTradeAmountType  TradeAmount;
    ///交易日
    TZQThostFtdcDateType   TradingDay;
    ///转账时间
    TZQThostFtdcTimeType  TradeTime;
    ///摘要
    TZQThostFtdcDigestType      Digest;
    ///出入金方向
    TZQThostFtdcFundDirectionType FundDirection;
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```


6.3.29. OnRspQryFundIOCTPAccount 方法

资金转入转出 CTP 查询应答。当客户端发出资金转入转出 CTP 查询请求指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryFundIOCTPAccount (
                        CZQThostFtdcFundIOCTPAccountField
                        *pRspFundIOCTPAccount,
                        CZQThostFtdcRspInfoField *pRspInfo,
                        int nRequestID,
                        bool bIsLast)；
```

参数：

CZQThostFtdcFundIOCTPAccountField：指向资金转入转出结构的地址。

```
///资金转入转出 CTP
struct CZQThostFtdcFundIOCTPAccountField
{
    ///证券公司代码
    TZQThostFtdcBrokerIDType  BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///投资者资金帐号
    TZQThostFtdcAccountIDType      AccountID;
    ///资金帐户密码
    TZQThostFtdcPasswordType  Password;
    ///用户代码
    TZQThostFtdcUserIDType     UserID;
    ///会话编号
    TZQThostFtdcSessionIDType SessionID;
    ///CTP 核心流水号
    TZQThostFtdcFutureSerialType    CTPSerial;
    ///转账平台流水号
```

```
    TZQThostFtdcPlateSerialType     PlateSerial;
    ///第三方流水号
    TZQThostFtdcBankSerialType     SettlementSerial;
    ///交易金额
    TZQThostFtdcTradeAmountType  TradeAmount;
    ///交易日
    TZQThostFtdcDateType   TradingDay;
    ///转账时间
    TZQThostFtdcTimeType  TradeTime;
    ///摘要
    TZQThostFtdcDigestType     Digest;
    ///出入金方向
    TZQThostFtdcFundDirectionType FundDirection;
    ///错误代码
    TZQThostFtdcErrorIDType    ErrorID;
    ///错误信息
    TZQThostFtdcErrorMsgType  ErrorMsg;
};
```

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。

6.4. 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. GetTradingDay 方法

获得当前交易日。只有当与交易托管系统连接建立后才会取到正确的值。

函数原形：

const char *GetTradingDay();

返回值：

返回一个指向日期信息字符串的常量指针。

6.4.6. 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。

6.4.7. 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。

6.4.8. 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

6.4.9. 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

6.4.10. 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：

```
struct CZQThostFtdcReqUserLoginField
{
    ///交易日
    TZQThostFtdcDateType TradingDay;
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///密码
    TZQThostFtdcPasswordType Password;
    ///用户端产品信息
    TZQThostFtdcProductInfoType    UserProductInfo;
    ///接口端产品信息
    TZQThostFtdcProductInfoType    InterfaceProductInfo;
    ///协议信息
    TZQThostFtdcProtocolInfoType    ProtocolInfo;
};
```

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：SFITTraderV100。

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;

memset(&qryFld,0,sizeof(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);
```

6.4.11. 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：

```
struct CZQThostFtdcUserLogoutField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
};
```

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

6.4.12. 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：

```
struct CZQThostFtdcUserPasswordUpdateField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///原来的口令
    TZQThostFtdcPasswordType OldPassword;
    ///新的口令
    TZQThostFtdcPasswordType NewPassword;
};
```

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;
memset(&qryField,0,sizeof(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);
```

6.4.13. ReqTradingAccountPasswordUpdate 方法

资金账户口令更新请求。

函数原形：

```
int ReqTradingAccountPasswordUpdate(
                    CZQThostFtdcTradingAccountPasswordUpdateField
                    *pTradingAccountPasswordUpdate,
                    int nRequestID) ;
```

参数：

pUserPasswordUpdate：指向资金账户口令修改结构的地址。

资金账户口令修改结构：

```
struct CZQThostFtdcTradingAccountPasswordUpdateField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者帐号
    TZQThostFtdcAccountIDType    AccountID;
    ///原来的口令
    TZQThostFtdcPasswordType OldPassword;
    ///新的口令
    TZQThostFtdcPasswordType NewPassword;
};
```

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：

```
CZQThostFtdcTradingAccountPasswordUpdateField qryField;
memset(&qryField,0,sizeof(qryField));
strcpy(qryField.BrokerID, "8000");
strcpy(qryField.AccountID, "000001");
strcpy(qryField.OldPassword, "1");
strcpy(qryField.NewPassword, "2");
int iResult = =g_pTradeStockApi->ReqTradingAccountPasswordUpdate(&qryField,0));
```

6.4.14. 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：

```
struct CThostFtdcInputOrderField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///报单引用
```



```
    TZQThostFtdcOrderRefType OrderRef;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///报单价格条件
    TZQThostFtdcOrderPriceTypeType      OrderPriceType;
    ///买卖方向
    TZQThostFtdcDirectionType Direction;
    ///组合开平标志
    TZQThostFtdcCombOffsetFlagType     CombOffsetFlag;
    ///组合投机套保标志
    TZQThostFtdcCombHedgeFlagType     CombHedgeFlag;
    ///价格
    TZQThostFtdcPriceType       LimitPrice;
    ///数量
    TZQThostFtdcVolumeType    VolumeTotalOriginal;
    ///有效期类型
    TZQThostFtdcTimeConditionType       TimeCondition;
    ///GTD 日期
    TZQThostFtdcDateType GTDDate;
    ///成交量类型
    TZQThostFtdcVolumeConditionType    VolumeCondition;
    ///最小成交量
    TZQThostFtdcVolumeType    MinVolume;
    ///触发条件
    TZQThostFtdcContingentConditionType      ContingentCondition;
    ///止损价
    TZQThostFtdcPriceType       StopPrice;
    ///强平原因
    TZQThostFtdcForceCloseReasonType  ForceCloseReason;
    ///自动挂起标志
    TZQThostFtdcBoolType IsAutoSuspend;
    ///业务单元
    TZQThostFtdcBusinessUnitType   BusinessUnit;
    ///请求编号
    TZQThostFtdcRequestIDType      RequestID;
};
```

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。

因为交易后台按照字符串比较 OrdeRef 的大小，所以在设置 OrderRef 时要填满 TZQThostFtdcOrderRefType 的全部空间。

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。

6.4.15. 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 *pOrderAction,
                    int nRequestID);
```

参数：

pOrderAction：指向报单操作结构的地址。

报单操作结构：

```
struct CZQThostFtdcOrderActionField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///报单操作引用
    TZQThostFtdcOrderActionRefType      OrderActionRef;
    ///报单引用
    TZQThostFtdcOrderRefType OrderRef;
    ///请求编号
    TZQThostFtdcRequestIDType       RequestID;
```

```
    ///前置编号
    TZQThostFtdcFrontIDType    FrontID;
    ///会话编号
    TZQThostFtdcSessionIDType    SessionID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///报单编号
    TZQThostFtdcOrderSysIDType    OrderSysID;
    ///操作标志
    TZQThostFtdcActionFlagType    ActionFlag;
    ///价格
    TZQThostFtdcPriceType    LimitPrice;
    ///数量变化
    TZQThostFtdcVolumeType    VolumeChange;
    ///操作日期
    TZQThostFtdcDateType ActionDate;
    ///操作时间
    TZQThostFtdcTimeType    ActionTime;
    ///交易所交易员代码
    TZQThostFtdcTraderIDType  TraderID;
    ///安装编号
    TZQThostFtdcInstallIDType  InstallID;
    ///本地报单编号
    TZQThostFtdcOrderLocalIDType  OrderLocalID;
    ///操作本地编号
    TZQThostFtdcOrderLocalIDType  ActionLocalID;
    ///会员代码
    TZQThostFtdcParticipantIDType   ParticipantID;
    ///客户代码
    TZQThostFtdcClientIDType  ClientID;
    ///业务单元
    TZQThostFtdcBusinessUnitType   BusinessUnit;
    ///报单操作状态
    TZQThostFtdcOrderActionStatusType  OrderActionStatus;
    ///用户代码
    TZQThostFtdcUserIDType    UserID;
    ///状态信息
    TZQThostFtdcErrorMsgType StatusMsg;
};
```

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;
memset(&fldAction, 0, sizeof(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 = THOST_FTDC_AF_Delete;
int iReq = g_pTradeStockApi->ReqOrderAction(&fldAction,0);
```

6.4.16. 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：

```
struct CZQThostFtdcQryOrderField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///报单编号
    TZQThostFtdcOrderSysIDType    OrderSysID;
};
```

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



```
strcpy(qryField.InvestorId, "000001");

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

6.4.17. 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：

```
struct CZQThostFtdcQryTradeField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///成交编号
    TZQThostFtdcTradeIDType   TradeID;
};
```

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;
    memset(&qryField, 0, sizeof(qryField));
    int iReq = g_pTradeStockApi-> ReqQryTrade (&qryField, 0);
//查询投资者 000001 的成交
    CZQThostFtdcQryTradeField qryField;
    memset(&qryField, 0, sizeof(qryField));
    strcpy(qryField.BrokerID, "8000 ");
    strcpy(qryField.InvestorId, "000001");
    int iReq = g_pTradeStockApi-> ReqQryTrade (&qryField, 0);
```

6.4.18. ReqQryInvestor 方法

会员客户查询请求。

函数原形：

```
int ReqQry Investor (
                CZQThostFtdcQryInvestorField *pQryInvestor,
                int nRequestID);
```

参数：

pQry Investor：指向客户查询结构的地址。

客户查询结构：

```
struct CZQThostFtdcQryInvestorField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
};
```

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

6.4.19. ReqQryInvestorPosition 方法

会员持仓查询请求。

函数原形：

```
int ReqQryInvestorPosition(
                    CZQThostFtdcQryInvestorPositionField *pQryInvestorPosition,
                    int nRequestID);
```

参数：

pQryInvestorPosition：指向会员持仓查询结构的地址。

会员持仓查询结构：

```
struct CZQThostFtdcQryInvestorPositionField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
};
```


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
    CZQThostFtdcQryInvestorField qryField;
    memset(&qryField, 0, sizeof(qryField));
    int iReq = g_pTradeStockApi-> ReqQryInvestorPosition(&qryField, 0);
//查询投资者 000001 的持仓
    CZQThostFtdcQryTradeField qryField;
    memset(&qryField, 0, sizeof(qryField));
    strcpy(qryField.BrokerID, "8000");
    strcpy(qryField.InvestorId, "000001");
    int iReq = g_pTradeStockApi-> ReqQryInvestorPosition(&qryField, 0);
```

6.4.20. 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：

```
struct CZQThostFtdcQryTradingAccountField
{
    ///经纪公司代码
```

```
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
};
```

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;

    memset(&qryField, 0, sizeof(qryField));

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

//查询投资者 000001 的资金账户

    CZQThostFtdcQryTradingAccountField qryField;

    memset(&qryField, 0, sizeof(qryField));

    strcpy(qryField.BrokerID, "8000");

    strcpy(qryField.InvestorId, "000001");

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

6.4.21. ReqQryTradingCode 方法

请求查询交易编码。

函数原形：

```
int ReqQryTradingCode(
                    CThostFtdcQryTradingCodeField *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：

```
struct CThostFtdcQryTradingCodeField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///交易编码
    TZQThostFtdcClientIDType  ClientID;
};
```

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;

    memset(&qryField, 0, sizeof(qryField));

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

//查询投资者 000001 在 SSE 的交易编码

    CThostFtdcQryTradingCodeField qryField;

    memset(&qryField, 0, sizeof(qryField));

    strcpy(qryField.BrokerID, "8000 ");

    strcpy(qryField.InvestorId, "000001");

    strcpy(qryField.ExchangeId, "SSE");

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

6.4.22. 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：

```
struct CZQThostFtdcQryExchangeField
{
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
};
```

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;
    memset(&qryField, 0, sizeof(qryField));
    int iReq = g_pTradeStockApi-> ReqQryExchange(&qryField, 0);
// 查询 SSE 的信息
    CZQThostFtdcQryExchangeField qryField;
    memset(&qryField, 0, sizeof(qryField));
```



```
strcpy(qryField.ExchangId, "SSE");
int iReq = g_pTradeStockApi-> ReqQryExchange(&qryField, 0);
```

6.4.23. ReqQryInstrument 方法

请求查询合约。

函数原形：

```
int ReqQryInstrument(
        CZQThostFtdcQryInstrumentField *pQryInstrument,
        int nRequestID);
```

参数：

pQryInstrument：指向查询查询合约结构的地址。

查询合约结构：

```
struct CZQThostFtdcQryInstrumentField
{
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///合约在交易所的代码
    TZQThostFtdcExchangeInstIDType      ExchangeInstID;
    ///产品代码
    TZQThostFtdcInstrumentIDType   ProductID;
};
```

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

6.4.24. 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：

```
struct CZQThostFtdcQryDepthMarketDataField
{
    ///合约代码
    TZQThostFtdcInstrumentIDType  InstrumentID;
};
```

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 000001 的合约行情信息
//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;
    memset(&qryField, 0, sizeof(qryField));
    strcpy(qryField.InstrumentID, "000001");
    int iReq = g_pTradeStockApi-> ReqQryDepthMarketData(&qryField, 0);
```

6.4.25. ReqQryInvestorPositionDetail 方法

请求查询投资者持仓明细。

函数原形：

```
int ReqQryInvestorPositionDetail(
        CZQThostFtdcQryInvestorPositionDetailField *pQryInvestorPositionDetail,
        int nRequestID);
```

参数：

pQryInvestorPositionDetail：指向查询投资者持仓明细结构的地址。

查询投资者持仓明细结构：

```
struct CZQThostFtdcQryInvestorPositionDetailField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
```

```
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
};
```

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：

```
// 所有投资者持仓明细
    CZQThostFtdcQryInvestorPositionDetailField qryField;
    memset(&qryField, 0, sizeof(qryField));
    int iReq = g_pTradeStockApi-> ReqQryInvestorPositionDetail(&qryField, 0);
// 查询投资者 000001 的持仓明细
    CZQThostFtdcQryInvestorPositionDetailField qryField;
    memset(&qryField, 0, sizeof(qryField));
    strcpy(qryField.BrokerID, "8000 ");
    strcpy(qryField.InvestorId, "000001");
    strcpy(qryField.ExchangId, "SSE");
    int iReq = g_pTradeStockApi-> ReqQryInvestorPositionDetail(&qryField, 0);
```

6.4.26. ReqQryBondInterest 方法

请求查询债券利息。

函数原形：

```
int ReqQryBondInterest (
                        CZQThostFtdcQryBondInterestField
                       *pQryBondInterest, int nRequestID);
```

参数：

pQryBondInterest：指向债券利息结构的地址。

查询债券利息结构：

```
struct CZQThostFtdcQryBondInterestField
{
    ///交易所代码
    TZQThostFtdcExchangeIDType    ExchangeID;
    ///合约代码
    TZQThostFtdcInstrumentIDType   InstrumentID;
};
```

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：

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

6.4.27. ReqFundOutCTPAccount 方法

资金转出 CTP 请求。

函数原形：

```
int ReqFundOutCTPAccount (
                                CZQThostFtdcReqFundIOCTPAccountField
                               *pReqFundIOCTPAccount, int nRequestID);
```

参数：

pReqFundIOCTPAccount：指向资金转入转出请求的地址。

资金转入转出请求结构：

```
struct CZQThostFtdcReqFundIOCTPAccountField
{
    ///证券公司代码
    TZQThostFtdcBrokerIDType  BrokerID;
    ///投资者代码
    TZQThostFtdcInvestorIDType      InvestorID;
    ///投资者资金帐号
```

```
    TZQThostFtdcAccountIDType      AccountID;
    ///资金帐户密码
    TZQThostFtdcPasswordType  Password;
    ///用户代码
    TZQThostFtdcUserIDType     UserID;
    ///会话编号
    TZQThostFtdcSessionIDType SessionID;
    ///CTP 核心流水号
    TZQThostFtdcFutureSerialType    CTPSerial;
    ///转账平台流水号
    TZQThostFtdcPlateSerialType      PlateSerial;
    ///第三方流水号
    TZQThostFtdcBankSerialType      SettlementSerial;
    ///交易金额
    TZQThostFtdcTradeAmountType  TradeAmount;
    ///交易日
    TZQThostFtdcDateType   TradingDay;
    ///转账时间
    TZQThostFtdcTimeType  TradeTime;
    ///摘要
    TZQThostFtdcDigestType      Digest;
};
```

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：

```
CZQThostFtdcReqFundIOCTPAccountField qryField;
memset(&qryField,0,sizeof(qryField));
//必须填写的字段
strcpy(qryField.BrokerID, "8000");
strcpy(qryField.InvestorID, "000001");
strcpy(qryField.UserID, "8000_admin");
qryField.SessionID = -25668890;
strcpy(qryField.Password, "1");
qryField.TradeAmount = 1000.0;
//其余字段可以不填
int iReq = g_pTradeStockApi->ReqFundOutCTPAccount(&qryField, 0);
```

6.4.28. ReqQryFundIOCTPAccount 方法

请求查询债券利息。

函数原形：

int ReqQryFundIOCTPAccount (
CZQThostFtdcQryFundIOCTPAccountField
*pQryFundIOCTPAccount, int nRequestID);

参数：

pQryFundIOCTPAccount：指向查询资金转入转出 CTP 的地址。

查询资金转入转出 CTP 结构：

```
struct CZQThostFtdcQryFundIOCTPAccountField
{
    ///经纪公司代码
    TZQThostFtdcBrokerIDType  BrokerID;
    ///投资者资金帐号
    TZQThostFtdcAccountIDType      AccountID;
```

};

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：

```
// 所有出入金记录
    CZQThostFtdcQryFundIOCTPAccountField qryField;
    memset(&qryField, 0, sizeof(qryField));
    int iReq = g_pTradeStockApi-> ReqQryFundIOCTPAccount (&qryField, 0);
// 查询 01000000047401 的出入金记录
    CZQThostFtdcQryFundIOCTPAccountField qryField;
    memset(&qryField, 0, sizeof(qryField));
    strcpy(qryField.BrokerID, "8000");
    strcpy(qryField.AccountID, "01000000047401");
    int iReq = g_pTradeStockApi-> ReqQryFundIOCTPAccount (&qryField, 0);
```

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);
            }
            // 端登成功,发出报单录入请求
        CZQThostFtdcInputOrderField ordField;
        memset(&ordField,0,sizeof(ordField));
    //界面交互字段
        strcpy(ordField.InvestorID, "698049");
```

```
        strcpy(ordField.InstrumentID, "0000001");
        strcpy(ordField.ExchangeID,"SZE") :
        ordField.Direction =  THOST_FTDC_D_Buy;
        strcpy(ordField.LimitPrice, "10.34");
        ordField.VolumeTotalOriginal = 100;
        //身份信息
        strcpy(ordField.BrokerID, g_chBrokerID);
        strcpy(ordField.UserID, g_chUserID);
       //以下是报单默认字段
       ordField.OrderPriceType = THOST_FTDC_OPT_LimitPrice;
        ordField.CombOffsetFlag[0] = THOST_FTDC_OF_Open;
        ordField.CombHedgeFlag[0] = THOST_FTDC_HF_Speculation;
        ordField.TimeCondition  = THOST_FTDC_TC_GFD;
        ordField.VolumeCondition = THOST_FTDC_VC_AV;
        ordField.ContingentCondition = THOST_FTDC_CC_Immediately;
        ordField.ForceCloseReason = THOST_FTDC_FCC_NotForceClose;
        ordField.IsAutoSuspend = 0;
        ordField.RequestID =  1;
        ordField.UserForceClose = 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;
```

