介绍

Xele-Trade介绍

Xele-Trade是业内领先的基于FPGA、微秒级极速交易系统，独创完整流程的FPGA数据传输系统，拥有纳秒级响应速度，提供最快速准确的资讯通道；是为证券、期货高端人士及机构、基金类专业投资机构及产业巨头量身打造的高性能交易系统。
该文档是极速交易系统Xele-Trade的投资者使用手册，它提供API功能及其使用说明，展示投资者开发客户端程序(Client Program)的通用步骤。

TraderAPI简介

TraderAPI用于与Xele-Trade进行通信。 通过API，投资者可以向多个交易所（SHFE，CFFEX, INE, CZCE, DCE）发送交易指令，获得相应的回复和交易状态回报。
TraderAPI 是一个基于 C++的类库， 通过使用和扩展类库提供的接口来实现全部的交易功能。发布给用户的文件包以**xele-futures-trader-api-linux-版本号.tgz** 命名， 包含的文件如下：

文件名	说明
include/CXeleTraderApi.hpp	API头文件, 定义了目前支持的接口
include/CXeleFtdcUserApiStruct.h	定义了函数接口的参数结构体类型
include/CXeleFtdcUserApiDataType.h	定义了使用到的字段的类型
libXeleTdAPI64.so	Linux 64位版本的API动态库
userdemo.cpp	userdemo示例程序的源代码文件
config_userdemo.ini	userdemo示例程序的配置文件
Makefile	Userdemo示例程序的编译文件

TraderAPI发行平台

目前发布的主要基于**Linux 64位操作系统**的版本，包括动态库.so文件和.h头文件；
备注说明: *如果客户程序使用API出现异常奔溃，这并不表示一定是API自身异常，但API会捕捉该异常信号，并记录异常调用栈信息协助定位，生成的异常文件一般在客户程序所在目录，命名格式为xeleapi-year-month-day-hour-minute-second.coredump（例如xeleapi-2020-09-11-16_46_02.coredump）,生成的codrdump文件大小限制最大为8G；*

TraderAPI通讯模式和数据流

通讯模式

Xele-Trade系统所有的通讯都是基于某个通讯模式，通讯模式实际上就是通讯双方协同工作的方式，主要的通讯模式有两种：**对话通讯模式**和**私有通讯模式**。

对话通讯模式

对话通讯模式是指由用户主动发起的通讯请求。该请求被柜台系统接收和处理，并给予响应。例如报单、查询等。这种通讯模式与普通的客户/服务器模式相同。

私有通讯模式

私有通讯模式是指柜台系统主动向用户发出的信息，例如成交回报等。
通讯模式和网络的连接不一定存在简单的一对一的关系。也就是说，一个网络连接中可能传送多种不同通讯模式的报文，一种通讯模式的报文也可以在多个不同的连接中传送。比如一个网络链接中，存在报单请求、响应以及成交回报等。

数据流

按照通讯模式划分

- **对话通讯模式**：该模式下支持**对话数据流。**
  对话数据流：是一个双向数据流，用户发送请求，柜台系统反馈应答。柜台系统不维护对话流的状态。柜台系统故障时，对话数据流会重置，在途的数据可能会丢失。
- **私有通讯模式**：该模式下支持**私有数据流。**
  私有数据流：是一个单向数据流，主要是柜台系统主动发送给用户的数据；柜台系统目前不维护私有流的状态。柜台系统故障时，私有数据流会重置，在途的数据可能会丢失。

按照业务类型划分

Xele-Trade系统主要支持交易和查询两大类业务，所以涉及到交易相关的数据流称为**交易数据流**，查询相关的称为**查询数据流；**

- **交易数据流**：用户交易过程中所涉及的相关数据流，包括了对话通讯模式下的对话数据流和私有通讯模式下的私有数据流、例如报单过程中的报单请求、报单响应、报单回报等；
- **查询数据流**：用户查询过程中所涉及的相关数据流，包括了对话通讯模式下的查询数据流；例如合约查询过程中的查询请求、查询响应等；
  备注说明：
  数据流的描述方式并不是孤立的，比如交易数据流可以是对话数据流（报单请求和响应），同时也可以是私有数据流（成交回报等）。此外，本档中的其他地方关于数据流描述不再详细区分数据流的划分模式，用户在阅读文档时，可以结合本章节，具体理解文档中数据流的描述的方式。

TraderAPI接口类型及参数说明

对话通讯模式接口类型

TraderAPI提供了2类接口，分别为**CXeleTraderApi**和**CXeleTraderSpi**。

```
////请求
int CXeleTraderApi::ReqXXX(
    CXeleFtdcXXXField *pReqXXX,
    int nRequestID)
////回应：
void CXeleTraderSpi::OnRspXXX(
    CXeleFtdcXXXField *pRspXXX,
    CXeleFtdcRspInfoField *pRspInfo,
    int nRequestID,
bool bIsLast)
```

该模式的相关接口以ReqXXX，OnRspXXX来命名。该模式调用API的ReqXXX来发起标识为nRequestID的请求，在OnRspXXX中回复相应的nRequestID的响应结果。

1.请求接口参数说明
接口名称：ReqXXX

- pReqXXX：请求的参数结构体指针
- nRequestID：由用户管理的请求标识， 与响应关联

2.响应接口参数说明

接口名称：OnRspXXX

- pRspXXX: 响应报文的参数结构体指针
- pRspInfo：RspInfo表示结果域，含有ErrorID和ErrorMsg表达该次请求的状态
- nRequestID：响应的请求ID
- bIsLast：该次请求ID是否响应完毕，即一次响应报文中，该响应回调函数可能会被多次调用，标识报文结束时的最后一次调用

私有通讯模式接口类型

```
void    CXeleTraderSpi::OnRtnXXX(CXeleFtdcXXXField *pXXX);
////或者
void     CXeleTraderSpi::OnErrRtnXXX(CXeleFtdcXXXField *pXXX,
                            CXeleFtdcRspInfoField *pRspInfo);
```

该模式的相关接口以OnRtnXXX, OnErrRtnXXX来命名，分别表示正常通知响应私有流回报和错误通知响应私有流回报。

1.标准通知响应接口参数说明
接口名称：OnRtnXXX

- pXXX：相应的通知回调数据域指针

2.错误通知响应接口参数说明
接口名称：OnErrRtnXXX

- pXXX：相应的通知回调数据域指针
- pRspInfo：错误类型和错误信息

TraderAPI运行机制

工作流程

客户端和柜台交易系统的交互过程分为2个阶段：初始化阶段和功能调用阶段。

初始化阶段

在初始化阶段，Xele-Tradede交易系统的程序必须完成如下步骤：

- CreateTraderApi(): 获取可用的API对象指针；
- egisterSpi(): 注册用户继承自CXeleTraderSpi的自有子类对象的地址；
- RegisterFront(): 设置分配给用户的交易数据流和查询数据流的通道连接地址和端口以及用户本地报撤单的IP和port， 格式为“<protocal>://<ip_address>:<port>”, 如“tcp://127.0.0.1:8080”；参数的具体说明详见5.2.6章节；
- RegisterChannelBlock()：设置报单回报套接字为阻塞/非阻塞模式，需在Init前调用。不设置的话，默认是非阻塞；

- Init(): 初始化通道并连接流, 如果所有流都连接成功，Spi的OnFrontConnected()会被调用；

示例代码：API初始化流程：

```
/* 初始化流程 */
CXeleTraderApi* pTraderApi = CXeleTraderApi::CreateTraderApi();
  CSimpleOrderManager som(pTraderApi);
  pTraderApi->RegisterSpi(&som);
  pTraderApi->RegisterFront(g_FrontAddress_str, g_QueryFrontAddress_str,
g_LocalAddress_str);
  /* 默认报单通讯是非阻塞模式，若设置阻塞模式请在调用Init函数之前设置 */
  pTraderApi->RegisterChannelBlock(1);
  /* 客户端程序开始与交易柜台建立连接*/
  pTraderApi->Init();
```

功能调用阶段

在功能调用阶段，客户端程序可以任意调用交易接口中的请求方法，如：ReqUserLogin、ReqOrderInsert 等，同时用户需要继承SPI回调函数以获取响应回报信息。
备注说明：
*API 请求的输入参数不能为 NULL；API 请求对应的响应接口返回参数**pRspInfo 中的**ErrorID**，**0**表示正确，其他表示错误，详细错误编码请参考附录中的错误码说明。*
示例代码：功能调用及回调函数

```
/* 客户端程序和柜台建立链接后，在SPI的链接通知回调函数中，发送登录请求进行登录操作*/
  virtual void OnFrontConnected()
  {
   example_ReqUserLogin(&login_info);
/* 登录请求函数*/
   m_pTraderApi->ReqUserLogin(&login_info, 1);
  }
```

工作线程

客户端程序和交易前端之间的通信是由API工作线程驱动的，客户端程序至少存在4个线程组成：
客户端主线程：主要是客户发送请求和处理接收到的API的回报信息；
API心跳线程：主要是维护客户端和柜台的心跳；
API回报处理线程：主要是接收和处理柜台系统的交易回报数据；
API预热加速线程：主要是提升API的报单性能的预热线程；
其中只有交易回报处理线程和查询回报处理线程支持绑核操作，具体详见5.2.9 RegisterWorkerAffinity方法。
备注说明：
*1.由CXeleTraderApi和CXeleTraderSpi提供的接口**不是线程安全的**。多个线程同时为单个**CXeleTraderApi**实例发送请求是不允许的。*
*2.用户注册的SPI回调函数需要尽快处理相关数据，如果SPI回调接口阻塞了API的查询回报/交易回报处理线程，**将会影响该处理线程正常接收后续的数据**。*

心跳机制

Xele-Trade柜台系统每隔一定时间对查询流和交易流分别发送心跳报文给客户端来维持连接，API也会定时发送心跳报文来维持链接（间隔为10s, 超时时间为60s），API和柜台系统之间的心跳由系统自动维护，无需用户干预。

响应报文的通知机制

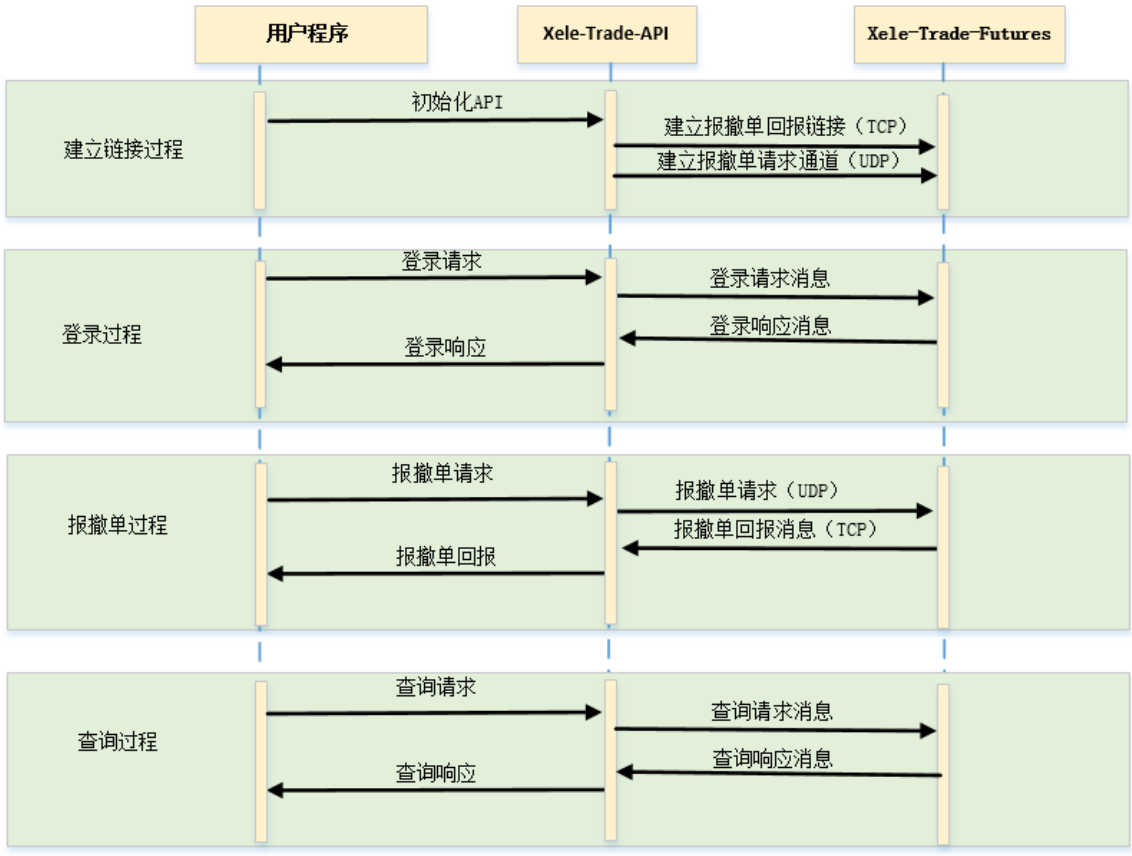
用户自定义响应处理的方法为创建一个类为CXeleTraderSpi的子类，并使用RegisterSpi接口注册给API，具体示例参见3.2.1初始化阶段。在CXeleTraderSpi回调基类中有两个接口OnPackageStart()和OnPackageEnd()，参数均为（topic，sequenceNo），用于通知用户收到了topic类型的流，该流的编号是sequenceNo。所有回调类型的响应接口均会在调用之前调用OnPackageStart，在回调完毕后调用OnPackageEnd。

示例代码： 响应报文的回调函数

```
/* 报单插入的响应报文回调处理示例*/
   virtual void OnRspOrderInsert(CXeleFtdcInputOrderField *pInputOrder,
CXeleFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
  {
    if (pRspInfo->ErrorID != 0)
    {
/* 接收到错误响应后的处理流程*/
      ……
      exit(pRspInfo->ErrorID);
    }
    else
    {
  /* 接收到正确响应后的处理流程*/
      ……
  }
```

TraderAPI基本操作说明

TraderAPI和Xele-Trade柜台系统的信息交互主要包括：建立链接、登录登出、报撤单操作、查询操作。TraderAPI和柜台系统的的上述信息交互流程如下图所示。
用户需要通过API的ReqUserLogin接口登录柜台系统成功后才可以调用ReqOrderInsert或者ReqOrderAction接口进行报撤单操作。



登录过程

初始化网络连接后，就可以通过接口ReqUserLogin进行登录请求了，调用OnRspUserLogin回调接口处理登录响应，只有登录成功后才能进行业务处理。**登录成功后，在响应报文中返回用户一组或者两组（柜台是双交易所模式）的ClientIndex和Token信息，用户报撤单需要填写该信息。**

备注说明：

1、如果柜台是双交易所模式（当前只支持上期和原油双交易所），则第一组ClientIndex和Token是上期的，第二组是原油的；其他情况只会返回一组信息（Token为0和-1均表示无效值）；

2、用户登录过程中，柜台会向客户端推送流水报单回报OnRtnHistoryOrder和历史成交回报OnRtnHistoryTrade，只有历史回报接收完成后，才会返回登录响应回报OnRspUserLogin。

3、如果盘中客户端掉线重连并且重新登录，此时API可能有较多的历史流水需要回调，由于只有历史流水回调完成后API才会回调登录响应接口，此时用户可以通过历史流水回调接口OnRtnHistoryTrade和OnRtnHistoryOrder来判断流水是否回调完成；

4、用户每次重新登录后，都需要重新获取token以及合约序号，才可以进行报撤单操作；

示例代码： 用户登录：

```
  /* 当API与交易柜台建立起通信连接后，客户端程序可以进行登录操作 */
  virtual void OnFrontConnected()
  {
    example_ReqUserLogin(&login_info);
    m_pTraderApi->ReqUserLogin(&login_info, 0);
  }
  /* 当客户端程序发出登录请求之后，该方法会被调用，通知客户端程序登录是否成功  */
  virtual void OnRspUserLogin(CXeleFtdcRspUserLoginField *pRspUserLogin,
CXeleFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
  {
    if (pRspInfo->ErrorID != 0)
    {
      /* 登录请求失败时的处理 */
      ……
      exit(pRspInfo->ErrorID);
    }
    else
    {
      /*登录请求成功时的处理 ，记录相关信息方便进行报撤单操作*/
      m_exchange_num    = pRspUserLogin->ExchangeNum;
      m_client_index[0] = pRspUserLogin->ClientIndex[0];
      m_token[0]        = pRspUserLogin->Token[0];
      m_client_index[1] = pRspUserLogin->ClientIndex[1];
      m_token[1]        = pRspUserLogin->Token[1];
    }
  }
```

查询过程

客户端可以根据查询内容的不同调用对应的ReqQryXXX查询接口，对应的OnRspQryXXX响应接口会发送相关响应信息。

备注说明：

*1. 只有等待上一次查询操作结束后才能进行下一次查询，否则Rspinfo中会返回Errorid=2044查询未完成的错误；

2. 如果某个查询操作有多个响应信息，则柜台系统会向客户端发送多个对应的响应报文，客户可以根据响应中bIsLast字段是否为true来判断是否为最后一个响应报文；*

示例代码： 成交单查询：

```
/* 成交单查询请求 */
  void ReqQryTrade(size_t trader = 0)
  {
    CXeleFtdcQryTradeField qryTrade;
    memset(&qryTrade, 0, sizeof(CXeleFtdcQryTradeField));
    snprintf(qryTrade.AccountID, sizeof(qryTrade.AccountID), "%s",
g_AccountID_str[trader]);    /* 客户号 */
    strcpy(qryTrade.InstrumentID, "cu1705");      /* 合约代码 */
    strcpy(qryTrade.TradeID, "233");                    /* 成交编号 */
    m_pTraderApi->ReqQryTrade(&qryTrade, 1); /* 查询请求 */
  }
  /* 成交单查询响应 */
  virtual void OnRspQryTrade(CXeleFtdcTradeField* pTradeField,
CXeleFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
  {
    if (pRspInfo->ErrorID != 0) {
      PRINT_RSP_ERR(pRspInfo);      /* 查询错误响应的处理  */
      exit(pRspInfo->ErrorID);
    } else  {
      PRINT_RSP(pRspInfo);    /* 查询成功的处理 */
    }
  }
```

报单过程

如上图所示，报单的流程主要有上述三种场景。

报单的报文说明：

- 不管哪种场景，柜台系统都会发送报单响应OnRspOrderInsert报文；
- 当报单未通过柜台系统或者交易所风控时，柜台系统都会发送错误回报OnErrRtnOrderInsert报文;
- 当报单成功进入交易所后，报单发生的任何状态变化，柜台系统都会发送报单回报OnRtnOrder报文；
- 当报单成功进入交易所后，报单发生的任何成交，柜台系统都会发送成交回报OnRtnTrade报文；

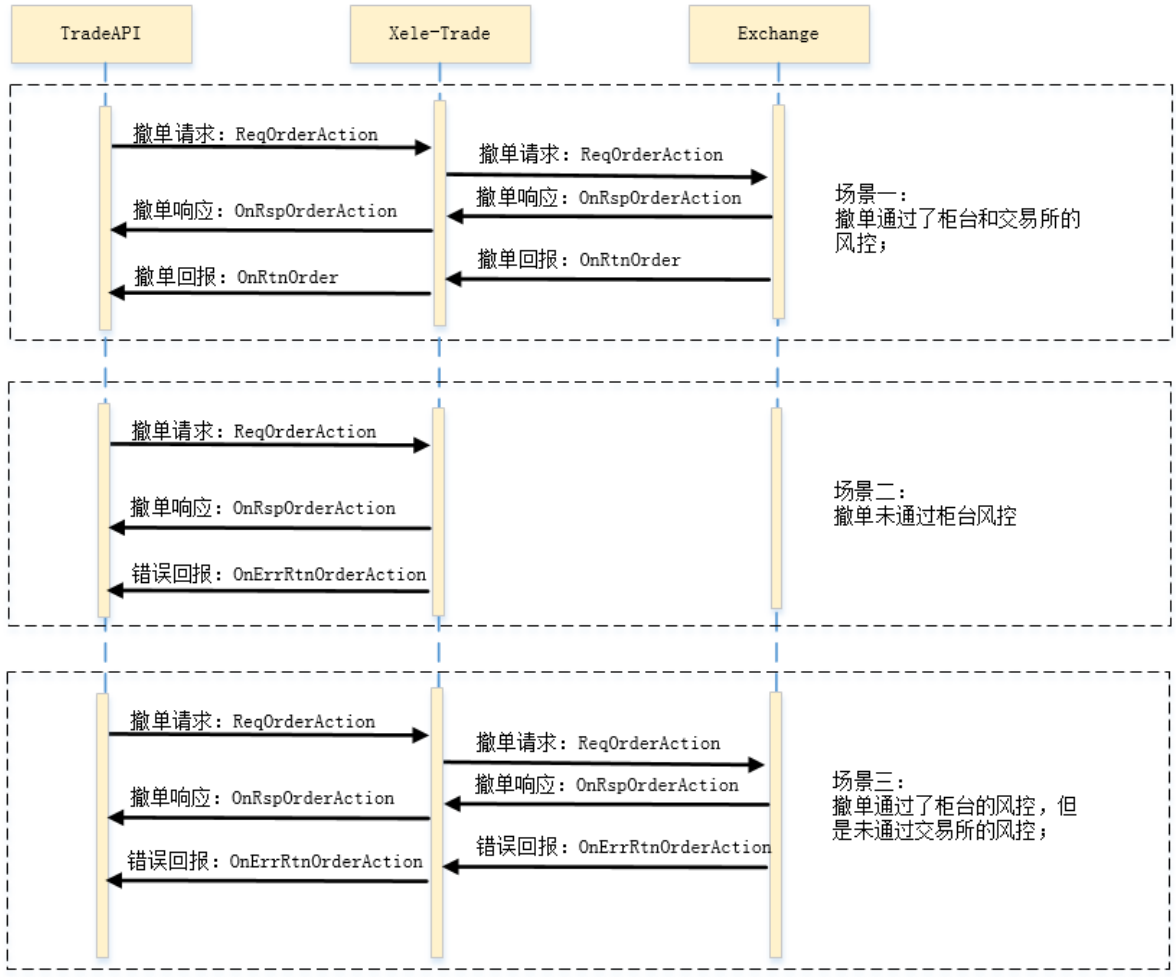
报单的单号管理：

- 报单时需要维护的单号字段有OrderLocalNo和OrderSystemNo。
- 在报单请求的ReqOrderInsert接口参数的结构体字段中，OrderLocalNo是用户自己管理的报单编号（**建议单调递增**）；
- 在报单响应的OnRspOrderInsert接口参数的结构体字段中，OrderLocalNo是柜台系统返回给用户的自己管理的报单编号；**OrderSystemNo是柜台系统生成的系统报单编号，该编号也是用户进行撤单的唯一依据。**
- 在报单回报的OnRtnOrder接口参数的结构体字段中，OrderLocalNo是用户自己管理的报单编号，OrderSystemNo 是柜台系统生成的报单编号；
- 在成交回报的OnRtnTrade接口参数的结构体字段中，OrderLocalNo是用户自己管理的报单编号，OrderSystemNo 是柜台系统生成的报单编号；TradeID是交易所生成的成交编号；
  举例说明：
  报单请求：用户在调用ReqOrderInsert中，填写了自己的报单编号OrderLocalNo=0001，报单响应：在OnRspOrderInsert中会返回用户自己的报单编号OrderLocalNo=0001，并且还返回柜台生成的系统报单编号OrderSystemNo=000009，如果用户进行撤单操作，必须使用OrderSystemNo；

示例代码： 报单输入：

```
/* 报单输入代码示例 */
/* 如果是双交易所模式，可以循环对每个交易所进行报单输入*/
    for(i = 0; i < m_exchange_num; i++)
    {
      memset(&order, 0, sizeof(CXeleFairInputOrderField));
      example_makeup_order(&order, 0, m_client_index[i], m_token[i]);
      sleep(2);
/* 调用报单输入接口进行报单 */
      m_pTraderApi->ReqOrderInsert(&order, 1);
    }
```

撤单过程



如上图所示，撤单的流程主要有上述三种场景。

撤单的报文说明：

- 不管哪种场景，柜台系统都会发送撤单响应OnRspOrderAction报文；
- 当撤单未通过柜台系统或者交易所风控时，柜台系统都会发送错误回报OnErrRtnOrderAction报文;
- 当撤单成功进入交易所后，撤单发生的任何状态变化，柜台系统都会发送撤单回报OnRtnOrder报文；

  撤单的单号管理：

  撤单时需要维护的单号字段有ActionLocalNo和OrderSysNo。
- 在撤单请求的ReqOrderAction接口参数的结构体字段中，ActionLocalNo是用户自己管理的撤单编号（**建议单调递增**），OrderSysNo是对应用户报单响应中OnRspOrderInsert返回的系统报单编号OrderSystemNo；
- 在撤单响应的OnRspOrderAction接口参数的结构体字段中：

  ActionLocalID 是柜台系统生成的本次撤单的系统编号；

ActionLocalNo是柜台系统返回的撤单请求中用户自己管理的撤单编号（对应ReqOrderAction中的ActionLocalNo）；
OrderSystemNo是柜台系统返回给用户的对应的报单的柜台系统报单编号（对应OnRspOrderInsert中的OrderSystemNo）；
OrderLocalNo是柜台系统返回给用户的对应的报单的用户自己管理的报单编号（对应ReqOrderInsert中的OrderLocalNo）

- 在撤单回报的OnRtnOrder接口参数的结构体字段中，OrderLocalNo用户自己管理的报单编号，OrderSystemNo 是柜台系统生成的撤单编号；
  举例说明：
- *报单请求：用户在调用ReqOrderInsert中，填写了自己的报单编OrderLocalNo=0001；*
- *报单响应：在OnRspOrderInsert中会返回用户自己的报单编号OrderLocalNo=0001，并且*
- *还返回柜台生成的系统报单编号OrderSystemNo=000009；*
- *撤单请求：调用ReqOrderAction中，填写了自己管理的撤单编号ActionLocalNo=0003，以及撤单的系统编号OrderSysNo=000009；*
- *撤单响应：在OnRspOrderAction中，返回了用户自己管理的撤单编号ActionLocalNo=0003；柜台系统生成的撤单系统编号ActionLocalID=0004；对应的报单的系统编号OrderSystemNo=000009；对应的报单的用户自己管理的报单编号**OrderLocalNo=0001；*

示例代码：撤单请求：

```
/* 如果是双交易所模式，可以循环对每个交易所进行撤单*/
     for(i = 0; i < m_exchange_num; i++)
    {
      memset(&action, 0, sizeof(CXeleFairOrderActionField));
      example_makeup_action(&action, 0, m_client_index[i], m_token[i],
m_ordersysno[i]);
      m_pTraderApi->ReqOrderAction(&action, 1); /*撤单API接口*/
    }
```

TraderAPI接口参考说明

TraderAPI接口分类

非业务接口

1.TraderApi非业务接口

接口类型	接口分类	说明
生命周期管理接口	CXeleTraderApi:: CreateTraderApi	创建API对象
生命周期管理接口	CXeleTraderApi:: GetVersion	获取当前API的版本信息
生命周期管理接口	CXeleTraderApi:: Release	销毁API对象
生命周期管理接口	CXeleTraderApi:: Init	初始化网络连接
生命周期管理接口	CXeleTraderApi:: Join	等待线程结束
注册管理接口	CXeleTraderApi::RegisterFront	注册连接地址
注册管理接口	CXeleTraderApi:: RegisterSpi	注册回调对象
注册管理接口	CXeleTraderApi:: RegisterAuthentication	注册授权接口
注册管理接口	CXeleTraderApi::RegisterWorkerAffinity	注册接口线程亲和性
注册管理接口	CXeleTraderApi:: RegisterChannelBlock	注册通道阻塞属性

2.TradeSpi非业务接口

接口类型	接口分类	说明
网络连接	CXeleTraderSpi::OnFrontConnected	当客户端成功建立网络连接时被调用
网络连接	CXeleTraderSpi:: OnFrontDisconnected	当客户端失去网络连接时被调用
报文回调	CXeleTraderSpi:: OnPackageStart	报文回调开始
报文回调	CXeleTraderSpi:: OnPackageEnd	报文回调结束

业务接口

业务类型	业务	请求接口 / 响应接口	数据流
登陆	登陆	CxeleTraderApi::ReqUserLogin CXeleTraderSpi::OnRspUserLogin	对话流
订阅	订阅公有流(当前不支持)	CXeleTraderApi:: SubscribePrivateTopic	NA
订阅	订阅公有流(当前不支持)	CXeleTraderApi:: SubscribePublicTopic	NA
订阅	订阅公有流(当前不支持)	CXeleTraderApi:: SubscribeUserTopic	NA
交易	报单录入	CXeleTraderApi::ReqOrderInsert CXeleTraderSpi::OnRspOrderInsert	对话流
交易	报单操作	CXeleTraderApi::ReqOrderAction CXeleTraderSpi::OnRspOrderAction	对话流
交易	批量委托单	CXeleTraderApi:: ReqOrderInsertActionV	对话流
查询	客户令牌查询	CXeleTraderApi:: GetClientToken	NA
查询	合约序号查询（异步)	CXeleTraderApi:: ReqQryInstrumentIndex	对话流
查询	合约序号查询（同步)	CXeleTraderApi:: GetInstrumentIndex	NA
查询	客户资金查询	CXeleTraderApi::ReqQryClientAccount CXeleTraderSpi::OnRspQryClientAccount	查询流
查询	客户持仓查询	CXeleTraderApi::ReqQryClientPosition CXeleTraderSpi::OnRspQryClientPosition	查询流
查询	合约查询	CXeleTraderApi::ReqQryInstrument CXeleTraderSpi::OnRspQryInstrument	查询流

业务类型	业务	请求接口 / 响应接口	数据流
查询	合约保证金率查询	CXeleTraderApi::ReqQryInstrumentMarginRate CXeleTraderSpi::OnRspQryInstrumentMarginRat	查询流
查询	合约手续费率查询	CXeleTraderApi::ReqQryInstrumentCommissionRate CXeleTraderSpi::OnRspQryInstrumentCommissionRate	查询流
查询	用户密码更新	CXeleTraderApi:: ReqUserPasswordUpdate	
查询	报单查询	CXeleTraderApi::ReqQryOrder CXeleTraderSpi::OnRspQryOrder	查询流
查询	成交查询	CXeleTraderApi:: ReqQryTrade CXeleTraderSpi::OnRspQryTrade	查询流
查询	交易所前置查询	CXeleTraderApi:: ReqQryExchangeFront CXeleTraderSpi::OnRspQryExchangeFront	查询流
查询	合约价格查询	CXeleTraderApi:: ReqQryInstrumentPrice CXeleTraderSpi:: OnRspInstrumentPrice	查询流
私有回报	成交回报	CXeleTraderSpi:: OnRtnTrade	私有流
私有回报	报单回报	CXeleTraderSpi:: OnRtnOrder	私有流
私有回报	历史成交回报	CXeleTraderSpi:: OnRtnHistoryTrade	私有流
私有回报	历史报单回报	CXeleTraderSpi:: OnRtnHistoryOrder	私有流

业务类型	业务	请求接口 / 响应接口	数据流
私有回报	增加合约通知	CXeleTraderSpi:: OnRtnInsInstrument	私有流
私有回报	合约交易状态改变回报	CXeleTraderSpi:: OnRtnInstrumentStatus	私有流
私有回报	报单录入错误回报	CXeleTraderSpi:: OnErrRtnOrderInsert	私有流
私有回报	报单操作错误回报	CXeleTraderSpi:: OnErrRtnOrderAction	私有流

API非业务类接口

CreateTraderApi方法

功能：创建一个API对象
函数原形：

```
static CXeleTraderApi *CreateTraderApi(int exchangeID=0)
```

参数：exchangeID入参，预留字段，暂时未使用；
返回值：合法的API对象指针

GetVersion方法

功能：获取当前API的版本信息
函数原形：

```
static const char *GetVersion()
```

参数：无；
返回值：API版本信息字符串的常量指针；

Release方法

功能：释放创建的API对象，不能使用delete方法；
函数原形：

```
void Release()
```

参数：无；
返回值：无；
说明：调用者为CreateTraderApi生成的合法指针对象；

Init方法

功能：初始化客户端和柜台系统的网络连接，链接成功后可以进行登录操作；
函数原形：

```
void Init(bool mode=true)
```

参数：mode， 入参，在API的3.2.19版本及之后的版本中，该参数预留不再有效；在API的3.2.19的之前版本中，该参数为false表示查询模式（仅支持查询操作），为true表示报单模式（支持报单和查询操作）；
返回值：无；

Join方法

功能：API会等待接口实例的线程结束；
函数原形：

```
int Join()
```

参数： 无；
返回值：0 表示正常，其他表示异常；
说明：阻塞等待回调线程结束，否则立即返回。

RegisterFront方法

功能：设置客户端和柜台系统的网络通讯地址；
函数原形：

```
void RegisterFront(
char *pszFrontAddress,
char *pszQueryFrontAddress,
char *pszLocalTradeAddress)
```

参数：

- pszFrontAddress
  入参，柜台系统交易数据流上行UDP链路IP地址和端口，格式为<*protocol*>://<*address*>:<*port*>，如udp://1.1.1.1:32001
- pszQueryFrontAddress
  入参，柜台系统查询数据流上下行TCP链接IP地址和端口，格式为<*protocol*>://<*address*>:<*port*>， 如tcp://1.1.1.1:33333；
- pszLocalTradeAddress
  入参，客户端本地报撤单的IP地址和port，格式为<*protocol*>://<*address*>:<*port*>， 如udp://1.1.1.1:32000；
  该IP地址需要是客户端运行环境上的同一机器上的IP，port可填写或不填写；
  返回值： 无；
  说明：
- 需要在Init之前调用，格式需要按照要求的格式填写；

- 柜台系统查询数据流上下行都为TCP链接，只有1个链接地址和port（即pszQueryFrontAddress）；
- 柜台系统的交易数据流上行是UDP链路（即pszFrontAddress ），下行是TCP链接（客户端不感知，该链接的IP和查询数据流的pszQueryFrontAddress中的IP一样，port为 查询数据流的port+1）；
- **本地报撤单IP需要是客户端运行环境上的同一机器上的IP，port可以填写或者不填写，在客户端报撤单时会柜台系统会进行报撤单IP校验，如果IP错误，报撤单会失败；**

RegisterSpi方法

功能：注册一个派生自CxeleTraderSpi接口类的实例， 该实例将完成事件处理
函数原形：

```
void RegisterSpi(CXeleTraderSpi *pSpi)
```

参数：

- pSpi：入参，实现了CXeleTraderSpi接口的实例指针。
  返回值： 无；

RegisterAuthentication方法

功能：注册授权接口，注册APPID与授权码。
函数原形：

```
void RegisterAuthentication(
CXeleFtdcAuthenticationInfoField  *pAuthenticationInfoField)
```

参数:

- pAuthenticationInfoField ：入参，终端软件授权，结构体CxeleFtdcAuthenticationInfoField如下

```
struct CXeleFtdcAuthenticationInfoField {
///终端软件AppID
TXeleFtdcAppIDType AppID;
//终端软件授权码
TXeleFtdcAuthCodeType AuthCode;
}
```

返回值： 无；
说明： 要在ReqUserLogin接口之前调用。

RegisterWorkerAffinity方法

功能：注册接口线程亲和性，亲和性在操作系统中子进程或线程会继承，在接口线程上下文中起新的线程要注意 。
函数原形：

```
int RegisterWorkerAffinity(int *cores, int size)
```

参数:

- cores: 入参，每个工作线程被分配的处理器核心序号，有效范围: [0, 系统总核心数-1]；

- size：入参，cores数组的大小，代表需要配置亲和性的线程数；
  返回值: 0表示成功， 其他值为异常；
  说明：
- 需要在Init()之前调用，默认为操作系统自动决定。
- 该接口当前只对设置查询数据流和交易数据流回报处理线程有效；如果cores设置了多个元素，则第1个元素表示API接收回报线程的CPU核心，第2个元素表示预热加速线程CPU核心（建议预热加速线程的CPU核心绑定到报单CPU核心上）

```
/* 示例： 要把工作线程分别绑定到CPU核心3 和4上*/
CXeleTraderApi *api = CXeleTraderApi::CreateTraderApi();
……
int cores[2];
int core_size = 2;
cores[0] = 3;
cores[1] = 4;
int ret;
ret = api->RegisterWorkerAffinity(cores, core_size);
api->Init();
```

RegisterChannelBlock方法

功能：注册交易数据流通道阻塞属性 。
函数原形：

```
int RegisterChannelBlock( short flag=0)
```

参数: flag: 入参，通道阻塞属性。(0:非阻塞,1: 阻塞)
返回值: 0表示成功， 其他值为异常；
说明： 要在Init( )之前调用，当前该接口只对交易数据流的下行TCP链路设置有效，查询数据流的TCP链路默认是阻塞；

API业务类接口

API业务接口的通用说明：
1.业务接口中的RequestID范围：柜台系统目前支持的request_id范围为0~0x0fffffff(十进制268435455)， 超出这个范围的request_id会报错；
2.接口返回值：如果接口带有返回值，0表示函数处理正常，其他值表示处理异常；

ReqUserLogin方法

功能：用户登录请求接口
函数原形：

```
int ReqUserLogin(
CXeleFtdcReqUserLoginField      *pReqUserLogin,
int     nRequestID)
```

参数：

- pReqUserLogin：入参，登录请求域，指向用户请求结构的地址。
- nRequestID：入参，请求ID。
  用户登录请求结构体CxeleFtdcReqUserLoginField如下：

```
struct CXeleFtdcReqUserLoginField {
  ///未用字段
  char  unused1_[7];
  ///保留字段
  TXeleFtdcSessionType  Session;
  ///交易用户代码
  TXeleFtdcAccountIDType    AccountID;
  ///未用字段
  char  unused2_[14];
  ///交易用户密码
  TXeleFtdcPasswordType     Password;
  ///用户端产品信息（程序自动填写）
  TXeleFtdcProductInfoType  UserProductInfo;
  ///接口端产品信息（程序自动填写）
  TXeleFtdcProductInfoType  InterfaceProductInfo;
  ///组合合约授权标记(0:不授权，1:授权，只有大商所使用)
  TXeleFtdcCombinedContractAuthorizeType    CombinedContractAuthorize;
  ///产品保留字段(程序自动填写)
  TXeleFtdcLoginReserve1Type    ProductReserve1;
  ///产品保留字段(程序自动填写)
  TXeleFtdcLoginReserve1Type    ProductReserve2;
  ///产品保留字段(程序自动填写)
  TXeleFtdcLoginReserve1Type    ProductReserve3;
  ///产品保留字段(程序自动填写)
  TXeleFtdcLoginReserve1Type     ProductReserve4;
  ///未用字段
  char unused3_[28];
};
```

返回值： 0表示正常， 其他值异常；

ReqUserPasswordUpdate方法

功能：修改用户密码
函数原形：

```
int ReqUserPasswordUpdate(
CXeleFtdcUserPasswordUpdateField        *pUserPasswordUpdate,
 int    nRequestID)
```

参数：

- pUserPasswordUpdate：入参，密码修改域，指向用户口令修改结构的地址。
- nRequestID：入参，请求ID。
  用户密码修改结构体CxeleFtdcUserPasswordUpdateField如下：

```
struct CXeleFtdcUserPasswordUpdateField {
  ///交易用户代码
  TXeleFtdcAccountIDType        AccountID;
  ///未用字段
  Char    Padding_Three[3];
  ///会员代码
  TXeleFtdcParticipantIDType    ParticipantID;
  ///旧密码
  TXeleFtdcPasswordType         OldPassword;
  ///新密码
  TXeleFtdcPasswordType     NewPassword;
};
```

返回值：0表示正常， 其他值异常；

SubscribePrivateTopic方法

功能：订阅会员私有流，该方法要在Init之前调用；
函数原形：

```
void SubscribePrivateTopic(XELE_TE_RESUME_TYPE  nResumeType)
```

参数：

- nResumeType：入参，报文流的重传发送方式：

```
enum XELE_TE_RESUME_TYPE
{
XELE_TERT_RESTART = 0, //从第一个报文序号发送
XELE_TERT_RESUME,      // 从记录的报文序号发送
XELE_TERT_QUICK        // 从当前流最大序号发送
};
```

返回值：无；
说明：当前柜台仅支持XELE_TERT_QUICK方式重传；

SubscribePublicTopic方法

功能：订阅公有流，该方法要在Init方法前调用。
函数原形：

```
void SubscribePublicTopic(XELE_TE_RESUME_TYPE   nResumeType)
```

参数：

- nResumeType：入参，报文流的重传发送方式：

```
enum XELE_TE_RESUME_TYPE
{
XELE_TERT_RESTART = 0, //从第一个报文序号发送
XELE_TERT_RESUME,      // 从记录的报文序号发送
XELE_TERT_QUICK        // 从当前流最大序号发送
};
```

返回值：无；
说明：当前柜台仅支持XELE_TERT_QUICK方式重传；

SubscribeUserTopic方法

功能：订阅交易员私有流。该方法要在Init方法前调用。
函数原形：

```
void SubscribeUserTopic(XELE_TE_RESUME_TYPE  nResumeType)
```

参数：

- nResumeType：入参，报文流的重传发送方式：

```
{
XELE_TERT_RESTART = 0, //从第一个报文序号发送
XELE_TERT_RESUME,      // 从记录的报文序号发送
XELE_TERT_QUICK        // 从当前流最大序号发送
};
```

返回值：无；
说明：当前柜台仅支持XELE_TERT_QUICK方式重传；

ReqOrderInsert方法

功能：报单录入请求
函数原形：

```
int ReqOrderInsert(
CXeleFairInputOrderField    *pInputOrder,
    int     nRequestID)
```

参数：

- pInputOrder：入参，报单录入域，指向报单录入结构的地址。
- nRequestID：入参，请求ID。
  报单录入请求结构体CxeleFairInputOrderField：

```
struct CXeleFairInputOrderField {
 ///单笔输入报单时，由程序自动填写； 批量委托单时，该值需客户填写为0x65
TXeleFtdcProductReserve1Type    ProductReserve1;
///客户编号
 TXeleFtdcClientIndexType       ClientIndex;
///客户令牌
TXeleFtdcClientTokenType        ClientToken;
      ///产品保留字段，由程序自动填写
  TXeleFtdcProductReserve2Type     ProductReserve2;
    ///单笔输入报单时，由程序自动填写；批量委托单，该值表示RequestID，需客户填写
TXeleFtdcProductReserve3Type    ProductReserve3;
///本地报单编号
     TXeleFtdcOrderLocalNoType   OrderLocalNo;
///报单价格
  TXeleFtdcPriceType             LimitPrice;
///合约代码
TXeleFtdcInstruIDType           InstrumentID;
```

```
///报单数量
TXeleFtdcVolumeTotalOriginalType  VolumeTotalOriginal;
///输入报单类型
    TXeleFtdcInsertType         InsertType;
///最小成交数量
    TXeleFtdcMinVolumeType      MinVolume;
    ///前置信息;
    TXeleFtdcExchangeFrontEnumType ExchangeFront;
    ///未用字段
     char_unused_1[2];
};
```

必填字段说明：

- ClientIndex，客户编号，登录响应中返回；
- ClientToken，客户令牌，登录响应中返回；
- OrderLocalNo，客户报单编号号，形如000001；
- LimitPrice，价格，形如 3500.00；
- InstrumentID，合约代码，形如“IF1109”；
- VolumeTotalOriginal，数量，例如5表示5手；
- MinVolume，最小成交数量，例如3表示3手；
- ExchangeFront，指定前置编号，具体值如下：
- 0:不指定前置；
- 1~9:保留；
- 10~25:表示指定某个前置；当指定前置时，字段值为实际前置编号X+10，例如
- 如果指定前置3，则该值配置为3+10=13; 在报单响应里面，返回的是实际前置值
- X，比如返回的是3；
- InsertType，报单类型， TxeleFtdcInsertType，是一个报单录入类型；为了统一各个交易所报单类型，大商所的枚举值9和10、12和13等都是对应的FAK，对于大商所来讲含义是一样的；

将买卖方向，开平、价格类型，有效期类型都放在一起生成枚举型，具体枚举值含义如下：

ComHedgeFlag 投机套保标志	OrderPriceType 订单价格类型	TimeCondition 时间条件	ComOffsetFlag	Direction 买卖方向	Volume Condition 数量条件	Value 枚举值
只支持投机	只支持限价	当日有效	开仓	买	任何数量	1
只支持投机	只支持限价	当日有效	开仓	卖	任何数量	2
只支持投机	只支持限价	当日有效	平仓	买	任何数量	3
只支持投机	只支持限价	当日有效	平仓	卖	任何数量	4
只支持投机	只支持限价	当日有效	平今仓	买	任何数量	5
只支持投机	只支持限价	当日有效	平今仓	卖	任何数量	6
只支持投机	只支持限价	当日有效	平昨仓	买	任何数量	7
只支持投机	只支持限价	当日有效	平昨仓	卖	任何数量	8
只支持投机	只支持限价	立即完成	开仓	买	任何数量	9 FAK（大商）
只支持投机	只支持限价	立即完成	开仓	买	最小数量	10 FAK（大商）
只支持投机	只支持限价	立即完成	开仓	买	全部数量	11 FOK（大商）
只支持投机	只支持限价	立即完成	开仓	卖	任何数量	12 FAK（大商）
只支持投机	只支持限价	立即完成	开仓	卖	最小数量	13 FAK（大商）
只支持投机	只支持限价	立即完成	开仓	卖	全部数量	14 FOK（大商）
只支持投机	只支持限价	立即完成	平仓	买	任何数量	15 FAK（大商）
只支持投机	只支持限价	立即完成	平仓	买	最小数量	16 FAK（大商）
只支持投机	只支持限价	立即完成	平仓	买	全部数量	17 FOK（大商）
只支持投机	只支持限价	立即完成	平仓	卖	任何数量	18 FAK（大商）
只支持投机	只支持限价	立即完成	平仓	卖	最小数量	19 FAK（大商）
只支持投机	只支持限价	立即完成	平仓	卖	全部数量	20 FOK（大商）
只支持投机	只支持限价	立即完成	平今仓	买	任何数量	21 FAK（大商）
只支持投机	只支持限价	立即完成	平今仓	买	最小数量	22 FAK（大商）

ComHedgeFlag 投机套保标志	OrderPriceType 订单价格类型	TimeCondition 时间条件	ComOffset Flag	Direction 买卖方向	Volume Condition 数量条件	Value 枚举值
只支持投机	只支持限价	立即完成	平今仓	买	全部数量	23 FOK（大商）
只支持投机	只支持限价	立即完成	平今仓	卖	任何数量	24 FAK（大商）
只支持投机	只支持限价	立即完成	平今仓	卖	最小数量	25 FAK（大商）
只支持投机	只支持限价	立即完成	平今仓	卖	全部数量	26 FOK（大商）
只支持投机	只支持限价	立即完成	平昨仓	买	任何数量	27 FAK（大商）
只支持投机	只支持限价	立即完成	平昨仓	买	最小数量	28 FAK（大商）
只支持投机	只支持限价	立即完成	平昨仓	买	全部数量	29 FOK（大商）
只支持投机	只支持限价	立即完成	平昨仓	卖	任何数量	30 FAK（大商）
只支持投机	只支持限价	立即完成	平昨仓	卖	最小数量	31 FAK（大商）
只支持投机	只支持限价	立即完成	平昨仓	卖	全部数量	32 FOK（大商）

返回值：0表示成功， 其他值异常；
说明：

- 当客户成功登陆柜台后，使用ReqOrderInsert，可以进行报单录入工作。
- 无论报单录入是否通过风控，都会返回一个OnRspOrderInsert回报；如果未通过风控，则还会返回OnErrRtnOrderInsert回报；
- 用户可以通过交查询接口(ReqQryExchangeFront/OnRspQryExchangeFront)到可用交易所前置描述符列表，直接取列表中的交易所前置值（例如13）。往交易所前置13发单可这样填写：

```
pInputOrder->ExchangeFront = 13;
```

ReqOrderAction方法

功能：报单操作请求，包括报单的撤销、报单的挂起、报单的激活、报单的修改。
函数原形：

```
int ReqOrderAction(
CXeleFairOrderActionField   *pOrderAction,
 int    nRequestID)
```

参数：

- pOrderAction：入参，报单操作域，指向报单操作结构的地址。
- nRequestID：入参，请求ID；
  报单操作请求结构体CXeleFairOrderActionField：

```
struct CXeleFairOrderActionField {
///单笔报单操作时，由程序自动填写；批量委托单时，该值需客户填写为0x67
 TXeleFtdcProductReserve1Type   ProductReserve1;
///客户编号
TXeleFtdcClientIndexType        ClientIndex;
///客户令牌
TXeleFtdcClientTokenType        ClientToken;
///产品保留字段2，由程序自动填写
TXeleFtdcProductReserve2Type     ProductReserve2;
///单笔报单操作时，由程序自动填写；批量委托单，该值表示RequestID，需客户填写
TXeleFtdcProductReserve3Type     ProductReserve3;
///本地报单操作编号
TXeleFtdcActionLocalNoType  ActionLocalNo;
///被撤单柜台编码
TXeleFtdcOrderSysNoType     OrderSysNo;
///报单操作标志(预留扩展字段)
TXeleFtdcActionFlagType         ActionFlag;
///未用字段
char    _unused_1[27]
};
```

必填字段说明：：

1. ClientIdnex，用户编号；登录响应中返回；
   2.Token，用户令牌，登录响应中返回；
   3.OrderSysNo，用户报单时系统生成的ID；
   4.ActionLocalNo，用户撤单本地ID；
   5.ActionFlag，‘0’：删除‘1’：挂起‘2’：激活；
   6.RequestID：请求ID
   返回值： 0表示成功， 其他值异常；
   说明：报单操作请求如果未通过风控，会返回OnErrRtnOrderAction回报信息。

ReqOrderInsertActionV方法

功能：批量委托单请求
函数原形：

```
int ReqOrderInsertActionV (
CXeleFairOrderFieldV        *pOrderV,
int     OrderCount)
```

参数：

- pOrderV：入参，指向批量委托单结构体数组首地址。批量委托单，每笔委托单都是独立的，且结构体大小都相同。
- OrderCount：入参，批量委托单笔数，批量委托单最大支持16笔；
  批量委托单请求结构体 CxeleFairOrderFieldV如下：

```
struct CXeleFairOrderFieldV {
union FairOrderField {
///报单输入结构体
struct CXeleFairInputOrderField         input;
///报单操作结构体
struct CXeleFairOrderActionField     action;
} order;
};
```

返回值： 0表示成功， 其他值异常；
说明：

- 批量委托单请求同时支持报单和报单操作，最多支持16笔，如果超过16笔，会自动截取前16笔委托单，余下的会直接丢弃；
- 批量委托单中的结构体和报单录入以及报单操作的结构体相同，字段含义和报单录入以及报单操作一样。**但是批量委托单结构体中的部分产品预留字段需要用户按照要求填写:**
  在批量委托单的结构体CXeleFairInputOrderField 中：
  ProductReserve1 需要固定填写为0x65;
  ProductReserve3 表示RequestID， 需要客户自己填写维护；
  在批量委托单的结构体 CXeleFairInputOrderField 中：
  ProductReserve1 需要固定填写为0x67;
  ProductReserve3 表示RequestID， 需要客户自己填写维护；
- 批量委托单的回报是分别按照单笔回报处理的。

示例代码： 批量委托单接口使用：

```
 /* 示例表示的是2笔批量委托单，第1笔是报单录入，第2笔是报单操作；其他多笔类似以下赋值处理*/
CXeleFairOrderFieldV order_action[16];
memset(order_action, 0, sizeof(order_action));
//第1笔是报单录入
order_action[0].order.input.ProductReserve1 = 0x65;    //报单录入时，需要客户固定填写为0x65
order_action[0].order.input.ProductReserve3 = 1;         //表示该笔委托单的RequestID，需要客户自己填写维护
order_action[0].order.input.OrderLocalNo = 1;
order_action[0].order.input.LimitPrice = 99.4;
order_action[0].order.input.ClientIndex = 1;
order_action[0].order.input.ClientToken = 0x6578;
strcpy(order_action[0].order.input.InstrumentID, "au2104");
order_action[0].order.input.VolumeTotalOriginal = 10;
order_action[0].order.input.InsertType = 0x2;
order_action[0].order.input.MinVolume = 25;
order_action[0].order.input.ExchangeFront = 0;
//第2笔是报单操作
order_action[1].order.action.ProductReserve1 = 0x67; //报单操作时，需要客户固定填写为0x67
order_action[1].order.action.ClientIndex = 2;               //表示该笔委托单的ClientIndex，需要客户自己填写维护
order_action[1].order.action.ClientToken = 0x6578;
order_action[1].order.action.ProductReserve3 = 2;
order_action[1].order.action.ActionFlag = '0';
order_action[1].order.action.OrderSysNo = 2;
//其他笔依次类推
……
// 发送批量委托单请求
m_pTraderApi->ReqOrderInsertActionV(order_action, 2);
```

ReqQryClientAccount方法

功能：客户资金查询请求
函数原形：

```
int ReqQryClientAccount(
CXeleFtdcQryClientAccountField  *pQryClientAccount,
 int    nRequestID)
```

参数：

- pQryClientAccount：入参，客户资金查询域，指向会员资金查询结构的地址。
- nRequestID：入参，请求ID。
  资金查询结构体CxeleFtdcQryClientAccountField如下：

```
struct CXeleFtdcQryClientAccountField {
///资金帐号
TXeleFtdcAccountIDType   AccountID;
};
```

返回值： 0表示成功， 其他值异常；
说明：如果某用户绑定了多个帐户，那么需多次使用ReqQryClientAccount对指定账户进行查询，在账户资金响应事件回报OnRspQryClientAccount中获取相应信息。

ReqQryClientPosition方法

功能：客户持仓查询请求
函数原形：

```
int ReqQryClientPosition(
CXeleFtdcQryClientPositionField     *pQryClientPosition,
 int    nRequestID)
```

参数：

- pQryClientPosition：入参，客户持仓查询域，指向客户持仓查询结构的地址。
- nRequestID：入参，请求ID。
  客户持仓查询结构体CxeleFtdcQryClientPositionField如下：

```
struct CXeleFtdcQryClientPositionField {
///资金帐号
TXeleFtdcAccountIDType  AccountID;
///合约代码
TXeleFtdcInstrumentIDType    InstrumentID;
///交易所描述符
TXeleFtdcExchangeDescriptorType     ExchangeDescriptor;
};
```

返回值： 0表示成功， 其他值异常；
说明：

- 客户端登陆柜台系统成功后，用户可以查询账户的仓位信息，在客户持仓查询事件回报OnRspQryClientPosition中获取相应信息。

- AccountID必填、InstrumentID选填；
- ExchangeDescriptor：交易所描述符和查询类型，1字节整型（非字符类型）具体含义如下：bit0-bit3表示交易所描述符，bit4-bit5表示查询类型，bit6-bit7预留。

交易所描述符	含义
0	无效的交易所描述符
1	上海期货交易所
2	上海国际能源交易中心股份有限公司
3	大连商品交易所
4	郑州商品交易所
5	中国金融交易所

查询类型	含义
0	单腿合约持仓
1	组合合约持仓（当前仅大商支持）
2	合约持仓

例如： 大商所的ExchangeDescriptor字段的值部分枚举如下：

ExchangeDescriptor	含义
0x03	查询大商所单腿合约持仓
0x13	查询大商所组合合约持仓
0x23	查询大商所合约持仓

InstrumentID：合约名称，字符串类型；

InstrumentID	含义
不填	表示查询指定ExchangeDescriptor值下的所有合约持仓；
单腿合约名称（如"bb2100"）	表示查询指定ExchangeDescriptor值下的所有单腿合约持仓；
组合合约名称 (如"a2002,-a2003")	表示查询指定ExchangeDescriptor值下的所有组合合约持仓；
单腿品种名称（如"bb"）	表示查询指定ExchangeDescriptor值下的所有该品种的单腿合约持仓；
组合第一个合约名称（如"a2002"）	表示查询指定ExchangeDescriptor值下的所有包含该合约的组合合约持仓（如"a2002,-a2003"、"-a2002,a2001"，不包括" -a2009,a2002"）
组合第一个品种名称（如"a"）	表示查询指定ExchangeDescriptor值下的所有包含该品种的组合合约持仓（如"a2002,-b2003"、"a2009,a2011"）

- 合约名称需要和ExchangeDescriptor类型匹配，如果不匹配则查询结果为空；
- 每个查询结果都是通过单独回报返回，当上述查询有多个结果时，则会有多个查询回报。

ReqQryInstrument方法

功能：合约查询
函数原形：

```
int ReqQryInstrument(
CXeleFtdcQryInstrumentField         *pQryInstrument,
int     nRequestID)
```

参数：

- QryInstrument：入参，合约查询域，指向合约查询结构的地址。
- nRequestID：入参，请求ID。
  合约查询结构体CxeleFtdcQryInstrumentField如下：

```
struct CXeleFtdcQryInstrumentField {
    ///产品代码
     TXeleFtdcProductIDType  ProductID;
    ///合约代码
    TXeleFtdcInstrumentIDType   InstrumentID;
};
```

返回值： 0表示成功， 其他值异常；
说明：

- 当客户端登陆柜台系统后，用户可以使用ReqQryInstrument以获取合约列表信息，在合约查询回报OnRspQryInstrument中获取相应信息。
- 只填写InstrumentID，查该合约信息；只填写ProductID，查该品种所有合约；都不填写查询所有合约；
- 每个查询结果都是通过单独回报返回，当上述查询有多个结果时，则会有多个查询回报。

ReqQryInstrumentMarginRate方法

功能：合约保证金率查询请求
函数原形：

```
int ReqQryInstrumentMarginRate(
CxeleFtdcQryInstrumentMarginRateField    *pQryInstrumentMarginRate,
int     nRequestID)
```

参数：

- pQryInstrumentMarginRate：入参，合约保证金率查询域，指向合约保证金率查询结构的地址。
- nRequestID：入参，请求ID。
  合约保证金率查询结构体CxeleFtdcQryInstrumentMarginRateField如下：

```
struct CXeleFtdcQryInstrumentMarginRateField {
///合约代码（选填）
TXeleFtdcInstrumentIDType   InstrumentID;
///资金帐号（必填）
 TXeleFtdcAccountIDType AccountID;
///投机套保标志（必填，目前仅支持1）
 TXeleFtdcHedgeFlagType HedgeFlag;
};
```

返回值： 0表示成功， 其他值异常；
说明：

- 当客户端登陆柜台系统后，用户可以使用ReqQryInstrumentMarginRate进行合约保证金率查询，在合约保证金率查询回报OnRspQryInstrumentMarginRate中获取相应信息。
- 查询合约保证金率支持批量查询，AccountID客户代码、HddgeFlag投机套保标志填写正确，InstrumentID合约代码不填为批量查询所有合约保证金率，如果AccountID填写错误查询结果为空，有错误回报信息；
- 每个查询结果都是通过单独回报返回，当上述查询有多个结果时，则会有多个查询回报。

ReqQryInstrumentCommissionRate方法

功能：合约手续费率查询请求
函数原形：

```
int ReqQryInstrumentCommissionRate(
CXeleFtdcQryInstrumentCommissionRateField     *pQryInstrumentCommissionRate,
                int     nRequestID)
```

参数：

- pQryInstrumentCommissionRate：入参，合约手续费率域，指向合约手续费率结构地址
- nRequestID：入参，请求ID。
  合约手续费率结构体CxeleFtdcQryInstrumentCommissionRateField如下：

```
struct CXeleFtdcQryInstrumentCommissionRateField {
    ///合约代码（选填）
TXeleFtdcInstrumentIDType    InstrumentID;
///资金帐号（必填）
TXeleFtdcAccountIDType   AccountID;
};
```

返回值： 0表示成功， 其他值异常；
说明：

- 当客户端登陆柜台系统后，用户可以使用ReqQryInstrumentCommissionRate进行合约手续费率查询，在合约手续费率查询回报OnRspQryInstrumentCommissionRate中获取相应信息。
- 查询合约手续费率支持批量查询，AccountID产品代码填写正确，InstrumentID合约代码不填为批量查询所有合约，如果AccountID与InstrumentID不匹配则查询结果为空，并有错误回报信息；
- 每个查询结果都是通过单独回报返回，当上述查询有多个结果时，则会有多个查询回报。

ReqQryOrder方法

功能：报单查询
函数原形：

```
int ReqQryOrder(
CXeleFtdcQryOrderField  *pQryOrder,
int     nRequestID)
```

参数：

- pQryOrderField：入参，报单查询请求域，指向报单查询结构的地址。
- nRequestID：入参，请求ID。
  报单查询结构体CxeleFtdcQryOrderField如下：

```
struct CXeleFtdcQryOrderField {
///资金帐号（必填）
TXeleFtdcAccountIDType   AccountID;
///合约代码（选填，不填时查所有报单）
TXeleFtdcInstrumentIDType   InstrumentID;
///报单编号（选填）
TXeleFtdcOrderSysIDType     OrderSysID;
///开始时间（选填）
TXeleFtdcTimeType       TimeStart;
///结束时间（选填）
TXeleFtdcTimeType TimeEnd;
///交易所描述符（选填，不填或填0时查询所有支持交易所。且该字段表示1字节整型非字符型；）
TXeleFtdcExchangeDescriptorType     ExchangeDescriptor;
};
```

返回值： 0表示成功， 其他值异常；
说明：当客户端登陆柜台系统后，用户可以使用ReqQryOrder进行合约手续费率查询，在报单查询回报OnRtnOrder中获取相应信息。

ReqQryTrade方法

功能：成交查询
函数原形：

```
int ReqQryTrade(
CXeleFtdcQryTradeField  *pQryTrade,
int     nRequestID)
```

- pQryOrderField：入参，成交查询请求域，指向成交查询结构的地址。
- nRequestID：入参，请求ID。
  成交查询结构体CxeleFtdcQryTradeField如下：

```
struct CXeleFtdcQryTradeField {
    ///资金帐号（必填）
TXeleFtdcAccountIDType  AccountID;
///合约代码（选填，不填时查询所有成交报单）
TXeleFtdcInstrumentIDType   InstrumentID;
///成交编号（选填）
TXeleFtdcTradeIDType        TradeID;
```

```
///开始时间（选填）
TXeleFtdcTimeType        TimeStart;
///结束时间（选填）
TXeleFtdcTimeType       TimeEnd;
///交易所描述符（选填，不填或填0时查询所有支持交易所。且字段表示1字节整型非字符型；）
TXeleFtdcExchangeDescriptorType     ExchangeDescriptor;
};
```

返回值： 0表示成功， 其他值异常；
说明：当客户端登陆柜台系统后，用户可以使用ReqQryTrade进行成交报单查询，成交查询回报OnRspQryTrade中获取相应信息。一个账户下可以有多个成交，因此需多次从事件中获取。每笔交易订单可以有多个成交记录，但是成交ID是唯一。

ReqQryExchangeFront方法

功能：交易所前置查询
函数原形：

```
int ReqQryExchangeFront(int nRequestID)
```

参数：

- nRequestID：入参，请求ID
  返回值： 0表示成功， 其他值异常；
  说明：当客户端登陆柜台系统后，用户可以使用ReqQryExchangeFront进行交易所前置查询，在交易所查询回报OnRspQryExchangeFront中获取相应信息。

ReqQryInstrumentPrice方法

功能：合约涨跌停价查询
函数原形：

```
int ReqQryInstrumentPrice
(CXeleFtdcQryInstrumentField *pQryInstrumentPrice,
int     nRequestID)
```

参数：

- pQryInstrumentPrice：入参，合约查询域，指向合约查询结构的地址
- RequestID：入参，请求ID。
  合约查询结构体CxeleFtdcQryInstrumentField如下：

```
struct CXeleFtdcQryInstrumentField {
TXeleFtdcProductIDType   ProductID; ///产品代码
TXeleFtdcInstrumentIDType   InstrumentID; ///合约代码
};
```

返回值： 0表示成功， 其他值异常；
说明：

- 当客户端登陆柜台系统后，用户可以使用ReqQryInstrumentPrice进行合约涨跌停价查询，在合约涨跌停价回报OnRspInstrumentPrice中获取相应信息。
- 查询合约涨跌停价支持批量查询，ProductID产品代码选填，不填视为查询所有品种、InstrumentID合约代码不填为批量查询所有合约，如果ProductID与InstrumentID不匹配则查询结

果为空，并有错误回报信息；
- 每个查询结果都是通过单独回报返回，当上述查询有多个结果时，则会有多个查询回报。

ReqQryInstrumentIndex方法

功能：合约序号查询
函数原形：

```
int ReqQryInstrumentPrice
(CXeleFtdcQryInstrumentIndexField * pQryInstrumentIndex,
int nRequestID)
```

参数：

- pQryInstrumentIndex：入参，合约查询域，指向合约查询结构的地址
- nRequestID：入参，请求ID。
  合约查询结构体CXeleFtdcQryInstrumentIndexField如下：

```
struct CXeleFtdcQryInstrumentIndexField{
TXeleFtdcProductIDType   ProductID; ///产品代码
TXeleFtdcInstrumentIDType   InstrumentID; ///合约代码
char Rsv[24];///预留字段
};
```

返回值： 0表示成功， 其他值异常；
说明：

- 当客户端登陆柜台系统后，用户可以使用ReqQryInstrumentIndex进行合约序号查询，在合约序号可以在OnRspQryInstrumentIndex中获取相应信息。
- 查询合约序号支持批量查询，ProductID产品代码选填，不填视为查询所有品种、InstrumentID合约代码不填为批量查询所有合约，如果ProductID与InstrumentID不匹配则查询结果为空，并有错误回报信息；
- 每个查询结果都是通过单独回报返回，当上述查询有多个结果时，则会有多个查询回报。
- 该接口是异步查询合约序号接口，同时还提供同步合约序号查询接口GetInstrumentIndex， 详见5.3.19章节；

GetInstrumentIndex方法

功能：合约涨跌停价查询
函数原形：

```
uint32_t GetInstrumentIndex
(const char *instrumentID)
```

参数：

- **instrumentID**：入参，合约编号字符串

返回值： **(uint32_t)(-1)表示无效的序号值**，其他表示对应合约的序号值。

GetClientToken方法

功能：获取客户令牌接口
函数原形：

```
uint32_t GetClientToken
(uint8_t nClientIndex)
```

参数：

- **nClientIndex**：入参，客户编号
  返回值： **0****表示无效的令牌值**，其他表示对应客户号的令牌值。

Spi连接管理类接口

OnFrontConnected方法

功能：当客户端与柜台系统的查询数据中心和交易数据中心建立链接成功后，该方法被调用。该连接是由API自动建立的。
函数原形：

```
void OnFrontConnected()
```

参数：无；
返回值：无；
说明：

- OnFrontConnected只有在API和柜台的所有链接都成功后才被调用；
- OnFrontConnected被调用仅说明客户端和柜台的连接成功，客户端必须自行登录，才能进行后续的业务操作。登录失败不会回调该方法。

OnFrontDisconnected方法

功能：当客户端与柜台系统的某条链路断开时，该方法都会被调用。当发生这个情况后，API会自动重新连接，客户端可不做处理。
函数原形：

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

参数：

- nReason： 错误代码：低16位[15:0]为操作系统错误号，高16位[31:16]为API内部错误编号；
  返回值： 无；

OnPackageStart方法

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

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

参数：

- nTopic：主题代码（如私有流、公共流、行情流等）。
- nSequenceNo：报文序号；
  返回值： 无；

OnPackageEnd方法

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

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

参数：

- nTopic：主题代码（如私有流、公共流、行情流等）。
- SequenceNo：报文序号；
  返回值： 无；

Spi业务应答接口

SPI业务接口的通用说明：
1.**业务接口中的RequestID**：应答接口中的该值是柜台系统直接返回客户端在发送API请求中的requestid；
2.**接口返回值：**如果接口带有返回值，0表示函数处理正常，其他值表示处理异常；
3.**bIsLast：** 表示报文结束标记，当某个响应含有多个报文时，该值表示响应报文的结束；

OnRspError方法

功能：错误应答
函数原形：

```
void OnRspError(
CXeleFtdcRspInfoField *pRspInfo,
int  nRequestID,
bool bIsLast)
```

参数：

- pRspInfo：响应信息域，返回用户响应信息的地址。
- nRequestID：请求ID
- bIsLast：指示该次返回是否为针对nRequestID的最后一次返回。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
TXeleFtdcErrorIDType    ErrorID; ///错误代码
TXeleFtdcErrorMsgType   ErrorMsg; ///错误信息
};
```

返回值： 无；
说明： 具体的错误代码和错误信息见附录错误代码表；

OnRspUserLogin方法

功能：用户登录应答
函数原形：

```
void OnRspUserLogin(
CXeleFtdcRspUserLoginField *pRspUserLogin,
CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- pRspUserLogin：用户登录应答域，返回用户登录响应信息的地址。
  用户登录响应信息结构体 CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspUserLoginField {
///交易日
TXeleFtdcDateType       TradingDay;
///登录成功时间
TXeleFtdcTimeType        LoginTime;
///最大本地报单号（未使用）
TXeleFtdcOrderLocalIDType    MaxOrderLocalID;
///交易用户代码
TXeleFtdcAccountIDType  AccountID;
///当前登录的交易所数目
TXeleFtdcExchangeNumType     ExchangeNum;
///当前登录交易所的交易用户编号;
TXeleFtdcClientIndexType     ClientIndex[XELE_EXCHANGE_LOGIN_NUM];
///当前登录交易所的交易用户令牌;
TXeleFtdcClientTokenType    Token[XELE_EXCHANGE_LOGIN_NUM];
///会员代码
TXeleFtdcParticipantIDType   ParticipantID;
///交易系统名称
TXeleFtdcTradingSystemNameType  TradingSystemName;
///数据中心代码 （未使用）
TXeleFtdcDataCenterIDType   DataCenterID;
///会员私有流当前长度（未使用）
TXeleFtdcSequenceNoType     PrivateFlowSize;
///交易员私有流当前长度（未使用）
TXeleFtdcSequenceNoType     UserFlowSize;
};
```

- pRspInfo:响应信息域，返回用户响应信息的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
TXeleFtdcErrorIDType        ErrorID; ///错误代码
TXeleFtdcErrorMsgType       ErrorMsg; ///错误信息
};
```

- nRequestID：请求ID
- bIsLast： 此次请求的响应的最后一次回调标志
  返回值： 无；

OnRspUserPasswordUpdate方法

功能：修改用户密码应答
函数原形：

```
void OnRspUserPasswordUpdate(
            CXeleFtdcUserPasswordUpdateField  *pUserPasswordUpdate,
            CXeleFtdcRspInfoField *pRspInfo,
            int nRequestID,
             bool bIsLast)
```

参数：

- **pUserPasswordUpdate**：修改用户密码应答域，返回修改用户密码响应信息的地址。
  修改用户密码响应信息结构体CXeleFtdcUserPasswordUpdateField如下：

```
struct CXeleFtdcUserPasswordUpdateField {
  ///交易用户代码
  TXeleFtdcAccountIDType AccountID;
  ///未用字段
  char Padding_Three[3];
  ///会员代码
  TXeleFtdcParticipantIDType ParticipantID;
  ///旧密码
  TXeleFtdcPasswordType OldPassword;
  ///新密码
  TXeleFtdcPasswordType NewPassword;
};
```

- pRspInfo:响应信息域，返回用户响应信息的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
TXeleFtdcErrorIDType        ErrorID; ///错误代码
TXeleFtdcErrorMsgType       ErrorMsg; ///错误信息
};
```

- nRequestID：请求ID
- bIsLast： 此次请求的响应的最后一次回调标志
  返回值： 无；

OnRspOrderInsert方法

功能：报单录入应答
函数原形：

```
void OnRspOrderInsert(
CXeleFtdcInputOrderField *pInputOrder,
CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- pInputOrder：报单录入域，指向报单录入结构的地址。
  报单录入响应结构体CxeleFtdcInputOrderField如下：

```
struct CXeleFtdcInputOrderField {
///柜台系统报单编号
TXeleFtdcOrderSystemNoType  OrderSystemNo;
///会员代码（未使用）
TXeleFtdcParticipantIDType   ParticipantID;
///客户代码（未使用）
TXeleFtdcClientIDType   ClientID;
///交易用户代码(未使用)
TXeleFtdcUserIDType     UserID;
///合约代码(未使用)
TXeleFtdcInstrumentIDType   InstrumentID;
///报单价格条件(未使用)
TXeleFtdcOrderPriceTypeType     OrderPriceType;
///买卖方向(未使用)
TXeleFtdcDirectionType  Direction;
///组合开平标志(未使用)
TXeleFtdcCombOffsetFlagType     CombOffsetFlag;
///组合投机套保标志(未使用)
TXeleFtdcCombHedgeFlagType  CombHedgeFlag;
///价格(未使用)
TXeleFtdcPriceType      LimitPrice;
///数量(未使用)
TXeleFtdcVolumeType             VolumeTotalOriginal;
///有效期类型(未使用)
TXeleFtdcTimeConditionType      TimeCondition;
///GTD日期(未使用)
TXeleFtdcDateType           GTDDate;
///成交量类型(未使用)
TXeleFtdcVolumeConditionType    VolumeCondition;
///最小成交量(未使用)
TXeleFtdcVolumeType         MinVolume;
///触发条件(未使用)
TXeleFtdcContingentConditionType    ContingentCondition;
///止损价(未使用)
TXeleFtdcPriceType          StopPrice;
///强平原因(未使用)
TXeleFtdcForceCloseReasonType   ForceCloseReason;
///客户端本地报单编号
TXeleFtdcOrderLocalNoType       OrderLocalNo;
///自动挂起标志(未使用)
TXeleFtdcBoolType           IsAutoSuspend;
///交易所报单编号, RspOrderInsert时有意义
TXeleFtdcExchangeOrderSysIDType     ExchangeOrderSysID;
///用户定义交易所前置发单, 可选.
TXeleFtdcExchangeFrontType      ExchangeFront;
};
```

- nRequestID：请求ID
- bIsLast：此次请求的响应的最后一次回调标志
  返回值： 无；

OnRspOrderAction方法

功能：报单操作应答。报单操作包括报单的撤销、报单的挂起、报单的激活、报单的修改。
函数原形：

```
void OnRspOrderAction(
CXeleFtdcOrderActionField *pOrderAction,
CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- pOrderAction：报单操作域，指向报单操作结构的地址，包含了提交报单操作的输入数据，和交易系统返回的报单编号。
  报单操作响应结构体CxeleFtdcOrderActionField如下：

```
struct CXeleFtdcOrderActionField {
///柜台系统报单编号
TXeleFtdcOrderSystemNoType OrderSystemNo;
///未用字段
char_unused_1[9];
///客户端本地报单编号
TXeleFtdcOrderLocalNoType   OrderLocalNo;
///未用字段
char _unused_2[9];
///报单操作标志
TXeleFtdcActionFlagType ActionFlag;
///会员代码(未使用)
TXeleFtdcParticipantIDType ParticipantID;
///客户代码(未使用)
TXeleFtdcClientIDTypeClientID;
///交易用户代码
TXeleFtdcUserIDType UserID;
///价格(未使用)
  TXeleFtdcPriceType LimitPrice;
///数量变化(未使用)
TXeleFtdcVolumeType VolumeChange;
///柜台系统的撤单编号
TXeleFtdcOrderLocalIDType   ActionLocalID;
///未用字段
char _unused_3[9];
///用户自己管理的撤单编号
TXeleFtdcActionLocalNoType  ActionLocalNo;
///未用字段
char _unused_4[12];
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
  TXeleFtdcErrorIDType ErrorID;  ///错误代码
   TXeleFtdcErrorMsgType ErrorMsg; ///错误信息
} ;
```

- nRequestID：请求ID
- bIsLast： 此次请求的响应的最后一次回调标志
  返回值： 无；

OnRspQryClientAccount方法

功能：客户资金查询应答
函数原形：

```
void OnRspQryClientAccount(
CXeleFtdcRspClientAccountField *pClientAccount,
CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- pClientAccount：客户资金应答域，指向客户资金应答结构的地址。
  客户资金应答结构体CxeleFtdcRspClientAccountField如下：

```
struct CXeleFtdcRspClientAccountField {
///交易日
  TXeleFtdcDateType     TradingDay;
///结算组代码(未使用)
  TXeleFtdcSettlementGroupIDType    SettlementGroupID;
///结算编号(未使用)
  TXeleFtdcSettlementIDType     SettlementID;
///上次结算准备金
  TXeleFtdcMoneyType        PreBalance;
///当前保证金总额
  TXeleFtdcMoneyType        CurrMargin;
///平仓盈亏
   TXeleFtdcMoneyType       CloseProfit;
///期权权利金收支
    TXeleFtdcMoneyType  Premium;
///入金金额
    TXeleFtdcMoneyType  Deposit;
///出金金额
  TXeleFtdcMoneyType        Withdraw;
///期货结算准备金
   TXeleFtdcMoneyType       Balance;
///可提资金
    TXeleFtdcMoneyType  Available;
///资金帐号
  TXeleFtdcAccountIDType    AccountID;
///冻结的保证金
  TXeleFtdcMoneyType        FrozenMargin;
///冻结的权利金
  TXeleFtdcMoneyType        FrozenPremium;
///基本准备金(未使用)
  TXeleFtdcMoneyType        BaseReserve;
///浮动盈亏
  TXeleFtdcMoneyType        floatProfitAndLoss;
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
  TXeleFtdcErrorIDType ErrorID;  ///错误代码
  TXeleFtdcErrorMsgType ErrorMsg; ///错误信息
};
```

- nRequestID：请求ID
- bIsLast： 此次请求的响应的最后一次回调标志
  返回值：无；
  说明： 查询结果为空时, 参数的传值, pClientAccount置空，ErrorID置0。

OnRspQryClientPosition方法

功能：客户持仓查询应答。当客户程序发出客户持仓查询指令后，交易柜台返回响应时，该方法会被调用。
参数原形：

```
void OnRspQryClientPosition(
CXeleFtdcRspClientPositionField *pRspClientPosition,
CXeleFtdcRspInfoField *pRspInfo,
 int nRequestID,
 bool bIsLast)
```

参数：

- pRspClientPosition：客户持仓应答域，指向客户持仓应答结构的地址。
  客户持仓应答结构体CxeleFtdcRspClientPositionField如下：

```
struct CXeleFtdcRspClientPositionField {
///交易日
TXeleFtdcDateType           TradingDay;
///结算组代码(未使用)
TXeleFtdcSettlementGroupIDType  SettlementGroupID;
///结算编号(未使用)
TXeleFtdcSettlementIDType       SettlementID;
///投机套保标志
TXeleFtdcHedgeFlagType      HedgeFlag;
///多头上日持仓
TXeleFtdcVolumeType             LongYdPosition;
///多头今日持仓
TXeleFtdcVolumeType         LongPosition;
///空头昨日持仓
TXeleFtdcVolumeType         ShortYdPosition;
///空头今日持仓
TXeleFtdcVolumeType         ShortPosition;
///持仓成本
TXeleFtdcMoneyType      PositionCost;
///昨日持仓成本
TXeleFtdcMoneyType      YdPositionCost;
///合约代码
TXeleFtdcInstrumentIDType    InstrumentID;
///资金帐号
TXeleFtdcAccountIDType   AccountID;
```

```
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
  TXeleFtdcErrorIDType      ErrorID;
///错误信息
   TXeleFtdcErrorMsgType    ErrorMsg;
};
```

- nRequestID：请求ID
- bIsLast： 此次请求的响应的最后一次回调标志

返回值： 无；
说明： 查询结果为空时, 参数的传值, **pRspClientPosition**置空，ErrorID置0。

OnRspQryInstrument方法

功能：合约查询应答
函数原形：

```
void OnRspQryInstrument(
CXeleFtdcRspInstrumentField *pRspInstrument,
CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- pRspInstrument：合约查询应答域，指向合约结构的地址。
  合约查询应答结构体CxeleFtdcRspInstrumentField如下：

```
struct CXeleFtdcRspInstrumentField {
///结算组代码(未使用)
  TXeleFtdcSettlementGroupIDType    SettlementGroupID;
///产品代码
  TXeleFtdcProductIDType         ProductID;
///产品组代码(未使用)
  TXeleFtdcProductGroupIDType   ProductGroupID;
///基础商品代码
  TXeleFtdcInstrumentIDType         UnderlyingInstrID;
///产品类型
  TXeleFtdcProductClassType         ProductClass;
///持仓类型
  TXeleFtdcPositionTypeType         PositionType;
///执行价(未使用)
  TXeleFtdcPriceType            StrikePrice;
///期权类型(未使用)
  TXeleFtdcOptionsTypeType      OptionsType;
///合约数量乘数
  TXeleFtdcVolumeMultipleType       VolumeMultiple;
///合约基础商品乘数
  TXeleFtdcUnderlyingMultipleType   UnderlyingMultiple;
///合约代码
```

```
  TXeleFtdcInstrumentIDType         InstrumentID;
///合约名称(未使用)
    TXeleFtdcInstrumentNameType     InstrumentName;
///交割年份
  TXeleFtdcYearType             DeliveryYear;
///交割月
  TXeleFtdcMonthType            DeliveryMonth;
///提前月份(未使用)
  TXeleFtdcAdvanceMonthType         AdvanceMonth;
///当前是否交易
  TXeleFtdcBoolType             IsTrading;
///创建日
  TXeleFtdcDateType             CreateDate;
///上市日
  TXeleFtdcDateType         OpenDate;
///到期日
  TXeleFtdcDateType             ExpireDate;
///开始交割日
  TXeleFtdcDateType             StartDelivDate;
///最后交割日
 TXeleFtdcDateType          EndDelivDate;
///挂牌基准价(未使用)
  TXeleFtdcPriceType            BasisPrice;
///市价单最大下单量
  TXeleFtdcVolumeType       MaxMarketOrderVolume;
///市价单最小下单量
  TXeleFtdcVolumeType       MinMarketOrderVolume;
///限价单最大下单量
   TXeleFtdcVolumeType      MaxLimitOrderVolume;
///限价单最小下单量
   TXeleFtdcVolumeType      MinLimitOrderVolume;
///最小变动价位
   TXeleFtdcPriceType           PriceTick;
///交割月自然人开仓(未使用)
   TXeleFtdcMonthCountType      AllowDelivPersonOpen;
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
  TXeleFtdcErrorIDType      ErrorID;
///错误信息
  TXeleFtdcErrorMsgType  ErrorMsg;
};
```

- nRequestID：请求ID
- bIsLast：此次请求的响应的最后一次回调标志
  返回值： 无；
  说明： 查询结果为空时, 参数的传值, **pRspInstrument**置空，ErrorID置0。

OnRspQryInstrumentMarginRate方法

功能：合约保证金率查询应答
函数原形：

```
void OnRspQryInstrumentMarginRate(
CXeleFtdcRspInstrumentMarginRateField *pRspInstrumentMarginRate,
CXeleFtdcRspInfoField *pRspInfo,
 int nRequestID,
bool bIsLast)
```

参数：

- pRspInstrumentMarginRate：合约保证金率域，指向合约保证金率应答结构的地址。
  合约保证金率应答结构体CxeleFtdcRspInstrumentMarginRateField如下：

```
struct CXeleFtdcRspInstrumentMarginRateField {
///合约代码
   TXeleFtdcInstrumentIDType    InstrumentID;
///资金帐号
    TXeleFtdcAccountIDType  AccountID;
///投机套保标志
  TXeleFtdcHedgeFlagType    HedgeFlag;
///多头保证金率
   TXeleFtdcRatioType       LongMarginRatioByMoney;
///多头保证金费
  TXeleFtdcRatioType        LongMarginRatioByVolume;
///空头保证金率
  TXeleFtdcRatioType        ShortMarginRatioByMoney;
///空头保证金费
  TXeleFtdcRatioType        ShortMarginRatioByVolume;
///是否相对交易所收取
  TXeleFtdcBoolType         IsRelative;
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
  TXeleFtdcErrorIDType rrorID;
///错误信息
   TXeleFtdcErrorMsgType ErrorMsg;
};
```

- nRequestID：请求ID
- bIsLast：此次请求的响应的最后一次回调标志
  返回值： 无；

OnRspQryInstrumentCommissionRate方法

功能：合约手续费率查询应答
函数原形：

```
void OnRspQryInstrumentCommissionRate(
CXeleFtdcRspInstrumentCommissionRateField *pRspInstrumentCommissionRate,
 CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- pRspInstrumentCommissionRate:合约手续费率域，指向合约手续费率应答结构地址。
  合约手续费率应答结构体CxeleFtdcRspInstrumentCommissionRateField如下：

```
struct CXeleFtdcRspInstrumentCommissionRateField {
///合约代码
  TXeleFtdcInstrumentIDType     InstrumentID;
///资金帐号
  TXeleFtdcAccountIDType    AccountID;
///开仓手续费率
  TXeleFtdcRatioType        OpenRatioByMoney;
///开仓手续费
  TXeleFtdcRatioType        OpenRatioByVolume;
///平仓手续费率
  TXeleFtdcRatioType        CloseRatioByMoney;
///平仓手续费
  TXeleFtdcRatioType        CloseRatioByVolume;
///平今手续费率
   TXeleFtdcRatioType       CloseTodayRatioByMoney;
///平今手续费
    TXeleFtdcRatioType      CloseTodayRatioByVolume;
};
```

- pRspInfo:响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoFieldt如下：

```
struct CXeleFtdcRspInfoField {
  TXeleFtdcErrorIDType      ErrorID;  ///错误代码
   TXeleFtdcErrorMsgType    ErrorMsg; ///错误信息
} ;
```

- RequestID　：请求ID
- bIsLast：此次请求的响应的最后一次回调标志
  返回值： 无；
  说明： 查询结果为空时参数传值**pRspInstrumentCommissionRate**置空，ErrorID置0。

OnRspQryTrade方法

功能：成交单查询应答
函数原形：

```
void OnRspQryTrade(
CXeleFtdcTradeField *pTradeField,
 CXeleFtdcRspInfoField *pRspInfo,
 int    nRequestID,
bool    bIsLast)
```

- pTradeField：成交信息域，指向成交信息结构的地址。
  成交信息结构体CxeleFtdcTradeField如下：

```
struct CXeleFtdcTradeField {
///交易日
   TXeleFtdcDateType            TradingDay;
///结算组代码
   TXeleFtdcSettlementGroupIDType   SettlementGroupID;
///结算编号
   TXeleFtdcSettlementIDType        SettlementID;
///成交编号
   TXeleFtdcTradeIDType         TradeID;
///买卖方向
   TXeleFtdcDirectionType       Direction;
///报单编号
TXeleFtdcOrderSystemNoType  OrderSystemNo;
///会员代码
  TXeleFtdcParticipantIDType        ParticipantID;
///客户代码
  TXeleFtdcClientIDType     ClientID;
///交易角色(未使用)
  TXeleFtdcTradingRoleType      TradingRole;
///资金帐号(未使用)
  TXeleFtdcAccountIDType        AccountID;
///合约代码
  TXeleFtdcInstrumentIDType     InstrumentID;
///开平标志
  TXeleFtdcOffsetFlagType       OffsetFlag;
///投机套保标志
  TXeleFtdcHedgeFlagType        HedgeFlag;
///价格
  TXeleFtdcPriceType            Price;
///数量
  TXeleFtdcVolumeType       Volume;
///成交时间
  TXeleFtdcTimeType             TradeTime;
///成交类型(未使用)
  TXeleFtdcTradeTypeType        TradeType;
///成交价来源(未使用)
  TXeleFtdcPriceSourceType      PriceSource;
///交易用户代码
  TXeleFtdcUserIDType           UserID;
///本地报单编号
  TXeleFtdcOrderLocalNoType         OrderLocalNo;
///交易所报单编号
   TXeleFtdcExchangeOrderSysIDType  ExchangeOrderSysID;
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
   TXeleFtdcErrorIDType     ErrorID;
///错误信息
   TXeleFtdcErrorMsgType    ErrorMsg;
};
```

- nRequestID：请求ID
- bIsLast：此次请求的响应的最后一次回调标志
  返回值： 无。
  说明： 查询结果为空时, 参数的传值, pTradeField置空，ErrorID置0。

OnRspQryOrder方法

功能：报单查询应答
函数原形：

```
void OnRspQryOrder(
CXeleFtdcOrderField *pOrderField,
 CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- pOrderField：报单信息域，指向报单信息结构的地址。
  报单信息结构体CxeleFtdcOrderField如下：

```
struct CXeleFtdcOrderField {
///交易日
  TXeleFtdcDateType             TradingDay;
///结算组代码(未使用)
  TXeleFtdcSettlementGroupIDType    SettlementGroupID;
///结算编号(未使用)
  TXeleFtdcSettlementIDType         SettlementID;
///本地报单编号
  TXeleFtdcOrderSystemNoType        OrderSystemNo;
///未用字段
char        _unused_1[9];
///会员代码
  TXeleFtdcParticipantIDType        ParticipantID;
///客户代码
  TXeleFtdcClientIDType         ClientID;
///交易用户代码
  TXeleFtdcUserIDType           UserID;
///合约代码
  TXeleFtdcInstrumentIDType         InstrumentID;
///报单价格条件
  TXeleFtdcOrderPriceTypeType       OrderPriceType;
///买卖方向
   TXeleFtdcDirectionType       Direction;
///组合开平标志
  TXeleFtdcCombOffsetFlagType   CombOffsetFlag;
///组合投机套保标志
```

```
  TXeleFtdcCombHedgeFlagType    CombHedgeFlag;
///价格
   TXeleFtdcPriceType           LimitPrice;
///数量
  TXeleFtdcVolumeType       VolumeTotalOriginal;
///有效期类型
  TXeleFtdcTimeConditionType        TimeCondition;
///GTD日期，未使用
  TXeleFtdcDateType             GTDDate;
///成交量类型
   TXeleFtdcVolumeConditionType     VolumeCondition;
///最小成交量
  TXeleFtdcVolumeType       MinVolume;
///触发条件
  TXeleFtdcContingentConditionType  ContingentCondition;
///止损价，未使用
  TXeleFtdcPriceType            StopPrice;
///强平原因
  TXeleFtdcForceCloseReasonType     ForceCloseReason;
///本地报单编号
  TXeleFtdcOrderLocalNoType         OrderLocalNo;
///未用字段
char    _unused_2[9];
///自动挂起标志
   TXeleFtdcBoolType            IsAutoSuspend;
///报单来源(未使用)
TXeleFtdcOrderSourceType        OrderSource;
///报单状态
  TXeleFtdcOrderStatusType      OrderStatus;
///报单类型(未使用)
  TXeleFtdcOrderTypeType        OrderType;
///今成交数量(未使用)
  TXeleFtdcVolumeType       VolumeTraded;
///剩余数量
  TXeleFtdcVolumeType       VolumeTotal;
///报单日期
  TXeleFtdcDateType             InsertDate;
///插入时间
  TXeleFtdcTimeType             InsertTime;
///激活时间(未使用)
  TXeleFtdcTimeType             ActiveTime;
///挂起时间(未使用)
  TXeleFtdcTimeType             SuspendTime;
///最后修改时间(未使用)
   TXeleFtdcTimeType            UpdateTime;
///撤销时间(未使用)
  TXeleFtdcTimeType             CancelTime;
///最后修改交易用户代码
  TXeleFtdcUserIDType           ActiveUserID;
///优先权(未使用)
  TXeleFtdcPriorityType             Priority;
///按时间排队的序号(未使用)
  TXeleFtdcTimeSortIDType       TimeSortID;
///交易所报单编号
  TXeleFtdcExchangeOrderSysIDType   ExchangeOrderSysID;
///未用字段
char         _unused_3[32];
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
    ///错误代码
   TXeleFtdcErrorIDType     ErrorID;
///错误信息
   TXeleFtdcErrorMsgType    ErrorMsg;
};
```

- nRequestID：请求ID
- bIsLast ：此次请求的响应的最后一次回调标志
  返回值： 无；
  说明： 查询结果为空时, 参数的传值, **pOrderField**置空，ErrorID置0。

OnRspQryExchangeFront方法

功能：交易所前置查询应答
函数原形：

```
void OnRspQryExchangeFront(
CXeleFtdcRspExchangeFrontField *pRspExchangeFront,
CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- pRspExchangeFront:交易所交易前置代码域，指向交易所前置代码查询结构的地址。
  交易所前置代码查询结构体CxeleFtdcRspExchangeFrontField如下：

```
struct CXeleFtdcRspExchangeFrontField {
///交易所标志
TXeleFtdcExchangNoType  ExchangeID;
///交易前置数量
TXeleFtdcFrontCountType FrontCount;
///交易前置代码列表
TXeleFtdcFrontListType      FrontList;
///交易前置IP地址列表
 TXeleFtdcFrontIpListType   FrontIpList;
  };
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
TXeleFtdcErrorIDType        ErrorID;
///错误信息
TXeleFtdcErrorMsgType   ErrorMsg;
};
```

- nRequestID：请求ID

- bIsLast ：此次请求的响应的最后一次回调标志
  返回值： 无；
  说明：
- 当柜台为双交易所模式时，客户端会分别收到2个交易所的前置列表信息的回报， 其中ExchangeID表示交易所ID，FrontCount是该交易所的前置数目；
- FrontList 表示的是交易所前置的编号列表，数组的下标是从0开始的，且前置编号已经是经过偏移处理了（偏移为10，比如交易所前置编号为3，则返回的是13）；
- FrontIpList表示的是交易所前置的IP列表，数组的下标是和对应的前置编号开始的；
  举例说明：
  比如当前柜台为上期和原油的双交易所模式，
  上期的前置比如为：前置2：IP为192.168.1.2；前置5：IP为192.168.1.5；
  原油的前置比如为：前置11：IP为192.168.1.11；
  则查询前置的请求会收到2个回报，分别表示上期和原油的前置列表等信息：

对应上期的前置查询回报内容	***对应原油的前置查询回报内容***
ExchangeID=1	*ExchangeID=2*
FrontCount=2	*FrontCount=1*
FrontList[0] = 12 FrontList[1] =15	*FrontList[0] = 21*
FrontIpList[2] = “192.168.1.2” FrontIpList[5] = “192.168.1.5”	*FrontIpList[11] = “192.168.1.11”*

OnRspInstrumentPrice方法

功能：合约价格查询应答
函数原形：

```
void OnRspInstrumentPrice(
CXeleFtdcRspInstrumentPriceField    *pRspInstrumentPriceField,
 CXeleFtdcRspInfoField      * pRspInfo,
int     nRequestID,
bool    bIsLast)
```

参数：

- **pInstrumentPriceField**：合约涨跌停价查询应答域，指向合约涨跌停价查询应答结构的地址。
  合约涨跌停价查询应答结构体CxeleFtdcRspInstrumentPriceField如下：

```
struct CXeleFtdcRspInstrumentPriceField {
///合约代码
  TXeleFtdcInstrumentIDType     InstrumentID;
///涨停价
  TXeleFtdcPriceType        UpperLimitPrice;
///跌停价
  TXeleFtdcPriceType        LowerLimitPrice;
///昨结算
  TXeleFtdcPriceType        PreSettlementPrice;
///预留字段
  char      _unused_1[32];
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
  TXeleFtdcErrorIDType      ErrorID;
///错误信息
   TXeleFtdcErrorMsgType    ErrorMsg;
};
```

- nRequestID：请求ID
- bIsLast：此次请求的响应的最后一次回调标志
  返回值： 无；

OnRspQryInstrumentIndex方法

功能：合约序号查询应答接口
函数原形：

```
void OnRspQryInstrumentIndex (
CXeleFtdcRspInstrumentIndexField *pRspInstrumentIndex,
 CXeleFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)
```

参数：

- **pRspInstrumentIndex**：合约序号查询应答域，指向合约序号查询应答结构的地址。
  合约序号查询应答结构体CXeleFtdcRspInstrumentIndexField如下：

```
struct CXeleFtdcRspInstrumentIndexField{
    ///品种代码
  TXeleFtdcProductIDType    ProductID;
///合约代码
  TXeleFtdcInstrumentIDType      InstrumentID;
///合约序号
  TXeleFtdcInstrumentIndexType InstrumentIndex;
///预留字段
  char      Rsv [20];
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
  TXeleFtdcErrorIDType      ErrorID;
///错误信息
   TXeleFtdcErrorMsgType    ErrorMsg;
};
```

- nRequestID：请求ID
- bIsLast：此次请求的响应的最后一次回调标志
  返回值： 无；

Spi业务回报类接口

OnRtnTrade方法

功能：成交回报，当发生成交时柜台系统会通知客户端，该方法会被调用。
参数原形：

```
void OnRtnTrade(CXeleFtdcTradeField *pTrade)
```

- pTrade ：成交回报域，指向成交回报结构的地址。注意：成交回报中的某些字段未使用，柜台系统返回为空值。
  成交回报结构体CxeleFtdcTradeField如下：

```
struct CXeleFtdcTradeField {
///交易日
  TXeleFtdcDateType             TradingDay;
///结算组代码
  TXeleFtdcSettlementGroupIDType    SettlementGroupID;
///结算编号
  TXeleFtdcSettlementIDType         SettlementID;
///成交编号
  TXeleFtdcTradeIDType      TradeID;
///买卖方向
  TXeleFtdcDirectionType        Direction;
///柜台系统报单编号
  TXeleFtdcOrderSystemNoType        OrderSystemNo;
///未用字段
  char       _unused_1[9];
///会员代码
  TXeleFtdcParticipantIDType        ParticipantID;
///客户代码
  TXeleFtdcClientIDType         ClientID;
///交易角色(未使用)
  TXeleFtdcTradingRoleType      TradingRole;
///资金帐号(未使用)
  TXeleFtdcAccountIDType        AccountID;
///合约代码
  TXeleFtdcInstrumentIDType     InstrumentID;
///开平标志
  TXeleFtdcOffsetFlagType       OffsetFlag;
///投机套保标志
  TXeleFtdcHedgeFlagType        HedgeFlag;
///价格
  TXeleFtdcPriceType            Price;
///数量
  TXeleFtdcVolumeType       Volume;
///成交时间
    TXeleFtdcTimeType           TradeTime;
///成交类型(未使用)
  TXeleFtdcTradeTypeType        TradeType;
///成交价来源(未使用)
  TXeleFtdcPriceSourceType      PriceSource;
///交易用户代码
  TXeleFtdcUserIDType           UserID;
///未用字段
char        _unused_2[9];
///客户端本地报单编号
  TXeleFtdcOrderLocalNoType         OrderLocalNo;
```

```
///交易所报单编号
  TXeleFtdcExchangeOrderSysIDType   ExchangeOrderSysID;
///未用字段
  char       _unused_3[32];
};
```

返回值： 无；

OnRtnOrder方法

功能：报单回报，当客户端进行报单录入、报单操作及其它原因（如部分成交）导致报单状态发生变化时，交柜台系统会主动通知客户端，该方法会被调用。
函数原形：

```
void OnRtnOrder(CXeleFtdcOrderField *pOrder)
```

参数：

- pOrder ：报单回报域，指向报单回报结构的地址。注意：报单回报中的某些字段未使用，柜台系统返回为空值。
  报单回报结构体CxeleFtdcOrderField如下：

```
///报单
struct CXeleFtdcOrderField {
    ///交易日
    TXeleFtdcDateType           TradingDay;
    ///结算组代码(未使用)
    TXeleFtdcSettlementGroupIDType  SettlementGroupID;
    ///结算编号(未使用)
    TXeleFtdcSettlementIDType       SettlementID;
    ///柜台系统报单编号
     TXeleFtdcOrderSystemNoType         OrderSystemNo;
    ///未用字段
     char   _unused_1[9];
    ///会员代码
  TXeleFtdcParticipantIDType        ParticipantID;
///客户代码
  TXeleFtdcClientIDType     ClientID;
///交易用户代码
  TXeleFtdcUserIDType           UserID;
///合约代码
  TXeleFtdcInstrumentIDType         InstrumentID;
///报单价格条件
  TXeleFtdcOrderPriceTypeType       OrderPriceType;
  ///买卖方向
  TXeleFtdcDirectionType         Direction;
///组合开平标志
  TXeleFtdcCombOffsetFlagType   CombOffsetFlag;
///组合投机套保标志
  TXeleFtdcCombHedgeFlagType    CombHedgeFlag;
///价格
  TXeleFtdcPriceType            LimitPrice;
///数量
  TXeleFtdcVolumeType       VolumeTotalOriginal;
///有效期类型
  TXeleFtdcTimeConditionType        TimeCondition;
```

```
///GTD日期
  TXeleFtdcDateType             GTDDate;
///成交量类型
  TXeleFtdcVolumeConditionType  VolumeCondition;
///最小成交量
  TXeleFtdcVolumeType       MinVolume;
///触发条件
  TXeleFtdcContingentConditionType  ContingentCondition;
///止损价
   TXeleFtdcPriceType           StopPrice;
 ///强平原因
  TXeleFtdcForceCloseReasonType     ForceCloseReason;
 ///客户端本地报单编号
   TXeleFtdcOrderLocalNoType        OrderLocalNo;
///未用字段
Char    _unused_2[9];
///自动挂起标志
  TXeleFtdcBoolType             IsAutoSuspend;
 ///报单来源(未使用)
  TXeleFtdcOrderSourceType      OrderSource;
///报单状态
  TXeleFtdcOrderStatusType      OrderStatus;
///报单类型(未使用)
  TXeleFtdcOrderTypeType        OrderType;
///今成交数量(未使用)
  TXeleFtdcVolumeType       VolumeTraded;
 ///剩余数量
   TXeleFtdcVolumeType      VolumeTotal;
///报单日期
    TXeleFtdcDateType           InsertDate;
///插入时间
  TXeleFtdcTimeType             InsertTime;
 ///激活时间(未使用)
    TXeleFtdcTimeType           ActiveTime;
///挂起时间(未使用)
   TXeleFtdcTimeType            SuspendTime;
///最后修改时间(未使用)
  TXeleFtdcTimeType             UpdateTime;
///撤销时间(未使用)
  TXeleFtdcTimeType             CancelTime;
///最后修改交易用户代码
TXeleFtdcUserIDType         ActiveUserID;
///优先权(未使用)
TXeleFtdcPriorityType           Priority;
///按时间排队的序号(未使用)
    TXeleFtdcTimeSortIDType     TimeSortID;
///交易所报单编号
    TXeleFtdcExchangeOrderSysIDType ExchangeOrderSysID;
///未用字段
    char    _unused_3[32];
  };
```

返回值：无；

OnRtnInsInstrument方法

功能：增加合约通知
函数原形：

```
void OnRtnInsInstrument(CXeleFtdcInstrumentField *pInstrument)
```

参数：

- pInstrument ：合约域，指向合约结构的地址。
  合约结构体CxeleFtdcInstrumentField如下：

```
struct CXeleFtdcInstrumentField {
///结算组代码
TXeleFtdcSettlementGroupIDType  SettlementGroupID;
 ///产品代码
 TXeleFtdcProductIDType         ProductID;
///产品组代码
   TXeleFtdcProductGroupIDType  ProductGroupID;
///基础商品代码
  TXeleFtdcInstrumentIDType         UnderlyingInstrID;
///产品类型
   TXeleFtdcProductClassType        ProductClass;
///持仓类型
  TXeleFtdcPositionTypeType         PositionType;
///执行价
  TXeleFtdcPriceType            StrikePrice;
///期权类型
  TXeleFtdcOptionsTypeType      OptionsType;
///合约数量乘数
  TXeleFtdcVolumeMultipleType   VolumeMultiple;
///合约基础商品乘数
  TXeleFtdcUnderlyingMultipleType   UnderlyingMultiple;
///合约代码
  TXeleFtdcInstrumentIDType         InstrumentID;
///合约名称
  TXeleFtdcInstrumentNameType   InstrumentName;
///交割年份
  TXeleFtdcYearType             DeliveryYear;
///交割月
  TXeleFtdcMonthType            DeliveryMonth;
///提前月份
  TXeleFtdcAdvanceMonthType         AdvanceMonth;
///当前是否交易
  TXeleFtdcBoolType             IsTrading;
};
```

返回值： 无；
说明： API暂不支持该接口；

OnRtnInstrumentStatus方法

功能：合约交易状态改变回报
函数原形：

```
void OnRtnInstrumentStatus(CXeleFtdcInstrumentStatusField *pInstrumentStatus)
```

参数：

- pInstrumentStatus：合约状态域，指向合约状态结构的地址。
  合约状态结构体CxeleFtdcInstrumentStatusField如下：

```
struct CXeleFtdcInstrumentStatusField {
///结算组代码
   TXeleFtdcSettlementGroupIDType   SettlementGroupID;
///合约代码
  TXeleFtdcInstrumentIDType         InstrumentID;
///合约交易状态
  TXeleFtdcInstrumentStatusType     InstrumentStatus;
///交易阶段编号
  TXeleFtdcTradingSegmentSNType     TradingSegmentSN;
///进入本状态时间
  TXeleFtdcTimeType             EnterTime;
///进入本状态原因
   TXeleFtdcInstStatusEnterReasonType   EnterReason;
};
```

返回值： 无；
说明： API暂不支持该接口；

OnErrRtnOrderInsert方法

功能：报单录入错误回报
函数原形：

```
void OnErrRtnOrderInsert(CXeleFtdcInputOrderField   *pInputOrder,
 CXeleFtdcRspInfoField *pRspInfo)
```

参数：

- pInputOrder：报单录入域，指向报单录入结构的地址，包含了提交报单录入时的输入数据，和交易系统返回的报单编号。
  报单录入结构体CxeleFtdcInputOrderField如下：

```
struct CXeleFtdcInputOrderField {
    ///报单编号
  TXeleFtdcOrderSystemNoType         OrderSystemNo;
///未用字段
char     _unused_1[9];
///会员代码
  TXeleFtdcParticipantIDType         ParticipantID;
///客户代码
  TXeleFtdcClientIDType         ClientID;
///交易用户代码
  TXeleFtdcUserIDType           UserID;
```

```
///合约代码
  TXeleFtdcInstrumentIDType         InstrumentID;
///报单价格条件
  TXeleFtdcOrderPriceTypeType       OrderPriceType;
///买卖方向
  TXeleFtdcDirectionType        Direction;
///组合开平标志
  TXeleFtdcCombOffsetFlagType   CombOffsetFlag;
///组合投机套保标志
    TXeleFtdcCombHedgeFlagType  CombHedgeFlag;
///价格
  TXeleFtdcPriceType            LimitPrice;
///数量
  TXeleFtdcVolumeType       VolumeTotalOriginal;
///有效期类型
  TXeleFtdcTimeConditionType        TimeCondition;
///GTD日期
  TXeleFtdcDateType             GTDDate;
///成交量类型
  TXeleFtdcVolumeConditionType  VolumeCondition;
///最小成交量
  TXeleFtdcVolumeType       MinVolume;
///触发条件
  TXeleFtdcContingentConditionType  ContingentCondition;
///止损价
  TXeleFtdcPriceType            StopPrice;
///强平原因
  TXeleFtdcForceCloseReasonType ForceCloseReason;
///本地报单编号
  TXeleFtdcOrderLocalNoType         OrderLocalNo;
///未用字段
   char         _unused_2[9];
///自动挂起标志
  TXeleFtdcBoolType             IsAutoSuspend;
///交易所报单编号, RspOrderInsert时有意义
  TXeleFtdcExchangeOrderSysIDType   ExchangeOrderSysID;
///用户定义交易所前置发单, 可选.
  TXeleFtdcExchangeFrontType        ExchangeFront;
///未用字段
  char      _unused_3[9];
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
  TXeleFtdcErrorIDType          ErrorID;
///错误信息
   TXeleFtdcErrorMsgType        ErrorMsg;
};
```

返回值： 无；

OnErrRtnOrderAction方法

功能：报单操作错误回报
函数原形：

```
void OnErrRtnOrderAction(CXeleFtdcOrderActionField *pOrderAction,
CXeleFtdcRspInfoField *pRspInfo)
```

参数：

- pOrderAction：报单操作域，指向报单操作结构的地址，包含了提交报单操作的输入数据，和交易系统返回的报单编号。
  报单操作结构体CxeleFtdcOrderActionField为：

```
struct CXeleFtdcOrderActionField {
///报单编号
  TXeleFtdcOrderSystemNoType        OrderSystemNo;
///未用字段
  char  _unused_1[9];
///本地报单编号
  TXeleFtdcOrderLocalNoType         OrderLocalNo;
///未用字段
  char  _unused_2[9];
///报单操作标志
  TXeleFtdcActionFlagType       ActionFlag;
///会员代码
  TXeleFtdcParticipantIDType        ParticipantID;
///客户代码
  TXeleFtdcClientIDType         ClientID;
///交易用户代码
  TXeleFtdcUserIDType           UserID;
///价格
  TXeleFtdcPriceType            LimitPrice;
///数量变化
  TXeleFtdcVolumeType       VolumeChange;
///操作本地编号
   TXeleFtdcOrderLocalIDType        ActionLocalID;
///未用字段
  char   _unused_3[9];
///操作本地代码
  TXeleFtdcActionLocalNoType        ActionLocalNo;
///未用字段
  char  _unused_4[12];
};
```

- pRspInfo：响应信息域，指向响应信息结构的地址。
  响应信息结构体CxeleFtdcRspInfoField如下：

```
struct CXeleFtdcRspInfoField {
///错误代码
 TXeleFtdcErrorIDType   ErrorID;
///错误信息
   TXeleFtdcErrorMsgType    ErrorMsg;
};
```

OnRtnHistoryTrade方法

功能：历史成交回报，当客户端登录时，如果此时有历史成交回报，该方法会被调用。
参数原形：

```
void OnRtnHistoryTrade (CXeleFtdcTradeField *pTrade)
```

- pTrade ：成交回报域，指向成交回报结构的地址。注意：成交回报中的某些字段未使用，柜台系统返回为空值。
  成交回报结构体CxeleFtdcTradeField和5.6.1 OnRtnTrade中的结构体完全一样。

备注说明： 该方法和 5.6.1中的OnRtnTrade 方法参数以及使用方式完全一样，只是该方法表示的是客户在登录过程中推送的历史成交流水数据；用户可以通过该回调接口判断历史成交回报流水是否推送完成；

OnRtnHistoryOrder方法

功能：历史报单回报，当客户端登录时，如果此时有历史报单回报，该方法会被调用。
函数原形：

```
void OnRtnOrder(CXeleFtdcOrderField *pOrder)
```

参数：

- pOrder ：报单回报域，指向报单回报结构的地址。注意：报单回报中的某些字段未使用，柜台系统返回为空值。
  报单回报结构体CxeleFtdcOrderField和5.6.2中的OnRtnOrder结构体完全一样。
  备注说明： 该方法和 5.6.2中的OnRtnOrder方法参数、使用方式完全一样，只是该方法表示的是客户在登录过程中推送的历史回报流水数据；用户可以通过该回调接口判断历史报单回报流水是否推送完成；

TraderAPI开发示例

下列是我们打包发给您的userdemo.cpp代码，由于文档篇幅有限，仅摘录主函数、重要的**CSimpleOrderManager**以及报撤单部分，供您参考使用，具体详见“userdemo.cpp”。

```
 /*本例将演示以及基本操作流程：
 *    1. 登录操作
 *    2. 报单查询操作
 *    3. 报撤单操作
 */
 /*
 * @class CSimpleOrderManager
 * @brief 利用报单管理场景，演示本API接口调用。
 * @notes CXeleTraderSpi中OnXXX()形式的方法为回调函数，不能阻塞过长时间，
 * 否则，会影响接收其他来自交易柜台的消息。
 */
class CSimpleOrderManager: public CXeleTraderSpi
{
public:
  CXeleFtdcReqUserLoginField login_info;
  CXeleFtdcAuthenticationInfoField auth_info;
  /* 构造函数，需要一个有效的指向CXeleTraderApi实例的指针 */
  CSimpleOrderManager(CXeleTraderApi* pTraderApi) :m_pTraderApi(pTraderApi)
```

```
  {
  }
  ~CSimpleOrderManager()
  {
  }
  /*当客户端与柜台所有链接都建立成功时，会回调该函数*/
  virtual void OnFrontConnected()
  {
    example_ReqUserLogin(&login_info);
    m_RspEvent = false;
example_RegisterAuthentication(&auth_info);
/* 注册授权码*/
m_pTraderApi->RegisterAuthentication(&auth_info);
/* 调用登录请求API接口 */
    m_pTraderApi->ReqUserLogin(&login_info, 0);
  }
  /* 当客户端与柜台通信连接断开时，该方法被调用， nReason表明了断链原因*/
  virtual void OnFrontDisconnected(int nReason)
  {
      fprintf(stdout, "OnFrontDisconnected:%d\n", nReason);
  }
  /*登录响应 */
  virtual void OnRspUserLogin(CXeleFtdcRspUserLoginField *pRspUserLogin,
CXeleFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
  {
    if (pRspInfo->ErrorID != 0)
    {
      /* 失败， 客户端程序需进行错误处理  */
      PRINT_RSP_ERR(pRspInfo);
      exit(pRspInfo->ErrorID);
    }
    else
    {
      /* 成功，处理数据...  */
      PRINT_RSP(pRspInfo);
/* 记录登录响应中的关键信息，方便进行报撤单操作 */
      m_exchange_num    = pRspUserLogin->ExchangeNum;
      m_client_index[0] = pRspUserLogin->ClientIndex[0];
      m_token[0]        = pRspUserLogin->Token[0];
      m_client_index[1] = pRspUserLogin->ClientIndex[1];
      m_token[1]        = pRspUserLogin->Token[1];
    }
    m_RspEvent = true;
  }
/* 报单响应 */
  virtual void OnRspOrderInsert(CXeleFtdcInputOrderField *pInputOrder,
CXeleFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
  {
    if (pRspInfo->ErrorID != 0)
    {
      /* 失败， 客户端程序需进行错误处理  */
      PRINT_RSP_ERR(pRspInfo);
      exit(pRspInfo->ErrorID);
    }
    else
    {
      /* 成功，处理数据...  */
      PRINT_RSP(pRspInfo);
```

```
/* 记录柜台生成的系统报单编号，用于撤单 */
      if(m_order_cnt < m_exchange_num) {
        m_ordersysno[m_order_cnt++] = pInputOrder->OrderSystemNo;
      }
    }
      m_RspEvent = true;
  }
/* 撤单响应 */
  virtual void OnRspOrderAction(CXeleFtdcOrderActionField *pOrderAction,
CXeleFtdcRspInfoField *pRspInfo, int nRequestID, bool bIsLast)
  {
    if (pRspInfo->ErrorID != 0)
    {
      /* 失败， 客户端程序需进行错误处理  */
      PRINT_RSP_ERR(pRspInfo);
      exit(pRspInfo->ErrorID);
    }
    else
    {
      /* 成功，处理数据...  */
      PRINT_RSP(pRspInfo);
    }
      m_RspEvent = true;
  }
  /* 报单回报 */
  virtual void OnRtnOrder(CXeleFtdcOrderField *pOrder)
  {
    printf("OnRtnOrder: OrderLocalNo=[%d]\n", pOrder->OrderLocalNo);
  }
/* 成交回报 */
  virtual void OnRtnTrade(CXeleFtdcTradeField *pTrade)
  {
    printf("OnRtnTrade: OrderLocalNo=[%d]\n", pTrade ->OrderLocalNo);
  }
  /* 针对用户请求的出错通知 */
  virtual void OnRspError(CXeleFtdcRspInfoField *pRspInfo, int nRequestID, bool
bIsLast)
  {
    PRINT_RSP(pRspInfo);
    /* 客户端程序需进行错误处理 */
    exit(-1);
  }
  /* 报单查询响应 */
  virtual void OnRspQryOrder(CXeleFtdcOrderField* pOrderField,
CXeleFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
  {
    if (pRspInfo->ErrorID != 0)
    {
 /* 失败， 客户端程序需进行错误处理  */
         PRINT_RSP_ERR(pRspInfo);
         exit(pRspInfo->ErrorID);
    }
    else
    {
          /* 成功，处理数据...  */
          PRINT_RSP(pRspInfo);
    }
    m_RspEvent = true;
```

```
  }
  /* 等待响应事件 */
  void WaitRspEvent()
  {
    do
    {
      sched_yield();
    } while (true != m_RspEvent);
  }
  /* 报单查询示例*/
  void ReqQryOrder(size_t trader = 0)
  {
    CXeleFtdcQryOrderField qryOrder;
    memset(&qryOrder, 0, sizeof(CXeleFtdcQryOrderField));
    /* 客户代码 */
    snprintf(qryOrder.AccountID, sizeof(qryOrder.AccountID), "%s",
g_AccountID_str[trader]);
m_RspEvent = false;
/* 调用API接口查询报单 */
    m_pTraderApi->ReqQryOrder(&qryOrder, 1);
  }
  /* 查询管理示例 流程*/
  bool SimpleQueryManagement()
  {
/* 报单查询请求 */
    ReqQryOrder();
    WaitRspEvent();
    return true;
  }
  /*报单管理示例流程 */
  void SimpleOrderManagement()
  {
    int i = 0;
    CXeleFairInputOrderField order;
    CXeleFairOrderActionField action;
    CXeleFairOrderFieldV order_action[MAX_ORDER_NUM];
    /* 等待登入成功 */
    WaitRspEvent();
    /* 查询管理示例接口 */
    (void)SimpleQueryManagement();
    /* 构造报单数据并调用接口报单 */
m_order_cnt = 0;
/* for循环是柜台支持双交易所模式，demo也支持分别报单 */
    for(i = 0; i < m_exchange_num; i++)
    {
      memset(&order, 0, sizeof(CXeleFairInputOrderField));
     /* 构造报单数据*/
      example_makeup_order(&order, 0, m_client_index[i], m_token[i]);
      sleep(2);
/* 调用API的报单接口进行报单 */
      m_pTraderApi->ReqOrderInsert(&order, 1);
     /* 等待报单响应回报 */
      WaitRspEvent();
    }
    /* 撤单接口，处理和报单类似 */
    for(i = 0; i < m_exchange_num; i++)
    {
      memset(&action, 0, sizeof(CXeleFairOrderActionField));
```

```
        example_makeup_action(&action, 0, m_client_index[i], m_token[i],
m_ordersysno[i]);
        sleep(2);
        m_pTraderApi->ReqOrderAction(&action, 1);
        WaitRspEvent();
      }
  /* 主函数*/
int main(int argc, char* argv[])
{
  /*读取demo的配置文件config_userdemo.ini*/
  loadConfigFile((char*) "config_userdemo.ini");
/*创建1个API实例*/
  CXeleTraderApi* pTraderApi = CXeleTraderApi::CreateTraderApi();
/*创建spi类实例*/
  CSimpleOrderManager som(pTraderApi);
/*注册spi类实例*/
  pTraderApi->RegisterSpi(&som);
/*注册柜台前置地址*/
  pTraderApi->RegisterFront(g_FrontAddress_str, g_QueryFrontAddress_str,
g_LocalAddress_str);
  /* 默认报单通讯是非阻塞模式，若设置阻塞模式请在调用Init函数之前设置 */
  pTraderApi->RegisterChannelBlock(1);
  /* 订阅相应的流*/
  pTraderApi->SubscribePrivateTopic(XELE_TERT_RESTART);
  /* 初始化API，使客户端程序开始与交易柜台建立连接*/
  pTraderApi->Init();
  /*打印API版本信息*/
  fprintf(stdout, "%s\n", pTraderApi->GetVersion());
 /* 实例的报单管理 函数*/
  som.SimpleOrderManagement();
/* 主线程打印，等待退出*/
  std::string msg;
  do
  {
    cout << "Input 'q' to disconnect API:";
    getline(cin, msg);
  } while (msg != "q");
  /* 释放API实例 */
  pTraderApi->Release();
  fprintf(stdout, "API release done. Exit Demo.\n");
  return 0;
}
```

字段类型

字段类型总表

自定义的基本数据类型如下（其中需要注意uint8_t表示的是1字节整型，非char型的字符类型）：

数据类型	定义	数据类型说明
uint8_t	unsigned char	1字节无符号整型
uint16_t	unsigned short	2字节无符号整型
uint32_t	unsigned int	4字节无符号整型
int8_t	signed char	1字节有符号整型
int16_t	signed short	2字节有符号整型
int32_t	signed int	4字节有符号整型

业务字段数据类型如下：

序号	数据类型名	数据类型	数据类型说明
1	TXeleFtdcClientIndexType	uint8_t，1字节整型	客户端Index类型
2	TXeleFtdcClientTokenType	uint16_t，2字节整型	客户端令牌类型
3	TXeleFtdcExchangeNumType	uint8_t， 1字节整型	交易所数目类型
4	TXeleFtdcSequenceNoType	Int，4字节整型	序列号类型
5	TXeleFtdcSequenceSeriesType	Short，2字节短整型	序列系列号类型
6	TXeleFtdcCombinedContractAuthorizeType	uint8_t，1字节整型	组合合约保证金授权字段
7	TXeleFtdcSettlementGroupIDType	char[9]，字符数组	结算组代码类型
8	TXeleFtdcInstrumentIDType	char[31]，字符数组	合约代码类型
9	TXeleFtdcTradingSegmentSNType	Int，4字节整型	交易阶段编号类型
10	TXeleFtdcTimeType	char[9]，字符数组	时间类型
11	TXeleFtdcDateType	char[9]，字符数组	日期类型

序号	数据类型名	数据类型	数据类型说明
12	TxeleFtdcInstStatusEnterReasonType	char， 字符类型	原因类型
13	TXeleFtdcSettlementIDType	int， 4字节整型	结算编号类型
14	TXeleFtdcParticipantIDType	char[11]，字符数组	会员代码类型
15	TXeleFtdcClientIDType	char[11]，字符数组	客户代码类型
16	TXeleFtdcVolumeTotalOriginalType	short， 2字节短整型	报单数量类型
17	TXeleFtdcMinVolumeType	short， 2字节短整型	最小成交量类型
18	TXeleFtdcVolumeType	int， 4字节整型	数量类型
19	TXeleFtdcMoneyType	double，浮点类型	资金类型
20	TXeleFtdcBulletinIDType	int，4字节整型	公告编号类型
21	TXeleFtdcNewsTypeType	char[3]， 字符数组	公告类型类型
22	TXeleFtdcNewsUrgencyType	char， 字符	紧急程度类型
23	TXeleFtdcAbstractType	char[81]， 字符数组	消息摘要类型
24	TXeleFtdcComeFromType	char[21]， 字符数组	消息来源类型

序号	数据类型名	数据类型	数据类型说明
25	TXeleFtdcOrderSysIDType	char[13]， 字符数组	报单编号类型
26	TXeleFtdcOrderLocalNoType	int， 4字节整型	本地报单编号类型
27	TXeleFtdcOrderSystemNoType	int， 4字节整型	系统报单编号类型
28	TXeleFtdcInsertType	uint8_t: 1字节整型	报单输入类型
29	TXeleFtdcActionLocalNoType	int，4字节整型	报单操作本地编号类型
30	TXeleFtdcOrderLocalIDType	char[13]，字符数组	被撤单本地编号类型
31	TXeleFtdcOrderSysNoType	int，4字节整型	被撤单系统报单编号类型
32	TXeleFtdcExchangeFrontEnumType	uint8_t，1字节整型	交易所前置枚举类型
33	TXeleFtdcFrontIpListType	char[16][16]，第1个数组下标表示对应的前置编号，第2个表示交易所的前置IP，点分十进制字符数组类型	交易所前置IP地址列表类型

序号	数据类型名	数据类型	数据类型说明
34	TXeleFtdcUserIDType	char[16]，字符数组	交易用户代码类型
35	TXeleFtdcSessionNumberType	short，2字节短整型	会话编号类型
36	TXeleFtdcTradeIDType	char[13]，字符数组	成交编号类型
37	TXeleFtdcProductIDType	char[9]，字符数组	产品代码类型
38	TXeleFtdcProductGroupIDType	char[9]，字符数组	产品组代码类型
39	TXeleFtdcProductInfoType	char[41]，字符数组	产品信息类型
40	TXeleFtdcDataCenterIDType	int，4字节整型	数据中心代码类型
41	TXeleFtdcPriceType	double，浮点类型	价格类型
42	TXeleFtdcBoolType	int，4字节整型	布尔型类型
43	TXeleFtdcOrderLocalIDType	char[13]，字符数组	本地报单编号类型
44	TXeleFtdcContentType	char[501]，字符数组	消息正文类型
45	TXeleFtdcBusinessUnitType	char[21]，字符数组	业务单元类型

序号	数据类型名	数据类型	数据类型说明
46	TXeleFtdcBusinessLocalIDType	int，4字节整型	本地业务标识类型
47	TXeleFtdcMonthCountType	int，4字节整型	月份数量类型
48	TXeleFtdcYearType	Int，4字节整型	年份类型
49	TXeleFtdcMonthType	Int，4字节整型	月份类型
50	TXeleFtdcAdvanceMonthType	char[4]，字符数组	提前月份类型
51	TXeleFtdcURLLinkType	char[9]，字符数组	WEB地址类型
52	TXeleFtdcMarketIDType	char[9]，字符数组	市场代码类型
53	TXeleFtdcPriorityType	int，4字节整型	优先权类型
54	TXeleFtdcPriorityType	Int，4字节整型	按时间排队的序号类型
55	TXeleFtdcCombOffsetFlagType	char[5]，字符数组	组合开平标志类型
56	TXeleFtdcCombHedgeFlagType	char[5]，字符数组	组合投机套保标志类型

序号	数据类型名	数据类型	数据类型说明
57	TXeleFtdcVolumeMultipleType	int，4字节整型	合约数量乘数类型
58	TXeleFtdcUnderlyingMultipleType	double，浮点类型	合约基础商品乘数类型
59	TXeleFtdcInstrumentIDType	char[16]，字符数组	合约ID类型
60	TXeleFtdcInstrumentNameType	char[21]，字符数组	合约名称类型
61	TXeleFtdcErrorIDType	int，4字节整型	错误代码类型
62	TXeleFtdcErrorMsgType	char[81]，字符数组	错误信息类型
63	TXeleFtdcTradingSystemNameType	char[57]，字符数组	交易系统名称类型
64	TXeleFtdcAccountIDType	char[13]，字符数组	资金帐号类型
65	TXeleFtdcPartyNameType	char[81]，字符数组	参与人名称类型
66	TXeleFtdcIdCardTypeType	char[16]，字符数组	证件类型类型
67	TXeleFtdcIdentifiedCardNoV1Type	char[21]，字符数组	原证件号码类型

序号	数据类型名	数据类型	数据类型说明
68	TXeleFtdcIdentifiedCardNoType	char[51]，字符数组	证件号码类型
69	TXeleFtdcPasswordType	char[41]， 字符数组	密码类型
70	TXeleFtdcProtocolInfoType	char[41]，字符数组	协议信息类型
71	TXeleFtdcFrontListType	char[16]，1个字节整型数组	交易所交易前置代码列表类型
72	TXeleFtdcExchangNoType	uint16_t，2字节短整型 0:NULL，1:SHFE，2:INE，3:DCE， 4:CZCE， 5:CFFEX	交易所标志
73	TXeleFtdcFrontCountType	uint16_t， 2字节短整型；0:NULL，1:SHFE，2:INE，3:DCE， 4:CZCE， 5:CFFEX	交易所前置数量
74	TXeleFtdcExchangeFrontType	char[3]，char[0]:字符类型，char[1]和char[2]是1字节整型	交易所交易前置描述符类型
75	TXeleFtdcExchangeOrderSysIDType	char[13]，字符数组	交易所订单编号类型
76	TXeleFtdcAppIDType	char[29]，字符数组	终端软件AppID类型
77	TXeleFtdcAuthCodeType	char[17]，字符数组	终端软件授权码类型

序号	数据类型名	数据类型	数据类型说明
78	TXeleFtdcRatioType	double，浮点型	比率类型
79	TXeleFtdcProductReserve1Type	uint8_t，1字节整型	产品保留类型
80	TxeleFtdcProductReserve3Type	int，4字节整型	产品保留类型
81	TXeleFtdcInstrumentStatusType	字符枚举（详见7.2.1）	合约交易状态类型
82	TXeleFtdcOrderPriceTypeType	字符枚举（详见7.2.2）	报单价格条件类型
83	TXeleFtdcPositionTypeType	字符枚举（详见7.2.3）	持仓类型类型
84	TXeleFtdcClientTypeType	字符枚举（详见7.2.4）	客户类型
85	TXeleFtdcHedgeFlagType	字符枚举（详见7.2.5）	投机套保标志类型
86	TXeleFtdcPosiDirectionType	字符枚举（详见7.2.6）	持仓多空方向类型
87	TXeleFtdcProductClassType	字符枚举（详见7.2.7）	产品类型类型
88	TXeleFtdcOrderSourceType	字符枚举（详见7.2.8）	报单来源类型
89	TXeleFtdcOrderStatusType	字符枚举（详见7.2.9）	报单状态类型

序号	数据类型名	数据类型	数据类型说明
90	TXeleFtdcOrderTypeType	字符枚举（详见7.2.10）	报单类型类型
91	TXeleFtdcDirectionType	字符枚举（详见7.2.11）	买卖方向类型
92	TXeleFtdcTimeConditionType	字符枚举（详见7.2.12）	有效期类型类型
93	TXeleFtdcVolumeConditionType	字符枚举（详见7.2.13）	成交量类型类型
94	TXeleFtdcContingentConditionType	字符枚举（详见7.2.14）	触发条件类型
95	TXeleFtdcForceCloseReasonType	字符枚举（详见7.2.15）	强平原因类型
96	TXeleFtdcOptionsTypeType	字符枚举（详见7.2.16）	期权类型类型
97	TXeleFtdcActionFlagType	字符枚举（详见7.2.17）	操作标志类型
98	TXeleFtdcTradingRoleType	字符枚举（详见7.2.18）	交易角色类型
99	TXeleFtdcOffsetFlagType	字符枚举（详见7.2.19）	开平标志类型
100	TXeleFtdcTradeTypeType	字符枚举（详见7.2.20）	成交类型类型
101	TXeleFtdcPriceSourceType	字符枚举（详见7.2.21）	报价来源类型

序号	数据类型名	数据类型	数据类型说明
102	TXeleFtdcExchangeDescriptorType	uint8_t，1字节整型， bit0-bit3：表示交易所描述符，bit4-bit5：表示查询类型，bit6-bit7：预留。 （详见 7.2.22 ）	交易所描述符类型

枚举类型

TXeleFtdcInstrumentStatusType合约交易状态类型

类型：字符枚举

合约状态类型	说明	枚举值(可显示字符)
XELE_FTDC_IS_BeforeTrading	开盘前	‘0’
XELE_FTDC_IS_NoTrading	非交易	‘1’
XELE_FTDC_IS_Continous	连续交易	‘2’
XELE_FTDC_IS_AuctionOrdering	集合竞价报单	‘3’
XELE_FTDC_IS_AuctionBalance	集合竞价价格平衡	‘4’
XELE_FTDC_IS_AuctionMatch	集合竞价撮合	‘5’
XELE_FTDC_IS_Closed	收盘	‘6’

TXeleFtdcOrderPriceTypeType报单价格条件类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_OPT_AnyPrice	任意价	‘1’
XELE_FTDC_OPT_LimitPrice	限价	‘2’
XELE_FTDC_OPT_BestPrice	最优价	‘3’
XELE_FTDC_OPT_FiveLevelPrice	5档价	‘4’

TXeleFtdcPositionTypeType持仓类型类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_PT_Net	净持仓	‘1’
XELE_FTDC_PT_Gross	综合持仓	‘2’

TXeleFtdcClientTypeType客户类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_CT_Person	自然人	‘0’
XELE_FTDC_CT_Company	法人	‘1’
XELE_FTDC_CT_Fund	投资基金	‘2’

TXeleFtdcHedgeFlagType投机套保标志类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_HF_Speculation	投机	‘1’
XELE_FTDC_HF_Arbitrage	套利	‘2’
XELE_FTDC_HF_Hedge	套保	‘3’

TXeleFtdcPosiDirectionType持仓多空方向类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_PD_Net	净	‘1’
XELE_FTDC_PD_Long	多头	‘2’
XELE_FTDC_PD_Short	空头	‘3’

TXeleFtdcProductClassType产品类型类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_PC_Futures	期货	'1'
XELE_FTDC_PC_Options	期权	'2'
XELE_FTDC_PC_Combination	组合	'3'
XELE_FTDC_PC_Spot	即期	'4'
XELE_FTDC_PC_EFP	期转现	'5'
XELE_FTDC_PC_IO	股指期权	'6'

TXeleFtdcOrderSourceType报单来源类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_OSRC_Participant	来自参与者	'0'
XELE_FTDC_OSRC_Administrator	来自管理员	'1'

TXeleFtdcOrderStatusType报单状态类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_OST_AllTraded	全部成交	'0'
XELE_FTDC_OST_PartTradedQueueing	部分成交还在队列中	'1'
XELE_FTDC_OST_PartTradedNotQueueing	部分成交不在队列中	'2'
XELE_FTDC_OST_NoTradeQueueing	未成交还在队列中	'3'
XELE_FTDC_OST_NoTradeNotQueueing	未成交不在队列中	'4'
XELE_FTDC_OST_Canceled	撤单	'5'

TXeleFtdcOrderTypeType报单类型类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_ORDT_Normal	正常	'0'
XELE_FTDC_ORDT_DeriveFromQuote	报价衍生	'1'
XELE_FTDC_ORDT_DeriveFromCombination	组合衍生	'2'

TXeleFtdcDirectionType买卖方向类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_D_Buy	买	‘0’
XELE_FTDC_D_Sell	卖	‘1’

TXeleFtdcTimeConditionType有效期类型类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_TC_IOC	立即完成，否则撤销	‘1’
XELE_FTDC_TC_GFS	本节有效	‘2’
XELE_FTDC_TC_GFD	当日有效	‘3’
XELE_FTDC_TC_GTD	指定日期前有效	‘4’
XELE_FTDC_TC_GTC	撤销前有效	‘5’
XELE_FTDC_TC_GFA	集合竞价有效	‘6’

TXeleFtdcVolumeConditionType成交量类型类型

类型: 字符枚举

枚举类型	说明	枚举值
XELE_FTDC_VC_AV	任何数量	‘1’
XELE_FTDC_VC_MV	最小数量	‘2’
XELE_FTDC_VC_CV	全部数量	‘3’

TXeleFtdcContingentConditionType触发条件类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_CC_Immediately	立即	‘1’
XELE_FTDC_CC_Touch	止损	‘2’

TXeleFtdcForceCloseReasonType强平原因类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_FCC_NotForceClose	非强平	‘0’
XELE_FTDC_FCC_LackDeposit	资金不足	‘1’
XELE_FTDC_FCC_ClientOverPositionLimit	客户超仓	‘2’
XELE_FTDC_FCC_MemberOverPositionLimit	会员超仓	‘3’
XELE_FTDC_FCC_NotMultiple	持仓非整数倍	‘4’
XELE_FTDC_FCC_Violation	违规	‘5’
XELE_FTDC_FCC_Other	其它	‘6’
XELE_FTDC_FCC_PersonDeliv	自然人临近交割	‘7’

TXeleFtdcOptionsTypeType期权类型类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_OT_NotOptions	非期权	‘0’
XELE_FTDC_OT_CallOptions	看涨	‘1’
XELE_FTDC_OT_PutOptions	看跌	‘2’

TXeleFtdcActionFlagType操作标志类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_AF_Delete	删除	‘0’
XELE_FTDC_AF_Suspend	挂起	‘1’
XELE_FTDC_AF_Active	激活	‘2’
XELE_FTDC_AF_Modify	修改	‘3’

TXeleFtdcTradingRoleType交易角色类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_ER_Broker	代理	‘1’
XELE_FTDC_ER_Host	自营	‘2
XELE_FTDC_ER_MarketMaker	做市商	‘3’

TXeleFtdcOffsetFlagType开平标志类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_OF_Open	开仓	‘0’
XELE_FTDC_OF_Close	平仓	‘1’
XELE_FTDC_OF_ForceClose	强平	‘2’
XELE_FTDC_OF_CloseToday	平今	‘3’
XELE_FTDC_OF_CloseYesterday	平昨	‘4’

TXeleFtdcTradeTypeType成交类型类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_TRDT_Common	普通成交	‘0’
XELE_FTDC_TRDT_OptionsExecution	期权执行	‘1’
XELE_FTDC_TRDT_OTC	OTC成交	‘2’
XELE_FTDC_TRDT_EFPDerived	期转现衍生成交	‘3’
XELE_FTDC_TRDT_CombinationDerived	组合衍生成交	‘4’

TXeleFtdcPriceSourceType报价来源类型

类型：字符枚举

枚举类型	说明	枚举值
XELE_FTDC_PSRC_LastPrice	前成交价	‘0’
XELE_FTDC_PSRC_Buy	买委托价	‘1’
XELE_FTDC_PSRC_Sell	卖委托价	‘2’

TXeleFtdcExchangeDescriptorType交易所描述符类型

类型： 1字节整型枚举，bit0-bit3表示交易所描述符，bit4-bit5：表示查询类型，bit6-bit7：预留。

交易所描述符枚举类型	说明	枚举值
XELE_FTDC_ED_NULL	无效的交易所	0
XELE_FTDC_ED_SHFE	上海期货交易所	1
XELE_FTDC_ED_INE	上海国际能源交易中心股份有限公司	2
XELE_FTDC_ED_DCE	大连商品交易所	3
XELE_FTDC_ED_CZCE	郑州商品交易所	4
XELE_FTDC_ED_CFFEX	中国金融交易所	5

查询类型枚举值	说明
0	单腿合约持仓
1	组合合约持仓（当前仅大商所支持）
2	合约持仓