备注：版本和修改历史暂未增加

X-Speed 交易系统

<应用程序接口说明>

<v1.0.15.4>

<2016 年 07 月>

文档信息

项目名称	X-Speed 交易系统项目
文档名称	<X-Speed 交易系统应用程序接口说明文档>
版本号	<v1.0.15.4>
作者	X-Speed 交易系统项目组

文档修订历史(依据交易 API 的版本)

版本	日期	修订内容	修订者	备注
1.0.10.0	2012-09-07	版本发布	X-Speed API 开发组	
1.0.10.1	2012-10-16	提供对期权的基本下单和查询功能的支持。	X-Speed API 开发组	下单、撤单和委托回报，成交回报以及查询中都加入了合约类型这个字段，0 代表期货，1 代表期权。在资金信息中加入了期权的本日权利金收入、本日权利金付出、昨权利金收付和期权市值。合约信息数据结构加入了委托上限、合约乘数和最小变动价位。
1.0.10.2	2012-11-16	增强 API 的稳定性	X-Speed API 开发组	修改 API 端在频繁连接时导致端口重用无法连接的问题
1.0.11.0	2012-12-28	在 linux 版本 API 的委托和成交回报中添加报单信息	X-Speed API 开发组	委托回报中委托价格委托数量目前返回的为空，这个随后版本会加上
1.0.11.6	2013-01-25	增加对套利组合合约的支持，支持接口参数 DFITCErrorRtnField 的使用。替代前几个版本中 OnRtnErrorMsg，降低集中处理产生的混乱。行情前置增加行情缓冲的功能。	X-Speed API 开发组	增加对套利组合合约的支持。增加 API 一个实例对应一个 SOCKET 的连接机制，以对接口参数 DFITCErrorRtnField 替代 OnRtnErrorMsg 的集中处理。修改部分 bug。行情前置增加行情缓冲的功能。
1.0.11.9	2013-02-24	1.在下单结构中替换自动单字段为 DFITCInsertType 自动单类别。 2.在委托回报和委托查询结构中，将原	X-Speed API 开发组	

		来委托单类别字段替换为DFITCInsertType 。 3. 在委托回报、委托查询、成交回报和成交查询结构中增加两个预留字段。 4.成交查询中，将原来的委托单类别替换为DFITCMatchType。 5.支持开市自动单功能。 6.API添加流量控制机制。		
1.0.11.12	2013-03-08	添加ReqQrySpecifyInstrument()和ReqQryPositionDetail()接口。	X-Speed API 开发组	
1.0.12.15	2013-08-05	添加ReqResetPassword() ReqQryTradeCode() ReqBillConfirm() ReqConfirmProductInfo() ReqEquityComputMode() 和ReqQryBill()接口。	X-Speed API 开发组	下单、撤单的请求和响应函数中增加了请求 ID 字段，并且在下单响应中增加手续费字段。
1.0.13.69	2013-10-30	添加ReqQuoteInsertOrder() ReqQuoteCancelOrder() ReqQryExchangeStatus() ReqForQuote() 接口	X-Speed API 开发组	增加做市商模块 增加交易所状态查询接口
1.0.14.56	2015-01-28	委托查询和成交查询中，增加查询条件。	X-Speed API 开发组	委托查询增加“委托状态”和“报单类型”以及委托号查询条件。成交查

		ReqInsertOrder() OnRtnOrder() OnRspQryOrderInfo()		询增加“报单类型”和按指定委托号查询条件。 委托及回报和委托查询增加止损止盈价，订单属性字段，委托增加上期FAK单最小成交量字段
1.0.14.66	2015-04-22		X-Speed API 开发组	将询价通知订阅和退订移到行情接口中
1.0.14.70	2015-04-27	修改API使用异步socket；去除API结构体的默认构造函数	X-Speed API 开发组	
1.0.14.81	2015-05-06	做市相关接口中增加自定义类别字段。 成交查询增加报单编号字段。 添加 ReqQryQuoteOrderInfo() ReqQryQuoteNotice() ReqCancelAllOrder() ReqQryDepthMarketData() ReqQryForQuote()	X-Speed API 开发组	
1.0.14.83	2015-06-30	持仓查询响应接口，增加平昨挂单量字段，兼容v13版前置	X-Speed API 开发组	
1.0.14.85	2015-07-01	合约查询接口中增加：交易所期权最低保障风险系数、期权合约最小保证金、期权开仓单位 单合约查询增加：多头保证金费(定额)、空头保证金费(定额)	X-Speed API 开发组	主要是合约查询接口增加期权保证金计算相关字段
1.0.15.4	2016.07.22	1.新增API版本号查询接口 2.新增条件单相关接口	XSpeed 柜台开发组	

目录

1. 简介

XSpeed 交易系统应用程序接口（API）是一个基于 C++的类库，通过使用和扩展类库提供的功能来实现相关的交易功能，包括资金账户登陆，获取行情，报单、撤单、资金查询、持仓查询、委托查询，成交查询、保证金率/手续费率以及合约查询等功能。

本文档的主要内容包括：

- 接口类库文件说明
- 接口的系统架构
- 接口开发规范
- 业务与接口的对照表
- 接口定义

2. 接口类库文件说明

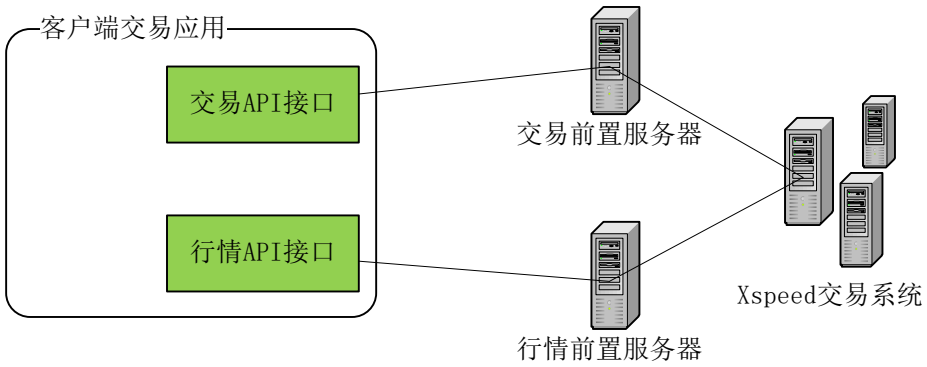
接口类库包含如下文件：

文件名	版本	文件描述
DFITCApiDataType.h	v1.14	定义接口所需的数据类型的头文件
DFITCApiStruct.h	v1.14	定义接口所需的数据接口的头文件
DFITCTraderApi.h	v1.14	定义交易接口的头文件
DFITCMdApi.h	v1.14	定义行情接口的头文件
DFITCTraderApi.dll	v1.14	交易接口动态链接库二进制文件
DFITCTraderApi.lib	v1.14	交易接口导入库文件
DFITCMdApi.dll	v1.14	行情接口动态链接库二进制文件
DFITCMdApi.lib	v1.14	行情接口导入库文件
DFITCMdApi.so	v1.14	Linux 版本行情 API 的动态链接库文件
DFITCTraderApi.so	v1.14	Linux 版本交易 API 的动态链接库文件

支持 Microsoft VisualC++6.0，VS2005 等集成开发环境。

3. 系统架构

3.1 接口与其他系统关系



说明：

- 本 API 接口在客户端交易应用（简称客户端）与 XSpeed 交易系统的通信时使用；
- 客户端可以是人工交易客户端或者程序化交易系统；
- 交易 API 接口负责与交易前置连接；
- 行情 API 接口负责与行情前置连接；

3.2 通讯模式

API 与前置的通讯协议是建立在 TCP 协议上的通讯协议，一旦建立 TCP 连接，双方将保持该连接（长连接），通讯模式分为 3 种：

- 对话通讯模式：客户端主动发起请求，前置机接收请求，并立即将应答返回给客户端；
- 私有通讯模式：由前置机主动发起，通过 TCP 长连接向客户端发送特定信息，比如委托回报，成交回报等等；
- 广播通讯模式：由前置机主动发起，向所有连接客户端发送广播信息，比如行情信息等等。

3.3 接口模式

API 封装了两个接口，分别为 DFITCTraderApi 和 DFITCTraderSpi，两个接口对 API 与前置的通信及通信报文协议进行了封装，方便客户端应用程序的开发。客户端应用程序可以通过 DFITCTraderApi 的接口发出操作请求，通过继承 DFITCTraderSpi 并重载回调函数来处理后台服务的响应。

3.3.1 对话流和查询流编程接口通常如下：

```
请求：int  DFITCTraderApi::ReqXXX(DFITCxxxField * pReqXXX)
响应：void DFITCTraderSpi::OnRspXXX(DFITCxxxField * pRspXXX,
                                    DFITCErrorRtnField * pErrorInfo,
                                    bool bIsLast);
```

其中请求接口的参数内容不能为空，每次请求时，需要检查接口的返回值是否为 0，每个结构体里包含了一个 lRequestID 字段，当请求查询信息返回时，可以通过该字段将请求与响应对应起来。

当 API 收到后台服务应答时，DFITCTraderSpi 的回调函数将被调用，即会调用客户端继承并实现的 Spi 函数。如果响应数据不止一个，比如委托查询数据，则回调函数会被多次调用，最后一条数据时，bIsLast 将为 true。回调函数的第一个参数为响应的具体数据，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。当失败时该值不为 NULL，所以当 Spi 函数被调用时，首先应该检测该值是否为 NULL，并从中获取错误 ID 和错误信息。

最后一个参数为响应结束标志，表示是否是这次请求响应的最后一次回调。

3.3.2 私有流编程接口

私有流中的数据中有客户的私有信息，如委托回报、成交回报等
通常私有流接收回报的编程如下：

```
void DFITCTraderSpi::OnRtnXXX(DFITCxxxField * pRtnXXX);
void DFITCTraderSpi::OnRtnErrorMsg(DFITCErrorRtnField * pErrorInfo);
```

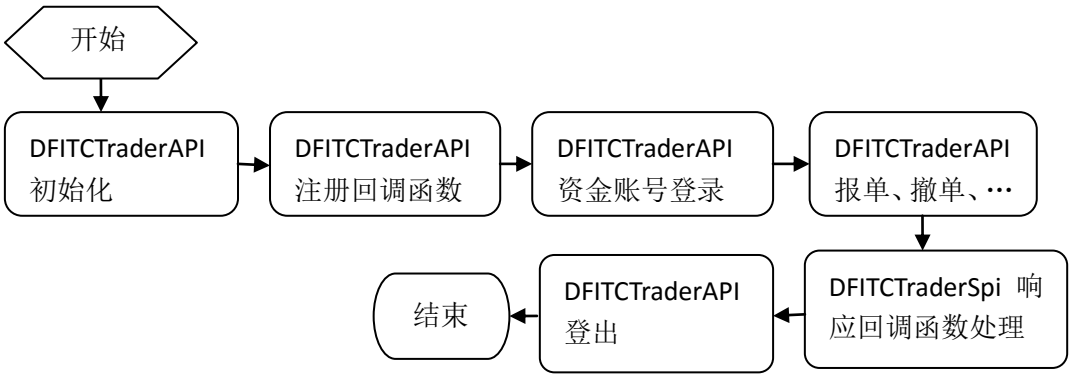
当收到交易所的确认时，该类接口将被调用，如委托回报，委托单为废单等，回调参数为具体内容。

4. 接口开发规范

4.1 命名空间

客户端交易 API 方法的命名空间为“DFITCXSPEEDAPI”，使用该接口时，请添加命名空间的引用：using namespace DFITCXSPEEDAPI;（行情 API 命名空间为行情 API 命名空间为：DFITCXSPEEDMDAPI）

4.2 开发流程



说明：

- 初始化时，可以指明多个前置机地址，接口将自动按顺序进行尝试连接，第一个连接成功后，将不再继续尝试其他连接；
- 如果客户端应用同时涉及多个资金账户，需要创建多个 DFITCTraderApi 实例；
- 报单与回调函数在不同线程中调用，报单线程是安全的线程，可以同时在多个线程中被调用，回调函数统一在一个线程中进行处理，如果在回调函数中有耗时过多的操作，将影响后续回报处理的速度，通常 Spi 应迅速返回，可放入一个缓冲区或队列，以勉影响下一次数据的接口
- **从 v1.0.14 版 API 开始，API 中的结构体不再提供默认构造函数，在接口请求时，需要先进行 memset 清空操作**

4.3 DFITCTraderSpi 接口

- DFITCTraderSpi 接口定义了事件通知接口，开发人员必须正确继承并实现 DFITCTraderSpi 接口，编写对应的事件处理方法。注意，在 Spi 函数接口中有 DFITCErrorRtnField 的参数时，若该参数不 NULL 表示有错误产生。其它参数在任何时候，均不会为 NULL。

4.4 lRequestID 字段

- 由于报文的发送和接收是异步处理的，接口定义了每次请求与响应报文的唯一标识：

lRequestID 字段，客户端应用通过该字段，将请求报文与响应报文对应起来，该 id 由客户应用产生并维护，且只有对话和查询时有用。

4.5 LocalOrderID 本地委托号字段

- 本地委托号唯一标识了一个会话中的每次报单，以及关联该报单的后续委托、成交、撤单等回报信息。在一个资金账号的一次登录会话范围内，本地委托号不能重复，且必须是大于等于 1 的数字（推荐从 2 开始），可选的处理方式：每次报单在上一本地委托号基础上加 1。注意，如果网络连接断开了再重连后，不能再使用本地委托号撤单，可用柜台委托号来撤单，但是委托、成交查询等，本地委托号都将原样返回的。

4.6 SpdOrderID 柜台委托号字段

- 柜台委托号在 xspeed 系统里唯一标识了一笔报单，该笔报单在每个交易日里不会重复，且一般总是从 1 开始递增的（0 为无效的柜台委托号）。该委托号在委托响应成功时，会返回。

4.7 连接断开与重连

- 当 TCP 连接断开时，将使用 OnFrontDisconnected（）回调方法通知客户端；此时客户端 API 也会自动检测与前置机之间的连接，当网络可用，将自动建立连接，并使用 OnFrontConnected（）方法通知客户端，客户端可以在该方法中完成登录请求任务。

4.8 查询频率控制

- X-Speed API 前置对 API 的查询频率有严格的控制，默认情况下，频率控制规则为 1)一秒内不允许查询 2 次，2)对于在途的查询，不允许再查。以上均会提示频率过快的错误信息或接口请求失败。频率控制由柜台前置的一个配置决定，也可根据实际需要，把配置选项关闭，不进行频率控制。推荐用户是一个查询结束后，再查询下一个数据。大量的查询会影响 xspeed 系统的报单撤单性能。

4.9 pErrorInfo 异常信息数据结构

- pErrorInfo:返回异常信息的地址。在 Spi 函数进入时，若该指针不为 NULL，表示有错误，此时应处理错误信息，对于正确的调用，该变量值为 NULL。一般情况下，程序可先判断该变量是否为 NULL，再根据情况进行后续处理。

```
struct DFITCErrorRtnField
{
    DFITCRequestIDType          requestID;          //请求 ID
    DFITCSessionIDType          sessionID;          //会话标识
    DFITCAccountIDType          accountID;          //资金账号
    DFITCErrorIDType            nErrorID;           //错误 ID
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCErrorMsgInfoType       errorMsg;           //错误信息
    DFITCInstrumentIDType       instrumentID;       //合约代码
 };
```

5. 业务与接口对照表

业务类型	业务	请求接口	响应接口	数据流
登录	登录	DFITCTraderApi::ReqUserLogin	DFITCTraderSpi::OnRspUserLogin	对话流
	登出	DFITCTraderApi::ReqUserLogout	DFITCTraderSpi::OnRspUserLogout	对话流
交易	下单	DFITCTraderApi:: ReqInsertOrder	DFITCTraderSpi::OnRspInsertOrder	对话流
	撤单	DFITCTraderApi::ReqCancelOrder	DFITCTraderSpi:: OnRspCancelOrder	对话流
	做市商报单	DFITCTraderApi:: ReqQuoteInsertOrder	DFITCTraderSpi:: OnRspQuoteInsertOrder	对话流
	做市商撤单	DFITCTraderApi:: ReqQuoteCancelOrder	DFITCTraderSpi:: OnRspQuoteCancelOrder	对话流
	批量撤单	DFITCTraderApi:: ReqCancelAllOrder	DFITCTraderSpi:: OnRspCancelAllOrder	
私有回报	错误回报	N/A	DFITCTraderSpi::OnRtnErrorMsg	私有流
	成交回报	N/A	DFITCTraderSpi::OnRtnMatchedInfo	私有流
	委托回报	N/A	DFITCTraderSpi:: OnRtnOrder	私有流
	撤单回报	N/A	DFITCTraderSpi:: OnRtnCancelOrder	私有流
	做市商报单回报	N/A	DFITCTraderSpi :: OnRtnQuoteInsert	私有流
	做市商撤单回报	N/A	DFITCTraderSpi:: OnRtnQuoteCancel	私有流
查询	委托查询	DFITCTraderApi::ReqQryOrderInfo	DFITCTraderSpi::OnRspQryOrderInfo	查询流
	成交查询	DFITCTraderApi::ReqQryMatchInfo	DFITCTraderSpi::OnRspQryMatchInfo	查询流
	持仓查询	DFITCTraderApi:: ReqQryPosition	DFITCTraderSpi::OnRspQryPosition	查询流
	持仓明细查询	DFITCTraderApi::ReqQryPositionDetail	DFITCTraderSpi::OnRspQryPositionDetail	查询流
	资金查询	DFITCTraderApi:: ReqQryCustomerCapital	DFITCTraderSpi::OnRspCustomerCapital	查询流

	交易所合约查询	DFITCTraderApi:: ReqQryExchangeInstrument	DFITCTraderSpi::OnRspQryExchangeInstrument	查询流
	指定合约查询	DFITCTraderApi::ReqQrySpecifyInstrument	DFITCTraderSpi::OnRspQrySpecifyInstrument	查询流
	套利合约查询	DFITCTraderApi:: ReqQryArbitrageInstrument	DFITCTraderSpi::OnRspArbitrageInstrument	查询流
	交易编码查询	DFITCTraderApi::ReqQryTradeCode	DFITCTraderSpi::OnRspQryTradeCode	查询流
	客户权益计算方式查询	DFITCTraderApi::ReqEquityComputMode	DFITCTraderSpi::OnRspEquityComputMode	查询流
	账单查询	DFITCTraderApi::ReqQryBill	DFITCTraderSpi::OnRspQryBill	查询流
	交易所状态查询	DFITCTraderApi::ReqQryExchangeStatus	DFITCTraderSpi::OnRspQryExchangeStatus	查询流
	报价查询	DFITCTraderApi::ReqQryQuoteOrderInfo	DFITCTraderSpi:: OnRspQryQuoteOrderInfo	查询流
	询价通知查询	DFITCTraderApi:: ReqQryQuoteNotice	DFITCTraderSpi:: OnRspQryQuoteNotice	查询流
	深度行情查询	DFITCTraderApi::ReqQryDepthMarketData	DFITCTraderSpi:: OnRspQryDepthMarketData	查询流
其他	密码修改	DFITCTraderApi::ReqResetPassword	DFITCTraderSpi:: OnRspResetPassword	对话流
	账单确认	DFITCTraderApi::ReqBillConfirm	DFITCTraderSpi::OnRspBillConfirm	对话流
	询价请求	DFITCTraderApi:: ReqForQuote	DFITCTraderSpi:: OnRspForQuote	对话流
	交易所状态通知	N/A	DFITCTraderSpi::OnRtnExchangeStatus	对话流

6. DFITCTraderAPI 使用参考手册

6.1. DFITCTraderApi 接口

从 v1.0.14.xx 版本开始,XSpeed API 去掉了请求结构体的默认构造函数,用户调用接口前,需要进行 memset 清空结构体内容，再填写需要的字段信息

6.1.1. CreateDFITCTraderApi 方法

产生一个 DFITCTraderApi 实例

函数原型：

```
static DFITCTraderApi *CreateDFITCTraderApi(const char *pszFlowPath = "");
```

参数：

pszFlowPath(预留)

返回值：

返回一个指向 DFITCTraderApi 实例的指针。

6.1.2. Init 方法

该方法会和交易服务器建立连接，并启动一个接收线程，同时该方法注册一个回调函数集。

函数原型：

```
int Init( char *pszFrontAddr, DFITCTraderSpi *pSpi);
```

参数：

pszFrontAddr：前置机地址。可以指向同一个 XSpeed 的多个前置，采用如下的格式："tcp://172.16.0.31:10910;tcp://172.16.0.32:10910"的形式

pSpi：指向回调函数集的指针。

返回值：

0：初始化成功

-1：初始化失败，此时需要检查 OnRtnErrorMsg 给出的错误信息

Init 失败原因可能：

1) 填写的 addr 格式不正确或者 addr 中的 ip 地址及端口不正确，如使用交易 API 连接到了行情前置的端口
2) API 版本过高/低（等同于连接的前置版本过低/高），此时，我们在 OnRtnErrorMsg 给出了具体的错误信息，此时可咨询飞创客服或期货公司要正确的 API 版本。
3) 网络问题，可 telnet 连接 ip 及 port，检查是否畅通

6.1.3. Join 方法

等待接口线程结束运行

函数原型：

```
int Join(void);
```

返回值：

0：线程退出成功

-1：线程退出失败

6.1.4. Release 方法

退出 API 各线程，释放 API 的各项资源

函数原型：

```
void Release(void);
```

返回值：无

当调用 CreateDFITCTraderApi 生成一个 API 实例后，退出时必须调用 Release 接口，否则会造成资源泄漏。就像 C 中 malloc 了一块内存，需要 free 一样。

6.1.5. GetVersion 方法

查询当前 API 版本信息

函数原型：

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

参数(传出)：

nMajorVersion: 主版本号;

nMinorVersion: 子版本号;

返回值：

当前版本信息的字符串.

6.1.6. SubscribePrivateTopic 方法

订阅私有流方法.(预留方法)

函数原型：

```
void SubscribePrivateTopic(DFITC_TE_RESUME_TYPE nResumeType);
```

参数：

nResumeType: 私有流订阅类型;

```
enum DFITC_TE_RESUME_TYPE
{
    DFITC_TERT_RESTART = 0,
    DFITC_TERT_RESUME,
    DFITC_TERT_QUICK
};
```

6.1.7. OpenApiLog 方法

打开 API 日志方法.

函数原型：

```
int OpenApiLog(const char * pszLogFileName);
```

参数：

pszLogFileName: 指定的 API 日志文件名;

返回值：

0: 成功;

-1: 失败;

6.1.8. ReqUserLogin 方法

用户发出请求登录。

函数原型：

```
int ReqUserLogin(struct DFITCUserLoginField *pUserLoginData);
```

参数：

pUserLoginData：指向用户登录请求结构的地址。用户请求登录结构：

```
struct DFITCUserLoginField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金账户 ID
    DFITCPasswdType             passwd;              //密码
    DFITCCompanyIDType          companyID;           //厂商 ID
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.9. ReqUserLogout 方法

用户发出退出请求。

函数原型：

```
int ReqUserLogout( struct DFITCUserLogoutField *pUserLogoutData );
```

参数：

pUserLogoutData:指向用户退出请求结构的地址。用户请求退出结构：

```
struct DFITCUserLogoutField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金帐号 ID
    DFITCSessionIDType          sessionID;           //会话 ID
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.10. ReqInsertOrder 方法

用户发出下单请求。

函数原型：

```
int ReqInsertOrder( struct DFITCInsertOrderField *pInsertOrderData );
```

参数：

pInsertOrderData:指向用户请求报单结构地址。用户请求报单结构：

```
struct DFITCInsertOrderField
{
    DFITCAccountIDType      accountID;      //资金账户
    DFITCLocalOrderIDType   localOrderID;   //本地委托号
    DFITCInstrumentIDType   instrumentID;   //合约代码
    DFITCPriceType          insertPrice;    //报单价格
    DFITCAmountType         orderAmount;    //报单数量
    DFITCBuySellTypeType    buySellType;    //买卖标志
    DFITCOpenCloseTypeType  openCloseType;  //开平标志
    DFITCSpeculatorType     speculator;     //投保类型, 支持投机、套利、套保
    DFITCInsertType         insertType;     //自动单类别,普通单(默认)、自动单
    DFITCOrderTypeType      orderType;      //报单类型,支持限价、市价;
    DFITCOrderPropertyType  orderProperty;  //报单附加属性, None/FAK/FOK
    DFITCInstrumentTypeType instrumentType; //合约类型, 可选值：期货、期权
    DFITCAmountType         minMatchAmount; //最小成交量
    DFITCReservedType       reservedType2;  //预留字段 2
    DFITCRequestIDType      lRequestID;     //请求 ID
    DFITCCustomCategoryType customCategory; //自定义类别（业务单元/策略 ID）
    DFITCPriceType          profitLossPrice;//止盈止损价格
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

-3：填入的字段中有非法字段，需要检查各字段是否填写正确

说明：

1. 报单前，需要将结构体进行清空，否则可能会返回”-3”的错误
2. 报止盈止损单（目前只有大连交易所支持），设置 orderType 为 DFITC_PROFIT_LIMITORDER(限价止盈委托)，然后再填入止盈价格即可
3. 报开市自动单（XSpeed 服务器收到开市信号后就立刻将这些预埋单报到交易所），设置 insertType 为 DFITC_AUTO_ORDER(自动单)，否则默认为普通单
4. 报套利委托时，设置 orderType 为 DFITC_ARBITRAGE(套利委托)
5. Linux xspeed 柜台，报止盈止损单，FAK/FOK 单，委托回报 OnRtnOrder 中委托状态为 DFITC_SPD_PLACED，表示**未成交不在队列**状态，与普通订单状态稍有差别

6.1.11. ReqCancelOrder 方法

用户发出撤单请求。

函数原型：

```
int ReqCancelOrder( struct DFITCCancelOrderField *pCancelOrderData );
```

参数：

pCancelOrderData:指向请求撤单结构地址。用户请求撤单结构：

```
struct DFITCCancelOrderField
{
    DFITCAccountIDType      accountID;      //资金账户 ID
    DFITCSPDOrderIDType     spdOrderID;     //柜台委托号
    DFITCLocalOrderIDType   localOrderID;   //本地委托号
    DFITCInstrumentIDType   instrumentID;   //合约代码
    DFITCRequestIDType      lRequestID;     //请求 ID
    DFITCSessionIDType      sessionID;      //会话 ID
    DFITCExchangeIDType     exchangeID;     //交易所编码
    DFITCOrderSysIDType     OrderSysID;     //交易所报单编号
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

说明：

当发起撤单时，可以使用以下 4 种方式进行撤单：

1. 柜台委托号(spdOrderID)；
2. 本地委托号(localOrderID)；
3. 本地委托号(localOrderID) + 会话 ID (sessionID)；
4. 主场单号(orderSysID) + 交易所(exchangeID)；

注意，当传入参数均有效时(指传入的参数均大于零)，撤单逻辑判断优先级为 1>(2,3)>4，情况 2 与 3 是同级关系，如果 API release 之后，就不能单独使用本地委托号进行撤单，需要使用本地委托号+会话 ID 的方式进行撤单，情况 1，2，3 都不满足时，采用第 4 种方式。

6.1.12. ReqQryPosition 方法

用户发出持仓查询请求。

函数原型：

```
int ReqQryPosition( struct DFITCPositionField *pPositionData );
```

参数：

pPositionData：指向请求持仓查询结构地址。用户请求持仓查询结构：

```
struct APISTRUCT DFITCPositionField
{
    DFITCRequestIDType              lRequestID;       //请求 ID
    DFITCAccountIDType              accountID;        //资金账户 ID
    DFITCInstrumentIDType           instrumentID;     //合约代码
    DFITCInstrumentTypeType         instrumentType;   //合约类型
    DFITCIsReturnRealizedPNLType    retRealizedPNL;   //是否返回未持仓合约的
                                                      //平仓盈亏
};
```

说明：如果没有提供合约代码，则查询全部持仓信息

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.13. ReqQryPositionDetail 方法

用户发出持仓明细查询请求。

函数原型：

```
int ReqQryPositionDetail ( struct DFITCPositionDetailField *pPositionDetailData );
```

参数：

pPositionDetailData:指向请求持仓明细查询结构地址。用户请求持仓明细查询结构：

```
struct DFITCPositionDetailField
{
    DFITCRequestIDType          lRequestID;             //请求 ID
    DFITCAccountIDType          accountID;              //资金账户 ID
    DFITCInstrumentIDType       instrumentID;           //合约代码
    DFITCInstrumentTypeType     instrumentType;         //合约类型
};
```

说明：如果没有提供合约代码，则查询所有合约的持仓明细信息

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.14. ReqQryOrderInfo 方法

用户发出当日委托查询请求。

函数原型：

```
int ReqQryOrderInfo (struct DFITCOrderField *pOrderData);
```

参数：

pOrderData：指向请求当日委托查询结构地址。用户请求当日委托查询结构：

```
struct DFITCOrderField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCAccountIDType          accountID;          //资金账户 ID
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCCustomCategoryType     customCategory;     //自定义类别
    DFITCOrderAnswerStatusType  orderStatus;        //委托状态(暂不支持)
    DFITCOrderTypeType          orderType;          //报单类型(暂不支持)
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCSessionIDType          sessionID;          //会话 ID
    DFITCInstrumentIDType       instrumentID;       //合约代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

说明：这里有丰富的查询条件设置，如果想查询所有委托记录，则只需要输入 accountID 和 instrumentType 即可。

查询处于某状态的报单，则中需要将 orderStatus 设置为相应报单状态的值即可，如 3 表示未成交在队列的单子。**（注意 orderStatus 和 orderType 查询条件该版本暂不支持）**

查询指定报单类型的单子，如限价单或市价单等。

可查询某笔委托的信息，只需要输入 spdOrderID 即可，也可使用 localOrderID，但当网络连接断开再重连后，可使用 localOrderID+ sessionID 进行查询。

6.1.15. ReqQryMatchInfo 方法

用户发出当日成交查询请求。

函数原型：

```
int ReqQryMatchInfo (struct DFITCMatchField* pMatchData);
```

参数：

pMatchData：指向请求当日成交查询结构地址。用户请求当日成交查询结构：

```
struct DFITCMatchField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCAccountIDType          accountID;          //资金账户 ID
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCCustomCategoryType     customCategory;     //自定义类别
    DFITCOrderTypeType          orderType;          //报单类型(暂不支持)
```

```
    DFITCSPDOrderIDType          spdOrderID;          //柜台委托号
    DFITCInstrumentIDType        instrumentID;        //合约代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常

说明：这里有丰富的查询条件设置，如果想查询所有委托记录，则只需要输入 accountID 和 instrumentType 即可。当然，也可以查询指定的委托号 spdOrderID 的成交信息。**（注意 orderType 查询条件该版本暂不支持）**

6.1.16. ReqQrySpecifyInstrument 方法

用户发出指定合约信息查询请求。

函数原型：

```
int ReqQrySpecifyInstrument (struct DFITCSpecificInstrumentField* pInstrument);
```

参数：

pInstrument:指向请求指定合约信息查询结构地址。用户请求指定合约信息查询结构：

```
struct DFITCSpecificInstrumentField
{
    DFITCRequestIDType           lRequestID;          //请求 ID
    DFITCAccountIDType           accountID;           //资金账户 ID
    DFITCInstrumentIDType        InstrumentID;        //合约代码
    DFITCExchangeIDType          exchangeID;          //交易所 ID
    DFITCInstrumentTypeType      instrumentType;      //合约类型
    DFITCSpeculatorType          speculator;          //投保类型
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

说明：exchangeID 并非必须输入选项，因为目前期货市场上，一个合约代码在各交易所是唯一存在的。

6.1.17. ReqQryCustomerCapital 方法

用户发出资金查询请求。

函数原型：

```
int ReqQryCustomerCapital( struct DFITCCapitalField *pCapitalData );
```

参数：

pCapitalData：指向请求资金查询结构地址。用户请求资金查询结构：

```
struct DFITCCapitalField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金账户 ID
    DFITCCurrencyType           currencyID;          //币种代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.18. ReqQryExchangeInstrument 方法

查询交易所合约列表。

函数原型：

```
Int          ReqQryExchangeInstrument(struct          DFITCExchangeInstrumentField
*pExchangeInstrumentData );
```

参数：

pExchangeInstrumentData：指向交易所合约查询结构的地址。

```
struct DFITCExchangeInstrumentField
{
    DFITCRequestIDType              lRequestID;              //请求 ID
    DFITCAccountIDType              accountID;               //资金账户 ID
    DFITCExchangeIDType             exchangeID;              //交易所编码
    DFITCInstrumentTypeType         instrumentType;          //合约类型
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

说明：当 exchangeID 为空时，表示查询各交易所的所有合约代码，也可指定查询某交易所的所有合约代码信息。但该接口不能查询到套利合约代码

6.1.19. ReqQryArbitrageInstrument 方法

查询交易所套利合约列表。

函数原型：

```
Int        ReqQryExchangeInstrument(struct        DFITCAbiInstrumentField        *
pAbtriInstrumentData);
```

参数：

pAbtriInstrumentData：指向套利合约结构的地址。

```
struct DFITCAbiInstrumentField
{
    DFITCRequestIDType              lRequestID;             //请求 ID
    DFITCAccountIDType              accountID;              //资金账户 ID
    DFITCExchangeIDType             exchangeID;             //交易所代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

说明：当 exchangeID 为空时，表示查询各交易所的所有合约代码，也可指定查询某交易所的所有合约代码信息。

6.1.20. ReqResetPassword 方法

用户发出密码修改请求。

函数原型：

```
int ReqResetPassword (struct DFITCResetPwdField *pResetPasswordData)
```

参数：

pResetPasswordData：指向密码修改结构的地址。

```
struct DFITCResetPwdField
{
    DFITCRequestIDType              lRequestID;             //请求 ID
    DFITCAccountIDType              accountID;              //资金帐号 ID
    DFITCPasswdType                 oldpasswd;              //旧密码
    DFITCPasswdType                 newpasswd;              //新密码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.21. ReqBillConfirm 方法

用户发出账单确认请求。

函数原型：

```
int ReqBillConfirm(struct DFITCBillConfirmField *pBillConfirmData)
```

参数：

pBillConfirmData：指向账单确认结构的地址。

```
struct DFITCBillConfirmField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金帐号 ID
    DFITCDateType               date;                //确认日期
    DFITCConfirmMarkType        confirmFlag;         //确认标志
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

说明：确认日期格式为：**yyyy.mm.dd**。XSpeed 交易系统，目前是不用进行账单确认即可进行交易，这点与其它主席系统有所不同。

6.1.22. ReqQryTradeCode 方法

用户发出查询交易编码请求。

函数原型：

```
int ReqQryTradeCode(struct DFITCQryTradeCodeField *pTradeCodeData)
```

参数：

pTradeCodeData：指向查询交易编码结构的地址。

```
struct DFITCQryTradeCodeField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金账户 ID
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.23. ReqEquityComputMode 方法

用户发出查询客户权益计算方式请求。

函数原型：

```
int ReqEquityComputMode()
```

返回值：

0：请求发送成功。
-1：请求发送失败。
-2：检测异常。

6.1.24. ReqQryBill 方法

用户发出账单查询请求。

函数原型：

```
int ReqQryBill(struct DFITCQryBillField *pQryBillData)
```

参数：

pQryBillData：指向账单查询结构的地址。

```
struct DFITCQryBillField
{
    DFITCRequestIDType              lRequestID;         //请求 ID
    DFITCAccountIDType              accountID;          //资金账户
    DFITCDateType                   date;               //查询日期
};
```

返回值：

0：请求发送成功。
-1：请求发送失败。
-2：检测异常。

说明：查询日期格式为：yyyy.mm.dd

6.1.25. ReqTradingDay 方法

用户发出交易日查询请求。

函数原型：

```
int ReqTradingDay(struct DFITCTradingDayField *pTradingDay)
```

参数：

pTradingDay：指向交易日查询结构的地址。

```
struct DFITCTradingDayField
{
    DFITCRequestIDType              lRequestID;             //请求 ID
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.26. ReqConfirmProductInfo 方法

用户发出厂商 ID 确认请求。

函数原型：

```
int ReqConfirmProductInfo(struct DFITCProductField * pConfirmProductData)
```

参数：

pConfirmProductData：指向厂商 ID 确认结构的地址。

```
struct DFITCProductField
{
    DFITCProductIDType              productID;          //产品编号
    DFITCSoftwareVendorIDType       vendorID;           //软件供应商编号
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.27. ReqQryExchangeStatus 方法

用户发出交易所状态查询请求

函数原型：

```
int           ReqQryExchangeStatus(struct           DFITCQryExchangeStatusField
*pQryExchangeStatusData)
```

参数：

pQryExchangeStatusData：指向交易所状态查询请求结构的地址。

```
struct DFITCQryExchangeStatusField
{
    DFITCRequestIDType              lRequestID;             //请求 ID
    DFITCExchangeIDType             exchangeID;             //交易所编码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.28. ReqQryDepthMarketData 方法

用户发出合约行情查询请求

函数原型：

```
int ReqQryDepthMarketData (struct DFITCQryDepthMarketDataField *pQryDepthMarketData)
```

参数：

pQryDepthMarketData：指向行情查询请求结构地址。

```
struct DFITCQryDepthMarketDataField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCExchangeIDType         exchangeID;         //交易所编码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

说明：exchangeID 可不用填写，输入合约代码即可，一次只能查询一个合约代码的最新行情

6.1.29. ReqQuoteInsert 方法

用户发出做市商报单请求

函数原型：

```
int ReqQuoteInsert(struct DFITCQuoteInsertField * pQuoteInsertOrderData)
```

参数：

pQuoteInsertOrderData：指向做市商报单请求结构的地址。

```
struct DFITCQuoteInsertField
{
    DFITCAccountIDType          accountID;          //资金账号
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCInsertType             insertType;         //自动单类别
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCAmountType             bOrderAmount;       //报单数量（买）
    DFITCAmountType             sOrderAmount;       //报单数量（卖）
    DFITCPriceType              bInsertPrice;       //委托价格（买）
    DFITCPriceType              sInsertPrice;       //委托价格（卖）
```

```
    DFITCOpenCloseTypeType          bOpenCloseType;         //开平标志（买）
    DFITCOpenCloseTypeType          sOpenCloseType;         //开平标志（卖）
    DFITCSpeculatorType             bSpeculator;            //投资类别（买）
    DFITCSpeculatorType             sSpeculator;            //投资类别（卖）
    DFITCStayTimeType               stayTime;               //停留时间
    DFITCCustomCategoryType         customCategory;         //自定义类别
};
```

备注：

stayTime 停留时间字段：仅支持郑州。其它情况可设置为 0

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.30. ReqQuoteCancel 方法

用户发出做市商撤单请求

函数原型：

```
int ReqQuoteCancel(struct DFITCCancelOrderField * pQuoteCancelOrderData)
```

参数：

pQuoteCancelOrderData：指向做市商撤单请求结构的地址。

```
struct DFITCCancelOrderField
{
    DFITCAccountIDType              accountID;              //资金账户 ID
    DFITCSPDOrderIDType             spdOrderID;             //柜台委托号
    DFITCLocalOrderIDType           localOrderID;           //本地委托号
    DFITCInstrumentIDType           instrumentID;           //合约代码
    DFITCRequestIDType              lRequestID;             //请求 ID
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.31. ReqCancelAllOrder 方法

用户发出做市商全部撤单请求

函数原型：

```
int ReqCancelAllOrder(struct DFITCCancelAllOrderField * pCancelAllOrderData)
```

参数：

pCancelAllOrderData：指向做市商全部撤单请求结构的地址。

```
struct DFITCCancelOrderField
{
    DFITCRequestIDType  lRequestID;      //请求 ID
    DFITCAccountIDType  accountID;       //资金账户 ID
    DFITCExchangeIDType exchangeID;      //交易所编码(目前只支持大商所)
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

说明：

该指令目前只支持大商所的做市商报价订单，其它交易所暂不支持，普通期货期权订单也不支持

6.1.32. ReqForQuote 方法

用户发出询价请求

函数原型：

```
int ReqForQuote(struct DFITCForQuoteField * pForQuoteData)
```

参数：

pForQuoteData：指向用户发出询价请求结构的地址。

```
struct DFITCForQuoteField
{
    DFITCRequestIDType      lRequestID;          //请求 ID
    DFITCAccountIDType      accountID;           //资金账户 ID
    DFITCInstrumentIDType   instrumentID;        //合约代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

6.1.33. ReqQryQuoteOrderInfo 方法

用户发出应/报价查询请求

函数原型：

```
int ReqQryQuoteOrderInfo(struct DFITCQuoteOrderField * pQuoteOrderData)
```

参数：

pQuoteOrderData：指向用户发出应/报价查询请求结构的地址。

```
struct DFITCQuoteOrderField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCExchangeIDType         exchangeID;          //交易所
    DFITCAccountIDType          accountID;           //资金账户
    DFITCInstrumentIDType       instrumentID;        //合约代码
    DFITCLocalOrderIDType       localOrderID;        //本地委托号
    DFITCSPDOrderIDType         spdOrderID;          //柜台委托号
    DFITCOrderAnswerStatusType  orderStatus;         //委托状态
};
```

返回值：

0：请求发送成功。
-1：请求发送失败。
-2：检测异常。

6.1.34. ReqQryQuoteNotice 方法

用户发出询价通知查询请求

函数原型：

```
int ReqQryQuoteNotice (struct DFITCQryQuoteNoticeField * pQryQuoteNoticeData)
```

参数：

pQryQuoteNoticeData：指向用户发出查询询价通知请求结构地址。

```
struct DFITCQryQuoteNoticeField
{
    DFITCAccountIDType          accountID;           //资金账户
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCExchangeIDType         exchangeID;          //交易所
    DFITCInstrumentIDType       instrumentID;        //合约代码
};
```

返回值：

0：请求发送成功。
-1：请求发送失败。
-2：检测异常。

6.1.35. ReqQryForQuote 方法

用户发出询价委托查询请求

函数原型：

```
int ReqQryForQuote(struct DFITCQryForQuoteField * pQryForQuoteData)
```

参数：

pQryForQuoteData：指向用户发出查询询价委托请求结构地址。

```
struct DFITCQryForQuoteField
{
    DFITCRequestIDType          lRequestID;             //请求 ID
    DFITCAccountIDType          accountID;              //资金账户 ID
    DFITCInstrumentIDType       instrumentID;           //合约代码
    DFITCExchangeIDType         exchangeID;             //交易所
};
```

返回值：

0：请求发送成功。
-1：请求发送失败。
-2：检测异常。

6.1.36. ReqQryTransferBank 方法

用户发出查询转账银行请求.

函数原型：

```
int ReqQryTransferBank(struct DFITCQryTransferBankField * pQryTransferBank);
```

参数：

pQryTransferBank: 指向用户发出查询转账银行请求结构地址;

```
struct DFITCQryTransferBankField
{
    DFITCAccountIDType      accountID;          //资金账号
    DFITCBankIDType         bankID;             //银行代码
    DFITCRequestIDType      lRequestID;         //请求 ID
};
```

返回值：

0: 请求发送成功;
-1: 请求发送失败;
-2: 检测异常;

6.1.37. ReqQryTransferSerial 方法

用户发出查询转账流水请求.

函数原型：

```
int       ReqQryTransferSerial(struct       DFITCQryTransferSerialField       *
pQryTransferSerial);
```

参数：

pQryTransferSerial: 指向用户发出查询转帐流水请求结构地址;

```
struct DFITCQryTransferSerialField
{
    DFITCAccountIDType        accountID;        //资金账号
    DFITCBankIDType           bankID;           //银行代码
    DFITCBankAccountType      bankAccount;      //银行账号
    DFITCRequestIDType        lRequestID;       //请求 ID
};
```

返回值：

0：请求发送成功；
-1：请求发送失败；
-2：检测异常；

6.1.38. ReqFromBankToFutureByFuture 方法

用户发出期货发起银行资金转期货请求.

函数原型：

```
int ReqFromBankToFutureByFuture(struct DFITCReqTransferField * pReqTransfer);
```

参数：

pReqTransfer：指向用户发出银行资金转期货请求结构地址；

```
struct DFITCReqTransferField
{
    DFITCBankIDType           bankID;           //银行代码
    DFITCBankAccountType      bankAccount;      //银行账号
    DFITCPasswdType           bankPassword;     //银行密码
    DFITCAccountIDType        accountID;        //投资者账号
    DFITCPasswdType           password;         //期货密码
    DFITCCurrencyType         currency;         //币种代码
    DFITCPriceType            tradeAmount;      //转账金额
    DFITCRequestIDType        lRequestID;       //请求 ID
};
```

返回值：

0：请求发送成功；
-1：请求发送失败；
-2：检测异常；

6.1.39. ReqFromFutureToBankByFuture 方法

用户发出期货发起银行资金转期货请求.

函数原型：

```
int ReqFromFutureToBankByFuture(struct DFITCReqTransferField * pReqTransfer);
```

参数：

pReqTransfer: 指向用户发出期货资金转银行请求结构地址;

```
struct DFITCReqTransferField
{
    DFITCBankIDType             bankID;             //银行代码
    DFITCBankAccountType        bankAccount;        //银行账号
    DFITCPasswdType             bankPassword;       //银行密码
    DFITCAccountIDType          accountID;          //投资者账号
    DFITCPasswdType             password;           //期货密码
    DFITCCurrencyType           currency;           //币种代码
    DFITCPriceType              tradeAmount;        //转账金额
    DFITCRequestIDType          lRequestID;         //请求 ID
};
```

返回值：

0: 请求发送成功;
-1: 请求发送失败;
-2: 检测异常;

6.1.40. ReqQryExchangeRate 方法

用户发出汇率查询请求.

函数原型：

```
int ReqQryExchangeRate(struct DFITCQryExchangeRateField *pQryExchangeRate);
```

参数：

pQryExchangeRate: 指向用户发出汇率查询请求结构地址;

```
struct DFITCQryExchangeRateField
{
    DFITCRequestIDType          lRequestID;             //请求 ID
    DFITCCurrencyType           fromCurrencyID;         //源币种
    DFITCCurrencyType           toCurrencyID;           //目标币种
};
```

返回值：

0: 请求发送成功;
-1: 请求发送失败;
-2: 检测异常;

6.1.41. ReqPricesTrigger 方法

用户发出行情触发请求.

函数原型：

```
int ReqPricesTrigger(struct DFITCPricesTriggerField * pPricesTriggerData);
```

参数：

pPricesTriggerData: 指向用户发出行情触发请求结构地址;

```
struct DFITCPricesTriggerField
{
    DFITCAccountIDType          accountID;        //资金账户
    DFITCLocalOrderIDType       localOrderID;     //本地委托号
    DFITCInstrumentIDType       instrumentID;     //合约代码
    DFITCPriceType              insertPrice;      //委托价格
    DFITCAmountType             orderAmount;      //委托数量
    DFITCSpeculatorType         speculator;       //投保类型
    DFITCExtOrderPriceTypeType  extOrderPriceType;//条件单报单类型
    DFITCBuySellTypeType        buySellType;      //买卖标志
    DFITCOpenCloseTypeType      openCloseType;    //开平标志
    DFITCRequestIDType          lRequestID;       //请求 ID
    DFITCCompareFlagType        compareFlag;      //比较标志(价格)
    DFITCPriceType              comparePrice;     //触发价格(触发条件为行情价
                                                  //格大于或小于等于触发价格)
    DFITCPriceReferenceType     priceReference;   //价格参照
    DFITCBreakDownTimesType     breakDownTimes;   //击穿次数
    DFITCDateType               validate;         //有效日期(yyyy.mm.dd,
                                                  //暂不支持)
    DFITCFrozenTypeType         frozenType;       //是否冻结资金类型
    DFITCInstrumentTypeType     instrumentType;   //合约类型
    DFITCAmountType             limitAmount;      //数量限制
    DFITCCompareFlagType        qtyCmpFlag;       //比较标志(数量)
    DFITCTriggerType            triggerType;      //触发类型
    DFITCBreakDownTypeType      breakDownType;    //击穿属性
};
```

返回值:

0: 请求发送成功;
-1: 请求发送失败;
-2: 检测异常;

6.1.42. ReqQryExtOrder 方法

用户发出条件单查询请求.

函数原型:

```
int ReqQryExtOrder(struct DFITCQryExtOrderField  *pQryEXOrderData);
```

参数:

pQryEXOrderData: 指向用户发出条件单查询请求结构地址;

```
struct DFITCQryExtOrderField
{
    DFITCAccountIDType          accountID;          //资金账户
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCSPDOrderIDType         extSpdOrderID;      //条件单编号(暂不支持)
    DFITCExtOrderType           extOrderType;       //条件单类型
};
```

返回值：

0：请求发送成功；
-1：请求发送失败；
-2：检测异常；

6.1.43. ReqCancelExtOrder 方法

用户发出条件单撤单请求.

函数原型：

```
int ReqCancelExtOrder(struct DFITCCancelExtOrderField  *pCancelEXOrderData);
```

参数：

pCancelEXOrderData：指向用户发出条件单撤单请求结构地址；

```
struct DFITCCancelExtOrderField
{
    DFITCAccountIDType          accountID;          //资金账户
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCSPDOrderIDType         extSpdOrderID;      //条件单编号
    DFITCExtOrderType           extOrderType;       //条件单类型
};
```

返回值：

0：请求发送成功；
-1：请求发送失败；
-2：检测异常；

6.1.44. ReqQryBillConfirm 方法

用户发出查询结算账单信息确认请求.(预留方法)

函数原型：

```
int ReqQryBillConfirm(struct DFITCQryBillConfirmField * pQryBillConfirm);
```

参数：

pQryBillConfirm：指向用户发出查询账单是否确认的请求结构地址；

```
struct APISTRUCT DFITCQryBillConfirmField
{
    DFITCRequestIDType          lRequestID;       //请求 ID
    DFITCAccountIDType          accountID;        //资金账户
};
```

返回值：

0：请求发送成功；
-1：请求发送失败；
-2：检测异常；

6.1.45. ReqQryTradingNotice 方法

用户发出查询交易通知请求.(预留方法)

函数原型：

```
int ReqQryTradingNotice(struct DFITCQryTradingNoticeField * pQryTradingNotice);
```

参数：

pQryTradingNotice：指向用户发出查询交易通知请求结构地址；

```
struct APISTRUCT DFITCQryTradingNoticeField
{
    DFITCAccountIDType          accountID;        //资金帐号 ID
    DFITCRequestIDType          lRequestID;       //请求 ID
};
```

返回值：

0：请求发送成功；
-1：请求发送失败；
-2：检测异常；

6.2. DFITCTraderSpi 接口

DFITCTraderSpi 实现了事件通知接口，用户需要实现此类接口，编写事件处理方法来处理用户感兴趣的事件。

6.2.1. OnFrontConnected 方法

该方法是在Api和前置机建立连接后被调用,该调用仅仅是说明tcp连接已经建立成功。用户需要自行登录才能进行后续的业务操作，当然也可以在该函数内进行登录请求。连接失败则此方法不会被调用。

函数原型：

```
void OnFrontConnected();
```

6.2.2. OnFrontDisconnected 方法

该方法是在 Api 和前置机连接断开后被调用。

函数原型：

```
void OnFrontDisconnected(int nReason);
```

6.2.3. OnRspUserLogin 方法

当用户发出登录请求后，前置机返回响应时此方法会被调用，通知用户登录是否成功。

函数原型：

```
Void OnRspUserLogin(struct DFITCUserLoginInfoRtnField *pUserLoginInfoRtn,struct
 DFITCErrorRtnField *pErrorInfo )
```

参数：

pUserLoginInfoRtn：返回用户登录信息结构地址：

```
struct DFITCUserLoginInfoRtnField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCAccountIDType          accountID;          //资金帐号 ID
    DFITCAccountLoginResultType loginResult;        //登录结果
    DFITCLocalOrderIDType       initLocalOrderID;   //初始本地委托号
    DFITCSessionIDType          sessionID;          //sessionID(会话 ID)
    DFITCErrorIDType            nErrorID;           //错误 ID
    DFITCErrorMsgInfoType       errorMsg;           //错误信息
    DFITCDateType               tradingDay;         //交易日 yyyy.mm.dd
    DFITCTimeType               DCEtime;            //大商所时间
    DFITCTimeType               SHFETime;           //上期所时间
    DFITCTimeType               CFFEXTime;          //中金所时间
    DFITCTimeType               CZCETime;           //郑商所时间
    DFITCTimeType               INETime;            //上能所时间
};
```

当 loginResult 为 0 时表示登录成功，且登录成功时，pErrorInfo 为 NULL，否则 pErrorInfo 中将包含错误 ID 和错误信息。成功时，用户将获取一个会话 ID 和初始报单编号 initLocalOrderID，默认是从 1 开始，可选方式是每次报单，依次递增即可。

6.2.4. OnRspUserLogout 方法

当用户发出退出请求后，前置机返回响应此方法会被调用，通知用户退出状态。

函数原型：

```
void        OnRspUserLogout(        struct        DFITCUserLogoutInfoRtnField
*pUserLogoutInfoRtn,struct
 DFITCErrorRtnField *pErrorInfo );
```

参数：

pUserLogoutInfoRtn:返回用户退出信息结构地址：

```
struct DFITCUserLogoutInfoRtnField
{
    DFITCRequestIDType              lRequestID;         //请求 ID
    DFITCAccountIDType              accountID;          //资金账户 ID
    DFITCAccountLogoutResultType    logoutResult;       //退出结果
    DFITCErrorIDType                nErrorID;           //错误 ID
    DFITCErrorMsgInfoType           errorMsg;           //错误信息
};
```

6.2.5. OnRspInsertOrder 方法

下单应答。当用户录入报单后，前置返回响应时，该方法会被调用。

函数原型：

```
void OnRspInsertOrder(struct DFITCOrderRspDataRtnField *pOrderRtn,struct
                    DFITCErrorRtnField
*pErrorInfo );
```

参数：

pOrderRtn：返回下单响应信息结构地址。下单响应信息结构：

```
struct DFITCOrderRspDataRtnField
{
    DFITCLocalOrderIDType       localOrderID;    //本地委托号
    DFITCSPDOrderIDType         spdOrderID;      //柜台委托号
    DFITCOrderAnswerStatusType  orderStatus;     //委托状态
    DFITCRequestIDType          lRequestID;      //请求 ID
    DFITCPriceType              fee;             //手续费,该字段仅供下单时使用
    DFITCPriceType              margin;    //冻结保证金,该字段仅供下单时使用
    DFITCCustomCategoryType     customCategory;  //自定义类别
    DFITCAccountIDType          accountID;       //资金账户 ID
    DFITCInstrumentIDType       instrumentID;    //合约代码
    DFITCSessionIDType          sessionID;       //会话 ID
    DFITCExchangeIDType         exchangeID;      //交易所
    DFITCBuySellTypeType        buySellType;     //买卖
    DFITCOpenCloseTypeType      openCloseType;   //开平
    DFITCInstrumentTypeType     instrumentType;  //合约类型
    DFITCSpeculatorType         speculator;      //投资类别
```

```
    DFITCPriceType              insertPrice;     //委托价
    DFITCPriceType              profitLossPrice; //止盈止损价格
    DFITCAmountType             minMatchAmount;  //最小成交量
    DFITCAmountType             orderAmount;     //委托数量
    DFITCInsertType             insertType;      //自动单类别
    DFITCOrderTypeType          orderType;       //订单类型
    DFITCOrderPropertyType      orderProperty;   //订单属性
    DFITCClientIDType           clientID;        //交易编码
};
```

当报单发生错误时，pErrorInfo 不为 NULL，并在其中包含了错误 ID 及错误信息，和请求报单时的 localOrderID，用于对应客户程序的报单。报单成功时，是表示 XSpeed 柜台系统确认了该笔报单，该报单也同时报到了交易所，但交易所还未确认。

6.2.6. OnRspCancelOrder 方法

撤单应答。当用户撤单后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspCancelOrder(struct DFITCOrderRspDataRtnField * pOrderCanceledRtn,struct DFITCErrorRtnField *pErrorInfo );
```

参数：

pOrderCanceledRtn：返回撤单响应信息结构。撤单响应信息结构：

```
struct DFITCOrderRspDataRtnField
{
    DFITCLocalOrderIDType       localOrderID;    //本地委托号
    DFITCSPDOrderIDType         spdOrderID;      //柜台委托号
    DFITCOrderAnswerStatusType  orderStatus;     //委托状态
    DFITCRequestIDType          lRequestID;      //请求 ID
    DFITCPriceType              fee;             //手续费,该字段仅供下单时使用
    DFITCPriceType              margin;   //冻结保证金,该字段仅供下单时使用
    DFITCCustomCategoryType     customCategory;  //自定义类别
    DFITCAccountIDType          accountID;       //资金账户 ID
    DFITCInstrumentIDType       instrumentID;    //合约代码
    DFITCSessionIDType          sessionID;       //会话 ID
    DFITCExchangeIDType         exchangeID;      //交易所
    DFITCBuySellTypeType        buySellType;     //买卖
    DFITCOpenCloseTypeType      openCloseType;   //开平
    DFITCInstrumentTypeType     instrumentType;  //合约类型
    DFITCSpeculatorType         speculator;      //投资类别
    DFITCPriceType              insertPrice;     //委托价
    DFITCPriceType              profitLossPrice; //止盈止损价格
    DFITCAmountType             minMatchAmount;  //最小成交量
```

```
    DFITCAmountType             orderAmount;      //委托数量
    DFITCInsertType             insertType;       //自动单类别
    DFITCOrderTypeType          orderType;        //订单类型
    DFITCOrderPropertyType      orderProperty;    //订单属性
    DFITCClientIDType           clientID;         //交易编码
};
```

撤单发生错误时，需要检查 pErrorInfo 是否为 NULL。且客户端收到该响应时，只表示柜台确认了这笔撤单请求。

6.2.7. OnRspQryPosition 方法

查询持仓响应。当用户发出持仓查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspQryPosition( struct DFITCPositionInfoRtnField *pPositionInfoRtn,struct DFITCErrorRtnField *pErrorInfo, bool bIsLast );
```

参数：

bIsLast:表示是否是最后一条消息

pPositionInfoRtn：返回持仓信息结构地址。持仓信息结构：

```
struct DFITCPositionInfoRtnField
{
    DFITCRequestIDType          lRequestID;             //请求 ID
    DFITCAccountIDType          accountID;              //资金帐号 ID
    DFITCExchangeIDType         exchangeID;             //交易所代码
    DFITCInstrumentIDType       instrumentID;           //合约号
    DFITCBuySellTypeType        buySellType;            //买卖
    DFITCPriceType              openAvgPrice;           //开仓均价
    DFITCPriceType              positionAvgPrice;       //持仓均价
    DFITCAmountType             positionAmount;         //持仓量
    DFITCAmountType             totalAvaiAmount;        //总可用
    DFITCAmountType             todayAvaiAmount;        //今可用
    DFITCAmountType             lastAvaiAmount;         //昨可用
    DFITCAmountType             todayAmount;            //今仓
    DFITCAmountType             lastAmount;             //昨仓
    DFITCAmountType             tradingAmount;          //平今挂单量
    DFITCProfitLossType         datePositionProfitLoss; //盯市持仓盈亏
    DFITCProfitLossType         dateCloseProfitLoss;    //(盯市)平仓盈亏
    DFITCProfitLossType         dPremium;               //权利金
    DFITCProfitLossType         floatProfitLoss;        //浮动盈亏
    DFITCProfitLossType         dMargin;                //占用保证金
    DFITCSpeculatorType         speculator;             //投保类别
    DFITCClientIDType           clientID;               //交易编码
    DFITCPriceType              preSettlementPrice;     //昨结算价
```

```
    DFITCInstrumentTypeType    instrumentType;           //合约类型
    DFITCAmountType            yesterdayTradingAmount;   //平昨挂单量
    DFITCProfitLossType        optionValue;              //期权市值
};
```

6.2.8. OnRspQryPositionDetail 方法

查询持仓明细响应。当用户发出持仓明细查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspQryPositionDetail( struct DFITCPositionDetailRtnField *
                    pPositionDetailRtn,
struct DFITCErrorRtnField * pErrorInfo, bool bIsLast );
```

参数：

bIsLast:表示是否是最后一条消息

pPositionDetailRtn：返回持仓明细信息结构地址。持仓明细信息结构：

```
struct DFITCPositionDetailRtnField
{
    DFITCRequestIDType              lRequestID;             //请求 ID
    DFITCAccountIDType              accountID;              //资金帐号 ID
    DFITCExchangeIDType             exchangeID;             //交易所代码
    DFITCInstrumentIDType           instrumentID;           //合约号
    DFITCBuySellTypeType            buySellType;            //买卖
    DFITCPriceType                  openPrice;              //开仓价
    DFITCAmountType                 volume;                 //手数
    DFITCMatchIDType                matchID;                //成交编号
    DFITCDateType                   matchedDate;            //成交日期
    DFITCProfitLossType             datePositionProfitLoss; //盯市持仓盈亏
    DFITCProfitLossType             dateCloseProfitLoss;    //盯市平仓盈亏
    DFITCProfitLossType             floatProfitLoss;        //浮动盈亏
    DFITCProfitLossType             dMargin;                //占用保证金
    DFITCSpeculatorType             speculator;             //投保类别
    DFITCClientIDType               clientID;               //交易编码
    DFITCPriceType                  preSettlementPrice;     //昨结算价
    DFITCInstrumentTypeType         instrumentType;         //合约类型
    DFITCSPDOrderIDType             spdOrderID;             //柜台委托号
    DFITCCustomCategoryType         customCategory;         //自定义类别
    DFITCAmountType                 closeOrderVol;          //平仓委托数量
    DFITCAmountType                 closeMatchVol;          //平仓成交数量
    DFITCPositionDateType           positionDateType;       //持仓日期类型
};
```

6.2.9. OnRspCustomerCapital 方法

资金查询应答。当用户发出资金查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspCustomerCapital(struct DFITCCapitalInfoRtnField
                          *pCapitalInfoRtn,struct
DFITCErrorRtnField *pErrorInfo );
```

参数：

pCapitalInfoRtn：返回资金信息结构地址。资金信息结构：

```
struct DFITCCapitalInfoRtnField
{
    DFITCRequestIDType          requestID;              //请求 ID
    DFITCAccountIDType          accountID;              //资金帐号
    DFITCEquityType             preEquity;              //上日权益
    DFITCEquityType             todayEquity;            //当日客户权益
    DFITCProfitLossType         closeProfitLoss;        //平仓盈亏
    DFITCProfitLossType         positionProfitLoss;     //持仓盈亏
    DFITCProfitLossType         frozenMargin;           //冻结资金
    DFITCProfitLossType         margin;                 //持仓保证金
    DFITCProfitLossType         fee;                    //当日手续费
    DFITCProfitLossType         available;              //可用资金
    DFITCProfitLossType         withdraw;               //可取资金
    DFITCRiskDegreeType         riskDegree;             //风险度
    DFITCPremiumType            todayPremiumIncome;     //本日权利金收入
    DFITCPremiumType            todayPremiumPay;        //本日权利金付出
    DFITCPremiumType            yesterdayPremium;       //昨权利金收付
    DFITCMarketValueType        optMarketValue;         //期权市值
    DFITCProfitLossType         floatProfitLoss;        //浮动盈亏
    DFITCProfitLossType         totFundOut;             //总出金
    DFITCProfitLossType         totFundIn;              //总入金
    DFITCCurrencyType           currencyID;             //币种代码
    DFITCProfitLossType         mortgage;               //质押金额
    DFITCProfitLossType         fundMortgageIn;         //货币质入金额
    DFITCProfitLossType         fundMortgageOut;        //货币质出金额
    DFITCProfitLossType         fundMortgageAvailable;//货币质押余额
};
```

6.2.10. OnRspQryExchangeInstrument 方法

合约查询请求应答。当用户发出合约查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspQryExchangeInstrument( struct
                  DFITCExchangeInstrumentRtnField*pInstrumentData,
struct DFITCErrorRtnField  *pErrorInfo, bool bIsLast );
```

参数：

bIsLast:表示是否是最后一条消息

pInstrumentData：返回合约信息结构地址。合约信息结构：

```
struct DFITCExchangeInstrumentRtnField
{
    DFITCRequestIDType              lRequestID;          //请求 ID
    DFITCExchangeIDType             exchangeID;          //交易所编码
    DFITCInstrumentIDType           instrumentID;        //合约代码
    DFITCVarietyNameType            VarietyName;         //品种名称
    DFITCInstrumentTypeType         instrumentType;      //合约类型
    DFITCAmountType                 orderTopLimit;       //限价委托上限
    DFITCAmountType                 mktOrderTopLimit;    //市价委托上限
    DFITCPriceType                  contractMultiplier; //合约乘数
    DFITCPriceType                  minPriceFluctuatio  //最小变动价位
    DFITCInstrumentMaturityType     instrumentMaturity; //合约最后交易日
    DFITCPriceType                  upperLimitPrice;     //涨停板价
    DFITCPriceType                  lowerLimitPrice;     //跌停板价
    DFITCPriceType                  preClosePrice;       //昨收盘
    DFITCPriceType                  preSettlementPrice; //昨结算价
    DFITCPriceType                  settlementPrice;     //结算价
    DFITCAmountType                 preOpenInterest;     //昨持仓量
    DFITCInstrumentPrefixType       instrumentPrefix;    //合约前缀
    DFITCInstrumenExpirationDateType instrumentExpiration;//合约到期日
    DFITCInstrumentIDType           underlying;           //期权对应的标的合约
代码
    DFITCOptionTypeType             optionType;          //期权类型
    DFITCPriceType                  strikePrice;         //执行价格
    DFITCRiskDegreeType             exchangeRiskDegree; //交易所期权
                                                         //最低保障风险系数
    DFITCPriceType                  minMargin;           //单位（手）期权
                                                         //合约最小保证金
    DFITCAmountType                 tradeSize;           //期权开仓单位
};
```

6.2.11. OnRspArbitrageInstrument 方法

套利合约查询应答。当用户发出套利合约查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspArbitrageInstrument( struct DFITCAbiInstrumentRtnField
*pAbiInstrumentData,struct DFITCErrorRtnField *pErrorInfo, bool bIsLast );
```

参数：

bIsLast:表示是否是最后一条消息

pAbiInstrumentData：返回合约信息结构地址。合约信息结构：

```
struct DFITCAbiInstrumentRtnField
{
    DFITCRequestIDType              lRequestID;         //请求 ID
    DFITCExchangeIDType             exchangeID;         //交易所编码
    DFITCInstrumentIDType           InstrumentID;       //合约代码
    DFITCInstrumentNameType         instrumentName;     //合约名称
    DFITCPriceType                  upperLimitPrice;    //涨停板价
    DFITCPriceType                  lowerLimitPrice;    //跌停板价
    DFITCPriceType                  priceTick;          //最小变动价位
};
```

6.2.12. OnRtnErrorMsg 方法

错误回报。一般在发生交易所废单时（包括报单和撤单请求）会触发该方法。

函数原型：

```
void OnRtnErrorMsg( struct DFITCErrorRtnField  *pErrorInfo );
```

说明：

1. 在发生废单时，如何区分是报单的废单还是撤单的废单（废单即交易所拒绝的订单操作请求）：撤单废单返回的 ErrorID 可能为:-102(老 windows 柜台返回),-910(linux 柜台返回),-920(做市撤单废单返回)，其它 ID 则可能为报单的废单 ID。可根据该类错误 ID 区分该废单回报是报单产生的还是撤单产生的
2. 报撤单产生废单时会返回的错误 ID 为：-100,-900,-901,-902,-905

6.2.13. OnRtnMatchedInfo 方法

成交回报，当委托成功交易后此方法会被调用。

函数原型：

```
void OnRtnMatchedInfo( struct DFITCMatchRtnField *pRtnMatchData );
```

参数：

pRtnMatchData：指向成交回报的结构。成交回报数据结构：

```
struct DFITCMatchRtnField
{
    DFITCLocalOrderIDType       localOrderID;   //本地委托号
    DFITCOrderSysIDType         OrderSysID;     //报单编号(交易所报单编号)
    DFITCMatchIDType            matchID;        //成交编号
    DFITCInstrumentIDType       instrumentID;   //合约代码
    DFITCBuySellTypeType        buySellType;    //买卖
    DFITCOpenCloseTypeType      openCloseType;  //开平标志
    DFITCPriceType              matchedPrice;   //成交价格
    DFITCAmountType             orderAmount;    //委托数量
    DFITCAmountType             matchedAmount;  //成交数量
    DFITCDateType               matchedTime;    //成交时间
    DFITCPriceType              insertPrice;    //报价
    DFITCSPDOrderIDType         spdOrderID;     //柜台委托号
    DFITCMatchType              matchType;      //成交类型
    DFITCSpeculatorType         speculator;     //投保
    DFITCExchangeIDType         exchangeID;     //交易所 ID
    DFITCFeeType                fee;            //手续费
    DFITCSessionIDType          sessionID;      //会话标识
    DFITCInstrumentTypeType     instrumentType; //合约类型
    DFITCAccountIDType          accountID;      //资金账号
    DFITCOrderAnswerStatusType  orderStatus;    //申报结果
    DFITCPriceType              margin;         //开仓为保证金,平仓为解冻保证金
    DFITCPriceType              frozenCapita;   //成交解冻委托冻结的资金
    DFITCAdjustmentInfoType     adjustmentInfo; //组合或对锁的保证金调整信息,
                                                //格式:[合约代码,买卖标志,
                                                //投资类别,调整金额;]
    DFITCCustomCategoryType     customCategory; //自定义类别
    DFITCPriceType              turnover;       //成交金额
    DFITCOrderTypeType          orderType;      //报单类型
    DFITCInsertType             insertType;     //自动单类别
    DFITCClientIDType           clientID;       //交易编码
    DFITCProfitLossType         dateCloseProfitLoss;  //盯市平仓盈亏
    DFITCAmountType             remainingAmount;//剩余数量
};
```

6.2.14. OnRtnOrder 方法

下单委托成功后，或交易所拒绝报单时此方法会被调用。

函数原型：

```
void OnRtnOrder( struct DFITCOrderRtnField *pRtnOrderData );
```

参数：

pRtnOrderData：指向委托回报地址。委托回报数据结构：

```
struct DFITCOrderRtnField
{
    DFITCLocalOrderIDType          localOrderID;          //本地委托号
    DFITCSPDOrderIDType            spdOrderID;            //柜台委托号
    DFITCOrderSysIDType            OrderSysID;            //报单编号
    DFITCOrderAnswerStatusType     orderStatus;           //委托状态
    DFITCSessionIDType             sessionID;             //会话 ID
    DFITCDateType                  SuspendTime;           //挂起时间
    DFITCInstrumentIDType          instrumentID;          //合约代码
    DFITCExchangeIDType            exchangeID;            //交易所
    DFITCBuySellTypeType           buySellType;           //买卖
    DFITCOpenCloseTypeType         openCloseType;         //开平
    DFITCInstrumentTypeType        instrumentType;        //合约类型
    DFITCSpeculatorType            speculator;            //投资类别
    DFITCPriceType                 insertPrice;           //委托价
    DFITCPriceType                 profitLossPrice;       //止盈止损价格
    DFITCAccountIDType             accountID;             //资金账号
    DFITCAmountType                cancelAmount;          //撤单数量
    DFITCAmountType                orderAmount;           //委托数量
    DFITCInsertType                insertType;            //自动单类别
    DFITCOrderTypeType             orderType;             //报单类型
    DFITCSPDOrderIDType            extSpdOrderID;         //条件单编号
    DFITCReservedType              reservedType2;         //预留字段 2
    DFITCCustomCategoryType        customCategory;        //自定义类别
    DFITCOrderPropertyType         orderProperty;         //订单属性
    DFITCAmountType                minMatchAmount;        //最小成交量
    DFITCClientIDType              clientID;              //交易编码
    DFITCErrorMsgInfoType          statusMsg;             //状态信息
    DFITCExtOrderType              extOrderType;          //条件单类型
};
```

说明：

3. 该方法触发时，orderStatus 可能为多种状态：未成交在队列，未成交不在队列(FAK/FOK 单，止盈止损单)、废单状态。废单时，errorMsg 中有对应的错误信息
4. 该方法一般只会触发一次（与其它柜台可能不同），表示交易所已确认该订单（止盈止损在触发后，也会触发一次，但截止 2015.06.10 xspeed 还未实现触发回报）
5. 报单被拒绝时，如休市期间的报单，也会触发，但订单状态为 DFITC_SPD_ERROR

6.2.15. OnRtnCancelOrder 方法

当撤单成功后或撤单被交易所拒绝时该方法会被调用。

函数原型：

```
void OnRtnCancelOrder(struct DFITCOrderCanceledRtnField * pCancelOrderData)
```

参数：

pCancelOrderData：指向撤单回报结构。撤单回报结构

```
struct DFITCOrderCanceledRtnField
{
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCOrderSysIDType         OrderSysID;         //报单编号
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCPriceType              insertPrice;        //报单价格
    DFITCBuySellTypeType        buySellType;        //买卖类型
    DFITCOpenCloseTypeType      openCloseType;      //开平标志
    DFITCAmountType             cancelAmount;       //撤单数量
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCSpeculatorType         speculator;         //投保
    DFITCExchangeIDType         exchangeID;         //交易所 ID
    DFITCDateType               canceledTime;       //撤单时间
    DFITCSessionIDType          sessionID;          //会话标识
    DFITCOrderAnswerStatusType  orderStatus;        //申报结果
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCAccountIDType          accountID;          //资金账号
    DFITCAmountType             orderAmount;        //委托数量
    DFITCPriceType              margin;             //保证金
    DFITCPriceType              fee;                //手续费
    DFITCCustomCategoryType     customCategory;     //自定义类别
    DFITCPriceType              profitLossPrice;    //止盈止损价格
    DFITCAmountType             minMatchAmount;     //最小成交量
    DFITCInsertType             insertType;         //自动单类别
    DFITCClientIDType           clientID;           //交易编码
    DFITCErrorMsgInfoType       statusMsg;          //状态信息
    DFITCOrderPropertyType      orderProperty;      //报单附加属性 FAK/FOK
};
```

说明：

1. 撤单被交易所拒绝时，如休市期间的撤请求单，也会触发，但订单状态为 DFITC_SPD_ERROR，表示该订单未被撤销，需要在开市后再撤

6.2.16. OnRspQryOrderInfo 方法

查询当日委托响应，当用户发出委托查询后，该方法会被调用。

函数原型：

```
void OnRspQryOrderInfo(struct DFITCOrderCommRtnField * pRtnOrderData, struct
DFITCErrorRtnField * pErrorInfo, bool bIsLast)
```

参数：

bIsLast：表明是否是最后一条响应信息

pRtnOrderData：指向委托回报结构。委托回报数据结构

```
struct DFITCOrderCommRtnField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCOrderAnswerStatusType  orderStatus;        //委托状态
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCBuySellTypeType        buySellType;        //买卖
    DFITCOpenCloseTypeType      openClose;          //开平标志
    DFITCPriceType              insertPrice;        //委托价
    DFITCAmountType             orderAmount;        //委托数量
    DFITCPriceType              matchedPrice;       //成交价格
    DFITCAmountType             matchedAmount;      //成交数量
    DFITCAmountType             cancelAmount;       //撤单数量
    DFITCInsertType             insertType;         //自动单类别
    DFITCSpeculatorType         speculator;         //投保
    DFITCDateType               commTime;           //委托时间
    DFITCDateType               submitTime;         //申报时间
    DFITCClientIDType           clientID;           //交易编码
    DFITCExchangeIDType         exchangeID;         //交易所 ID
    DFITCFrontAddrType          operStation;        //委托地址
    DFITCAccountIDType          accountID;          //客户号
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCSessionIDType          sessionId;          //会话 ID
    DFITCReservedType           reservedType2;      //预留字段 2
    DFITCOrderSysIDType         OrderSysID;         //报单编号
    DFITCCustomCategoryType     customCategory;     //自定义类别
    DFITCPriceType              margin;             //保证金
    DFITCPriceType              fee;                //手续费
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCPriceType              profitLossPrice;    //止损止盈价
    DFITCOrderTypeType          orderType;          //报单类别
    DFITCOrderPropertyType      orderProperty;      //订单属性
};
```

在委托查询返回的数据中，sessionID 表示当时下单时的该单子的会话 ID。localOrderID 也是下单时该单子的本地委托号，如果同一账号从多个客户端下单，则查询返回的 localOrderID 可能是重复的。

6.2.17. OnRspQryMatchInfo 方法

查询当日成交信息响应，当用户发出成交查询后该方法会被调用。

函数原型：

```
void OnRspQryMatchInfo(struct DFITCMatchedRtnField * pRtnMatchData, struct
DFITCErrorRtnField * pErrorInfo, bool bIsLast){};
```

参数：

bIsLast：表明是否是最后一条响应信息。

pRtnMatchData：指向成交回报地址。成交回报结构：

```
struct DFITCMatchedRtnField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCSPDOrderIDType         spdOrderID;          //柜台委托号
    DFITCAccountIDType          accountID;           //资金账号
    DFITCExchangeIDType         exchangeID;          //交易所 ID
    DFITCInstrumentIDType       instrumentID;        //合约代码
    DFITCBuySellTypeType        buySellType;         //买卖
    DFITCOpenCloseTypeType      openClose;           //开平
    DFITCPriceType              matchedPrice;        //成交价格
    DFITCAmountType             matchedAmount;       //成交数量
    DFITCPriceType              matchedMort;         //成交金额
    DFITCSpeculatorType         speculator;          //投保类别
    DFITCDateType               matchedTime;         //成交时间
    DFITCMatchIDType            matchedID;           //成交编号
    DFITCLocalOrderIDType       localOrderID;        //本地委托号
    DFITCClientIDType           clientID;            //交易编码
    DFITCMatchType              matchType;           //成交类型
    DFITCInstrumentTypeType     instrumentType;      //合约类型
    DFITCSessionIDType          sessionId;           //会话 ID
    DFITCReservedType           reservedType2;       //预留字段 2
    DFITCCustomCategoryType     customCategory;      //自定义类别
    DFITCPriceType              fee;                 //手续费
    DFITCOrderTypeType          orderType;           //报单类型
    DFITCOrderSysIDType         OrderSysID;          //报单编号
};
```

6.2.18. OnRspQrySpecifyInstrument 方法

查询指定合约信息响应，当用户发出查询制定合约指令后该方法会被调用。

函数原型：

```
void OnRspQrySpecifyInstrument(struct DFITCInstrumentRtnField *
pInstrument,struct DFITCErrorRtnField  * pErrorInfo, bool bIsLast)
```

参数：

pInstrument：指向返回的合约信息结构的地址，该结构如下：

```
Struct DFITCInstrumentRtnField
{
    DFITCRequestIDType          lRequestID;           //请求 ID
    DFITCInstrumentIDType       instrumentID;         //合约代码
    DFITCRatioType              longMarginRatio;      //多头保证金率
    DFITCRatioType              shortMarginRatio;     //空头保证金率
    DFITCPriceType              longMarginRatioByVolume;//多头保证金费(定额)
    DFITCPriceType              shortMarginRatioByVolume;//空头保证金费(定额)
    DFITCRatioType              openFeeVolRatio;      //开仓手续费 按手数计算
    DFITCRatioType              closeFeeVolRatio;     //平仓手续费 按手数计算
    DFITCRatioType              closeTodayFeeVolRatio;//平今手续费 按手数计算
    DFITCRatioType              openFeeAmtRatio;      //开仓手续费率 按金额计算
    DFITCRatioType              closeFeeAmtRatio;     //平仓手续费率 按金额计算
    DFITCRatioType              closeTodayFeeAmtRatio;//平今手续费率 按金额计算
    DFITCInstrumentTypeType     orderTopLimit;        //委托上限
    DFITCPriceType              contractMultiplier;   //合约乘数
    DFITCPriceType              minimumPriceChange;   //最小变动价位
    DFITCInstrumentTypeType     instrumentType;       //合约类型
    DFITCInstrumentMaturityType instrumentMaturity;   //合约最后交易日
    DFITCComputeModeType        computeMode;          //计算方式
    DFITCPriceType              atMoneyNorm;          //平值按定额(卖期权)
    DFITCPriceType              upperLimitPrice;      //涨停板价
    DFITCPriceType              lowerLimitPrice;      //跌停板价
    DFITCPriceType              preClosePrice;        //昨收盘
    DFITCPriceType              preSettlementPrice;   //昨结算价
    DFITCPriceType              settlementPrice;      //结算价
    DFITCAmountType             preOpenInterest;      //昨持仓量
    DFITCRatioType              optExecRatio;         //期权：行权按比例
                                                      //期货：交割按比例
    DFITCRatioType              optExecRatioPerVol;   //期权：行权按定额
                                                      //期货：交割按定额
};
```

6.2.19. OnRtnTradingNotice 方法

交易通知响应:用于接收 XSPEED 柜台手动发送通知，即支持指定客户，也支持系统广播。

函数原型：

```
void OnRtnTradingNotice(DFITCTradingNoticeInfoField *pTradingNoticeInfo)
```

参数：

pTradingNoticeInfo：返回用户事件通知结构的地址，该结构如下：

```
struct APISTRUCT DFITCTradingNoticeInfoField
{
    DFITCAccountIDType        accountID;          //资金帐号 ID
    DFITCTimeType             sendTime;           //发送时间
    DFITCContentType          fieldContent;       //消息正文
    DFITCNoticeType           noticeType;         //消息类型
};
```

6.2.20. OnRspResetPassword 方法

密码修改响应，当用户发出密码修改指令指令后该方法会被调用。

函数原型：

```
void OnRspResetPassword(struct DFITCResetPwdRspField *pResetPassword,struct
DFITCErrorRtnField * pErrorInfo)
```

参数：

pResetPassword：返回密码修改结构的地址，该结构如下：

```
struct DFITCResetPwdRspField
{
    DFITCRequestIDType            lRequestID;          //请求 ID
    DFITCAccountIDType            accountID;           //资金账户 ID
    DFITCExecStateType            execState;           //状态标志
};
```

6.2.21. OnRspBillConfirm 方法

账单确认响应，当用户发出账单确认指令后该方法会被调用。

函数原型：

```
void OnRspBillConfirm(struct DFITCBillConfirmRspField *pBillConfirm,struct
DFITCErrorRtnField     * pErrorInfo)
```

参数：

pBillConfirm：返回账单确认结构的地址，该结构如下：

```
struct DFITCBillConfirmRspField
{
    DFITCRequestIDType              lRequestID;             //请求 ID
    DFITCAccountIDType              accountID;              //资金账户 ID
    DFITCExecStateType              execState;              //状态标志
};
```

6.2.22. OnRspQryTradeCode 方法

查询交易编码响应，当用户发出查询交易编码指令后该方法会被调用。

函数原型：

```
void OnRspQryTradeCode(struct DFITCQryTradeCodeRtnField *pTradeCode,struct
DFITCErrorRtnField * pErrorInfo, bool bIsLast)
```

参数：

pTradeCode：返回交易编码结构的地址，该结构如下：

```
struct DFITCQryTradeCodeRtnField
{
    DFITCRequestIDType              lRequestID;         //请求 ID
    DFITCAccountIDType              accountID;          //资金账户
    DFITCExchangeIDType             exchangeCode;       //交易所编码
    DFITCClientIDType               clientID;           //交易编码
    DFITCClientStatusType           clientStatus;       //交易编码状态
    DFITCSpeculatorType             clientIDType;       //交易编码类型
};
```

6.2.23. OnRspEquityComputMode 方法

查询客户权益计算方式响应，当用户发出查询客户权益计算方式指令后该方法会被调用。

函数原型：

```
void OnRspEquityComputMode(struct DFITCEquityComputModeRtnField
*pEquityComputMode)
```

参数：

pEquityComputMode：返回客户权益计算方式结构的地址，该结构如下：

```
struct DFITCEquityComputModeRtnField
{
    DFITCCapControlModeType         capConMode;     //资金控制方式
    DFITCPriceNoteType              priceNote;      //期权保证金计算说明
```

```
};
```

6.2.24. OnRspQryBill 方法

查询账单响应，当用户发出查询账单指令后该方法会被调用。

函数原型：

```
void OnRspQryBill(struct DFITCQryBillRtnField *pQryBill,struct DFITCErrorRtnField
*pErrorInfo, bool bIsLast)
```

参数：

pQryBill：返回账单查询结构的地址，该结构如下：

```
struct DFITCQryBillRtnField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金账户
    DFITCMsgInfoType            message;             //返回信息
};
```

6.2.25. OnRspTradingDay 方法

查询交易日响应，当用户发出查询交易日期指令后该方法会被调用。

函数原型：

```
void OnRspTradingDay(struct DFITCTradingDayRtnField * pTradingDayRtnData)
```

参数：

pTradingDayRtnData：返回交易日查询结构的地址，该结构如下：

```
struct DFITCTradingDayRtnField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCDateType               date;                //交易日 yyyy.mm.dd
};
```

6.2.26. OnRspConfirmProductInfo 方法

厂商 ID 确认响应，当用户发出厂商 ID 确认指令后该方法会被调用。

函数原型：

```
void OnRspConfirmProductInfo(struct DFITCProductRtnField * pProductRtnData)
```

参数：

pProductRtnData：返回厂商 ID 确认结构的地址，该结构如下：

```
struct DFITCProductRtnField
{
    DFITCProductIDType              productID;              //产品编号
    DFITCSoftwareVendorIDType       vendorID;               //软件供应商
    DFITCProductOnlineCountType     productOnlineCount;     //产品在线数量
    DFITCBrokerInfoType             brokerInfoName;         /期货公司名称
    DFITCFrontIDType                frontID;                //前置机 ID
};
```

6.2.27. OnRspQryExchangeStatus 方法

交易所状态查询响应，当用户发出交易所状态查询指令后该方法会被调用。

函数原型：

```
void OnRspQryExchangeStatus(struct DFITCExchangeStatusRspField *
pRspExchangeStatusData)
```

参数：

pRspExchangeStatusData：返回交易所状态结构的地址，该结构如下：

```
struct DFITCExchangeStatusRspField
{
    DFITCRequestIDType              lRequestID;         //请求 ID
    DFITCExchangeStatusType         exchangeStatus;     //交易所状态
    DFITCExchangeIDType             exchangeID;         //交易所编码
};
```

备注：

若收不到开闭市信号，可先咨询期货公司运维同事，是否在报盘的配置中，打开了<PushJYZT Value=”YES”/>的选项

6.2.28. OnRtnExchangeStatus 方法

交易所状态通知，当交易所有交易状态变化后该方法会被调用。

函数原型：

```
void OnRtnExchangeStatus(struct DFITCExchangeStatusRtnField *
pRtnExchangeStatusData)
```

参数：

pRtnExchangeStatusData：返回交易所状态通知结构的地址，该结构如下：

```
struct APISTRUCT DFITCExchangeStatusRtnField
{
    DFITCExchangeIDType             exchangeID;          //交易所
    DFITCInstrumentIDType           instrumentID;        //合约代码
    DFITCExchangeStatusType         exchangeStatus;      //交易所状态
    DFITCTimeType                   enterTime;           //进入本状态时间
    DFITCInstStatusEnterReasonType enterReason;         //进入本状态原因
};
```

备注：目前只支持交易所市场状态推送

6.2.29. OnRspQuoteInsert 方法

做市商报单响应，当用户发出做市商报单请求指令后该方法会被调用

函数原型：

```
void OnRspQuoteInsert(struct DFITCQuoteRspField * pRspQuoteData, struct
DFITCErrorRtnField * pErrorInfo)
```

参数：

pRspQuoteData：返回做市商报单响应，该结构如下：

```
struct DFITCQuoteRspField
{
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCPriceType              fee;                //手续费(仅报价使用)
    DFITCPriceType              margin;             //保证金(仅报价使用)
    DFITCDateType               orderTime;          //委托时间(仅报价使用)
    DFITCOrderAnswerStatusType  orderStatus;        //委托状态
    DFITCCustomCategoryType     customCategory;     //自定义类别
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCAccountIDType          accountID;          //资金账号
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCSessionIDType          sessionID;          //会话 ID
```

```
    DFITCClientIDType           clientID;           //交易编码
};
```

6.2.30. OnRtnQuoteInsert 方法

做市商报单回报，当用户发出做市商报单请求指令后该方法会被调用

函数原型：

```
void OnRtnQuote(struct DFITCQuoteRtnField * pRtnQuoteData)
```

参数：

pRtnQuoteData：返回做市商报单回报，该结构如下：

```
struct DFITCQuoteRtnField
{
    DFITCExchangeIDType         exchangeID;         //交易所
    DFITCClientIDType           clientID;           //交易编码
    DFITCOrderSysIDType         orderSysID;         //报单编号
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCSeatCodeType           seatCode;           //席位代码
    DFITCOpenCloseTypeType      bOpenCloseType;     //开平标志（买）
    DFITCOpenCloseTypeType      sOpenCloseType;     //开平标志（卖）
    DFITCSpeculatorType         speculator;         //投资类别
    DFITCAmountType             bOrderAmount;       //委托数量（买）
    DFITCAmountType             sOrderAmount;       //委托数量（卖）
    DFITCPriceType              bInsertPrice;       //委托价   （买）
    DFITCPriceType              sInsertPrice;       //委托价   （卖）
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCAccountIDType          accountID;          //资金账号
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCDateType               suspendTime;        //挂单时间
    DFITCEntrusTellerType       entrusTeller;       //委托柜员
    DFITCOrderAnswerStatusType  orderStatus;        //委托状态
    DFITCSessionIDType          sessionID;          //会话 ID
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCErrorMsgInfoType       errorMsg;           //错误信息
    DFITCCustomCategoryType     customCategory;     //自定义类别
};
```

6.2.31. OnRspQuoteCancel 方法

做市商撤单响应，当用户发出做市商撤单请求指令后该方法会被调用

函数原型：

```
void OnRspQuoteCancel( struct DFITCQuoteRspField * pRspQuoteCanceledData,struct DFITCErrorRtnField * pErrorInfo)
```

参数：

pRspQuoteCanceledData：返回做市商撤单响应，该结构如下：

```
struct DFITCQuoteRspField
{
    DFITCLocalOrderIDType        localOrderID;        //本地委托号
    DFITCSPDOrderIDType          spdOrderID;          //柜台委托号
    DFITCRequestIDType           lRequestID;          //请求 ID
    DFITCPriceType               fee;                 //手续费(仅报价使用)
    DFITCPriceType               margin;              //保证金(仅报价使用)
    DFITCDateType                orderTime;           //委托时间(仅报价使用)
    DFITCOrderAnswerStatusType   orderStatus;         //委托状态
    DFITCCustomCategoryType      customCategory;      //自定义类别
    DFITCInstrumentIDType        instrumentID;        //合约代码
    DFITCAccountIDType           accountID;           //资金账号
    DFITCQuoteIDType             quoteID;             //询价编号
    DFITCSessionIDType           sessionID;           //会话 ID
    DFITCClientIDType            clientID;            //交易编码
};
```

6.2.32. OnRtnQuoteCancel 方法

做市商撤单回报，当用户发出做市商撤单请求指令后该方法会被调用

函数原型：

```
void OnRtnQuoteCancel(struct DFITCQuoteCanceledRtnField * pRtnQuoteCanceledData)
```

参数：

pRtnQuoteCanceledData：返回做市商撤单回报，该结构如下：

```
struct DFITCQuoteCanceledRtnField
{
    DFITCExchangeIDType        exchangeID;          //交易所
    DFITCClientIDType          clientID;            //交易编码
    DFITCOrderSysIDType        orderSysID;          //报单编号
    DFITCInstrumentIDType      instrumentID;        //合约代码
    DFITCLocalOrderIDType      localOrderID;        //本地委托号
    DFITCSeatCodeType          seatCode;            //席位代码
    DFITCOpenCloseTypeType     bOpenCloseType;      //开平标志（买）
```

```
    DFITCOpenCloseTypeType      sOpenCloseType;      //开平标志（卖）
    DFITCSpeculatorType         speculator;          //投资类别
    DFITCSPDOrderIDType         spdOrderID;          //柜台委托号
    DFITCAccountIDType          accountID;           //资金账号
    DFITCEntrusTellerType       entrusTeller;        //委托柜员
    DFITCOrderAnswerStatusType orderStatus;          //委托状态
    DFITCAmountType             cancelAmount;        //撤单数量
    DFITCPriceType              fee;                 //解冻手续费
    DFITCPriceType              margin;              //解冻保证金
    DFITCSessionIDType          sessionID;           //会话 ID
    DFITCBuySellTypeType        buySellType;         //买卖标志
    DFITCQuoteIDType            quoteID;             //询价编号
    DFITCDateType               canceledTime;        //撤单时间
    DFITCCustomCategoryType     customCategory;      //自定义类别
};
```

6.2.33. OnRspCancelAllOrder 方法

做市商批量撤单响应，当用户发出做市商批量撤单请求指令后该方法会被调用

函数原型：

```
void OnRspCancelAllOrder(struct DFITCCancelAllOrderRspField*
pRspCancelAllOrderData, struct DFITCErrorRtnField * pErrorInfo)
```

参数：

pRspCancelAllOrderData：返回做市商批量撤单响应，该结构如下：

```
struct DFITCCancelAllOrderRspField
{
    DFITCRequestIDType              lRequestID;          //请求 ID
    DFITCAccountIDType              accountID;           //资金账号
    DFITCOrderAnswerStatusType      orderStatus;         //委托状态
};
```

6.2.34. OnRspForQuote 方法

询价响应，用户调用询价请求后该接口会被调用

函数原型：

```
void OnRspForQuote(struct DFITCForQuoteRspField * pRspForQuoteData,struct
DFITCErrorRtnField * pErrorInfo)
```

参数：

pRspForQuoteData：返回用户询价响应，该结构如下：

```
struct DFITCForQuoteRspField
{
```

```
    DFITCRequestIDType              lRequestID;         //请求 ID
    DFITCSPDOrderIDType             spdOrderID;         //柜台委托号
    DFITCDateType                   commTime;           //委托时间
};
```

6.2.35. OnRtnForQuote 方法

询价回报，当交易所收到请求后该接口会调用

函数原型：

```
void OnRtnForQuote(struct DFITCForQuoteRtnField * pRtnForQuoteData)
```

参数：

pRtnForQuoteData：返回用户询价回报，该结构如下：

```
struct DFITCForQuoteRtnField
{
    DFITCSPDOrderIDType         spdOrderID;             //柜台委托号
    DFITCSessionIDType          sessionID;              //会话 ID
    DFITCInstrumentIDType           instrumentID;           //合约代码
    DFITCExchangeIDType         exchangeID;             //交易所
    DFITCAccountIDType          accountID;              //资金账号
    DFITCOrderAnswerStatusType    orderStatus;              //委托状态
};
```

6.2.36. OnRspQryQuoteOrderInfo 方法

查询应/报价响应，当用户发出查询应/报价委托的请求时该接口会调用

函数原型：

```
void OnRspQryQuoteOrderInfo(struct DFITCQuoteOrderRtnField * pRtnQuoteOrderData,
struct DFITCErrorRtnField * pErrorInfo, bool bIsLast)
```

参数：

pRtnQuoteOrderData：返回用户应/报价信息，该结构如下：

```
struct DFITCQuoteOrderRtnField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCOrderAnswerStatusType  orderStatus;        //委托状态
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCPriceType              margin;             //保证金
    DFITCPriceType              fee;                //手续费
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCAccountIDType          accountID;          //客户号
    DFITCDateType               commTime;           //委托时间
    DFITCDateType               submitTime;         //申报时间
```

```
    DFITCExchangeIDType         exchangeID;         //交易所 ID
    DFITCAmountType             bOrderAmount;       //委托数量（买）
    DFITCAmountType             bMatchedAmount;     //成交数量（买）
    DFITCAmountType             bCancelAmount;      //撤单数量（买）
    DFITCPriceType              bInsertPrice;       //委托价格（买）
    DFITCPriceType              bMatchedPrice;      //成交价格（买）
    DFITCOpenCloseTypeType      bOpenCloseType;     //开平标志（买）
    DFITCAmountType             sOrderAmount;       //委托数量（卖）
    DFITCAmountType             sMatchedAmount;     //成交数量（卖）
    DFITCAmountType             sCancelAmount;      //撤单数量（卖）
    DFITCPriceType              sInsertPrice;       //成交价格（卖）
    DFITCPriceType              sMatchedPrice;      //成交价格（卖）
    DFITCOpenCloseTypeType      sOpenCloseType;     //开平标志（卖）
    DFITCFrontAddrType          operStation;        //操作站点
    DFITCSessionIDType          sessionID;          //会话 ID
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCCustomCategoryType     customCategory;     //自定义类别
};
```

6.2.37. OnRspQryQuoteNotice 方法

询价通知查询响应，当用户发出询价通知查询请求时该接口会调用

函数原型：

```
void OnRspQryQuoteNotice (struct DFITCQryQuoteNoticeRtnField *
pRtnQryQuoteNoticeData, struct DFITCErrorRtnField * pErrorInfo, bool bIsLast)
```

参数：

pRtnQryQuoteNoticeData：返回询价通知信息，该结构如下：

```
struct DFITCQryQuoteNoticeRtnField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCExchangeIDType         exchangeID;         //交易所 ID
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCSourceType             source;             //来源
    DFITCDateType               quoteTime;          //询价时间
};
```

6.2.38. OnRspQryDepthMarketData 方法

查询合约行情响应，当用户发出行情查询请求时该接口会调用

函数原型：

```
void OnRspQryDepthMarketData (struct DFITCDepthMarketDataField *
```

```
pDepthMarketData, struct DFITCErrorRtnField * pErrorInfo, bool bIsLast)
```

参数：

pDepthMarketData：返回合约行情信息，该结构如下：

```
struct APISTRUCT DFITCDepthMarketDataField
{
    DFITCDateType               tradingDay;          //交易日
    DFITCInstrumentIDType       instrumentID;        //合约代码
    DFITCExchangeIDType         exchangeID;          //交易所代码
    DFITCInstrumentIDType       exchangeInstID;      //合约在交易所的代码
    DFITCPriceType              lastPrice;           //最新价
    DFITCPriceType              preSettlementPrice;  //上次结算价
    DFITCPriceType              preClosePrice;       //昨收盘
    DFITCAmountType             preOpenInterest;     //昨持仓量
    DFITCPriceType              openPrice;           //今开盘
    DFITCPriceType              highestPrice;        //最高价
    DFITCPriceType              lowestPrice;         //最低价
    DFITCAmountType             Volume;              //成交数量
    DFITCPriceType              turnover;            //成交金额
    DFITCAmountType             openInterest;        //持仓量
    DFITCPriceType              closePrice;          //今收盘
    DFITCPriceType              settlementPrice;     //本次结算价
    DFITCPriceType              upperLimitPrice;     //涨停板价
    DFITCPriceType              lowerLimitPrice;     //跌停板价
    DFITCDeltaType              preDelta;            //昨虚实度
    DFITCDeltaType              currDelta;           //今虚实度
    DFITCDateType               UpdateTime;          //最后修改时间
    DFITCMilliSecType           UpdateMillisec;      //最后修改毫秒
    DFITCPriceType              BidPrice1;           //申买价一
    DFITCVolumeType             BidVolume1;          //申买量一
    DFITCPriceType              AskPrice1;           //申卖价一
    DFITCVolumeType             AskVolume1;          //申卖量一
    DFITCPriceType              BidPrice2;           //申买价二
    DFITCVolumeType             BidVolume2;          //申买量二
    DFITCPriceType              AskPrice2;           //申卖价二
    DFITCVolumeType             AskVolume2;          //申卖量二
    DFITCPriceType              BidPrice3;           //申买价三
    DFITCVolumeType             BidVolume3;          //申买量三
    DFITCPriceType              AskPrice3;           //申卖价三
    DFITCVolumeType             AskVolume3;          //申卖量三
    DFITCPriceType              BidPrice4;           //申买价四
    DFITCVolumeType             BidVolume4;          //申买量四
    DFITCPriceType              AskPrice4;           //申卖价四
    DFITCVolumeType             AskVolume4;          //申卖量四
```

```
    DFITCPriceType              BidPrice5;          //申买价五
    DFITCVolumeType             BidVolume5;         //申买量五
    DFITCPriceType              AskPrice5;          //申卖价五
    DFITCVolumeType             AskVolume5;         //申卖量五
    DFITCPriceType              AveragePrice;       //当日均价
    DFITCDateType               XSpeedTime;         //柜台系统时间
};
```

6.2.39. OnRspQryForQuote 方法

询价委托查询响应，当用户发出询价委托查询请求时该接口会调用

函数原型：

```
void OnRspQryForQuote(struct DFITCQryForQuoteRtnField * pRtnQryForQuoteData,
struct DFITCErrorRtnField * pErrorInfo, bool bIsLast)
```

参数：

pRtnQryForQuoteData：返回询价委托信息，该结构如下：

```
struct DFITCQryForQuoteRtnField
{
    DFITCRequestIDType              lRequestID;         //请求 ID
    DFITCAccountIDType              accountID;          //资金账号
    DFITCSPDOrderIDType             spdOrderID;         //柜台委托号
    DFITCInstrumentIDType           instrumentID;       //合约代码
    DFITCExchangeIDType             exchangeID;         //交易所
    DFITCDateType                   SuspendTime;        //挂起时间
    DFITCOrderAnswerStatusType      orderStatus;        //委托状态
};
```

6.2.40. OnRtnQuoteMatchedInfo 方法

做市商成交回报，收到做市商成交回报时该接口会调用

函数原型：

```
void OnRtnQuoteMatchedInfo(struct DFITCQuoteMatchRtnField * pRtnQuoteMatched);
```

参数：

pRtnQuoteMatched：返回做市商成交回报，该结构如下：

```
struct DFITCQuoteMatchRtnField
{
    DFITCExchangeIDType         exchangeID;     //交易所 ID
    DFITCClientIDType           clientID;       //交易编码
    DFITCInstrumentIDType       instrumentID;   //合约代码
    DFITCSeatCodeType           seatCode;       //席位代码
```

```
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCOpenCloseTypeType      openCloseType;      //开平标志
    DFITCSpeculatorType         speculator;         //投资类别
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCOrderSysIDType         OrderSysID;         //报单编号(交易所报单编号)
    DFITCMatchIDType            matchID;            //成交编号
    DFITCAmountType             matchedAmount;      //成交数量
    DFITCPriceType              matchedPrice;       //成交价格
    DFITCAccountIDType          accountID;          //资金账号
    DFITCPriceType              turnover;           //成交金额
    DFITCEntrusTellerType       entrusTeller;       //委托柜员
    DFITCDateType               matchedTime;        //成交时间
    DFITCFeeType                fee;                //手续费
    DFITCPriceType              insertPrice;        //委托价格
    DFITCAmountType             orderAmount;        //委托数量
    DFITCOrderAnswerStatusType  orderStatus;        //申报结果
    DFITCPriceType              margin;             //开仓为保证金,
                                                    //平仓为解冻保证金
    DFITCBuySellTypeType        buySellType;        //买卖
    DFITCAmountType             closeTodayAmount;   //平今数量
    DFITCPriceType              closePrice;         //平仓金额
    DFITCPriceType              closeTodayPrice;    //平今金额
    DFITCAdjustmentInfoType     adjustmentInfo;     //组合或对锁的保证金调整
                                                    //信息,格式:[合约代码,买卖
                                                    //标志,投资类别,调整金额;]
    DFITCPriceType              frozenCapita;       //成交解冻委托冻结的资金
    DFITCProfitLossType         dateCloseProfitLoss;//盯市平仓盈亏
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCSessionIDType          sessionID;          //会话标识
    DFITCLargeMarginDirectType  largeMarginDirect; //大边保证金方向
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCCustomCategoryType     customCategory;     //自定义类别
};
```

6.2.41. OnRtnForQuoteRsp 方法

做市商询价通知,无需订阅,做市商客户自动收到询价通知.

函数原型：

```
void OnRtnForQuoteRsp(struct DFITCQuoteSubscribeRtnField * pForQuoteRspData);
```

参数：

pForQuoteRspData：返回价通知回报，该结构如下：

```
struct DFITCQuoteSubscribeRtnField
```

```
{
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCExchangeIDType         exchangeID;         //交易所
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCSourceType             source;             //来源
    DFITCDateType               quoteTime;          //询价时间
};
```

6.2.42. OnRspQryTransferBank 方法

查询转帐银行响应.

函数原型：

```
void OnRspQryTransferBank(struct DFITCTransferBankRspField * pTransferBank,
struct DFITCErrorRtnField * pErrorInfo, bool bIsLast);
```

参数：

pTransferBank：返回查询转帐银行回报，该结构如下：

```
struct DFITCTransferBankRspField
{
    DFITCAccountIDType          accountID;          //客户号
    DFITCBankIDType             bankID;             //银行代码
    DFITCBankAccountType        bankAccount;        //银行账号
    DFITCCurrencyType           currency;           //币种
    DFITCDateType               registDate;         //登记日期
    DFITCRequestIDType          lRequestID;         //请求 ID
};
```

6.2.43. OnRspQryTransferSerial 方法

查询转帐流水响应.

函数原型：

```
void OnRspQryTransferSerial(struct DFITCTransferSerialRspField * pTransferSerial,
struct DFITCErrorRtnField * pErrorInfo, bool bIsLast);
```

参数：

pTransferSerial：返回查询转帐流水回报，该结构如下：

```
struct DFITCTransferSerialRspField
{
    DFITCAccountIDType          accountID;              //资金账号
    DFITCBankIDType             bankID;                 //银行代码
    DFITCBankAccountType        bankAccount;            //银行账号
    DFITCCurrencyType           currency;               //币种代码
    DFITCApplyNumberType        applyNum;               //申请号
```

```
    DFITCTransferType          type;                   //转账业务类别
    DFITCPriceType             tradeAmount;            //转账金额
    DFITCPriceType             curFutAccountFund;      //本次资金余额
    DFITCSerialType            bankSerialNum;          //银行流水号
    DFITCTimeType              reqTransferTime;        //发起转账时间
    DFITCTimeType              dealTransferTime;       //转账成功时间
    DFITCProcResultType        procResult;             //转账处理结果
    DFITCRequestIDType         lRequestID;             //请求 ID
};
```

6.2.44. OnRspFromBankToFutureByFuture 方法

期货发起银行资金转期货应答.

函数原型：

```
void OnRspFromBankToFutureByFuture(struct DFITCTransferRspField * pRspTransfer,
struct DFITCErrorRtnField * pErrorInfo);
```

参数：

pRspTransfer：返回期货发起银行资金转期货应答，该结构如下：

```
struct DFITCTransferRspField
{
    DFITCBankIDType            bankID;                 //银行代码
    DFITCBankAccountType       bankAccount;            //银行账号
    DFITCAccountIDType         accountID;              //投资者账号
    DFITCPriceType             tradeAmount;            //转账金额
    DFITCApplyNumberType       applyNumber;            //转账申请号
    DFITCRequestIDType         lRequestID;             //请求 ID
};
```

6.2.45. OnRspFromFutureToBankByFuture 方法

期货发起期货资金转银行应答.

函数原型：

```
void OnRspFromFutureToBankByFuture(struct DFITCTransferRspField * pRspTransfer,
struct DFITCErrorRtnField * pErrorInfo);
```

参数：

pRspTransfer：返回期货发起期货资金转银行应答，该结构如下：

```
struct DFITCTransferRspField
{
    DFITCBankIDType            bankID;                 //银行代码
    DFITCBankAccountType       bankAccount;            //银行账号
    DFITCAccountIDType         accountID;              //投资者账号
    DFITCPriceType             tradeAmount;            //转账金额
```

```
    DFITCApplyNumberType        applyNumber;        //转账申请号
    DFITCRequestIDType          lRequestID;         //请求 ID
};
```

6.2.46. OnRtnFromBankToFutureByFuture 方法

期货发起银行资金转期货通知.

函数原型：

```
void OnRtnFromBankToFutureByFuture(DFITCTransferRtnField * pRtnTransfer, struct
DFITCErrorRtnField * pErrorInfo);
```

参数：

pRtnTransfer：返回期货发起银行资金转期货通知，该结构如下：

```
struct DFITCTransferRtnField
{
    DFITCAccountIDType          accountID;          //投资者账号
    DFITCBankIDType             bankID;             //银行代码
    DFITCBankAccountType        bankAccount;        //银行账号
    DFITCTransferType           type;               //转账类别
    DFITCPriceType              tradeAmount;        //转账金额
    DFITCSerialType             bankSerialNum;      //银行流水号
    DFITCApplyNumberType        applyNumber;        //转账申请号
    DFITCSessionIDType          sessionID;          //会话 ID
};
```

6.2.47. OnRtnFromFutureToBankByFuture 方法

期货发起期货资金转银行通知.

函数原型：

```
void OnRtnFromFutureToBankByFuture(DFITCTransferRtnField * pRtnTransfer, struct
DFITCErrorRtnField * pErrorInfo);
```

参数：

pRtnTransfer：返回期货发起期货资金转银行通知，该结构如下：

```
struct DFITCTransferRtnField
{
    DFITCAccountIDType          accountID;          //投资者账号
    DFITCBankIDType             bankID;             //银行代码
    DFITCBankAccountType        bankAccount;        //银行账号
    DFITCTransferType           type;               //转账类别
    DFITCPriceType              tradeAmount;        //转账金额
    DFITCSerialType             bankSerialNum;      //银行流水号
    DFITCApplyNumberType        applyNumber;        //转账申请号
    DFITCSessionIDType          sessionID;          //会话 ID
```

```
};
```

6.2.48. OnRtnRepealFromFutureToBankByBank 方法

银行发起冲正期货转银行通知.

函数原型：

```
void OnRtnRepealFromFutureToBankByBank(DFITCRepealRtnField * pRspRepeal);
```

参数：

pRspRepeal：返回期货发起期货资金转银行冲正通知，该结构如下：

```
struct DFITCRepealRtnField
{
    DFITCAccountIDType          accountID;          //投资者账号
    DFITCBankIDType             bankID;             //银行代码
    DFITCBankAccountType        bankAccount;        //银行账号
    DFITCTransferType           type;               //转账类别
    DFITCPriceType              tradeAmount;        //转账金额
    DFITCSerialType             bankSerialNum;      //银行流水号
    DFITCSerialType             repealSerial;       //被冲正流水号
};
```

6.2.49. OnRspQryExchangeRate 方法

汇率查询响应.

函数原型：

```
void OnRspQryExchangeRate(struct DFITCExchangeRateField *pExchangeRate,
DFITCErrorRtnField *pRspInfo, bool bIsLast);
```

参数：

pExchangeRate：返回汇率查询响应，该结构如下：

```
struct DFITCExchangeRateField
{
    DFITCRequestIDType          lRequestID;             //请求 ID
    DFITCCurrencyType           fromCurrencyID;         //源币种
    DFITCCurrencyUnitType       fromCurrencyUnit;       //源币种单位数量
    DFITCCurrencyType           toCurrencyID;           //目标币种
    DFITCExchangeRateType       exchangeRate;           //汇率
};
```

6.2.50. OnRspQryPricesTrigger 方法

行情触发查询响应，当用户发出行情触发查询指令后,前置返回响应时该方法会被调用.

函数原型：

```
void OnRspQryPricesTrigger(struct DFITCQryPricesTriggerField
*pQryPricesTriggerRspData, struct DFITCErrorRtnField * pErrorInfo, bool bIsLast);
```

参数：

pQryPricesTriggerRspData：返回行情查询信息，该结构如下：

```
struct DFITCQryPricesTriggerField
{
    DFITCAccountIDType              accountID;          //资金账户
    DFITCLocalOrderIDType           localOrderID;       //本地委托号
    DFITCInstrumentIDType           instrumentID;       //合约代码
    DFITCPriceType                  insertPrice;        //委托价格
    DFITCAmountType                 orderAmount;        //委托数量
    DFITCSpeculatorType             speculator;         //投保类型
    DFITCOrderTypeType              orderType;          //报单类型
    DFITCBuySellTypeType            buySellType;        //买卖标志
    DFITCOpenCloseTypeType          openCloseType;      //开平标志
    DFITCRequestIDType              lRequestID;         //请求 ID
    DFITCCompareFlagType            compareFlag;        //比较标志
    DFITCPriceType                  comparePrice;       //触发价格
    DFITCPriceReferenceType         priceReference;     //价格参照
    DFITCBreakDownTimesType         breakDownTimes;     //击穿次数
    DFITCDateType                   validate;           //有效日期
    DFITCDateType                   modifiedtime;       //修改时间
    DFITCDateType                   commTime;           //委托时间
    DFITCDateType                   commdate;           //委托日期
    DFITCSPDOrderIDType             spdOrderID;         //柜台委托号
    DFITCDateType                   canceledTime;       //撤单时间
    DFITCFrozenTypeType             frozentype;         //冻结类型
    DFITCSPDOrderIDType             extSpdOrderID;      //条件单编号
    DFITCExtOrderType               extOrderType;       //条件单类型
    DFITCProfitLossType             frozenMargin;       //冻结资金
    DFITCAmountType                 frozenAmount;       //冻结数量
    DFITCOrderAnswerStatusType      orderStatus;        //委托状态
    DFITCExtOrderTriggerStatusType orderTriggerStatus; //条件单触发状态
    DFITCAmountType                 limitAmount;        //数量限制
    DFITCCompareFlagType            qtyCmpFlag;         //比较标志(数量)
    DFITCTriggerType                triggerType;        //触发类型
    DFITCBreakDownTypeType          breakDownType;      //击穿属性
};
```

6.2.51. OnRspExtInsertOrder 方法

条件单委托报单响应,当用户录入报单后,前置返回响应时该方法会被调用.

函数原型：

```
void OnRspExtInsertOrder(struct DFITCExtOrderRspDataField * pOrderRsp, struct
DFITCErrorRtnField * pErrorInfo);
```

参数：

pOrderRsp：返回用户下单信息，该结构如下：

```
struct DFITCExtOrderRspDataField
{
    DFITCLocalOrderIDType          localOrderID;       //本地委托号
    DFITCSPDOrderIDType            spdOrderID;         //柜台委托号
    DFITCSPDOrderIDType            extSpdOrderID;      //条件单编号
    DFITCOrderAnswerStatusType     orderStatus;        //委托状态
    DFITCRequestIDType             lRequestID;         //请求 ID
    DFITCPriceType                 frozenMargin;       //冻结资金(仅下单使用)
    DFITCExtOrderType              extOrderType;       //条件单类型
    DFITCAccountIDType             accountID;          //资金账户
};
```

6.2.52. OnRspExtCancelOrder 方法

条件单委托撤单响应，当用户撤单后，前置返回响应是该方法会被调用.

函数原型：

```
void OnRspExtCancelOrder(struct DFITCExtOrderRspDataField * pOrderCancelRsp,
struct DFITCErrorRtnField * pErrorInfo);
```

参数：

pOrderCancelRsp：返回撤单响应信息，该结构如下：

```
struct DFITCExtOrderRspDataField
{
    DFITCLocalOrderIDType          localOrderID;       //本地委托号
    DFITCSPDOrderIDType            spdOrderID;         //柜台委托号
    DFITCSPDOrderIDType            extSpdOrderID;      //条件单编号
    DFITCOrderAnswerStatusType     orderStatus;        //委托状态
    DFITCRequestIDType             lRequestID;         //请求 ID
    DFITCPriceType                 frozenMargin;       //冻结资金(仅下单使用)
    DFITCExtOrderType              extOrderType;       //条件单类型
    DFITCAccountIDType             accountID;          //资金账户
};
```

6.2.53. OnRtnPricesTrigger 方法

条件单委托回报，当用户录入报单并被触发后该方法会被调用.

函数原型：

```
void OnRtnPricesTrigger(struct DFITCPricesTriggerRtnField * pOrderRtn);
```

参数：

pOrderRtn：返回用户下单信息，该结构如下：

```
struct DFITCPricesTriggerRtnField
{
    DFITCAccountIDType              accountID;          //资金账户
    DFITCLocalOrderIDType           localOrderID;       //本地委托号
    DFITCInstrumentIDType           instrumentID;       //合约代码
    DFITCPriceType                  insertPrice;        //委托价格
    DFITCAmountType                 orderAmount;        //委托数量
    DFITCSpeculatorType             speculator;         //投保类型
    DFITCExtOrderPriceTypeType      extOrderPriceType;  //条件单报单类型
    DFITCBuySellTypeType            buySellType;        //买卖标志
    DFITCOpenCloseTypeType          openCloseType;      //开平标志
    DFITCCompareFlagType            compareFlag;        //比较标志
    DFITCPriceType                  comparePrice;       //触发价格
    DFITCPriceReferenceType         priceReference;     //价格参照
    DFITCBreakDownTimesType         breakDownTimes;     //击穿次数
    DFITCDateType                   modifiedtime;       //修改时间
    DFITCDateType                   commTime;           //委托时间
    DFITCSPDOrderIDType             spdOrderID;         //柜台委托号
    DFITCDateType                   canceledTime;       //撤单时间
    DFITCFrozenTypeType             frozentype;         //冻结类型
    DFITCSPDOrderIDType             extSpdOrderID;      //条件单编号
    DFITCProfitLossType             frozenMargin;       //冻结资金
    DFITCAmountType                 frozenAmount;       //冻结数量
    DFITCOrderAnswerStatusType      orderStatus;        //委托状态
    DFITCSessionIDType              sessionID;          //会话 ID
    DFITCErrorMsgInfoType           statusMsg;          //状态信息
    DFITCAmountType                 limitAmount;        //数量限制
    DFITCCompareFlagType            qtyCmpFlag;         //比较标志(数量)
    DFITCTriggerType                triggerType;        //触发类型
    DFITCBreakDownTypeType          breakDownType;      //击穿属性
    DFITCExtOrderTriggerStatusType orderTriggerStatus;  //条件单触发状态
};
```

6.2.54. OnErrRtnCancelOrder 方法

撤单操作交易所错误回报.

函数原型：

```
void OnErrRtnCancelOrder(struct DFITCOrderCancelErrField * pOrderCancel);
```

参数：

pOrderCancel：返回订单信息，该结构如下：

```
struct DFITCOrderCancelErrField
```

```
{
    DFITCAccountIDType          accountID;          //资金账号
    DFITCSessionIDType          sessionID;          //会话 ID
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCPriceType              insertPrice;        //委托价
    DFITCAmountType             orderAmount;        //委托数量
    DFITCBuySellTypeType        buySellType;        //买卖
    DFITCOpenCloseTypeType      openCloseType;      //开平
    DFITCSpeculatorType         speculator;         //投资类别
    DFITCInsertType             insertType;         //自动单类别
    DFITCOrderTypeType          orderType;          //报单类型
    DFITCOrderPropertyType      orderProperty;      //订单属性
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCAmountType             minMatchAmount;     //最小成交量
    DFITCCustomCategoryType     customCategory;     //自定义类别
    DFITCPriceType              profitLossPrice;    //止盈止损价格
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCOrderAnswerStatusType  orderStatus;        //委托状态
    DFITCExchangeIDType         exchangeID;         //交易所
    DFITCOrderSysIDType         OrderSysID;         //报单编号
    DFITCErrorMsgInfoType       errorMsg;           //错误信息
    DFITCReservedType           reservedType;       //预留字段
};
```

6.2.55. OnErrRtnQuoteCancel 方法

做市商撤单操作交易所错误回报.

函数原型：

```
void OnErrRtnQuoteCancel(struct DFITCQuoteCancelErrField * pQuoteCancel);
```

参数：

pQuoteCancel：返回报价信息，该结构如下：

```
struct DFITCQuoteCancelErrField
{
    DFITCAccountIDType          accountID;          //资金账号
    DFITCSessionIDType          sessionID;          //会话 ID
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCOrderAnswerStatusType  orderStatus;        //委托状态
    DFITCCustomCategoryType     customCategory;     //自定义类别
```

```
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCExchangeIDType         exchangeID;         //交易所
    DFITCOrderSysIDType         OrderSysID;         //报单编号
    DFITCErrorMsgInfoType       errorMsg;           //错误信息
    DFITCReservedType           reservedType;       //预留字段
};
```

6.2.56. OnRspQryBillConfirm 方法

查询账单确认响应.

函数原型：

```
void OnRspQryBillConfirm(struct DFITCQryBillConfirmRspField * pBillConfirmRsp,
struct DFITCErrorRtnField * pErrorInfo);
```

参数：

pBillConfirmRsp：返回结算账单确认状态，该结构如下：

```
struct DFITCQryBillConfirmRspField
{
    DFITCRequestIDType          lRequestID;             //请求 ID
    DFITCAccountIDType          accountID;              //资金账户
    DFITCDateType               confirmDate;            //确认日期
    DFITCDateType               confirmTime;            //确认时间
};
```

6.2.57. OnRspQryTradingNotice 方法

交易通知查询响应，当用户发出查询交易通知接口后，前置返回响应时该方法会被调用.

函数原型：

```
void OnRspQryTradingNotice(struct DFITCTradingNoticeField * pTradingNotice,
struct DFITCErrorRtnField * pErrorInfo, bool bIsLast);
```

参数：

pTradingNotice：返回结算账单确认状态，该结构如下：

```
struct DFITCTradingNoticeField
{
    DFITCAccountIDType          accountID;          //资金帐号 ID
    DFITCTimeType               sendTime;           //发送时间
    DFITCContentType            fieldContent;       //消息正文
    DFITCNoticeType             noticeType;         //消息类型
    DFITCRequestIDType          lRequestID;         //请求 ID
};
```

7. DFITCMdApi 使用参考手册

7.1 DFITCMdApi 接口

7.1.1. CreateDFITCMdApi 方法

行情 API 命名空间为：“DFITCXSPEEDMDAPI ”，使用时需要引入命名空间，如

```
using namespace DFITCXSPEEDMDAPI;
```

产生一个行情 api 实例。

函数原型：

```
static DFITCMdApi *CreateDFITCMdApi();
```

7.1.2 Init 方法

初始化工作：注册回调函数集，连接行情前置。

函数原型：

```
virtual int Init( char *pszSvrAddr, DFITCMdSpi *pSpi);
```

参数：

pszSvrAddr:行情前置地址，格式： protocol://ipaddress:port，其中：

protocol ：tcp 或者 udp 接收行情数据，如果是 udp 接收行情数据，udp 的端口将由 API 自行确定。

ipaddress：行情前置的地址。

port：行情前置端口。

pSpi：回调函数指针。

7.1.3 Release 方法

销毁一个行情 api 实例。

函数原型：

```
void Release();
```

不能使用 delete 来销毁对象，调用该函数时，会断开 API 与前置的连接，并退出线程和释放相应的资源。

7.1.4 ReqUserLogin 方法

用户发出请求登录。

函数原型：

```
int ReqUserLogin(struct DFITCUserLoginField *pUserLoginData);
```

参数：

pUserLoginData：指向用户登录请求结构的地址。用户请求登录结构：

```
struct DFITCUserLoginField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金账户 ID
    DFITCPasswdType             passwd;              //密码
    DFITCCompanyIDType          companyID;           //厂商 ID
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

7.1.5. ReqUserLogout 方法

用户发出退出请求。

函数原型：

```
int ReqUserLogout( struct DFITCUserLogoutField *pUserLogoutData );
```

参数：

pUserLogoutData:指向用户退出请求结构的地址。用户请求退出结构：

```
struct DFITCUserLogoutField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金帐号 ID
    DFITCSessionIDType          sessionID;           //会话 ID
};
```

返回值：

0： 请求发送成功。

-1：请求发送失败。

-2：检测异常。

7.1.6. SubscribeMarketData 方法

该方法发出订阅某个或某些合约的行情请求。

函数原型：

```
virtual int SubscribeMarketData(char *ppInstrumentID[], int nCount,int nRequestID)
```

参数：

ppInstrument:指针数组，每个指针指向一个合约。

nCount:合约个数.

返回值：

0： 请求发送成功。

-1：请求发送失败如果合约名为”*”，

说明：

如果合约名为”*”，则默认订阅所有合约的行情。也可以按交易所名字订阅，如合约名为”DCE”，则订阅大商所所有合约的行情，同理，”SHFE”则订阅上期交易所所有合约行情。

7.1.7. UnSubscribeMarketData 方法

该方法发出退订某个/某些合约行情请求。

函数原型：

```
virtual int UnSubscribeMarketData(char *ppInstrumentID[], int nCount , int nRequestID)
```

参数：

同上。使用方法及也同上。

7.1.8. SubscribeForQuoteRsp 方法

该方法发出订阅某个或某些合约的询价通知请求。

函数原型：

```
virtual int SubscribeForQuoteRsp(char * ppInstrumentID[], int nCount, int nRequestID)
```

参数：

ppInstrument:指针数组，每个指针指向一个合约。

nCount:合约个数.

返回值：

0： 请求发送成功。

-1：请求发送失败如果合约名为”*”，

说明：

如果合约名为”*”，则默认订阅所有合约的询价通知。也可以按交易所名字订阅，如

合约名为”DCE”，则订阅大商所所有合约的询价通知，同理，”SHFE”则订阅上期交易所所有合约询价通知。

7.1.9. UnSubscribeForQuoteRsp 方法

该方法发出退订某个/某些合约询价通知请求。。

函数原型：

```
virtual int UnSubscribeForQuoteRsp(char * ppInstrumentID[], int nCount, int nRequestID)
```

参数：

同上。使用方法及也同上。

7.1.10. ReqTradingDay 方法

交易日查询请求.(预留方法)

函数原型：

```
int ReqTradingDay(struct DFITCTradingDayField * pTradingDay);
```

参数：

pTradingDay：查询请求号.

```
struct APISTRUCT DFITCTradingDayField
{
    DFITCRequestIDType            lRequestID;            //请求 ID
};
```

返回值：

0：请求发送成功;
-1：请求发送失败;
-2：检测异常;

7.2DFITCMdSpi 接口

7.2.1. OnRspUserLogin 方法

当用户发出登录请求后，前置机返回响应时此方法会被调用，通知用户登录是否成功。

函数原型：

```
Void OnRspUserLogin(struct DFITCUserLoginInfoRtnField *pUserLoginInfoRtn,struct DFITCErrorRtnField *pErrorInfo )
```

参数：

pUserLoginInfoRtn：返回用户登录信息结构地址：

```
struct DFITCUserLoginInfoRtnField
{
    DFITCRequestIDType          lRequestID;          //请求 ID
    DFITCAccountIDType          accountID;           //资金帐号 ID
    DFITCAccountLoginResultType loginResult;         //登录结果
    DFITCLocalOrderIDType       initLocalOrderID;    //初始本地委托号
    DFITCSessionIDType          sessionID;           //sessionID(会话 ID)
    DFITCErrorIDType            nErrorID;            //错误 ID
    DFITCErrorMsgInfoType       errorMsg;            //错误信息
    DFITCDateType               tradingDay;          //交易日 yyyy.mm.dd
    DFITCTimeType               DCEtime;             //大商所时间
    DFITCTimeType               SHFETime;            //上期所时间
    DFITCTimeType               CFFEXTime;           //中金所时间
    DFITCTimeType               CZCETime;            //郑商所时间
    DFITCTimeType               INETime;             //上能所时间
};
```

在行情 API 登录返回的信息中，不包含 localOrderID, sessionID 和各交易所的时间

7.2.2. OnRspUserLogout 方法

当用户发出退出请求后，前置机返回响应此方法会被调用，通知用户退出状态。

函数原型：

```
void OnRspUserLogout( struct DFITCUserLogoutInfoRtnField *pUserLogoutInfoRtn,struct DFITCErrorRtnField Data *pErrorInfo );
```

参数：

pUserLogoutInfoRtn:返回用户退出信息结构地址：

```
struct DFITCUserLogoutInfoRtnField
{
    DFITCRequestIDType              lRequestID;          //请求 ID
    DFITCAccountIDType              accountID;           //资金账户 ID
    DFITCAccountLogoutResultType    logoutResult;        //退出结果
    DFITCErrorIDType                nErrorID;            //错误 ID
    DFITCErrorMsgInfoType           errorMsg;            //错误信息
};
```

7.2.3. OnRspError 方法

错误回报

函数原型：

```
void OnRspError(struct DFITCErrorRtnField  *pRspInfo)
```

参数

pRspInfo:返回响应用户信息的地址：

```
struct DFITCErrorRtnField
{
    DFITCRequestIDType          requestID;          //请求 ID
    DFITCSessionIDType          sessionID;          //会话标识
    DFITCAccountIDType          accountID;          //资金账号
    DFITCErrorIDType            nErrorID;           //错误 ID
    DFITCSPDOrderIDType         spdOrderID;         //柜台委托号
    DFITCLocalOrderIDType       localOrderID;       //本地委托号
    DFITCErrorMsgInfoType       errorMsg;           //错误信息
    DFITCInstrumentType         instrumentID;       //合约代码
 };
```

7.2.4. OnRspSubMarketData 方法

行情订阅应答，当用户发出行情订阅该方法会被调用。

函数原型：

```
void          OnRspSubMarketData(struct          DFITCSpecificInstrumentField
*pSpecificInstrument, struct DFITCErrorRtnField  *pRspInfo)
```

参数：

pSpecificInstrument:指向合约响应结构。

pRspInfo:错误信息，如果发生错误，该结构含有错误信息。

```
struct DFITCSpecificInstrumentField
{
    DFITCRequestIDType              lRequestID;             //请求 ID
    DFITCAccountIDType              accountID;              //资金账户 ID
    DFITCInstrumentIDType           InstrumentID;           //合约代码
    DFITCExchangeIDType             exchangeID;             //交易所 ID
    DFITCInstrumentTypeType         instrumentType;         //合约类型
    DFITCSpeculatorType             speculator;             //投保类型
};
```

在该订阅响应信息中，由于订阅合约行情时就没有输入客户号 accountID，则该订阅返回信息中也没有包含，只指明了订阅该合约的行情是成功还是失败。

7.2.5. OnRspUnSubMarketData 方法

取消行情订阅应答，当用户发出退订请求后该方法会被调用。

函数原型：

```
void          OnRspUnSubMarketData(struct          DFITCSpecificInstrumentField
*pSpecificInstrument, struct DFITCErrorRtnField  *pRspInfo)
```

参数：

同上.

7.2.6. OnMarketData 方法

行情函数，当订阅行情成功且有行情返回时，该方法会被调用。

函数原型：

```
void OnMarketData (struct DFITCDepthMarketDataField *pMarketDataField)
```

参数：

pMarketDataField:指向行情信息结构.

```
struct DFITCDepthMarketDataField
{
    DFITCDateType                   tradingDay;          //交易日
    DFITCInstrumentIDType           instrumentID;        //合约代码
    DFITCExchangeIDType             exchangeID;          //交易所代码
    DFITCInstrumentIDType           exchangeInstID;      //合约在交易所代码
    DFITCPriceType                  lastPrice;           //最新价
    DFITCPriceType                  preSettlementPrice; //上次结算价
    DFITCPriceType                  preClosePrice;       //昨收盘
    DFITCAmountType                 preOpenInterest;     //昨持仓量
    DFITCPriceType                  openPrice;           //今开盘
    DFITCPriceType                  highestPrice;        //最高价
    DFITCPriceType                  lowestPrice;         //最低价
    DFITCAmountType                 Volume;              //数量
    DFITCPriceType                  turnover;            //成交金额
    DFITCAmountType                 openInterest;        //持仓量
    DFITCPriceType                  closePrice;          //今收盘
    DFITCPriceType                  settlementPrice;     //本次结算价
    DFITCPriceType                  upperLimitPrice;     //涨停板价
    DFITCPriceType                  lowerLimitPrice;     //跌停板价
    DFITCDeltaType                  preDelta;            //昨虚实度
    DFITCDeltaType                  currDelta;           //今虚实度
    DFITCDateType                   UpdateTime;          //最后修改时间
    DFITCMilliSecType               UpdateMillisec;      //最后修改毫秒
    DFITCPriceType                  BidPrice1;           //申买价一
    DFITCVolumeType                 BidVolume1;          //申买量一
    DFITCPriceType                  AskPrice1;           //申卖价一
    DFITCVolumeType                 AskVolume1;          //申卖量一
```

```
    DFITCPriceType              BidPrice2;          //申买价二
    DFITCVolumeType             BidVolume2;         //申买量二
    DFITCPriceType              AskPrice2;          //申卖价二
    DFITCVolumeType             AskVolume2;         //申卖量二
    DFITCPriceType              BidPrice3;          //申买价三
    DFITCVolumeType             BidVolume3;         //申买量三
    DFITCPriceType              AskPrice3;          //申卖价三
    DFITCVolumeType             AskVolume3;         //申卖量三
    DFITCPriceType              BidPrice4;          //申买价四
    DFITCVolumeType             BidVolume4;         //申买量四
    DFITCPriceType              AskPrice4;          //申卖价四
    DFITCVolumeType             AskVolume4;         //申卖量四
    DFITCPriceType              BidPrice5;          //申买价五
    DFITCVolumeType             BidVolume5;         //申买量五
    DFITCPriceType              AskPrice5;          //申卖价五
    DFITCVolumeType             AskVolume5;         //申卖量五
    DFITCPriceType              AveragePrice;       //当日均价
    DFITCDateType               XSpeedTime;         //柜台系统时间
};
```

每次订阅合约行情时，前置至少返回一条该合约的最新快照行情，但该行情的时间戳默认为”99:99:99.999”，也可联系期货公司改成你特意指定的时间戳格式，如果不关心，可忽略该条行情，后期交易所有合约行情更新时，该函数将被触发。

7.2.7. OnRspSubForQuoteRsp 方法

询价通知订阅应答，当用户发出询价通知订阅该方法会被调用。

函数原型：

```
void OnRspSubForQuoteRsp(struct DFITCSpecificInstrumentField * pSpecificInstrument, struct DFITCErrorRtnField * pRspInfo)
```

参数：

pSpecificInstrument:指向合约响应结构。

pRspInfo:错误信息，如果发生错误，该结构含有错误信息。

```
struct APISTRUCT DFITCSpecificInstrumentField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCAccountIDType          accountID;          //资金账户 ID
    DFITCInstrumentIDType       InstrumentID;       //合约代码
    DFITCExchangeIDType         exchangeID;         //交易所 ID
    DFITCInstrumentTypeType     instrumentType;     //合约类型
    DFITCSpeculatorType         speculator;         //投保类型
};
```

7.2.8. OnRspUnSubForQuoteRsp 方法

取消询价通知订阅应答，当用户发出退订请求后该方法会被调用。

函数原型：

```
void OnRspUnSubForQuoteRsp(struct DFITCSpecificInstrumentField * pSpecificInstrument, struct DFITCErrorRtnField * pRspInfo)
```

参数：

同上.

7.2.9. OnRtnForQuoteRsp 方法

询价通知函数，当订阅询价通知成功且有询价通知返回时，该方法会被调用。

函数原型：

```
void OnRtnForQuoteRsp(struct DFITCQuoteSubscribeRtnField * pForQuoteField)
```

参数：

pForQuoteField:指向询价通知信息结构.

```
struct APISTRUCT DFITCQuoteSubscribeRtnField
{
    DFITCQuoteIDType            quoteID;            //询价编号
    DFITCExchangeIDType         exchangeID;         //交易所
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCSourceType             source;             //来源
    DFITCDateType               quoteTime;          //询价时间
};
```

7.2.10. OnCustomMarketData 方法(暂不支持)

行情函数，当订阅组合行情成功且有行情返回时，该方法会被调用。

函数原型：

```
void OnCustomMarketData(struct DFITCCustomMarketDataField * pMarketDataField)
```

参数：

pMarketDataField:指向行情信息结构.

```
struct DFITCCustomMarketDataField
{
    DFITCInstrumentIDType       instrumentID;       //合约代码
    DFITCExchangeIDType         exchangeID;         //交易所
    DFITCVolumeType             bidVolume1;         //买一量
```

```
    DFITCPriceType              bidPrice1;          //买一价(挂价价差)
    DFITCVolumeType             askVolume1;         //卖一量
    DFITCPriceType              askPrice1;          //卖一价(对价价差)
    DFITCPriceType              lastPrice;          //最新价价差
};
```

7.2.11. OnFrontConnected 方法

该方法是在Api和前置机建立连接后被调用,该调用仅仅是说明tcp连接已经建立成功。用户需要自行登录才能进行后续的业务操作。登录失败则此方法不会被调用。

函数原型：

```
void OnFrontConnected();
```

7.2.12. OnFrontDisconnected 方法

该方法是在 Api 和前置机连接断开后被调用。

函数原型：

```
void OnFrontDisconnected(int nReason);
```

7.2.13. OnRspTradingDay 方法

交易日确认响应,用于接收交易日信息.

函数原型：

```
void OnRspTradingDay(struct DFITCTradingDayRtnField * pTradingDayRtnData);
```

参数：

pTradingDayRtnData: 返回交易日请求确认响应,结构体如下:

```
struct APISTRUCT DFITCTradingDayRtnField
{
    DFITCRequestIDType      lRequestID;         //请求 ID
    DFITCDateType           date;               //交易日 yyyy.mm.dd
};
```

7.3使用行情 API 直接接收广播/组播行情

为提高行情速度，X-Speed API 支持直接从报盘或其它行情源接收 udp 广播/组播行情。使用方式的不同点主要是在 Init 时指定的协议和 IP 及端口不同。示例：

原：char host[] = "tcp://172.21.200.53:10915"; //tcp 连接行情前置的 ip 和端口

新：char host[] = "udpb://172.16.16.73:7813"; //udpb 方式，再接广播地址端口

```
// 正常调用接口来做 API  的初始化
DFITCMdApi* pMdApi = DFITCMdApi::CreateDFITCMdApi();
CMdSpi spi(pMdApi);
int iret = pMdApi->Init(host, &spi);
```

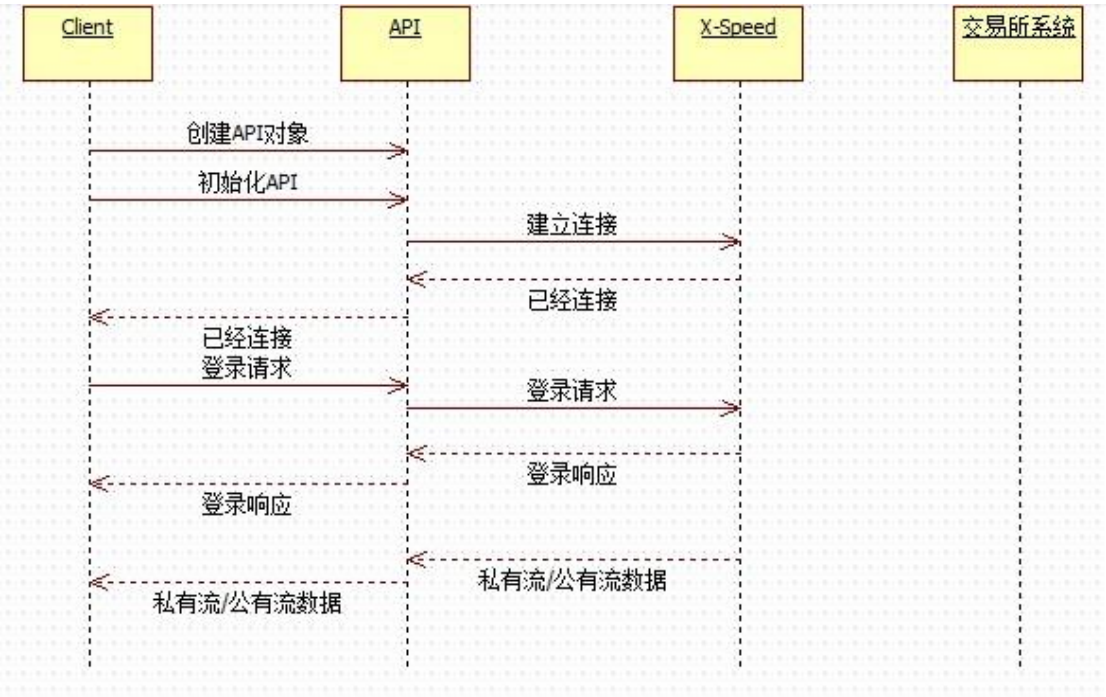
当 iret 返回值为 0 时，则 Init 成功，然后就可以登录，再正常订阅/退订行情了。

注意：

1. 当发现 API 收不到行情时，首先检查一下当前机器是否在此端口上有新行情过来
2. 如果想用 API 接收广播行情(直接由报盘广播出来),则 host 地址写成“udpb://ip:port”。首先,IP 不能是 127.0.0.1。**然后在 Windows 上,ip 指定为本机 IP 地址即可,但在 Linux 上，ip 需要指定为广播 IP 地址才行，比如”** udpb://172.16.16.255**”** 。端口可询问期货公司的技术人员。Init 成功后，也可正常订阅行情。使用此类方式的优势在于行情速度更快，因为行情不经过行情前置转发，从行情源通过 udp 直接到达了 API。
3. 在某些多网卡的机器上，可能出现收不到组播行情的情况，此时需要做一下特殊处理，有以下 3 种方案 （任选一种方案即可，适用于 Windows）：
   a) 禁用掉其它无用的虚拟网卡（我机器的 VMWare 不禁用就收不到）
   b) 将上面的 host 地址写成此格式：udpm://组播 IP 地址,本机 IP 地址:组播端口，如“udpm://224.1.2.3,172.16.16.73:10000”，其中有两个 IP 地址，以逗号分隔，前一个为组播地址，后一个 172.16.16.73 是我的本机 IP 地址，最后是组播端口，这样后，行情 API 会自动绑定到该 IP 地址和端口上收行情了
   c) 采用配置文件方式，在程序的当前目录下增加”dfitc_api.conf”文本文件，内容为一行内容：“LocalIp=172.16.16.73”，即在文件中指定本机 IP 地址，示例配置文件在“windows api” 这个目录下边可以找到。

8. 特别说明

8.1 初始化过程



首先，终端程序创建 DFITCTraderApi 对象(CreateDFITCTraderApi)，创建用户定义的事件处理对象(DFITCTraderSpi)，注册前置机（可以注册多个前置机，DFITCTraderApi 自动选择 1 个可用前置机登入）。完成上述 DFITCTraderApi 设置后，初始化 DFITCTraderApi (Init)，连接 X-Speed。
X-Speed 接受 DFITCTraderApi 的连接后，终端程序就会收到(OnFrontConnected)的通知。终端程序收到连线通知后，发出登入请求(ReqUserLogin)，登入成功后，会收到登入响应(OnRspUserLogin)

用户登入时需要输入：

```
///请求 ID
long lRequestID;
///资金账户 ID
DFITCAccountIDType accountID;
///密码
DFITCPasswdType passwd;
///登录结果
IDDFITCAccountLoginResultType loginResult;
///初始本地委托号
DFITCLocalOrderIDType initLocalOrderID;
///会话 ID
DFITCSessionIDType sessionID;
```

```
///错误 ID
DFITCErrorIDType  nErrorID;
///错误信息
DFITCErrorMsgInfoType errorMsg;
///大商所时间
DFITCTimeType DCEtime;
///上期所时间
DFITCTimeType SHFETime;
///中金所时间
DFITCTimeType CFFEXTime;
///郑商所时间
DFITCTimeType CZCETime;
```

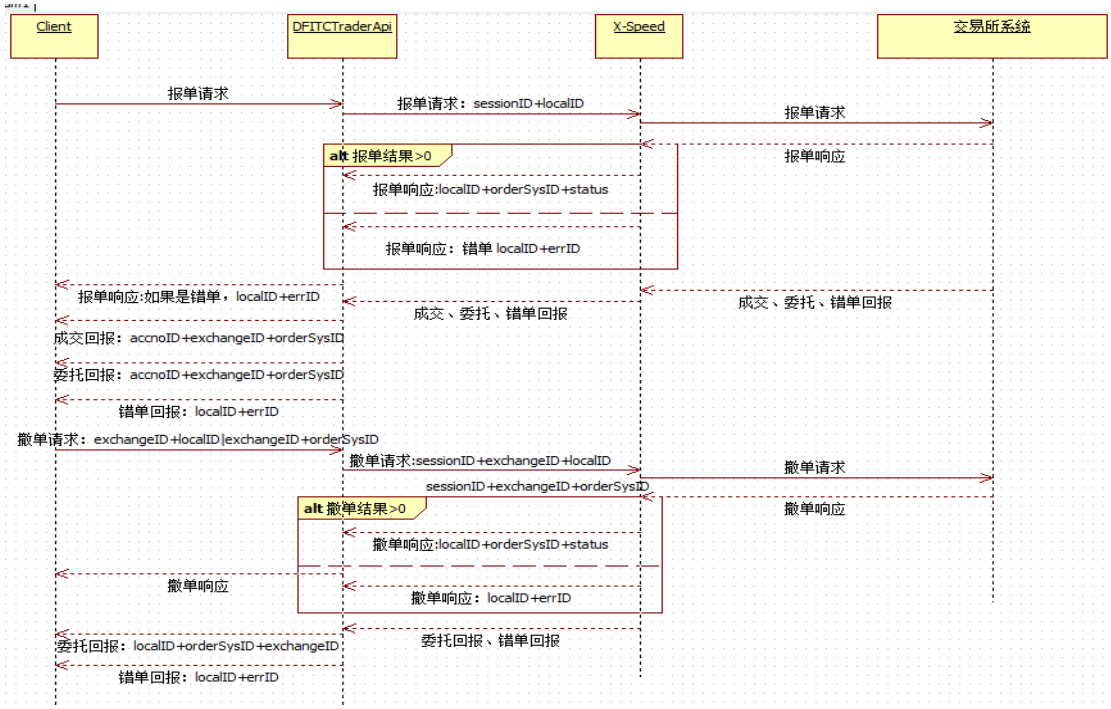
登入成功后，终端程序还会收到私有流数据，如委托回报、成交回报等。

在正式开始交易之前，客户端可能还需要查询如下基础数据，包括：

查询资金：ReqQryCustomerCapital
查询合约：ReqQrySpecifyInstrument
查询持仓：ReqQryPosition
查询持仓明细：ReqQryPositionDetail
查询成交：ReqQryMatchInfo
……

8.2 报单指令

报单指令： ReqInsertOrder。



因为 DFITCTraderApi是异步处理的，所以交易指令都是分2阶段提交。

首先，X-Speed收到交易指令后，客户端会收到报单响应，确认收到客户端的交易指令。

同时，X-Speed把交易指令转发到交易所。

之后，X-Speed 收到来自交易所的响应和回报，通过 DFITCTraderApi 的回报事件通知给客户端。

在报单交易过程中，会产生如下几组交易序列号：

sessionID LocalID lRequestID

用户登入成功后，会收到会话编号SessionID和初始本地委托号initLocalOrderID。

用户在报单时设定本地委托号localOrderID。localOrderID可以从initLocalOrderID开始递增。

请求ID lRequestID用以标示客户每次的请求操作。

spdOrderID orderStatus

X-Speed在收到用户报单时，会为每笔报单生成一个柜台委托号spdOrderID。

在没有得到报单响应前，用户通过本地委托号和合约ID进行撤单操作。在得到报单响应之后，用户就选择使用柜台委托号和合约ID进行撤单操作。

报单响应和回报：

X-Speed 收到报单指令，如果没有通过参数校验，拒绝接受报单指令。用户收到的OnRspInsertOrder委托响应中，“pErrorInfo”指针将不为空，其中包含了错误编码和错误消息。

如果X-Speed接受了报单指令，用户收到OnRspInsertOrder中，“pErrorInfo”指针将为空。

用来更新委托状态。

交易所收到报单后，通过校验。用户会收到OnRtnOrder、OnRtnMatchedInfo。

如果交易所认为报单错误，用户就会收到 OnRtnErrorMsg。

8.3 撤单指令

撤单指令： ReqCancelOrder

撤单输入参数

```
///资金账户ID
DFITCAccountIDType accountID;
///柜台委托号
DFITCSPDOrderIDType spdOrderID;
///本地委托号
DFITCLocalOrderIDType localOrderID;
///合约代码
DFITCInstrumentIDType instrumentID;
```

以上 4 个交易字段中，其中柜台委托号和本地委托号可以二选一使用，如果都设置，会按照柜台委托号，本地委托号的顺序进行处理。

如果报单之后与前置连接断开或者 API 客户端软件关闭重启，在委托查询中可以看到本地委托号都为-1，此时只能通过柜台委托号进行撤单。

撤单响应和回报：

和报单响应和回报相似。

X-Speed 收到撤单指令，如果没有通过参数校验，拒绝接受撤单指令。用户就会收到OnRspCancelOrder撤单响应，且“pErrorInfo”指针将不为空，其中包含了错误编码和错误消息。

如果 X-Speed 接受了撤单指令，用户收到的OnRspCancelOrder中，“pErrorInfo”指针将为空。

交易所收到撤单后，通过校验，执行了撤单操作。用户会收到OnRtnCancelOrder。

如果交易所认为撤单错误，用户就会收到 OnRtnErrorMsg。

8.4 期权开发说明

期权和期货使用的是同一套接口 API，只是在部分结构体中增加如下字段：

DFITCInstrumentTypeType instrumentType; //合约类型

如果设置合约类型字段设置为 0，则表示为期货。（结构体默认初始化为 0）

如果设置合约类型字段设置为 1，则表示为期权。

具体开发规范请参考 DFITCApiStruct.h。

8.5 v14 版本 API 开发注意事项

从该版本 API v1.0.14.81 开始，当 API 连接的是 windows 版本的 xspeed 柜台及前置时，API 可以正常做交易，但是由于 windows 版本的前置相对 API 版本比较低，v13_sp5 版本的 API 相对该版本的 API 没有的字段，则无法获取。

另外，该版本的 API 的请求结构体中，没有了默认构造函数，因此请求接口前，需要进行 memset 操作。

9. 开发样例

SimApiDemo.cpp

具体样例请查看发布包中 demo 工程。