飞马平台
TraderAPI 说明书

文档标识

项目名称	FEMAS
文档名称	Trader API
版本号	1.00.00
状况	○ 草案 ◉ 评审过的 ○ 更新过的 ○ 定为基线的

文档修订历史

版本	日期	描述	修订者
V0.6	2013/03/04	删除报单操作中用户自定义字段；增加登录请求中 IP，MAC 占位字段字段	佘鹏飞
V0.7	2013/05/03	增加手续费率和保证金率查询接口	佘鹏飞
V0.8	2013/05/30	优化查询接口	佘鹏飞
V0.9	2013/06/18	实测版本	佘鹏飞
V1.0	2013/11/26	新增出入金回报接口	陈超

此版本文档的正式核准

姓名	签字	日期

目录

1. 介绍

飞马平台 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
FtdcTraderApi.h			交易接口头文件
FtdcUserApiDataType.h			定义了 API 所需的一系列数据类型的头文件
FtdcUserApiStruct.h			定义了一系列业务相关的数据结构的头文件
USTPtraderapi.dll			动态链接库二进制文件
USTPtraderapi.lib			导入库文件
USTPmduserapi.dll			动态链接库二进制文件
USTPmduserapi.lib			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 协议与飞马平台进行通讯，飞马平台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用户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，向某个特定的用户发出的信息。例如成交回报等。

广播通讯模式是指交易所端主动，向市场中的所有用户都发出相同的信息。

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：

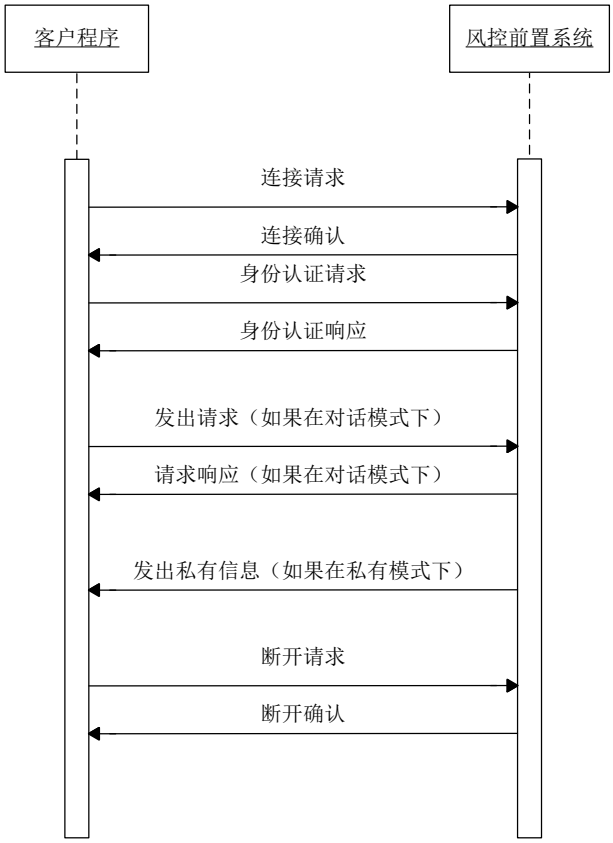

图 1-各种通讯模式的工作过程

本接口暂时没有使用广播通讯方式。

2.2 数据流

飞马平台支持对话通讯模式、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，飞马平台发送交易请求，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，飞马平台发送查询请求，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，由交易系统发向飞马平台，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，交易系统维护每个飞马平台的私有流，在一个交易日内，飞马平台断线后恢复连接时，可以请求交易系统发送指定序号之后的私有流数据。私有数据流向飞马平台提供报单状态报告、成交回报等信息。

3. 接口模式

交易员 API 提供了二个接口,分别为 CUSTPFtdcTraderApi 和 CUSTPFtdcTraderSpi。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 CUSTPFtdcTraderApi 发出操作请求，通继承 CUSTPFtdcTraderSpi 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 CUSTPFtdcTraderApi::ReqXXX(
            CUSTPFtdcXXXField *pReqXXX,
            int nRequestID)
响应：void CUSTPFtdcTraderSpi::OnRspXXX(
            CUSTPFtdcXXXField *pRspXXX,
            CUSTPFtdcRspInfoField *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。在接收飞马平台的响应时，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，CUSTPFtdcTraderSpi 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 CUSTPFtdcTraderSpi::OnRtnXXX(CUSTPFtdcXXXField *pXXX) 或
```

```
void CUSTPFtdcTraderSpi::OnErrRtnXXX(CUSTPFtdcXXXField *pXXX,
    CUSTPFtdcRspInfoField *pRspInfo)
```

当收到飞马平台通过私有流发布的回报数据时，CUSTPFtdcTraderSpi 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。

CUSTPFtdcTraderApi 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。

CUSTPFtdcTraderSpi 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，可以从飞马平台收取所需数据。

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 CUSTPFtdcTraderSpi 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	CUSTPFtdcTraderApi:: ReqUserLogin	CUSTPFtdcTraderSpi::OnRspUserLogin	对话流
	登出	CUSTPFtdcTraderApi::ReqUserLogout	CUSTPFtdcTraderSpi::OnRspUserLogout	对话流
	修改用户口令	CUSTPFtdcTraderApi::ReqUserPasswordUpdate	CUSTPFtdcTraderSpi::OnRspUserPasswordUpdate	对话流
报单	报单录入	CUSTPFtdcTraderApi::ReqOrderInsert	CUSTPFtdcTraderSpi::OnRspOrderInsert	对话流
	报单操作	CUSTPFtdcTraderApi::ReqOrderAction	CUSTPFtdcTraderSpi::OnRspOrderAction	对话流
私有回报	成交回报	N/A	CUSTPFtdcTraderSpi::OnRtnTrade	私有流
	报单回报	N/A	CUSTPFtdcTraderSpi::OnRtnOrder	私有流
	出入金回报	N/A	CUSTPFtdcTraderSpi::OnRtnInvestorAccountDeposit	私有流
	报单录入错误回报	N/A	CUSTPFtdcTraderSpi::OnErrRtnOrderInsert	私有流
	报单操作错误回报	N/A	CUSTPFtdcTraderSpi::OnErrRtnOrderAction	私有流
查询	报单查询	CUSTPFtdcTraderApi::ReqQryOrder	CUSTPFtdcTraderSpi::OnRspQryOrder	查询流

	成交查询	CUSTPFtdcTraderApi::ReqQryTrade	CUSTPFtdcTraderSpi::OnRspQryTrade	查询流
	合约查询	CUSTPFtdcTraderApi::ReqQryInstrument	CUSTPFtdcTraderSpi::OnRspQryInstrument	查询流
	可用投资者查询	CUSTPFtdcTraderApi::ReqQryUserInvestor	CUSTPFtdcTraderSpi::OnRspQryUserInvestor	查询流
	资金账户查询	CUSTPFtdcTraderApi::ReqQryInvestorAccount	CUSTPFtdcTraderSpi::OnRspQryInvestorAccount	查询流
	交易编码查询	CUSTPFtdcTraderApi::ReqQryTradingCode	CUSTPFtdcTraderSpi::OnRspQryTradingCode	查询流
	交易所查询	CUSTPFtdcTraderApi::ReqQryExchange	CUSTPFtdcTraderSpi::OnRspQryExchange	查询流
	投资者持仓查询	CUSTPFtdcTraderApi::ReqQryInvestorPosition	CUSTPFtdcTraderSpi::OnRspQryInvestorPosition	查询流
	合规参数查询	CUSTPFtdcTraderApi:: ReqQryComplianceParam	CUSTPFtdcTraderSpi:: OnRspQryComplianceParam	查询流
	手续费率查询	CUSTPFtdcTraderApi:: ReqQryInvestorFee	CUSTPFtdcTraderSpi:: OnRspQryInvestorFee	查询流
	保证金率查询	CUSTPFtdcTraderApi:: ReqQryInvestorMargin	CUSTPFtdcTraderSpi:: OnRspQryInvestorMargin	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。

6. 开发接口

6.1 通用规则

客户端和飞马平台的通讯过程分为 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 CUstpFtdcTraderApi 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, 设置飞马平台服务的地址

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。

6.2 CUstpFtdcTraderSpi 接口

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

6.2.1 OnFrontConnected 方法

当客户端与飞马平台建立起通信连接时（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。

6.2.2 OnFrontDisconnected 方法

当客户端与飞马平台通信连接断开时，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

6.2.3 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

6.2.4 OnRspUserLogin 方法

当客户端发出登录请求之后，飞马平台返回响应时，该方法会被调用，通知客户端登录是否成功。

函数原形：

```
void OnRspUserLogin(
     CUstpFtdcRspUserLoginField *pRspUserLogin,
     CUstpFtdcRspInfoField *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 CUstpFtdcRspUserLoginField
 {
     ///交易日
     TUstpFtdcDateType    TradingDay;
     ///经纪公司编号
```

```
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易用户代码
    TUstpFtdcUserIDType UserID;
    ///登录成功时间
    TUstpFtdcTimeType   LoginTime;
    ///最大本地报单号
    TUstpFtdcOrderLocalIDType   MaxOrderLocalID;
    ///交易系统名称
    TUstpFtdcTradingSystemNameType  TradingSystemName;
};
```

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 CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType    ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType   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。

6.2.5 OnRspUserLogout 方法

当客户端发出退出请求之后，飞马平台返回响应时，该方法会被调用，通知客户端退出是否成功。

函数原形：

```
void OnRspUserLogout(
       CUstpFtdcRspUserLogoutField *pRspUserLogout,
       CUstpFtdcRspInfoField *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 CUstpFtdcRspUserLogoutField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
   ///交易用户代码
    TUstpFtdcUserIDType UserID;
};
```

pRspInfo：返回用户响应信息的地址。
响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.6 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，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspUserPasswordUpdate(
       CUstpFtdcUserPasswordUpdateField *pUserPasswordUpdate,
      CUstpFtdcRspInfoField *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 CUstpFtdcUserPasswordUpdateField
{
     ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
    ///交易用户代码
     TUstpFtdcUserIDType UserID;
     ///旧密码
     TUstpFtdcPasswordType   OldPassword;
     ///新密码
     TUstpFtdcPasswordType   NewPassword;
};
```

pRspInfo：返回用户响应信息的地址。
响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
```

```
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.7 OnRspError 方法

针对用户请求的出错通知。

函数原形：

```
void OnRspError(
        CUstpFtdcRspInfoField * 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 CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.8 OnRspOrderInsert 方法

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

函数原形：

```
void OnRspOrderInsert(
        CUstpFtdcInputOrderField *pInputOrder,
        CUstpFtdcRspInfoField *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 CUstpFtdcInputOrderField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///系统报单编号
    TUstpFtdcOrderSysIDType OrderSysID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
    ///用户本地报单号
    TUstpFtdcUserOrderLocalIDType   UserOrderLocalID;
    ///报单类型
    TUstpFtdcOrderPriceTypeType OrderPriceType;
    ///买卖方向
    TUstpFtdcDirectionType  Direction;
    ///开平标志
    TUstpFtdcOffsetFlagType OffsetFlag;
    ///投机套保标志
    TUstpFtdcHedgeFlagType  HedgeFlag;
    ///价格
    TUstpFtdcPriceType  LimitPrice;
    ///数量
    TUstpFtdcVolumeType Volume;
    ///有效期类型
    TUstpFtdcTimeConditionType  TimeCondition;
    ///GTD 日期
    TUstpFtdcDateType   GTDDate;
    ///成交量类型
    TUstpFtdcVolumeConditionType    VolumeCondition;
    ///最小成交量
    TUstpFtdcVolumeType MinVolume;
    ///止损价
    TUstpFtdcPriceType  StopPrice;
    ///强平原因
    TUstpFtdcForceCloseReasonType   ForceCloseReason;
    ///自动挂起标志
    TUstpFtdcBoolType   IsAutoSuspend;
```

```
    ///业务单元
    TUstpFtdcBusinessUnitType   BusinessUnit;
    ///用户自定义域 64 字节
    TUstpFtdcCustomType UserCustom;
};
```

pRspInfo：返回用户响应信息的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.9 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，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspOrderAction(
      CUstpFtdcOrderActionField * pOrderAction,
      CUstpFtdcRspInfoField * 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 CUstpFtdcOrderActionField
{
        ///交易所代码
        TUstpFtdcExchangeIDType ExchangeID;
        ///交易所系统报单编号
        TUstpFtdcOrderSysIDType OrderSysID;
        ///经纪公司编号
        TUstpFtdcBrokerIDType   BrokerID;
        ///投资者编号
```

```
        TUstpFtdcInvestorIDType InvestorID;
        ///用户操作本地编号
        TUstpFtdcOrderLocalIDType   UserOrderActionLocalID;
        ///本地报单编号
        TUstpFtdcUserOrderLocalIDType   UserOrderLocalID;
        ///报单操作标志
        TUstpFtdcActionFlagType ActionFlag;
        ///价格 -保留域
        TUstpFtdcPriceType  LimitPrice;
        ///数量变化-保留域
        TUstpFtdcVolumeType VolumeChange;
};
```

pRspInfo：返回用户响应信息的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.10 OnRspQryOrder 方法

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

函数原形：

```
void OnRspQryOrder(
     CUstpFtdcOrderField  *pOrder,
     CUstpFtdcRspInfoField *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 CUstpFtdcOrderField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
```

```
///系统报单编号
TUstpFtdcOrderSysIDType OrderSysID;
///投资者编号
TUstpFtdcInvestorIDType InvestorID;
///用户代码
TUstpFtdcUserIDType UserID;
///合约代码
TUstpFtdcInstrumentIDType   InstrumentID;
///用户本地报单号
TUstpFtdcUserOrderLocalIDType   UserOrderLocalID;
///报单类型
TUstpFtdcOrderPriceTypeType OrderPriceType;
///买卖方向
TUstpFtdcDirectionType  Direction;
///开平标志
TUstpFtdcOffsetFlagType OffsetFlag;
///投机套保标志
TUstpFtdcHedgeFlagType  HedgeFlag;
///价格
TUstpFtdcPriceType  LimitPrice;
///数量
TUstpFtdcVolumeType Volume;
///有效期类型
TUstpFtdcTimeConditionType  TimeCondition;
///GTD 日期 保留字段
TUstpFtdcDateType   GTDDate;
///成交量类型
TUstpFtdcVolumeConditionType    VolumeCondition;
///最小成交量
TUstpFtdcVolumeType MinVolume;
///止损价 保留字段
TUstpFtdcPriceType  StopPrice;
///强平原因 保留字段
TUstpFtdcForceCloseReasonType   ForceCloseReason;
///自动挂起标志 保留字段
TUstpFtdcBoolType   IsAutoSuspend;
///业务单元 保留字段
TUstpFtdcBusinessUnitType   BusinessUnit;
///用户自定义域
TUstpFtdcCustomType UserCustom;
///交易日
TUstpFtdcTradingDayType TradingDay;
///会员编号
TUstpFtdcParticipantIDType  ParticipantID;
```

```
    ///客户号
    TUstpFtdcClientIDType   ClientID;
    ///下单席位号
    TUstpFtdcSeatIDType SeatID;
    ///插入时间
    TUstpFtdcTimeType   InsertTime;
    ///本地报单编号
    TUstpFtdcOrderLocalIDType   OrderLocalID;
    ///报单来源
    TUstpFtdcOrderSourceType    OrderSource;
    ///报单状态
    TUstpFtdcOrderStatusType    OrderStatus;
    ///撤销时间
    TUstpFtdcTimeType   CancelTime;
    ///撤单用户编号
    TUstpFtdcUserIDType CancelUserID;
    ///今成交数量
    TUstpFtdcVolumeType VolumeTraded;
    ///剩余数量
    TUstpFtdcVolumeType VolumeRemain;
};
```

pRspInfo：返回用户响应信息的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.11 OnRspQryTrade 方法

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

函数原形：

```
void OnRspQryTrade(
    CUstpFtdcTradeField * pTrade,
    CUstpFtdcRspInfoField *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 CUstpFtdcTradeField
{
        ///经纪公司编号
         TUstpFtdcBrokerIDType   BrokerID;
         ///交易所代码
         TUstpFtdcExchangeIDType ExchangeID;
         ///交易日
         TUstpFtdcTradingDayType TradingDay;
         ///会员编号
         TUstpFtdcParticipantIDType  ParticipantID;
         ///下单席位号
         TUstpFtdcSeatIDType SeatID;
         ///投资者编号
         TUstpFtdcInvestorIDType InvestorID;
         ///客户号
         TUstpFtdcClientIDType   ClientID;
         ///用户编号
         TUstpFtdcUserIDType UserID;
         ///成交编号
         TUstpFtdcTradeIDType    TradeID;
         ///本地报单编号
         TUstpFtdcOrderLocalIDType   UserOrderLocalID;
         ///合约代码
         TUstpFtdcInstrumentIDType   InstrumentID;
         ///买卖方向
         TUstpFtdcDirectionType  Direction;
         ///开平标志
         TUstpFtdcOffsetFlagType OffsetFlag;
         ///投机套保标志
         TUstpFtdcHedgeFlagType  HedgeFlag;
         ///成交价格
         TUstpFtdcTradePriceType TradePrice;
         ///成交数量
         TUstpFtdcTradeVolumeType    TradeVolume;
         ///成交时间
         TUstpFtdcTradeTimeType  TradeTime;
         ///清算会员编号
         TUstpFtdcParticipantIDType  ClearingPartID;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.12 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，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestorAccount(
    CUstpFtdcRspInvestorAccountField *pRspInvestorAccount,
    CUstpFtdcRspInfoField *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 CUstpFtdcRspInvestorAccountField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///资金帐号
    TUstpFtdcAccountIDType  AccountID;
    ///上次结算准备金
    TUstpFtdcMoneyType  PreBalance;
    ///入金金额
    TUstpFtdcMoneyType  Deposit;
    ///出金金额
    TUstpFtdcMoneyType  Withdraw;
    ///冻结的保证金
    TUstpFtdcMoneyType  FrozenMargin;
    ///冻结手续费
    TUstpFtdcMoneyType  FrozenFee;
    ///冻结权利金
```

```
    TUstpFtdcMoneyType  FrozenPremium;
    ///手续费
    TUstpFtdcMoneyType  Fee;
    ///平仓盈亏
    TUstpFtdcMoneyType  CloseProfit;
    ///持仓盈亏
    TUstpFtdcMoneyType  PositionProfit;
    ///可用资金
    TUstpFtdcMoneyType  Available;
    ///多头冻结的保证金
    TUstpFtdcMoneyType  LongFrozenMargin;
    ///空头冻结的保证金
    TUstpFtdcMoneyType  ShortFrozenMargin;
    ///多头占用保证金
    TUstpFtdcMoneyType  LongMargin;
    ///空头占用保证金
    TUstpFtdcMoneyType  ShortMargin;
    ///当日释放保证金
    TUstpFtdcMoneyType  ReleaseMargin;
    ///动态权益
    TUstpFtdcMoneyType  DynamicRights;
    ///今日出入金
    TUstpFtdcMoneyType  TodayInOut;
    ///占用保证金
    TUstpFtdcMoneyType  Margin;
    ///期权权利金收支
    TUstpFtdcMoneyType  Premium;
    ///风险度
    TUstpFtdcMoneyType  risk;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.13 OnRspQryTradingCode 方法

交易编码查询。当客户端发出交易编码查询指令后，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryTradingCode(
     CUstpFtdcRspTradingCodeField *pTradingCode,
     CUstpFtdcRspInfoField *pRspInfo,
     int nRequestID,
     bool bIsLast)
```

参数：

pTradingCode 指向交易编码查询返回信息的地址。

交易编码结构：

```
struct CUstpFtdcRspTradingCodeField
{
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
     ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
     ///投资者编号
     TUstpFtdcInvestorIDType InvestorID;
     ///客户代码
     TUstpFtdcClientIDType   ClientID;
     ///客户代码权限
     TUstpFtdcTradingRightType   ClientRight;
     ///是否活跃
     TUstpFtdcIsActiveType   IsActive;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
     ///错误代码
     TUstpFtdcErrorIDType ErrorID;
     ///错误信息
     TUstpFtdcErrorMsgType 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。

6.2.14 OnRspQryExchange 方法

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

函数原形：

```
void OnRspQryExchange(
     CUstpFtdcRspExchangeField *pExchange,
     CUstpFtdcRspInfoField *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 CUstpFtdcRspExchangeField
{
         ///交易所代码
         TUstpFtdcExchangeIDType ExchangeID;
         ///交易所名称
         TUstpFtdcExchangeNameType   ExchangeName;
         ///交易所状态
         TUstpFtdcExchangeStatusType ExchangeStatus;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
     ///错误代码
     TUstpFtdcErrorIDType ErrorID;
     ///错误信息
     TUstpFtdcErrorMsgType 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。

6.2.15 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，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryUstpInstrument(
     CUstpFtdcRspInstrumentField *pRspInstrument,
     CUstpFtdcRspInfoField *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 CUstpFtdcRspInstrumentField
{
        ///交易所代码
        TUstpFtdcExchangeIDType ExchangeID;
        ///品种代码
        TUstpFtdcProductIDType  ProductID;
        ///品种名称
        TUstpFtdcProductNameType    ProductName;
        ///合约代码
        TUstpFtdcInstrumentIDType   InstrumentID;
        ///合约名称
        TUstpFtdcInstrumentNameType InstrumentName;
        ///交割年份
        TUstpFtdcDeliveryYearType   DeliveryYear;
        ///交割月
        TUstpFtdcDeliveryMonthType  DeliveryMonth;
        ///限价单最大下单量
        TUstpFtdcMaxLimitOrderVolumeType    MaxLimitOrderVolume;
        ///限价单最小下单量
        TUstpFtdcMinLimitOrderVolumeType    MinLimitOrderVolume;
        ///市价单最大下单量
        TUstpFtdcMaxMarketOrderVolumeType   MaxMarketOrderVolume;
        ///市价单最小下单量
        TUstpFtdcMinMarketOrderVolumeType   MinMarketOrderVolume;
        ///数量乘数
        TUstpFtdcVolumeMultipleType VolumeMultiple;
        ///报价单位
        TUstpFtdcPriceTickType  PriceTick;
        ///币种
        TUstpFtdcCurrencyType   Currency;
        ///多头限仓
        TUstpFtdcLongPosLimitType   LongPosLimit;
        ///空头限仓
        TUstpFtdcShortPosLimitType  ShortPosLimit;
        ///跌停板价
        TUstpFtdcLowerLimitPriceType    LowerLimitPrice;
        ///涨停板价
        TUstpFtdcUpperLimitPriceType    UpperLimitPrice;
        ///昨结算
```

```
        TUstpFtdcPreSettlementPriceType PreSettlementPrice;
        ///合约交易状态
        TUstpFtdcInstrumentStatusType   InstrumentStatus;
        ///创建日
        TUstpFtdcDateType   CreateDate;
        ///上市日
        TUstpFtdcDateType   OpenDate;
        ///到期日
        TUstpFtdcDateType   ExpireDate;
        ///开始交割日
        TUstpFtdcDateType   StartDelivDate;
        ///最后交割日
        TUstpFtdcDateType   EndDelivDate;
        ///挂牌基准价
        TUstpFtdcPriceType  BasisPrice;
        ///当前是否交易
        TUstpFtdcBoolType   IsTrading;
        ///基础商品代码
        TUstpFtdcInstrumentIDType   UnderlyingInstrID;
        ///持仓类型
        TUstpFtdcPositionTypeType   PositionType;
        ///执行价
        TUstpFtdcPriceType  StrikePrice;
        ///期权类型
        TUstpFtdcOptionsTypeType    OptionsType;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.16 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，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryUserInvestor(
     CUstpFtdcUserInvestorField *pUserInvestor,
     CUstpFtdcRspInfoField *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 CUstpFtdcUserInvestorField
{
         ///经纪公司编号
         TUstpFtdcBrokerIDType   BrokerID;
         ///交易用户代码
         TUstpFtdcUserIDType UserID;
        ///投资者编号
         TUstpFtdcInvestorIDType InvestorID;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
     ///错误代码
     TUstpFtdcErrorIDType ErrorID;
     ///错误信息
     TUstpFtdcErrorMsgType 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。

6.2.17 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，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestorPosition (
     CUstpFtdcRspInvestorPositionField *pInvestorPosition,
     CUstpFtdcRspInfoField *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 CUstpFtdcRspInvestorPositionField
```

```
{
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///客户代码
    TUstpFtdcClientIDType   ClientID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
    ///买卖方向
    TUstpFtdcDirectionType  Direction;
    ///投机套保标志
    TUstpFtdcHedgeFlagType  HedgeFlag;
    ///占用保证金
    TUstpFtdcMoneyType  UsedMargin;
    ///今持仓量
    TUstpFtdcVolumeType Position;
    ///今日持仓成本
    TUstpFtdcPriceType  PositionCost;
    ///昨持仓量
    TUstpFtdcVolumeType YdPosition;
    ///昨日持仓成本
    TUstpFtdcMoneyType  YdPositionCost;
    ///冻结的保证金
    TUstpFtdcMoneyType  FrozenMargin;
    ///开仓冻结持仓
    TUstpFtdcVolumeType FrozenPosition;
    ///平仓冻结持仓
    TUstpFtdcVolumeType FrozenClosing;
    ///冻结的权利金
    TUstpFtdcMoneyType  FrozenPremium;
    ///最后一笔成交编号
    TUstpFtdcTradeIDType    LastTradeID;
    ///最后一笔本地报单编号
    TUstpFtdcOrderLocalIDType   LastOrderLocalID;
    ///币种
    TUstpFtdcCurrencyIDType Currency;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
```

```
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.18 OnRspQryComplianceParam 方法

合规参数查询应答。当客户端发出合规参数查询指令后，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryComplianceParam (
    CUstpFtdcRspComplianceParamField *pRspComplianceParam,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

参数：

pRspComplianceParam 指向合规参数结构的地址。
合规参数结构：

```
struct CUstpFtdcRspComplianceParamField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///客户号
    TUstpFtdcClientIDType   ClientID;
    ///每日最大报单笔
    TUstpFtdcVolumeType DailyMaxOrder;
    ///每日最大撤单笔
    TUstpFtdcVolumeType DailyMaxOrderAction;
    ///每日最大错单笔
    TUstpFtdcVolumeType DailyMaxErrorOrder;
    ///每日最大报单手
    TUstpFtdcVolumeType DailyMaxOrderVolume;
```

```
    ///每日最大撤单手
    TUstpFtdcVolumeType DailyMaxOrderActionVolume;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.19 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，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestorFee(
    CUstpFtdcInvestorFeeField *pInvestorFee,
    CUstpFtdcRspInfoField *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 CUstpFtdcInvestorFeeField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///客户号
    TUstpFtdcClientIDType   ClientID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
    ///品种代码
    TUstpFtdcProductIDType  ProductID;
    ///开仓手续费按比例
    TUstpFtdcRatioType  OpenFeeRate;
    ///开仓手续费按手数
```

```
    TUstpFtdcRatioType  OpenFeeAmt;
    ///平仓手续费按比例
    TUstpFtdcRatioType  OffsetFeeRate;
    ///平仓手续费按手数
    TUstpFtdcRatioType  OffsetFeeAmt;
    ///平今仓手续费按比例
    TUstpFtdcRatioType  OTFeeRate;
    ///平今仓手续费按手数
    TUstpFtdcRatioType  OTFeeAmt;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.20 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，飞马平台返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryInvestorMargin (
    CUstpFtdcInvestorMarginField *pInvestorMargin,
    CUstpFtdcRspInfoField *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 CUstpFtdcInvestorMarginField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///客户号
    TUstpFtdcClientIDType   ClientID;
    ///交易所代码
```

```
    TUstpFtdcExchangeIDType ExchangeID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
    ///品种代码
    TUstpFtdcProductIDType  ProductID;
    ///多头占用保证金按比例
    TUstpFtdcRatioType  LongMarginRate;
    ///多头保证金按手数
    TUstpFtdcRatioType  LongMarginAmt;
    ///空头占用保证金按比例
    TUstpFtdcRatioType  ShortMarginRate;
    ///空头保证金按手数
    TUstpFtdcRatioType  ShortMarginAmt;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType 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。

6.2.21 OnRtnTrade 方法

成交回报。当发生成交时飞马平台会通知客户端，该方法会被调用。

函数原形：

```
void OnRtnTrade(
    CUstpFtdcTradeField *pTrade);
```

参数：

pTrade：指向成交信息结构的地址。

成交信息结构：

```
struct CUstpFtdcTradeField
{
        ///经纪公司编号
        TUstpFtdcBrokerIDType   BrokerID;
        ///交易所代码
```

```
        TUstpFtdcExchangeIDType ExchangeID;
        ///交易日
        TUstpFtdcTradingDayType TradingDay;
        ///会员编号
        TUstpFtdcParticipantIDType  ParticipantID;
        ///下单席位号
        TUstpFtdcSeatIDType SeatID;
        ///投资者编号
        TUstpFtdcInvestorIDType InvestorID;
        ///客户号
        TUstpFtdcClientIDType   ClientID;
        ///用户编号
        TUstpFtdcUserIDType UserID;
        ///成交编号
        TUstpFtdcTradeIDType    TradeID;
        ///本地报单编号
        TUstpFtdcOrderLocalIDType   UserOrderLocalID;
        ///合约代码
        TUstpFtdcInstrumentIDType   InstrumentID;
        ///买卖方向
        TUstpFtdcDirectionType  Direction;
        ///开平标志
        TUstpFtdcOffsetFlagType OffsetFlag;
        ///投机套保标志
        TUstpFtdcHedgeFlagType  HedgeFlag;
        ///成交价格
        TUstpFtdcTradePriceType TradePrice;
        ///成交数量
        TUstpFtdcTradeVolumeType    TradeVolume;
        ///成交时间
        TUstpFtdcTradeTimeType  TradeTime;
        ///清算会员编号
        TUstpFtdcParticipantIDType  ClearingPartID;
};
```

6.2.22 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，飞马平台会主动通知客户端，该方法会被调用。

函数原形：

```
void OnRtnOrder(
    CUstpFtdcOrderField *pOrder);
```

参数：

pOrder：指向报单信息结构的地址。
报单信息结构：

```
struct CUstpFtdcOrderField
{
 ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
     ///系统报单编号
     TUstpFtdcOrderSysIDType OrderSysID;
     ///投资者编号
     TUstpFtdcInvestorIDType InvestorID;
     ///用户代码
     TUstpFtdcUserIDType UserID;
     ///合约代码
     TUstpFtdcInstrumentIDType   InstrumentID;
     ///用户本地报单号
     TUstpFtdcUserOrderLocalIDType   UserOrderLocalID;
     ///报单类型
     TUstpFtdcOrderPriceTypeType OrderPriceType;
     ///买卖方向
     TUstpFtdcDirectionType  Direction;
     ///开平标志
     TUstpFtdcOffsetFlagType OffsetFlag;
     ///投机套保标志
     TUstpFtdcHedgeFlagType  HedgeFlag;
     ///价格
     TUstpFtdcPriceType  LimitPrice;
     ///数量
     TUstpFtdcVolumeType Volume;
     ///有效期类型
     TUstpFtdcTimeConditionType  TimeCondition;
     ///GTD 日期 保留字段
     TUstpFtdcDateType   GTDDate;
     ///成交量类型
     TUstpFtdcVolumeConditionType    VolumeCondition;
     ///最小成交量
     TUstpFtdcVolumeType MinVolume;
     ///止损价 保留字段
     TUstpFtdcPriceType  StopPrice;
     ///强平原因 保留字段
     TUstpFtdcForceCloseReasonType   ForceCloseReason;
     ///自动挂起标志 保留字段
     TUstpFtdcBoolType   IsAutoSuspend;
```

```
    ///业务单元 保留字段
    TUstpFtdcBusinessUnitType   BusinessUnit;
    ///用户自定义域
    TUstpFtdcCustomType UserCustom;
    ///交易日
    TUstpFtdcTradingDayType TradingDay;
    ///会员编号
    TUstpFtdcParticipantIDType  ParticipantID;
    ///客户号
    TUstpFtdcClientIDType   ClientID;
    ///下单席位号
    TUstpFtdcSeatIDType SeatID;
    ///插入时间
    TUstpFtdcTimeType   InsertTime;
    ///本地报单编号
    TUstpFtdcOrderLocalIDType   OrderLocalID;
    ///报单来源
    TUstpFtdcOrderSourceType    OrderSource;
    ///报单状态
    TUstpFtdcOrderStatusType    OrderStatus;
    ///撤销时间
    TUstpFtdcTimeType   CancelTime;
    ///撤单用户编号
    TUstpFtdcUserIDType CancelUserID;
    ///今成交数量
    TUstpFtdcVolumeType VolumeTraded;
    ///剩余数量
    TUstpFtdcVolumeType VolumeRemain;
};
```

6.2.23 OnRtnInstrumentStatus 方法

合约交易状态通知。由飞马平台主动通知客户端，该方法会被调用。

函数原形：

```
void OnRtnInstrumentStatus(
     CUstpFtdcInstrumentStatusField *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 CUstpFtdcInstrumentStatusField
{
        ///交易所代码
        TUstpFtdcExchangeIDType ExchangeID;
```

```
///品种代码
TUstpFtdcProductIDType  ProductID;
///品种名称
TUstpFtdcProductNameType    ProductName;
///合约代码
TUstpFtdcInstrumentIDType   InstrumentID;
///合约名称
TUstpFtdcInstrumentNameType InstrumentName;
///交割年份
TUstpFtdcDeliveryYearType   DeliveryYear;
///交割月
TUstpFtdcDeliveryMonthType  DeliveryMonth;
///限价单最大下单量
TUstpFtdcMaxLimitOrderVolumeType    MaxLimitOrderVolume;
///限价单最小下单量
TUstpFtdcMinLimitOrderVolumeType    MinLimitOrderVolume;
///市价单最大下单量
TUstpFtdcMaxMarketOrderVolumeType   MaxMarketOrderVolume;
///市价单最小下单量
TUstpFtdcMinMarketOrderVolumeType   MinMarketOrderVolume;
///数量乘数
TUstpFtdcVolumeMultipleType VolumeMultiple;
///报价单位
TUstpFtdcPriceTickType  PriceTick;
///币种
TUstpFtdcCurrencyType   Currency;
///多头限仓
TUstpFtdcLongPosLimitType   LongPosLimit;
///空头限仓
TUstpFtdcShortPosLimitType  ShortPosLimit;
///跌停板价
TUstpFtdcLowerLimitPriceType    LowerLimitPrice;
///涨停板价
TUstpFtdcUpperLimitPriceType    UpperLimitPrice;
///昨结算
TUstpFtdcPreSettlementPriceType PreSettlementPrice;
///合约交易状态
TUstpFtdcInstrumentStatusType   InstrumentStatus;
///创建日
TUstpFtdcDateType   CreateDate;
///上市日
TUstpFtdcDateType   OpenDate;
///到期日
TUstpFtdcDateType   ExpireDate;
```

```
        ///开始交割日
        TUstpFtdcDateType   StartDelivDate;
        ///最后交割日
        TUstpFtdcDateType   EndDelivDate;
        ///挂牌基准价
        TUstpFtdcPriceType  BasisPrice;
        ///当前是否交易
        TUstpFtdcBoolType   IsTrading;
        ///基础商品代码
        TUstpFtdcInstrumentIDType   UnderlyingInstrID;
        ///持仓类型
        TUstpFtdcPositionTypeType   PositionType;
        ///执行价
        TUstpFtdcPriceType  StrikePrice;
        ///期权类型
        TUstpFtdcOptionsTypeType    OptionsType;
};
```

6.2.24 OnRtnInvestorAccountDeposit 方法

出入金结果回报。由飞马平台主动通知客户端，该方法会被调用。

函数原形：

```
void OnRtnInvestorAccountDeposit(
CUstpFtdcInvestorAccountDepositResField* 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 CUstpFtdcInvestorAccountDepositResField
{
 ///经纪公司编号
 TUstpFtdcBrokerIDType   BrokerID;
 ///用户代码
 TUstpFtdcUserIDType UserID;
 ///投资者编号
 TUstpFtdcInvestorIDType InvestorID;
 ///资金帐号
 TUstpFtdcAccountIDType  AccountID;
 ///资金流水号
 TUstpFtdcAccountSeqNoType   AccountSeqNo;
```

```
///金额
TUstpFtdcMoneyType  Amount;
///出入金方向
TUstpFtdcAccountDirectionType   AmountDirection;
///可用资金
TUstpFtdcMoneyType  Available;
///结算准备金
TUstpFtdcMoneyType  Balance;
};
```

6.2.25 OnErrRtnOrderInsert 方法

报单录入错误回报。由飞马平台主动通知客户端，该方法会被调用。

函数原形：

```
void OnErrRtnOrderInsert(
    CUstpFtdcInputOrderField *pInputOrder,
    CUstpFtdcRspInfoField  * 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 CUstpFtdcInputOrderField
{
        ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///系统报单编号
    TUstpFtdcOrderSysIDType OrderSysID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
    ///用户本地报单号
    TUstpFtdcUserOrderLocalIDType   UserOrderLocalID;
    ///报单类型
    TUstpFtdcOrderPriceTypeType OrderPriceType;
    ///买卖方向
```

```
    TUstpFtdcDirectionType  Direction;
    ///开平标志
    TUstpFtdcOffsetFlagType OffsetFlag;
    ///投机套保标志
    TUstpFtdcHedgeFlagType  HedgeFlag;
    ///价格
    TUstpFtdcPriceType  LimitPrice;
    ///数量
    TUstpFtdcVolumeType Volume;
    ///有效期类型
    TUstpFtdcTimeConditionType  TimeCondition;
    ///GTD 日期
    TUstpFtdcDateType   GTDDate;
    ///成交量类型
    TUstpFtdcVolumeConditionType    VolumeCondition;
    ///最小成交量
    TUstpFtdcVolumeType MinVolume;
    ///止损价
    TUstpFtdcPriceType  StopPrice;
    ///强平原因
    TUstpFtdcForceCloseReasonType   ForceCloseReason;
    ///自动挂起标志
    TUstpFtdcBoolType   IsAutoSuspend;
    ///业务单元
    TUstpFtdcBusinessUnitType   BusinessUnit;
    ///用户自定义域 64 字节
    TUstpFtdcCustomType UserCustom;
};
```

pRspInfo：返回用户响应信息的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
    ///错误代码
    TUstpFtdcErrorIDType ErrorID;
    ///错误信息
    TUstpFtdcErrorMsgType ErrorMsg;
};
```

6.2.26 OnErrRtnOrderAction 方法

报价操作错误回报。由飞马平台主动通知客户端，该方法会被调用。

函数原形：

```
void OnErrRtnOrderAction (
     CUstpFtdcOrderActionField *pOrderAction,
     CUstpFtdcRspInfoField *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 CUstpFtdcOrderActionField
{
         ///交易所代码
         TUstpFtdcExchangeIDType ExchangeID;
         ///交易所系统报单编号
         TUstpFtdcOrderSysIDType OrderSysID;
         ///经纪公司编号
         TUstpFtdcBrokerIDType   BrokerID;
         ///投资者编号
         TUstpFtdcInvestorIDType InvestorID;
         ///用户操作本地编号
         TUstpFtdcOrderLocalIDType   UserOrderActionLocalID;
         ///本地报单编号
         TUstpFtdcUserOrderLocalIDType   UserOrderLocalID;
         ///报单操作标志
         TUstpFtdcActionFlagType ActionFlag;
         ///价格 - 保留域
         TUstpFtdcPriceType  LimitPrice;
         ///数量变化-保留域
         TUstpFtdcVolumeType VolumeChange;
};
```

pRspInfo：返回用户响应信息的地址。

响应信息结构：

```
struct CUstpFtdcRspInfoField
{
     ///错误代码
     TUstpFtdcErrorIDType ErrorID;
     ///错误信息
     TUstpFtdcErrorMsgType ErrorMsg;
};
```

6.3 CUSTPFtdcTraderApi 接口

CUstpFtdcTraderApi 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。

6.3.1 CreateFtdcTraderApi 方法

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

函数原形：

```
static CUstpFtdcTradeApi *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飞马平台发布消息的状态。默认值代表当前目录。

返回值：

返回一个指向 CUstpFtdcTradeApi 实例的指针。

6.3.2 Release 方法

释放一个 CUstpFtdcTradeApi 实例。不能使用 delete 方法

函数原形：

```
void Release();
```

6.3.3 Init 方法

使客户端开始与飞马平台建立连接，连接成功后可以进行登陆。

函数原形：

```
void Init();
```

6.3.4 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();
```

6.3.5 GetTradingDay 方法

获得当前交易日。只有当与飞马平台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。

6.3.6 RegisterSpi 方法

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

函数原形：

void RegisterSpi(CUstpFtdcTraderSpi *pSpi) ;

参数：

pSpi：实现了 CUstpFtdcTraderSpi 接口的实例指针。

6.3.7 RegisterFront 方法

设置飞马平台的网络通讯地址，飞马平台拥有多个通信地址，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。

6.3.8 RegisterNameServer 方法

设置飞马 NameServer 的网络通讯地址，用于获取行情服务列表。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);
```

参数：

pszNsAddress：指向飞马服务端 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！

6.3.9 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(USTP_TE_RESUME_TYPE nResumeType);
```

参数：

nResumeType： 私有流重传方式
USTP_TERT_RESTART:从本交易日开始重传
USTP_TERT_RESUME:从上次收到的续传
USTP_TERT_QUICK:只传送登录后私有流的内容

6.3.10 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(USTP_TE_RESUME_TYPE nResumeType);
```

参数：

nResumeType： 公共流重传方式
USTP_TERT_RESTART:从本交易日开始重传
USTP_TERT_RESUME:从上次收到的续传
USTP_TERT_QUICK:只传送登录后公共流的内容

6.3.11 ReqUserLogin 方法

用户发出登陆请求。

函数原形：

```
int ReqUserLogin (
     CUstpFtdcReqUserLoginField *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 CUstpFtdcReqUserLoginField
{
        ///交易日，API 自动填写
         TUstpFtdcDateType    TradingDay;
         ///交易用户代码
```

```
        TUstpFtdcUserIDType UserID;
        ///经纪公司编号
        TUstpFtdcBrokerIDType   BrokerID;
        ///密码
        TUstpFtdcPasswordType   Password;
        ///用户端产品信息
        TUstpFtdcProductInfoType    UserProductInfo;
        ///接口端产品信息—占位字段，API 自动填写
        TUstpFtdcProductInfoType    InterfaceProductInfo;
        ///IP 地址—占位字段，API 自动填写
        TUstpFtdcIPAddressType  IPAddress;
        ///Mac 地址—占位字段，API 自动填写
        TUstpFtdcMacAddressType MacAddress;
        ///协议信息—占位字段，API 自动填写
        TUstpFtdcProtocolInfoType   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。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。

6.3.12 ReqUserLogout 方法

用户发出登出请求。

函数原形：

```
int ReqUserLogout (
     CUstpFtdcReqUserLogoutField * 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 CUstpFtdcReqUserLogoutField
{
        ///经纪公司编号
        TUstpFtdcBrokerIDType   BrokerID;
        ///交易用户代码
        TUstpFtdcUserIDType 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。

6.3.13 ReqUserPasswordUpdate 方法

用户密码修改请求。

函数原形：

```
int ReqUserPasswordUpdate (
     CUstpFtdcUserPasswordUpdateField *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 CUstpFtdcUserPasswordUpdateField
{
         ///经纪公司编号
         TUstpFtdcBrokerIDType   BrokerID;
        ///交易用户代码
         TUstpFtdcUserIDType UserID;
         ///旧密码
         TUstpFtdcPasswordType   OldPassword;
         ///新密码
         TUstpFtdcPasswordType   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。

6.3.14 ReqOrderInsert 方法

客户端发出报单录入请求。

函数原形：

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

参数：

pInputOrder：指向输入报单结构的地址。

输入报单结构：

```
struct CUstpFtdcInputOrderField
{
        ///经纪公司编号
        TUstpFtdcBrokerIDType   BrokerID;
        ///交易所代码
        TUstpFtdcExchangeIDType ExchangeID;
        ///系统报单编号
        TUstpFtdcOrderSysIDType OrderSysID;
        ///投资者编号
        TUstpFtdcInvestorIDType InvestorID;
        ///用户代码
        TUstpFtdcUserIDType UserID;
        ///合约代码
        TUstpFtdcInstrumentIDType   InstrumentID;
        ///用户本地报单号
        TUstpFtdcUserOrderLocalIDType   UserOrderLocalID;
        ///报单价格条件
        TUstpFtdc OrderPriceTypeType    OrderPriceType;
        ///买卖方向
        TUstpFtdcDirectionType  Direction;
        ///开平标志 只支持开仓和平仓，不支持强平、平昨、平今
        TUstpFtdcOffsetFlagType OffsetFlag;
        ///投机套保标志
        TUstpFtdcHedgeFlagType  HedgeFlag;
        ///价格
        TUstpFtdcLimitPriceType LimitPrice;
        ///数量
        TUstpFtdcVolumeType Volume;
        ///有效期类型,“立即完成，否则撤销”1 OR“当日有效”3 市价单只能‘1‘
        TUstpFtdcTimeConditionType  TimeCondition;
        ///GTD 日期-未使用
        TUstpFtdcDateType   GTDDate;
        ///成交量类型，只支持”任意数量”
        TUstpFtdcVolumeConditionType    VolumeCondition;
        ///最小成交量-未使用
        TUstpFtdcVolumeType MinVolume;
        ///止损价-未使用
        TUstpFtdcPriceType  StopPrice;
        ///强平原因，只支持”非强平”
        TUstpFtdcForceCloseReasonType   ForceCloseReason;
        ///自动挂起标志-未使用
```

```
        TUstpFtdcBoolType    IsAutoSuspend;
        ///业务单元-未使用
        TUstpFtdcBusinessUnitType    BusinessUnit;
        ///用户自定义域 64 字节
        TUstpFtdcCustomType UserCustom;
};
```

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。

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。

注意：

用户本地报单号 UserOrderLocalID 是一个 12 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。

6.3.15 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（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(
     CUstpFtdcOrderActionField *pOrderAction,
     int nRequestID);
```

参数：

pOrderAction：指向报单操作结构的地址。

报单操作结构：

```
struct CUstpFtdcOrderActionField
{
        ///交易所代码
        TUstpFtdcExchangeIDType ExchangeID;
        ///交易所系统报单编号，不为空按该字段操作，为空按本地报单编号操作
        TUstpFtdcOrderSysIDType OrderSysID;
        ///经纪公司编号
        TUstpFtdcBrokerIDType    BrokerID;
        ///投资者编号
        TUstpFtdcInvestorIDType InvestorID;
        ///用户代码
        TUstpFtdcUserIDType UserID;
        ///本次撤单 Req 的本地编号，需要保证按字典序递增
        TUstpFtdcUserOrderLocalIDType    UserOrderActionLocalID;
        ///被撤订单的本地报单编号
```

```
        TUstpFtdcUserOrderLocalIDType   UserOrderLocalID;
        ///报单操作标志
        TUstpFtdcActionFlagType ActionFlag;
        ///价格 -保留域
        TUstpFtdcPriceType  LimitPrice;
        ///数量变化-保留域
        TUstpFtdcVolumeType VolumeChange;
};
```

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。

6.3.16 ReqQryOrder 方法

报单查询请求。

函数原形：

```
int ReqQryUstpOrder (
     CUstpFtdcQryOrderField *pQryOrder,
     int nRequestID);
```

参数：

pQryOrder：指向报单查询结构的地址。

报单查询结构：

```
struct CUstpFtdcQryOrderField
{
     ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
     ///用户代码
     TUstpFtdcUserIDType UserID;
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
     ///投资者编号
     TUstpFtdcInvestorIDType InvestorID;
     ///报单编号
     TUstpFtdcOrderSysIDType OrderSysID;
     ///合约代码
     TUstpFtdcInstrumentIDType   InstrumentID;
};
```

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。

6.3.17 ReqQryTrade 方法

成交单查询请求。

函数原形：

```
int ReqQryUstpTrade (
     CUstpFtdcQryTradeField *pQryTrade,
     int nRequestID);
```

参数：

pQryTrade：指向成交查询结构的地址。
成交查询结构：

```
struct CUstpFtdcQryTradeField
{
     ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
     ///用户代码
     TUstpFtdcUserIDType UserID;
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
     ///投资者编号
     TUstpFtdcInvestorIDType InvestorID;
     ///成交编号
     TUstpFtdcTradeIDType    TradeID;
     ///合约代码
     TUstpFtdcInstrumentIDType   InstrumentID;
};
```

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。

6.3.18 ReqQryInvestorAccount 方法

投资者资金账户查询。

函数原形：

```
int ReqQryInvestorAccount(
     CUstpFtdcQryInvestorAccountField *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 CUstpFtdcQryInvestorAccountField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
};
```

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。

6.3.19 ReqQryTradingCode 方法

交易编码查询。

函数原形：

```
int ReqQryTradingCode (
    CUstpFtdcQryTradingCodeField *pQryTradingCode,
    int nRequestID)
```

参数：

pQryTradingCode 指向交易编码查询结构的地址。

交易编码结构：

```
struct CUstpFtdcQryTradingCodeField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
 };
```

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。

6.3.20 ReqQryExchange 方法

交易所查询。

函数原形：

```
int ReqQryExchange(
     CUstpFtdcQryExchangeField *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 CUstpFtdcQryExchangeField
{
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
};
```

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。

6.3.21 ReqQryInstrument 方法

合约信息查询。

函数原形：

```
int ReqQryInstrument (
     CUstpFtdcQryInstrumentField *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 CUstpFtdcQryInstrumentField
{
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
     ///产品代码
```

```
    TUstpFtdcProductIDType  ProductID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
};
```

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。

6.3.22 ReqQryUserInvestor 方法

可用投资者账户查询。

函数原形：

```
int ReqQryUserInvestor(
    CUstpFtdcQryUserInvestorField *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 CUstpFtdcQryUserInvestorField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
};
```

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。

6.3.23 ReqQryInvestorPosition 方法

投资者持仓查询。

函数原形：

```
void ReqQryInvestorPosition (
    CUstpFtdcQryInvestorPositionField *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 CUstpFtdcQryInvestorPositionField
{
     ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
     ///用户代码
     TUstpFtdcUserIDType UserID;
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
     ///投资者编号
     TUstpFtdcInvestorIDType InvestorID;
     ///合约代码
     TUstpFtdcInstrumentIDType   InstrumentID;
 };
```

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。

6.3.24 ReqQryComplianceParam 方法

合规参数查询。

函数原形：

```
void ReqQryComplianceParam (
     CUstpFtdcQryComplianceParamField *pQryComplianceParam,
     int nRequestID)
```

参数：

pQryComplianceParam 指向合规参数查询结构的地址。

合规参数查询结构

```
struct CUstpFtdcQryComplianceParamField
{
     ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
     ///用户代码
     TUstpFtdcUserIDType UserID;
     ///投资者编号
     TUstpFtdcInvestorIDType InvestorID;
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
```

};
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。

6.3.25 ReqQryInvestorFee 方法

投资者手续费率查询。

函数原形：

```
void ReqQryInvestorFee(
     CUstpFtdcQryInvestorFeeField *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 CUstpFtdcQryInvestorFeeField
{
     ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
     ///用户代码
     TUstpFtdcUserIDType UserID;
     ///投资者编号
     TUstpFtdcInvestorIDType InvestorID;
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
     ///合约代码
     TUstpFtdcInstrumentIDType   InstrumentID;
};
```

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。

6.3.26 ReqQryInvestorMargin 方法

投资者保证金率查询

函数原形：

```
void ReqQryInvestorMargin (
     CUstpFtdcQryInvestorMarginField *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 CUstpFtdcQryInvestorMarginField
{
     ///经纪公司编号
     TUstpFtdcBrokerIDType   BrokerID;
     ///用户代码
     TUstpFtdcUserIDType UserID;
     ///投资者编号
     TUstpFtdcInvestorIDType InvestorID;
     ///交易所代码
     TUstpFtdcExchangeIDType ExchangeID;
     ///合约代码
     TUstpFtdcInstrumentIDType   InstrumentID;
 };
```

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。

7. 开发实例

```
// testtradeapi.cpp :
// 一个简单的例子，介绍 CUstpFtdcTraderApi 和 CUstpFtdcTraderSpi 接口的使用。
// 本例将演示一个报单录入操作的过程
#include <stdio.h>
#include <windows.h>
#include "USTPFtdcTraderApi.h"
// 报单录入操作是否完成的标志
// Create a manual reset event with no signal
HANDLE g_hEvent = CreateEvent(NULL, true, false, NULL);
// 经纪公司代码
 TUstpFtdcBrokerIDType   g_ch BrokerID;
// 交易用户代码
TUstpFtdcUserIDType g_chUserID;
//用户本地最大报单号
TUstpFtdcUserOrderLocalIDType g_UserOrderLocalID;
class CSimpleHandler : public CUstpFtdcTraderSpi
{
     public:
     // 构造函数，需要一个有效的指向 CUstpFtdcMduserApi 实例的指针
     CSimpleHandler(CUstpFtdcTraderApi *pUserApi) :
     m_pUserApi (pUserApi) {}
     ~CSimpleHandler () {}
     // 当客户端与飞马平台建立起通信连接，客户端需要进行登录
     virtual void OnFrontConnected()
     {
         CUstpFtdcReqUserLoginField reqUserLogin;
         // get BrokerID
         printf("BrokerID:");
         scanf("%s", &g_chBrokerID);
         strcpy(reqUserLogin. BrokerID, g_chBrokerID);
         // get userid
         printf("userid:");
         scanf("%s", &g_chUserID);
         strcpy(reqUserLogin.UserID, g_chUserID);
         // get password
         printf("password:");
         scanf("%s", &reqUserLogin.Password);
         // 发出登陆请求
         m_pUserApi->ReqUserLogin(&reqUserLogin, 0);
     }
```

```
// 当客户端与飞马平台通信连接断开时，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(
                CUstpFtdcRspUserLoginField *pRspUserLogin,
                CUstpFtdcRspInfoField  *pRspInfo,
                int nRequestID,
                bool bIsLast)
  {
      printf("OnRspUserLogin:\n");
      printf("ErrorCode=[%d], ErrorMsg=[%s]\n",
      pRspInfo->ErrorID, pRspInfo->ErrorMsg);
      printf("RequestID=[%d], Chain=[%d]\n", nRequestID, bIsLast);
      if (pRspInfo->ErrorID != 0)
      {
          // 端登失败，客户端需进行错误处理
          printf("Failed to login, errorcode=%d errormsg=%s
              requestid=%d chain=%d",
             pRspInfo->ErrorID, pRspInfo->ErrorMsg,
              nRequestID, bIsLast);
          exit(-1);
      }
  //用户最大本地报单号
      strcpy(g_UserOrderLocalID ,pRspUserLogin->MaxOrderLocalID);
  // 端登成功,发出报单录入请求
      CUstpFtdcInputOrderField ord;
      memset(&ord, 0, sizeof(ord));
  //经纪公司代码
      strcpy(ord.BrokerID, g_chBrokerID);
  // 合约代码
      strcpy(ord.InstrumentID, "IF1306");
  //投资者代码
      strcpy(ord.InvestorID , "000101");
  // 用户代码
      strcpy(ord.UserID, g_chUserID);
  // 买卖方向
      strcpy(ord.Direction , "2");
  // 开平标志
      strcpy(ord.OffsetFlag, "0");
  // 投机套保标志
```

```
    strcpy(ord.HedgeFlag, "1");
// 价格
    ord.LimitPrice = 50000;
// 数量
    ord.VolumeTotalOriginal = 10;
// 有效期类型
    strcpy(ord.TimeCondition,"1" );
// 自动挂起标志
    ord.IsAutoSuspend = 0;
//交易所
    strcpy(ord.ExchangeID,"CFFEX");
//本地报单号
    strcpy(ord.OrderLocalID, g_UserOrderLocalID);
    m_pUserApi->ReqOrderInsert(&ord, 1);
}
// 报单录入应答
virtual void OnRspOrderInsert(
              CUstpFtdcInputOrderField  *pInputOrder,
               CUstpFtdcRspInfoField  *pRspInfo,
               int nRequestID,
              bool bIsLast)
{
// 输出报单录入结果
    printf("ErrorCode=[%d], ErrorMsg=[%s]\n",
    pRspInfo->ErrorID, pRspInfo->ErrorMsg);
// 通知报单录入完成
    SetEvent(g_hEvent);
};
///报单回报
virtual void OnRtnOrder(CUstpFtdcOrderField  *pOrder)
{
    printf("OnRtnOrder:\n");
    printf("OrderSysID=[%s]\n", pOrder->OrderSysID);
}
// 针对用户请求的出错通知
virtual void OnRspError(
             CUstpFtdcRspInfoField  *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
        }
    }
private:
    // 指向 CUstpFtdcMduserApi 实例的指针
    CUstpFtdcTraderApi *m_pUserApi;
};
int main()
{
    // 产生一个 CUstpFtdcTraderApi 实例
        CUstpFtdcTraderApi *pUserApi =
        CUstpFtdcTraderApi::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
    // USTP_TERT_RESTART:从本交易日开始重传
    // USTP_TERT_RESUME:从上次收到的续传
    // USTP_TERT_QUICK:只传送登录后私有流的内容
        pUserApi->SubscribePrivateTopic(USTP_TERT_RESUME);
    // 订阅公共流
    // USTP_TERT_RESTART:从本交易日开始重传
    // USTP_TERT_RESUME:从上次收到的续传
    // USTP_TERT_QUICK:只传送登录后公共流的内容
        pUserApi->SubscribePublicTopic(USTP_TERT_RESUME);
    // 设置飞马平台服务的地址，可以注册多个地址备用
        pUserApi->RegisterFront("tcp://172.28.21.133:15555");
    // 使客户端开始与后台服务建立连接
        pUserApi->Init();
    // 客户端等待报单操作完成
        WaitForSingleObject(g_hEvent, INFINITE);
    // 释放 API 实例
        pUserApi->Release();
    return 0;
}
```