LTS 系统
交易员应用程序接口

2014年 1月

1. 文件属性

文件属性	内容
文件名称	LTS系统TradeAPI接口
文件编号	
文件版本号	V0.1
文件状态	草稿
作　　者	华宝证券有限责任公司
文档编写日期	2014-1-21
文档发布日期	

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	文件描述
SecurityFtdcTraderApi.h	V1.0	交易接口头文件
SecurityFtdcUserApiStruct.h	V1.0	定义了 API 所需的一系列数据类型的头文件
SecurityFtdcUserApiDataType.h	V1.0	定义了一系列业务相关的数据结构的头文件
securitytraderapi.dll	V1.0	动态链接库二进制文件
securitytraderapi.lib	V1.0	导入库文件
securitymduserapi.dll	V1.0	动态链接库二进制文件
securitymduserapi.lib	V1.0	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:

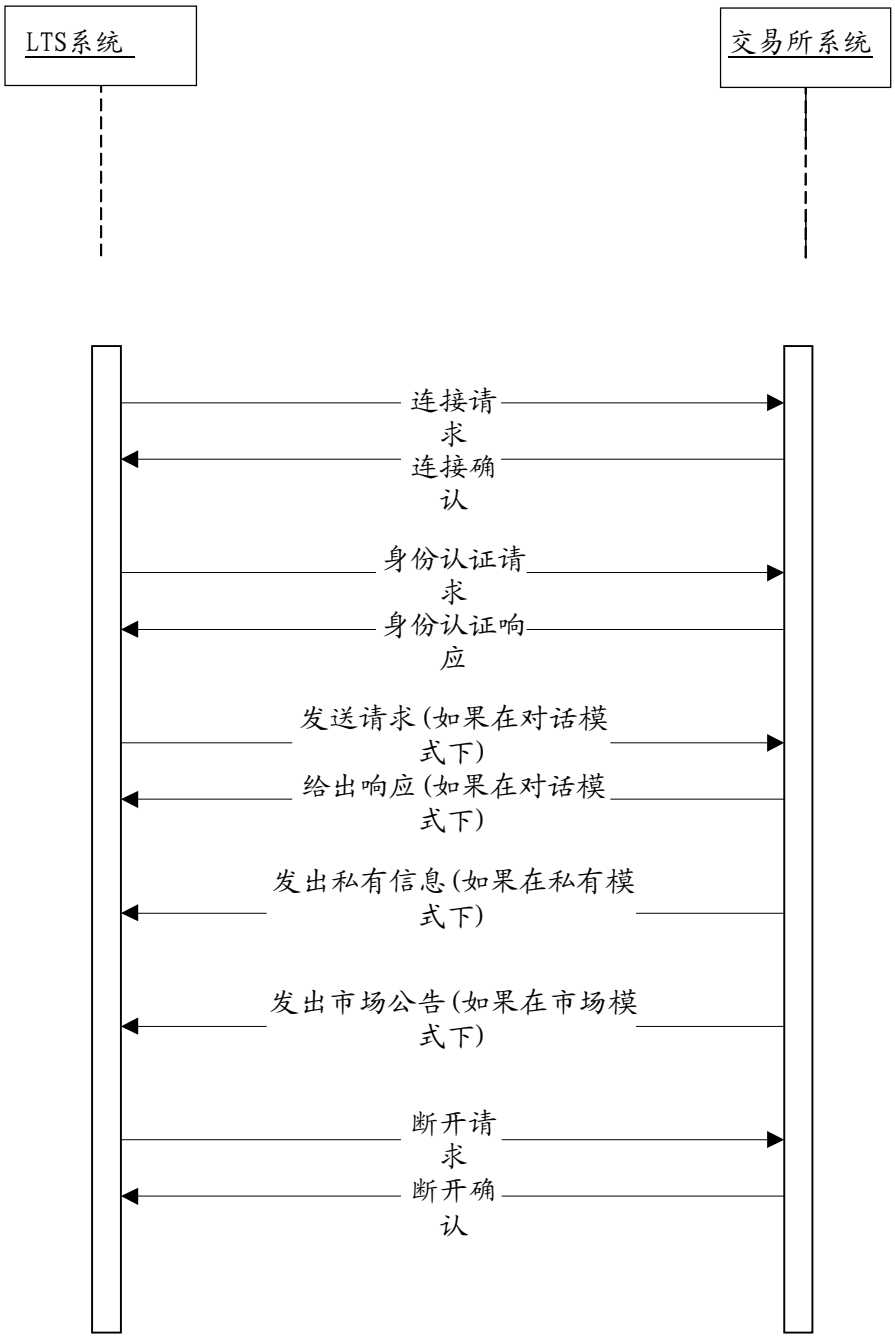


图1) 各通讯模式的工作过程

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

交易员API提供了两个接口，分别为CSecurityFtdcTraderApi和CSecurityFtdcTraderSpi。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CSecurityFtdcTraderApi发出操作请求，通过继承CSecurityFtdcTraderSpi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 CSecurityFtdcTraderApi::ReqXXX(
                CSecurityFtdcXXXField *pReqXXX,
                int nRequestID)
响应: void CSecurityFtdcTraderSpi::OnRspXXX(
                CSecurityFtdcXXXField *pRspXXX,
                CSecurityFtdcRspInfoField *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，CSecurityFtdcTraderSpi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 CSecurityFtdcTraderSpi::OnRtnXXX(CSecurityFtdcXXXField *pXXX)  或
void CSecurityFtdcTraderSpi::OnErrRtnXXX(

CSecurityFtdcXXXField *pXXX,

CSecurityFtdcRspInfoField *pRspInfo)
```

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

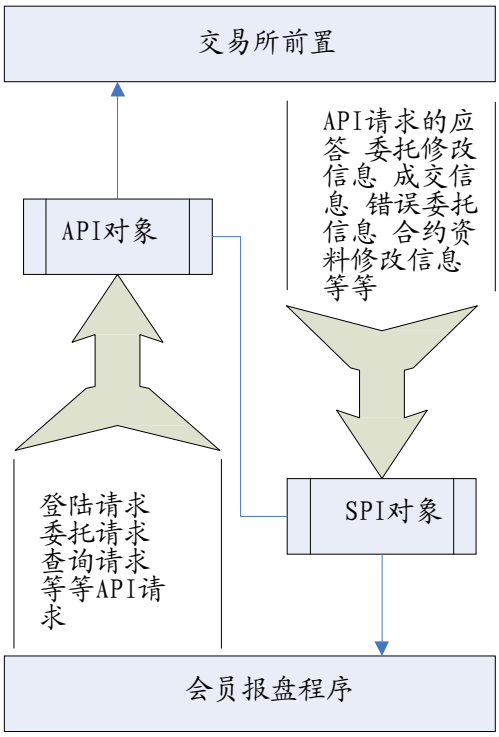
4. 运行模式

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。

CSecurityFtdcTraderApi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。

CSecurityFtdcTraderSpi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CSecurityFtdcTraderSpi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	CSecurityFtdcTraderApi:: ReqUserLogin	CSecurityFtdcTraderSpi::OnRspUserLogin	对话流
	登出	CSecurityFtdcTraderApi::ReqUserLogout	CSecurityFtdcTraderSpi::OnRspUserLogout	对话流
	用户口令更新	CSecurityFtdcTraderApi::ReqUserPasswordUpdat	CSecurityFtdcTraderSpi::OnRspUserPasswordUpdat	对话流
	资金账户口令更新请求	CSecurityFtdcTraderApi:: ReqTradingAccountPasswordUpdate	CSecurityFtdcTraderSpi:: OnRspTradingAccountPasswordUpdate	对话流
交易	报单录入	CSecurityFtdcTraderApi::ReqOrderInsert	CSecurityFtdcTraderSpi::OnRspOrderInsert	对话流
	报单操作	CSecurityFtdcTraderApi::ReqOrderAction	CSecurityFtdcTraderSpi::OnRspOrderAction	对话流
私有回报	成交回报	N/A	CSecurityFtdcTraderSpi::OnRtnTrade	私有流
	报单回报	N/A	CSecurityFtdcTraderSpi::OnRtnOrder	私有流
	报单录入错误回报	N/A	CSecurityFtdcTraderSpi::OnErrRtnOrderInsert	私有流
	报单操作错误回报	N/A	CSecurityFtdcTraderSpi::OnErrRtnOrderAction	私有流
查询	报单查询	CSecurityFtdcTraderApi::ReqQryOrder	CSecurityFtdcTraderSpi::OnRspQryOrder	查询流
	成交查询	CSecurityFtdcTraderApi::ReqQryTrade	CSecurityFtdcTraderSpi::OnRspQryTrade	查询流
	投资者查询	CSecurityFtdcTraderApi::ReqQryInvestor	CSecurityFtdcTraderSpi::OnRspQryInvestor	查询流
	请求查询交易所	CSecurityFtdcTraderApi:: ReqQryExchange	CSecurityFtdcTraderSpi:: OnRspQryExchange	查询流
	请求查询交易编码	CSecurityFtdcTraderApi:: ReqQryTradingCode	CSecurityFtdcTraderSpi:: OnRspQryTradingCode	查询流
	请求查询资金账户	CSecurityFtdcTraderApi::	CSecurityFtdcTraderSpi::	查询流
	请求查询行情	CSecurityFtdcTraderApi::	CSecurityFtdcTraderSpi::	查询流
	请求查询投资者持仓明细	CSecurityFtdcTraderApi::	CSecurityFtdcTraderSpi::	查询流
	请求查询债券利息	CSecurityFtdcTraderApi:: ReqQryBondInterest	CSecurityFtdcTraderSpi:: OnRspQryBondInterest	查询流
	投资者持仓查询	CSecurityFtdcTraderApi::ReqQry Investor	CSecurityFtdcTraderSpi::OnRspQry Investor	查询流
	合约查询	CSecurityFtdcTraderApi::ReqQryInstrument	CSecurityFtdcTraderSpi::OnRspQryInstrument	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CSecurityFtdcTraderApi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

1. 交易系统现在提供多个接入地址。
2. 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.经纪公司代码设置要求

客户端需要保存客户选择的经纪公司代码：2011

6.4.CSecurityFtdcTraderSpi 接口

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

6.4.1. 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。

6.4.2. 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

6.4.3. 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

6.4.4. 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(

```
                    CSecurityFtdcRspUserLoginField *pRspUserLogin,
                    CSecurityFtdcRspInfoField *pRspInfo,
                    int RequestID,
                    bool IsLast);
```

参数:

pRspUserLogin: 返回用户登录信息的地址。

用户登录信息结构:

```
struct CSecurityFtdcRspUserLoginField
{
    ///交易日
    TSecurityFtdcDateType     TradingDay;
    ///登录成功时间
    TSecurityFtdcTimeType     LoginTime;
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///交易系统名称
    TSecurityFtdcSystemNameTypeSystemName;
};
```

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 CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.5. 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(
                    CSecurityFtdcUserLogoutField*pUserLogout,
                    CSecurityFtdcRspInfoField *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 CSecurityFtdcUserLogoutField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///用户代码
    TSecurityFtdcUserIDType UserID;
};
```

pRspInfo: 返回用户响应信息的地址。

响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.6. 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(
                        CSecurityFtdcUserPasswordUpdateField
                        *pUserPasswordUpdate, CSecurityFtdcRspInfoField
                        *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 CSecurityFtdcUserPasswordUpdateField
{
    ///经纪公司代码
```

```
    TSecurityFtdcBrokerIDType BrokerID;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///原来的口令
    TSecurityFtdcPasswordType OldPassword;
    ///新的口令
    TSecurityFtdcPasswordType NewPassword;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.7. 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(
        CSecurityFtdcTradingAccountPasswordUpdateField
        *pTradingAccountPasswordUpdate, CSecurityFtdcRspInfoField *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 CSecurityFtdcTradingAccountPasswordUpdateField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者帐号
    TSecurityFtdcAccountIDType    AccountID;
    ///原来的口令
    TSecurityFtdcPasswordType OldPassword;
```

```
    ///新的口令
    TSecurityFtdcPasswordType NewPassword;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.8. OnRspError 方法

针对用户请求的出错通知。

函数原形:

```
void OnRspError(
            CSecurityFtdcRspInfoField
            *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 CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.9. 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(CSecurityFtdcInputOrderField *pInputOrder,
                      CSecurityFtdcRspInfoField *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 CSecurityFtdcInputOrderField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///报单引用
    TSecurityFtdcOrderRefType OrderRef;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///报单价格条件
    TSecurityFtdcOrderPriceTypeType  OrderPriceType;
    ///买卖方向
    TSecurityFtdcDirectionType       Direction;
    ///组合开平标志
    TSecurityFtdcCombOffsetFlagType   CombOffsetFlag;
    ///组合投机套保标志
    TSecurityFtdcCombHedgeFlagType  CombHedgeFlag;
    ///价格
    TSecurityFtdcPriceType    LimitPrice;
    ///数量
    TSecurityFtdcVolumeType VolumeTotalOriginal;
    ///有效期类型
    TSecurityFtdcTimeConditionType   TimeCondition;
    ///GTD 日期
    TSecurityFtdcDateType     GTDDate;
    ///成交量类型
    TSecurityFtdcVolumeConditionType VolumeCondition;
    ///最小成交量
    TSecurityFtdcVolumeType MinVolume;
    ///触发条件
    TSecurityFtdcContingentConditionType    ContingentCondition;
    ///止损价
    TSecurityFtdcPriceType    StopPrice;
    ///强平原因
```

```
    TSecurityFtdcForceCloseReasonType       ForceCloseReason;
    ///自动挂起标志
    TSecurityFtdcBoolType     IsAutoSuspend;
    ///业务单元
    TSecurityFtdcBusinessUnitTypeBusinessUnit;
    ///请求编号
    TSecurityFtdcRequestIDType       RequestID;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.10. 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(
                    CSecurityFtdcOrderActionField
                    pOrderAction,
                    CSecurityFtdcRspInfoField *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 CSecurityFtdcOrderActionField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///报单操作引用
```

```
TSecurityFtdcOrderActionRefType    OrderActionRef;
///报单引用
TSecurityFtdcOrderRefType OrderRef;
///请求编号
TSecurityFtdcRequestIDType       RequestID;
///前置编号
TSecurityFtdcFrontIDType FrontID;
///会话编号
TSecurityFtdcSessionIDType       SessionID;
///交易所代码
TSecurityFtdcExchangeIDType  ExchangeID;
///报单编号
TSecurityFtdcOrderSysIDType  OrderSysID;
///操作标志
TSecurityFtdcActionFlagType    ActionFlag;
///价格
TSecurityFtdcPriceType    LimitPrice;
///数量变化
TSecurityFtdcVolumeType  VolumeChange;
///操作日期
TSecurityFtdcDateType     ActionDate;
///操作时间
TSecurityFtdcTimeType     ActionTime;
///交易所交易员代码
TSecurityFtdcTraderIDType TraderID;
///安装编号
TSecurityFtdcInstallIDType       InstallID;
///本地报单编号
TSecurityFtdcOrderLocalIDType    OrderLocalID;
///操作本地编号
TSecurityFtdcOrderLocalIDType    ActionLocalID;
///会员代码
TSecurityFtdcParticipantIDType   ParticipantID;
///客户代码
TSecurityFtdcClientIDType ClientID;
///业务单元
TSecurityFtdcBusinessUnitTypeBusinessUnit;
///报单操作状态
TSecurityFtdcOrderActionStatusType      OrderActionStatus;
///用户代码
TSecurityFtdcUserIDType  UserID;
///状态信息
TSecurityFtdcErrorMsgType StatusMsg;
```

```
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.11. 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(
                    CSecurityFtdcOrderField
                    *pOrder,
                    CSecurityFtdcRspInfoField
                    *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 CSecurityFtdcOrderField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///报单引用
    TSecurityFtdcOrderRefType OrderRef;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///报单价格条件
    TSecurityFtdcOrderPriceTypeType  OrderPriceType;
```

```
///买卖方向
TSecurityFtdcDirectionType	Direction;
///组合开平标志
TSecurityFtdcCombOffsetFlagType	CombOffsetFlag;
///组合投机套保标志
TSecurityFtdcCombHedgeFlagType	CombHedgeFlag;
///价格
TSecurityFtdcPriceType	LimitPrice;
///数量
TSecurityFtdcVolumeType	VolumeTotalOriginal;
///有效期类型
TSecurityFtdcTimeConditionType	TimeCondition;
///GTD 日期
TSecurityFtdcDateType	GTDDate;
///成交量类型
TSecurityFtdcVolumeConditionType	VolumeCondition;
///最小成交量
TSecurityFtdcVolumeType	MinVolume;
///触发条件
TSecurityFtdcContingentConditionType	ContingentCondition;
///止损价
TSecurityFtdcPriceType	StopPrice;
///强平原因
TSecurityFtdcForceCloseReasonType	ForceCloseReason;
///自动挂起标志
TSecurityFtdcBoolType	IsAutoSuspend;
///业务单元
TSecurityFtdcBusinessUnitTypeBusinessUnit;
///请求编号
TSecurityFtdcRequestIDType	RequestID;
///本地报单编号
TSecurityFtdcOrderLocalIDType	OrderLocalID;
///交易所代码
TSecurityFtdcExchangeIDType	ExchangeID;
///会员代码
TSecurityFtdcParticipantIDType	ParticipantID;
///客户代码
TSecurityFtdcClientIDType ClientID;
///合约在交易所的代码
TSecurityFtdcExchangeInstIDType	ExchangeInstID;
///交易所交易员代码
TSecurityFtdcTraderIDType TraderID;
///安装编号
```

```
TSecurityFtdcInstallIDType	InstallID;
///报单提交状态
TSecurityFtdcOrderSubmitStatusType	OrderSubmitStatus;
///报单提示序号
TSecurityFtdcSequenceNoType NotifySequence;
///交易日
TSecurityFtdcDateType	TradingDay;
///结算编号
TSecurityFtdcSettlementIDTypeSettlementID;
///报单编号
TSecurityFtdcOrderSysIDType	OrderSysID;
///报单来源
TSecurityFtdcOrderSourceType OrderSource;
///报单状态
TSecurityFtdcOrderStatusType	OrderStatus;
///报单类型
TSecurityFtdcOrderTypeType	OrderType;
///今成交数量
TSecurityFtdcVolumeType VolumeTraded;
///剩余数量
TSecurityFtdcVolumeType VolumeTotal;
///报单日期
TSecurityFtdcDateType	InsertDate;
///插入时间
TSecurityFtdcTimeType	InsertTime;
///激活时间
TSecurityFtdcTimeType	ActiveTime;
///挂起时间
TSecurityFtdcTimeType	SuspendTime;
///最后修改时间
TSecurityFtdcTimeType	UpdateTime;
///撤销时间
TSecurityFtdcTimeType	CancelTime;
///最后修改交易所交易员代码
TSecurityFtdcTraderIDType ActiveTraderID;
///结算会员编号
TSecurityFtdcParticipantIDType	ClearingPartID;
///序号
TSecurityFtdcSequenceNoType SequenceNo;
///前置编号
TSecurityFtdcFrontIDType FrontID;
///会话编号
TSecurityFtdcSessionIDType	SessionID;
```

```
    ///用户端产品信息
    TSecurityFtdcProductInfoType  UserProductInfo;
    ///状态信息
    TSecurityFtdcErrorMsgType StatusMsg;
};
```

pRspInfo: 指向响应信息结构的地址。

响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.12. 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(
        CSecurityFtdcTradeField *pTrade,
        CSecurityFtdcRspInfoField *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 CSecurityFtdcTradeField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType       BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType     InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType   InstrumentID;
    ///报单引用
    TSecurityFtdcOrderRefType       OrderRef;
    ///用户代码
    TSecurityFtdcUserIDType UserID;
    ///交易所代码
```

```
TSecurityFtdcExchangeIDType ExchangeID;
///成交编号
TSecurityFtdcTradeIDType TradeID;
///买卖方向
TSecurityFtdcDirectionType      Direction;
///报单编号
TSecurityFtdcOrderSysIDType  OrderSysID;
///会员代码
TSecurityFtdcParticipantIDType  ParticipantID;
///客户代码
TSecurityFtdcClientIDType       ClientID;
///交易角色
TSecurityFtdcTradingRoleType TradingRole;
///合约在交易所的代码
TSecurityFtdcExchangeInstIDType   ExchangeInstID;
///开平标志
TSecurityFtdcOffsetFlagType     OffsetFlag;
///投机套保标志
TSecurityFtdcHedgeFlagType      HedgeFlag;
///价格
TSecurityFtdcPriceType   Price;
///数量
TSecurityFtdcVolumeType Volume;
///成交时期
TSecurityFtdcDateType    TradeDate;
///成交时间
TSecurityFtdcTimeType    TradeTime;
///成交类型
TSecurityFtdcTradeTypeType      TradeType;
///成交价来源
TSecurityFtdcPriceSourceType  PriceSource;
///交易所交易员代码
TSecurityFtdcTraderIDType       TraderID;
///本地报单编号
TSecurityFtdcOrderLocalIDType   OrderLocalID;
///结算会员编号
TSecurityFtdcParticipantIDType  ClearingPartID;
///业务单元
TSecurityFtdcBusinessUnitTypeBusinessUnit;
///序号
TSecurityFtdcSequenceNoType SequenceNo;
///交易日
TSecurityFtdcDateType    TradingDay;
```

```
    ///结算编号
    TSecurityFtdcSettlementIDTypeSettlementID;
};
```

pRspInfo: 指向响应信息结构的地址。

响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.13. 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 (
                CSecurityFtdcInvestorField
                *pInvestor,
                CSecurityFtdcRspInfoField
                *pRspInfo, int nRequestID,
                bool
                bIsLast);
```

参数:

pInvestor: 指向投资者信息结构的地址。 投资者信息结构:

```
struct CSecurityFtdcInvestorField
{
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者分组代码
    TSecurityFtdcInvestorIDType      InvestorGroupID;
    ///投资者名称
    TSecurityFtdcPartyNameType    InvestorName;
    ///证件类型
    TSecurityFtdcIdCardTypeType   IdentifiedCardType;
    ///证件号码
    TSecurityFtdcIdentifiedCardNoType IdentifiedCardNo;
    ///是否活跃
    TSecurityFtdcBoolType     IsActive;
```

```
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.14. 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(
                        CSecurityFtdcInvestorPositionField
                        *pInvestorPosition,
                        CSecurityFtdcRspInfoField *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 CSecurityFtdcInvestorPositionField
{
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///持仓多空方向
    TSecurityFtdcPosiDirectionType   PosiDirection;
    ///投机套保标志
    TSecurityFtdcHedgeFlagType       HedgeFlag;
    ///持仓日期
```

```
TSecurityFtdcPositionDateType PositionDate;
///上日持仓
TSecurityFtdcVolumeType YdPosition;
///今日持仓
TSecurityFtdcVolumeType Position;
///多头冻结
TSecurityFtdcVolumeType LongFrozen;
///空头冻结
TSecurityFtdcVolumeType ShortFrozen;
///开仓冻结金额
TSecurityFtdcMoneyType  LongFrozenAmount;
///开仓冻结金额
TSecurityFtdcMoneyType  ShortFrozenAmount;
///开仓量
TSecurityFtdcVolumeType OpenVolume;
///平仓量
TSecurityFtdcVolumeType CloseVolume;
///开仓金额
TSecurityFtdcMoneyType  OpenAmount;
///平仓金额
TSecurityFtdcMoneyType  CloseAmount;
///持仓成本
TSecurityFtdcMoneyType  PositionCost;
///上次占用的保证金
TSecurityFtdcMoneyType  PreMargin;
///占用的保证金
TSecurityFtdcMoneyType  UseMargin;
///冻结的保证金
TSecurityFtdcMoneyType  FrozenMargin;
///冻结的资金
TSecurityFtdcMoneyType  FrozenCash;
///冻结的手续费
TSecurityFtdcMoneyType  FrozenCommission;
///资金差额
TSecurityFtdcMoneyType  CashIn;
///手续费
TSecurityFtdcMoneyType  Commission;
///平仓盈亏
TSecurityFtdcMoneyType  CloseProfit;
///持仓盈亏
TSecurityFtdcMoneyType  PositionProfit;
///上次结算价
TSecurityFtdcPriceType    PreSettlementPrice;
```

```
    ///本次结算价
    TSecurityFtdcPriceType    SettlementPrice;
    ///交易日
    TSecurityFtdcDateType     TradingDay;
    ///结算编号
    TSecurityFtdcSettlementIDTypeSettlementID;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.15. 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(
                CSecurityFtdcTradingAccountField
                *pTradingAccount,
                CSecurityFtdcRspInfoField *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 CSecurityFtdcTradingAccountField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者帐号
    TSecurityFtdcAccountIDType    AccountID;
    ///上次质押金额
    TSecurityFtdcMoneyType  PreMortgage;
    ///上次信用额度
```

```
    TSecurityFtdcMoneyType  PreCredit;
    ///上次存款额
    TSecurityFtdcMoneyType  PreDeposit;
    ///上次结算准备金
    TSecurityFtdcMoneyType  PreBalance;
    ///上次占用的保证金
    TSecurityFtdcMoneyType  PreMargin;
    ///利息基数
    TSecurityFtdcMoneyType  InterestBase;
    ///利息收入
    TSecurityFtdcMoneyType  Interest;
    ///入金金额
    TSecurityFtdcMoneyType  Deposit;
    ///出金金额
    TSecurityFtdcMoneyType  Withdraw;
    ///冻结的保证金
    TSecurityFtdcMoneyType  FrozenMargin;
    ///冻结的资金
    TSecurityFtdcMoneyType  FrozenCash;
    ///冻结的手续费
    TSecurityFtdcMoneyType  FrozenCommission;
    ///当前保证金总额
    TSecurityFtdcMoneyType  CurrMargin;
    ///资金差额
    TSecurityFtdcMoneyType  CashIn;
    ///手续费
    TSecurityFtdcMoneyType  Commission;
    ///平仓盈亏
    TSecurityFtdcMoneyType  CloseProfit;
    ///持仓盈亏
    TSecurityFtdcMoneyType  PositionProfit;
    ///期货结算准备金
    TSecurityFtdcMoneyType  Balance;
    ///可用资金
    TSecurityFtdcMoneyType  Available;
    ///可取资金
    TSecurityFtdcMoneyType  WithdrawQuota;
    ///基本准备金
    TSecurityFtdcMoneyType  Reserve;
    ///交易日
    TSecurityFtdcDateType    TradingDay;
    ///结算编号
    TSecurityFtdcSettlementIDTypeSettlementID;
    ///信用额度
    TSecurityFtdcMoneyType  Credit;
    ///质押金额
    TSecurityFtdcMoneyType  Mortgage;
    ///交易所保证金
    TSecurityFtdcMoneyType  ExchangeMargin;
};
```

pRspInfo: 指向响应信息结构的地址。 响

应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.16. 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(
                CSecurityFtdcTradingCodeField *pTradingCode,
                CSecurityFtdcRspInfoField *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 CSecurityFtdcTradingCodeField
{
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///交易所代码
    TSecurityFtdcExchangeIDType  ExchangeID;
    ///交易编码
    TSecurityFtdcClientIDType ClientID;
    ///是否活跃
    TSecurityFtdcBoolType     IsActive;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.17. 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(CSecurityFtdcExchangeField *pExchange,
                      CSecurityFtdcRspInfoField *pRspInfo,
                      int nRequestID,
                      bool bIsLast)  ;
```

参数:

pExchange: 指向交易所结构的地址。

交易所结构:

```
struct CSecurityFtdcExchangeField
{
    ///交易所代码
    TSecurityFtdcExchangeIDType ExchangeID;
    ///交易所名称
    TSecurityFtdcExchangeNameType    ExchangeName;
    ///交易所属性
    TSecurityFtdcExchangePropertyType       ExchangeProperty;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.18. 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(
                CSecurityFtdcInstrumentField *pInstrument,
                CSecurityFtdcRspInfoField *pRspInfo,
                int nRequestID, bool bIsLast)  ;
```

参数:

pInstrument: 指向合约结构的地址。 合约结构:

```
struct CSecurityFtdcInstrumentField
{
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///交易所代码
    TSecurityFtdcExchangeIDType ExchangeID;
    ///合约名称
    TSecurityFtdcInstrumentNameType   InstrumentName;
    ///合约在交易所的代码
    TSecurityFtdcExchangeInstIDType    ExchangeInstID;
    ///产品代码
    TSecurityFtdcInstrumentIDType    ProductID;
    ///产品类型
    TSecurityFtdcProductClassType ProductClass;
    ///交割年份
    TSecurityFtdcYearType     DeliveryYear;
    ///交割月
    TSecurityFtdcMonthType  DeliveryMonth;
    ///市价单最大下单量
    TSecurityFtdcVolumeType MaxMarketOrderVolume;
    ///市价单最小下单量
    TSecurityFtdcVolumeType MinMarketOrderVolume;
    ///限价单最大下单量
    TSecurityFtdcVolumeType MaxLimitOrderVolume;
    ///限价单最小下单量
    TSecurityFtdcVolumeType MinLimitOrderVolume;
    ///合约数量乘数
    TSecurityFtdcVolumeMultipleType   VolumeMultiple;
    ///最小变动价位
    TSecurityFtdcPriceType    PriceTick;
    ///创建日
    TSecurityFtdcDateType     CreateDate;
    ///上市日
    TSecurityFtdcDateType     OpenDate;
    ///到期日
    TSecurityFtdcDateType     ExpireDate;
    ///开始交割日
    TSecurityFtdcDateType     StartDelivDate;
    ///结束交割日
    TSecurityFtdcDateType     EndDelivDate;
    ///合约生命周期状态
    TSecurityFtdcInstLifePhaseType   InstLifePhase;
```

```
    ///当前是否交易
    TSecurityFtdcBoolType    IsTrading;
    ///持仓类型
    TSecurityFtdcPositionTypeType PositionType;
    ///持仓日期类型
    TSecurityFtdcPositionDateTypeType PositionDateType;
    ///多头保证金率
    TSecurityFtdcRatioType    LongMarginRatio;
    ///空头保证金率
    TSecurityFtdcRatioType    ShortMarginRatio;
};
```

pRspInfo: 指向响应信息结构的地址。

响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.19. 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(
                CSecurityFtdcDepthMarketDataField *pDepthMarketData,
                CSecurityFtdcRspInfoField *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 CSecurityFtdcDepthMarketDataField
{
    ///交易日
    TSecurityFtdcDateType    TradingDay;
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///交易所代码
    TSecurityFtdcExchangeIDType ExchangeID;
    ///合约在交易所的代码
    TSecurityFtdcExchangeInstIDType    ExchangeInstID;
```

```
///最新价
TSecurityFtdcPriceType    LastPrice;
///上次结算价
TSecurityFtdcPriceType    PreSettlementPrice;
///昨收盘
TSecurityFtdcPriceType    PreClosePrice;
///昨持仓量
TSecurityFtdcLargeVolumeType PreOpenInterest;
///今开盘
TSecurityFtdcPriceType    OpenPrice;
///最高价
TSecurityFtdcPriceType    HighestPrice;
///最低价
TSecurityFtdcPriceType    LowestPrice;
///数量
TSecurityFtdcVolumeType Volume;
///成交金额
TSecurityFtdcMoneyType  Turnover;
///持仓量
TSecurityFtdcLargeVolumeType OpenInterest;
///今收盘
TSecurityFtdcPriceType    ClosePrice;
///本次结算价
TSecurityFtdcPriceType    SettlementPrice;
///涨停板价
TSecurityFtdcPriceType    UpperLimitPrice;
///跌停板价
TSecurityFtdcPriceType    LowerLimitPrice;
///昨虚实度
TSecurityFtdcRatioType    PreDelta;
///今虚实度
TSecurityFtdcRatioType    CurrDelta;
///最后修改时间
TSecurityFtdcTimeType     UpdateTime;
///最后修改毫秒
TSecurityFtdcMillisecType UpdateMillisec;
///申买价一
TSecurityFtdcPriceType    BidPrice1;
///申买量一
TSecurityFtdcVolumeType BidVolume1;
///申卖价一
TSecurityFtdcPriceType    AskPrice1;
///申卖量一
TSecurityFtdcVolumeType AskVolume1;
///申买价二
TSecurityFtdcPriceType    BidPrice2;
///申买量二
TSecurityFtdcVolumeType BidVolume2;
///申卖价二
TSecurityFtdcPriceType    AskPrice2;
///申卖量二
TSecurityFtdcVolumeType AskVolume2;
```

```
    ///申买价三
    TSecurityFtdcPriceType    BidPrice3;
    ///申买量三
    TSecurityFtdcVolumeType BidVolume3;
    ///申卖价三
    TSecurityFtdcPriceType    AskPrice3;
    ///申卖量三
    TSecurityFtdcVolumeType AskVolume3;
    ///申买价四
    TSecurityFtdcPriceType    BidPrice4;
    ///申买量四
    TSecurityFtdcVolumeType BidVolume4;
    ///申卖价四
    TSecurityFtdcPriceType    AskPrice4;
    ///申卖量四
    TSecurityFtdcVolumeType AskVolume4;
    ///申买价五
    TSecurityFtdcPriceType    BidPrice5;
    ///申买量五
    TSecurityFtdcVolumeType BidVolume5;
    ///申卖价五
    TSecurityFtdcPriceType    AskPrice5;
    ///申卖量五
    TSecurityFtdcVolumeType AskVolume5;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.20. 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(
                    CSecurityFtdcInvestorPositionDetailField
                    *pInvestorPositionDetail, CSecurityFtdcRspInfoField
```

```
                    *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 CSecurityFtdcInvestorPositionDetailField
{
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///投机套保标志
    TSecurityFtdcHedgeFlagType       HedgeFlag;
    ///买卖方向
    TSecurityFtdcDirectionType       Direction;
    ///开仓日期
    TSecurityFtdcDateType     OpenDate;
    ///成交编号
    TSecurityFtdcTradeIDType TradeID;
    ///数量
    TSecurityFtdcVolumeType Volume;
    ///开仓价
    TSecurityFtdcPriceType    OpenPrice;
    ///交易日
    TSecurityFtdcDateType     TradingDay;
    ///结算编号
    TSecurityFtdcSettlementIDTypeSettlementID;
    ///成交类型
    TSecurityFtdcTradeTypeType       TradeType;
    ///组合合约代码
    TSecurityFtdcInstrumentIDType    CombInstrumentID;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.21. 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(
                        CSecurityFtdcInstrumentField
                        *pInstrument, CSecurityFtdcRspInfoField
                        *pRspInfo,
                        int nRequestID,
                        bool bIsLast);
```

参数:

pRspInstrument: 指向合约结构的地址。 合约结构:

```
struct CSecurityFtdcInstrumentField
{
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///交易所代码
    TSecurityFtdcExchangeIDType  ExchangeID;
    ///合约名称
    TSecurityFtdcInstrumentNameType   InstrumentName;
    ///合约在交易所的代码
    TSecurityFtdcExchangeInstIDType    ExchangeInstID;
    ///产品代码
    TSecurityFtdcInstrumentIDType    ProductID;
    ///产品类型
    TSecurityFtdcProductClassType ProductClass;
    ///交割年份
    TSecurityFtdcYearType     DeliveryYear;
    ///交割月
    TSecurityFtdcMonthType   DeliveryMonth;
    ///市价单最大下单量
    TSecurityFtdcVolumeType MaxMarketOrderVolume;
    ///市价单最小下单量
    TSecurityFtdcVolumeType MinMarketOrderVolume;
    ///限价单最大下单量
    TSecurityFtdcVolumeType MaxLimitOrderVolume;
    ///限价单最小下单量
    TSecurityFtdcVolumeType MinLimitOrderVolume;
    ///合约数量乘数
    TSecurityFtdcVolumeMultipleType   VolumeMultiple;
```

```
    ///最小变动价位
    TSecurityFtdcPriceType    PriceTick;
    ///创建日
    TSecurityFtdcDateType     CreateDate;
    ///上市日
    TSecurityFtdcDateType     OpenDate;
    ///到期日
    TSecurityFtdcDateType     ExpireDate;
    ///开始交割日
    TSecurityFtdcDateType     StartDelivDate;
    ///结束交割日
    TSecurityFtdcDateType     EndDelivDate;
    ///合约生命周期状态
    TSecurityFtdcInstLifePhaseType   InstLifePhase;
    ///当前是否交易
    TSecurityFtdcBoolType     IsTrading;
    ///持仓类型
    TSecurityFtdcPositionTypeType PositionType;
    ///持仓日期类型
    TSecurityFtdcPositionDateTypeType PositionDateType;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType 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。

6.4.22. 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(CSecurityFtdcTradeField *pTrade);
```

参数:

pTrade: 指向成交信息结构的地址。 成交信息结构:

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

```
TSecurityFtdcBrokerIDType BrokerID;
///投资者代码
TSecurityFtdcInvestorIDType      InvestorID;
///合约代码
TSecurityFtdcInstrumentIDType    InstrumentID;
///报单引用
TSecurityFtdcOrderRefType OrderRef;
///用户代码
TSecurityFtdcUserIDType  UserID;
///交易所代码
TSecurityFtdcExchangeIDType  ExchangeID;
///成交编号
TSecurityFtdcTradeIDType TradeID;
///买卖方向
TSecurityFtdcDirectionType       Direction;
///报单编号
TSecurityFtdcOrderSysIDType  OrderSysID;
///会员代码
TSecurityFtdcParticipantIDType   ParticipantID;
///客户代码
TSecurityFtdcClientIDType ClientID;
///交易角色
TSecurityFtdcTradingRoleType  TradingRole;
///合约在交易所的代码
TSecurityFtdcExchangeInstIDType    ExchangeInstID;
///开平标志
TSecurityFtdcOffsetFlagType      OffsetFlag;
///投机套保标志
TSecurityFtdcHedgeFlagType       HedgeFlag;
///价格
TSecurityFtdcPriceType    Price;
///数量
TSecurityFtdcVolumeType Volume;
///成交时期
TSecurityFtdcDateType     TradeDate;
///成交时间
TSecurityFtdcTimeType     TradeTime;
///成交类型
TSecurityFtdcTradeTypeType       TradeType;
///成交价来源
TSecurityFtdcPriceSourceType  PriceSource;
///交易所交易员代码
TSecurityFtdcTraderIDType TraderID;
```

```
    ///本地报单编号
    TSecurityFtdcOrderLocalIDType    OrderLocalID;
    ///结算会员编号
    TSecurityFtdcParticipantIDType   ClearingPartID;
    ///业务单元
    TSecurityFtdcBusinessUnitTypeBusinessUnit;
    ///序号
    TSecurityFtdcSequenceNoType SequenceNo;
    ///交易日
    TSecurityFtdcDateType     TradingDay;
    ///结算编号
    TSecurityFtdcSettlementIDTypeSettlementID;
};
```

6.4.23. 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(CSecurityFtdcOrderField *pOrder);
```

参数:

pOrder: 指向报单信息结构的地址。

报单信息结构:

```
struct CSecurityFtdcOrderField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///报单引用
    TSecurityFtdcOrderRefType OrderRef;
    ///用户代码
    TSecurityFtdcUserIDType UserID;
    ///报单价格条件
    TSecurityFtdcOrderPriceTypeType  OrderPriceType;
    ///买卖方向
    TSecurityFtdcDirectionType       Direction;
    ///组合开平标志
    TSecurityFtdcCombOffsetFlagType   CombOffsetFlag;
    ///组合投机套保标志
    TSecurityFtdcCombHedgeFlagType  CombHedgeFlag;
    ///价格
```

```
TSecurityFtdcPriceType    LimitPrice;
///数量
TSecurityFtdcVolumeType VolumeTotalOriginal;
///有效期类型
TSecurityFtdcTimeConditionType   TimeCondition;
///GTD 日期
TSecurityFtdcDateType     GTDDate;
///成交量类型
TSecurityFtdcVolumeConditionType VolumeCondition;
///最小成交量
TSecurityFtdcVolumeType MinVolume;
///触发条件
TSecurityFtdcContingentConditionType    ContingentCondition;
///止损价
TSecurityFtdcPriceType    StopPrice;
///强平原因
TSecurityFtdcForceCloseReasonType       ForceCloseReason;
///自动挂起标志
TSecurityFtdcBoolType     IsAutoSuspend;
///业务单元
TSecurityFtdcBusinessUnitTypeBusinessUnit;
///请求编号
TSecurityFtdcRequestIDType       RequestID;
///本地报单编号
TSecurityFtdcOrderLocalIDType    OrderLocalID;
///交易所代码
TSecurityFtdcExchangeIDType ExchangeID;
///会员代码
TSecurityFtdcParticipantIDType   ParticipantID;
///客户代码
TSecurityFtdcClientIDType ClientID;
///合约在交易所的代码
TSecurityFtdcExchangeInstIDType    ExchangeInstID;
///交易所交易员代码
TSecurityFtdcTraderIDType TraderID;
///安装编号
TSecurityFtdcInstallIDType       InstallID;
///报单提交状态
TSecurityFtdcOrderSubmitStatusType      OrderSubmitStatus;
///报单提示序号
TSecurityFtdcSequenceNoType NotifySequence;
///交易日
TSecurityFtdcDateType     TradingDay;
```

```
    ///结算编号
    TSecurityFtdcSettlementIDTypeSettlementID;
    ///报单编号
    TSecurityFtdcOrderSysIDType  OrderSysID;
    ///报单来源
    TSecurityFtdcOrderSourceType OrderSource;
    ///报单状态
    TSecurityFtdcOrderStatusType  OrderStatus;
    ///报单类型
    TSecurityFtdcOrderTypeType       OrderType;
    ///今成交数量
    TSecurityFtdcVolumeType VolumeTraded;
    ///剩余数量
    TSecurityFtdcVolumeType VolumeTotal;
    ///报单日期
    TSecurityFtdcDateType     InsertDate;
    ///插入时间
    TSecurityFtdcTimeType     InsertTime;
    ///激活时间
    TSecurityFtdcTimeType     ActiveTime;
    ///挂起时间
    TSecurityFtdcTimeType     SuspendTime;
    ///最后修改时间
    TSecurityFtdcTimeType     UpdateTime;
    ///撤销时间
    TSecurityFtdcTimeType     CancelTime;
    ///最后修改交易所交易员代码
    TSecurityFtdcTraderIDType ActiveTraderID;
    ///结算会员编号
    TSecurityFtdcParticipantIDType  ClearingPartID;
    ///序号
    TSecurityFtdcSequenceNoType SequenceNo;
    ///前置编号
    TSecurityFtdcFrontIDType FrontID;
    ///会话编号
    TSecurityFtdcSessionIDType       SessionID;
    ///用户端产品信息
    TSecurityFtdcProductInfoType  UserProductInfo;
    ///状态信息
    TSecurityFtdcErrorMsgType StatusMsg;
};
```

6.4.24. 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(
                CSecurityFtdcInputOrderField *pInputOrder,
                CSecurityFtdcRspInfoField *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 CSecurityFtdcInputOrderField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///报单引用
    TSecurityFtdcOrderRefType OrderRef;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///报单价格条件
    TSecurityFtdcOrderPriceTypeType  OrderPriceType;
    ///买卖方向
    TSecurityFtdcDirectionType       Direction;
    ///组合开平标志
    TSecurityFtdcCombOffsetFlagType   CombOffsetFlag;
    ///组合投机套保标志
    TSecurityFtdcCombHedgeFlagType  CombHedgeFlag;
    ///价格
    TSecurityFtdcPriceType    LimitPrice;
    ///数量
    TSecurityFtdcVolumeType VolumeTotalOriginal;
    ///有效期类型
    TSecurityFtdcTimeConditionType   TimeCondition;
    ///GTD 日期
    TSecurityFtdcDateType     GTDDate;
    ///成交量类型
```

```
    TSecurityFtdcVolumeConditionType VolumeCondition;
    ///最小成交量
    TSecurityFtdcVolumeType MinVolume;
    ///触发条件
    TSecurityFtdcContingentConditionType    ContingentCondition;
    ///止损价
    TSecurityFtdcPriceType    StopPrice;
    ///强平原因
    TSecurityFtdcForceCloseReasonType       ForceCloseReason;
    ///自动挂起标志
    TSecurityFtdcBoolType     IsAutoSuspend;
    ///业务单元
    TSecurityFtdcBusinessUnitTypeBusinessUnit;
    ///请求编号
    TSecurityFtdcRequestIDType       RequestID;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType ErrorMsg;
};
```

6.4.25. 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 (
                CSecurityFtdcOrderActionField *pOrderAction,
                CSecurityFtdcRspInfoField *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 CSecurityFtdcOrderActionField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
```

```
TSecurityFtdcInvestorIDType      InvestorID;
///报单操作引用
TSecurityFtdcOrderActionRefType    OrderActionRef;
///报单引用
TSecurityFtdcOrderRefType OrderRef;
///请求编号
TSecurityFtdcRequestIDType       RequestID;
///前置编号
TSecurityFtdcFrontIDType FrontID;
///会话编号
TSecurityFtdcSessionIDType       SessionID;
///交易所代码
TSecurityFtdcExchangeIDType  ExchangeID;
///报单编号
TSecurityFtdcOrderSysIDType  OrderSysID;
///操作标志
TSecurityFtdcActionFlagType    ActionFlag;
///价格
TSecurityFtdcPriceType    LimitPrice;
///数量变化
TSecurityFtdcVolumeType VolumeChange;
///操作日期
TSecurityFtdcDateType     ActionDate;
///操作时间
TSecurityFtdcTimeType     ActionTime;
///交易所交易员代码
TSecurityFtdcTraderIDType TraderID;
///安装编号
TSecurityFtdcInstallIDType       InstallID;
///本地报单编号
TSecurityFtdcOrderLocalIDType    OrderLocalID;
///操作本地编号
TSecurityFtdcOrderLocalIDType    ActionLocalID;
///会员代码
TSecurityFtdcParticipantIDType   ParticipantID;
///客户代码
TSecurityFtdcClientIDType ClientID;
///业务单元
TSecurityFtdcBusinessUnitTypeBusinessUnit;
///报单操作状态
TSecurityFtdcOrderActionStatusType     OrderActionStatus;
///用户代码
TSecurityFtdcUserIDType  UserID;
```

```
    ///状态信息
    TSecurityFtdcErrorMsgType StatusMsg;
};
```

pRspInfo: 指向响应信息结构的地址。 响应信息结构:

```
struct CSecurityFtdcRspInfoField
{
    ///错误代码
    TSecurityFtdcErrorIDType ErrorID;
    ///错误信息
    TSecurityFtdcErrorMsgType ErrorMsg;
};
```

6.5.CSecurityFtdcTraderApi 接口

CSecurityFtdcTraderApi 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、合约交易状态查询、交易所公告查询等功能。

6.5.1. CreateFtdcTraderApi 方法

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

函数原形:

```
static CSecurityFtdcTradeApi *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 CSecurityFtdcTradeApi 实例的指针。

6.5.2. Release 方法

释放一个 CSecurityFtdcTradeApi 实例。不能使用 delete 方法

函数原形:

```
void Release();
```

6.5.3. 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();
```

6.5.4. 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();
```

6.5.5. 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。

6.5.6. RegisterSpi 方法

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

函数原形:

```
void RegisterSpi(CSecurityFtdcTraderSpi *pSpi) ;
```

参数:

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

6.5.7. 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。

6.5.8. 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

6.5.9. 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

6.5.10. ReqUserLogin 方法

用户发出登陆请求。

函数原形:

```
int ReqUserLogin(
                CSecurityFtdcReqUserLoginField
                *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 CSecurityFtdcReqUserLoginField
{
    ///交易日
    TSecurityFtdcDateType     TradingDay;
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///密码
    TSecurityFtdcPasswordType Password;
    ///用户端产品信息
```

```
    TSecurityFtdcProductInfoType  UserProductInfo;
    ///接口端产品信息
    TSecurityFtdcProductInfoType  InterfaceProductInfo;
    ///协议信息
    TSecurityFtdcProtocolInfoType 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。

6.5.11. ReqUserLogout 方法

用户发出登出请求。

函数原形:

```
int ReqUserLogout(
                CSecurityFtdcUserLogoutField *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 CSecurityFtdcUserLogoutField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///用户代码
    TSecurityFtdcUserIDType  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。

6.5.12. ReqUserPasswordUpdate 方法

用户密码修改请求。

函数原形:

```
int ReqUserPasswordUpdate(
        CSecurityFtdcUserPasswordUpdateField *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 CSecurityFtdcUserPasswordUpdateField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///原来的口令
    TSecurityFtdcPasswordType OldPassword;
    ///新的口令
    TSecurityFtdcPasswordType 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。

6.5.13. ReqTradingAccountPasswordUpdate 方法

资金账户口令更新请求。

函数原形:

```
int ReqTradingAccountPasswordUpdate(
            CSecurityFtdcTradingAccountPasswordUpdateField
              *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 CSecurityFtdcTradingAccountPasswordUpdateField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者帐号
    TSecurityFtdcAccountIDType    AccountID;
    ///原来的口令
```

```
    TSecurityFtdcPasswordType OldPassword;
    ///新的口令
    TSecurityFtdcPasswordType 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。

6.5.14. ReqOrderInsert 方法

客户端发出报单录入请求。

函数原形:

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

参数:

pInputOrder: 指向输入报单结构的地址。

输入报单结构:

```
struct CSecurityFtdcInputOrderField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///报单引用
    TSecurityFtdcOrderRefType OrderRef;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///报单价格条件
    TSecurityFtdcOrderPriceTypeType  OrderPriceType;
    ///买卖方向
    TSecurityFtdcDirectionType       Direction;
    ///组合开平标志
    TSecurityFtdcCombOffsetFlagType   CombOffsetFlag;
    ///组合投机套保标志
    TSecurityFtdcCombHedgeFlagType   CombHedgeFlag;
    ///价格
```

```
    TSecurityFtdcPriceType    LimitPrice;
    ///数量
    TSecurityFtdcVolumeType VolumeTotalOriginal;
    ///有效期类型
    TSecurityFtdcTimeConditionType   TimeCondition;
    ///GTD 日期
    TSecurityFtdcDateType     GTDDate;
    ///成交量类型
    TSecurityFtdcVolumeConditionType VolumeCondition;
    ///最小成交量
    TSecurityFtdcVolumeType MinVolume;
    ///触发条件
    TSecurityFtdcContingentConditionType    ContingentCondition;
    ///止损价
    TSecurityFtdcPriceType    StopPrice;
    ///强平原因
    TSecurityFtdcForceCloseReasonType       ForceCloseReason;
    ///自动挂起标志
    TSecurityFtdcBoolType     IsAutoSuspend;
    ///业务单元
    TSecurityFtdcBusinessUnitTypeBusinessUnit;
    ///请求编号
    TSecurityFtdcRequestIDType       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CSecurityFtdcRspUserLoginField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TSecurityFtdcOrderRefType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。

6.5.15. 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(
```

```
                    CSecurityFtdcOrderActionField
                    *pOrderAction, int nRequestID);
```

参数:

pOrderAction: 指向报单操作结构的地址。

报单操作结构:

```
struct CSecurityFtdcOrderActionField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///报单操作引用
    TSecurityFtdcOrderActionRefType    OrderActionRef;
    ///报单引用
    TSecurityFtdcOrderRefType OrderRef;
    ///请求编号
    TSecurityFtdcRequestIDType       RequestID;
    ///前置编号
    TSecurityFtdcFrontIDType FrontID;
    ///会话编号
    TSecurityFtdcSessionIDType       SessionID;
    ///交易所代码
    TSecurityFtdcExchangeIDType  ExchangeID;
    ///报单编号
    TSecurityFtdcOrderSysIDType  OrderSysID;
    ///操作标志
    TSecurityFtdcActionFlagType    ActionFlag;
    ///价格
    TSecurityFtdcPriceType    LimitPrice;
    ///数量变化
    TSecurityFtdcVolumeType VolumeChange;
    ///操作日期
    TSecurityFtdcDateType     ActionDate;
    ///操作时间
    TSecurityFtdcTimeType     ActionTime;
    ///交易所交易员代码
    TSecurityFtdcTraderIDType TraderID;
    ///安装编号
    TSecurityFtdcInstallIDType       InstallID;
    ///本地报单编号
    TSecurityFtdcOrderLocalIDType    OrderLocalID;
    ///操作本地编号
    TSecurityFtdcOrderLocalIDType    ActionLocalID;
    ///会员代码
```

```
    TSecurityFtdcParticipantIDType  ParticipantID;
    ///客户代码
    TSecurityFtdcClientIDType ClientID;
    ///业务单元
    TSecurityFtdcBusinessUnitTypeBusinessUnit;
    ///报单操作状态
    TSecurityFtdcOrderActionStatusType     OrderActionStatus;
    ///用户代码
    TSecurityFtdcUserIDType  UserID;
    ///状态信息
    TSecurityFtdcErrorMsgType 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。

6.5.16 ReqQryOrder 方法

报单查询请求。

函数原形:

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

参数:

pQryOrder: 指向报单查询结构的地址。

报单查询结构:

```
struct CSecurityFtdcQryOrderField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType     InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType   InstrumentID;
    ///交易所代码
    TSecurityFtdcExchangeIDType ExchangeID;
    ///报单编号
    TSecurityFtdcOrderSysIDType  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。

注：不写 BrokerID 可以收全所有报单。

6.5.17. ReqQryTrade 方法

成交单查询请求。

函数原形:

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

参数:

pQryTrade: 指向成交查询结构的地址。 成交查询结构:

```
struct CSecurityFtdcQryTradeField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///交易所代码
    TSecurityFtdcExchangeIDType  ExchangeID;
    ///成交编号
    TSecurityFtdcTradeIDType 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。

6.5.18. ReqQry Investor 方法

会员客户查询请求。

函数原形:

```
int ReqQry Investor (
            CSecurityFtdcQryInvestorField
            *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 CSecurityFtdcQryInvestorField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      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。

6.5.19. ReqQryInvestorPosition 方法

会员持仓查询请求。

函数原形:

```
int ReqQryInvestorPosition(
                    CSecurityFtdcQryInvestorPositionField
                    *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 CSecurityFtdcQryInvestorPositionField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType    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。

6.5.20. ReqQryTradingAccount 方法

请求查询资金账户。

函数原形:

```
int ReqQryTradingAccount(
        CSecurityFtdcQryTradingAccountField *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 CSecurityFtdcQryTradingAccountField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      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。

6.5.21. ReqQryTradingCode 方法

请求查询交易编码。

函数原形:

```
int ReqQryTradingCode(
        CSecurityFtdcQryTradingCodeField
        *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 CSecurityFtdcQryTradingCodeField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///交易所代码
```

```
    TSecurityFtdcExchangeIDType ExchangeID;
    ///交易编码
    TSecurityFtdcClientIDType 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。

6.5.22. ReqQryExchange 方法

请求查询交易所。

函数原形:

```
int ReqQryExchange(
        CSecurityFtdcQryExchangeField *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 CSecurityFtdcQryExchangeField
{
    ///交易所代码
    TSecurityFtdcExchangeIDType 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。

6.5.23. ReqQryInstrument 方法

请求查询合约。

函数原形:

```
int ReqQryInstrument(
        CSecurityFtdcQryInstrumentField *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 CSecurityFtdcQryInstrumentField
{
    ///合约代码
    TSecurityFtdcInstrumentIDType    InstrumentID;
    ///交易所代码
    TSecurityFtdcExchangeIDType ExchangeID;
    ///合约在交易所的代码
    TSecurityFtdcExchangeInstIDType    ExchangeInstID;
    ///产品代码
    TSecurityFtdcInstrumentIDType    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。

6.5.24. ReqQryDepthMarketData 方法

请求查询行情。

函数原形:

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

参数:

pQryDepthMarketData: 指向查询行情结构的地址。

查询行情结构:

```
struct CSecurityFtdcQryDepthMarketDataField
{
    ///合约代码
    TSecurityFtdcInstrumentIDType    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。

6.5.25. ReqQryInvestorPositionDetail 方法

请求查询投资者持仓明细。

函数原形:

```
int ReqQryInvestorPositionDetail(
            CSecurityFtdcQryInvestorPositionDetailField
            *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 CSecurityFtdcQryInvestorPositionDetailField
{
    ///经纪公司代码
    TSecurityFtdcBrokerIDType BrokerID;
    ///投资者代码
    TSecurityFtdcInvestorIDType      InvestorID;
    ///合约代码
    TSecurityFtdcInstrumentIDType    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。