

# 飞马平台

## TraderAPI 说明书



### 目录

#### 目录

1.	介绍 .....	4
2.	体系结构 .....	4
2.1	通讯模式 .....	4
2.2	数据流 .....	5
3.	接口模式 .....	6
3.1	对话流和查询流编程接口 .....	6
3.2	私有流编程接口 .....	7
4.	运行模式 .....	7
4.1	工作线程 .....	7
4.2	本地文件 .....	7
5.	业务和接口对照 .....	8
6.	开发接口 .....	10
6.1	通用规则 .....	10
6.2	CUstpFtdcTraderSpi 接口 .....	11
6.2.1	OnFrontConnected 方法 .....	11
6.2.2	OnQryFrontConnected 方法 .....	11

6.2.3	OnFrontDisconnected 方法.....	11
6.2.4	OnQryFrontDisconnected 方法.....	12
6.2.5	OnHeartBeatWarning 方法.....	12
6.2.6	OnPackageStart 方法.....	12
6.2.7	OnPackageEnd 方法.....	12
6.2.8	OnRspError 方法 .....	13
6.2.9	OnRspUserLogin 方法 .....	13
6.2.10	OnRspQueryUserLogin 方法 .....	15
6.2.11	OnRspUserLogout 方法.....	16
6.2.12	OnRspUserPasswordUpdate 方法.....	17
6.2.14	OnRspOrderInsert 方法.....	19
6.2.15	OnRspOrderAction 方法.....	21
6.2.16	OnRspQuoteInsert 方法.....	22
6.2.17	OnRspQuoteAction 方法.....	24
6.2.18	OnRspForQuote 方法.....	25
6.2.19	OnRspExecOrderInsert 方法.....	26
6.2.20	OnRspExecOrderAction 方法.....	27
6.2.21	OnRspMarginCombAction 方法.....	29
6.2.22	OnRspQryOrder 方法 .....	30
6.2.23	OnRspQryTrade 方法 .....	32
6.2.24	OnRspQryInvestorAccount 方法.....	34
6.2.25	OnRspQryTradingCode 方法.....	36
6.2.26	OnRspQryExchange 方法 .....	37
6.2.27	OnRspQryInstrument 方法.....	37
6.2.28	OnRspQryUserInvestor 方法.....	38
6.2.29	OnRspQryInvestorPosition 方法 .....	39
6.2.30	OnRspQryInvestorFee 方法.....	41
6.2.31	OnRspQryInvestorMargin 方法.....	42
6.2.32	OnRspQryInvestorLegPosition 方法.....	43
6.2.33	OnRspQryInvestorCombPosition 方法.....	45
6.2.45	OnRtnTrade 方法 .....	58
6.2.46	OnRtnOrder 方法 .....	60
1.1.1	OnRtnQuote 方法.....	62
1.1.1	OnRtnExecOrder 方法.....	64
1.1.2	OnRtnForQuote 方法 .....	65

1.1.3	OnRtnInstrumentStatus 方法.....	66
1.1.2	OnRtnMarginCombinationLeg方法.....	69
1.1.4	OnRtnInvestorAccountDeposit 方法.....	71
1.1.5	OnErrRtnOrderInsert 方法.....	71
1.1.6	OnErrRtnOrderAction 方法.....	73
1.1.7	OnErrRtnQuoteInsert 方法.....	74
1.1.8	OnErrRtnQuoteAction 方法.....	76
1.1.9	OnErrRtnExecOrderInsert方法.....	77
1.1.10	OnErrRtnExecOrderAction 方法.....	78
1.2	CUSTPFtdcTraderApi 接口.....	79
1.2.1	CreateFtdcTraderApi 方法.....	79
1.2.3	Release 方法.....	80
1.2.4	Init 方法.....	80
1.2.5	Join 方法.....	80
1.2.6	GetTradingDay 方法.....	80
1.2.7	RegisterSpi 方法.....	81
1.2.8	RegisterFront 方法.....	81
1.2.9	RegisterQryFront 方法.....	81
1.2.10	RegisterNameServer 方法.....	81
1.2.11	RegisterCertificateFile 方法（未启用） 订阅证书。.....	82
1.2.12	SubscribePrivateTopic 方法.....	82
1.2.13	SubscribePublicTopic 方法.....	83
1.2.14	SubscribeUserTopic 方法（未启用）.....	83
1.2.15	SubscribeForQuote 方法.....	83
1.2.16	SetHeartbeatTimeout 方法.....	84
1.2.20	ReqDSUserCertification 方法.....	84
1.2.21	ReqOrderInsert 方法.....	86
1.2.22	ReqOrderAction 方法.....	91
1.2.23	ReqQuoteInsert 方法.....	93
1.2.24	ReqExecOrderInsert方法.....	97
1.2.25	ReqExecOrderAction方法.....	98
1.2.26	ReqMarginCombAction 方法.....	99
1.2.27	ReqQryOrder 方法.....	100
1.2.28	ReqQryTrade 方法.....	101
1.2.29	ReqQryInvestorAccount 方法.....	102

1.2.30	ReqQryTradingCode 方法 .....	102
1.2.31	ReqQryExchange 方法.....	103
1.2.32	ReqQryInstrument 方法.....	103
1.2.33	ReqQryUserInvestor 方法 .....	104
1.2.34	ReqQryInvestorPosition 方法.....	104
1.2.35	ReqQryInvestorFee 方法.....	105
1.2.36	ReqQryInvestorMargin 方法.....	105
1.2.37	ReqQryQuote 方法.....	106
1.2.38	ReqQryInvestorCombPosition 方法.....	106
1.2.41	ReqQryInstrumentGroup 方法.....	108
1.2.42	ReqQryClientMarginCombType 方法.....	108
1.2.44	ReqQrySystemTime 方法 .....	109

## 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：

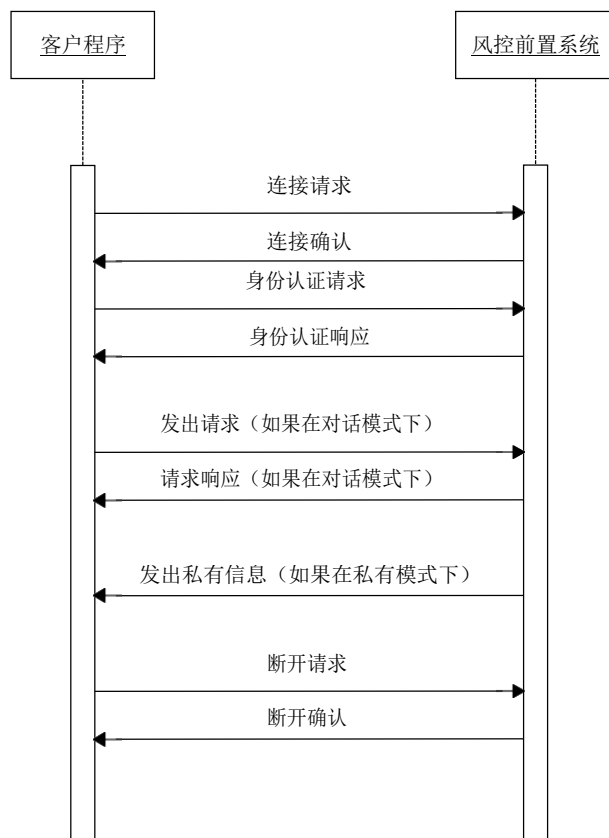


图 1-各种通讯模式的工作过程本接口

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( CUSTPFtdc  
    XXXField *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
登录	登录	CUSTPFtdcTraderApi::ReqUserLogin	CUSTPFtdcTraderSpi::OnRsp CUSTPFtdcTraderSpi::OnRsp
	登出	CUSTPFtdcTraderApi::ReqUserLogout	CUSTPFtdcTraderSpi::OnRsp
	修改用户口令	CUSTPFtdcTraderApi::ReqUserPasswordUpdate	CUSTPFtdcTraderSpi::OnRsp
报单	报单录入	CUSTPFtdcTraderApi::ReqOrderInsert	CUSTPFtdcTraderSpi::OnRsp
	报单操作	CUSTPFtdcTraderApi::ReqOrderAction	CUSTPFtdcTraderSpi::OnRsp
报价	报价录入	CUSTPFtdcTraderApi::ReqQuoteInsert	CUSTPFtdcTraderSpi::OnRsp
	报价操作	CUSTPFtdcTraderApi::ReqQuoteAction	CUSTPFtdcTraderSpi::OnRsp
行权	行权录入	CUSTPFtdcTraderApi::ReqExecOrderInsert	CUSTPFtdcTraderSpi::OnRsp
	行权操作	CUSTPFtdcTraderApi::ReqExecOrderAction	CUSTPFtdcTraderSpi::OnRsp
询价	询价请求	CUSTPFtdcTraderApi::ReqForQuote	CUSTPFtdcTraderSpi::OnRsp

组保	组合拆分请求	CUSTPFtdcTraderApi::ReqMarginCombAction	CUSTPFtdcTraderSpi::OnRsp
穿透式监管	客户认证请求	CUSTPFtdcTraderApi::ReqDSUserCertification	CUSTPFtdcTraderSpi::OnRsp
	中继处用户证书认证	CUSTPFtdcTraderApi::RegisterDSProxyUserCert	N/A
	信息采集中继上传信息	CUSTPFtdcTraderApi::ReqDSProxySubmitInfo	CUSTPFtdcTraderSpi::OnRsp
出入金	用户出入金请求	CUSTPFtdcTraderApi::ReqUserDeposit	CUSTPFtdcTraderSpi::OnRsp
	成交回报	N/A	CUSTPFtdcTraderSpi::OnRtn
	报单回报	N/A	CUSTPFtdcTraderSpi::OnRtn



私有回报	柜台出入金回报	N/A	CUSTPFtdcTraderSpi::OnRtnSit
	报单录入错误回报	N/A	CUSTPFtdcTraderSpi::OnErr
	报单操作错误回报	N/A	CUSTPFtdcTraderSpi::OnErr
	报价回报	N/A	CUSTPFtdcTraderSpi::OnRtn
	报价录入错误回报	N/A	CUSTPFtdcTraderSpi::OnErr
	行权通知	N/A	CUSTPFtdcTraderSpi::OnRtn

	行权录入错误回报	N/A	CUSTPFtdcTraderSpi::OnErr
	行权操作错误回报	N/A	CUSTPFtdcTraderSpi::OnErr
	资金同步通知回报	N/A	CUSTPFtdcTraderSpi::OnRtn
	用户出入金回报	N/A	CUSTPFtdcTraderSpi::OnRtn
组保回报	组合拆分申请回报	N/A	CUSTPFtdcTraderSpi::OnRtn
	组合规则回报	N/A	CUSTPFtdcTraderSpi::OnRtn
询价回报	询价回报	N/A	CUSTPFtdcTraderSpi::OnRtn
查询	报单查询	CUSTPFtdcTraderApi::ReqQryOrder	CUSTPFtdcTraderSpi::OnRsp
	成交查询	CUSTPFtdcTraderApi::ReqQryTrade	CUSTPFtdcTraderSpi::OnRsp
	合约查询	CUSTPFtdcTraderApi::ReqQryInstrument	CUSTPFtdcTraderSpi::OnRsp
	可用投资者查询	CUSTPFtdcTraderApi::ReqQryUserInvestor	CUSTPFtdcTraderSpi::OnRsp
	资金账户查询	CUSTPFtdcTraderApi::ReqQryInvestorAccount	CUSTPFtdcTraderSpi::OnRsp
	交易编码查询	CUSTPFtdcTraderApi::ReqQryTradingCode	CUSTPFtdcTraderSpi::OnRsp
	报价查询	CUSTPFtdcTraderApi::ReqQryQuote	CUSTPFtdcTraderSpi::OnRsp

	交易所查询	CUSTPFtdcTraderApi::ReqQryExchange	CUSTPFtdcTraderSpi::OnRsp
	投资者持仓查询	CUSTPFtdcTraderApi::ReqQryInvestorPosition	CUSTPFtdcTraderSpi::OnRsp
	投资者组合持仓查询	CUSTPFtdcTraderApi::ReqQryInvestorCombPosition	CUSTPFtdcTraderSpi::OnRsp
	投资者单腿持仓查询	CUSTPFtdcTraderApi::ReqQryInvestorLegPosition	CUSTPFtdcTraderSpi::OnRsp
	合规参数查询	CUSTPFtdcTraderApi::ReqQryComplianceParam	CUSTPFtdcTraderSpi::OnRsp
	交叉汇率查询	CUSTPFtdcTraderApi::ReqQryExchangeRate	CUSTPFtdcTraderSpi::OnRsp
	手续费率查询	CUSTPFtdcTraderApi::ReqQryInvestorFee	CUSTPFtdcTraderSpi::OnRsp
	保证金率查询	CUSTPFtdcTraderApi::ReqQryInvestorMargin	CUSTPFtdcTraderSpi::OnRsp
	合约组信息查询	CUSTPFtdcTraderApi::ReqQryInstrumentGroup	CUSTPFtdcTraderSpi::
	组合保证金类型查询	CUSTPFtdcTraderApi::ReqQryClientMarginCombType	CUSTPFtdcTraderSpi::

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 OnQryFrontConnected 方法

当客户端与飞马平台查询前置建立起通信连接时 (还未登录前), 该方法被调用。

函数原形:

```
void OnQryFrontConnected ();
```

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。

### 6.2.3 OnFrontDisconnected 方法

当客户端与飞马平台交易前置通信连接断开时, 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 (4097) 网络读失败

0x1002 (4098) 网络写失败

0x2001 (8193) 接收心跳超时

0x2002 (8194) 发送心跳失败

0x2003 (8195) 收到错误报文

#### 6.2.4 OnQryFrontDisconnected 方法

当客户端与飞马平台查询前置通信连接断开时，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 OnQryFrontDisconnected (int nReason);
```

##### 参数：

nReason:连接断开原因

0x1001 (4097) 网络读失败

0x1002 (4098) 网络写失败

0x2001 (8193) 接收心跳超时

0x2002 (8194) 发送心跳失败

0x2003 (8195) 收到错误报文

#### 6.2.5 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

#### 6.2.6 OnPackageStart 方法

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

##### 函数原形：

```
void OnPackageStart(int nTopicID, int nSequenceNo);
```

##### 参数：

nTopicID:主题代码（如私有流、公共流、行情流等）

nSequenceNo:报文序号

#### 6.2.7 OnPackageEnd 方法

报文回调结束通知。当 API 收到一个报文后，首先调用报文回调开始通知，然后是各

数据域的回调，最后调用本方法。

#### 函数原形:

```
void OnPackageEnd(int nTopicID, int nSequenceNo);
```

#### 参数:

nTopicID:主题代码（如私有流、公共流、行情流等）

nSequenceNo:报文序号

### 6.2.8 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。

### 6.2.9 OnRspUserLogin 方法

当客户端发出登录请求之后，飞马平台交易前置返回响应时，该方法会被调用，通知客户端交易前置登录是否成功。

#### 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。

用户登录响应信息结构:

```
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;
    ///登录成功时的交易所时间
    TUstpFtdcTimeType    ExchangeTime;
    ///用户最大本地报单号
    TUstpFtdcUserOrderLocalIDType    MaxOrderLocalID;
    ///交易系统名称
    TUstpFtdcTradingSystemNameType    TradingSystemName;
    ///数据中心代码
    TUstpFtdcDataCenterIDType    DataCenterID;
    ///会员私有流当前长度
    TUstpFtdcSequenceNoType    PrivateFlowSize;
    ///交易员私有流当前长度
    TUstpFtdcSequenceNoType    UserFlowSize;
    ///业务发生日期
    TUstpFtdcDateType    ActionDay;
    ///飞马后台版本号
    TUstpFtdcFemasVersionType    FemasVersion;
    ///飞马生命周期号TUstpFtdcFemasLifeCycleType
    FemasLifeCycle;
};
```

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。

### 6.2.10 OnRspQueryUserLogin 方法

当客户端发出登录请求之后, 飞马平台查询前置返回响应时, 该方法会被调用, 通知客户端查询前置登录是否成功。

```
void OnRspQueryUserLogin(
    CUstpFtdcRspUserLoginField *pRspUserLogin,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

#### 参数:

pRspUserLogin:返回用户登录信息的地址。

用户登录响应信息结构:

```
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;
    ///登录成功时的交易所时间
    TUstpFtdcTimeType    ExchangeTime;
    ///用户最大本地报单号
    TUstpFtdcUserOrderLocalIDType    MaxOrderLocalID;
    ///交易系统名称
    TUstpFtdcTradingSystemNameType    TradingSystemName;
    ///数据中心代码
    TUstpFtdcDataCenterIDType    DataCenterID;
    ///会员私有流当前长度
    TUstpFtdcSequenceNoType    PrivateFlowSize;
    ///交易员私有流当前长度
};
```

```

    TUstpFtdcSequenceNoType UserFlowSize;
    ///业务发生日期
    TUstpFtdcDateType      ActionDay;
    ///飞马后台版本号
    TUstpFtdcFemasVersionType FemasVersion;
    ///飞马生命周期号
    TUstpFtdcFemasLifeCycleType FemasLifeCycle;
};

```

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。

### 6.2.11 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。

### 6.2.12 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( CUstpFtdcUserPasswordUpdateFi
        eld *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。

```

### 6.2.13 OnRspQueryUserLogin 方法（未启用）

查询前置系统用户登录应答。

**函数原形：**

```

void
OnRspQueryUserLogin( CUstpFtdcRspUserLogin
    Field *pRspUserLogin,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)

```

**参数：**

pRspUserLogin:指向系统用户登录应答结构的地址

系统用户登录应答结构：

```

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;
    ///登录成功时的交易所时间
    TUstpFtdcTimeType    ExchangeTime;
    ///用户最大本地报单号
    TUstpFtdcUserOrderLocalIDType    MaxOrderLocalID;
    ///交易系统名称
    TUstpFtdcTradingSystemNameType    TradingSystemName;
    ///数据中心代码
    TUstpFtdcDataCenterIDType    DataCenterID;

```

```

///会员私有流当前长度
TUstpFtdcSequenceNoType PrivateFlowSize;
///交易员私有流当前长度
TUstpFtdcSequenceNoType UserFlowSize;
///业务发生日期
TUstpFtdcDateType ActionDay;
///飞马版本号
TUstpFtdcFemasVersionType FemasVersion;
///飞马生命周期号TUstpFtdcFemasLifeCycleType
FemasLifeCycle;
};

```

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:返回查询前置登陆请求的 ID, 该 ID 由用户在查询前置登陆时指定。

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

#### 6.2.14 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;
///指定下单席位编号（取值范围[1-N]，N 为可用席位数目，超出范围的随机分配席位）
TUstpFtdcSeatNoType SeatNo;
///合约代码/套利组合合约号
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（暂不支持，保留域）
TUstpFtdcDateType GTDDate;
///成交量类型
TUstpFtdcVolumeConditionType VolumeCondition;
///最小成交量
TUstpFtdcVolumeType MinVolume;
///止损价/止盈价
TUstpFtdcPriceType StopPrice;
/// 强 平 原 因 （ 暂 不 支 持 ， 保 留 域 ）
TUstpFtdcForceCloseReasonType ForceCloseReason;
///自动挂起标志（暂不支持，保留域）
TUstpFtdcBoolType IsAutoSuspend;
///业务单元（暂不支持，保留域）
TUstpFtdcBusinessUnitType BusinessUnit;
///用户自定义域 64 字节
```

```

    TUstpFtdcCustomType UserCustom;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
    ///业务发生日期
    TUstpFtdcDateType ActionDay;
    ///策略类别
    TUstpFtdcArbiTypeType ArbiType;
};

```

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。

### 6.2.15 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（暂不支持，保留域）
    TUstpFtdcPriceType LimitPrice;
    ///数量变化（暂不支持，保留域）
    TUstpFtdcVolumeType VolumeChange;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
};

```

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。

## 6.2.16 OnRspQuoteInsert 方法

报价录入应答。当客户端发出报价录入指令后，飞马平台返回响应时，该方法会被调用。

**函数原形:**

```

void OnRspQuoteInsert
(
    CUstpFtdcInputQuoteField
    *pInputQuote, CUstpFtdcRspInfoField
    *pRspInfo,
    int nRequestID,
    bool bIsLast)

```

**参数:**

pInputQuote 指向报价结构的地址。

报价结构:

```

struct CUstpFtdcInputQuoteField
{
    ///经纪公司编号

```

```
TUstpFtdcBrokerIDType BrokerID;
///交易所代码
TUstpFtdcExchangeIDType ExchangeID;
///投资者编号
TUstpFtdcInvestorIDType InvestorID;
///用户代码
TUstpFtdcUserIDType UserID;
///指定通过此席位序号下单，取值范围[1-N]，N 为可用席位数目，
//超出范围的随机分配席位
TUstpFtdcSeatNoType SeatNo;
///合约代码
TUstpFtdcInstrumentIDType InstrumentID;
///交易系统返回的系统报价编号
TUstpFtdcQuoteSysIDType QuoteSysID;
///用户设定的本地报价编号
TUstpFtdcUserQuoteLocalIDType UserQuoteLocalID;
///飞马向交易系统报的本地报价编号
TUstpFtdcQuoteLocalIDType QuoteLocalID;
///买方买入数量
TUstpFtdcVolumeType BidVolume;
///买方开平标志
TUstpFtdcOffsetFlagType BidOffsetFlag;
///买方投机套保标志
TUstpFtdcHedgeFlagType BidHedgeFlag;
///买方买入价格
TUstpFtdcPriceType BidPrice;
///卖方卖出数量
TUstpFtdcVolumeType AskVolume;
///卖方开平标志
TUstpFtdcOffsetFlagType AskOffsetFlag;
///卖方投机套保标志
TUstpFtdcHedgeFlagType AskHedgeFlag;
///卖方卖出价格
TUstpFtdcPriceType AskPrice;
///业务单元
TUstpFtdcBusinessUnitType BusinessUnit;
///用户自定义域
TUstpFtdcCustomType UserCustom;
///拆分出来的买方用户本地报单编号
TUstpFtdcUserOrderLocalIDType BidUserOrderLocalID;
///拆分出来的卖方用户本地报单编号
TUstpFtdcUserOrderLocalIDType AskUserOrderLocalID;
///询价编号
TUstpFtdcQuoteSysIDType ReqForQuoteID;
```

```

    ///报价停留时间(秒)
    TUstpFtdcMeasureSecType StandByTime;
    ///客户编码
    TUstpFtdcClientIDType ClientID;
    ///下单用户编号
    TUstpFtdcUserIDType QuoteUserID;
    ///指定通过此席位序号下单
    TUstpFtdcSeatNoType SeatNo;
};
报价录入时必须满足: MaxOrderLocalID< UserQuoteLocalID< BidUserOrderLocalID<
AskUserOrderLocalID

```

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:返回用户报价请求的 ID, 该 ID 由用户在报价请求时指定。

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

### 6.2.17 OnRspQuoteAction 方法

报价操作应答。当客户端发出报价操作指令后, 飞马平台返回响应时, 该方法会被调用。

**函数原形:**

```

void OnRspQuoteAction
(
    CustpFtdcQuoteActionField* pQuoteAction,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)

```

**参数:**

pQuoteAction 指向报价操作结构的地址。

报价操作结构:

```

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

```



```

    TUstpFtdcUserIDType UserID;
    ///交易系统返回的系统报价编号
    TUstpFtdcQuoteSysIDType QuoteSysID;
    ///用户设定的被撤的本地报价编号
    TUstpFtdcUserQuoteLocalIDType UserQuoteLocalID;
    ///用户向飞马报的本地撤消报价编号
    TUstpFtdcUserQuoteLocalIDType UserQuoteActionLocalID;
    ///报单操作标志
    TUstpFtdcActionFlagType ActionFlag;
    ///业务单元
    TUstpFtdcBusinessUnitType BusinessUnit;
    ///用户自定义域
    TUstpFtdcCustomType UserCustom;
};

```

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:返回用户报价操作请求的 ID, 该 ID 由用户在报价操作请求时指定。

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

### 6.2.18 OnRspForQuote 方法

询价应答。当客户端发出询价指令后, 飞马平台返回响应时, 该方法会被调用。

**函数原形:**

```

void
    OnRspForQuote( CUstpFtdcReqForQuoteField
        *pReqForQuote CUstpFtdcRspInfoField
        *pRspInfo,
        int nRequestID,
        bool bIsLast)

```

**参数:**

pReqForQuote 指向询价结构的地址。

询价结构:

```

struct CUstpFtdcReqForQuoteField
{
    ///询价编号
    TUstpFtdcQuoteSysIDType ReqForQuoteID;
    ///经纪公司编号

```

```

TUstpFtdcBrokerIDType BrokerID;
///交易所代码
TUstpFtdcExchangeIDType ExchangeID;
///投资者编号
TUstpFtdcInvestorIDType InvestorID;
///用户代码
TUstpFtdcUserIDType UserID;
///合约代码
TUstpFtdcInstrumentIDType InstrumentID;
///交易日
TUstpFtdcDateType TradingDay;
///询价时间
TUstpFtdcTimeType ReqForQuoteTime;
};

```

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:返回询价请求的 ID, 该 ID 由用户在询价请求时指定。

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

## 6.2.19 OnRspExecOrderInsert 方法

报价录入应答。当客户端发出报价录入指令后, 飞马平台返回响应时, 该方法会被调用。

### 函数原形:

```

void
OnRspExecOrderInsert( CUstpFtdcInputExecOrderF
    ield *pInputExecOrder, CUstpFtdcRspInfoField
    *pRspInfo,
    int nRequestID,
    bool bIsLast)

```

### 参数:

pInputExecOrder:指向行权录入结构的地址。

```

struct CUstpFtdcInputExecOrderField
{
    ///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;
    ///委托类型
    TUstpFtdcOrderTypeType OrderType;
    ///期权对冲标识
    TUstpFtdcDeliveryFlagType DeliveryFlag;
    ///投机套保标志
    TUstpFtdcHedgeFlagType HedgeFlag;
    ///数量
    TUstpFtdcVolumeType Volume;
    ///用户自定义域
    TUstpFtdcCustomType UserCustom;
    ///业务发生日期
    TUstpFtdcDateType ActionDay;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
    ///业务单元
    TUstpFtdcBusinessUnitType BusinessUnit;
};

```

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:返回询价请求的 ID, 该 ID 由用户在询价请求时指定。

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

## 6.2.20 OnRspExecOrderAction 方法

报价录入应答。当客户端发出报价录入指令后, 飞马平台返回响应时, 该方法会被调用。

**函数原形:**

```
void OnRspExecOrderAction(
    CUstpFtdcInputExecOrderActionField * pInputExecOrderAction,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

**参数:**

pInputExecOrderAction:指向行权操作结构的地址。

```
struct CUstpFtdcInputExecOrderActionField
{
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///系统报单编号
    TUstpFtdcOrderSysIDType OrderSysID;
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///本次撤单操作的本地编号
    TUstpFtdcUserOrderLocalIDType UserOrderActionLocalID;
    ///被撤订单的本地报单编号
    TUstpFtdcUserOrderLocalIDType UserOrderLocalID;
    ///报单操作标志
    TUstpFtdcActionFlagType ActionFlag;
    ///数量变化
    TUstpFtdcVolumeType VolumeChange;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
    ///委托类型
    TUstpFtdcOrderTypeType OrderType;
};
```

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: 返回询价请求的 ID, 该 ID 由用户在询价请求时指定。

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

### 6.2.21 OnRspMarginCombAction 方法

客户申请组合应答。当客户端发起组合操作请求 ReqMarginCombAction 时, 该方法会被调用。

#### 函数原形:

```
void OnRspMarginCombAction(
    CUstpFtdcInputMarginCombActionField *pInputMarginCombAction,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

#### 参数:

pInputMarginCombAction: 指向组合策略操作结构的地址

```
struct CUstpFtdcInputMarginCombActionField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///交易用户代码
    TUstpFtdcUserIDType   UserID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///投机套保标志
    TUstpFtdcHedgeFlagType HedgeFlag;
    ///用户本地编号
    TUstpFtdcUserOrderLocalIDType   UserActionLocalID;
    ///组合合约代码
    TUstpFtdcInstrumentIDType   CombInstrumentID;
    ///组合数量
    TUstpFtdcVolumeType   CombVolume;
    ///组合动作方向
    TUstpFtdcCombDirectionType   CombDirection;
    ///本地编号
    TUstpFtdcOrderLocalIDType   ActionLocalID;
};
```

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:返回组合操作请求的 ID, 该 ID 由用户在组合操作请求时指定。

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

## 6.2.22 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;
    ///指定下单席位序号
    TUstpFtdcSeatNoType SeatNo;
    ///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（暂不支持，保留域）
TUstpFtdcDateType GTDDate;
///成交量类型
TUstpFtdcVolumeConditionType VolumeCondition;
///最小成交量
TUstpFtdcVolumeType MinVolume;
///止损价（暂不支持，保留域）
TUstpFtdcPriceType StopPrice;
///强平原因（暂不支持，保留域）
TUstpFtdcForceCloseReasonType ForceCloseReason;
///自动挂起标志（暂不支持，保留域）
TUstpFtdcBoolType IsAutoSuspend;
///业务单元（暂不支持，保留域）
TUstpFtdcBusinessUnitType BusinessUnit;
///用户自定义域
TUstpFtdcCustomType UserCustom;
///交易日
TUstpFtdcTradingDayType TradingDay;
///会员编号
TUstpFtdcParticipantIDType ParticipantID;
///下单用户编号
TUstpFtdcUserIDType OrderUserID;
///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。

## 6.2.23 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;
///客户编码
TUstpFtdcClientIDType ClientID;
///用户编号
TUstpFtdcUserIDType UserID;
///下单用户编号
TUstpFtdcUserIDType OrderUserID;
///成交编号
TUstpFtdcTradeIDType TradeID;
///系统报单编号
TUstpFtdcOrderSysIDType OrderSysID;
///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;
///本次成交手续费
TUstpFtdcMoneyType UsedFee;
///本次成交占用保证金
TUstpFtdcMoneyType UsedMargin;
///本次成交占用权利金
TUstpFtdcMoneyType Premium;
///持仓表今持仓量
TUstpFtdcVolumeType Position;
///持仓表今日持仓成本
TUstpFtdcPriceType PositionCost;
```

```

    ///资金表可用资金
    TUstpFtdcMoneyType Available;
    ///资金表占用保证金
    TUstpFtdcMoneyType Margin;
    ///资金表冻结的保证金
    TUstpFtdcMoneyType FrozenMargin;
};

```

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。

## 6.2.24 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( CUstpFtdcRspInvestorAccountFi
    eld *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。

```

### 6.2.25 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;
    ///客户编码
    TUstpFtdcClientIDType ClientID;
    ///客户编码权限
    TUstpFtdcTradingRightType ClientRight;
    ///客户保值类型
    TUstpFtdcHedgeFlagType ClientHedgeFlag;
    ///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。

```

### 6.2.26 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( CUstpFtdcRspExchang
eField *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。

### 6.2.27 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( CUSTpFtdcRspInstrume
                           ntField *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
{

};
```

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。

### 6.2.28 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。

### 6.2.29 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 编 码
    TUstpFtdcClientIDType   ClientID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
};

```

```

    /// 买 卖 方 向
    TUstpFtdcDirectionType  Direction;
    ///投机套保标志
    TUstpFtdcHedgeFlagType  HedgeFlag;
    ///优惠前占用保证金
    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;
    /// 平 昨 仓 冻 结 持 仓
    TUstpFtdcVolumeType  YdFrozenClosing;
    /// 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;

    ///投机持仓量
    TUstpFtdcVolumeType  SpeculationPosition;
    ///套利持仓量
    TUstpFtdcVolumeType  ArbitragePosition;
    ///套保持仓量
    TUstpFtdcVolumeType  HedgePosition;
    ///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。

### 6.2.30 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( CUstpFtdcInvestorFee
    eField *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;
    ///行权手续费按比例
    TUstpFtdcRatioType ExecFeeRate;
    ///行权手续费按手数
    TUstpFtdcRatioType ExecFeeAmt;
    ///每笔委托的申报费
    TUstpFtdcRatioType PerOrderAmt;
    ///每笔撤单申报费
    TUstpFtdcRatioType PerCancelAmt;
    ///投机套保标志
    TUstpFtdcHedgeFlagType HedgeFlag;
};

```

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。

### 6.2.31 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;
    ///客户编码
    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:指示该次返回是否针对 nRequestID 的最后一次返回。

## 6.2.32 OnRspQryInvestorLegPosition 方法

投资者单腿持仓查询应答。当客户端发出投资者单腿持仓查询指令后, 飞马平台返回响应时, 该方法会被调用。

**函数原形:**

```

void OnRspQryInvestorLegPosition
( CUSTPFTDCRSPINVESTORLEGPOSITIONFIELD
  *pInvestorPosition, CUSTPFTDCRSPINFOFIELD *pRspInfo,
  int nRequestID,

```

```
bool bIsLast)
```

### 参数:

pInvestorPosition 指向投资者单腿持仓结构的地址。

投资者单腿持仓结构:

```
struct CUstpFtdcRspInvestorLegPositionField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///投机套保标志
    TUstpFtdcHedgeFlagType  HedgeFlag;
    ///客户编码
    TUstpFtdcClientIDType   ClientID;
    ///单腿合约代码
    TUstpFtdcInstrumentIDType InstrumentID;
    ///多头持仓
    TUstpFtdcVolumeType LongPosition;
    ///空头持仓
    TUstpFtdcVolumeType ShortPosition;
    ///多头占用保证金
    TUstpFtdcMoneyType LongMargin;
    ///空头占用保证金
    TUstpFtdcMoneyType ShortMargin;
    ///多头冻结持仓
    TUstpFtdcVolumeType LongFrozenPosition;
    ///空头冻结持仓
    TUstpFtdcVolumeType ShortFrozenPosition;
    ///多头冻结保证金
    TUstpFtdcMoneyType LongFrozenMargin;
    ///空头冻结保证金
    TUstpFtdcMoneyType ShortFrozenMargin;
};
```

pRspInfo:指向响应信息结构的地址。

响应信息结构:

版权所有©上海金融期货信息技术有限公司

```

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

```

nRequestID:返回用户单腿持仓查询请求的 ID, 该 ID 由用户在单腿持仓查询时指定。

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

### 6.2.33 OnRspQryInvestorCombPosition 方法

投资者组合持仓查询应答。当客户端发出投资者组合持仓查询指令后, 飞马平台返回响应时, 该方法会被调用。

#### 函数原形:

```

void OnRspQryInvestorCombPosition
( CUstpFtdcRspInvestorCombPositionField
 *pInvestorPosition, CUstpFtdcRspInfoField *pRspInfo,
 int nRequestID,
 bool bIsLast)

```

#### 参数:

pInvestorPosition 指向投资者组合持仓结构的地址。

投资者组合持仓结构:

```

struct CUstpFtdcRspInvestorCombPositionField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;

    /// 持 仓 方 向
    TUstpFtdcDirectionType Direction;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///投机套保标志
    TUstpFtdcHedgeFlagType HedgeFlag;
    /// 客 户 编 码
    TUstpFtdcClientIDType ClientID;
    ///组合合约代码
    TUstpFtdcCombInstrumentIDType CombInstrumentID;

```

```

    ///单腿 1 合约代码
    TUstpFtdcInstrumentIDType  Leg1InstrumentID;
    ///单腿 2 合约代码
    TUstpFtdcInstrumentIDType  Leg2InstrumentID;
    ///组合持仓
    TUstpFtdcVolumeType  CombPosition;
    ///冻结组合持仓
    TUstpFtdcVolumeType  CombFrozenPosition;
    ///组合一手释放的保证金
    TUstpFtdcMoneyType  CombFreeMargin;
};

```

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:返回用户组合持仓查询请求的 ID, 该 ID 由用户在组合持仓查询时指定。

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

## 6.2.34 OnRspQryInstrumentGroup 方法

合约组信息查询应答。

### 函数原形:

```

void OnRspQryInstrumentGroup(
    CUstpFtdcRspInstrumentGroupField *pRspInstrumentGroup,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);

```

### 参数:

pRspInstrumentGroup 指向合约组信息查询应答结构的地址。

合约组信息查询应答结构:

```

struct CUstpFtdcRspInstrumentGroupField
{
    ///交易所代码
    TUstpFtdcExchangeIDType  ExchangeID;

```

```

///经纪公司编号
TUstpFtdcBrokerIDType   BrokerID;
///合约代码
TUstpFtdcInstrumentIDType   InstrumentID;
///合约组代码
TUstpFtdcInstrumentGroupIDType   InstrumentGroupID;
};

```

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:返回合约组查询请求的 ID, 该 ID 由用户在合约组查询时指定。

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

### 6.2.35 OnRspQryClientMarginCombType 方法

合约组信息查询应答。

**函数原形:**

```

void
OnRspQryClientMarginCombType( CUstpFtdcRspClientMarginCombTypeField *pRspClientMarginCombType, CUstpFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast);

```

**参数:**pRspClientMarginCombType 指向组合保证金类型查询应答结构的地址。组合保证金类型查询应答结构:

```

struct CUstpFtdcRspClientMarginCombTypeField
{
    ///交易所代码
    TUstpFtdcExchangeIDType   ExchangeID;
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///会员代码
    TUstpFtdcParticipantIDType   ParticipantID;
    ///客 户 编 码

```

```

TUstpFtdcClientIDType   ClientID;
///投资者编号
TUstpFtdcInvestorIDType InvestorID;
///投机套保标志
TUstpFtdcHedgeFlagType  HedgeFlag;
///合约组代码
TUstpFtdcInstrumentGroupIDType InstrumentGroupID;
/// 保 证 金 组 合 类 型
TUstpFtdcClientMarginCombTypeType MarginCombType;
};

```

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:返回用户保证金收取类型查询请求的 ID, 该 ID 由用户在保证金收取类型查询时指定。

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

## 6.2.36 OnRspQrySystemTime 方法

系统时间查询应答。

**函数原形:**

```

void OnRspQrySystemTime(
    CUstpFtdcRspQrySystemTimeField *pRspQrySystemTime,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);

```

**参数:**

pRspQrySystemTime 指向系统时间结构的地址。

系统时间结构

```

///系统时间
struct CUstpFtdcRspQrySystemTimeField
{
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///系统时间

```



```
TUstpFtdcTimeType    SystemTime;
};
```

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:返回系统时间查询请求的 ID, 该 ID 由用户在系统时间查询时指定。

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

### 6.2.37 OnRspQryCFMMCTradingAccountKey 方法（未启用）

查询保证金监管系统经纪公司资金账户密钥响应。

**函数原形:**

```
void
OnRspQryCFMMCTradingAccountKey( CUstpFtdcCFMMCTradingAccountKeyRspFi
    eld *pCFMMCTradingAccountKeyRsp, CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

**参数:**

pCFMMCTradingAccountKeyRsp 指向保证金监管系统经纪公司资金账户密钥应答结构的地址。

保证金监管系统经纪公司资金账户密钥应答结构:

```
struct CUstpFtdcCFMMCTradingAccountKeyRspField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType    BrokerID;
    ///会员编号
    TUstpFtdcParticipantIDType ParticipantID;
    /// 资 金 帐 号
    TUstpFtdcAccountIDType    AccountID;
    ///密钥编号
    TUstpFtdcSequenceNoTypeType KeyID;
    ///动态密钥
    TUstpFtdcCFMMCKeyType    CurrentKey;
};
```

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:返回用户查询密钥请求的 ID, 该 ID 由用户在密钥查询时指定。

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

## 6.2.38 OnRspQrySettlementInfo 方法 (未启用)

请求查询投资者结算结果响应。

**函数原形:**

```
void
OnRspQrySettlementInfo( CUstpFtdcSettlementR
    spField *pSettlementRsp,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

**参数:**

pSettlementRsp 指向查询投资者结算结果应答结构的地址。

查询投资者结算结果应答结构:

```
struct CUstpFtdcSettlementRspField
{
    ///交易日
    TUstpFtdcDateType TradingDay;
    ///结算编号
    TUstpFtdcSettlementIDType SettlementID;
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///序号
    TUstpFtdcSequenceNoType SequenceNo;
    ///消息内容
    TUstpFtdcLogStrType Content;
};
```

pRspInfo:指向响应信息结构的地址。

版权所有©上海金融期货信息技术有限公司

响应信息结构:

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

nRequestID:返回用户结算单查询请求的 ID, 该 ID 由用户在结算单查询时指定。

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

## 6.2.39 OnRspTradingAccountPasswordUpdate 方法 (未启用)

资金账户口令更新请求响应。

**函数原形:**

```
void
OnRspTradingAccountPasswordUpdate( CUstpFtdcT
    radingAccountPasswordUpdateRspField
    *pTradingAccountPasswordUpdateRsp,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

**参数:**

pTradingAccountPasswordUpdateRsp 指向资金账户口令更新请求响应结构的地址。

资金账户口令更新请求响应结构:

```
struct CUstpFtdcTradingAccountPasswordUpdateRspField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///旧密码
    TUstpFtdcPasswordType OldPassword;
    ///新密码
    TUstpFtdcPasswordType NewPassword;
};
```

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:返回资金帐号密码修改请求的 ID, 该 ID 由用户在资金帐号密码修改时指定。

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

#### 6.2.40 OnRspQryTransferBank 方法（未启用）

请求查询转帐银行响应。

**函数原形:**

```

void OnRspQryTransferBank(
    CUstpFtdcTransferBankQryRspField *pTransferBankQryRsp,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);

```

**参数:**

pTransferBankQryRsp 指向查询转帐银行响应结构的地址。

查询转帐银行响应结构:

```

struct CUstpFtdcTransferBankQryRspField
{
    ///银行代码
    TUstpFtdcBrokerIDType BankID;
    ///银行分中心代码
    TUstpFtdcInvestorIDType BankBrchID;
    ///银行名称
    TUstpFtdcBankNameType BankName;
    ///是否活跃
    TUstpFtdcBoolType 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:返回查询转账银行请求的 ID, 该 ID 由用户在转账银行查询时指定。

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

版权所有©上海金融期货信息技术有限公司

### 6.2.41 OnRspQryEWarrantOffset 方法（未启用）

请求查询仓单折抵信息响应。

#### 函数原形：

```
void OnRspQryEWarrantOffset(
    CUstpFtdcEWarrantOffsetFieldRspField *pEWarrantOffsetFieldRsp,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

#### 参数：

pEWarrantOffsetFieldRsp 指向查询仓单折抵信息响应结构的地址。

查询仓单折抵信息响应结构：

```
struct CUstpFtdcEWarrantOffsetFieldRspField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///合约代码
    TUstpFtdcInstrumentIDType InstrumentID;
    ///买卖方向
    TUstpFtdcDirectionType Direction;
    ///投机套保标志
    TUstpFtdcHedgeFlagType HedgeFlag;
    ///数量
    TUstpFtdcVolumeType Volume;
};
```

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:返回用户仓单查询请求的 ID, 该 ID 由用户在仓单查询时指定。

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

## 6.2.42 OnRspQryTransferSerialOffset 方法（未启用）

请求查询转帐流水响应。

**函数原形:**

```
void
    OnRspQryTransferSerialOffset( CUstpFtdcTransferSerialFieldRspField
        *pTransferSerialFieldRsp, CUstpFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

**参数:**

pTransferSerialFieldRsp 指向查询转帐流水响应响应结构的地址。

查询转帐流水响应响应结构:

```
struct CUstpFtdcTransferSerialFieldRspField
{
    ///平台流水号
    TUstpFtdcPlateSerialType    PlateSerial;
    ///交易发起方日期
    TUstpFtdcDateType    TradeDate;
    ///交易日期
    TUstpFtdcDateType    TradingDay;
    ///交易时间
    TUstpFtdcTimeType    TradeTime;
    /// 交 易 代 码
    TUstpFtdcTradeCodeType    TradeCode;
    /// 会 话 编 号
    TUstpFtdcSessionIDType    SessionID;
    ///银行代码
    TUstpFtdcBrokerIDType    BankID;
    ///银行分中心代码
    TUstpFtdcInvestorIDType    BankBrchID;
    /// 银 行 帐 号 类 型
    TUstpFtdcBankAccType    BankAccType;
    ///银行帐号
    TUstpFtdcBankAccountType    BankAccount;
    ///银行流水号
    TUstpFtdcSerialType    Serial;
};
```

```

TUstpFtdcBankSerialType BankSerial;
///经纪公司编号
TUstpFtdcBrokerIDType BrokerID;
///期商分支机构代码
TUstpFtdcFutureBranchIDType BrokerBranchID;
///期货公司帐号类型
TUstpFtdcFutureAccType FutureAccType;
/// 资 金 帐 号
TUstpFtdcAccountIDType AccountID;
///投资者编号
TUstpFtdcInvestorIDType InvestorID;
///期货公司流水号
TUstpFtdcFutureSerialType FutureSerial;
/// 证 件 类 型
TUstpFtdcIdCardTypeType IdCardType;
///证件号码
TUstpFtdcIdentifiedCardNoType IdentifiedCardNo;
/// 币 种 代 码
TUstpFtdcCurrencyIDType CurrencyID;
///金额
TUstpFtdcMoneyType TradeAmount;
///应收客户费用
TUstpFtdcMoneyType CustFee;
///应收期货公司费用
TUstpFtdcMoneyType BrokerFee;
///有效标志
TUstpFtdcAvailabilityFlagTypeType AvailabilityFlag;
///操作员
TUstpFtdcOperatorCodeType OperatorCode;
///新银行帐号
TUstpFtdcBankAccountType BankNewAccount;
};
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: 返回用户转账流水查询请求的 ID, 该 ID 由用户在转账流水查询时指定。

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

#### 6.2.43 OnRspQryAccountregister 方法 (未启用)

请求查询银期签约关系响应。

**函数原形:**

```

void
    OnRspQryAccountregister( CUstpFtdcAccountregisterRspFi
        eld *pAccountregisterRsp, CUstpFtdcRspInfoField
        *pRspInfo,
        int nRequestID,
        bool bIsLast);

```

**参数:**

pAccountregisterRsp 指向查询银期签约关系响应结构的地址。

查询银期签约关系响应结构:

```

struct CUstpFtdcAccountregisterRspField
{
    ///交易日期
    TUstpFtdcDateType    TradeDate;
    ///银行代码
    TUstpFtdcBrokerIDType    BankID;
    ///银行分中心代码
    TUstpFtdcInvestorIDType    BankBrchID;
    ///银行帐号
    TUstpFtdcBankAccountType    BankAccount;
    ///经纪公司编号
    TUstpFtdcBrokerIDType    BrokerID;
    ///资 金 帐 号
    TUstpFtdcAccountIDType    AccountID;
    ///证 件 类 型
    TUstpFtdcIdCardTypeType    IdCardType;
    ///证件号码
    TUstpFtdcIdentifiedCardNoType    IdentifiedCardNo;
    ///客户姓名

```

版权所有©上海金融期货信息技术有限公司



```

TUstpFtdcIndividualNameType CustomerName;
/// 币种代码
TUstpFtdcCurrencyIDType CurrencyID;
///开销户类别
TUstpFtdcOpenOrDestroyTypeType OpenOrDestroy;
/// 签约日期
TUstpFtdcDateType RegDate;
/// 解约日期
TUstpFtdcDateType OutDate;
/// 交易ID
TUstpFtdcTIDType TID;
/// 客户类型
TUstpFtdcCustTypeType CustType;
/// 银行帐号类型
TUstpFtdcBankAccType BankAccType;
};

```

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:返回用户签约查询请求的 ID, 该 ID 由用户在签约关系查询时指定。

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

#### 6.2.44 OnRspQryContractBank 方法（未启用）

查询签约银行响应。

**函数原形:**

```

void OnRspQryContractBank(
    CUstpFtdcContractBankFieldRspField *pContractBankFieldRsp,
    CUstpFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);

```

**参数:**

pContractBankFieldRsp 指向查询签约银行响应结构的地址。

查询签约银行响应结构:

```
struct CUstpFtdcContractBankFieldRspField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///银行代码
    TUstpFtdcBrokerIDType   BankID;
    ///银行分中心代码
    TUstpFtdcInvestorIDType BankBrchID;
    ///银 行 名 称
    TUstpFtdcBankNameType   BankName;
};
```

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:返回用户签约银行查询请求的 ID, 该 ID 由用户在签约银行查询时指定。

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

## 6.2.45 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;
///下单用户编号
TUstpFtdcUserIDType OrderUserID;
///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;
///本次成交手续费
TUstpFtdcMoneyType UsedFee;
///本次成交占用保证金
TUstpFtdcMoneyType UsedMargin;
///本次成交占用权利金
TUstpFtdcMoneyType Premium;
///持仓表今持仓量
TUstpFtdcVolumeType Position;
///持仓表今日持仓成本
TUstpFtdcPriceType PositionCost;
///资金表可用资金
```

```

    TUstpFtdcMoneyType Available;
    ///资金表占用保证金
    TUstpFtdcMoneyType Margin;
    ///资金表冻结的保证金
    TUstpFtdcMoneyType FrozenMargin;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
    ///业务发生日期
    TUstpFtdcDateType ActionDay;
    ///策略类别
    TUstpFtdcArbiTypeType ArbiType;
    ///合约代码
    TUstpFtdcInstrumentIDType ArbiInstrumentID;
};

```

#### 6.2.46 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;
    ///指定下单席位编号
    TUstpFtdcSeatNoType SeatNo;
    ///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（暂不支持，保留域）
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 （ 暂 不 支 持 ， 保 留 域 ）
TUstpFtdcForceCloseReasonType  ForceCloseReason;
///自动挂起标志（暂不支持，保留域）
TUstpFtdcBoolType  IsAutoSuspend;
///业务单元（暂不支持，保留域）
TUstpFtdcBusinessUnitType  BusinessUnit;
///用户自定义域
TUstpFtdcCustomType  UserCustom;
///本地业务标识
TUstpFtdcBusinessLocalIDType  BusinessLocalID;
///业务发生日期
TUstpFtdcDateType  ActionDay;
/// 策 略 类 别
TUstpFtdcArbiTypeType  ArbiType;
///交易日
TUstpFtdcTradingDayType  TradingDay;
///会员编号
TUstpFtdcParticipantIDType  ParticipantID;
```

```

    ///最初下单用户编号
    TUstpFtdcUserIDType OrderUserID;
    ///客户编码
    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;
    ///委托类型
    TUstpFtdcOrderTypeType OrderType;

    ///期权对冲标识
    TUstpFtdcDeliveryFlagType DeliveryFlag;
};

```

### 6.2.47 OnRtnQuote 方法

报价回报。由飞马平台主动通知客户端，该方法会被调用。

**函数原形：**

```
void OnRtnQuote (CustpFtdcRtnQuoteField *pRtnQuote);
```

**参数：**

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

报价回报结构：

```

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

```

```
TUstpFtdcExchangeIDType ExchangeID;
///投资者编号
TUstpFtdcInvestorIDType InvestorID;
///用户代码
TUstpFtdcUserIDType UserID;
///合约代码
TUstpFtdcInstrumentIDType InstrumentID;
/// 买 卖 方 向
TUstpFtdcDirectionType Direction;
///交易系统返回的系统报价编号
TUstpFtdcQuoteSysIDType QuoteSysID;
///用户设定的本地报价编号
TUstpFtdcUserQuoteLocalIDType UserQuoteLocalID;
///飞马向交易系统报的本地报价编号
TUstpFtdcQuoteLocalIDType QuoteLocalID;
///买方买入数量
TUstpFtdcVolumeType BidVolume;
/// 买 方 开 平 标 志
TUstpFtdcOffsetFlagType BidOffsetFlag;
///买方投机套保标志
TUstpFtdcHedgeFlagType BidHedgeFlag;
///买方买入价格
TUstpFtdcPriceType BidPrice;
///卖方卖出数量
TUstpFtdcVolumeType AskVolume;
/// 卖 方 开 平 标 志
TUstpFtdcOffsetFlagType AskOffsetFlag;
///卖方投机套保标志
TUstpFtdcHedgeFlagType AskHedgeFlag;
///卖方卖出价格
TUstpFtdcPriceType AskPrice;
///业务单元
TUstpFtdcBusinessUnitType BusinessUnit;
///用户自定义域
TUstpFtdcCustomType UserCustom;
///拆分出来的买方用户本地报单编号
TUstpFtdcUserOrderLocalIDType BidUserOrderLocalID;
///拆分出来的卖方用户本地报单编号
TUstpFtdcUserOrderLocalIDType AskUserOrderLocalID;
///拆分出来的买方本地报单编号
TUstpFtdcOrderLocalIDType BidOrderLocalID;
///拆分出来的卖方本地报单编号
TUstpFtdcOrderLocalIDType AskOrderLocalID;
///询价编号
TUstpFtdcQuoteSysIDType ReqForQuoteID;
```

```

    /// 报 价 停 留 时 间 ( 秒 )
    TustpFtdcMeasureSecType StandByTime;
    /// 买方系统报单编号
    TustpFtdcQuoteSysIDType BidOrderSysID;
    /// 卖方系统报单编号
    TustpFtdcQuoteSysIDType AskOrderSysID;
    /// 报价单状态
    TustpFtdcQuoteStatusType QuoteStatus;
    /// 插入时间
    TustpFtdcTimeType InsertTime;
    /// 撤销时间
    TustpFtdcTimeType CancelTime;
    /// 成交时间
    TustpFtdcTimeType TradeTime;
    /// 客 户 编 码
    TustpFtdcClientIDType ClientID;

    /// 下单用户编号
    TustpFtdcUserIDType QuoteUserID;
};

```

## 6.2.48 OnRtnExecOrder 方法

行权回报通知。与报单通知类似

**函数原形:**

```

void OnRtnExecOrder(
    CUSTPFTDCExecOrderField *pExecOrder)

```

**参数:**

pExecOrder: 指向行权通知结构的地址。

行权通知结构:

```

struct CUSTPFTDCExecOrderField
{
    /// 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;
/// 委 托 类 型
TUstpFtdcOrderTypeType    OrderType;
///期权对冲标识
TUstpFtdcDeliveryFlagType    DeliveryFlag;
///投机套保标志
TUstpFtdcHedgeFlagType    HedgeFlag;
///数量
TUstpFtdcVolumeType    Volume;
///用户自定义域
TUstpFtdcCustomType    UserCustom;
///业务发生日期
TUstpFtdcDateType    ActionDay;
///本地业务标识
TUstpFtdcBusinessLocalIDType    BusinessLocalID;
///业务单元
TUstpFtdcBusinessUnitType    BusinessUnit;
///交易日
TUstpFtdcTradingDayType    TradingDay;
///会员编号
TUstpFtdcParticipantIDType    ParticipantID;
///最初下单用户编号
TUstpFtdcUserIDType    OrderUserID;
/// 客 户 编 码
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;
};
```

#### 6.2.49 OnRtnForQuote 方法

询价回报通知。

### 函数原形：

```
void OnRtnForQuote(
    CUstpFtdcReqForQuoteField *pReqForQuote)
```

### 参数：

pReqForQuote:指向询价请求结构的地址。

询价请求结构：

```
struct CUstpFtdcReqForQuoteField
{
    ///询价编号
    TUstpFtdcQuoteSysIDType ReqForQuoteID;
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///合约代码
    TUstpFtdcInstrumentIDType InstrumentID;
    /// 买 卖 方 向
    TUstpFtdcDirectionType Direction;
    ///交易日
    TUstpFtdcDateType TradingDay;
    ///询价时间
    TUstpFtdcTimeType ReqForQuoteTime;
    ///客户编码
    TUstpFtdcClientIDType ClientID;
};
```

## 6.2.50 OnRtnInstrumentStatus 方法

合约交易状态通知。当合约状态发生变化时，由飞马平台主动通知客户端，该方法会被调用。

### 函数原形：

```
void
版权所有©上海金融期货信息技术有限公司
```

```
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;
///币种代码
TUstpFtdcCurrencyIDType CurrencyID;
///策略类别
TUstpFtdcArbiTypeType ArbiType;
///第一腿合约代码
TUstpFtdcInstrumentIDType InstrumentID_1;
///第一腿买卖方向
TUstpFtdcDirectionType Direction_1;
///第一腿数量比例
TUstpFtdcRatioType Ratio_1;
///第二腿合约代码
TUstpFtdcInstrumentIDType InstrumentID_2;
///第二腿买卖方向
TUstpFtdcDirectionType Direction_2;
///第二腿数量比例
TUstpFtdcRatioType Ratio_2;
///进入本状态日期
TUstpFtdcDateType EnterDate;
};
```

### 6.2.51 OnRtnMarginCombinationLeg 方法

组合规则变化通知。当组合策略规则发生变化时，由飞马平台主动通知客户端，该方法会被调用。

#### 函数原形：

```
void OnRtnMarginCombinationLeg(  
    CUstpFtdcMarginCombinationLegField *pMarginCombinationLeg);
```

#### 参数：

pMarginCombinationLeg: 指向组合策略规则结构的地址。

组合策略规则结构：

```
///组合规则  
struct CUstpFtdcMarginCombinationLegField  
{  
    ///交易所代码  
    TUstpFtdcExchangeIDType ExchangeID;  
    ///组合合约代码  
    TUstpFtdcCombInstrumentIDType CombInstrumentID;  
    ///单腿编号  
    TUstpFtdcLegIDType LegID;  
    ///单腿合约代码  
    TUstpFtdcInstrumentIDType LegInstrumentID;  
    /// 买 卖 方 向  
    TUstpFtdcDirectionType Direction;  
    ///单腿乘数  
    TUstpFtdcLegMultipleType LegMultiple;  
    ///优先级  
    TUstpFtdcPriorityType Priority;  
};
```

### 6.2.52 OnRtnMarginCombAction 方法

客户申请组合确认。

**函数原形:**

```
void OnRtnMarginCombAction(  
    CUstpFtdcInputMarginCombActionField * pInputMarginCombAction);
```

**参数:**

pInputMarginCombAction:指向组合操作请求结构的地址。

组合操作请求结构:

```
struct CUstpFtdcInputMarginCombActionField  
{  
    ///经纪公司编号  
    TUstpFtdcBrokerIDType    BrokerID;  
    ///交易所代码  
    TUstpFtdcExchangeIDType  ExchangeID;  
    ///交易用户代码  
    TUstpFtdcUserIDType      UserID;  
    ///投资者编号  
    TUstpFtdcInvestorIDType   InvestorID;  
    ///投机套保标志  
    TUstpFtdcHedgeFlagType    HedgeFlag;  
    ///用户本地编号  
    TUstpFtdcUserOrderLocalIDType  UserActionLocalID;  
    ///组合合约代码  
    TUstpFtdcInstrumentIDType  CombInstrumentID;  
    ///组 合 数 量  
    TUstpFtdcVolumeType        CombVolume;  
    ///组合动作方向  
    TUstpFtdcCombDirectionType CombDirection;  
    ///本地编号  
    TUstpFtdcOrderLocalIDType   ActionLocalID;  
    ///组合持仓方向  
    TUstpFtdcDirectionType      Direction;  
    ///系 统 编 号  
    TUstpFtdcOrderSysIDType      OrderSysID;  
    ///组合操作状态  
    TUstpFtdcCombActionStatusType  CombActionStatus;  
};
```

版权所有©上海金融期货信息技术有限公司

### 6.2.53 OnRtnInvestorAccountDeposit 方法

出入金结果回报。当柜台端发起了出入金时，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;  
};
```

### 6.2.54 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;  
    ///指定下单席位编号 (取值范围[1-N], N 为可用席位数目, 超出范围的随机分配席位)  
    TUstpFtdcSeatNoType SeatNo;  
    ///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;
};

```

### 6.2.55 OnErrRtnOrderAction 方法

报单操作错误回报。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:指向报单操作结构的地址，包含了报单操作请求的输入数据。

报单操作结构:

```

struct 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;
};

```

### 6.2.56 OnErrRtnQuoteInsert 方法

报价录入错误回报。由飞马平台主动通知客户端，该方法会被调用。

#### 函数原形:

```

void OnErrRtnQuoteInsert (
    CUstpFtdcInputQuoteField *pInputQuote,
    CUstpFtdcRspInfoField *pRspInfo);

```

#### 参数:

pInputQuote:指向报价录入错误回报结构的地址，包含了报价操作请求的输入数据。

报价录入错误回报结构:

```

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

```

```

TUstpFtdcInstrumentIDType  InstrumentID;
///交易系统返回的系统报价编号
TUstpFtdcQuoteSysIDType  QuoteSysID;
///用户设定的本地报价编号
TUstpFtdcUserQuoteLocalIDType  UserQuoteLocalID;
///飞马向交易系统报的本地报价编号
TUstpFtdcQuoteLocalIDType  QuoteLocalID;
///买方买入数量
TUstpFtdcVolumeType  BidVolume;
/// 买 方 开 平 标 志
TUstpFtdcOffsetFlagType  BidOffsetFlag;
///买方投机套保标志
TUstpFtdcHedgeFlagType  BidHedgeFlag;
///买方买入价格
TUstpFtdcPriceType  BidPrice;
///卖方卖出数量
TUstpFtdcVolumeType  AskVolume;
/// 卖 方 开 平 标 志
TUstpFtdcOffsetFlagType  AskOffsetFlag;
///卖方投机套保标志
TUstpFtdcHedgeFlagType  AskHedgeFlag;
///卖方卖出价格
TUstpFtdcPriceType  AskPrice;
///业务单元
TUstpFtdcBusinessUnitType  BusinessUnit;
///用户自定义域
TUstpFtdcCustomType  UserCustom;
///拆分出来的买方用户本地报单编号
TUstpFtdcUserOrderLocalIDType  BidUserOrderLocalID;
///拆分出来的卖方用户本地报单编号
TUstpFtdcUserOrderLocalIDType  AskUserOrderLocalID;
///询价编号
TUstpFtdcQuoteSysIDType  ReqForQuoteID;
/// 报 价 停 留 时 间 ( 秒 )
TUstpFtdcMeasureSecType  StandByTime;
/// 客 户 编 码
TUstpFtdcClientIDType  ClientID;
};
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;
};

```

### 6.2.57 OnErrRtnQuoteAction 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:

```

///报价撤单错误回报
void OnErrRtnQuoteAction(
    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。

报价操作结构:

//报价操作

```

struct CUstpFtdcQuoteActionField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///交易系统返回的系统报价编号
    TUstpFtdcQuoteSysIDType QuoteSysID;
    ///用户设定的被撤的本地报价编号
    TUstpFtdcUserQuoteLocalIDType UserQuoteLocalID;
    ///用户向飞马报的本地撤消报价编号
    TUstpFtdcUserQuoteLocalIDType UserQuoteActionLocalID;
    ///报单操作标志
    TUstpFtdcActionFlagType ActionFlag;
    ///业务单元
    TUstpFtdcBusinessUnitType BusinessUnit;
    ///用户自定义域
    TUstpFtdcCustomType UserCustom;
    ///买卖方向
    TUstpFtdcDirectionType Direction;
    ///客户编码
    TUstpFtdcClientIDType ClientID;

```

```
};
```

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;
};
```

## 6.2.58 OnErrRtnExecOrderInsert 方法

行权录入错误回报。由飞马平台主动通知客户端，该方法会被调用。

**函数原形:**

```
void
    OnErrRtnExecOrderInsert( CUstpFtdcExecInputOrderField *pInputExecOrder,
        CUstpFtdcRspInfoField * pRspInfo);
```

**参数:**

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

输入行权结构:

```
struct CUstpFtdcInputExecOrderField
{
    ///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;
    ///委托类型
    TUstpFtdcOrderTypeType OrderType;
    ///期权对冲标识
```

```

    TUstpFtdcDeliveryFlagType   DeliveryFlag;
    ///投机套保标志
    TUstpFtdcHedgeFlagType   HedgeFlag;
    ///数量
    TUstpFtdcVolumeType   Volume;
    ///用户自定义域
    TUstpFtdcCustomType   UserCustom;
    ///业务发生日期
    TUstpFtdcDateType   ActionDay;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType   BusinessLocalID;
    ///业务单元
    TUstpFtdcBusinessUnitType   BusinessUnit;
};
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;
};

```

## 6.2.59 OnErrRtnExecOrderAction 方法

行权操作错误回报。由飞马平台主动通知客户端，该方法会被调用。

**函数原形:**

```

void OnErrRtnExecOrderAction
( CUstpFtdcExecOrderActionField
 *pExecOrderAction, CUstpFtdcRspInfoField
 *pRspInfo);

```

**参数:**

pExecOrderAction:指向行权操作结构的地址，包含了行权操作请求的输入数据。

报单操作结构:

```

struct CUstpFtdcInputExecOrderActionField
{
    ///交易所代码
    TUstpFtdcExchangeIDType   ExchangeID;
    /// 报 单 编 号
    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 操 作 的 本 地 编 号
    TUstpFtdcUserOrderLocalIDType UserOrderActionLocalID;
    ///被撤订单的本地报单编号
    TUstpFtdcUserOrderLocalIDType UserOrderLocalID;
    ///报单操作标志
    TUstpFtdcActionFlagType ActionFlag;
    ///数量变化
    TUstpFtdcVolumeType VolumeChange;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
    /// 委 托 类 型
    TUstpFtdcOrderTypeType OrderType;
};

```

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 CUstpFtdcTraderApi *CreateFtdcTraderApi(const char *pszFlowPath = "", const bool
bRunAtMaximalSpeed = false);

```

**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。

bRunAtMaximalSpeed: 运行模式变量, 用于指定API所在线程是否开启全速运行模式, 全速模式

下API性能提升，CPU使用率将达到100%。默认值不开启全速模式。

返回值：

返回一个指向 CUstpFtdcTradeApi 实例的指针。

### 6.3.2 GetVersion 方法

获取系统版本号。

**函数原形：**

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

**参数：**

nMajorVersion:主版本号。

nMinorVersion:子版本号

返回值：

返回一个飞马版本描述字符串。

### 6.3.3 Release 方法

释放一个 CUstpFtdcTradeApi 实例。不能使用 delete 方法

**函数原形：**

```
void Release();
```

### 6.3.4 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();
```

### 6.3.5 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();
```

### 6.3.6 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。

### 6.3.7 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。

### 6.3.8 RegisterFront 方法

设置飞马平台的交易前置机网络通讯地址，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。

该方法要在 Init 方法之前调用。

**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。

### 6.3.9 RegisterQryFront 方法

设置飞马平台的查询前置机网络通讯地址，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。

该方法要在 Init 方法之前调用。

**函数原形:**

```
void RegisterQryFront(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。

### 6.3.10 RegisterNameServer 方法（未启用）

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

### 6.3.11 RegisterCertificateFile 方法（未启用）

订阅证书。

#### 函数原形：

```
int  
  
RegisterCertificateFile( con  
st char *pszCertFileName,  
const char *pszKeyFileName,  
const char *pszCaFileName,  
const char *pszKeyFilePassword)
```

#### 参数：

pszCertFileName:用户证书文件名

pszKeyFileName:用户私钥文件名

pszCaFileName:可信任 CA 证书文件名

pszKeyFilePassword:用户私钥文件密码

#### 返回值：

返回操作结果。

- 0 操作成功
- 1 可信任 CA 证书载入失败
- 2 用户证书载入失败
- 3 用户私钥载入失败
- 4 用户证书校验失败

### 6.3.12 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

**6.3.13 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

**6.3.14 SubscribeUserTopic 方法（未启用）**

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

**函数原形:**

```
void SubscribeUserTopic(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

**6.3.15 SubscribeForQuote 方法**

订阅询价流。该方法要在 Init 方法前调用。若不调用则不会收到询价流的数据。

**函数原形:**

```
void SubscribeForQuote(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

版权所有©上海金融期货信息技术有限公司

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

### 6.3.16 SetHeartbeatTimeout 方法

设置心跳超时时间。

**函数原形:**

```
void SetHeartbeatTimeout(unsigned int timeout);
```

**参数:**

timeout: 心跳超时时间, 单位秒。

### 6.3.17 ReqDSUserCertification 方法

穿透监管客户认证请求

**函数原形:**

```
int
    ReqDSUserCertification( CUSTpFtdcDSUse
        rInfoField *pDSUserInfo, int
        nRequestID)
```

**参数:**

穿透监管客户信息结构

```
struct CUSTpFtdcDSUserInfoField
{
    /// 用 户 AppID
    TUSTpFtdcDSAppIDType   AppID;
    ///用户授权码
    TUSTpFtdcDSAuthorizationCodeType   AuthCode;
    ///密钥加密类型
    TUSTpFtdcDSKeyEncryptType   EncryptType;
};
```

### 6.3.18 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;
    ///接口端产品信息一占位字段, API 自动填写
    TUstpFtdcProductInfoType    InterfaceProductInfo;
    ///IP 地址一占位字段, API 自动填写
    TUstpFtdcIPAddressType  IPAddress;
    ///Mac 地址一占位字段, API 自动填写
    TUstpFtdcMacAddressType MacAddress;
    ///协议信息一占位字段, API 自动填写
    TUstpFtdcProtocolInfoType  ProtocolInfo;
    ///数据中心代码
    TUstpFtdcDataCenterIDType  DataCenterID;
    ///客户端运行文件大小
    TUstpFtdcFileSizeType    UserProductFileSize;
    ///客户认证类型
    TUstpFtdcAuthenticate2TypeType  Authenticate2Type;
    ///客户认证密码
    TUstpFtdcAuthenticate2PasswordType  Authenticate2Password;
    ///开发厂商终端编码
    TUstpFtdcTerminalCodeType  TerminalCode;
    ///密码加密类型, 应填” ”
    TUstpFtdcPasswordEncryptType  PasswordEncrypt;
};
```

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。

### 6.3.19 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;  
};
```

### 6.3.20 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;  
};
```

### 6.3.21 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, 必填字段  
    TUstpFtdcBrokerIDType BrokerID;  
    ///交易所代码, 必填字段 (中金所:CFEX)  
    TUstpFtdcExchangeIDType ExchangeID;  
    ///系统报单编号  
    TUstpFtdcOrderSysIDType OrderSysID;  
    ///投资者编号, 必填字段 (客户资金账户)  
    TUstpFtdcInvestorIDType InvestorID;  
    ///用户代码, 必填字段 (飞马用户代码)  
    TUstpFtdcUserIDType UserID;  
    ///指定下单席位编号 (取值范围[1-N], N 为可用席位数目, 超出范围的随机分配席位)  
    TUstpFtdcSeatNoType SeatNo;  
    ///合约代码, 必填字段  
    TUstpFtdcInstrumentIDType InstrumentID;  
    ///用户本地报单号, 必填字段  
    TUstpFtdcUserOrderLocalIDType UserOrderLocalID;  
    ///报单价格条件, 必填字段 (1:市价; 2:限价; 3:最优市价; 4:五  
    档市价)  
    TUstpFtdcOrderPriceTypeType OrderPriceType;  
    ///买卖方向, 必填字段 (0:买; 1:卖)  
    TUstpFtdcDirectionType Direction;  
    ///开平标志, 必填字段 (0:开仓; 1:平仓; 3:平今; 4:平昨)  
    TUstpFtdcOffsetFlagType OffsetFlag;  
    ///投机套保标志, 必填字段 (1:投机; 2:套利; 3:套保; 4:做市商)  
    TUstpFtdcHedgeFlagType HedgeFlag;  
    ///价格, 必填字段  
    TUstpFtdcLimitPriceType LimitPrice;
```

```

    ///数量, 必填字段
    TUstpFtdcVolumeType Volume;

    ///有效期类型, 必填字段 1:IOC单; 3:当日有效)
    TUstpFtdcTimeConditionType TimeCondition;

    ///GTD 日期 (暂不支持, 保留域)
    TUstpFtdcDateType GTDDate;

    ///成交量类型, 必填字段 (1:任意数量 3:全部数量)
    TUstpFtdcVolumeConditionType VolumeCondition;

    ///最小成交量 (暂不支持, 保留域)
    TUstpFtdcVolumeType MinVolume;

    ///止损价
    TUstpFtdcPriceType StopPrice;

    /// 强 平 原 因 , 只 支 持 " 0:非 强 平 "
    TUstpFtdcForceCloseReasonType ForceCloseReason;

    ///自动挂起标志 (暂不支持, 保留域)
    TUstpFtdcBoolType IsAutoSuspend;

    ///业务单元 (暂不支持, 保留域)
    TUstpFtdcBusinessUnitType BusinessUnit;

    ///用户自定义域 64 字节
    TUstpFtdcCustomType UserCustom;

    ///本地业务标识 (暂不支持, 保留域)
    TUstpFtdcBusinessLocalIDType BusinessLocalID;

    ///业务发生日期
    TUstpFtdcDateType ActionDay;

    ///策略类别 (0:普通单; 1-3, a-h:套
    利单) TUstpFtdcArbiTypeType
        ArbiType;

    /// 客 户 编 码
    TUstpFtdcClientIDType ClientID;
};

```

**注意:**

用户本地报单号 UserOrderLocalID 是一个 21 位的由数字组成的字符串, 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), 其比较方式为数字比较。

*MaxOrderLocalID < UserOrderLocalID*

**普通限价单**必须填充的字段包括:

1. BrokerID, 会员号, 形如 "2008";
2. ExchangeID, 交易所代码, "CFFEX"
3. InvestorID, 投资者编号, 形如 "10000029";



4. UserID, 用户代码, 形如 “test1”;
5. InstrumentID, 合约代码, 形如 “IF1109”;
6. **OrderPriceType, 报单价格条件, 限价单 USTP\_FTDC\_OPT\_LimitPrice;**
7. Direction, 买卖方向, USTP\_FTDC\_D\_Buy 表示买, USTP\_FTDC\_D\_Sell 表示卖;
8. OffsetFlag, 开平标志, “0” 表示开仓, “1” 表示平仓;
9. HedgeFlag, 投机套保标志, “1” 为投机, “2” 为套保, “3” 为套利, “4” 为做市商;
10. LimitPrice, 价格, 形如 3500.00;
11. Volume, 数量, 例如 5 表示 5 手;
12. **TimeCondition, 有效期类型, 为 USTP\_FTDC\_TC\_GFD (“当日有效”);**
13. VolumeCondition, 成交量类型, 为 USTP\_FTDC\_VC\_AV (“任意数量”);
14. **ArbiType, 套利类型, 为 USTP\_FTDC\_AT\_Basic (“普通单”)**
15. ContingentCondition, 触发条件, 为 USTP\_FTDC\_CC\_Immediately (“立即”);
16. ForceCloseReason, 强平原因, 为 USTP\_FTDC\_FCC\_NotForceClose (“非强平”);
17. UserOrderLocalID, 用户本地报单编号, 形如 “00000025”。

普通市价单必须填充的字段包括:

- a) BrokerID, 会员号, 形如 “2008”;
- b) ExchangeID, 交易所代码, “CFFEX”
- c) InvestorID, 客户号, 形如 “10000029”;
- d) UserID, 交易用户代码, 形如 “test1”;
- e) InstrumentID, 合约代码, 形如 “IF1109”;
- f) **OrderPriceType, 报单价格条件, 市价单为 USTP\_FTDC\_OPT\_AnyPrice;**
- g) Direction, 买卖方向, USTP\_FTDC\_D\_Buy 表示买, USTP\_FTDC\_D\_Sell 表示卖;
- h) OffsetFlag, 开平标志, “0” 表示开仓, “1” 表示平仓;
- i) HedgeFlag, 投机套保标志, “1” 为投机, “2” 为套保, “3” 为套利, “4” 为做市商;
- j) Volume, 数量, 例如 5 表示 5 手;
- k) **TimeCondition, 有效期类型, 为 USTP\_FTDC\_TC\_IOC (“立即成交, 否则撤销”);**
- l) VolumeCondition, 成交量类型, 为 USTP\_FTDC\_VC\_AV (“任意数量”);
- m) **ArbiType, 套利类型, 为 USTP\_FTDC\_AT\_Basic (“普通单”)**
- n) ContingentCondition, 触发条件, 只能为 USTP\_FTDC\_CC\_Immediately (“立即”);
- o) ForceCloseReason, 强平原因, 只能为 USTP\_FTDC\_FCC\_NotForceClose (“非强平”);
- p) UserOrderLocalID, 本地报单编号, 形如 “00000025”。

期货套利组合单必须填充的字段包括:

- 1 BrokerID, 会员号, 形如 “2008”;
- 2 **ExchangeID**, 交易所代码, “**DCE/ZCE**”
- 3 InvestorID, 客户号, 形如 “10000029”;
- 4 UserID, 交易用户代码, 形如 “test1”;

- 5 **InstrumentID**, 合约代码, 形如 “SP jm1301&jm1303/SPC a1301&m1301/SPD SR609&SR611”;
- 6 **OrderPriceType**, 报单价格条件, 只能为 USTP\_FTDC\_OPT\_AnyPrice;
- 7 **Direction**, 买卖方向, USTP\_FTDC\_D\_Buy 表示买, USTP\_FTDC\_D\_Sell 表示卖;
- 8 **OffsetFlag**, 开平标志, “0” 表示开仓, “1” 表示平仓;
- 9 **HedgeFlag**, 投机套保标志, 套利组合单类型只能为 “3” - 套利;
- 10 **Volume**, 数量, 例如 5 表示 5 手;
- 11 **TimeCondition**, 有效期类型, 为 USTP\_FTDC\_TC\_GFD (“当日有效”);
- 12 **VolumeCondition**, 成交量类型, 为 USTP\_FTDC\_VC\_AV (“任意数量”);
- 13) **ArbiType**, 套利类型, 只支持 USTP\_FTDC\_AT\_SP (“跨期套利”) 或者 USTP\_FTDC\_AT\_SPC (“跨品种套利”);
- 14 **ContingentCondition**, 触发条件, 只能为 USTP\_FTDC\_CC\_Immediately (“立即”);
- 15 **ForceCloseReason**, 强平原因, 只能为 USTP\_FTDC\_FCC\_NotForceClose (“非强平”);
- 16 **UserOrderLocalID**, 本地报单编号, 形如“00000025”。

期权组合单必须填充的字段包括:

- 1 **BrokerID**, 会员号, 形如 “2008”;
- 2 **ExchangeID**, 交易所代码, “ZCE”
- 3 **InvestorID**, 客户号, 形如 “10000029”;
- 4 **UserID**, 交易用户代码, 形如 “test1”;
- 5 **InstrumentID**, 合约代码, 形如 “SP jm1301&jm1303/SPC a1301&m1301/SPD SR609&SR611”;
- 6 **OrderPriceType**, 报单价格条件, 只能为 USTP\_FTDC\_OPT\_AnyPrice;
- 7 **Direction**, 买卖方向, USTP\_FTDC\_D\_Buy 表示买, USTP\_FTDC\_D\_Sell 表示卖;
- 8 **OffsetFlag**, 开平标志, “0” 表示开仓, “1” 表示平仓;
- 9) **HedgeFlag**, 投机套保标志, 套利组合单类型只能为 “3” - 套利;
- 10 **Volume**, 数量, 例如 5 表示 5 手;
- 11 **TimeCondition**, 有效期类型, 为 USTP\_FTDC\_TC\_GFD (“当日有效”);
- 12 **VolumeCondition**, 成交量类型, 为 USTP\_FTDC\_VC\_AV (“任意数量”);
- 13) **ArbiType**, 套利类型, 只支持 USTP\_FTDC\_AT\_SP (“跨期套利”) 或者 USTP\_FTDC\_AT\_SPC (“跨品种套利”);
- 14 **ContingentCondition**, 触发条件, 只能为 USTP\_FTDC\_CC\_Immediately (“立即”);
- 15 **ForceCloseReason**, 强平原因, 只能为 USTP\_FTDC\_FCC\_NotForceClose (“非强平”);

⑤ UserOrderLocalID, 本地报单编号, 形如"00000025"。

中金所支持 9 种订单类型, 报单时 ExchangeID 和 InstrumentID 应填写中金所交易所代码和合约代码, 相关详细说明如下:

限价单	USTP_FTDC_OPT_Li	USTP_FTDC_TC_GFD	USTP_FTDC_VC_AV	当日有效, 限价成交
订单类型	OrderPriceType	TimeCondition	VolumeCondition	备注
FOK	USTP_FTDC_OPT_Li mitPrice	USTP_FTDC_TC_IOC	USTP_FTDC_VC_CV	限价立即全部成交否则自动撤销
FAK	USTP_FTDC_OPT_Li mitPrice	USTP_FTDC_TC_IOC	USTP_FTDC_VC_AV	限价立即成交剩余自动撤销
市价单	USTP_FTDC_OPT_An yPrice	USTP_FTDC_TC_IOC	USTP_FTDC_VC_AV	市价立即成交否则撤单
市价转限价	USTP_FTDC_OPT_An yPrice	USTP_FTDC_TC_GFD	USTP_FTDC_VC_AV	市价单未成交部分转为最新价限价单
五档市价	USTP_FTDC_OPT_Fi veLevelPrice	USTP_FTDC_TC_IOC	USTP_FTDC_VC_AV	市价单与对手方五档价格报单尝试成交, 剩余未成交部分撤销
五档市价转 限价	USTP_FTDC_OPT_Fi veLevelPrice	USTP_FTDC_TC_GFD	USTP_FTDC_VC_AV	市价单与对手方五档价格报单尝试成交, 剩余未成交部分转为最新价限价单
最优价	USTP_FTDC_OPT_Be stPrice	USTP_FTDC_TC_IOC	USTP_FTDC_VC_AV	市价单与对手方最优一档价格报单尝试成交, 剩余未成交部分撤销
最优价转限 价	USTP_FTDC_OPT_Be stPrice	USTP_FTDC_TC_GFD	USTP_FTDC_VC_AV	市价单与对手方最优一档价格报单尝试成交, 剩余未成交部分转为最新价限价单

FOK 单和 FAK 单为限价单, 除这 3 个字段外的其他字段填充方式参照限价单。

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

### 6.3.22 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, 必填字段
    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, 必填字段
    TUstpFtdcBrokerIDType BrokerID;
    ///投资者编号, 必填字段
    TUstpFtdcInvestorIDType InvestorID;
    ///用户代码, 必填字段
    TUstpFtdcUserIDType UserID;
    ///本次撤单 Req 的本地编号，必填字段 （数字按序递增）
    TUstpFtdcUserOrderLocalIDType UserOrderActionLocalID;
    ///被撤订单的本地报单编号
    TUstpFtdcUserOrderLocalIDType UserOrderLocalID;
    ///报单操作标志，必填字段（只支持删除）
    TUstpFtdcActionFlagType ActionFlag;
    ///价格 （暂不支持，保留域）
    TUstpFtdcPriceType LimitPrice;
    ///数量变化（暂不支持，保留域）
    TUstpFtdcVolumeType VolumeChange;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
    ///客 户 编 码
    TUstpFtdcClientIDType ClientID;
};

```

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。

**撤单必须填充的字段包括:**

- 1 BrokerID, 会员号, 形如 “2008”;
- 2 ExchangeID, 交易所代码, 形如 “CFFEX”
- 3 InvestorID, 投资者编号, 形如 “10000029”;
- 4 UserID, 用户代码, 形如 “test1”;
- 5 OrderSysID 或者 UserOrderLocalID 任选其一, 被撤订单的订单编号, 优先按照 OrderSysID 定位, 形如 “00000025”;
- 6 UserOrderActionLocalID, 撤单请求的本地编号, 与 UserOrderLocalID 同一个序列按字典序增加, 形如 “00000025”;
- 7 ActionFlag, 操作标志, 只能填 **USTP\_FTDC\_AF\_Delete (“删除”)**;

### 6.3.23 ReqQuoteInsert 方法

报价录入请求，做市商进行双边报价的接口

**函数原形：**

```
int ReqQuoteInsert
( CUstpFtdcInputQuoteField
  *pInputQuote, int nRequestID)
```

**参数：**

pInputQuote 指报价录入结构的地址。

报价录入结构

```
struct CUstpFtdcInputQuoteField
{
    /// 经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///合约代码
    TUstpFtdcInstrumentIDType InstrumentID;
    /// 买卖方向
    TUstpFtdcDirectionType Direction;
    ///交易系统返回的系统报价编号
    TUstpFtdcQuoteSysIDType QuoteSysID;
    ///用户设定的本地报价编号
    TUstpFtdcUserQuoteLocalIDType UserQuoteLocalID;
    ///飞马向交易系统报的本地报价编号
    TUstpFtdcQuoteLocalIDType QuoteLocalID;
    ///买方买入数量
    TUstpFtdcVolumeType BidVolume;
    /// 买方开平标志
    TUstpFtdcOffsetFlagType BidOffsetFlag;
    ///买方投机套保标志
    TUstpFtdcHedgeFlagType BidHedgeFlag;
    ///买方买入价格
    TUstpFtdcPriceType BidPrice;
    ///卖方卖出数量
    TUstpFtdcVolumeType AskVolume;
    /// 卖方开平标志
    TUstpFtdcOffsetFlagType AskOffsetFlag;
    ///卖方投机套保标志
    TUstpFtdcHedgeFlagType AskHedgeFlag;
```

```

    ///卖方卖出价格
    TustpFtdcPriceType  AskPrice;
    ///业务单元
    TustpFtdcBusinessUnitType  BusinessUnit;
    ///用户自定义域
    TustpFtdcCustomType  UserCustom;
    ///拆分出来的买方用户本地报单编号
    TustpFtdcUserOrderLocalIDType  BidUserOrderLocalID;
    ///拆分出来的卖方用户本地报单编号
    TustpFtdcUserOrderLocalIDType  AskUserOrderLocalID;
    ///拆分出来的买方本地报单编号
    TustpFtdcOrderLocalIDType  BidOrderLocalID;
    ///拆分出来的卖方本地报单编号
    TustpFtdcOrderLocalIDType  AskOrderLocalID;
    ///询价编号
    TustpFtdcQuoteSysIDType  ReqForQuoteID;
    /// 报 价 停 留 时 间 ( 秒 )
    TustpFtdcMeasureSecType  StandByTime;
};

```

**报价单**必须填充的字段包括:

1. BrokerID, 会员号, 形如 “2008”;
2. ExchangeID, 交易所代码, 只支持 “CFFEX”
3. InvestorID, 投资者编号, 形如 “10000029”;
4. UserID, 用户代码, 形如 “test1”;
5. InstrumentID, 合约代码, 形如 “IF1109”;
6. **UserQuoteLocalID, 用户本地报价单号, 形如 “000000101”;**
7. BidVolume, 买方买入数量, 例如 5 表示 5 手;
8. BidOffsetFlag, 买方开平标志, “0” 表示开仓, “1” 表示平仓;
9. BidHedgeFlag, 买方投机套保标志, 必须跟交易编码类型一致;
10. BidPrice, 价格, 形如 3510.00;
11. AskVolume, 买方买入数量, 例如 5 表示 5 手;
12. AskOffsetFlag, 买方开平标志, “0” 表示开仓, “1” 表示平仓;
13. AskHedgeFlag, 买方投机套保标志, 必须跟交易编码类型一致;
14. AskPrice, 价格, 形如 3500.00;
15. ReqForQuoteID, 询价编号
16. **BidUserOrderLocalID, 报价单中买单的用户本地报单号, 例如 “000000102”;**

17. AskUserOrderLocalID, 报价单中卖单的用户本地报单号, 例如“000000103”。

注意:  $MaxOrderLocalID < UserQuoteLocalID < BidUserOrderLocalID < AskUserOrderLocalID$

### 6.3.24 ReqQuoteAction 方法

报价操作请求, 撤销某一笔报价, 仅做市商有权限进行此操作。

**函数原形:**

```
int ReqQuoteAction
(
    CUSTpFtdcQuoteActionField
    *pQuoteAction, int nRequestID)

```

**参数:**

pQuoteAction 指报价操作结构的地址。

报价操作结构

```
struct CUSTpFtdcQuoteActionField
{
    ///经纪公司编号
    TUSTpFtdcBrokerIDType BrokerID;
    ///交易所代码
    TUSTpFtdcExchangeIDType ExchangeID;
    ///投资者编号
    TUSTpFtdcInvestorIDType InvestorID;
    ///用户代码
    TUSTpFtdcUserIDType UserID;
    ///交易系统返回的系统报价编号
    TUSTpFtdcQuoteSysIDType QuoteSysID;
    ///用户设定的被撤的本地报价编号
    TUSTpFtdcUserQuoteLocalIDType UserQuoteLocalID;
    ///用户向飞马报的本地撤消报价编号
    TUSTpFtdcUserQuoteLocalIDType UserQuoteActionLocalID;
    ///报单操作标志
    TUSTpFtdcActionFlagType ActionFlag;
    ///业务单元
    TUSTpFtdcBusinessUnitType BusinessUnit;
    ///用户自定义域
    TUSTpFtdcCustomType UserCustom;
};

```

**撤销报价单**必须填充的字段包括:

- 1 BrokerID, 会员号, 形如“2008”;
- 2 ExchangeID, 交易所代码, 只支持“CFFEX”
- 3 InvestorID, 投资者编号, 形如“10000029”;

- 4 UserID, 用户代码, 形如 “test1”;
- 5 InstrumentID, 合约代码, 形如 “IF1109”;
- 6 QuoteSysID /UserQuoteLocalID, 要撤销报价单的系统报价单号/用户本地报价单号, 形如  
“000000101”。二者选其一即可, 优先按照系统报价单号撤销;
- 7 UserQuoteActionLocalID, 撤销报价操作的用户本地操作编号, 形如 “000000101”;

注意: *MaxOrderLocalID < UserQuoteActionLocalID*

### 6.3.25 ReqForQuote 方法

询价请求, 由客户发起的询价请求。针对某一个合约询价后, 做市商的报价在该合约的行情上反映。

#### 函数原形:

```
int ReqForQuote (
    CUstpFtdcReqForQuoteField *pReqForQuote,
    int nRequestID)
```

#### 参数:

pReqForQuote 指向询价结构的地址。

询价结构

```
struct CUstpFtdcReqForQuoteField
{
    ///询价编号
    TUstpFtdcQuoteSysIDType ReqForQuoteID;
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///用户代码
    TUstpFtdcUserIDType UserID;
    ///合约代码
    TUstpFtdcInstrumentIDType InstrumentID;
    ///交易日
    TUstpFtdcDateType TradingDay;
    ///询价时间
    TUstpFtdcTimeType ReqForQuoteTime;
};
```

询价单必须填充的字段包括:

- 8 BrokerID, 会员号, 形如 “2008”;
- 9 ExchangeID, 交易所代码, 只支持 “CFFEX”



- 0 InvestorID, 投资者编号, 形如 “10000029”;
- 1 UserID, 用户代码, 形如 “test1”;
- 2 InstrumentID, 合约代码, 形如 “IF1109”;

### 6.3.26 ReqExecOrderInsert 方法

行权录入请求此接口用于发起行权请求。使用方法与报单接口类似。通过 OrderType 选择是行权还是放弃行权, 通过 DeliveryFlag 选择是否需要对冲。

**函数原形:**

```
int ReqExecOrderInsert
( CUstpFtdcInputExecOrderField
  *pInputExecOrder,,
  int nRequestID)
```

**参数:**

CUstpFtdcInputExecOrderField指向行权请求的结构体

```
struct CUstpFtdcInputExecOrderField
{
    ///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;
    ///委托类型
    TUstpFtdcOrderTypeType   OrderType;
    ///期权对冲标识
    TUstpFtdcDeliveryFlagType   DeliveryFlag;
    ///投机套保标志
    TUstpFtdcHedgeFlagType   HedgeFlag;
    ///数量
    TUstpFtdcVolumeType   Volume;
    ///用户自定义域
```

```

    TUstpFtdcCustomType UserCustom;
    ///业务发生日期
    TUstpFtdcDateType ActionDay;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
    ///业务单元
    TUstpFtdcBusinessUnitType BusinessUnit;
};

```

### 6.3.27 ReqExecOrderAction 方法

行权操作请求，此接口用于对已经发起的行权请求进行撤销。

**函数原形：**

```

int ReqExecOrderAction (
    CUstpFtdcInputExecOrderActionField *pInputExecOrderAction,
    int nRequestID)

```

**参数：**

CUstpFtdcInputExecOrderActionField指向行权撤销的结构体

```

struct CUstpFtdcInputExecOrderActionField
{
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;

    /// 报 单 编 号
    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 操 作 的 本 地 编 号
    TUstpFtdcUserOrderLocalIDType UserOrderActionLocalID;
    ///被撤订单的本地报单编号
    TUstpFtdcUserOrderLocalIDType UserOrderLocalID;
    ///报单操作标志
    TUstpFtdcActionFlagType ActionFlag;
    ///数量变化
    TUstpFtdcVolumeType VolumeChange;
    ///本地业务标识
    TUstpFtdcBusinessLocalIDType BusinessLocalID;
    /// 委 托 类 型
    TUstpFtdcOrderTypeType OrderType;

```

```
};
```

### 6.3.28 ReqMarginCombAction 方法

策略组合/策略拆分请求，此接口用于对已有持仓进行组合，或者对已有组合进行拆分。  
该接口只能对中金所公布的策略组合规则表中的组合合约进行组合。

#### 函数原形：

```
int ReqMarginCombAction(
    CUstpFtdcInputMarginCombActionField *pInputMarginCombAction,
    int nRequestID)
```

#### 参数：

pInputMarginCombAction 指向策略组合结构的地址。

询价结构

```
struct CUstpFtdcInputMarginCombActionField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///交易用户代码
    TUstpFtdcUserIDType UserID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///投机套保标志
    TUstpFtdcHedgeFlagType HedgeFlag;
    ///用户本地编号
    TUstpFtdcUserOrderLocalIDType UserActionLocalID;
    ///组合合约代码
    TUstpFtdcCombInstrumentIDType CombInstrumentID;
    ///组合数量
    TUstpFtdcVolumeType CombVolume;
    ///组合动作方向
    TUstpFtdcCombDirectionType CombDirection;
    ///本地编号
    TUstpFtdcOrderLocalIDType ActionLocalID;
};
```

**策略组合/策略拆分**必须填充的字段包括：

- 1 BrokerID, 会员号, 形如 “2008”;
- 2 ExchangeID, 交易所代码, 只支持 “CFFEX”
- 3 InvestorID, 投资者编号, 形如 “10000029”;
- 4 UserID, 用户代码, 形如 “test1”;
- 5 HedgeFlag, 合约代码, 形如 “IF1109”;
- 6 **UserActionLocalID, 用户本地报价单号, 形如 “000000101”;**
- 7 CombInstrumentID, 组合合约代码, 形如 “2213”;
- 8 CombVolume, 组合数量, 形如 “1”;
- 9 CombDirection, 组合方向。“USTP\_FTDC\_CA\_Combine” 为组合, “USTP\_FTDC\_CA\_Uncombine” 为拆分组合

### 6.3.29 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;
    ///报单状态
    TUstpFtdcOrderStatusType OrderStatus;
    ///委托类型
    TUstpFtdcOrderTypeType OrderType;
    ///客户编码
```

```
TUstpFtdcClientIDType ClientID;
};
```

按投资者查询至少应该填充的字段包括:

- 1 BrokerID, 会员号, 形如“2008”;
- 2 InvestorID, 投资者编号, 形如“10000029”;
- 3 OrderStatus, 填写‘ ’(空格), 查询所有满足条件订单

按用户查询至少应该填充的字段包括:

- 1 BrokerID, 会员号, 形如“2008”;
- 2 UserID, 用户代码(登陆用户, 不能查询其他用户), 形如“test1”;
- 3 OrderStatus, 订单状态, 填写‘ ’(空格), 查询所有满足条件订单

按订单状态查询至少应该填充的字段包括:

- 1 BrokerID, 会员号, 形如“2008”;
- 2 UserID, 用户代码(登陆用户, 不能查询其他用户), 形如“test1”;

- 3 OrderStatus, 订单状态, 填写’ ’或者 null, 查询错单, 填写‘ ’(空格), 查询所有订单(包括错单)

按委托类型查询, 可以区分普通单(基础单和组合单)、行权单至少应该填充的字段包括:

- 1 BrokerID, 会员号, 形如“2008”;
- 2 UserID, 用户代码(登陆用户, 不能查询其他用户), 形如“test1”;
- 3 OrderStatus, 订单状态, 填写‘ ’(空格), 查询所有订单(包括错单)
- 4 OrderType, 填写相应的类型, 填写‘USTP\_FTDC\_OT\_Common’可查所有普通单(包括组合单),  
填写‘USTP\_FTDC\_OT\_OptExec’可查所有行权单, ‘USTP\_FTDC\_OT\_OptAbandon’可查所有  
放弃行权单。

### 6.3.30 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;
};

```

### 6.3.31 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;
};

```

### 6.3.32 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;
    ///客 户 编 码
    TUstpFtdcClientIDType ClientID;
};
```

### 6.3.33 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;
};
```

### 6.3.34 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;
};

```

### 6.3.35 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;
};

```

### 6.3.36 ReqQryInvestorPosition 方法

投资者持仓查询。

**函数原形：**

```

int 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;
};

```

### 6.3.37 ReqQryInvestorFee 方法

投资者手续费率查询。

**函数原形：**

```

int 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;
    ///客户编码
    TUstpFtdcClientIDType ClientID;
};

```

### 6.3.38 ReqQryInvestorMargin 方法

投资者保证金率查询

**函数原形：**

```

int 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;
    ///客户编码
    TUstpFtdcClientIDType   ClientID;
};
```

### 6.3.39 ReqQryQuote方法

报价查询请求

函数原形:

```
int ReqQryQuote
(CUstpFtdcQryQuoteField
*pQryQuote, int nRequestID)
```

参数:

pQryQuote 指报价查询结构的地址。

```
struct CUstpFtdcQryQuoteField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType   ExchangeID;
    ///投资者编号
    TUstpFtdcInvestorIDType   InvestorID;
    ///用户代码
    TUstpFtdcUserIDType   UserID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
    ///交易系统返回的系统报价编号
    TUstpFtdcQuoteSysIDType   QuoteSysID;
    ///报价单状态
    TUstpFtdcQuoteStatusType   QuoteStatus;
};
```

### 6.3.40 ReqQryInvestorCombPosition 方法

投资者组合持仓查询。

版权所有©上海金融期货信息技术有限公司

**函数原形:**

```
int ReqQryInvestorCombPosition(
    CUstpFtdcQryInvestorCombPositionField *pQryInvestorCombPosition,
    int nRequestID)
```

**参数:**

pQryUserInvestorPosition 指向投资者组合持仓查询结构的地址。

投资者组合持仓查询结构

```
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;
    ///组合合约代码
    TUstpFtdcCombInstrumentIDType CombInstrumentID;
    ///客户编码
    TUstpFtdcClientIDType   ClientID;
};
```

**6.3.41 ReqQryInvestorLegPosition 方法**

投资者单腿持仓查询。

**函数原形:**

```
int ReqQryInvestorLegPosition
    ( CUstpFtdcQryInvestorLegPositionField
    *pQryInvestorLegPosition, int nRequestID)
```

**参数:**

pQryInvestorLegPosition 指向投资者单腿持仓查询结构的地址。

投资者单腿持仓查询结构

```
struct CUstpFtdcQryInvestorLegPositionField
{
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///投资者编号
    TUstpFtdcInvestorIDType InvestorID;
    ///投机套保标志
    TUstpFtdcHedgeFlagType  HedgeFlag;
    版权所有©上海金融期货信息技术有限公司
```

```

///单腿合约代码
TUstpFtdcInstrumentIDType   LegInstrumentID;
///客户编码
TUstpFtdcClientIDType       ClientID;

```

### 6.3.42 }ReqQryInstrumentGroup 方法

查询合约组信息请求，属于同一个合约组的合约可以在合约组内采用大边的方式收取保证金。

#### 函数原形：

```

int ReqQryInstrumentGroup(
    CUstpFtdcQryInstrumentGroupField *pQryInstrumentGroup,
    int nRequestID)

```

#### 参数：

```

struct CUstpFtdcQryUstpInstrumentGroupField
{
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///经纪公司编号
    TUstpFtdcBrokerIDType   BrokerID;
    ///合约代码
    TUstpFtdcInstrumentIDType   InstrumentID;
};

```

### 6.3.43 ReqQryClientMarginCombType 方法

查询客户的保证金收取类型信息，指定当前客户的保证金收取类型，包含双边，大边（也称单向大边或单边），手动策略组合，手动策略大边。

#### 函数原形：

```

int ReqQryClientMarginCombType
    ( CUstpFtdcQryClientMarginCombTypeField
    *pQryClientMarginCombType, int nRequestID)

```

#### 参数：

```

///查询组合保证金类型
struct CUstpFtdcQryClientMarginCombTypeField
{

```

```

///交易所代码
TUstpFtdcExchangeIDType ExchangeID;
///经纪公司编号
TUstpFtdcBrokerIDType BrokerID;
///投资者编号
TUstpFtdcInvestorIDType InvestorID;
///投机套保标志
TUstpFtdcHedgeFlagType HedgeFlag;
///合约组代码
TUstpFtdcInstrumentGroupIDType InstrumentGroupID;
};

```

#### 6.3.44 ReqQryExchangeRate 方法（未启用）

交叉汇率查询。

注意:外汇期货合约专用接口，目前尚未启用。函数原形:

```

int ReqQryExchangeRate(
    CUstpFtdcQryExchangeRateField *pQryExchangeRate,
    int nRequestID)

```

**参数:**

pQryExchangeRate 指向交叉汇率查询结构的地址。

交叉汇率查询结构

```

struct CUstpFtdcQryExchangeRateField
{
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
    ///品种代码
    TUstpFtdcProductIDType ProductID;
};

```

#### 6.3.45 ReqQrySystemTime 方法

注意:查询飞马系统时间信息。

**函数原形:**

```

int ReqQrySystemTime (
    CUstpFtdcReqQrySystemTimeField *pReqQrySystemTime,

```

```
int nRequestID)
```

**参数:** pReqQrySystemTime 指向系统时间查询结构的地址。  
系统时间查询结构

```
struct CUstpFtdcReqQrySystemTimeField
{
    ///交易所代码
    TUstpFtdcExchangeIDType ExchangeID;
};
```

## 7. 开发示例

本节将展示常见的交易及查询类API的使用方法。

### 7.1 初始化示例

```
int main(int argc, char* argv[])
{
    //init
    memset(g_frontaddr, 0, BUFLen);
    strcpy(g_frontaddr, "tcp://192.168.130.62:8000");
    //strcpy(g_frontaddr, argv[1]);

    memset(g_frontqryaddr, 0, BUFLen);
    strcpy(g_frontqryaddr, "tcp://192.168.130.62:8002");
    //strcpy(g_frontaddr, argv[1]);

    printf("输入信息:\n");
    printf("g_frontaddr=[%s]\n ", g_frontaddr);
    printf("g_frontqryaddr =[%s]\n ", g_frontqryaddr);

    CUstpFtdcTraderApi *pTrader = CUstpFtdcTraderApi::CreateFtdcTraderApi("", false);

    int nMajorVersion, nMinorVersion;
    printf("版本信息:%s\n", pTrader->GetVersion(nMajorVersion, nMinorVersion));

    CTraderSpi spi(pTrader);
    pTrader->RegisterFront(g_frontaddr); //注册交易前置
    pTrader->RegisterQryFront(g_qryfrontaddr); //注册查询前置
    pTrader->RegisterSpi(&spi); //注册回调
    pTrader->SubscribePublicTopic(USTP_TERT_RESTART); //订阅公有流
    pTrader->SubscribePrivateTopic(USTP_TERT_RESTART); //订阅私有流
    //pTrader->SubscribeQuoteTopic(USTP_TERT_RESTART); //订阅询价流

    pTrader->Init(); //初始化API

    pTrader->Join(); //等待线程
    pTrader->Release(); //注销API
```

```

    return 0;
}

```

## 7.2 认证与登录

```

//认证信息
const char *pAPPID          = "client_test_v1.0";
const char *pPassword       = "9876543210";

//登录信息
const char *pLoginUserID    = "33333";
const char *pLoginPassword  = "7890abcd";
const char *pBrokerID       = "0001";
char *pInvestorID;

class CTraderSpi: public CUstpFtdcTraderSpi
{
public:
    CTraderSpi (CUstpFtdcTraderApi *pUserApi)
    {
        m_pUserApi      = pUserApi;
        m_nRequestID    = 0;
        m_bIsUserLogIn  = false;
    }

    ///错误应答
    virtual void OnRspError(CUstpFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
    {
        printf(" %s, nRequestID[%d], errorID[%d], errmsg[%s]\n", __FUNCTION__, nRequestID, pRspInfo->ErrorID, pRspInfo->ErrorMsg);
        fflush(stdout);
    }

    ///当客户端与交易后台建立起通信连接时（还未登录前），该方法被调用。
    virtual void OnFrontConnected()
    {
        //发送认证信息
        CUSTPFtdcDSUserInfoField mDSUserCertInfoField;
        strcpy(mDSUserCertInfoField.AppID, pAPPID);
        strcpy(mDSUserCertInfoField.AuthCode, pPassword);
        mDSUserCertInfoField.EncryptType = '1'; // 加密类型填 '1'

        int retVal = m_pUserApi->ReqDSUserCertification(&mDSUserCertInfoField, ++m_nRequestID);

        printf(" send ReqDSUserCertification...retVal[%d].\n", retVal);
    }

    ///当客户端与交易后台通信连接断开时，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。
    ///@param nReason 错误原因
    ///      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

```

```

    virtual void OnFrontDisconnected(int nReason)
    {
        printf(" %s..... nReason[0x%04x]\n", __FUNCTION__, nReason);
    }

    ///将监控中心的公钥信息进行处理后作为应答
    virtual void OnRspDSUserCertification(CUstpFtdcDSUserCertRspDataField *pDSUserCertRspField,
CUstpFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
    {
        if (pRspInfo != NULL && pRspInfo->ErrorID != 0) {
            printf(" %s.....rsp failed. errorid[%d], errmsg[%s]\n", __FUNCTION__, pRspInfo->ErrorID,
pRspInfo->ErrorMsg);
            return;
        }

        printf(" %s.....success \n", __FUNCTION__);

        //认证成功，发送登录请求
        SendReqLogin();
    }

    ///用户登录应答
    virtual void OnRspUserLogin(CUstpFtdcRspUserLoginField *pRspUserLogin, CUstpFtdcRspInfoField
*pRspInfo, int nRequestID, bool bIsLast)
    {
        if (pRspInfo!=NULL&&pRspInfo->ErrorID!=0)
        {
            printf("登录失败...原因[%s]\n", pRspInfo->ErrorMsg);
            return;
        }

        g_nOrdLocalID=atoi(pRspUserLogin->MaxOrderLocalID)+1;
        REPORT_EVENT(LOG_INFO, "LoginSuccess", "MaxLocalID = %d", g_nOrdLocalID);

        printf("登录成功，最大本地报单号:%d\n", g_nOrdLocalID);

        StartOrder(); //连接成功后就开始起一个线程开始操作
    }

    ///用户登录应答
    virtual void OnRspQryUserLogin(CUstpFtdcRspUserLoginField *pRspUserLogin, CUstpFtdcRspInfoField
*pRspInfo, int nRequestID, bool bIsLast)
    {
        if (pRspInfo->ErrorID != 0) {
            printf(" %s.....rsp failed. errorid[%d], errmsg[%s]\n", __FUNCTION__, pRspInfo->ErrorID,
pRspInfo->ErrorMsg);
            return;
        }

        m_bIsUserLogIn = true;

        printf(" %s.....success \n", __FUNCTION__);

        StartQuery();
    }
}

```



Private:

```
void SendReqLogin()
{
    // 发送登录请求
    CUstpFtdcReqUserLoginField mUserLoginField;
    strcpy(mUserLoginField.UserID, pLoginUserID);
    strcpy(mUserLoginField.Password, pLoginPassword);
    strcpy(mUserLoginField.BrokerID, pBrokerID);

    int retVal = m_pUserApi->ReqUserLogin(&mUserLoginField, ++m_nRequestID);

    printf(" send ReqUserLogin....retVal[%d]\n", retVal);
}

bool StartOrder()
{
    //int dwIDThread;
    unsigned long dwIDThread;
    THREAD_HANDLE hThread; /**< 线程句柄 */
    bool ret = true;
#ifdef WIN32
    hThread = ::CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE) OrderFunc, NULL, 0, &dwIDThread);
    if (hThread==NULL)
    {
        ret = false;
    }
    SetThreadPriority(hThread, THREAD_PRIORITY_TIME_CRITICAL);
    ResumeThread(hThread);
#else
    ret = (::pthread_create(&hThread, NULL, &OrderFunc, NULL) == 0);
#endif
    return ret;
}
```

### 7.3 查询请求

查询类请求必选在查询登录响应OnRspQueryUserLogin收到之后才可以发起。

```
//查询合约，支持通配，可按交易所查询、按品种查询、按合约查询
QueryInstrument( )
{
    CUstpFtdcQryInstrumentField CQryInstrument;
    memset(&CQryInstrument, 0, sizeof(CUstpFtdcQryInstrumentField));

    strcpy(CQryInstrument.ExchangeID, "CFFEX");

    if (pUserAPI==NULL)
    {
        REPORT_EVENT(LOG_CRITICAL, "OrderFunc", " USERAPI未创建");
        return ;
    }

    pUserAPI->ReqQryInstrument(&CQryInstrument, g_nOrdLocalID++);
```

```

    return ;
}

//查询投资者信息编号
QueryUserInvestor ( )
{
    CUstpFtdcQryUserInvestorField CQryUserInvestor;
    memset(&CQryUserInvestor, 0, sizeof(CUstpFtdcQryUserInvestorField));

    if (pUserAPI==NULL)
    {
        REPORT_EVENT(LOG_CRITICAL, "OrderFunc", " USERAPI未创建");
        return ;
    }

    strcpy(CQryUserInvestor.BrokerID, pBrokerID);
    strcpy(CQryUserInvestor.UserID, pLoginUserID);

    g_puserapi->ReqQryUserInvestor(&CQryUserInvestor, g_nOrdLocalID++);
    return ;
}

void CTraderSpi::OnRspQryUserInvestor(CUstpFtdcRspUserInvestorField * pRspUserInvestor,
CUstpFtdcRspInfoField * pRspInfo, int nRequestID, bool bIsLast)
{
    if (pRspInfo!=NULL&& pRspInfo->ErrorID!=0)
    {
        printf("查询投资者失败... 原因[%s]\n", pRspInfo->ErrorMsg);
        return;
    }

    pInvestorID = pRspUserInvestor ->InvestorID;
    printf("查询投资者成功, 投资者编号为:%s\n", pInvestorID);
}

//查询资金
QueryInvestorAccount ()
{
    CUstpFtdcQryInvestorAccountField CQryInvestorAccount;
    memset(&CQryInvestorAccount, 0, sizeof(CUstpFtdcQryInvestorAccountField));

    if (pUserAPI==NULL)
    {
        REPORT_EVENT(LOG_CRITICAL, "OrderFunc", " USERAPI未创建");
        return ;
    }

    strcpy(CQryInvestorAccount.BrokerID, pBrokerID);
    strcpy(CQryInvestorAccount.UserID, pLoginUserID);
    strcpy(CQryInvestorAccount.InvestorID, pInvestorID);

    pUserAPI->ReqQryInvestorAccount(&CQryInvestorAccount, g_nOrdLocalID++);
    return ;
}

```

//查询持仓, 支持通配, 可按投资者、按交易所、按交易编码、按合约查询

```
QueryInvestorPosition()
{
    CUstpFtdcQryInvestorPositionField CQryInvestorPosition;
    memset(&CQryInvestorPosition, 0, sizeof(CUstpFtdcQryInvestorPositionField));

    if (pUserAPI==NULL)
    {
        REPORT_EVENT(LOG_CRITICAL, "OrderFunc", " USERAPI未创建");
        return NULL;
    }

    strcpy(CQryInvestorPosition.BrokerID, pBrokerID);
    strcpy(CQryInvestorPosition.UserID, pLoginUserID);
    //strcpy(CQryInvestorPosition.InvestorID, pInvestorID);
    //strncpy(CQryInvestorPosition.ExchangeID, "CFFEX", 5);
    //strncpy(CQryInvestorPosition.InstrumentID, "IF2020", 6);

    pUserAPI->ReqQryInvestorPosition(&CQryInvestorPosition, g_nOrdLocalID++);
    return 0;
}
```

//查询成交, 支持通配, 可按投资者、按交易所、按合约、按交易编码、按成交编号查询

```
void StartQryTrade()
{
    CUstpFtdcQryTradeField CQryTrade;
    memset(&CQryTrade, 0, sizeof(CUstpFtdcQryTradeField));

    if (g_puserapi==NULL)
    {
        printf("StartQryTrade  USERAPI未创建");
        return ;
    }

    strcpy(CQryTrade.BrokerID, pBrokerID);
    strcpy(CQryTrade.UserID, pLoginUserID);

    //strcpy(CQryTrade.InvestorID, pInvestorID);
    //strncpy(CQryTrade.ExchangeID, "CFFEX", 5);
    //strncpy(CQryTrade.ClientID, "00000002", 8);
    //strncpy(CQryTrade.InstrumentID, "IF2020", 6);
    //strcpy(CQryTrade.TradeID, "1234");

    g_puserapi->ReqQryTrade(&CQryTrade, g_nOrdLocalID++);
    return ;
}
```

//查询委托, 支持通配, 可按投资者、按交易所、按合约、按交易编码、按状态、按类型查询

```
void StartQryOrder()
{
    CUstpFtdcQryOrderField CQryOrder;
    memset(&CQryOrder, 0, sizeof(CUstpFtdcQryOrderField));

    if (g_puserapi==NULL)
    {
```

```

        printf("StartQryOrder  USERAPI未创建");
        return ;
    }

    strcpy(CQryOrder.BrokerID, pBrokerID);
    strcpy(CQryOrder.UserID, pLoginUserID);
    strcpy(CQryOrder. OrderStatus, " ");//[0:全部成交 1:部分成交 3:未成交 5:撤单] NULL:错单 空格:全部
    strcpy(CQryOrder. OrderType, "0");//[0:普通委托 1:执行 2:放弃执行 4:期转现] NULL:全部

    //strcpy(CQryOrder. InvestorID, pInvestorID);
    //strncpy(CQryOrder. ExchangeID, "CFFEX", 5);
    //strncpy(CQryOrder. ClientID, "00000002", 8);
    //strncpy(CQryOrder. InstrumentID, "IF2020", 6);
    // strcpy(CQryOrder. OrderSysID, "    1123");

    g_puserapi->ReqQryOrder (&QryOrder, g_nOrdLocalID++);
    return ;
}

```

#### 7.4 交易请求及回报

```

void * StartInputOrder (void *pParam)
{
    if (m_pUserApi ==NULL)
    {
        printf("StartInputOrder  USERAPI未创建");
        return ;
    }

    CUstpFtdcInputOrderField ord;
    memset(&ord, 0, sizeof(CUstpFtdcInputOrderField));
    printf("请输入报单信息\n");

    GetString("ExchangeID", ord.ExchangeID);
    GetString("InstrumentID", ord.InstrumentID);
    GetDouble("LimitPrice", ord.LimitPrice);
    GetInt("Volume", ord.Volume);
    GetCharDefault("OrderPriceType[1:市价 2:限价](2)", ord.OrderPriceType, '2');
    GetCharDefault("Direction[0:买 1:卖](0)", ord.Direction, '0');
    GetCharDefault("OffsetFlag[0:开仓 1:平仓](0)", ord.OffsetFlag, '0');
    GetCharDefault("HedgeFlag[1:投机 2:套利 3:套保](1)", ord.HedgeFlag, '1');
    GetCharDefault("TimeCondition[1:IOC 2:本节有效 3:当日有效](3)", ord.TimeCondition, '3');
    GetCharDefault("ArbiType[0:基本 1:跨期 2:跨品种 8:互换](0)", ord.ArbiType, '0');
    GetCharDefault("VolumeCondition[1:任何数量 2:最小数量 3:全部数量](1)", ord.VolumeCondition, '1');

    strcpy(ord.BrokerID, pBrokerID);
    strcpy(ord.UserID, pLoginUserID);
    strcpy(ord.InvestorID, pInvestorID);
    ord.ForceCloseReason='0';
    sprintf(ord.UserOrderLocalID, "%012d", g_nOrdLocalID++);

    g_puserapi->ReqOrderInsert (&ord, g_nOrdLocalID++);

    return ;
}

```

```

void StartOrderAction()
{
    /*支持系统报单号和本地报单号两种撤单方式*/
    //-----系统报单号撤单-----//
    int SysID;
    CUstpFtdcOrderActionField OrderAction;
    memset(&OrderAction, 0, sizeof(CUstpFtdcOrderActionField));
    if (g_puserapi==NULL)
    {
        printf("StartOrderAction  USERAPI未创建");
        return ;
    }

    strcpy(OrderAction.BrokerID, pBrokerID);
    strcpy(OrderAction.UserID, pLoginUserID);
    strcpy(OrderAction.InvestorID, pInvestorID);

    GetString("ExchangeID", OrderAction.ExchangeID);
    GetString("被撤单的OrderSysID", OrderAction.OrderSysID);
    GetString("被撤单的用户OrderLocalID", OrderAction.UserOrderLocalID);

    OrderAction.ActionFlag=USTP_FTDC_AF_Delete;
    sprintf(OrderAction.UserOrderActionLocalID, "%012d", g_nOrdLocalID++);

    g_puserapi->ReqOrderAction(&OrderAction, g_nOrdLocalID++);

    return ;
    //-----系统报单号撤单-----//

    //-----本地报单号撤单-----//
    /*
        int LocalID;
        CUstpFtdcOrderActionField OrderAction;
        memset(&OrderAction, 0, sizeof(CUstpFtdcOrderActionField));
        if (g_puserapi==NULL)
        {
            printf("StartOrderAction  USERAPI未创建");
            return ;
        }
        strcpy(OrderAction.ExchangeID, "CFFEX");
        strcpy(OrderAction.BrokerID, pBrokerID);
        strcpy(OrderAction.UserID, pLoginUserID);

        printf("input InvestorID\n");
        scanf("%s", (OrderAction.InvestorID));
        printf("InvestorID=[%s]\n", OrderAction.InvestorID);

        printf("请输入本地报单号:");
        scanf("%d", &LocalID);
        sprintf(OrderAction.UserOrderLocalID, "%012d", LocalID);

        printf("撤销本地报单号[%s]\n", OrderAction.UserOrderLocalID );
        strcpy(OrderAction.OrderSysID, "");
    */
}

```

```

        OrderAction.ActionFlag=USTP_FTDC_AF_Delete;
        sprintf (OrderAction.UserOrderActionLocalID, "%012d", g_nOrdLocalID++);

        g_puserapi->ReqOrderAction (&OrderAction, g_nOrdLocalID++);

        return ;
    */
    //-----本地报单号撤单-----//
}

//报单响应
void CTraderSpi::OnRspOrderInsert (CUstpFtdcInputOrderField *pInputOrder, CUstpFtdcRspInfoField
*pRspInfo, int nRequestID, bool bIsLast)
{
    if (pRspInfo!=NULL&& pRspInfo->ErrorID!=0)
    {
        printf("-----\n");
        printf("报单失败 错误原因: %s\n", pRspInfo->ErrorMsg);
        printf("-----\n");
        return;
    }
    if (pInputOrder==NULL)
    {
        printf("没有报单数据\n");
        return;
    }
    printf("-----\n");
    printf("报单成功\n");
    printf("经纪公司编号=[%s]\n", pInputOrder->BrokerID);
    printf("交易所代码=[%s]\n", pInputOrder->ExchangeID);
    printf("系统报单编号=[%s]\n", pInputOrder->OrderSysID);
    printf("用户代码=[%s]\n", pInputOrder->UserID);
    printf("合约代码=[%s]\n", pInputOrder->InstrumentID);
    printf("用户本地报单号=[%s]\n", pInputOrder->UserOrderLocalID);
    printf("投机套保标志=[%c]\n", pInputOrder->HedgeFlag);
    printf("数量=[%d]\n", pInputOrder->Volume);
    printf("业务单元=[%s]\n", pInputOrder->BusinessUnit);
    printf("用户自定义域=[%s]\n", pInputOrder->UserCustom);
    printf("客户号=[%s]\n", pInputOrder->ClientID);
    printf("-----\n");
    return ;
}

void CTraderSpi::OnRspOrderAction (CUstpFtdcOrderActionField *pOrderAction, CUstpFtdcRspInfoField
*pRspInfo, int nRequestID, bool bIsLast)
{
    if (pRspInfo!=NULL&& pRspInfo->ErrorID!=0)
    {
        printf("-----\n");
        printf("撤单失败 错误原因: %s\n", pRspInfo->ErrorMsg);
        printf("-----\n");
        return;
    }
    if (pOrderAction==NULL)
    {
        printf("没有撤单数据\n");
    }
}

```

```

        return;
    }
    printf("-----\n");
    printf("撤单成功\n");
    printf("-----\n");
    return ;
}

void CTraderSpi::OnRtnTrade(CUstpFtdcTradeField *pTrade)
{
    printf("-----\n");
    printf("收到成交回报\n");
    printf("经纪公司编号=[%s]\n", pTrade->BrokerID);
    printf("交易所代码=[%s]\n", pTrade->ExchangeID);
    printf("交易日=[%s]\n", pTrade->TradingDay);
    printf("会员编号=[%s]\n", pTrade->ParticipantID);
    printf("下单席位号=[%s]\n", pTrade->SeatID);
    printf("投资者编号=[%s]\n", pTrade->InvestorID);
    printf("客户号=[%s]\n", pTrade->ClientID);
    printf("用户编号=[%s]\n", pTrade->UserID);
    printf("下单用户编号=[%s]\n", pTrade->OrderUserID);
    printf("成交编号=[%s]\n", pTrade->TradeID);
    printf("用户本地报单号=[%s]\n", pTrade->UserOrderLocalID);
    printf("合约代码=[%s]\n", pTrade->InstrumentID);
    printf("买卖方向=[%c]\n", pTrade->Direction);
    printf("开平标志=[%c]\n", pTrade->OffsetFlag);
    printf("投机套保标志=[%c]\n", pTrade->HedgeFlag);
    printf("成交价格=[%lf]\n", pTrade->TradePrice);
    printf("成交数量=[%d]\n", pTrade->TradeVolume);
    printf("成交时间=[%s]\n", pTrade->TradeTime);
    printf("策略类别=[%c]\n", pTrade->ArbiType);
    printf("组合合约=[%s]\n", pTrade->ArbiInstrumentID);
    printf("清算会员编号=[%s]\n", pTrade->ClearingPartID);
    printf("本次成交手续费=[%19.4lf]\n", pTrade->UsedFee);
    printf("本次成交占用保证金=[%19.4lf]\n", pTrade->UsedMargin);
    printf("本次成交占用权利金=[%19.4lf]\n", pTrade->Premium);
    printf("持仓表今持仓量=[%d]\n", pTrade->Position);
    printf("持仓表今持仓成本=[%19.4lf]\n", pTrade->PositionCost);
    printf("资金表可用资金=[%19.4lf]\n", pTrade->Available);
    printf("资金表占用保证金=[%19.4lf]\n", pTrade->Margin);
    printf("资金表冻结的保证金=[%19.4lf]\n", pTrade->FrozenMargin);
    printf("-----\n");
    return;
}

void CTraderSpi::OnRtnOrder(CUstpFtdcOrderField *pOrder)
{
    printf("-----\n");
    printf("收到报单回报\n");
    printf("经纪公司编号=[%s]\n", pOrder->BrokerID);
    printf("交易所代码=[%s]\n", pOrder->ExchangeID);
    printf("系统报单编号=[%s]\n", pOrder->OrderSysID);
    printf("用户代码=[%s]\n", pOrder->UserID);
    printf("合约代码=[%s]\n", pOrder->InstrumentID);
    printf("用户本地报单号=[%s]\n", pOrder->UserOrderLocalID);
    printf("报单类型=[%c]\n", pOrder->OrderPriceType);
    printf("买卖方向=[%c]\n", pOrder->Direction);

```

```

printf("开平标志=[%c]\n", pOrder->OffsetFlag);
printf("投机套保标志=[%c]\n", pOrder->HedgeFlag);
printf("价格=[%lf]\n", pOrder->LimitPrice);
printf("数量=[%d]\n", pOrder->Volume);
printf("有效期类型=[%c]\n", pOrder->TimeCondition);
printf("GTD日期=[%s]\n", pOrder->GTDDate);
printf("最小成交量=[%d]\n", pOrder->MinVolume);
printf("止损价=[%lf]\n", pOrder->StopPrice);
printf("强平原因=[%c]\n", pOrder->ForceCloseReason);
printf("自动挂起标志=[%d]\n", pOrder->IsAutoSuspend);
printf("业务单元=[%s]\n", pOrder->BusinessUnit);
printf("用户自定义域=[%s]\n", pOrder->UserCustom);
printf("交易日=[%s]\n", pOrder->TradingDay);
printf("会员编号=[%s]\n", pOrder->ParticipantID);
printf("最初下单用户编号=[%s]\n", pOrder->OrderUserID);
printf("客户号=[%s]\n", pOrder->ClientID);
printf("下单席位号=[%s]\n", pOrder->SeatID);
printf("报单时间=[%s]\n", pOrder->InsertTime);
printf("本地报单编号=[%s]\n", pOrder->OrderLocalID);
printf("报单来源=[%c]\n", pOrder->OrderSource);
printf("报单状态=[%c]\n", pOrder->OrderStatus);
printf("撤销时间=[%s]\n", pOrder->CancelTime);
printf("撤单用户编号=[%s]\n", pOrder->CancelUserID);
printf("今成交数量=[%d]\n", pOrder->VolumeTraded);
printf("剩余数量=[%d]\n", pOrder->VolumeRemain);
printf("业务单元=[%s]\n", pOrder->BusinessUnit);
printf("-----\n");
return ;
}

```

## 7.5 报价请求及回报

//报价

```

void StartQuoteInsert()
{
    if (g_puserapi==NULL)
    {
        printf("StartInputOrder  USERAPI未创建");
        return ;
    }
    CUsdpFtdcInputQuoteField QuoteInsert;
    memset(&QuoteInsert, 0, sizeof(CUsdpFtdcInputQuoteField));

    strcpy(QuoteInsert.BrokerID, pBrokerID);
    strcpy(QuoteInsert.UserID, pLoginUserID);
    strcpy(QuoteInsert.InvestorID, pInvestorID);

    sprintf(QuoteInsert.UserQuoteLocalID, "%012d", g_nOrdLocalID++);
    sprintf(QuoteInsert.BidUserOrderLocalID, "%012d", g_nOrdLocalID++);
    sprintf(QuoteInsert.AskUserOrderLocalID, "%012d", g_nOrdLocalID++);

    GetString("ExchangeID", QuoteInsert.ExchangeID);
    GetString("InstrumentID", QuoteInsert.InstrumentID);
    GetCharDefault("BidOffsetFlag[0:开仓 1:平仓] (0)", QuoteInsert.BidOffsetFlag, '0');
    GetCharDefault("BidHedgeFlag[1:投机 2:套利 3:套保] (1)", QuoteInsert.BidHedgeFlag, '1');
    GetDouble("BidPrice", QuoteInsert.BidPrice);
}

```



```

    GetInt("BidVolume", QuoteInsert.BidVolume);

    GetCharDefault("AskOffsetFlag[0:开仓 1:平仓] (0)", QuoteInsert.AskOffsetFlag, '0');
    GetCharDefault("AskHedgeFlag[1:投机 2:套利 3:套保] (1)", QuoteInsert.AskHedgeFlag, '1');
    GetDouble("AskPrice", QuoteInsert.AskPrice);
    GetInt("AskVolume", QuoteInsert.AskVolume);

    g_puserapi->ReqQuoteInsert(&QuoteInsert, g_nOrdLocalID++);

    return;
}
//撤单
void StartQuoteAction()
{
    int SysID;
    if (g_puserapi==NULL)
    {
        printf("StartInputOrder  USERAPI未创建");
        return ;
    }
    CUstpFtdcQuoteActionField QuoteAction;
    memset(&QuoteAction, 0, sizeof(CUstpFtdcQuoteActionField));

    strcpy(QuoteAction.BrokerID, pBrokerID);
    strcpy(QuoteAction.UserID, pLoginUserID);
    strcpy(QuoteAction.InvestorID, pInvestorID);

    GetString("ExchangeID", QuoteAction.ExchangeID);
    GetString("被撤单的QuoteSysID", QuoteAction.QuoteSysID);
    GetString("被撤单的用户QuoteLocalID", QuoteAction.UserQuoteLocalID);

    QuoteAction.ActionFlag=USTP_FTDC_AF_Delete;
    sprintf(QuoteAction.UserQuoteActionLocalID, "%012d", g_nOrdLocalID++);

    g_puserapi->ReqQuoteAction(&QuoteAction, g_nOrdLocalID++);

    return ;
}

//报价响应
void CTraderSpi::OnRspQuoteInsert(CUstpFtdcInputQuoteField* pInputQuote, CUstpFtdcRspInfoField* pRspInfo,
int nRequestID, bool bIsLast)
{
    if (pRspInfo!=NULL&&pRspInfo->ErrorID!=0)
    {
        printf("-----\n");
        printf("报价录入失败 错误原因: %s\n", pRspInfo->ErrorMsg);
        printf("-----\n");
        return;
    }
    if(pInputQuote==NULL)
    {
        printf("没有报价录入数据\n");
        return;
    }
    printf("-----\n");
    printf("报价录入成功\n");
    printf("QuoteSysID=[%s]\n", pInputQuote->QuoteSysID);
}

```

```

    printf("用户=[%s]\n", pInputQuote->UserID);
    printf("报价用户=[%s]\n", pInputQuote->QuoteUserID);
    printf("-----\n");
    return ;
}

//撤销报价响应
void CTraderSpi::OnRspQuoteAction(CUstpFtdcQuoteActionField* pQuoteAction, CUstpFtdcRspInfoField*
pRspInfo, int nRequestID, bool bIsLast)
{
    if (pRspInfo!=NULL&& pRspInfo->ErrorID!=0)
    {
        printf("-----\n");
        printf("撤销报价失败 错误原因: [%d] [%s]\n", pRspInfo->ErrorID, pRspInfo->ErrorMsg);
        printf("-----\n");
    }
    else
    {
        printf("-----\n");
        printf("撤销报价成功\n");
        printf("-----\n");
    }

    if(pQuoteAction==NULL)
    {
        printf("没有撤销报价数据\n");
        return;
    }
    printf("-----\n");
    printf("UserQuoteActionLocalID=%s\n", pQuoteAction->UserQuoteActionLocalID);
    printf("UserQuoteLocalID=%s\n", pQuoteAction->UserQuoteLocalID);
    printf("QuoteSysID=%s\n", pQuoteAction->QuoteSysID);
    printf("-----\n");
    return ;
}

```

//报价回报

```

void CTraderSpi::OnRtnQuote(CUstpFtdcRtnQuoteField* pRtnQuote)
{
    printf("-----\n");
    printf("收到报价回报\n");
    printf("经纪公司编号=[%s]\n", pQuote->BrokerID);
    printf("交易所代码=[%s]\n", pQuote->ExchangeID);
    printf("投资者编号=[%s]\n", pQuote->InvestorID);
    printf("用户代码=[%s]\n", pQuote->UserID);
    printf("合约代码=[%s]\n", pQuote->InstrumentID);
    printf("系统报价编号=[%s]\n", pQuote->QuoteSysID);
    printf("用户本地报价编号=[%s]\n", pQuote->UserQuoteLocalID);
    printf("飞马向交易系统报的本地报价编号=[%s]\n", pQuote->QuoteLocalID);

    printf("买方买入数量=[%d]\n", pQuote->BidVolume);
    printf("买方开平标志=[%c]\n", pQuote->BidOffsetFlag);
    printf("买方投机套保标志=[%c]\n", pQuote->BidHedgeFlag);
    printf("买方买入价格=[%lf]\n", pQuote->BidPrice);
    printf("卖方卖出数量=[%d]\n", pQuote->AskVolume);
    printf("卖方开平标志=[%c]\n", pQuote->AskOffsetFlag);
    printf("卖方投机套保标志=[%c]\n", pQuote->AskHedgeFlag);
    printf("卖方卖出价格=[%lf]\n", pQuote->AskPrice);
}

```

```

printf("报价用户=[%s]\n", pQuote->QuoteUserID);

printf("业务单元=[%s]\n", pQuote->BusinessUnit);
printf("用户自定义域=[%s]\n", pQuote->UserCustom);

printf("拆分出来的买方用户本地报单编号=[%s]\n", pQuote->BidUserOrderLocalID);
printf("拆分出来的卖方用户本地报单编号=[%s]\n", pQuote->AskUserOrderLocalID);
printf("买方系统报单编号=[%s]\n", pQuote->BidOrderSysID);
printf("卖方系统报单编号=[%s]\n", pQuote->AskOrderSysID);

printf("报价单状态=[%c]\n", pQuote->QuoteStatus);
printf("-----\n");
return ;
}

```

## 7.6 询价请求及回报

//询价

```

void StartForQuote()
{
    if (g_puserapi==NULL)
    {
        printf("StartInputOrder  USERAPI未创建");
        return ;
    }
    CUSTPFTDCReqForQuoteField ReqForQuote;
    memset(&ReqForQuote, 0, sizeof(CUSTPFTDCReqForQuoteField));
    strcpy(ReqForQuote.BrokerID, pBrokerID);
    strcpy(ReqForQuote.UserID, );
    strcpy(ReqForQuote.InvestorID, pInvestorID);

    GetString("ExchangeID", ReqForQuote.ExchangeID);
    GetString("InstrumentID", ReqForQuote.InstrumentID);

    sprintf(ReqForQuote.ReqForQuoteID, "%012d", g_nOrdLocalID++);

    g_puserapi->ReqForQuote(&ReqForQuote, g_nOrdLocalID++);

    return ;
}

```

//询价响应

```

void CTraderSpi::OnRspForQuote(CUSTPFTDCReqForQuoteField* pReqForQuote, CUSTPFTDCRspInfoField* pRspInfo,
int nRequestID, bool bIsLast)
{
    if (pRspInfo!=NULL&&pRspInfo->ErrorID!=0)
    {
        printf("-----\n");
        printf("询价失败 错误原因: %s\n", pRspInfo->ErrorMsg);
        printf("-----\n");
        return;
    }
    if (pReqForQuote==NULL)
    {

```

```
        printf("没有询价数据\n");
        return;
    }
    printf("-----\n");
    printf("询价成功\n");
    printf("-----\n");
    return ;
}

//询价流
void CTraderSpi::OnRtnForQuote(CUstpFtdcReqForQuoteField *pReqForQuote)
{
    printf("-----\n");
    printf("收到询价回报\n");
    printf("经纪公司编号=[%s]\n", pReqForQuote->BrokerID);
    printf("交易所代码=[%s]\n", pReqForQuote->ExchangeID);
    printf("询价编号=[%s]\n", pReqForQuote->ReqForQuoteID);
    printf("用户代码=[%s]\n", pReqForQuote->UserID);
    printf("合约代码=[%s]\n", pReqForQuote->InstrumentID);
    printf("交易日=[%s]\n", pReqForQuote->TradingDay);
    printf("客户号=[%s]\n", pReqForQuote->ClientID);
    printf("报单时间=[%s]\n", pReqForQuote->ReqForQuoteTime);
    printf("-----\n");
    return ;
}
```