拓界科技黄金QDP

交易API开发手册

2025年10月日

文档标识

|  |  |
| --- | --- |
| 项目名称 | QDP黄金柜台交易系统 |
| 文档名称 | Trader API |
| 版本号 | 1.0 |
| 状况 |  |

文档修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **描述** | **修订者** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

此版本文档的正式核准

|  |  |  |
| --- | --- | --- |
| **姓名** | **签字** | **日期** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**目录**

[1. 介绍 1](#_Toc111121579)

[2. 体系结构 2](#_Toc111121580)

[2.1 通讯模式 2](#_Toc111121581)

[2.2 数据流 3](#_Toc111121582)

[3. 接口模式 5](#_Toc111121583)

[3.1 对话流和查询流编程接口 5](#_Toc111121584)

[3.2 私有流编程接口 5](#_Toc111121585)

[4. 运行模式 7](#_Toc111121586)

[4.1 工作线程 7](#_Toc111121587)

[4.2 本地文件 7](#_Toc111121588)

[5. 业务和接口对照 8](#_Toc111121589)

[6. 开发接口 11](#_Toc111121590)

[6.1 通用规则 11](#_Toc111121591)

[6.2 CQdpFtdcTraderSpi接口 11](#_Toc111121592)

[6.2.1 OnFrontConnected 方法 11](#_Toc111121593)

[6.2.2 OnFrontDisconnected 方法 11](#_Toc111121594)

[6.2.3 OnHeartBeatWarning方法 12](#_Toc111121595)

[6.2.4 OnPackageStart方法 12](#_Toc111121596)

[6.2.5 OnPackageEnd方法 12](#_Toc111121597)

[6.2.6 OnRspError 方法 13](#_Toc111121598)

[6.2.7 OnRspUserLogin方法 13](#_Toc111121599)

[6.2.8 OnRspUserLogout 方法 14](#_Toc111121600)

[6.2.9 OnRspUserPasswordUpdate方法 15](#_Toc111121601)

[6.2.10 OnRspOrderInsert 方法 16](#_Toc111121602)

[6.2.11 OnRspOrderAction 方法 18](#_Toc111121603)

[6.2.12 OnRtnFlowMessageCancel方法 20](#_Toc111121604)

[6.2.13 OnRtnTrade 方法 20](#_Toc111121605)

[6.2.14 OnRtnOrder 方法 22](#_Toc111121606)

[6.2.15 OnErrRtnOrderInsert方法 24](#_Toc111121607)

[6.2.16 OnErrRtnOrderAction方法 26](#_Toc111121608)

[6.2.17 OnRtnInstrumentStatus方法 27](#_Toc111121609)

[6.2.18 OnRtnInvestorAccountDeposit 方法 29](#_Toc111121610)

[6.2.19 OnRspQryOrder 方法 30](#_Toc111121611)

[6.2.20 OnRspQryTrade 方法 32](#_Toc111121612)

[6.2.21 OnRspQryUserInvestor方法 34](#_Toc111121613)

[6.2.22 OnRspQryInvestorAccount方法 35](#_Toc111121614)

[6.2.23 OnRspQryInstrument方法 37](#_Toc111121615)

[6.2.24 OnRspQryMarketData方法 39](#_Toc111121616)

[6.2.25 OnRspQryExchange方法 41](#_Toc111121617)

[6.2.26 OnRspQryInvestorPosition方法 42](#_Toc111121618)

[6.2.27 OnRspSubscribeTopic方法 44](#_Toc111121619)

[6.2.28 OnRspQryTopic方法 44](#_Toc111121620)

[6.2.29 OnRspQryInvestorFee方法 45](#_Toc111121621)

[6.2.30 OnRspQryInvestorMargin方法 46](#_Toc111121622)

[6.2.31 OnRspQrySGEDeferRate方法 48](#_Toc111121623)

[6.3 CQdpFtdcTraderApi 接口 49](#_Toc111121624)

[6.3.1 CreateFtdcTraderApi方法 49](#_Toc111121625)

[6.3.2 GetVersion方法 49](#_Toc111121626)

[6.3.3 Release 方法 49](#_Toc111121627)

[6.3.4 Init 方法 50](#_Toc111121628)

[6.3.5 Join 方法 50](#_Toc111121629)

[6.3.6 GetTradingDay方法 50](#_Toc111121630)

[6.3.7 RegisterFront 方法 50](#_Toc111121631)

[6.3.8 RegisterNameServer 方法 50](#_Toc111121632)

[6.3.9 RegisterSpi 方法 51](#_Toc111121633)

[6.3.10 SubscribePrivateTopic方法 51](#_Toc111121634)

[6.3.11 SubscribePublicTopic方法 51](#_Toc111121635)

[6.3.12 SubscribeUserTopic方法 52](#_Toc111121636)

[6.3.13 SetHeartbeatTimeout方法 52](#_Toc111121637)

[6.3.14 OpenRequestLog方法 52](#_Toc111121638)

[6.3.15 OpenResponseLog方法 53](#_Toc111121639)

[6.3.16 ReqUserLogin方法 53](#_Toc111121640)

[6.3.17 ReqUserLogout方法 54](#_Toc111121641)

[6.3.18 ReqUserPasswordUpdate方法 55](#_Toc111121642)

[6.3.19 ReqOrderInsert 方法 55](#_Toc111121643)

[6.3.20 ReqOrderAction 方法 59](#_Toc111121644)

[6.3.21 ReqQryOrder方法 60](#_Toc111121645)

[6.3.22 ReqQryTrade方法 60](#_Toc111121646)

[6.3.23 ReqQryUserInvestor方法 61](#_Toc111121647)

[6.3.24 ReqQryInvestorAccount方法 61](#_Toc111121648)

[6.3.25 ReqQryInstrument方法 62](#_Toc111121649)

[6.3.26 ReqQryMarketData方法 62](#_Toc111121650)

[6.3.27 ReqQryExchange方法 63](#_Toc111121651)

[6.3.28 ReqQryInvestorPosition方法 63](#_Toc111121652)

[6.3.29 ReqSubscribeTopic方法 64](#_Toc111121653)

[6.3.30 ReqQryTopic方法 64](#_Toc111121654)

[6.3.31 ReqQryInvestorFee方法 65](#_Toc111121655)

[6.3.32 ReqQryInvestorMargin方法 65](#_Toc111121656)

[6.3.33 ReqQrySGEDeferRate方法 66](#_Toc111121657)

[7. 开发实例 66](#_Toc111121658)

# 介绍

拓界科技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，包括报单录入、报单撤销、报单查询、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7个文件：

|  |  |  |  |
| --- | --- | --- | --- |
| 文件名 | 版本 | 文件大小 | 文件描述 |
| QdpFtdcTraderApi.h |  |  | 交易接口头文件 |
| QdpFtdcUserApiDataType.h |  |  | 定义了 API 所需的一系列数据类型的头文件 |
| QdpFtdcUserApiStruct.h |  |  | 定义了一系列业务相关的数据结构的头文件 |
| qdptraderapi.dll |  |  | WINDOWS版交易API动态链接库二进制文件 |
| qdptraderapi.lib |  |  | WINDOWS版交易API导入库文件 |
| qdptraderapi.so |  |  | LINUX版交易API动态链接库 |

Windows版本支持MS VC 6.0，MS VC.NET 2003编译器,LINUX版本采用64位编译，还需要打开多线程编译选项/MT。

# 体系结构

交易员API使用建立在TCP协议之上FTD协议与QDP服务进行通讯， QDP负责投资者的交易业务处理。

## 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用户终端主动发起的通讯请求。该请求被QDP服务端接收和处理，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QDP服务端主动，向某个特定的用户发出的信息。例如成交回报等。

广播通讯模式是指QDP服务端主动将公告、市场公共信息等信息发送给所有注册用户。

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-各种通讯模式的工作过程

本接口暂时没有使用广播通讯方式。

## 数据流

QDP支持对话通讯模式、私有通讯模式：

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，交易员使用用户终端发送交易请求，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，交易员使用用户终端发送查询请求，QDP后台交易系统反馈应答。QDP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，由交易员通过用户终端发向QDP交易系统，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，QDP交易系统维护每个用户终端的私有流，在一个交易日内，用户终端断线后恢复连接时，可以请求QDP后台交易系统发送指定序号之后的私有流数据。私有数据流向用户终端提供报单状态报告、成交回报等信息。

# 接口模式

交易员API 提供了二种编程接口，分别为CQdpFtdcTraderApi 和CQdpFtdcTraderSpi。这两种接口对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CQdpFtdcTraderApi 发出操作请求，通过继承CQdpFtdcTraderSpi并重载回调函数来处理后台服务的响应。

## 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 CQdpFtdcTraderApi::ReqXXX(

CQdpFtdcXXXField \*pReqXXX,

int nRequestID)

响应：void CQdpFtdcTraderSpi::OnRspXXX(

CQdpFtdcXXXField \*pRspXXX,

CQdpFtdcRspInfoField \*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QDP后台系统的响应时，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，CQdpFtdcTraderSpi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。

## 私有流编程接口

私有流中的数据中用户的私有信息，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 CQdpFtdcTraderSpi::OnRtnXXX(CQdpFtdcXXXField \*pXXX) 或

void CQdpFtdcTraderSpi::OnErrRtnXXX(CQdpFtdcXXXField \*pXXX,

CQdpFtdcRspInfoField \*pRspInfo)

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

# 运行模式

## 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。

CQdpFtdcTraderApi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。

CQdpFtdcTraderSpi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QDP后台收取所需数据。

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CQdpFtdcTraderSpi 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。

## 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”。

# 业务和接口对照

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 业务类型 | 业务 | 请求接口 | 响应接口 | 数据流 |
| 登录 | 登录 | CQdpFtdcTraderApi:: ReqUserLogin | CQdpFtdcTraderSpi::OnRspUserLogin | 对话流 |
| 登出 | CQdpFtdcTraderApi::ReqUserLogout | CQdpFtdcTraderSpi::OnRspUserLogout | 对话流 |
| 修改用户口令 | CQdpFtdcTraderApi::ReqUserPasswordUpdate | CQdpFtdcTraderSpi::OnRspUserPasswordUpdate | 对话流 |
| 报单 | 报单录入 | CQdpFtdcTraderApi::ReqOrderInsert | CQdpFtdcTraderSpi::OnRspOrderInsert | 对话流 |
| 报单操作 | CQdpFtdcTraderApi::ReqOrderAction | CQdpFtdcTraderSpi::OnRspOrderAction | 对话流 |
| 私有回报 | 成交回报 | N/A | CQdpFtdcTraderSpi::OnRtnTrade | 私有流 |
| 报单回报 | N/A | CQdpFtdcTraderSpi::OnRtnOrder | 私有流 |
| 出入金回报 | N/A | CQdpFtdcTraderSpi::OnRtnInvestorAccountDeposit | 私有流 |
| 报单录入错误回报 | N/A | CQdpFtdcTraderSpi::OnErrRtnOrderInsert | 私有流 |
| 报单操作错误回报 | N/A | CQdpFtdcTraderSpi::OnErrRtnOrderAction | 私有流 |
| 警告消息通知 | N/A | CQdpFtdcTraderSpi::OnRtnMessageNotify | 私有流 |
| 金交所递延费率通知 | N/A | CQdpFtdcTraderSpi::OnRtnSGEDeferRate | 私有流 |
| 查询 | 报单查询 | CQdpFtdcTraderApi::ReqQryOrder | CQdpFtdcTraderSpi::OnRspQryOrder | 查询流 |
| 成交查询 | CQdpFtdcTraderApi::ReqQryTrade | CQdpFtdcTraderSpi::OnRspQryTrade | 查询流 |
| 合约查询 | CQdpFtdcTraderApi::ReqQryInstrument | CQdpFtdcTraderSpi::OnRspQryInstrument | 查询流 |
| 可用投资者查询 | CQdpFtdcTraderApi::ReqQryUserInvestor | CQdpFtdcTraderSpi::OnRspQryUserInvestor | 查询流 |
| 资金账户查询 | CQdpFtdcTraderApi::ReqQryInvestorAccount | CQdpFtdcTraderSpi::OnRspQryInvestorAccount | 查询流 |
| 交易所查询 | CQdpFtdcTraderApi::ReqQryExchange | CQdpFtdcTraderSpi::OnRspQryExchange | 查询流 |
| 合约查询 | CQdpFtdcTraderApi::ReqQryInstrument | CQdpFtdcTraderSpi::OnRspQryInstrument | 查询流 |
| 行情查询 | CQdpFtdcTraderApi::ReqQryInstrument | CQdpFtdcTraderSpi::OnRspQryMarketData | 查询流 |
| 投资者持仓查询 | CQdpFtdcTraderApi::ReqQryInvestorPosition | CQdpFtdcTraderSpi::OnRspQryInvestorPosition | 查询流 |
| 手续费率查询 | CQdpFtdcTraderApi::ReqQryInvestorFee | CQdpFtdcTraderSpi::OnRspQryInvestorFee | 查询流 |
| 保证金率查询 | CQdpFtdcTraderApi::ReqQryInvestorMargin | CQdpFtdcTraderSpi::OnRspQryInvestorMargin | 查询流 |
| 金交所递延费率查询 | CQdpFtdcTraderApi::ReqQrySGEDeferRate | CQdpFtdcTraderSpi::OnRspQrySGEDeferRate | 查询流 |
| 期权手续费率查询 | CQdpFtdcTraderApi::ReqQryInvestorOptionFee | CQdpFtdcTraderSpi::OnRspQryInvestorOptionFee | 查询流 |
| 实时行情查询 | CQdpFtdcTraderApi::ReqQryMarketData | CQdpFtdcTraderSpi::OnRspQryMarketData | 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投资者下其他的用户接收到。

银期接口是当QDP作为次席交易系统时，实现从用户终端经QDP系统向主席系统发起银期转账、银行余额查询等银期功能。

# 开发接口

## 通用规则

客户端和QDP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CQdpFtdcTraderApi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QDP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应答方法。

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。

## CQdpFtdcTraderSpi接口

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

### OnFrontConnected 方法

当客户端与QDP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。

### OnFrontDisconnected 方法

当客户端与QDP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

### 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

### OnPackageStart方法

报文回调开始通知。当API收到一个报文后，首先调用本方法，然后是各数据域的回调，最后是报文回调结束通知。

**函数原形：**

void OnPackageStart(int nTopicID, int nSequenceNo)；

**参数：**

param nTopicID 主题代码（如私有流、公共流、行情流等）

param nSequenceNo 报文序号

### OnPackageEnd方法

报文回调结束通知。当API收到一个报文后，首先调用报文回调开始通知，然后是各数据域的回调，最后调用本方法。

**函数原形：**

void OnPackageStart(int nTopicID, int nSequenceNo)；

**参数：**

param nTopicID 主题代码（如私有流、公共流、行情流等）

param nSequenceNo 报文序号

### OnRspError 方法

针对用户请求的出错通知。

**函数原形：**

void OnRspError(

CQdpFtdcRspInfoField \* 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 CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### OnRspUserLogin方法

当客户端发出登录请求之后，QDP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(

CQdpFtdcRspUserLoginField \*pRspUserLogin,

CQdpFtdcRspInfoField \*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 CQdpFtdcRspUserLoginField

{

///交易日

TQdpFtdcDateType TradingDay;

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易用户代码

TQdpFtdcUserIDType UserID;

///登录成功时间

TQdpFtdcTimeType LoginTime;

///用户最大本地报单号

TQdpFtdcUserOrderLocalIDType MaxOrderLocalID;

///交易系统名称

TQdpFtdcTradingSystemNameType TradingSystemName;

///数据中心代码

TQdpFtdcDataCenterIDType DataCenterID;

///会员私有流当前长度

TQdpFtdcSequenceNoType PrivateFlowSize;

///交易员私有流当前长度

TQdpFtdcSequenceNoType UserFlowSize;

};

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 CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### OnRspUserLogout 方法

当客户端发出退出请求之后，QDP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(

CQdpFtdcRspUserLogoutField \*pRspUserLogout,

CQdpFtdcRspInfoField \*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 CQdpFtdcRspUserLogoutField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易用户代码

TQdpFtdcUserIDType UserID;

};

pRspInfo：返回用户响应信息的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### 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，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspUserPasswordUpdate(

CQdpFtdcUserPasswordUpdateField \*pUserPasswordUpdate,

CQdpFtdcRspInfoField \*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 CQdpFtdcUserPasswordUpdateField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易用户代码

TQdpFtdcUserIDType UserID;

///旧密码

TQdpFtdcPasswordType OldPassword;

///新密码

TQdpFtdcPasswordType NewPassword;

};

pRspInfo：返回用户响应信息的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### OnRspOrderInsert 方法

报单录入应答。当客户端发出过报单录入指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspOrderInsert(

CQdpFtdcInputOrderField \*pInputOrder,

CQdpFtdcRspInfoField \*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 CQdpFtdcInputOrderField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///系统报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///用户代码

TQdpFtdcUserIDType UserID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///用户本地报单号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///报单类型

TQdpFtdcOrderPriceTypeType OrderPriceType;

///买卖方向

TQdpFtdcDirectionType Direction;

///开平标志

TQdpFtdcOffsetFlagType OffsetFlag;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///价格

TQdpFtdcPriceType LimitPrice;

///数量

TQdpFtdcVolumeType Volume;

///有效期类型

TQdpFtdcTimeConditionType TimeCondition;

///GTD日期

TQdpFtdcDateType GTDDate;

///成交量类型

TQdpFtdcVolumeConditionType VolumeCondition;

///最小成交量

TQdpFtdcVolumeType MinVolume;

///止损价

TQdpFtdcPriceType StopPrice;

///强平原因

TQdpFtdcForceCloseReasonType ForceCloseReason;

///自动挂起标志

TQdpFtdcBoolType IsAutoSuspend;

///业务单元

TQdpFtdcBusinessUnitType BusinessUnit;

///用户自定义域

TQdpFtdcCustomType UserCustom;

///营业部代码

TQdpFtdcBranchIDType BranchID;

///记录编号

TQdpFtdcSequenceNoType RecNum;

///业务类别

TQdpFtdcBusinessTypeType BusinessType;

};

pRspInfo：返回用户响应信息的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### 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（暂不支持）、报单的激活（暂不支持）、报单的修改（暂不支持）。当客户端发出过报单操作指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspOrderAction(

CQdpFtdcOrderActionField \* pOrderAction,

CQdpFtdcRspInfoField \* 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 CQdpFtdcOrderActionField

{

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///用户代码

TQdpFtdcUserIDType UserID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///本次撤单操作的本地编号

TQdpFtdcUserOrderLocalIDType UserOrderActionLocalID;

///被撤订单的本地报单编号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///报单操作标志

TQdpFtdcActionFlagType ActionFlag;

///价格

TQdpFtdcPriceType LimitPrice;

///数量变化

TQdpFtdcVolumeType VolumeChange;

///记录编号

TQdpFtdcSequenceNoType RecNum;

};

pRspInfo：返回用户响应信息的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### OnRtnFlowMessageCancel方法

数据流回退通知。

**函数原形：**

void OnRtnFlowMessageCancel(

CQdpFtdcFlowMessageCancelField \*pFlowMessageCancel)；

**参数：**

pFlowMessageCancel：数据流回退。

struct CQdpFtdcFlowMessageCancelField

{

///序列系列号

TQdpFtdcSequenceSeriesType SequenceSeries;

///交易日

TQdpFtdcDateType TradingDay;

///数据中心代码

TQdpFtdcDataCenterIDType DataCenterID;

///回退起始序列号

TQdpFtdcSequenceNoType StartSequenceNo;

///回退结束序列号

TQdpFtdcSequenceNoType EndSequenceNo;

};

### OnRtnTrade 方法

成交回报。当发生成交时QDP会通知客户端，该方法会被调用。

**函数原形：**

void OnRtnTrade(

CQdpFtdcTradeField \*pTrade)；

**参数：**

pTrade：指向成交信息结构的地址。

成交信息结构：

struct CQdpFtdcTradeField

{

///交易日

TQdpFtdcTradingDayType TradingDay;

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///会员编号

TQdpFtdcParticipantIDType ParticipantID;

///下单席位号

TQdpFtdcSeatIDType SeatID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///客户号

TQdpFtdcClientIDType ClientID;

///用户编号

TQdpFtdcUserIDType UserID;

///成交编号

TQdpFtdcTradeIDType TradeID;

///报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///本地报单编号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///买卖方向

TQdpFtdcDirectionType Direction;

///开平标志

TQdpFtdcOffsetFlagType OffsetFlag;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///成交价格

TQdpFtdcPriceType TradePrice;

///成交数量

TQdpFtdcVolumeType TradeVolume;

///成交时间

TQdpFtdcTimeType TradeTime;

///清算会员编号

TQdpFtdcParticipantIDType ClearingPartID;

///成交金额

TQdpFtdcMoneyType TradeAmnt;

///记录编号

TQdpFtdcSequenceNoType RecNum;

};

### 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，QDP会主动通知客户端，该方法会被调用。

**函数原形：**

void OnRtnOrder(

CQdpFtdcOrderField \*pOrder);

**参数：**

pOrder：指向报单信息结构的地址。

报单信息结构：

struct CQdpFtdcOrderField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///系统报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///用户代码

TQdpFtdcUserIDType UserID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///用户本地报单号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///报单类型

TQdpFtdcOrderPriceTypeType OrderPriceType;

///买卖方向

TQdpFtdcDirectionType Direction;

///开平标志

TQdpFtdcOffsetFlagType OffsetFlag;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///价格

TQdpFtdcPriceType LimitPrice;

///数量

TQdpFtdcVolumeType Volume;

///有效期类型

TQdpFtdcTimeConditionType TimeCondition;

///GTD日期 保留字段

TQdpFtdcDateType GTDDate;

///成交量类型

TQdpFtdcVolumeConditionType VolumeCondition;

///最小成交量

TQdpFtdcVolumeType MinVolume;

///止损价 保留字段

TQdpFtdcPriceType StopPrice;

///强平原因 保留字段

TQdpFtdcForceCloseReasonType ForceCloseReason;

///自动挂起标志 保留字段

TQdpFtdcBoolType IsAutoSuspend;

///业务单元 保留字段

TQdpFtdcBusinessUnitType BusinessUnit;

///用户自定义域

TQdpFtdcCustomType UserCustom;

///营业部代码

TQdpFtdcBranchIDType BranchID;

///记录编号

TQdpFtdcSequenceNoType RecNum;

///业务类别

TQdpFtdcBusinessTypeType BusinessType;

///交易日

TQdpFtdcTradingDayType TradingDay;

///会员编号

TQdpFtdcParticipantIDType ParticipantID;

///客户号

TQdpFtdcClientIDType ClientID;

///下单席位号

TQdpFtdcSeatIDType SeatID;

///插入时间

TQdpFtdcTimeType InsertTime;

///本地报单编号

TQdpFtdcOrderLocalIDType OrderLocalID;

///报单来源

TQdpFtdcOrderSourceType OrderSource;

///报单状态

TQdpFtdcOrderStatusType OrderStatus;

///撤销时间

TQdpFtdcTimeType CancelTime;

///撤单用户编号

TQdpFtdcUserIDType CancelUserID;

///今成交数量

TQdpFtdcVolumeType VolumeTraded;

///剩余数量

TQdpFtdcVolumeType VolumeRemain;

};

### OnErrRtnOrderInsert方法

报单录入错误回报。由QDP主动通知客户端，该方法会被调用。

**函数原形：**

void OnErrRtnOrderInsert(

CQdpFtdcInputOrderField \*pInputOrder,

CQdpFtdcRspInfoField \* 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 CQdpFtdcInputOrderField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///系统报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///用户代码

TQdpFtdcUserIDType UserID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///用户本地报单号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///报单类型

TQdpFtdcOrderPriceTypeType OrderPriceType;

///买卖方向

TQdpFtdcDirectionType Direction;

///开平标志

TQdpFtdcOffsetFlagType OffsetFlag;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///价格

TQdpFtdcPriceType LimitPrice;

///数量

TQdpFtdcVolumeType Volume;

///有效期类型

TQdpFtdcTimeConditionType TimeCondition;

///GTD日期

TQdpFtdcDateType GTDDate;

///成交量类型

TQdpFtdcVolumeConditionType VolumeCondition;

///最小成交量

TQdpFtdcVolumeType MinVolume;

///止损价

TQdpFtdcPriceType StopPrice;

///强平原因

TQdpFtdcForceCloseReasonType ForceCloseReason;

///自动挂起标志

TQdpFtdcBoolType IsAutoSuspend;

///业务单元

TQdpFtdcBusinessUnitType BusinessUnit;

///用户自定义域

TQdpFtdcCustomType UserCustom;

///营业部代码

TQdpFtdcBranchIDType BranchID;

///记录编号

TQdpFtdcSequenceNoType RecNum;

///业务类别

TQdpFtdcBusinessTypeType BusinessType;

};

pRspInfo：返回用户响应信息的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

### OnErrRtnOrderAction方法

报价操作错误回报。由QDP主动通知客户端，该方法会被调用。

**函数原形：**

void OnErrRtnOrderAction (

CQdpFtdcOrderActionField \*pOrderAction,

CQdpFtdcRspInfoField \*pRspInfo);

**参数：**

pOrderAction：指向报单操作结构的地址，包含了报单操作请求的输入数据。

报单操作结构：

struct CQdpFtdcOrderActionField

{

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///用户代码

TQdpFtdcUserIDType UserID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///本次撤单操作的本地编号

TQdpFtdcUserOrderLocalIDType UserOrderActionLocalID;

///被撤订单的本地报单编号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///报单操作标志

TQdpFtdcActionFlagType ActionFlag;

///价格

TQdpFtdcPriceType LimitPrice;

///数量变化

TQdpFtdcVolumeType VolumeChange;

///记录编号

TQdpFtdcSequenceNoType RecNum;

};

pRspInfo：返回用户响应信息的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

### OnRtnInstrumentStatus方法

合约交易状态通知。由QDP主动通知客户端，该方法会被调用。

**函数原形：**

void OnRtnInstrumentStatus(

CQdpFtdcInstrumentStatusField \*pInstrumentStatus) ;

**参数：**

pInstrumentStatus：指向合约交易状态的地址，包含了后台返回的合约状态数据。

参数结构：

struct CQdpFtdcInstrumentStatusField

{

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///品种代码

TQdpFtdcProductIDType ProductID;

///品种名称

TQdpFtdcProductNameType ProductName;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///合约名称

TQdpFtdcInstrumentNameType InstrumentName;

///交割年份

TQdpFtdcDeliveryYearType DeliveryYear;

///交割月

TQdpFtdcDeliveryMonthType DeliveryMonth;

///限价单最大下单量

TQdpFtdcMaxLimitOrderVolumeType MaxLimitOrderVolume;

///限价单最小下单量

TQdpFtdcMinLimitOrderVolumeType MinLimitOrderVolume;

///市价单最大下单量

TQdpFtdcMaxMarketOrderVolumeType MaxMarketOrderVolume;

///市价单最小下单量

TQdpFtdcMinMarketOrderVolumeType MinMarketOrderVolume;

///数量乘数

TQdpFtdcVolumeMultipleType VolumeMultiple;

///报价单位

TQdpFtdcPriceTickType PriceTick;

///币种

TQdpFtdcCurrencyType Currency;

///多头限仓

TQdpFtdcLongPosLimitType LongPosLimit;

///空头限仓

TQdpFtdcShortPosLimitType ShortPosLimit;

///跌停板价

TQdpFtdcLowerLimitPriceType LowerLimitPrice;

///涨停板价

TQdpFtdcUpperLimitPriceType UpperLimitPrice;

///昨结算

TQdpFtdcPreSettlementPriceType PreSettlementPrice;

///合约交易状态

TQdpFtdcInstrumentStatusType InstrumentStatus;

///创建日

TQdpFtdcDateType CreateDate;

///上市日

TQdpFtdcDateType OpenDate;

///到期日

TQdpFtdcDateType ExpireDate;

///开始交割日

TQdpFtdcDateType StartDelivDate;

///最后交割日

TQdpFtdcDateType EndDelivDate;

///挂牌基准价

TQdpFtdcPriceType BasisPrice;

///当前是否交易

TQdpFtdcBoolType IsTrading;

///基础商品代码

TQdpFtdcInstrumentIDType UnderlyingInstrID;

///基础商品乘数

TQdpFtdcUnderlyingMultipleType UnderlyingMultiple;

///持仓类型

TQdpFtdcPositionTypeType PositionType;

///执行价

TQdpFtdcPriceType StrikePrice;

///期权类型

TQdpFtdcOptionsTypeType OptionsType;

};

### OnRtnInvestorAccountDeposit 方法

出入金结果回报。由QDP主动通知客户端，该方法会被调用。

**函数原形：**

void OnRtnInvestorAccountDeposit(

CQdpFtdcInvestorAccountDepositResField\* pInvestorAccountDepositRes)

**参数：**

pInvestorAccountDepositRes：指向出入金结果回报结构的地址。

出入金结果回报结构：

struct CQdpFtdcInvestorAccountDepositResField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///资金帐号

TQdpFtdcAccountIDType AccountID;

///资金流水号

TQdpFtdcAccountSeqNoType AccountSeqNo;

///连续递增的总资金流水号

TQdpFtdcAccountSeqNoType AccountInsideSeqNo;

///金额

TQdpFtdcMoneyType Amount;

///出入金方向

TQdpFtdcAccountDirectionType AmountDirection;

///可用资金

TQdpFtdcMoneyType Available;

///结算准备金

TQdpFtdcMoneyType Balance;

///用户代码

TQdpFtdcUserIDType UserID;

}

### OnRspQryOrder 方法

报单查询请求。当客户端发出报单查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryOrder(

CQdpFtdcOrderField \*pOrder,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)；

**参数：**

pOrder：指向报单查询的返回结构信息的地址。

报单信息结构：

struct CQdpFtdcOrderField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///系统报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///用户代码

TQdpFtdcUserIDType UserID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///用户本地报单号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///报单类型

TQdpFtdcOrderPriceTypeType OrderPriceType;

///买卖方向

TQdpFtdcDirectionType Direction;

///开平标志

TQdpFtdcOffsetFlagType OffsetFlag;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///价格

TQdpFtdcPriceType LimitPrice;

///数量

TQdpFtdcVolumeType Volume;

///有效期类型

TQdpFtdcTimeConditionType TimeCondition;

///GTD日期 保留字段

TQdpFtdcDateType GTDDate;

///成交量类型

TQdpFtdcVolumeConditionType VolumeCondition;

///最小成交量

TQdpFtdcVolumeType MinVolume;

///止损价 保留字段

TQdpFtdcPriceType StopPrice;

///强平原因 保留字段

TQdpFtdcForceCloseReasonType ForceCloseReason;

///自动挂起标志 保留字段

TQdpFtdcBoolType IsAutoSuspend;

///业务单元 保留字段

TQdpFtdcBusinessUnitType BusinessUnit;

///用户自定义域

TQdpFtdcCustomType UserCustom;

///营业部代码

TQdpFtdcBranchIDType BranchID;

///记录编号

TQdpFtdcSequenceNoType RecNum;

///业务类别

TQdpFtdcBusinessTypeType BusinessType;

///交易日

TQdpFtdcTradingDayType TradingDay;

///会员编号

TQdpFtdcParticipantIDType ParticipantID;

///客户号

TQdpFtdcClientIDType ClientID;

///下单席位号

TQdpFtdcSeatIDType SeatID;

///插入时间

TQdpFtdcTimeType InsertTime;

///本地报单编号

TQdpFtdcOrderLocalIDType OrderLocalID;

///报单来源

TQdpFtdcOrderSourceType OrderSource;

///报单状态

TQdpFtdcOrderStatusType OrderStatus;

///撤销时间

TQdpFtdcTimeType CancelTime;

///撤单用户编号

TQdpFtdcUserIDType CancelUserID;

///今成交数量

TQdpFtdcVolumeType VolumeTraded;

///剩余数量

TQdpFtdcVolumeType VolumeRemain;

};

pRspInfo：返回用户响应信息的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### OnRspQryTrade 方法

成交单查询应答。当客户端发出成交单查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryTrade(

CQdpFtdcTradeField \* pTrade,

CQdpFtdcRspInfoField \*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 CQdpFtdcTradeField

{

///交易日

TQdpFtdcTradingDayType TradingDay;

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///会员编号

TQdpFtdcParticipantIDType ParticipantID;

///下单席位号

TQdpFtdcSeatIDType SeatID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///客户号

TQdpFtdcClientIDType ClientID;

///用户编号

TQdpFtdcUserIDType UserID;

///成交编号

TQdpFtdcTradeIDType TradeID;

///报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///本地报单编号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///买卖方向

TQdpFtdcDirectionType Direction;

///开平标志

TQdpFtdcOffsetFlagType OffsetFlag;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///成交价格

TQdpFtdcPriceType TradePrice;

///成交数量

TQdpFtdcVolumeType TradeVolume;

///成交时间

TQdpFtdcTimeType TradeTime;

///清算会员编号

TQdpFtdcParticipantIDType ClearingPartID;

///成交金额

TQdpFtdcMoneyType TradeAmnt;

///记录编号

TQdpFtdcSequenceNoType RecNum;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### OnRspQryUserInvestor方法

可用投资者账户查询应答。当客户端发出可用投资者账户查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryUserInvestor(

CQdpFtdcRspUserInvestorField \*pUserInvestor,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pUserInvestor指向用户可用投资者账户结构的地址。

投资者账户结构：

struct CQdpFtdcUserInvestorField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易用户代码

TQdpFtdcUserIDType UserID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回可用投资者账户请求的ID，该ID 由用户在可用投资者账户查询时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### OnRspQryInvestorAccount方法

投资者资金账户查询。当客户端发出投资者资金账户查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryInvestorAccount(

CQdpFtdcRspInvestorAccountField \*pRspInvestorAccount,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pRspInvestorAccount指向投资者资金账户信息的地址。

投资者账户结构

struct CQdpFtdcRspInvestorAccountField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///资金帐号

TQdpFtdcAccountIDType AccountID;

///上次结算准备金

TQdpFtdcMoneyType PreBalance;

///上日可用资金

TQdpFtdcMoneyType PreAvailable;

///入金金额

TQdpFtdcMoneyType Deposit;

///出金金额

TQdpFtdcMoneyType Withdraw;

///占用保证金

TQdpFtdcMoneyType Margin;

///期权权利金收支

TQdpFtdcMoneyType Premium;

///手续费

TQdpFtdcMoneyType Fee;

///冻结的保证金

TQdpFtdcMoneyType FrozenMargin;

///冻结权利金

TQdpFtdcMoneyType FrozenPremium;

///冻结手续费

TQdpFtdcMoneyType FrozenFee;

///平仓盈亏

TQdpFtdcMoneyType CloseProfit;

///持仓盈亏

TQdpFtdcMoneyType PositionProfit;

///可用资金

TQdpFtdcMoneyType Available;

///结算准备金

TQdpFtdcMoneyType Balance;

///多头占用保证金

TQdpFtdcMoneyType LongMargin;

///空头占用保证金

TQdpFtdcMoneyType ShortMargin;

///多头冻结的保证金

TQdpFtdcMoneyType LongFrozenMargin;

///空头冻结的保证金

TQdpFtdcMoneyType ShortFrozenMargin;

///动态权益

TQdpFtdcMoneyType DynamicRights;

///风险度

TQdpFtdcMoneyType Risk;

///其他费用

TQdpFtdcMoneyType OtherFee;

///质押金额

TQdpFtdcMoneyType Mortgage;

///币种

TQdpFtdcCurrencyIDType Currency;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回投资者资金账户查询请求的ID，该ID 由用户在投资者资金账户查询时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### OnRspQryInstrument方法

合约信息查询应答。当客户端发出合约信息查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryQdpInstrument(

CQdpFtdcRspInstrumentField \*pRspInstrument,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pRspInstrument指向合约信息结构的地址。

合约信息结构：

struct CQdpFtdcRspInstrumentField

{

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///品种代码

TQdpFtdcProductIDType ProductID;

///品种名称

TQdpFtdcProductNameType ProductName;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///合约名称

TQdpFtdcInstrumentNameType InstrumentName;

///交割年份

TQdpFtdcDeliveryYearType DeliveryYear;

///交割月

TQdpFtdcDeliveryMonthType DeliveryMonth;

///限价单最大下单量

TQdpFtdcMaxLimitOrderVolumeType MaxLimitOrderVolume;

///限价单最小下单量

TQdpFtdcMinLimitOrderVolumeType MinLimitOrderVolume;

///市价单最大下单量

TQdpFtdcMaxMarketOrderVolumeType MaxMarketOrderVolume;

///市价单最小下单量

TQdpFtdcMinMarketOrderVolumeType MinMarketOrderVolume;

///数量乘数

TQdpFtdcVolumeMultipleType VolumeMultiple;

///报价单位

TQdpFtdcPriceTickType PriceTick;

///币种

TQdpFtdcCurrencyType Currency;

///多头限仓

TQdpFtdcLongPosLimitType LongPosLimit;

///空头限仓

TQdpFtdcShortPosLimitType ShortPosLimit;

///跌停板价

TQdpFtdcLowerLimitPriceType LowerLimitPrice;

///涨停板价

TQdpFtdcUpperLimitPriceType UpperLimitPrice;

///昨结算

TQdpFtdcPreSettlementPriceType PreSettlementPrice;

///合约交易状态

TQdpFtdcInstrumentStatusType InstrumentStatus;

///创建日

TQdpFtdcDateType CreateDate;

///上市日

TQdpFtdcDateType OpenDate;

///到期日

TQdpFtdcDateType ExpireDate;

///开始交割日

TQdpFtdcDateType StartDelivDate;

///最后交割日

TQdpFtdcDateType EndDelivDate;

///挂牌基准价

TQdpFtdcPriceType BasisPrice;

///当前是否交易

TQdpFtdcBoolType IsTrading;

///基础商品代码

TQdpFtdcInstrumentIDType UnderlyingInstrID;

///基础商品乘数

TQdpFtdcUnderlyingMultipleType UnderlyingMultiple;

///持仓类型

TQdpFtdcPositionTypeType PositionType;

///执行价

TQdpFtdcPriceType StrikePrice;

///期权类型

TQdpFtdcOptionsTypeType OptionsType;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType 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。

### OnRspQryMarketData方法

行情查询。当客户端发出合约行情查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryMarketData (

CQdpFtdcMarketDataField \*pMarketData,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pMarketData指向合约行情结构的地址。

合约行情结构

struct CQdpFtdcMarketDataField

{

///交易日

TQdpFtdcTradingDayType TradingDay;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///品种代码

TQdpFtdcProductIDType ProductID;

///今开盘

TQdpFtdcPriceType OpenPrice;

///申卖价一

TQdpFtdcPriceType AskPrice1;

///申卖量一

TQdpFtdcVolumeType AskVolume1;

///申买价一

TQdpFtdcPriceType BidPrice1;

///申买量一

TQdpFtdcVolumeType Bidvolume1;

///最高价

TQdpFtdcPriceType HighestPrice;

///最低价

TQdpFtdcPriceType LowestPrice;

///最新价

TQdpFtdcPriceType LastPrice;

///数量

TQdpFtdcVolumeType Volume;

///成交金额

TQdpFtdcMoneyType TurnOver;

///跌停板价

TQdpFtdcPriceType LowerLimitPrice;

///涨停板价

TQdpFtdcPriceType UpperLimitPrice;

///持仓量

TQdpFtdcVolumeType OpenInterest;

///昨收盘

TQdpFtdcPriceType PreClosePrice;

///昨持仓量

TQdpFtdcVolumeType PreOpenInterest;

///昨结算

TQdpFtdcPriceType PreSettlementPrice;

///今结算

TQdpFtdcPriceType SettlementPrice;

///最后修改毫秒

TQdpFtdcMillisecType Millisec;

///最后修改时间

TQdpFtdcTimeType UpdateTime;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

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

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### OnRspQryExchange方法

交易所查询。当客户端发出交易所查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryExchange(

CQdpFtdcRspExchangeField \*pExchange,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pExchange指向交易所结构的地址。

交易编码结构

struct CQdpFtdcRspExchangeField

{

///交易日

TQdpFtdcTradingDayType TradingDay;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///交易所名称

TQdpFtdcExchangeNameType ExchangeName;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回交易所查询请求的ID，该ID 由用户在交易所查询时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### 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，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryInvestorPosition (

CQdpFtdcRspInvestorPositionField \*pInvestorPosition,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pInvestorPosition指向投资者持仓结构的地址。

投资者持仓结构：

struct CQdpFtdcRspInvestorPositionField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///会员编号

TQdpFtdcParticipantIDType ParticipantID;

///客户代码

TQdpFtdcClientIDType ClientID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///买卖方向

TQdpFtdcDirectionType Direction;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///占用保证金

TQdpFtdcMoneyType UsedMargin;

///今持仓量

TQdpFtdcVolumeType Position;

///今日持仓成本

TQdpFtdcPriceType PositionCost;

///昨持仓量

TQdpFtdcVolumeType YdPosition;

///昨日持仓成本

TQdpFtdcMoneyType YdPositionCost;

///冻结的保证金

TQdpFtdcMoneyType FrozenMargin;

///开仓冻结持仓

TQdpFtdcVolumeType FrozenPosition;

///平仓冻结持仓

TQdpFtdcVolumeType FrozenClosing;

///冻结的权利金

TQdpFtdcMoneyType FrozenPremium;

///最后一笔成交编号

TQdpFtdcTradeIDType LastTradeID;

///最后一笔本地报单编号

TQdpFtdcOrderLocalIDType LastOrderLocalID;

///币种

TQdpFtdcCurrencyIDType Currency;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回投资者持仓查询请求的ID，该ID 由用户在投资者持仓查询查询时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### OnRspSubscribeTopic方法

订阅主题应答。

**函数原形：**

void OnRspSubscribeTopic (

CQdpFtdcDisseminationField \*pDissemination,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pDissemination指向主题结构的地址。

struct CQdpFtdcDisseminationField

{

///序列系列号

TQdpFtdcSequenceSeriesType SequenceSeries;

///序列号

TQdpFtdcSequenceNoType SequenceNo;

};

CQdpFtdcRspInfoField: 响应信息

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回用户订阅主题请求的ID，该ID 由用户在订阅主题时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### OnRspQryTopic方法

主题查询应答。

**函数原形：**

void OnRspQryTopic(

CQdpFtdcDisseminationField \*pDissemination,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast);

**参数：**

pDissemination指向主题结构的地址。

struct CQdpFtdcDisseminationField

{

///序列系列号

TQdpFtdcSequenceSeriesType SequenceSeries;

///序列号

TQdpFtdcSequenceNoType SequenceNo;

};

CQdpFtdcRspInfoField: 响应信息

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回用户主题查询请求的ID，该ID 由用户在主题查询时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### OnRspQryInvestorFee方法

投资者手续费率查询应答。当客户端发出投资者手续费率查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryInvestorFee(

CQdpFtdcInvestorFeeField \*pInvestorFee,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pInvestorFee指向投资者手续费率结构的地址。

投资者手续费率结构：

struct CQdpFtdcInvestorFeeField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///投资者编码

TQdpFtdcInvestorIDType InvestorID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///开仓手续费按比例

TQdpFtdcRatioType OpenFeeRate;

///开仓手续费按手数

TQdpFtdcRatioType OpenFeeAmt;

///平仓手续费按比例

TQdpFtdcRatioType OffsetFeeRate;

///平仓手续费按手数

TQdpFtdcRatioType OffsetFeeAmt;

///平今仓手续费按比例

TQdpFtdcRatioType OTFeeRate;

///平今仓手续费按手数

TQdpFtdcRatioType OTFeeAmt;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回投资者手续费请求的ID，该ID 由用户在投资者手续费查询时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### OnRspQryInvestorMargin方法

投资者保证金率查询应答。当客户端发出投资者保证金率查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQryInvestorMargin (

CQdpFtdcInvestorMarginField \*pInvestorMargin,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pInvestorMargin指向投资者保证金率结构的地址。

投资者保证金率结构：

struct CQdpFtdcInvestorMarginField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///投资者编码

TQdpFtdcInvestorIDType InvestorID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///多头占用保证金按比例

TQdpFtdcRatioType LongMarginRate;

///多头保证金按手数

TQdpFtdcRatioType LongMarginAmt;

///空头占用保证金按比例

TQdpFtdcRatioType ShortMarginRate;

///空头保证金按手数

TQdpFtdcRatioType ShortMarginAmt;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回投资者保证金率查询请求的ID，该ID 由用户在投资者保证金率查询时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

### OnRspQrySGEDeferRate方法

金交所递延费率查询应答。当客户端发出金交所递延费率查询指令后，QDP返回响应时，该方法会被调用。

**函数原形：**

void OnRspQrySGEDeferRate (

CQdpFtdcSGEDeferRateField \*pSGEDeferRate,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

**参数：**

pSGEDeferRate指向金交所递延费率查询请求结构的地址。

投资者保证金率结构：

struct CQdpFtdcSGEDeferRateField

{

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///交易日期

TQdpFtdcTradingDayType TradeDate;

///支付方向

TQdpFtdcPayDirectionType Direction;

///递延费率

TQdpFtdcRatioType DeferRate;

};

pRspInfo：指向响应信息结构的地址。

响应信息结构：

struct CQdpFtdcRspInfoField

{

///错误代码

TQdpFtdcErrorIDType ErrorID;

///错误信息

TQdpFtdcErrorMsgType ErrorMsg;

};

nRequestID：返回投资者保证金率查询请求的ID，该ID 由用户在投资者保证金率查询时指定。

bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。

## CQdpFtdcTraderApi 接口

CQdpFtdcTraderApi接口提供给用户的功能包括，报单的录入、报单的撤销、报单的查询、成交单查询、客户持仓查询、合约查询、合约交易状态查询、交易所公告查询等功能。

### CreateFtdcTraderApi方法

产生一个CQdpFtdcTradeApi的一个实例，不能通过new来产生。

**函数原形：**

static CQdpFtdcTradeApi \*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QDP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CQdpFtdcTradeApi实例的指针。

### GetVersion方法

获取系统版本号。

**函数原型：**

static const char \*GetVersion(int &nMajorVersion, int &nMinorVersion);

**参数：**

param nMajorVersion 主版本号

param nMinorVersion 子版本号

**返回值：**

系统标识字符串返回值。

### Release 方法

释放一个CQdpFtdcTradeApi实例。不能使用delete方法

**函数原形：**

void Release()；

### Init 方法

使客户端开始与QDP建立连接，连接成功后可以进行登陆。

**函数原形：**

void Init()；

### 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()；

### GetTradingDay方法

获得当前交易日。只有当与QDP连接建立后才会取到正确的交易日。

**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。

### RegisterFront 方法

设置QDP的网络通讯地址，QDP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QDP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。

### RegisterNameServer 方法

设置QDP的NameServer的网络通讯地址，用于获取行情服务列表。QDP交易系统拥有多个NameServer，用户可以同时注册多个NameServer的网络通讯地址。

该方法要在Init方法之前调用。

**函数原型：**

void RegisterNameServer(char \*pszNsAddress) = 0

**参数：**

**pszNsAddress**：指向QDP服务端NameServer网络通讯地址的指针。网络地址的格式为：“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。

注意：此接口保留，但目前并未启用！

### RegisterSpi 方法

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

**函数原形：**

void RegisterSpi(CQdpFtdcTraderSpi \*pSpi) ;

**参数：**

pSpi：实现了CQdpFtdcTraderSpi接口的实例指针。

### 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(QDP\_TE\_RESUME\_TYPE nResumeType);

**参数：**

nResumeType： 私有流重传方式

QDP\_TERT\_RESTART:从本交易日开始重传

QDP\_TERT\_RESUME:从上次收到的续传

QDP\_TERT\_QUICK:只传送登录后私有流的内容

### 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(QDP\_TE\_RESUME\_TYPE nResumeType);

**参数：**

nResumeType： 公共流重传方式

QDP\_TERT\_RESTART:从本交易日开始重传

QDP\_TERT\_RESUME:从上次收到的续传

QDP\_TERT\_QUICK:只传送登录后公共流的内容

### SubscribeUserTopic方法

订阅交易员流。该方法要在Init 方法前调用。若不调用则不会收到交易员流的数据。

**函数原形：**

void SubscribeUserTopic(QDP\_TE\_RESUME\_TYPE nResumeType) = 0;

**参数：**

nResumeType： 交易员重传方式

QDP\_TERT\_RESTART:从本交易日开始重传

QDP\_TERT\_RESUME:从上次收到的续传

QDP\_TERT\_QUICK:只传送登录后公共流的内容

### SetHeartbeatTimeout方法

设置心跳超时时间。

**函数原形：**

void SetHeartbeatTimeout(unsigned int timeout) = 0;

**参数：**

param timeout 心跳超时时间(秒)

### OpenRequestLog方法

打开请求日志文件。

**函数原形：**

int OpenRequestLog(const char \*pszReqLogFileName) = 0;

**参数：**

@param pszReqLogFileName 请求日志文件名

**返回值：**

@return 0 操作成功

@return -1 打开日志文件失败

### OpenResponseLog方法

打开应答日志文件。

**函数原形：**

int OpenResponseLog(const char \* pszRspLogFileName) = 0;

**参数：**

@param pszRspLogFileName应答日志文件名

**返回值：**

@return 0 操作成功

@return -1 打开日志文件失败

### ReqUserLogin方法

用户发出登陆请求。

**函数原形：**

int ReqUserLogin (

CQdpFtdcReqUserLoginField \*pReqUserLogin,

int nRequestID)；

**参数：**

pReqUserLogin：指向用户登录请求结构的地址。

用户登录请求结构：

struct CQdpFtdcReqUserLoginField

{

///交易日

TQdpFtdcDateType TradingDay;

///交易用户代码

TQdpFtdcUserIDType UserID;

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///密码

TQdpFtdcPasswordType Password;

///用户端产品信息

TQdpFtdcProductInfoType UserProductInfo;

///接口端产品信息

TQdpFtdcProductInfoType InterfaceProductInfo;

///协议信息

TQdpFtdcProtocolInfoType ProtocolInfo;

///IP地址

TQdpFtdcIPAddressType IPAddress;

///Mac地址

TQdpFtdcMacAddressType MacAddress;

///数据中心代码

TQdpFtdcDataCenterIDType DataCenterID;

///动态密码

TQdpFtdcPasswordType OneTimePassword;

///终端IP地址

TQdpFtdcIPAddressType ClientIPAddress;

};

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。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。

### ReqUserLogout方法

用户发出登出请求。

**函数原形：**

int ReqUserLogout (

CQdpFtdcReqUserLogoutField \* pReqUserLogout,

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 CQdpFtdcReqUserLogoutField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易用户代码

TQdpFtdcUserIDType UserID;

};

### ReqUserPasswordUpdate方法

用户密码修改请求。

**函数原形：**

int ReqUserPasswordUpdate (

CQdpFtdcUserPasswordUpdateField \*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 CQdpFtdcUserPasswordUpdateField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易用户代码

TQdpFtdcUserIDType UserID;

///旧密码

TQdpFtdcPasswordType OldPassword;

///新密码

TQdpFtdcPasswordType NewPassword;

};

### ReqOrderInsert 方法

客户端发出报单录入请求。

**函数原形：**

int ReqOrderInsert(

CQdpFtdcInputOrderField \*pInputOrder,

int nRequestID)；

**参数：**

pInputOrder：指向输入报单结构的地址。

输入报单结构：

struct CQdpFtdcInputOrderField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///系统报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///用户代码

TQdpFtdcUserIDType UserID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///用户本地报单号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///报单类型

TQdpFtdcOrderPriceTypeType OrderPriceType;

///买卖方向

TQdpFtdcDirectionType Direction;

///开平标志

TQdpFtdcOffsetFlagType OffsetFlag;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

///价格

TQdpFtdcPriceType LimitPrice;

///数量

TQdpFtdcVolumeType Volume;

///有效期类型

TQdpFtdcTimeConditionType TimeCondition;

///GTD日期

TQdpFtdcDateType GTDDate;

///成交量类型

TQdpFtdcVolumeConditionType VolumeCondition;

///最小成交量

TQdpFtdcVolumeType MinVolume;

///止损价

TQdpFtdcPriceType StopPrice;

///强平原因

TQdpFtdcForceCloseReasonType ForceCloseReason;

///自动挂起标志

TQdpFtdcBoolType IsAutoSuspend;

///业务单元

TQdpFtdcBusinessUnitType BusinessUnit;

///用户自定义域

TQdpFtdcCustomType UserCustom;

///营业部代码

TQdpFtdcBranchIDType BranchID;

///记录编号

TQdpFtdcSequenceNoType RecNum;

///业务类别

TQdpFtdcBusinessTypeType BusinessType;

};

**注意：**

**用户本地报单号UserOrderLocalID是一个21位的字符串，下一笔报单的本地报单编号需要比前一笔报单的本地报单编号大（不一定需要连续），其比较方式为字符串的比较。**

**MaxOrderLocalID < UserOrderLocalID**

**限价单**必须填充的字段包括：

1. BrokerID，会员号，形如“2008”；
2. ExchangeID，交易所代码，支持“CFFEX”、“SHFE”、“CZCE”、“DCE”等
3. InvestorID，投资者编号，形如“10000029”；
4. UserID，用户代码，形如“test1”；
5. InstrumentID，合约代码，形如“IF1109”；
6. **OrderPriceType，报单价格条件，只能为QDP\_FTDC\_OPT\_LimitPrice；**
7. Direction，买卖方向，QDP\_FTDC\_D\_Buy表示买，QDP\_FTDC\_D\_Sell表示卖；
8. OffsetFlag，开平标志，“0”表示开仓，“1”表示平仓；
9. HedgeFlag，投机套保标志，“1”为投机，“2”为套利，“3”为套保，“4”为做市商；
10. LimitPrice，价格，形如3500.00；
11. Volume，数量，例如5表示5手；
12. **TimeCondition，有效期类型，为QDP\_FTDC\_TC\_IOC（“立即成交，否则撤销”）或QDP\_FTDC\_TC\_GFD（“当日有效”）；**
13. VolumeCondition，成交量类型，只能为**QDP\_**FTDC\_VC\_AV（“任意数量”）；
14. ForceCloseReason，强平原因，只能为QDP\_FTDC\_FCR\_NotForceClose（“非强平”）；
15. UserOrderLocalID，用户本地报单编号，形如”00000025”。

**市价单**必须填充的字段包括：

1. BrokerID，会员号，形如“2008”；
2. ExchangeID，交易所代码，支持“CFFEX”、“SHFE”、“CZCE”、“DCE”等
3. InvestorID，客户号，形如“10000029”；
4. UserID，交易用户代码，形如“test1”；
5. InstrumentID，合约代码，形如“IF1109”；
6. **OrderPriceType，报单价格条件，只能为QDP\_FTDC\_OPT\_AnyPrice；**
7. Direction，买卖方向，QDP\_FTDC\_D\_Buy表示买，QDP\_FTDC\_D\_Sell表示卖；
8. OffsetFlag，开平标志，“0”表示开仓，“1”表示平仓；
9. HedgeFlag，投机套保标志，“1”为投机，“2”为套利，“3”为套保，“4”为做市商；
10. Volume，数量，例如5表示5手；
11. **TimeCondition，有效期类型，只能为QDP\_FTDC\_TC\_IOC（“立即成交，否则撤销”）；**
12. VolumeCondition，成交量类型，只能为QDP\_FTDC\_VC\_AV（“任意数量”）；
13. ForceCloseReason，强平原因，只能为QDP\_FTDC\_FCR\_NotForceClose（“非强平”）；
14. UserOrderLocalID，本地报单编号，形如”00000025”。

**新增6种订单类型，详细说明如下：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **订单类型** | **OrderPriceType** | **TimeCondition** | **VolumeCondition** | **备注** |
| **FOK** | **QDP\_FTDC\_OPT\_LimitPrice** | **QDP\_FTDC\_TC\_IOC** | **QDP\_FTDC\_VC\_CV** | **立即全部成交否则自动撤销** |
| **市价转限价** | **QDP\_FTDC\_OPT\_AnyPrice** | **QDP\_FTDC\_TC\_GFD** | **QDP\_FTDC\_VC\_AV** | **市价单未成交部分转为最新价限价单** |
| **五档市价** | **QDP\_FTDC\_OPT\_FiveLevelPrice** | **QDP\_FTDC\_TC\_IOC** | **QDP\_FTDC\_VC\_AV** | **市价单与对手方五档价格报单尝试成交，剩余未成交部分撤销** |
| **五档市价转限价** | **QDP\_FTDC\_OPT\_FiveLevelPrice** | **QDP\_FTDC\_TC\_GFD** | **QDP\_FTDC\_VC\_AV** | **市价单与对手方五档价格报单尝试成交，剩余未成交部分转为最新价限价单** |
| **最优价** | **QDP\_FTDC\_OPT\_BestPrice** | **QDP\_FTDC\_TC\_IOC** | **QDP\_FTDC\_VC\_AV** | **市价单与对手方最优一档价格报单尝试成交，剩余未成交部分撤销** |
| **最优价转限价** | **QDP\_FTDC\_OPT\_BestPrice** | **QDP\_FTDC\_TC\_GFD** | **QDP\_FTDC\_VC\_AV** | **市价单与对手方最优一档价格报单尝试成交，剩余未成交部分转为最新价限价单** |

FOK单为限价单，除这3个字段外的其他字段填充方式参照限价单。其余5种订单为市价单，除这3个字段外的其他字段填充方式参照市价单。

### 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（暂不支持）、报单的激活（暂不支持）、报单的修改（暂不支持）

**函数原形：**

int ReqOrderAction(

CQdpFtdcOrderActionField \*pOrderAction,

int nRequestID)；

**参数：**

pOrderAction：指向报单操作结构的地址。

报单操作结构：

struct CQdpFtdcOrderActionField

{

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///用户代码

TQdpFtdcUserIDType UserID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///本次撤单操作的本地编号

TQdpFtdcUserOrderLocalIDType UserOrderActionLocalID;

///被撤订单的本地报单编号

TQdpFtdcUserOrderLocalIDType UserOrderLocalID;

///报单操作标志

TQdpFtdcActionFlagType ActionFlag;

///价格

TQdpFtdcPriceType LimitPrice;

///数量变化

TQdpFtdcVolumeType VolumeChange;

///记录编号

TQdpFtdcSequenceNoType RecNum;

};

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。

### ReqQryOrder方法

报单查询请求。

**函数原形：**

int ReqQryQdpOrder (

CQdpFtdcQryOrderField \*pQryOrder,

int nRequestID)；

**参数：**

pQryOrder：指向报单查询结构的地址。

报单查询结构：

struct CQdpFtdcQryOrderField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///用户代码

TQdpFtdcUserIDType UserID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///报单编号

TQdpFtdcOrderSysIDType OrderSysID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

};

### ReqQryTrade方法

成交单查询请求。

**函数原形：**

int ReqQryQdpTrade (

CQdpFtdcQryTradeField \*pQryTrade,

int nRequestID)；

**参数：**

pQryTrade：指向成交查询结构的地址。

成交查询结构：

struct CQdpFtdcQryTradeField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///用户代码

TQdpFtdcUserIDType UserID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///成交编号

TQdpFtdcTradeIDType TradeID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

};

### ReqQryUserInvestor方法

可用投资者账户查询。

**函数原形：**

int ReqQryUserInvestor(

CQdpFtdcQryUserInvestorField \*pQryUserInvestor,

int nRequestID)

**参数：**

pQryUserInvestor指向可用投资者账户查询结构的地址。

可用投资者查询结构

struct CQdpFtdcQryUserInvestorField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///用户代码

TQdpFtdcUserIDType UserID;

};

### ReqQryInvestorAccount方法

投资者资金账户查询。

**函数原形：**

int ReqQryInvestorAccount(

CQdpFtdcQryInvestorAccountField \*pQryInvestorAccount,

int nRequestID)

**参数：**

pQryInvestorAccount指向投资者账户查询结构的地址。

投资者账户查询结构：

struct CQdpFtdcQryInvestorAccountField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///用户代码

TQdpFtdcUserIDType UserID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

};

### ReqQryInstrument方法

合约信息查询。

**函数原形：**

int ReqQryInstrument (

CQdpFtdcQryInstrumentField \*pQryInstrument,

int nRequestID)

**参数：**

pQryInstrument指向合约信息查询结构的地址。

合约查询结构

struct CQdpFtdcQryInstrumentField

{

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///产品代码

TQdpFtdcProductIDType ProductID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

};

### ReqQryMarketData方法

合约行情查询。

**函数原形：**

int ReqQryMarketData (

CQdpFtdcQryMarketDataField \*pQryMarketData,

int nRequestID)

**参数：**

pQryMarketData指向合约行情查询结构的地址。

合约行情查询结构

struct CQdpFtdcQryMarketDataField

{

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

};

### ReqQryExchange方法

交易所查询。

**函数原形：**

int ReqQryExchange(

CQdpFtdcQryExchangeField \*pQryExchange,

int nRequestID)

**参数：**

pQryExchange指向交易所查询结构的地址。

交易编码结构：

struct CQdpFtdcQryExchangeField

{

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

};

### ReqQryInvestorPosition方法

投资者持仓查询。

**函数原形：**

int ReqQryInvestorPosition (

CQdpFtdcQryInvestorPositionField \*pQryUserInvestorPosition,

int nRequestID)

**参数：**

pQryUserInvestorPosition指向投资者持仓查询结构的地址。

投资者持仓查询结构

struct CQdpFtdcQryInvestorPositionField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///用户代码

TQdpFtdcUserIDType UserID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

};

### ReqSubscribeTopic方法

订阅主题请求。

**函数原形：**

int ReqSubscribeTopic(CQdpFtdcDisseminationField \*pDissemination, int nRequestID) = 0;

**参数：**

pDissemination信息分发

struct CQdpFtdcDisseminationField

{

///序列系列号

TQdpFtdcSequenceSeriesType SequenceSeries;

///序列号

TQdpFtdcSequenceNoType SequenceNo;

};

### ReqQryTopic方法

主题查询请求。

**函数原形：**

int ReqQryTopic(CQdpFtdcDisseminationField \*pDissemination, int nRequestID) = 0;

**参数：**

pDissemination信息分发

struct CQdpFtdcDisseminationField

{

///序列系列号

TQdpFtdcSequenceSeriesType SequenceSeries;

///序列号

TQdpFtdcSequenceNoType SequenceNo;

};

### ReqQryInvestorFee方法

投资者手续费率查询。

**函数原形：**

int ReqQryInvestorFee(

CQdpFtdcQryInvestorFeeField \*pQryInvestorFee,

int nRequestID)

**参数：**

pQryInvestorFee指向投资者手续费率查询结构的地址。

投资者手续费率查询结构

struct CQdpFtdcQryInvestorFeeField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///用户代码

TQdpFtdcUserIDType UserID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

};

### ReqQryInvestorMargin方法

投资者保证金率查询

**函数原形：**

int ReqQryInvestorMargin (

CQdpFtdcQryInvestorMarginField \*pQryInvestorMargin,

int nRequestID)

**参数：**

pQryInvestorMargin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结构

struct CQdpFtdcQryInvestorMarginField

{

///经纪公司编号

TQdpFtdcBrokerIDType BrokerID;

///用户代码

TQdpFtdcUserIDType UserID;

///投资者编号

TQdpFtdcInvestorIDType InvestorID;

///交易所代码

TQdpFtdcExchangeIDType ExchangeID;

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

///投机套保标志

TQdpFtdcHedgeFlagType HedgeFlag;

};

### ReqQrySGEDeferRate方法

金交所递延费率查询

**函数原形：**

int ReqQrySGEDeferRate (

CQdpFtdcQrySGEDeferRateField \*pQrySGEDeferRate,

int nRequestID)

**参数：**

pQrySGEDeferRate指向金交所递延费率查询结构的地址。

金交所递延费率查询结构

struct CQdpFtdcQrySGEDeferRateField

{

///合约代码

TQdpFtdcInstrumentIDType InstrumentID;

};

# 开发实例

// testtradeapi.cpp :

// 一个简单的例子，介绍CQdpFtdcTraderApi和CQdpFtdcTraderSpi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 <string.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

#include <float.h>

#include <stdio.h>

#include <cstdlib>

#include <iostream>

#include "QdpFtdcTraderApi.h"

#ifdef WIN32

#define WINDOWS

#endif

#ifdef WINDOWS

#include <windows.h>

typedef HANDLE THREAD\_HANDLE;

#define SLEEP(ms) Sleep(ms)

#else

#include <pthread.h>

#include <sys/time.h>

#include <unistd.h>

typedef pthread\_t THREAD\_HANDLE;

#define SLEEP(ms) sleep((ms)/1000)

typedef unsigned int DWORD;

#endif

// 报单录入操作是否完成的标志

// 经纪公司代码

TQdpFtdcBrokerIDType g\_chBrokerID;

// 交易用户代码

TQdpFtdcUserIDType g\_chUserID;

//用户本地最大报单号

int g\_UserOrderLocalID;

/////////////////////////////////////////////////////////////

class CSimpleHandler : public CQdpFtdcTraderSpi

{

public:

// 构造函数，需要一个有效的指向CQdpFtdcMduserApi实例的指针

CSimpleHandler(CQdpFtdcTraderApi \*pUserApi) :

m\_pUserApi(pUserApi) {}

~CSimpleHandler() {}

int SendOrder()

{

// 端登成功,发出报单录入请求

CQdpFtdcInputOrderField ord;

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

//经纪公司代码

strcpy(ord.BrokerID, g\_chBrokerID);

//交易所

strcpy(ord.ExchangeID, "SGE");

//投资者代码

strcpy(ord.InvestorID, "lthj0100186983");

// 用户代码

strcpy(ord.UserID, g\_chUserID);

// 合约代码

//strcpy(ord.InstrumentID, "Ag(T+D)\_Deli");

strcpy(ord.InstrumentID, "Au99.99");

//strcpy(ord.InstrumentID, "Au(T+D)");

//本地报单号

sprintf(ord.UserOrderLocalID, "%012d", ++g\_UserOrderLocalID);

//sprintf(ord.UserOrderLocalID,"%012d", 91000001);

//报单类型

//ord.BusinessType = QDP\_FTDC\_BT\_SGEDeferDeliApp;

//价格类型

ord.OrderPriceType = QDP\_FTDC\_OPT\_LimitPrice;

//ord.OrderPriceType = QDP\_FTDC\_OPT\_AnyPrice;

// 买卖方向

//strcpy(&(ord.Direction) , "1");

//ord.Direction = QDP\_FTDC\_D\_Sell;

ord.Direction = QDP\_FTDC\_D\_Buy;

// 开平标志

strcpy(&(ord.OffsetFlag), "0");

// 投机套保标志

strcpy(&(ord.HedgeFlag), "1");

// 价格

ord.LimitPrice = 800;

//ord.LimitPrice = 5000;

// 数量

ord.Volume = 1;

// 有效期类型

//strcpy(&(ord.TimeCondition),"3" );

ord.TimeCondition = QDP\_FTDC\_TC\_GFD;

//ord.TimeCondition = QDP\_FTDC\_TC\_IOC;

//成交量类型

ord.VolumeCondition = QDP\_FTDC\_VC\_AV;

//触发条件

//ord = QDP\_FTDC\_CC\_Immediately

//强平原因

ord.ForceCloseReason = QDP\_FTDC\_FCR\_NotForceClose;

// 自动挂起标志

ord.IsAutoSuspend = 0;

int ret = m\_pUserApi->ReqOrderInsert(&ord, 0);

return ret;

}

//撤单

void OrderAction()

{

CQdpFtdcOrderActionField tActionField;

memset(&tActionField, 0, sizeof(CQdpFtdcOrderActionField));

//经纪公司代码

strcpy(tActionField.BrokerID, g\_chBrokerID);

//交易所

strcpy(tActionField.ExchangeID, "");

//投资者代码

strcpy(tActionField.InvestorID, "");

// 用户代码

strcpy(tActionField.UserID, g\_chUserID);

// 合约代码

strcpy(tActionField.InstrumentID, "");

//报单编号

strcpy(tActionField.OrderSysID, "316");

m\_pUserApi->ReqOrderAction(&tActionField, 0);

}

//合约行情查询

void QueryMarketData()

{

CQdpFtdcQryMarketDataField tQryMarketData;

memset(&tQryMarketData, 0, sizeof(CQdpFtdcQryMarketDataField));

//交易所

strcpy(tQryMarketData.ExchangeID, "SSE");

// 合约代码

strcpy(tQryMarketData.InstrumentID, "10002931");

m\_pUserApi->ReqQryMarketData(&tQryMarketData, 0);

}

//合约查询

void QueryInstrument()

{

CQdpFtdcQryInstrumentField tQryInstrument;

memset(&tQryInstrument, 0, sizeof(CQdpFtdcQryInstrumentField));

//交易所

strcpy(tQryInstrument.ExchangeID, "");

// 合约代码

strcpy(tQryInstrument.InstrumentID, "");

m\_pUserApi->ReqQryInstrument(&tQryInstrument, 0);

}

//资金查询

void QueryIvestorAccount()

{

CQdpFtdcQryInvestorAccountField tQryInvestorAccount;

memset(&tQryInvestorAccount, 0, sizeof(CQdpFtdcQryInvestorAccountField));

//经纪公司代码

strcpy(tQryInvestorAccount.BrokerID, g\_chBrokerID);

//投资者代码

strcpy(tQryInvestorAccount.InvestorID, "01500006354870");

// 用户代码

strcpy(tQryInvestorAccount.UserID, g\_chUserID);

m\_pUserApi->ReqQryInvestorAccount(&tQryInvestorAccount, 0);

}

//委托查询

void QueryOrder()

{

CQdpFtdcQryOrderField tQryOrderField;

memset(&tQryOrderField, 0, sizeof(CQdpFtdcQryOrderField));

//经纪公司代码

strcpy(tQryOrderField.BrokerID, g\_chBrokerID);

//交易所

strcpy(tQryOrderField.ExchangeID, "");

// 用户代码

strcpy(tQryOrderField.UserID, g\_chUserID);

m\_pUserApi->ReqQryOrder(&tQryOrderField, 0);

}

//成交查询

void QueryTrade()

{

CQdpFtdcQryTradeField tQryTradeField;

memset(&tQryTradeField, 0, sizeof(CQdpFtdcQryTradeField));

//经纪公司代码

strcpy(tQryTradeField.BrokerID, g\_chBrokerID);

//交易所

strcpy(tQryTradeField.ExchangeID, "");

// 用户代码

strcpy(tQryTradeField.UserID, g\_chUserID);

m\_pUserApi->ReqQryTrade(&tQryTradeField, 0);

}

void QueryDeferRate()

{

CQdpFtdcQrySGEDeferRateField tQryDeferRate;

memset(&tQryDeferRate, 0, sizeof(CQdpFtdcQrySGEDeferRateField));

strcpy(tQryDeferRate.InstrumentID, "Ag(T+D)");

m\_pUserApi->ReqQrySGEDeferRate(&tQryDeferRate, 0);

}

// 当客户端与拓界科技建立起通信连接，客户端需要进行登录

virtual void OnFrontConnected()

{

CQdpFtdcReqUserLoginField reqUserLogin;

memset(&reqUserLogin, 0, sizeof(CQdpFtdcReqUserLoginField));

printf("OnFrontConnected\n");

strcpy(g\_chBrokerID, "lthj");

strcpy(g\_chUserID, "0100186983");

//strcpy(g\_chUserID, "015000062166");

strcpy(reqUserLogin.BrokerID, g\_chBrokerID);

strcpy(reqUserLogin.UserID, g\_chUserID);

strcpy(reqUserLogin.Password, "123456");

// 发出登陆请求

m\_pUserApi->ReqUserLogin(&reqUserLogin, 0);

}

// 当客户端与拓界科技通信连接断开时，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(CQdpFtdcRspUserLoginField \*pRspUserLogin,

CQdpFtdcRspInfoField \*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);

printf("maxlocalid=[%s]\n", pRspUserLogin->MaxOrderLocalID);

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);

}

Sleep(1000);

#if 0

//用户最大本地报单号

g\_UserOrderLocalID = atoi(pRspUserLogin->MaxOrderLocalID) + 1;

printf("maxlocalid = [%s]\n", pRspUserLogin->MaxOrderLocalID);

printf("g\_UserOrderLocalID = [%d]\n", g\_UserOrderLocalID);

CQdpFtdcQryLockPositionField pQryInstrument;

memset(&pQryInstrument, 0x00, sizeof(CQdpFtdcQryLockPositionField));

m\_pUserApi->ReqQryLockPosition(&pQryInstrument, 0);

#endif

//下单

/\*for (int i = 0; i < 1001; i++)

{

SendOrder();

}\*/

//查询递延费率

QueryDeferRate();

//SendOrder();

//撤单

//OrderAction();

//客户合约持仓查询

//QryInvestorPosition();

//查询合约行情

//QueryMarketData();

//QueryInstrument();

//QueryIvestorAccount();

//QueryOrder();

//QueryTrade();

//QueryInvestor();

}

void OnRspOrderAction(CQdpFtdcOrderActionField \*pOrderAction, CQdpFtdcRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)

{

if (NULL != pOrderAction)

{

printf("交易所: %s\n", pOrderAction->ExchangeID);

printf("合约代码: %s; ", pOrderAction->InstrumentID);

printf("报单编号: %s;", pOrderAction->OrderSysID);

printf("价格: %lf;", pOrderAction->LimitPrice);

printf("boolIsLast: %d", bIsLast);

printf("\n");

}

}

void OnRspQryMarketData(CQdpFtdcMarketDataField \*pMarketData, CQdpFtdcRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)

{

printf("合约行情查询结果OnRspQryMarketData ErrorCode=[%d], ErrorMsg=[%s]\n", pRspInfo->ErrorID, pRspInfo->ErrorMsg);

if (NULL != pMarketData)

{

printf("交易所: %s\n", pMarketData->ExchangeID);

printf("合约代码: %s; ", pMarketData->InstrumentID);

printf("最高价: %lf;", pMarketData->HighestPrice);

printf("最低价: %lf;", pMarketData->LowestPrice);

printf("最新价: %lf;", pMarketData->LastPrice);

printf("boolIsLast: %d", bIsLast);

printf("\n");

}

}

void OnRspQryInstrument(CQdpFtdcRspInstrumentField \*pRspInstrument, CQdpFtdcRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)

{

if (pRspInstrument)

{

printf("交易所===>[%s]; ", pRspInstrument->ExchangeID);

printf("合约代码===>[%s]; ", pRspInstrument->InstrumentID);

printf("PriceTick===>[%lf]; ", pRspInstrument->PriceTick);

printf("boolIsLast: %d", bIsLast);

printf("\n");

}

}

void OnRspQryInvestorAccount(CQdpFtdcRspInvestorAccountField \*pRspInvestorAccount, CQdpFtdcRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)

{

// 输出报单录入结果

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

if (NULL != pRspInvestorAccount)

{

printf("投资者%s持仓信息:\n", pRspInvestorAccount->InvestorID);

printf("可用资金: %lf; ", pRspInvestorAccount->Available);

printf("boolIsLast: %d", bIsLast);

printf("\n");

}

}

void OnRspQryInvestorPosition(CQdpFtdcRspInvestorPositionField \*pRspInvestorPosition, CQdpFtdcRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)

{

// 输出报单录入结果

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

if (NULL != pRspInvestorPosition)

{

printf("投资者%s持仓信息:\n", pRspInvestorPosition->InvestorID);

printf("合约代码: %s; ", pRspInvestorPosition->InstrumentID);

printf("总持仓量:%d;", pRspInvestorPosition->Position);

printf("投机套保标志:%c;", pRspInvestorPosition->HedgeFlag);

printf("保证金:%lf;", pRspInvestorPosition->UsedMargin);

printf("买卖方向:%c;", pRspInvestorPosition->Direction);

printf("boolIsLast: %d", bIsLast);

printf("\n");

}

}

// 报单录入应答

virtual void OnRspOrderInsert(CQdpFtdcInputOrderField \*pInputOrder,

CQdpFtdcRspInfoField \*pRspInfo,

int nRequestID,

bool bIsLast)

{

// 输出报单录入结果

if (pRspInfo->ErrorID != 0)

{

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

//SendOrder();

return;

}

printf("OnRspOrderInsert ErrorCode=[%d], ErrorMsg=[%s], orderSysID=[%s]\n", pRspInfo->ErrorID, pRspInfo->ErrorMsg, pInputOrder->OrderSysID);

//SendOrder();

}

///报单回报

void OnRtnOrder(CQdpFtdcOrderField \*pOrder)

{

printf("OnRtnOrder->OrderSysID=[%s], OrderStatus=[%c]\n", pOrder->OrderSysID, pOrder->OrderStatus);

}

void OnRtnTrade(CQdpFtdcTradeField \*pTrade)

{

printf("OnRtnTrade->OrderSysID=[%s], tradeID=[%s]\n", pTrade->OrderSysID, pTrade->TradeID);

}

void OnRtnSGEDeferRate(CQdpFtdcSGEDeferRateField \*pSGEDeferRate)

{

printf("OnRtnSGEDeferRate: InstrumentID=[%s], ExchangeID=[%s], TradeDate=[%s], Direction=[%c], DeferRate=[%lf]\n",

pSGEDeferRate->InstrumentID, pSGEDeferRate->ExchangeID, pSGEDeferRate->TradeDate,

pSGEDeferRate->Direction, pSGEDeferRate->DeferRate);

}

// 针对用户请求的出错通知

virtual void OnRspError(CQdpFtdcRspInfoField \*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

{

//客户端的错误处理

}

}

void OnRspQryOrder(CQdpFtdcOrderField \*pOrder, CQdpFtdcRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)

{

// 输出报单录入结果

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

if (NULL != pOrder)

{

printf("委托代码: %s; ", pOrder->OrderSysID);

printf("合约代码: %s; ", pOrder->InstrumentID);

printf("价格:%lf;", pOrder->LimitPrice);

printf("数量:%d;", pOrder->Volume);

printf("boolIsLast: %d", bIsLast);

printf("\n");

}

}

void OnRspQryTrade(CQdpFtdcTradeField \*pTrade, CQdpFtdcRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)

{

// 输出报单录入结果

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

if (NULL != pTrade)

{

printf("合约代码: %s; ", pTrade->InstrumentID);

printf("成交编号: %s; ", pTrade->TradeID);

printf("买卖方向: %c; ", pTrade->Direction);

printf("开平标志: %c; ", pTrade->OffsetFlag);

printf("成交价格:%lf;", pTrade->TradePrice);

printf("成交数量:%d;", pTrade->TradeVolume);

printf("boolIsLast: %d", bIsLast);

printf("\n");

}

}

void OnRspQrySGEDeferRate(CQdpFtdcSGEDeferRateField \*pSGEDeferRate, CQdpFtdcRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)

{

if (NULL != pSGEDeferRate)

{

printf("InstrumentID: %s, ", pSGEDeferRate->InstrumentID);

printf("ExchangeID: %s, ", pSGEDeferRate->ExchangeID);

printf("TradeDate: %s, ", pSGEDeferRate->TradeDate);

printf("Direction: %c, ", pSGEDeferRate->Direction);

printf("DeferRate: %f, ", pSGEDeferRate->DeferRate);

printf("boolIsLast: %d", bIsLast);

printf("\n");

}

}

private:

// 指向CQdpFtdcMduserApi实例的指针

CQdpFtdcTraderApi \*m\_pUserApi;

};

int main()

{

// 产生一个CQdpFtdcTraderApi实例

CQdpFtdcTraderApi \*pUserApi = CQdpFtdcTraderApi::CreateFtdcTraderApi();

// 产生一个事件处理的实例

CSimpleHandler sh(pUserApi);

// 注册一事件处理的实例

pUserApi->RegisterSpi(&sh);

// 订阅私有流

// Qdp\_TERT\_RESTART:从本交易日开始重传

// Qdp\_TERT\_RESUME:从上次收到的续传

// Qdp\_TERT\_QUICK:只传送登录后私有流的内容

pUserApi->SubscribePrivateTopic(Qdp\_TERT\_RESUME);

// 订阅公共流

// Qdp\_TERT\_RESTART:从本交易日开始重传

// Qdp\_TERT\_RESUME:从上次收到的续传

// Qdp\_TERT\_QUICK:只传送登录后公共流的内容

pUserApi->SubscribePublicTopic(Qdp\_TERT\_RESUME);

// 设置拓界科技服务的地址，可以注册多个地址备用

pUserApi->RegisterFront("tcp://127.0.0.1:30005");

// pUserApi->SetHeartbeatTimeout(240);

// 使客户端开始与后台服务建立连接

pUserApi->Init();

pUserApi->Join();

// 释放API实例

pUserApi->Release();

return 0;

}