

简介

API简介

XeleSecuritiesTraderAPI用于与Xele-Trade-Securities进行通信。通过API，投资者可以向上海证券交易所（SSE）、深圳交易所（SZSE）发送交易指令，获得相应的回复和交易状态回报。

该文档是极速交易系统Xele-Trade-Securities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。本说明书适用于极速交易系统的交易API。

Xele-Trade-Securities简介

Xele-Trade-Securities是业内领先的基于FPGA、亚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。

Xele-Monitor-Securities是Xele-Trade-Securities的图形可视化交互客户端。

XeleSecuritiesTraderAPI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。

XeleSecuritiesTraderAPI支持平台：Linux。

发布给用户的档案包以XeleSecuritiesTraderAPI-版本号.tgz命名，包含的文件如下：

文件夹	文件名	说明
include/	XeleSecuritiesTraderApi.h	交易员API接口头文件, 定义了目前支持的接口
include/	XeleSecuritiesUserApiStruct.h	定义了函数接口的参数结构体，即域类型
include/	XeleSecuritiesUserApiData.h	定义了域中的字段的类型
lib/	libXeleSecuritiesAPI.so	API动态库
demo/example_stock/	config_file_process	demo中可能会用的代码文件
	demo_main.cpp	userdemo示例程序
demo/example_stock/	api_config.txt	API配置文件
demo/example_stock/	CmakeLists.txt	userdemo编译文件
config	api_config.txt	api运行需要的配置文件
Debug/lib	libXeleSecuritiesAPI_debug.so	Debug版本Api动态库
docs	Xele-Trade-Securities-交易员API接口手册.pdf	API的说明文件
顶层目录	changelog	API版本升级信息说明

修改历史

日期	API 版本	柜台 版本	版本修订
07/22/2021	2.0		API支持报撤单、相关查询（资金冻结、解冻及流水查询、柜台间资金调拨及流水查询、资金重构流水查询、报单回报流水查询等）操作; 支持心跳线程;
1/17/2022	2.4	2.5	API支持Manager(管理中心)接口功能 新增manager管理中心的错误码 各柜台（上交股票、深交股票、上交期权）的错误码分类 重新命名函数接口统一名称 修改CxeleRtnOrderField结构体
6/1/2022	3.0	3.0	修改API的api_config.txt配置文件
6/16/2022	3.0	3.0	新增reqLoginEx接口开放给用户，用于给不喜欢配置文件的投资者使用
7/13/2022	3.0	3.0	报单和成交查询的结构体中新增关于分页查询的成员变量 新增[排序类型]
7/21/2022	3.0	3.0	新增软件报单通道和硬件报单通道章节说明
8/2/2022	3.1	3.1	新增批量报单功能和系统配置巨页说明
9/21/2022	3.1	3.1	新增订单查询报单状态
9/26/2022	3.1	3.1	在API程序目录下自动生成日志文件
10/22/2022	3.1	3.1	中信站点信息修改，配套柜台中信版本使用
10/28/2022	3.1	3.1	增加可以测量半链路和全链路延迟的demo，在demo/example_order_delay目录下
5/5/2023	3.1	3.1	期权支持资金调拨相关接口 期权错误码更新
10/18/2023	3.2	3.2	api配置文件新增配置参数说明更新
11/24/2023	3.2	3.2	api头文件相关注释修改
11/28/2023	3.2	3.2	api头文件Operway注释修改
7/12/2023	3.2	3.2	查询报单及成交增加报单来源字段
12/12/2023	3.2	3.2	增加通用接口
2/28/2024	3.2	3.2	新增设置API日志路径及设置是否开启详细日志打印接口
10/29/2024	3.2	3.2	增加是否捕获异常参数
12/3/2024	3.2	3.2	新增部分券商出入仓及记录查询接口

日期	API 版本	柜台 版本	版本修订
12/25/2024	3.2	3.2	增加是否支持线程安全参数
01/16/2025	3.2	3.2	增加交易所网关查询接口
01/20/2025	3.2	3.2	支持使用TcpDirect技术实现通讯加速
2/7/2025	3.2	3.2	更新交易所错误码
02/07/2025	3.2	3.2	支持使用UserLocalID字段撤单
3/19/2024	3.2	3.2	部分券商支持仓位变化回报通知
4/1/2025	3.2	3.2	增加自动交易网关信息通知接口
4/7/2025	3.2	3.2	费率查询支持查询ETF申赎相关费率
12/5/2025	3.2	3.2	增加可用资金不足的错误码55115 调整错误码结构
15/5/2025	3.2	3.2	支持直连QFII柜台进行查询操作
17/6/2025	3.2	3.2	报单、撤单请求及报单查询响应增加委托方式扩展字段
6/8/2025	3.2	3.2	账户资金查询应答新增参考资产、总参考市值等字段 持仓查询应答新增参考市值字段
28/8/2025	3.2	3.2	修改及新增TXeleTransferType枚举说明
13/10/2025	3.2	3.2	支持UDP报撤单功能
11/12/2025	3.2	3.2	修改QFII持仓查询接口以及EFT申赎成份股信息查询接口说明
12/17/2025	3.2	3.2	增加要约收购和权证行权接口及相关数据结构定义
12/22/2025	3.2	3.2	增加市价单枚举
01/07/2026	3.2	3.2	修改TXeleOrderType类型中文含义
01/20/2026	3.2	3.2	报单回报结构体增加ExecID字段
03/10/2026	3.2	3.2	增加EstablishTradeLink配置参数，用来控制是否建立交易链路。默认为建立交易链路； 增加reqInitTrader接口用于发送初始化交易链路请求；
03/17/2026	3.2	3.2	增加checkPassowrd()接口，用于校验密码是否正确；
04/21/2026	3.2	3.2	增加配置参数TcpConnectTimeout，控制TCP首次建链的超时时间； 增加配置参数TcpReconnectTimeout，控制TCP断链重连的超时时间；

日期	API 版本	柜台 版本	版本修订
05/06/2026	3.2	3.2	增加getErrMsgEx()错误信息查询接口，支持中英文查询；原有静态方法getErrMsg()后续不再维护； 增加onDataReady()回调接口，用于通知异步数据加载完毕事件；
08/03/2026	3.2	3.2	更新期权相关说明

接口命名及通知响应

接口说明

```
////请求
int xeleSecuritiesTraderApi::reqXXX(
CXeleReqXXXField &inputField,
int nRequestID)

////响应
void xeleSecuritiesTraderSpi::onRspXXX(
CXeleXXXField *pInputField,
CXeleRspInfo *pRspInfo,
int nRequestID,
bool bIsLast)
```

艾科朗克柜台系统相关接口请求以reqXXX，响应以onRspXXX来命名。该模式调用API的reqXXX来发起标识为nRequestID的请求，在onRspXXX中处理相应的nRequestID的请求。

请求接口参数说明

接口名称：reqXXX

- inputField：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

响应接口参数说明

接口名称：onRspXXX

- pRspField: 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：响应请求的ID，标记请求报文和响应报文之间的对应关系
- bIsLast：该次请求ID是否响应完毕，即一次回应报文中，该响应回调函数可能会被多次调用，标识最后一次调用（bIsLast为TRUE标识是最后一次调用，为FALSE标识不是最后一次调用）

通知响应

```
void XeleSecuritiesTraderSpi::onRtnXXX(CXeleXXXField *pInputField, int
nRequestID, bool bIsLast);
////或者
void XeleSecuritiesTraderSpi::onErrRtnXXX(CXeleXXXField *pInputField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);
```

通知响应以onRtnXXX, onErrRtnXXX来命名。下面介绍这两种通知响应。

标准通知响应

接口名称：onRtnXXX

- pInputField：相应的通知回调数据域指针
- nRequestID：标记请求报文和响应报文之间的对应关系
- bIsLast：该次请求ID是否响应完毕，即一次回应报文中，该响应回调函数可能会被多次调用，标识最后一次调用（bIsLast为TRUE标识是最后一次调用，为FALSE标识不是最后一次调用）

错误通知响应

接口名称：onErrRtnXXX

- pInputField：相应的通知回调数据域指针
- pRspInfo：错误类型和错误信息
- nRequestID：标记请求报文和响应报文之间的对应关系
- bIsLast：该次请求ID是否响应完毕，即一次回应报文中，该响应回调函数可能会被多次调用，标识最后一次调用（bIsLast为TRUE标识是最后一次调用，为FALSE标识不是最后一次调用）

API运行模式

初始化连接阶段

客户端程序和柜台交易系统的交互过程分为2个阶段：

- 初始化连接阶段
- 功能调用阶段

初始化阶段

在初始化阶段，Xele-Trade-Securities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
- RegisterSpi(): 注册用户继承自XeleSecuritiesTraderSpi的自有子类对象的地址

注意事项：

XeleSecuritiesTraderApi提供的部分接口可以在配置参数的控制下实现线程安全，但前提是损耗一定的性能。具体支持线程安全的接口参考Api头文件注释。

初始化示例代码如下（详见示例代码“main.cpp”）：

```
/*
* api的初始化为以下三步：
* 1、创建一个接收回报的类，该类继承自XeleSecuritiesTraderSpi，用来接收艾科柜台的回报响应
```

- * 2、调用**createTradeApi**接口，创建一个请求类对象，该对象用来向艾科柜台发送操作请求
- * 3、调用**registerSpi**接口，将回报接收类对象传进请求类对象中
- * **api**获取操作权限的方式：
- * 1、调用**reqLogin**接口，传入配置文件路径、资金账户、资金账户密码以及本地维护的请求编号即可。
- * 其中，配置文件可以对**api**进行一些初始化的配置，包括**api**的连接方式（通过管理中心连接柜台还是直连柜台，这两种方式区别在于是否拥有跨柜台的资金操作权限，后者没有）、
- * **api**的绑核、心跳设置、**socket**类型选择、是否有地址、端口映射等
- * 当：
- * 1) 接口**onRspLoginManager**被回调，并且没有错误消息时，表明当前拥有艾科管理中心的操作权限
- * 2) 接口**onRspLogin**被回调，并且没有错误消息时，表明当前拥有艾科柜台的查询权限
- * 3) 接口**onRspInitTrader**被回调，并且没有错误消息时，表明当前拥有艾科柜台的报、撤单权限
- * */

```

auto* puserSpi = new DemoSpi();

pUserApi = xeleSecuritiesTraderApi::createTraderApi();
if(!puserApi){
    PRINT_INFO("can not create API");
    exit(0);
}

pUserApi->registerSpi(puserspi);

```

功能调用

在功能调用阶段，客户端程序可以任意调用登录、查询、交易接口中的请求方法，如reqLogin、reqInsertOrder等，同时提供回调函数以响应回报信息。

注意事项：

==API 请求的返回值参数RspInfo，ErrorID为0表示正确，其他表示错误，详细错误编码请查看附录A错误代码表==

API 配置文件

提示：

配置文件中必需变更项为ManagerURL和QueryURL。ManagerURL和QueryURL最少需要填写一个，填写ManagerURL字段表示通过manager模块连接柜台系统，只填写QueryURL字段表示直连柜台系统。其余字段为非必需变更项，需要根据券商特殊要求及个人功能需求变更配置。在使用新版本api时若不想替换新配置文件，需要满足使用版本在3.2.518（包括）之后。另外，若需要替换新配置文件，则只需将老配置文件中对应参数值重新配置到新配置文件中对应参数即可，其他新增参数按需配置，若保持默认值不变，则等价于使用老配置文件。

API 连接方式

通过Manager（管理中心）连接柜台（可获得管理中心和柜台的操作权限）

可以通过配置api_config.txt文件，配置Manager的地址（URL配置项）及其他相关配置。详见下文

API的配置文件

目前API的api_config.txt配置文件包含三部分：

- 网络配置
- Api参数配置

- 登录配置

网络配置

```
#####
##### Network configurations #####
#####
# 以下URL至少需要选择ManagerURL或者QueryURL其中之一进行配置，若两个都不配置则会报错。
# 在满足以上条件之后，其余的配置组合方式以是否填写值进行区分，若不为空则使用配置的值，若为空则使用默认值。

# 艾科管理中心地址，使用统一的格式tcp://地址:端口号
# 示例1: tcp://192.168.4.216:55555;tcp://192.168.4.214:55554
# 示例2: tcp://192.168.4.216:50000
# Api用来连接管理中心的参数，可以配置多个manager参数,使用;来分隔,多个地址表示逐个轮询连接
ManagerURL=tcp://192.168.4.216:55555;tcp://192.168.4.214:55554

# 艾科柜台查询链路连接地址，使用统一的格式tcp://地址:端口号
# 此参数为空，使用管理中心默认分配的地址，此参数为合法地址时，使用配置的值。地址、端口映射同理。配置格式如下示例：
# 示例1: tcp://192.168.4.67:30000
# 示例2: tcp://192.168.4.67:30000;tcp://192.168.4.68:30000
# 支持配置主备柜台地址，使用;来分隔，当主柜台连接失败会轮询连接备柜台
# Api自动重连也会轮询连接主备柜台
# 当同时配置了ManagerURL和QueryURL的情况下，Api连接manager成功之后会以配置项为准来连接柜台查询链路
QueryURL=

# 艾科报单链路连接地址，使用统一的格式tcp://地址:端口号
# 此参数为空，使用管理中心默认分配的地址，此参数为合法地址时，使用配置的值。地址、端口映射同理。配置格式如下示例：
# 示例: tcp://10.128.123.209:30005
# 场景: 例如原报单链路连接地址为tcp://10.128.123.209:30005，由于组网限制，需要将该地址映射为tcp://192.168.0.100:33333
# 此时，将TradeURL参数改为: TradeURL=tcp://192.168.0.100:33333; 如果没有此种情况只需要如下不填即可。
# 场景: 例如从管理中心查询到的地址为tcp://192.168.4.67:29999，现需要修改默认地址，
# 此时，将TradeURL参数改为: TradeURL=tcp://192.168.4.216:29999，如果没有这种需求，不填即可。
TradeURL=
```

图 API文件网络配置

• 3.0艾科管理中心地址

3.0艾科管理中心地址，使用统一的格式tcp://地址:端口号

示例1：tcp://192.168.4.216:50002;tcp://192.168.4.216:50003;tcp://192.168.4.216:50004

示例2：tcp://192.168.4.216:50002

Api用来连接管理中心的参数，可以配置多个manager参数,使用;来分隔

比如以50002或者62000这两个端口号结尾，表示该地址是合法的艾科管理中心连接地址

参考配置：

```
ManagerURL=tcp://192.168.2.37:50003
```

• 艾科查询链路连接地址

艾科柜台查询链路连接地址，使用统一的格式tcp://地址:端口号

此参数为空，使用管理中心默认分配的地址，此参数为合法地址时，使用配置的值。地址、端口映射同理。配置格式如下示例：

示例1：tcp://192.168.4.67:30000

示例2：tcp://192.168.4.67:30000;tcp://192.168.4.68:30000

支持配置主备柜台地址，使用;来分隔，当主柜台连接失败会轮询连接备柜台

Api自动重连也会轮询连接主备柜台

当同时配置了ManagerURL和QueryURL的情况下，Api连接manager成功之后会以配置项为准来连接柜台查询链路

在连接QFII柜台时，只需要配置该Url，ManagerURL和TradeURL配置为空即可

当同时配置了ManagerURL和QueryURL的情况下，Api连接manager成功之后会以配置项为准来连接柜台查询链路

参考配置：

QueryURL=

- 艾科报单链路连接地址

艾科报单链路连接地址，使用统一的格式tcp://地址:端口号

此参数只有发生了端口映射的情况下才需要配置，其他为空。配置格式如下示例，

示例：tcp://10.128.123.209:30005

场景：例如原报单链路连接地址为tcp://10.128.123.209:30005，由于组网限制，需要将该地址映射为tcp://192.168.0.100:33333

此时，将TradeURL参数改为：TradeURL=tcp://192.168.0.100:33333；如果没有此种情况只需要如下不填即可。

参考配置：

TradeURL=

在网络配置中，ManagerURL 和 QueryURL 要求必须配置其中的一个，当满足了以上要求后，剩下的配置项，如果配置即使用配置值，如果不配置就使用Api查询的默认值。显示配置是为了满足Api在外网模式下使用场景，详见配置文件对应字段说明。

Api参数配置

```
#####  
##### Api Parameter #####  
#####  
  
#当前Api支持柜台版本，默认支持一代柜台（2.5及以下版本），后续根据版本部署情况修改默认支持版本  
#当需要修改Api支持版本时，请确认当前艾科柜台版本  
#随意修改版本，或者格式异常时，可能会导致无法连接至艾科柜台或无法向艾科柜台报单  
#当柜台升级到二代柜台（3.0及以上版本）后，此参数需要修改为2（DependentCounterVersion=2）  
DependentCounterVersion=1
```

- 当前API支持柜台版本

当前Api支持柜台版本，默认支持一代柜台（2.5及以下版本），后续根据版本部署情况修改默认支持版本，当DependentCounterVersion=1时支持一代柜台（2.5及以下版本）

当需要修改Api支持版本时，请确认当前艾科柜台版本

随意修改版本，或者格式异常时，可能会导致无法连接至艾科柜台或无法向艾科柜台报单

当柜台升级到二代柜台（3.0及以上版本）后，此参数需要修改为2（DependentCounterVersion=2）

参考配置：

DependentCounterVersion=2

```
#####
#####  Api Parameter #####
#####

# Api与艾科柜台心跳间隔，此参数无特殊需求，不要进行修改
# 当API没有登录成功的时候使用此时间，登录成功后使用柜台和manager返回的心跳时间
# 示例：=6表示Api与柜台间每6s发送进行一次心跳交互,默认值，当api登录成功后以柜台的配置为主
HeartBeatInterval=6

# Api与艾科柜台心跳超时判定次数，此参数无特殊需求，不要进行修改
# 当API没有登录成功的时候使用此时间，登录成功后使用柜台和manager返回的心跳时间
# 示例：=3表示Api与柜台间超过三次心跳未收到，就判定tcp断链，需要重新登录来获取权限
HeartBeatTimeOutCnt=3

# Api报单链路warm开关，默认关闭，关闭此参数时，冷态报单时延可能会产生抖动
# 此参数在sfc网卡上使用会减轻链路数据处理压力，在其他网卡上使用时，会增加网卡数据处理压力，建议使用sfc网卡
# 示例：=1表示默认打开，=0表示关闭
# OrderWarm开关开启后，软件发送warm单，warm单能够使cpu处于活跃状态。
# 在发送正式的报单时，cpu可以尽量高的命中，以此提高报单的速度。
OrderWarm=0

# Api报单发送、报单回报线程绑定核心，在配置此参数前，需要先进行隔核操作
# 绑核只能在Api初始化的过程中进行修改，在单次登录、登出操作中无法修改绑定核心
# 示例：
# =-1,-1表示不进行绑核操作
# =3,4表示Api报单发送线程绑定核心3，报单回报线程绑定核心4
# 以上参数可以选择其中一个进行绑定(例如：=3,-1)，也可以选择都不绑定(例如：=-1,-1)，都不绑定会影响Api性能。
# 如果可用核心数不够，请尽量选择Api的报单发送线程进行绑定，这样可以尽可能保证报单速率
CpuCore=-1,-1

# Api登录软件节点后是否接收其他节点的回报信息，默认不接收
# 此参数对登录硬件节点的Api不生效
# 示例：
# =0表示默认关闭，=1表示打开
SoftRspRcv=0

# Api登录硬件节点后是否接收其他节点的回报信息，默认接收
# 此参数对登录软件节点的Api不生效
# 示例：
# =0表示关闭，=1表示打开，默认打开
# 仅部分券商支持配置为不抄送
FpgaRspRcv=1

# Api回报接收socket是否阻塞标记(无用，改用ApiMode参数)
# =0表示非阻塞socket，该模式下，cpu占用率高，时延相对较低
# =1表示阻塞socket，该模式下，cpu占用率低，时延相对较高
SocketBlockFlag=0
```

```
# 是否捕获异常，默认捕获
# =0表示不捕获异常
# =1表示捕获异常
# 说明，此处的异常是指程序中抛出的信号，Api中捕获SIGSEGV和SIGABRT两个信号，当关闭时，Api不处理任何信号
CaptureSignal=1

# 是否支持线程安全，默认不支持，注意当开启线程安全时会影响api的报单性能
# =0表示非线程安全
# =1表示线程安全
# 多线程使用同一个api实例且用户未对实例加锁时需开启，api内部保证功能接口的正常使用，部分系统接口不提供线程安全。
# 线程安全只能保证同一个api对象在多线程调用功能接口时不会出现线程冲突，部分系统接口在调用时无法保证安全性。
# 具体接口见头文件注释。
ThreadSafe=0
```

• 心跳配置

Api与艾科柜台心跳间隔，此参数无特殊需求，不要进行修改

示例：=6表示Api与柜台间每6s进行一次心跳交互

默认配置（无特殊情况，无需修改）

```
HeartBeatInterval=6
```

• 心跳超时判定次数

Api与艾科柜台心跳超时判定次数，此参数无特殊需求，不要进行修改

示例：=3表示Api与柜台间超过三次心跳未收到（Api超过3次未收到柜台心跳，或者柜台超过三次未收到Api心跳），就判定TCP断链，需要重新登录来获取权限

默认配置：（无特殊情况，无需修改）

HeartBeatTimeOutCnt=3

心跳机制说明，详见[心跳机制](#)

- **是否启用密文传输**

第一个1表示管理链路以密文传输，第二个1表示查询链路和交易链路以密文传输，为0时均表示明文传输。

ISMsgEncrypt=1,1

注意事项：

1. 使用之前，请与券商确认柜台版本是否支持密文传输；如果不支持密文传输，则务必关闭此选项；
2. 如果客户端启用密文传输，且柜台版本不支持；那么，客户端将会无法收到正确的应答消息，等待超时报后，断开连接；

默认以明文传输。

- **Api报单链路warm开关**

Api报单链路warm开关，默认打开，关闭此参数时，冷态报单时延可能会产生抖动

此参数在sfc网卡上使用会减轻链路数据处理压力，在其他网卡上使用时，会增加网卡数据处理压力，建议使用sfc网卡

示例：=1表示默认打开，=0表示关闭

OrderWarm=1

- **Api报单线程绑定核心**

Api报单回报线程绑定核心，在配置此参数前，需要先进行隔核操作

示例：

\=-1,-1表示不进行绑核操作

\=8,9表示Api报单发送线程绑定在cpu核心8上，报单回报线程绑定在cpu核心9上

默认配置：（默认不绑核）

CpuCore=-1,-1

- **Api登录软件节点后是否接收其他节点的回报信息**

Api登录软件节点后是否接收其他节点的回报信息，默认不接收

此参数对登录硬件节点的Api不生效

示例:

\=0表示默认关闭，=1表示打开

SoftRspRcv=0

- **Api登录硬件节点后是否接收其他节点的回报信息**

此参数对登录软件节点的Api不生效

示例:

\=0表示关闭，=1表示打开, 默认打开

仅部分券商支持配置为不抄送

FpgaRspRcv=1

- **Api回报接收socket是否阻塞标记**

\=0表示非阻塞socket，该模式下，cpu占用率高，时延相对较低

\=1表示阻塞socket，该模式下，cpu占用率低，时延相对较高

参考配置：

SocketBlockFlag=0

- **Api是否只连接管理中心（目前不对用户开放）**

\=0表示否定

\=1表示肯定

说明：正常情况下，此参数无需修改，若有特殊情况，需要只连接管理中心，则进行修改

默认配置：（无特殊情况，无需修改）

IsJustConnectManager=0

- **客户终端信息前缀**

中信证券使用，填写一个字符串信息，用于标识并上报给集中交易

PcPrefix=

- **是否捕获异常**

此处的异常是指程序中抛出的信号，Api中捕获SIGSEGV和SIGABRT两个信号，当关闭时，Api不处理任何信号。

\=0表示不捕获异常

\=1表示捕获异常

默认配置1：表示Api内部捕获异常。

参考配置：

CaptureSignal=1

- **是否支持线程安全**

\=0表示非线程安全

\=1表示线程安全

多线程使用同一个api实例且用户未对实例加锁时需开启，api内部保证功能接口的正常使用，部分系统接口不提供线程安全。

线程安全只能保证同一个api对象在多线程调用功能接口时不会出现线程冲突，部分系统接口在调用时无法保证安全性。

具体接口见头文件注释。

默认配置0：表示不支持线程安全。

参考配置：

ThreadSafe=0

```

# Api运行模式（高性能模式、普通模式）
# =0表示普通模式，在该模式下，api的cpu占用率和内存的消耗会有明显降低，报撤单速率会受到影响
# =1表示高性能模式，在该模式下，api会占用大量cpu和内存，报撤单速率有很大提高
# 默认Api运行在高性能模式下，如果没有特殊需求，不需要进行修改
# 修改了此参数后，需要重启Api
ApiMode=1

# 是否详细打印Api日志（详细日志包括请求的输入值、响应接口的输出值以及必要的Api日志，非详细的日志只有必要的Api日志），
# 其中必要的Api日志不会影响Api性能
# =0表示非详细打印
# =1表示详细打印
# 此参数默认关闭
# 注意：详细的Api日志会影响Api性能，建议在调试或者定位时打开此参数
SuperLog=0

# Api是否使用rtnTrade报文构造rtnOrder报文for_ci
# 该参数开启时，首先要保证柜台系统参数表第一百零四个参数处于关闭状态，
# 这时柜台不会在回rtnTrade报文时先推送rtnOrder报文，
# Api端在收到rtnTrade报文时会在Api端构造rtnOrder报文进行推送for_ci
# =0表示关闭for_ci
# =1表示开启for_ci
# 此参数默认关闭for_ci
# 注意：当开启该参数时，会对Api的性能产生影响：开启参数时如果柜台的参数没有关闭，
# 会导致Api端收到两条同样的rtnOrder报文，可能会对客户端程序产生影响for_ci
CreateRtnOrderByRtnTrade=0

# Api收发线程是否分开
# 参数开启时，api的收发线程分离，用户调用报撤单接口所发送的消息会直接通过网卡发往柜台；
# 在参数开启时，客户需要自己隔核并将调用reqInsertOrder、reqCancelOrder接口的线程绑核，
# 另外使用CpuCore参数的第二个变量将回报线程绑核，这样能保证api高性能运行。
# 参数开启时，OrderWarm、ApiMode以及CpuCore的第一个参数无效
# 此参数重启生效
# 示例：
# =0表示关闭，=1表示开启；默认关闭
RecvSendDetach=0

```

• Api运行模式

\=0表示普通模式，在该模式下，在该模式下，api内存的消耗会有明显降低，同时报单的速率会受到影响

\=1表示高性能模式，在该模式下，api会占用较多内存，同时报单的速率会有提升

默认Api运行在高性能模式下，如果没有特殊需求，不需要进行修改

修改了此参数后，需要重启Api

默认配置：（无特殊情况，无需修改）

如果 RecvSendDetach=1，则该参数不生效。

ApiMode=1

• 详细日志打印

是否详细打印Api日志（详细日志包括请求的输入值、响应接口的输出值以及必要的Api日志，非详细的日志只有必要的Api日志），其中必要的Api日志不会影响Api性能

\=0表示非详细打印

\=1表示详细打印

此参数默认关闭

注意：详细的Api日志会影响Api性能，建议在调试或者定位时打开此参数

SuperLog=0

默认配置：（无特殊情况，无需修改）

SuperLog=0

• 是否使用rtnTrade报文构造rtnOrder报文

该参数开启时，首先要保证柜台系统参数表第一百零四个参数处于关闭状态，这时柜台不会在回 rtnTrade 报文时先推送 rtnOrder 报文，Api 端在收到 rtnTrade 报文时会在 Api 端构造 rtnOrder 报文进行推送

\=0 表示关闭

\=1 表示开启

此参数默认关闭

注意：当开启该参数时，会对 Api 的性能产生影响；开启参数时如果柜台的参数没有关闭，会导致 Api 端收到两条同样的 rtnOrder 报文，可能会对客户程序产生影响

默认配置：（无特殊情况，无需修改）

CreateRtnOrderByRtnTrade=0

- **Api 收发线程是否分开**

参数开启时，api 的收发线程分离，用户调用报撤单接口所发送的消息会直接通过网卡发往柜台

在参数开启时，将调用 reqInsertOrder、reqCancelOrder 接口的线程绑核，另外使 CpuCore 参数的第二个变量将回报线程绑核，这样能设置 api 运行在高性能模式下

参数开启时，OrderWarm、ApiMode 以及 CpuCore 的第一个参数无效

此参数重启生效

示例：

\=0 表示关闭，=1 表示开启；默认关闭

默认配置：（无特殊情况，无需修改）

RecvSendDetach=1

- **使用 TcpDirect 加速**

```
#交易链路sfc网卡名称,用于报单加速  
#若不配置或环境异常采用原生socket通讯  
#RecvSendDetach配置为1时,此参数不生效  
#此参数使用时请与it确认TcpDirect版本  
SolarfareTradeEthName=
```

若不配置或环境异常采用原生 socket 通讯

RecvSendDetach 配置为 1 时，此参数不生效

使用前请与艾科确认 TcpDirect 版本

默认配置为空，如需使用 TcpDirect 技术加速，可配置交易链路的 sfc 网卡名称

SolarfareTradeEthName=

- **艾科 UDP 报单链路连接地址**

艾科 UDP 报单地址，使用统一的格式 `udp://地址:端口号`

若此参数为空，则使用柜台查询链路登录返回的地址，若此参数为合法地址时，则使用配置的值。地址、端口映射同理。配置格式如下示例：

示例：`udp://10.128.123.209:30005`

场景：例如原报单链路连接地址为 `udp://10.128.123.209:30005`，由于组网限制，需要将该地址映射为 `udp://192.168.0.100:33333`。此时，将 `UdpTradeURL` 参数改为：

`UdpTradeURL=udp://192.168.0.100:33333`；如果没有此种情况只需要如下不填即可。

场景：例如从管理中心查询到的地址为 `udp://192.168.4.67:29999`，现需要修改默认地址。此时，将 `UdpTradeURL` 参数改为：`UdpTradeURL=udp://192.168.4.216:29999`，如果没有这种需求，不填即可。

`UdpTradeURL=`

一般情况下，可以不用显式设置该配置选项。如果 `TradeProtocol` 不等于2，则该选项不生效。

- **交易使用的协议类型**

启用UDP交易之后，还需要根据当前柜台版本是否支持UDP交易，来决定是否启用UDP交易方式。

`TradeProtocol=`

配置参数取值范围：1-表示启用TCP交易，2-表示启用UDP交易；默认值为 1。

- **是否启用EFVI方式进行UDP交易**

该选项依赖于 `TradeProtocol` 选项，只有启用了UDP交易之后（`TradeProtocol=2`），该选项的值才生效，否则不生效（默认不启用）。

若启用EFVI方式进行报单，还需要配置 `SolarfareTradeEthName` 参数，指定报单的网卡。

`EnableUdpEfviTrade=`

配置参数取值范围：1-表示启用EFVI方式的UDP交易，0-表示不启用EFVI方式的UDP交易；默认值为 0。

- **是否建立交易链路**

该选项用来控制是否建立交易链路。如果不建立交易链路，无论是TCP方式还是UDP方式，都不会分配对应的资源，也无法通过交易接口，发送报撤单。

`EstablishTradeLink=`

配置参数取值范围：1-表示建立交易链路，0-表示不建立交易链路；默认值为 1。

不建立交易链路时的注意事项：1) 如果不建立交易链路，用户需要自己负责交易链路的建立操作，建立交易链路成功后，按约定发送心跳。2) 如果不建立交易链路，用户成功建立交易链路后，若发生断链；此时，柜台不再允许二次建立交易链路。需要调用`reqLogout()`接口退出后，重新调用`reqLogin()`接口，才允许建立交易链路；3) 如果不建立交易链路，且使用FPGA通道交易（`SubClientIndex=0`）用户成功建立交易链路后，需要调用`reqInitTrader()`接口初始化交易链路；4) 如果不建立交易链路，且使用软通道交易（`SubClientIndex!=0`）用户成功建立交易链路后，需要自己构造`InitTrader`请求消息，初始化交易链路，否则不允许通过软通道进行交易；

- **TCP链路首次建链超时时间**

TCP链路首次建链超时时间，单位秒，范围[1,10]。默认为5秒；建议使用默认值。

`TcpConnectTimeout=5`

- **TCP链路断链重连超时时间**

TCP链路异常断链后重连超时时间，单位秒，范围[1,10]。默认为3秒；建议使用默认值。

`TcpReconnectTimeout=3`

登录配置

```
#####  
##### Login configurations #####  
#####  
  
# 交易终端软件名称及版本，中信证券最大支持长度40，其他券商最大支持长度29，超过按券商所支持最大值截取  
AppID=  
  
# 流水重构标志，期权使用，上交和深交柜台不使用该字段，无需做修改  
# =0表示不进行流水重构  
# =1表示只进行资金流水重构；  
# =2表示只进行报文流水重构；  
# =3表示资金和报文流水重构；  
FlowRebuildFlag=0  
  
# 委托方式校验字段  
# 当有报单委托方式校验需求时填写，没有时无需修改(填写时具体取值需要和券商沟通)  
OperWay=1  
  
# 正式站点前缀，目前中信用(填写时具体取值需要和券商沟通)，其他券商不配置  
PcPrefix=  
  
# 营业部id，目前国泰证券使用  
BusinessCode=  
  
# 当链接超时之后是否需要重连 0=不自动重连 1=自动重连  
AutoReLogin=0  
  
# 自动重连后是否触发相关回调函数，配置自动重连后此参数生效  
# =0 不返回， =1 返回， 默认为0  
AutoReLoginRspSwitch=0  
  
# 第一次登录manager如果连接失败是否需要一直重复连接manager  
# =0 不会一直连接manager，连接主备manager3秒之后结束  
# =1 会一直循环的连接主备manager，直到手动结束进程  
LoopRepeatConnect=0  
  
#是否开启登录的时候进行用户名和密码加密  
#api登录manger和柜台的时候是否发送的账户和密码是密文  
# =0 表示关闭  
# =1 表示打开  
# 默认关闭  
#仅部分券商支持，使用前请确认  
openEncryption=0  
  
#公网信息  
#使用;号进行分割  
#目前支持配置公网IP为IIP，和端口号为IPORT  
#例如SuperviseExtraInfo=IIP=127.0.0.1;IPORT=50318  
#需要按照规定的格式来填写，IP和PORT要是正确的IP和PORT  
#IIP=127.0.0.1;IPORT=50318或者IPORT=50318;IIP=127.0.0.1都行  
#字段支持最大长度为128位  
#仅部分券商支持，使用前请确认  
SuperviseExtraInfo=
```

```
#版本校验，XeleSecuritiesTraderApi.h头文件位置  
#校验当前.so内的版本是否和头文件中的版本号匹配  
#填写值了会去获取指定目录下的相关文件,为空不校验  
#填写绝对路径，默认值为空  
#例如 TraderApiPath=/home/xele/api/  
ApiIncludePath=
```

```
#此字段填写后,PcPrefix, APPID, SuperviseExtraInfo配置项均不生效  
#客户登录时填写的完整监管信息,此信息不做校验,直接透传至柜台落库  
#最大支持到384字节,不支持中文格式字符,默认为空  
#柜台默认支持长度255，若超过则需要跟券商确认柜台是否支持  
#仅部分券商支持，使用前请确认  
AllSuperviseInfo=  
~
```

- 交易终端软件名称及版本

参考配置：(默认为空，由交易客户填写)

AppID=

- 流水重构标志（详见[流水重构（期权独有）](#)）

流水重构标志，期权使用，上交和深交柜台不使用该字段，无需做修改

\=0表示不进行流水重构

\=1表示只进行资金流水重构;

\=2表示只进行报文流水重构;

\=3表示资金和报文流水重构;

参考配置：

FlowRebuildFlag=0

- **委托方式校验字段(具体取值,请咨询券商)**

当有报单委托方式校验需求时填写，没有时无需修改

参考配置：

OperWay=1

- **正式站点前缀字段(具体取值,请咨询中信证券)**

没有前缀或者非中信券商，可以配置为空或者没有这个配置项，中信证券需要填站点前缀

参考配置：

PcPrefix=ZX_FPGA

- **机构编码（具体取值，看上场数据）**

没有前缀或者非国泰君安，可以配置为空，国泰君安需要配置正确的营业部代码否则会登录失败

参考配置：

BusinessCode=12345

- **API自动重连标志**

API异常断连后自动重连功能

\=0表示api异常断连后不自动重连

\=1表示api异常断连后自动重连

参考配置：

AutoReLogin=0

- **API自动重连打开后是否返回相关回调开关**

自动重连后是否触发相关回调函数，配置自动重连后此参数生效

\=0 不返回，=1 返回，默认为0

参考配置：

AutoReLoginRspSwitch=0

- **API第一次连接manager失败可以一直循环登录manager**

第一次登录manager如果连接失败是否需要一直重复连接manager

\=0 不会一直连接manager，连接主备manager3秒之后结束

\=1 会一直循环的连接主备manager，直到手动结束进程

默认为0s

参考配置：

`LoopRepeatConnect=0`

- **是否打开加密登录**

api登录manger和柜台的时候是否发送的账户和密码是密文

示例：

\=0表示关闭，=1表示开启；默认关闭

仅部分券商支持，使用前请确认

`openEncryption=0`

- **公网信息**

使用；号进行分割

目前支持配置公网IP为IIP，和端口号为IPORT

例如SuperviseExtraInfo=IIP=127.0.0.1;IPORT=50318

仅部分券商支持，使用前请确认

`SuperviseExtraInfo=`

- **版本校验**

版本校验, XeleSecuritiesTraderApi.h头文件位置

校验当前.so内的版本是否和头文件中的版本号匹配

填写值了会去获取指定目录下的相关文件,为空不校验

填写绝对路径, 默认值为空

例如 ApiIncludePath=/home/xele/api/

`ApiIncludePath=`

- **完整监管信息**

此字段填写后,PcPrefix, APPID, SuperviseExtraInfo配置项均不生效

客户登录时填写的完整监管信息,此信息不做校验,直接透传至柜台落库

最大支持到384字节,不支持中文格式字符,默认为空

柜台默认支持长度255，若超过则需要跟券商确认柜台是否支持仅部分券商支持，使用前请确认

仅部分券商支持，使用前请确认

`AllSuperviseInfo=`

TraderAPI用户操作流程

API连接方式

目前证券API通过Manager连接柜台，需通过api_config.txt配置文件进行配置（配置详见：[API 配置文件](#)）。

1. API通过管理中心连接到柜台



2. API直接连接到柜台

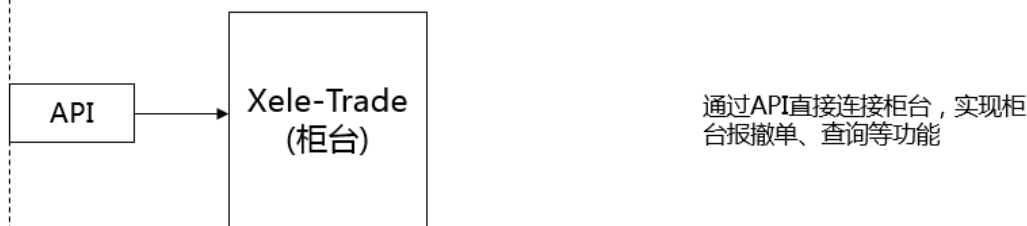


图 API连接方式

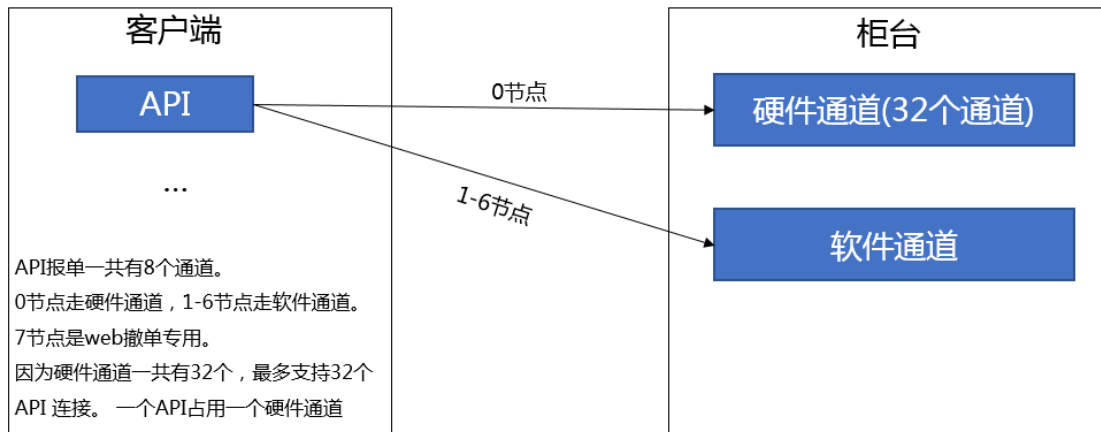
API通过管理中心连接到柜台(目前上交、深交股票支持该方式；上交期权暂不支持)

API登录到Manager(管理中心)，根据响应信息找到对应柜台地址，API根据柜台地址连接Xele-Trade柜台，API与柜台连接成功后，可实现Xele-Trade柜台报撤单、查询以及管理中心资金调拨、沪深柜台资金查询功能。

【注意】

1. 该连接方式，用户只需一次登录Manager管理中心，即可完成Manager系统和柜台系统的登录。
2. 该连接方式，是通过API连接柜台进行报撤单操作，报撤单操作不经过Manager管理中心。
3. Xele-Trade-Securities柜台可以最多支持32个API连接，Manager支持8*32个API连接

软件报单通道和硬件报单通道



目前柜台3.0版本支持软件报单和硬件报单。

柜台支持多点登录，API报单一共有7个（0-6）节点。交易客户可以选择0-6中任意一个节点登录。

0节点走硬件通道，1-6节点走软件通道，7节点是web撤单专用

硬件通道就是直连fpga，软件通道就是fpga之前加个软件模块。

登录柜台inittrade如果返回交易端口为30010 通常连接的是软件通道

软件通道的延时比较大，而且不支持大并发报单，建议报单控制在1s 50单以内

如果客户想要多个硬件通道报单，可以向券商申请，比如把1节点也改为硬件通道，这样用户可以使用0和1节点两个通道极速报单

【tips】

==用户使用softOrder 报单，只能用来应急，不能用来并发，不能高频==

登录

登录流程

目前艾科一共有四种证券柜台：沪深股票、沪深期权

登录柜台的方式：通过Manager管理中心登录柜台（期权暂不支持）、直接登录Xele-Trade-Securities柜台（上交期权支持），这两种登录方式API 提供了统一的登录接口reqLogin。

- 证券：通过Manager管理中心登录柜台（上交股票、深交股票）

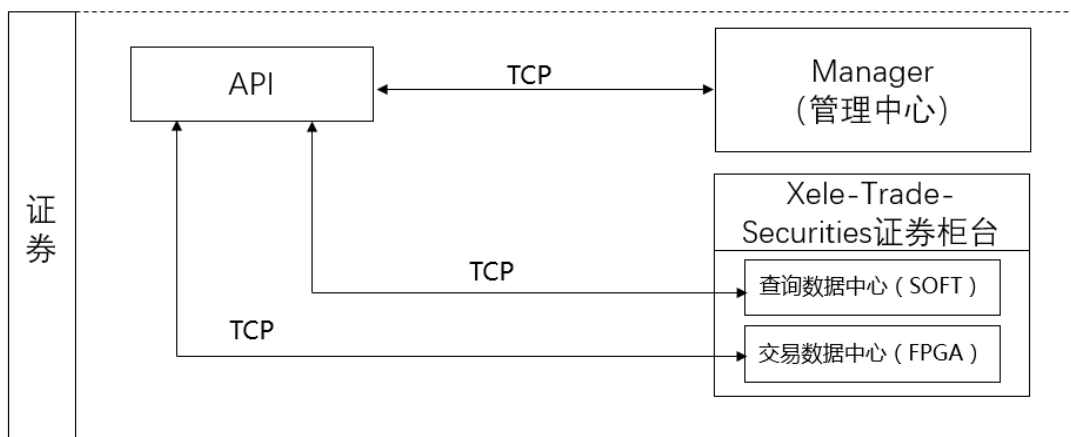


图 证券登录流程图

如上图所示：

通过API的api_config.txt配置文件，可设置通过Manager（管理中心）登录到柜台，API会收到onRspLoginManager的Manager登录应答，该应答包含柜台的URL地址信息，根据该URL地址，连接到对应的Xele-Trade-Securities柜台。

Manager登录后，登录柜台需要先连柜台的查询数据中心(SOFT)，再连接柜台的交易数据中心(FPGA)。

- a) 登录柜台查询数据中心(SOFT)成功后，API会收到onRspLogin的柜台登录应答
- b) 连接柜台交易数据中心(FPGA)成功后，API会收到onRspInitTrader的添加交易链路应答

【注意】

目前上交股票、深交股票支持通过Manager（管理中心）连接到柜台的方式，上交期权暂不支持。

通过先登录Manager后登录Xele-Trade-Securities柜台，正常成功登录情况下API会收到三个回报：onRspLoginManager、onRspLogin、onRspInitTrader

• 期权：直接登录柜台（期权）

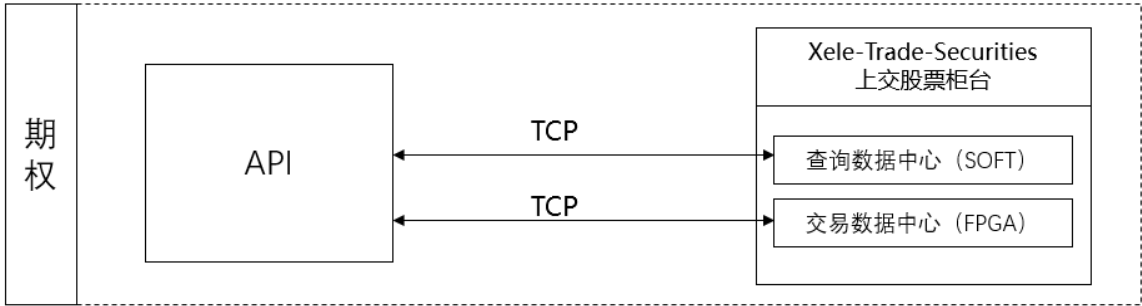


图 期权登录流程图

登录时序图

通过管理中心连接到柜台（上交股票、深交股票）

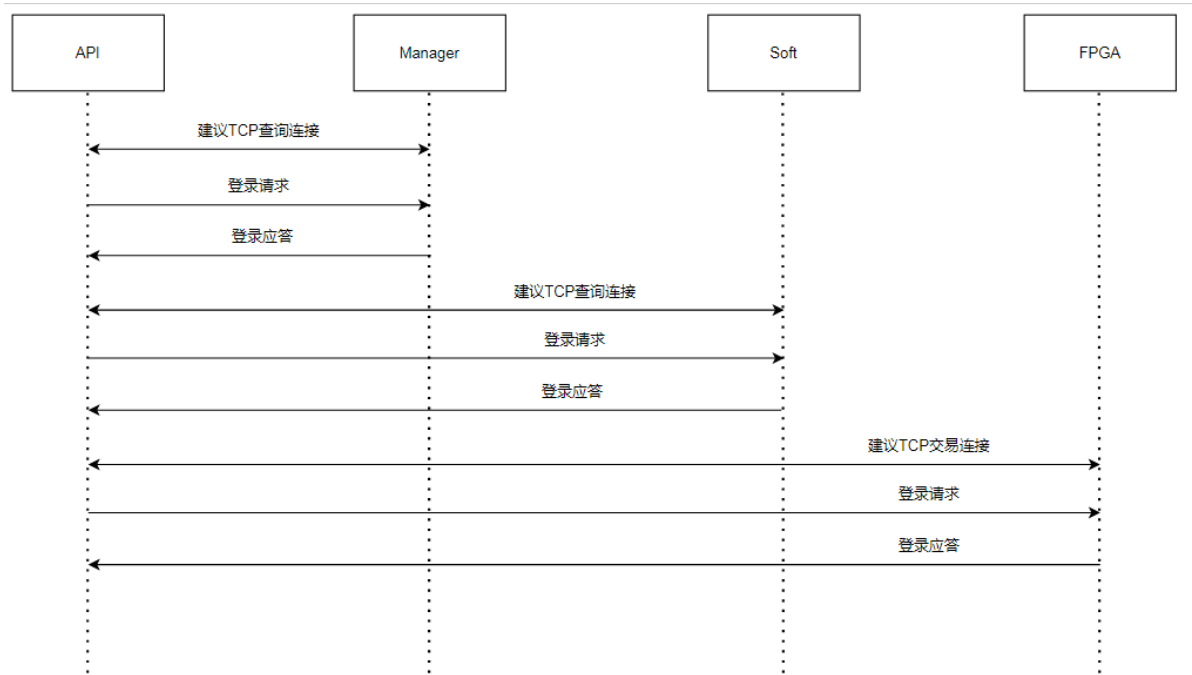


图 通过Manager登录柜台时序图

1. API登录Manager需要先建立TCP连接
2. API发起登录请求

3. 登录请求后Manager会有响应信息，该响应包含柜台的地址，收到该响应后，则拥有管理中心的操作权限
4. API根据上一步骤拿到的柜台地址，与柜台建立TCP连接
5. API查询连接请求
6. API查询连接响应，收到该响应后可以进行相关查询操作
7. 交易连接

柜台SOFT、FPGA说明

Xele-Trade-Securities交易系统包括查询数据中心(SOFT)和交易数据中心(FPGA)。

查询数据中心 (SOFT)：主要是提供相关报单、合约、资金等查询功能，使用的是TCP数据链路；该部分功能用户需要使用XeleSecuritiesTraderAPI来实现。

交易数据中心 (FPGA)：主要是处理报、撤单以及接收相关回报等操作。

【注意】对于Xele-Trade-Securities交易系统来讲，用户需要分别先登录查询数据中心，然后登录交易数据中心，只有两个中心都登录成功后，才可以进行报撤单以及接收回报等操作。

多点登录

Xele-Trade-Securities上交股票、深交股票、上交期权系统支持每个用户多点登录，默认0-6节点登录，同一个用户不同节点每次创建1个API实例，初始化完成后即可以进行一次登录。多点登录有以下特点：

- 每次需要登录成功后，才可以进行报撤单以及查询操作；
- 查询数据流的下行报文只发送给发起请求的用户连接；
- 交易数据流的下行报文是发送给该用户的所有连接；
- 查询数据流或者交易数据流下行如果断开，API均会自动重连；

用户登录代码示例（详见示例代码：“demo_main.cpp”）

```
///艾科管理中心登录应答,当只有登录管理中心的需求时，收到该回报即可进行管理中心相关接口操作
void onRspLoginManager(CXeleRspUserLoginManagerField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) override{
    if(pRspInfo->ErrorID == 0){
        canFundTransfer = true;
        PRINT_INFO("now can use manager interface");
    } else{
        PRINT_INFO("login manager error,ErrID[%d],ErrMsg[%s]",pRspInfo-
>ErrorID,pRspInfo->ErrMsg);
    }
};

///艾科柜台登录应答,当需要管理中心接口可用，但是只需求艾科柜台查询接口可用时，收到该回报即可
进行操作
void onRspLogin(CXeleRspUserLoginField *pRspField, CXeleRspInfo *pRspInfo, int
nRequestID, bool bIsLast) override{
    if(pRspInfo->ErrorID == 0){
        canQuery = true;
        maxUserLocalID = pRspField->MaxUserLocalID;
        maxUserLocalID += 1;
        PRINT_INFO("now can use query interface");
    } else{
        PRINT_INFO("login counter error,ErrID[%d],ErrMsg[%s]",pRspInfo-
>ErrorID,pRspInfo->ErrMsg);
    }
};
```

```

    ///添加交易链路应答,当收到该回报时,标记艾科柜台报、撤单接口可用
    void onRspInitTrader(CXeleRspInitTraderField *pRspField, CXeleRspInfo
    *pRspInfo, int nRequestID, bool bIsLast) override{
        if(pRspInfo->ErrorID == 0){
            canOrder = true;
            PRINT_INFO("now can use order interface");
        } else{
            PRINT_INFO("create order link error,ErrID[%d],ErrMsg[%s]",pRspInfo-
            >ErrorID,pRspInfo->ErrMsg);
        }
    };
    ///main函数中的登录
    ret = pUserApi-
    >reqLogin("./api_config.txt",accountID.data(),password.data(),g_RequestID++);
    if(ret){
        PRINT_INFO("login error,ret [%d]",ret);
        exit(0);
    }
}

```

查询

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。

【注意】

==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; ==

```

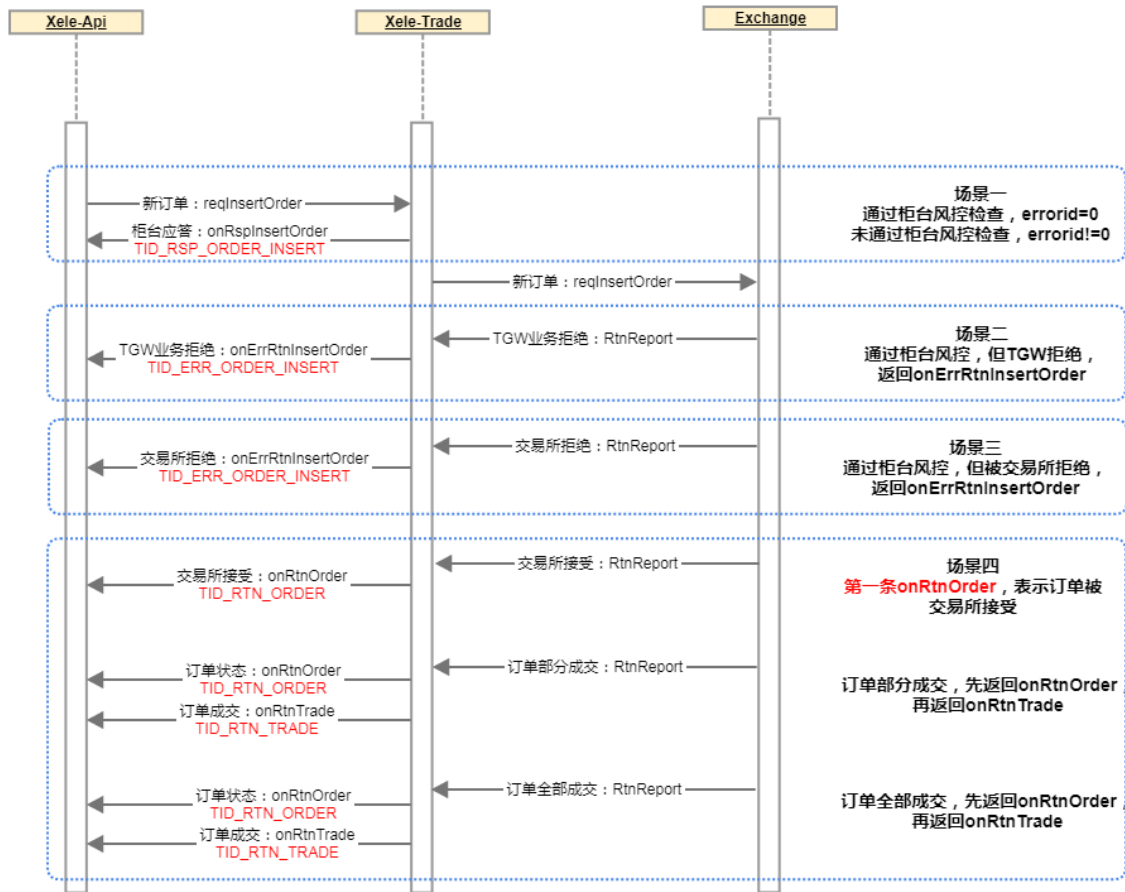
    ///证券持仓查询应答
    void onRspQryPosition(CXeleRspQryStockPositionField *pRspField, CXeleRspInfo
    *pRspInfo, int nRequestID, bool bIsLast) override{
        PRINT_INFO("Test reqQryPosition pass");
        if(pRspInfo->ErrorID == 0){
            ///demo中使用持仓查询结果来更新本地持仓
            AccountPosition accountPosition{};
            accountPosition.AvailablePosition = pRspField->AvailablePosition;
            accountPositionMap.insert(std::make_pair(pRspField-
            >SecuritiesID,accountPosition));
            ///demo test:查询最后一个持仓的合约信息
            if(bIsLast){
                usleep(200000);
                CXeleReqQrySecuritiesField reqQrySecuritiesField{};
                memset(&reqQrySecuritiesField,0,sizeof(CXeleReqQrySecuritiesField));
                memcpy(reqQrySecuritiesField.SecuritiesID,pRspField-
                >SecuritiesID,strlen(pRspField->SecuritiesID));

                PRINT_INFO("Test reqQrySecurities");
                pUserApi->reqQrySecurities(reqQrySecuritiesField,g_RequestID++);
            }
        } else{
            PRINT_INFO("onRspQryPosition Error,ErrMsg: %s",pRspInfo->ErrMsg);
        }
    }
}

```

```
};
```

报单



特别注意：

==场景四中，订单部分成交或订单全部成交时，先返回的onRtnOrder由系统参数 SECURITY_PARAM_IS_RSP_RTNORDER_BY_RTNTRADE控制，若配置为0，则不发送；若配置为1，则发送。若无此参数则不发送。==

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：

1. 不管哪种场景，柜台系统都会发送报单响应onRspInsertOrder报文，当未通过柜台系统风控检查时，会有错误id；
2. 当报单未通过交易所前置风控或者交易所风控时，柜台系统会发送错误回报onErrRtnInsertOrder报文；
3. 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；
4. 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；

• 报单单号管理：

报单时需要维护的单号字段有UserLocalID和OrderSysID。

1. 【报单请求】reqInsertOrder接口参数的结构体字段中：

UserLocalID是用户自己管理的报单编号，当该字段是单调递增时，也可以进行撤单，具体可以咨询券商，进行配置；

2. 【报单响应】 onRspInsertOrder接口参数的结构体字段中：

UserLocalID是柜台系统返回给用户自己管理的报单编号；

OrderSysID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依据之一；

3. 【报单回报】 onRtnOrder接口参数的结构体字段中：

OrderSysID是柜台系统生成的报单编号；

OrderExchangeID 是交易所生成的报单编号；

4. 【成交回报】 onRtnTrade接口参数的结构体字段中：

OrderSysID是柜台系统生成的报单编号；

OrderExchangeID是交易所生成的报单编号；

ExecID是交易所生成的成交编号；

举例说明：

报单请求：用户在调用reqInsertOrder中，填写了自己的报单编号UserLocalID=0001；

报单响应：在onRspInsertOrder中返回用户自己的报单编号UserLocalID=0001，并且还返回值柜台生成的系统报单编号OrderSysID=000009，如果用户进行撤单操作，OrderSysID在任何情况下都可以进行撤单，但使用UserLocalID字段撤单时（参考下方撤单描述），需要券商在柜台端进行配置，具体咨询券商。

报单示例代码（详见示例代码“demo_main.cpp”）

```
//demo test:将最后一个持仓合约卖出
CXeleReqOrderInsertField orderInsertField{};
memset(&orderInsertField,0,sizeof(CXeleReqOrderInsertField));
orderInsertField.UserLocalID = maxUserLocalID++;
memcpy(orderInsertField.SecuritiesID,pRspField->SecuritiesID,strlen(pRspField->SecuritiesID));
orderInsertField.Direction = XELE_ORDER_SELL;
orderInsertField.LimitPrice = pRspField->PreSettlePrice;
orderInsertField.Volume = 1000;
orderInsertField.OrderType = XELE_LIMIT_PRICE_TYPE;
orderInsertField.TimeCondition = XELE_TIMEINFORCE_TYPE_GFD;
orderInsertField.SecuritiesType = pRspField->SecuritiesType;
memcpy(orderInsertField.BusinessUnit,"demo test",9);
orderInsertField.Operway = API_OPERWAY;

PRINT_INFO("Test reqInsertOrder");
pUserApi->reqInsertOrder(orderInsertField,g_RequestID++);

accountPositionMap[pRspField->SecuritiesID].AvailablePosition -=
orderInsertField.volume;
} else{
    PRINT_INFO("onRspQrySecurities Error,Errmsg: %s",pRspInfo->ErrorMsg);
}

///报单应答
void onRspInsertOrder(CXeleRspOrderInsertField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) override{
    //当收取到onRspInsertOrder消息时，表明该条报单信息已经被艾科柜台接收，正在等待处理
    if(pRspInfo->ErrorID == 0){
        //本地记录下当前报单的状态，方便在需要时进行处理
```

```

        orderManagerMap[pRspField->OrderSysID].OrderStatus = ODRSTAT_REPORTED;
    } else{
//当报单应答中发生ErrID不为0的情况，表明该条报单没有通过艾科柜台的风控检查
//常见的异常为资金或者持仓风控未通过
//柜台的异常回报信息（CXelerRspInfo）中有该次异常的错误码以及具体原因
        PRINT_INFO("onRspInsertOrder ErrMsg: %s",pRspInfo->ErrorMsg);
        if(orderManagerMap.find(pRspField->OrderSysID) != orderManagerMap.end()){
            //报单出现异常后，修改本地的报单状态
            orderManagerMap[pRspField->OrderSysID].OrderStatus = ODRSTAT_ERROR;
        }
        //报单出现异常后，恢复本地的持仓信息，恢复本地的资金信息
updateAccountInfo(pRspField->AccountID,pRspField->SecuritiesID,pRspField->
>Direction,pRspField->Volume,pRspField->LimitPrice);
        canOrder = false;
        canQuery = false;
    }
};

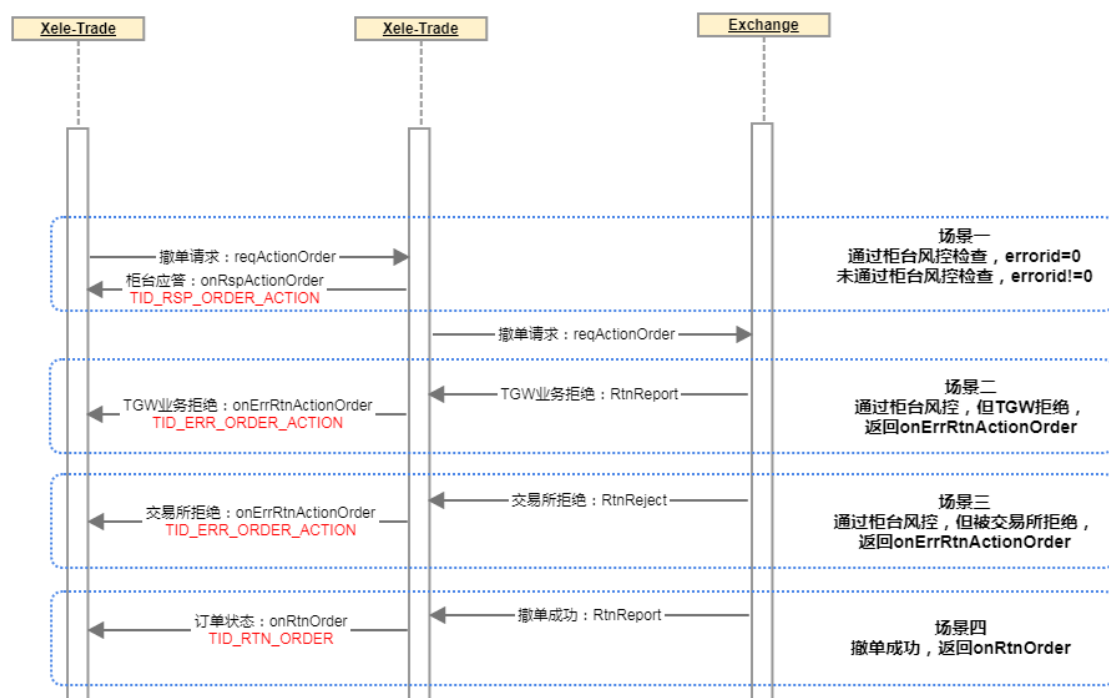
///报单错误回报
void onErrRtnInsertOrder(CXelerRspOrderInsertField *pRspField, CXelerRspInfo
*pRspInfo, int nRequestID, bool bIsLast) override{
//当收该条回报信息时，表明该条报单没有通过交易所风控检查，出现了异常
//常见的异常为资金或者持仓风控未通过
//柜台的异常回报信息（CXelerRspInfo）中有该次异常的错误码以及具体原因
    PRINT_INFO("ErrRtnInserOrder ErrMsg: %s",pRspInfo->ErrorMsg);
//本地记录下当前报单的状态，方便在需要时进行处理
    orderManagerMap[pRspField->OrderSysID].OrderStatus = ODRSTAT_ERROR;

    //报单出现异常后，恢复本地的持仓信息，恢复本地的资金信息
updateAccountInfo(pRspField->AccountID,pRspField->SecuritiesID,pRspField->
>Direction,pRspField->Volume,pRspField->LimitPrice);

    canOrder = false;
    canQuery = false;
};

```

撤单



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：

1. 不管哪种场景，柜台系统都会发送撤单响应onRspCancelOrder报文, 当未通过柜台系统风控检查时，会有错误id;
2. 当撤单未通过交易所前置风控或者交易所风控时，柜台系统会发送错误回报onErrRtnCancelOrder报文;
3. 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UserLocalID、OrigSysID、OrderSysID(该三个字段为主动撤单编号)。

OrigUserLocalID、OrigOrderSysID、OrigStrOrderSysID(该三个字段为被撤单的编号，柜台2.5及以上版本才使用)

1. 【撤单请求】reqCancelOrder接口参数的结构体字段中：

UserLocalID是用户自己管理的撤单编号

OrigSysID柜台系统中待撤单编号（对应onRspInsertOrder中的OrderSysID）；

OrigUserLocalID表示待撤单的用户本地编号。当服务端校验报单的用户LocalID单调递增时，支持FPGA通道通过OrigUserLocalID撤单，不支持软件通道使用OrigUserLocalID撤单。

2. 【撤单响应】onRspCancelOrder接口参数的结构体字段中：

UserLocalID是柜台系统返回给用户自己管理的撤单编号

OrigSysID是柜台系统中待撤单编号（对应onRspInsertOrder中的OrderSysID）；

OrigUserLocalID是用户维护的待撤单编号（对应onRspInsertOrder中的UserLocalID）；

OrderSysID是柜台系统生成的本次撤单的系统编号

3. 【撤单回报】

onRtnOrder：接口参数的结构体字段中

OrderSysID：被撤单的OrderSysID编号

OrderExchangeID：交易所生成的撤单编号；

举例说明：

1. 方式一：

报单请求：用户在调用reqInsertOrder中，填写了自己的报单编号UserLocalID=0001；

报单响应：在onRspInsertOrder中会返回用户自己的报单编号UserLocalID=0001，并且还返回柜台生成的系统报单编号OrderSysID=000009；

撤单请求：调用reqCancelOrder中，填写了自己管理的撤单编号UserLocalID=0003，以及撤单的系统编号OrderSysID= 000009；

撤单响应：在onRspCancelOrder中，返回了用户自己管理的撤单编号UserLocalID=0003；

柜台系统生成的撤单系统编号OrderSysID=0004；对应的待撤报单编号OrigSysID=000009；

2. 方式二：（需要柜台系统开启校验UserLocalID字段递增）

报单请求：用户调用reqInsertOrder中，填写了自己的报单编号UserLocalID=0001；

报单响应：在onRspInsertOrder中会返回用户自己的报单编号UserLocalID=0001，并且还返回柜台生成的系统报单编号OrderSysID=000009；

撤单请求：调用reqCancelOrder中，填写了自己管理的撤单编号UserLocalID=0003，以及撤单的用户本地编号OrigUserLocalID=0001，并且OrigOrderSysID=0000；（此时表示使用用户本地报单编号进行撤单）；

撤单响应：在onRspCancelOrder中，返回了用户自己管理的撤单编号UserLocalID=0003；柜台系统生成的撤单系统编号OrderSysID=0004；对应待撤报单编号OrigSysID=000009，OrigUserLocalID=0001；

撤单示例代码（详见示例代码“demo_main.cpp”）

```
//当进行报撤单请求时，需要对请求数据进行清零操作
memset(&actionField,0,sizeof(CXeleReqOrderActionField));
actionField.UserLocalID = maxUserLocalID++;
actionField.OrigSysID = pRspField->OrderSysID;
//使用UserLocalID进行撤单的方式
//actionField.OrigSysID = 0;
//actionField.OrigUserLocalID = pRspField->UserLocalID
memcpy(actionField.BusinessUnit,"Taction",8);
actionField.OwnerType = XELE_OWNER_PERSONAL_TYPE;
actionField.Operway = API_OPERWAY;

PRINT_INFO("Test reqCancelOrder");
pUserApi->reqCancelOrder(actionField,g_RequestID++);
}
} else{
    PRINT_INFO("onRspQryOrder Error,Errmsg: %s",pRspInfo->ErrorMsg);
}
};

///撤单应答
void onRspCancelOrder(CXeleRspOrderActionField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) override{
//当接收到onRspCancelOrder消息时，表明该条撤单信息已经被艾科柜台接收，正在等待处理
if(pRspInfo->ErrorID == 0){
//本地记录下当前报单的状态，方便在需要时进行处理
    orderManagerMap[pRspField->OrderSysID].OrderStatus = ODRSTAT_REPORTED;
```

```

    //当撤单请求通过艾科柜台和交易所风控检查后，后续被撤单的信息会通过onRtnOrder接口返回，此时
    可以进行本地的持仓和资金管理
    } else{
    //当撤单应答中发生ErrID不为0的情况，表明该条撤单没有通过艾科柜台的风控检查
    //常见的异常为被撤单不存在，可能已经成交或者被撤单信息填写错误
    //柜台的异常回报信息（CXelerRspInfo）中有该次异常的错误码以及具体原因
    PRINT_INFO("onRspCancelOrder ErrMsg: %s",pRspInfo->ErrorMsg);
    if(orderManagerMap.find(pRspField->OrderSysID) != orderManagerMap.end()){
        orderManagerMap[pRspField->OrderSysID].OrderStatus = ODRSTAT_ERROR;
    }

    //撤单出现异常后，表明该条撤单信息未被响应，被撤单信息未发生任何变化，故不需要对本地维护的资
    金和持仓信息进行更新
    }
};

///撤单错误回报
void onErrRtnCancelOrder(CXelerRspOrderActionField *pRspField, CXelerRspInfo
*pRspInfo, int nRequestID, bool bIsLast) override{
    //当收该条回报信息时，表明该条撤单没有通过交易所风控检查，出现了异常
    //常见的异常为被撤单不存在，可能已经成交或者被撤单信息填写错误
    //柜台的异常回报信息（CXelerRspInfo）中有该次异常的错误码以及具体原因
    PRINT_INFO("onErrRtnCancelOrder ErrMsg: %s",pRspInfo->ErrorMsg);
    //本地记录下当前撤单的状态，方便在需要时进行处理
    orderManagerMap[pRspField->OrderSysID].OrderStatus = ODRSTAT_ERROR;
    //撤单出现异常后，表明该条撤单信息未被响应，被撤单信息未发生任何变化，故不需要对本地维护的
    资金和持仓信息进行更新
};

```

TradeAPI开发示例

下列是我们打包发给您的“example_stock”代码，由于文档篇幅有限，仅摘录主函数，供您参考使用，具体详见“demo_main.cpp”。

```

/*
*本demo向用户展示了如何构建响应回调、如何创建api请求对象、以及简单的将部分业务逻辑串联起来
*demo提供了测试艾科柜台环境是否搭建完好的功能
*以上的操作并不意味着api的使用场景局限于此，尤其是业务串联部分的处理，需要用户结合具体的需求自行组织代码逻辑
*如果使用时有任何疑问，可以向艾科人员咨询
*/
int main(int argc,char* argv[]) {
    if(argc != 3){
        PRINT_INFO("input accountID,password when start demo");
        PRINT_INFO("example: ./demo accountID password");
        return 0;
    }

    std::string accountID(argv[1]);
    std::string password(argv[2]);

    init_g_param();

    /*

```

* **api**的初始化分为以下三步：

- * 1、创建一个接收回报的类，该类继承自**xeleSecuritiesTradersSpi**，用来接收艾科柜台的回报响应
- * 2、调用**createTradeApi**接口，创建一个请求类对象，该对象用来向艾科柜台发送操作请求
- * 3、调用**registersSpi**接口，将回报接收类对象传进请求类对象中
- * **api**获取操作权限的方式：
- * 1、调用**reqLogin**接口，传入配置文件路径、资金账户、资金账户密码以及本地维护的请求编号即可。
- * 其中，配置文件可以对**api**进行一些初始化的配置，包括**api**的连接方式（通过管理中心连接柜台还是直连柜台，这两种方式区别在于是否拥有跨柜台的资金操作权限，后者没有）、
- * **api**的绑核、心跳设置、**socket**类型选择、是否有地址、端口映射等
- * 当：
- * 1) 接口**onRspLoginManager**被回调，并且没有错误消息时，表明当前拥有艾科管理中心的操作权限
- * 2) 接口**onRspLogin**被回调，并且没有错误消息时，表明当前拥有艾科柜台的查询权限
- * 3) 接口**onRspInitTrader**被回调，并且没有错误消息时，表明当前拥有艾科柜台的报、撤单权限
- * */

```
auto* pUserSpi = new DemoSpi();

pUserApi = xeleSecuritiesTraderApi::createTraderApi();
if(!pUserApi){
    PRINT_INFO("can not create API");
    exit(0);
}

pUserApi->registersSpi(pUserSpi);

int ret = 0;
ret = pUserApi->reqLogin("./api_config.txt",accountID.data(),password.data(),g_RequestID++);
if(ret){
    PRINT_INFO("login error,ret [%d]",ret);
    exit(0);
}

//此处是demo为了得到当前api可以执行的权限而做的等待，实际运用时可以直接在回调函数中进行操作
while (true){
    if(pUserSpi->canOrder){
        break;
    }
    sleep(1);
}

//测试柜台接口
PRINT_INFO("Test Manager And Counter Interface");
CXeleReqQryClientAccountField qryClientAccountField{};
memset(&qryClientAccountField, 0, sizeof(CXeleReqQryClientAccountField));
//查询该资金账号下的可用资金，更新到本地
memcpy(qryClientAccountField.AccountID, accountID.data(), accountID.length());

PRINT_INFO("Test reqQryFund");
pUserApi->reqQryFund(qryClientAccountField, g_RequestID++);

//此处是为等待demo中程序走完登出流程
```

```
//此处选择的操作方式只是为了让demo的流程尽量完整，不代表必须使用该种方式
```

```
while (true){  
    sleep(1);  
    if(!puserSpi->canFundTransfer) {  
        sleep(2);  
        break;  
    }  
}  
//程序退出，调用release,删除接口对象  
puserApi->release();  
delete pusersSpi;  
printf("exit success\n");  
  
return 0;  
}
```

柜台系统其他重要说明

心跳机制

Xele-Trade-Securities交易系统与API以周期时间向对方发送心跳报文来维持连接，互相不需要回复心跳应答报文。

- **API心跳周期时间：**

配置API的api_config.txt配置文件中的HeartBeatInterval项进行配置。

根据配置的时间内进行一次心跳交互。

- **心跳超时次数：**

配置API的api_config.txt配置文件中的HeartBeatTimeOutCnt项进行配置。API超过配置的次数内未收到柜台心跳，则判定TCP断链，需要重新登录来获取权限。

- **断链：**

当API收到onFrontQueryDisconnected报文时，则代表断链，则需要重新获取权限

【注意】该心跳不需要用户进行维护

CoreDump

在程序不寻常退出时，内核会在当前工作目录下生成一个core文件（是一个内存映像，同时加上调试信息）。使用gdb来查看core文件，可以指示出导致程序出错的代码所在文件和行数。当用户使用ulimit -c查看配置，也可以用ulimit -c N（N为表示core文件大小数字，单位100K）来进行配置，当N不为0时，开启生成core文件机制。若用户N设置超过8G时，文件大小会被api限制为8G，当N为0时，不会生成core文件。

在api中注册了SIGNAL处理函数，捕捉并且处理SIGSEGV、SIGABRT、SIGPIPE信号，其中SIGSEGV、SIGABRT信号处理时，会生成coredump文件，用来输出backtrace信息，SIGPIPE信号则会被忽略。

系统支持巨页分配

为了降低Api时延抖动，在Api运行的操作系统中需要支持巨页分配。在不支持巨页分配的操作系统中，可能会影响到Api的时延数据，但是不会影响Api的正常运行与报单。下面提供在centos7.6版本下的巨页配置方式：

使用 `cat /etc/redhat-release` 确认操作系统版本，如下：

```
[root@localhost trader_api]# cat /etc/redhat-release
CentOS Linux release 7.6.1810 (Core)
```

打开/etc/grub2.conf文件，如下：

```
87 ## BEGIN /etc/grub.d/10_linux ##
88 menuentry 'CentOS Linux (3.10.0-957.el7.x86_64) 7 (Core)' --class centos --class gnu-linux --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-3.10.0-957.el7.x86_64-advanced-85a0ea82-9bc6-476a-83ba-6999b43b4509' {
89     load_video
90     set gfxpayload=keep
91     insmod gzio
92     insmod part_msdos
93     insmod xfs
94     set root='hd0,msdos1'
95     if [ x${bootloader_hints} = xy ]; then
96         search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' 0000c050-ba73-44d3-b103-34b1032a63d8
97     else
98         search --no-floppy --fs-uuid --set=root 0000c050-ba73-44d3-b103-34b1032a63d8
99     fi
100     linux16 /vmlinuz-3.10.0-957.el7.x86_64 root=dev/mapper/centos-root ro crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet LANG=en_US.UTF-8 $tuned_params
101     initrd16 /initramfs-3.10.0-957.el7.x86_64.img
102 }
103 menuentry 'CentOS Linux (0-rescue-170385b84a9e4a49cfff169cb1305e2) 7 (Core)' --class centos --class gnu-linux --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-0-rescue-170385b84a9e4a49cfff169cb1305e2-advanced-85a0ea82-9bc6-476a-83ba-6999b43b4509' {
104     load_video
```

在文件中找到相应位置，在其后添加如下字符串：

```
default_t_hugepagesz=1G hugepagesz=1G hugepages=20
```

其中：

hugepages：在内核中定义了开机启动时就分配的永久大页面的数量。默认为 0，即不分配。只有当系统有足够的连续可用页时，分配才会成功。

hugepagesz：在内核中定义了开机启动时分配的大页面的大小。可选值为 2MB 和 1GB。默认是 2MB。

default_hugepagesz：在内核中定义了开机启动时分配的大页面的默认大小

具体分配大小以及数量需要根据操作系统内存具体情况进行调整，Api至少需要配置1G的可用巨页内存才能达到稳定时延的效果，配置完成如下：

```
87 ## BEGIN /etc/grub.d/10_linux ##
88 menuentry 'CentOS Linux (3.10.0-957.el7.x86_64) 7 (Core)' --class centos --class gnu-linux --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-3.10.0-957.el7.x86_64-advanced-85a0ea82-9bc6-476a-83ba-6999b43b4509' {
89     load_video
90     set gfxpayload=keep
91     insmod gzio
92     insmod part_msdos
93     insmod xfs
94     set root='hd0,msdos1'
95     if [ x${bootloader_hints} = xy ]; then
96         search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' 0000c050-ba73-44d3-b103-34b1032a63d8
97     else
98         search --no-floppy --fs-uuid --set=root 0000c050-ba73-44d3-b103-34b1032a63d8
99     fi
100     linux16 /vmlinuz-3.10.0-957.el7.x86_64 root=dev/mapper/centos-root ro crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet LANG=en_US.UTF-8 $tuned_params default_t_hugepagesz=1G hugepagesz=1G hugepages=20
101     initrd16 /initramfs-3.10.0-957.el7.x86_64.img
102 }
103 menuentry 'CentOS Linux (0-rescue-170385b84a9e4a49cfff169cb1305e2) 7 (Core)' --class centos --class gnu-linux --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-0-rescue-170385b84a9e4a49cfff169cb1305e2-advanced-85a0ea82-9bc6-476a-83ba-6999b43b4509' {
104     load_video
```

重启操作系统即可生效。

Trader-API接口参考说明

API接口说明列表

证券（上交所股票、深交所股票）柜台使用的API接口类型为：系统相关、证券期权共用、证券相关
期权柜台（仅支持上交所期权）使用的API接口类型为：系统相关、证券期权共用、期权相关

接口类型	接口分类	说明
系统相关	CXeleTraderApi:: createTraderApi	创建API对象
系统相关	CXeleTraderApi:: getVersion	获取当前API的版本信息
系统相关	CXeleTraderApi:: join	等待线程结束

接口类型	接口分类	说明
系统相关	CXeleTraderApi:: release	销毁API对象
系统相关	CXeleTraderApi::registerSpi	注册回调对象
系统相关	CXeleTraderApi:: reqLogin	登录请求
系统相关	CXeleTraderApi:: reqLoginEx	登录请求
系统相关	CXeleTraderSpi:: onRspLogin	艾科柜台登录响应
系统相关	CXeleTraderSpi:: onRspLoginManager	艾科柜台登录应答
系统相关	CXeleTraderSpi:: onRspInitTrader	添加交易链路应答
系统相关	CXeleTraderApi:: reqLogout	登出请求
系统相关	CXeleTraderSpi:: onRspLogout	登出响应
系统相关	CXeleTraderSpi:: onRspLogoutManager	艾科管理中心登出应答
系统相关	CXeleTraderApi:: reqUpdatePwd	密码更新请求
系统相关	CXeleTraderSpi:: onRspUpdatePwd	密码更新应答
系统相关	CXeleTraderSpi:: onFrontQueryDisconnected	断开查询连接链路
系统相关	CXeleTraderSpi:: onFrontTradeDisconnected	断开交易连接链路
系统相关	CXeleTraderSpi::onApiMsg	api内部消息打印回调
证券、期权共用	CXeleTraderApi:: reqInsertOrder	报单请求
证券、期权共用	CxeleTraderApi:: reqInsertBatchOrder	批量报单请求
证券、期权共用	CXeleTraderSpi:: onRspInsertOrder	报单应答
证券、期权共用	CXeleTraderSpi::onErrRtnInsertOrder	报单错误回报
证券、期权共用	CXeleTraderApi:: reqCancelOrder	撤单请求
证券、期权共用	CXeleTraderSpi:: onRspCancelOrder	撤单应答
证券、期权共用	CXeleTraderSpi::onErrRtnCancelOrder	撤单错误回报
证券、期权共用	CXeleTraderSpi::onRtnOrder	报单回报
证券、期权共用	CXeleTraderSpi::onRtnTrade	成交回报

接口类型	接口分类	说明
证券、期权共用	CXeleTraderApi:: reqQryOrder	报单查询请求
证券、期权共用	CXeleTraderSpi:: onRspQryOrder	报单查询应答
证券、期权共用	CXeleTraderApi:: reqQryTrade	成交查询请求
证券、期权共用	CXeleTraderSpi:: onRspQryTrade	成交查询应答
证券相关	CXeleTraderApi:: reqQryRate	费率(印花税率、过户费率、佣金率、流量费)查询请求
证券相关	CXeleTraderSpi:: onRspQryRate	费率(印花税率、过户费率、佣金率、流量费)查询应答
证券相关	CXeleTraderApi::reqQryCentralTradingFund	集中交易柜台资金明细查询请求
证券相关	CXeleTraderSpi::onRspQryCentralTradingFund	集中交易资金查询响应
证券相关	CXeleTraderApi:: reqInFund	集中交易资金调入艾科柜台请求
证券相关	CXeleTraderSpi:: onRspInFund	集中交易资金调入艾科柜台应答
证券相关	CXeleTraderApi:: reqOutFund	艾科柜台资金调出集中交易请求
证券相关	CXeleTraderSpi:: onRspOutFund	艾科柜台资金调出集中交易应答
证券相关	CXeleTraderApi:: reqQryInOutFundRecord	集中交易资金调拨艾科柜台明细查询请求
证券相关	CXeleTraderSpi:: onRspQryInOutFundRecord	集中交易资金调拨艾科柜台明细查应答
证券相关	CXeleTraderApi:: reqBTFundTransfer	跨柜台资金调拨请求
证券相关	CXeleTraderSpi:: onRspBTFundTransfer	跨柜台资金调拨应答
证券相关	CXeleTraderSpi:: onRtnBTFundTransfer	跨柜台资金调拨结果返回
证券相关	CXeleTraderApi::reqQryBTCapTransferRecord	跨柜台资金调拨记录查询请求
证券相关	CXeleTraderSpi::onRspBTCapTransferRecord	跨柜台资金调拨记录查询应答

接口类型	接口分类	说明
证券相关	CXeleTraderApi::reqQryBtFund	沪深柜台资金查询请求
证券相关	CXeleTraderSpi::onRspQryBtFund	沪深柜台资金查询应答
证券相关	CXeleTraderApi::reqQryFund	证券资金查询请求
证券相关	CXeleTraderSpi::onRspQryFund	证券资金查询应答
证券相关	CXeleTraderApi::reqQryPosition	证券持仓查询请求
证券相关	CXeleTraderSpi::onRspQryPosition	证券持仓查询应答
证券相关	CXeleTraderApi::reqQrySecurities	证券信息查询请求
证券相关	CXeleTraderSpi::onRspQrySecurities	证券信息查询应答
证券相关	CXeleTraderApi::reqQryRightsAndInterests	权益查询-新股额度查询
证券相关	CXeleTraderSpi::onRspQryRightsAndInterests	权益查询-新股额度回报
证券相关	CXeleTraderApi::reqQryInvestorInfo	股东账号信息查询请求
证券相关	CXeleTraderSpi:: onRspQryInvestorInfo	股东账户信息查询应答
期权相关	CXeleTraderApi::getRebuildFlag	期权流水重构状态查询
期权相关	CXeleTraderSpi::onRspRebuildFinish	期权流水重构结束应答
期权相关	CXeleTraderApi::reqInsertCombOrder	期权组合报单请求
期权相关	CXeleTraderSpi::onRspInsertCombOrder	期权组合报单应答
期权相关	CXeleTraderSpi::onErrRtnInsertCombOrder	期权组合报单错误响应
期权相关	CXeleTraderSpi::onRtnCombOrder	期权组合报单回报
期权相关	CXeleTraderSpi::onRtnCombTrade	期权组合成交回报
期权相关	CXeleTraderApi::reqInsertExercise	期权单腿行权报单请求
期权相关	CXeleTraderSpi::onRspInsertExercise	单腿行权报单应答
期权相关	CXeleTraderSpi::onErrRtnInsertExercise	单腿行权错误回报
期权相关	CXeleTraderApi::reqCancelExercise	期权单腿行权报单请求
期权相关	CXeleTraderSpi::onRspCancelExercise	单腿行权撤单应答
期权相关	CXeleTraderSpi::onErrRtnCancelExercise	单腿行权撤单错误回报
期权相关	CXeleTraderSpi::onRtnExerciseOrder	单腿行权报单回报
期权相关	CXeleTraderApi::reqInsertExerciseComb	组合行权报单请求
期权相关	CXeleTraderSpi::onRspInsertExerciseComb	组合行权报单应答
期权相关	CXeleTraderSpi:: onErrRtnInsertExerciseComb	组合行权错误回报

接口类型	接口分类	说明
期权相关	CXeleTraderApi::reqCancelExerciseComb	组合行权撤单请求
期权相关	CXeleTraderSpi::onRspCancelExerciseComb	组合行权撤单应答
期权相关	CXeleTraderSpi::onErrRtnCancelExerciseComb	组合行权撤单错误回报
期权相关	CXeleTraderSpi::onRtnExerciseCombOrder	组合行权报单回报
期权相关	CXeleTraderApi::reqQryOptionPosition	期权持仓查询请求
期权相关	CXeleTraderSpi::onRspQryOptionPosition	期权持仓查询应答
期权相关	CXeleTraderApi::reqQryOptionFund	期权资金查询请求
期权相关	CXeleTraderSpi::onRspQryOptionFund	期权资金查询应答
期权相关	CXeleTraderApi::reqQryOptionSecurities	期权合约查询请求
期权相关	CXeleTraderSpi::onRspQryOptionSecurities	期权合约查询应答
期权相关	CXeleTraderApi::reqQryOptionRate	期权佣金费率、保证金率 查询请求
期权相关	CXeleTraderSpi:: onRspQryOptionRate	期权佣金费率、保证金率 查询应答
期权相关	CXeleTraderApi::reqQryOptionCombPosition	期权组合持仓查询请求
期权相关	CXeleTraderSpi::onRspQryOptionCombPosition	期权组合持仓查询应答
期权相关	CXeleTraderSpi:: onRtnCapitalTransferDetails	期权资金(出入金)流水明细 回报
期权相关	CXeleTraderApi:: reqOTT	(期权)会员申请转处置证券 账户请求
期权相关	CXeleTraderSpi:: onRspOTT	(期权)会员申请转处置证券 账户响应
期权相关	CXeleTraderSpi:: onRtnOTT	(期权)会员申请转处置证券 账户回报
期权相关	CXeleTraderSpi:: onErrRtnOTT	(期权)会员申请转处置证券 账户错误回报
期权相关	CXeleTraderApi::reqCancelOTT	(期权)会员申请转处置证券 账户撤单请求
期权相关	CXeleTraderSpi:: onRspCancelOTT	(期权)会员申请转处置证券 账户撤单响应
期权相关	CXeleTraderSpi:: onErrRtnCancelOTT	(期权)会员申请转处置证券 账户撤单错误回报

接口类型	接口分类	说明
期权相关	CXeleTraderApi::reqInsertOQO	期权双边报价请求
期权相关	CXeleTraderSpi::onRspInsertOQO	期权双边报价响应
期权相关	CXeleTraderSpi::onRtnInsertOQO	期权双边报价回报
期权相关	CXeleTraderSpi::onErrRtnInsertOQO	期权双边报价错误回报
期权相关	CXeleTraderApi::reqCancelOQO	期权双边报价撤单请求
期权相关	CXeleTraderSpi::onRspCancelOQO	期权双边报价撤单响应
期权相关	CXeleTraderSpi::onErrRtnCancelOQO	期权双边报价撤单错误回报
期权相关	CXeleTraderApi::reqOMR	期权保证金查询请求
期权相关	CXeleTraderSpi::onRspOMR	(期权) 保证金查询响应
期权相关	CXeleTraderSpi::onRtnOMR	(期权)保证金查询回报
其他（用户不使用）	CXeleTraderApi::getSupervisionServerInfo	webServer用户获取系统信息使用

系统相关

API类接口

createTraderApi方法

功能：创建API对象

函数原形：

```
static xeleSecuritiesTraderApi *createTraderApi();
```

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();
```

返回值：API版本信息字符串

setApiLogPath方法

功能：设置API日志路径

函数原形：

```
static bool setApiLogPath(const char *logPath);
```

参数：

- logPath：必选，日志路径，相对或绝对路径，调用方保证目录存在

返回值：true表示设置成功，false表示路径不存在，设置失败

说明：若不设置或设置失败，使用默认路径“../logs”；建议创建实例前设置，全局调用一次即可，该设置全局有效。

setApiSuperLog方法

功能：设置是否开启详细日志打印

函数原形：

```
static void setApiSuperLog(int superLog);
```

参数：

- superLog：必选，0表示关闭详细打印，非0表示开启详细打印

返回值：无

说明：若API实例登陆后，遇到问题，可使用此接口实时开启或关闭详细打印。该设置优先级大于配置参数ConfigParam中的SuperLog；若设置开启，配置参数中的SuperLog无效，若不设置或设置关闭，配置参数中的SuperLog生效。

【注意】==此接口为静态全局接口，影响所有API实例。==

join方法

功能：等待接口线程结束运行

函数原形：

```
int join();
```

返回值：0，成功；其他值：Errno

release 方法

功能：删除接口对象本身,不再使用本接口对象时,调用该函数删除接口对象

```
void release();
```

返回值：无

checkPassword 方法

功能：登录成功后，调用此接口校验当前账户的密码是否正确；

```
int checkPassword(const char *accountId, const char *password);
```

参数：

- accountId：必选，资金账户
- password：必选，账户密码

返回值：0:表示密码校验正确，其他值表示校验失败的错误码。

registerSpi方法

功能：注册回调对象。注册XeleSecuritiesTraderSpi的子类指针，用户可通过继承该类实现自定义。

函数原形：

```
void registersSpi(XeleSecuritiesTraderSpi *pspi);
```

返回值：无

reqLogin 方法

功能：用户登录请求

函数原形：

```
int reqLogin(const char* configPath, const char* accountId, const char* password, int node, char market, int nRequestID);
```

参数：

- configPath：必选，Api配置文件所在地址
- accountId：必选，资金账户
- password：必选，账户密码
- node: 若本地配置文件api_config.txt中有此字段且已配置，则此入参可选填，否则必填，建议必填，账户节点
- market：若本地配置文件api_config.txt中有此字段且已配置，则此入参可选填，否则必填，建议必填，柜台类型，'1'=上交，'2'=深交
- nRequestID：可选，请求ID，用户维护，自增，用来区分对应单

返回值：0表示正常，其他值异常；

说明：通过API配置文件api_config.txt，配置完成后，当客户端与柜台系统成功建立连接后，用户输入用户名和密码，使用reqLogin登录到柜台系统。通过登录回报onRspLogin、onRspLoginManager、onRspInitTrader获取相关信息。

reqLoginEx方法

功能：登录请求 (不使用配置文件，直接传参)

函数原形：

```
int reqLoginEx(const ConfigParam* param, int nRequestID)
```

参数：

- param：配置参数
- nRequestID：请求ID，用户维护，自增，用来区分对应单

返回值：0表示正常，其他值异常；

说明：==该接口是给不喜欢配置文件的投资者使用，该登录请求接口不使用配置文件，直接传参。ConfigParam具体数据格式见XeleSecuritiesTraderApi.h文件，针对有默认值的字段，如果没有说明中的使用场景，可以不进行赋值操作。==

reqUpdatePwd 方法

功能：密码更新请求

函数原形：

```
int reqUpdatePwd(CXeleReqUserPasswordUpdateField &inputField, int nRequestID) ;
```

参数：

- inputField：修改密码请求域（密码组成：数字字母下划线），其结构体 CXeleReqUserPasswordUpdateField 如下：

```
struct CXeleReqUserPasswordUpdateField {  
    ///资金账户  
    TXeleUserIDType      AccountID;  
    ///新密码  
    TXeleUserPasswordType NewPassword;  
    ///旧密码  
    TXeleUserPasswordType OldPassword;  
    ///预留  
    TXeleReserved1Type    Reserved;  
};
```

- nRequestID：请求ID。

返回值：0表示成功，其他值异常；

说明：当用户成功登录柜台后，使用reqUpdatePwd请求修改用户密码，可以在onRspUpdatePwd获取相关信息。

【注意】该方法暂时只支持修改柜台端的密码，集中交易密码由用户自己管理，客户登录时是直接到集中交易进行校验。

reqLogout 方法

功能：登出请求

函数原形：

```
int reqLogout(const char* accountId, int nRequestID);
```

参数：

- accountId：必选，资金账户
- nRequestID：可选，请求ID

返回值：0表示成功，其他值异常；

说明：当用户成功登录柜台后，使用reqLogout请求登出柜台，可以在onRspLogout获取相关信息。

reqInitTrader 方法

功能：发送交易链路初始化请求；

函数原形：

```
int reqInitTrader(const char *srcTcpIp, uint16_t srcTcpPort, const char
*srcUdpIp, uint16_t srcUdpPort, int nRequestID);
```

参数：

- srcTcpIp：必选，交易链路TCP连接源地址
- srcTcpPort：必选，交易链路TCP连接源端口
- srcUdpIp：UDP交易必选，交易链路UDP报单源地址
- srcUdpPort：：UDP交易必选，交易链路UDP报单源端口
- nRequestID：可选，请求ID

返回值：0表示成功，其他值异常；

说明：

用户在裸协议报单时，用于初始化自建的报单链路：

- 1) 当 InitTrader请求 发送成功后，可以在 onRsplnInitTrader() 接口获取应答相关信息；
- 2) 当配置参数 EstablishTradeLink=0 且通过FPGA通道进行裸协议交易时，该接口调用有效；
- 3) 如果用户需要使用软通道进行交易，那么需要自己构造InitTrader请求，通过交易链路（用户创建）发送给柜台进行链路的初始化；并从交易链路接收初始化应答消息。初始化链路的请求和应答消息协议如下：

```
///添加交易链路请求
struct CXeleReqInitTraderField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///交易ip
    TXeleTradeIPType         TradeDestIp;
    ///交易端口
    TXeleTradePortType       TradeDestPort;
    ///源ip
    TXeleTradeIPType         TradeSrcIp;
    ///源端口
    TXeleTradePortType       TradeSrcPort;
    ///UDP交易ip
    TXeleTradeIPType         UdpTradeDestIp;
    ///UDP交易端口
    TXeleTradePortType       UdpTradeDestPort;
    ///UDP源ip
    TXeleTradeIPType         UdpTradeSrcIp;
    ///UDP源端口
    TXeleTradePortType       UdpTradeSrcPort;
    ///UDP交易标志，1表示UDP交易相关的字段有效，0表示无效
    TXeleUdpTradeFlagType    UdpTradeFlag;
    /// 预留
    char                     Reserved[59];
};
```

```
static_assert(sizeof(struct CXeleReqInitTraderField) == 143, "struct size error.");

///添加交易链路请求响应
struct CXeleRspInitTraderField {
    ///资金账户
    TXeleUserIDType          AccountID;
    // 预留
    char                      Reserved[113];
};
```

commonFunc 方法

功能：通用接口

函数原形：

```
int commonFunc(
    const char* configPath,
    const char* command,
    const char* commandStruct,
    char market='1',
    int node = 0,
    int nRequestID=0);
```

参数：

- configPath：必选，Api配置文件所在地址
- command：必选，通用接口的功能号
- commandStruct：必选，对应功能号的结构体
- market: 可选，市场，默认为上交，'1'=上交，'2'=深交
- node：可选，节点
- nRequestID：可选，请求ID，用户维护，自增，用来区分对应单

返回值：0表示成功，其他值异常；

说明：当柜台相应成功之后，调用commonFunc通用接口对应的功能号，可以在onRspCommonFunc获取相关信息。

【注意】该方法暂时只支持西部证券，调用此接口不需要登录。

传参数command与commandStruct对应关系：

command (字符串值)	commandStruct (结构体类型)	说明
AccountInit	CXeleReqAccountInitField	初始化账号密码

```
struct CXeleReqAccountInitField {
    ///必选，资金账户
    TXeleUserIDType          AccountID;
    //必选，/交易用户密码
    TXeleUserPasswordType    Password;
    ///交易所类型（默认为上交）
    TXeleMarketType          Market;
```

```

    ///必选，命令号（目前只有AccountInit）
    TXeleCommandNum          CommandNum;
    ///委托方式（不需要填写，内部使用）
    TXeleOperwayType         Operway;
    ///柜台ip（不需要填写，内部使用）
    TXeleTradeIPType         QueryURL;
    ///预留
    char                      Reserved[66]; };

```

getErrMsgEx 方法

功能：获取错误码对应的错误信息，支持中英文。

函数原型：

```
const char* getErrMsgEx(unsigned int errorId, char market, int language);
```

参数：

- errorId：待查错误码；
- market：市场，具体定义参考 TXeleMarketType 枚举；
- language：错误信息版本，1表示中文，2表示英文；

返回值：

- 查询成功，返回错误信息的常量字符串指针；
- 查询失败，返回对应的说明字符串；

SPI类接口

onFrontManagerQueryConnect方法

功能：

- api与艾科管理中心tcp建连成功回调
- 该回调表明当前api已经与管理中心完成了tcp的连接，接下来可以进行登录报文交互（该交互不需要用户关心）
- 正常情况下无需关心此回调，
- 当发生异常情况时(例如，登录请求发出后，迟迟收不到相关登录响应，当前环境可能出现异常)，
- 需要简单排查问题，可以通过该回调大致了解到当前api异常模块，方便自行排查问题

函数原形：

```
void onFrontManagerQueryConnect() {}
```

返回值：无

onFrontQueryConnect方法

功能：

- api与艾科柜台查询tcp建连成功回调
- 该回调表明当前api已经与艾科柜台完成了查询tcp的连接，接下来可以进行登录报文交互（该交互不需要用户关心）
- 正常情况下无需关心此回调，
- 当发生异常情况时(例如，登录请求发出后，迟迟收不到相关登录响应，当前环境可能出现异常)
- 需要简单排查问题，可以通过该回调大致了解到当前api异常模块，方便自行排查问题

函数原形：

```
void onFrontQueryConnect() {}
```

返回值：无

onFrontTradeConnect方法

功能：

- api与艾科柜台交易tcp建连成功回调
- 该回调表明当前api已经与艾科柜台完成了交易tcp的连接，接下来可以进行登录报文交互（该交互不需要用户关心）
- 正常情况下无需关心此回调，当发生异常情况时(例如，登录请求发出后，迟迟收不到相关登录响应，当前环境可能出现异常)，
- 需要简单排查问题，可以通过该回调大致了解到当前api异常模块，方便自行排查问题

函数原形：

```
void onFrontTradeConnect () {}
```

返回值：无

onRspLoginManager方法

功能：艾科柜台登录应答,当需要管理中心接口可用，但是只需要艾科柜台查询接口可用时，收到该回报即可进行操作

函数原形：

```
void onRspLoginManager(CXeleRspUserLoginManagerField *pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast)
```

- pRspField：登录应答，其结构体CXeleRspUserLoginManagerField如下：

```
struct CXeleRspUserLoginManagerField {  
    ///交易日  
    TXeleDateType           TradingDay;  
    ///登录成功时间  
    TXeleShortTimeType      LoginTime;  
    ///资金账户  
    TXeleUserIDType         AccountID;  
    ///会话代码
```

TXeleSessionIDType	SessionId;
///校验用	
TXeleTokenType	Token;
///委托方式	
TXeleOperwayType	Operway;
///柜台连接地址	
TXeleCounterURL	CounterUrl;
///客户端登录子节点	
TXeleSubClientIndexType	SubClientIndex;
///心跳时间	
TXeleHeartBeatInterval	HeartBeatInterval;
///心跳超时时间	
TXeleHeartBeatTimeout	HeartBeatTimeout;
///登录的是互联网端口(内部使用)	
TXeleInternetType	IsInternetConnect;
///预留	
char	Reserved[69];};

onRspLogin 方法

功能：艾科柜台登录应答,当需要管理中心接口可用，但是只需要艾科柜台查询接口可用时，收到该回报即可进行操作

函数原形：

```
void onRspLogin(CXeleRspUserLoginField *pRspField,CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) ;
```

参数：

- pRspField：登录应答，其结构体CXeleRspUserLoginField如下：

```
struct CXeleRspUserLoginField {
    ///交易日
    TXeleDateType TradingDay;
    ///登录成功时间
    TXeleShortTimeType LoginTime;
    ///用户本地最大报单号
    TXeleOrderIDType MaxUserLocalID;
    ///资金账户
    TXeleUserIDType AccountID;
    ///会话代码
    TXeleSessionIDType SessionId;
    ///校验用
    TXeleTokenType Token;
    ///客户端登录子节点
    TXeleSubClientIndexType SubClientIndex;
    ///交易ip
    TXeleTradeIPType TradeDestIp;
    ///交易端口
    TXeleTradePortType TradeDestPort;
    ///交易通道有效标志，0:无效，1:有效
    TXeleTradeFlagType TradeFlag;
```

```

    ///委托方式
    TXeleOperwayType                Operway;
    ///交易所类型
    TXeleMarketType                Market;
    ///报单通道,'0':硬件通道 '1':软件通道
    TXeleTradeType                TradeType;
    ///心跳时间
    TXeleHeartBeatInterval        HeartBeatInterval;
    ///心跳超时时间
    TXeleHeartBeatTimeout        HeartBeatTimeout;
    ///登录的是互联网端口(内部使用)
    TXeleInternetType            IsInternetConnect;
    ///UDP交易ip
    TXeleTradeIPType                UdpTradeDestIp;
    ///UDP交易端口
    TXeleTradePortType            UdpTradeDestPort;
    ///系统特性
    TXeleSystemFeaturesType        SystemFeatures;
    ///预留
    char                            Reserved[76];
};

```

onRspInitTrader方法

功能：添加交易链路应答,当收到该回报时，标记艾科柜台报、撤单接口可用

函数原形：

```

void onRspInitTrader(CXeleRspInitTraderField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast)

```

参数：

- pRspField：用户添加交易链路，其结构体CXeleRspInitTraderField如下：

```

struct CXeleRspInitTraderField {
    ///资金账户
    TXeleUserIDType                AccountID;
    // 预留
    char                            Reserved[113];
};

```

onRspUpdatePwd 方法

功能：密码更新应答

函数原形：

```

void onRspUpdatePwd(CXeleRspUserPasswordUpdateField *pRspField, XeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast)

```

参数：

- pRspField：用户修改密码应答，其结构体CXeleRspUserPasswordUpdateField如下：

```
struct CXeleRspUserPasswordUpdateField {  
    ///资金账户  
    TXeleUserIDType          AccountID;  
    ///新密码  
    TXeleUserPasswordType     NewPassword;  
    ///旧密码  
    TXeleUserPasswordType     OldPassword;  
    ///预留  
    TXeleReserved1Type        Reserved;  
};
```

onRspLogoutManager方法

功能: 艾科管理中心登出应答

函数原形：

```
void onRspLogoutManager(CXeleRspUserLogoutManagerField *pRspField, CXeleRspInfo  
*pRspInfo, int nRequestID, bool bIsLast)
```

参数：

- pRspField：用户退出应答，其结构体CXeleRspUserLogoutManagerField如下：

```
struct CXeleRspUserLogoutManagerField {  
    ///交易日  
    TXeleDateType            TradingDay;  
    ///登出时间  
    TXeleShortTimeType        LogoutTime;  
    ///资金账户  
    TXeleUserIDType          AccountID;  
    ///会话代码  
    TXeleSessionIDType        SessionId;  
    ///预留  
    TXeleReserved1Type        Reserved;  
};
```

onRspLogout 方法

功能：登出应答

函数原形：

```
void onRspLogout(CXeleRspUserLogoutField *pRspField, CXeleRspInfo *pRspInfo, int  
nRequestID, bool bIsLast);
```

参数：

- pRspField：用户退出应答，其结构体CXeleRspUserLogoutFiel如下：

```
struct CXeleRspUserLogoutField {  
    ///交易日  
    TXeleDateType           TradingDay;  
    ///登出时间  
    TXeleShortTimeType      LogoutTime;  
    ///本地最大报单号  
    TXeleOrderIDType        MaxUserLocalID;  
    ///资金账户  
    TXeleUserIDType         AccountID;  
    ///会话代码  
    TXeleSessionIDType      SessionId;  
    ///预留  
    TXeleReserved1Type      Reserved;  
};
```

onFrontManagerQueryDisconnected方法

功能：当客户端与管理中心服务端查询通信连接断开时，该方法被调用。该回报触发条件为客户主动发起登出请求，或者链路发生了断连。出现以上情况时，若不是主动发起登出请求，建议先发起登出请求，再重新登录，即可再次使用相关接口。

函数原形：

```
void onFrontManagerQueryDisconnected (int nReason) ;
```

参数：

- nReason：错误码

返回值：无

说明：当api和管理中心的tcp连接断连后，此回调被调用，此时api会做处理，不会影响到api和柜台的交易和查询功能。如果对管理中心相关接口无操作需要时，可以不对此响应进行操作。

onFrontQueryDisconnected 方法

功能：当客户端与服务端查询通信连接断开时，该方法被调用。该回报触发条件为客户主动发起登出请求，或者链路发生了断连。出现以上情况时，若不是主动发起登出请求，建议先发起登出请求，再重新登录，即可再次使用相关接口。

函数原形：

```
void onFrontQueryDisconnected(int nReason) ;
```

参数：

- nReason：错误码

返回值：无

说明：当客户端与后台查询通信连接断开时，该方法被调用。

onFrontTradeDisconnected 方法

功能：api与柜台交易链路断连

函数原形：

```
void onFrontTradeDisconnected(int nReason)
```

参数：

- nReason：错误码

返回值：无

说明：当客户端与服务端交易通信连接断开时，该方法被调用。当收到该回调，表示当前连接失去了报、撤单功能。如需恢复，参考OnFrontQueryDisconnected处理方法

onRspCommonFunc方法

功能：通用接口应答，通用接口命令号看pRspField->CommandNum，结果看pRspInfo->ErrorID。

函数原形:

```
void onRspCommonFunc(CXeleRspCommonFuncField *pRspField, CXeleRspInfo *pRspInfo,
int nRequestID, bool bIsLast);
```

参数：

- pRspField：通用接口应答，其结构体CXeleRspCommonFuncField如下：

```
struct CXeleRspCommonFuncField {
    ///命令号
    TXeleCommandNum          CommandNum;
    ///预留
    char                      Reserved[108];
};
```

onDataReady 方法

```
///
///@param bSuccess: 数据加载结果
///@param nMsgId :
///@param
```

功能：数据准备完毕回调。该接口用来提供日初数据异步加载完成后，通知用户。

函数原型：

```
void onDataReady(bool bSuccess, int nMsgId, int nReason);
```

参数：

- bSuccess：数据加载是否成功，true表示成功，false表示失败
- mMsgId：数据加载失败时，表示加载失败对应的消息编号；加载成功时，暂未使用
- nReason：数据加载失败时，表示加载失败对应的原因；加载成功时，暂未使用

返回值：无

onApiMsg 方法

功能：消息打印回调。该接口旨在提供API内部打印信息接口，方便相关log保存，问题定位及客户使用。

函数原形:

```
void onApiMsg(int ret, const char *strFormat, ...);
```

参数:

- ret:标识正常或错误信息（-1，错误信息打印；0，正常信息打印）
- strFormat: 可变参数

返回值：无

该接口使用方式，示例如下，详参demo

```
///api内部消息打印回调
void onApiMsg(int ret, const char *strFormat, ...) override{
    char strLog[2048]{};
    va_list arglist;
    va_start(arglist, strFormat);
    vsprintf(strLog, strFormat, arglist);
    va_end(arglist);
    //1:error, 0:normal
    if (ret) {
        PRINT_INFO("Error:%s", strLog);
    } else {
        PRINT_INFO("%s", strLog);
    }
};
```

证券、期权共用

API类接口

reqInsertOrder 方法

功能：报单请求

函数原形：

```
int reqInsertOrder (CXeleReqOrderInsertField &inputField, int nRequestID);
```

参数：

- inputField：报单请求域，其结构体CXeleReqOrderInsertField如下：

```

struct CXeleReqOrderInsertField {
    ///必选，用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///必选，证券代码
    TXeleSecuritiesIDType     SecuritiesID;
    ///必选，买卖方向
    TXeleDirectionType        Direction;
    ///可选，开平标志(股票不填)
    TXeleOffsetFlagType        CmbOffsetFlag;
    ///必选，限价价格
    TXelePriceType             LimitPrice;
    ///必选，报单数量
    TXeleVolumeType            Volume;
    ///必选，报单类型
    TXeleOrderTypeType         OrderType;
    ///必选，有效期类型(股票填0x30)
    TXeleTimeConditionType     TimeCondition;
    ///可选，合约类型
    TXeleSecuritiesType        SecuritiesType;
    ///可选，备兑标志(期权使用)
    TXeleCoveredFlagType       CoveredOrUncovered;
    ///预留
    char                        Reserved2[2];
    ///可选，业务单元(用户定义)
    TXeleBusinessUnitType      BusinessUnit;
    ///委托方式
    TXeleOperwayType           Operway;
    ///交易前置id,默认或者未填写为0网关轮询,查询接口
    ///返回的网关需要转换成整型，如'1'转换成1后填充该字段
    TXeleExchangeIDType        ExchangeFrontID;
    ///内部使用
    unsigned char               ReservedNum;
    ///委托方式扩展字段,部分券商支持
    char                        OperwayEx[3];
    ///预留
    TXeleErrorIDType           ErrorID;
};

```

- nRequestID：请求ID。

返回值：0表示成功，其他值异常；

说明：当用户成功登录柜台，获取交易权限后，使用reqInsertOrder，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onRsplnInsertOrder回报；如果未通过交易所风控，则还会返回onErrRtnInsertOrder回报；详见[报单](#)

reqInsertBatchOrder方法

功能：批量报单

函数原型：

```

int reqInsertBatchOrder(CXeleReqBatchOrderInsertField &inputField, int
nRequestID);

```

参数：

- inputField：批量报单请求域，其结构体CXeleReqBatchOrderInsertField如下：

```
struct CXeleReqBatchOrderInsertField {
    ///批量报单类型
    TXeleBatchType                BatchType;
    ///单笔上限(等量拆单和递减拆单类型使用,等量拆单时表示单笔报单数量,递减拆单时表示递减起始值)
    TXeleVolumeType                MaxOrderQty;
    ///递减数量(递减拆单类型使用)
    TXeleVolumeType                DecreaseQty;
    ///多证券委托组合数量,表示多证券委托的数量(多证券委托组合使用)
    ///例如: 4表示本次多证券委托组合(ReqOrderInsertField)中共有4个合约
    TXeleVolumeType                BatchOrderQty;
    ///批量报单用户起始本地报单编号(等量拆单和递减拆单类型使用)
    TXeleOrderIDType                UserLocalID;
    ///证券代码(等量拆单和递减拆单类型使用)
    TXeleSecuritiesIDType            SecuritiesID;
    ///买卖方向(等量拆单和递减拆单类型使用)
    TXeleDirectionType            Direction;
    ///限价价格(等量拆单和递减拆单类型使用)
    TXelePriceType                LimitPrice;
    ///报单数量(等量拆单和递减拆单类型使用)
    TXeleVolumeType                Volume;
    ///报单类型(等量拆单和递减拆单类型使用)
    TXeleOrderTypeType            OrderType;
    ///业务单元(用户定义)(等量拆单和递减拆单类型使用)
    TXeleBusinessUnitType            BusinessUnit;
    ///多证券委托组合(多证券委托组合使用) 其中BatchMuiltMax为最大支持的合约数量,最大值为100
    CXeleReqOrderInsertField        ReqOrderInsertField[BatchMuiltMax];
    ///本次批量报单中,被拆分成子单数量,无需填写,接口内部填写,可以用来更新UserLocalID字段
    TXeleVolumeType                SplitOrderVolume;
    ///预留
    char                            Reserved[13];
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

当客户登录柜台系统，获取交易权限后，用户可以使用reqInsertBatchOrder进行批量报单操作，无论是否通过风控，都会返回一个onRsplInsertOrder回报；如果未通过交易所风控，则还会返回onErrRtnInsertOrder回报；详见[报单](#)

reqCancelOrder 方法

功能：撤单请求

函数原形：

```
int reqCancelOrder (CXeleReqOrderActionField &inputField, int nRequestID);
```

参数：

- inputField：撤单请求域，其结构体CXeleReqOrderActionField如下：

```
struct CXeleReqOrderActionField {
    ///必选，用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///可选，被撤单系统报单编号
    ///值为0时，使用OrigUserLocalID撤单；非0时，使用OrigSysID进行撤单
    TXeleOrigSysIDType        OrigSysID;
    ///可选，被撤单用户本地报单编号（暂不支持要约收购、权证行权通过该字段撤单）
    TXeleOrderIDType          OrigUserLocalID;
    ///预留
    char                       Reserved1[20];
    ///可选，业务单元，（用户定义）
    TXeleBusinessUnitType      BusinessUnit;
    ///可选，订单所有类型
    TXeleTradeOwnerType        OwnerType;
    ///委托方式
    TXeleOperwayType           Operway;
    ///可选，交易前置id(暂未使用)
    TXeleExchangeIDType        ExchangeFrontID;
    ///委托方式扩展字段，部分券商支持
    char                       OperWayEx[3];
    ///可选，错误编号，拒单使用
    TXeleErrorIDType           ErrorID;
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

当客户登录柜台系统，获取交易权限后，用户可以使用reqCancelOrder进行撤单操作，无论是否通过风控，该请求会有报单操作回报onRspCancelOrder。若未通过交易所风控，还会返回onErrRtnCancelOrder回报信息。详见[撤单](#)

支持以下接口的撤单：

- 报单录入；
- 要约收购；
- 权证行权；

reqQryOrder 方法

功能：报单查询请求

函数原形：

```
int reqQryOrder(CXeleReqQryOrderField &inputField, int nRequestID)
```

参数：

- inputField：报单查询请求，其结构体CXeleReqQryOrderField为：

```
struct CXeleReqQryOrderField {
    ///可选，柜台报单编号int类型，不为0表示精确查询，本地报单编号、合约和时间条件都无效
```

```

TXeleOrderIDType          OrderSysID;
///可选，证券代码 范围查询，配合分页查询变量使用
TXeleSecuritiesIDType     SecuritiesID;
///可选，本地报单编号，不为0表示精确查询，合约和时间条件无效
TXeleOrderIDType          UserLocalID;
///可选，开始时间 格式：HHMMSSmmm(时分秒毫秒) 范围查询
TXeleShortTimeType        TimeStart;
///可选，结束时间 格式：HHMMSSmmm(时分秒毫秒) 范围查询
TXeleShortTimeType        TimeEnd;
///可选，分页查询起始值不填，默认从第一条开始
TXeleVolumeType           StartNum;
///可选，单次分页查询数量不填，使用系统参数表配置分页数量
TXeleVolumeType           Num;
///可选，排序类型
TXeleSortType             SortType;
///订单状态位图(支持 XTS-3.1.1149-1104dd4_7.9 之后版本)
///默认为0，全量查询
///支持指定状态
///示例： 查询正报、已报
///QryOrderStatus=QRYSTAT_REPORTING | QRYSTAT_REPORTED;
TXeleQryStatusType        QryOrderStatus;
///预留
char                      Reserved[117];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryOrder进行报单查询请求，可以在onRspQryOrder报单查询回报中查看相关信息。

【注意】

[查询方式]

1. 如果 OrderSysID条件有效，只以OrderSysID为条件查询，则其他条件不生效
2. 如果 OrderSysID 条件无效UserLocalID条件有效，只以UserLocalID为条件查询，则其他条件不生效
3. 如果 OrderSysID和 UserLocalID都无效，则可以按SecuritiesID和时间条件联合查询

[分页查询]

为避免API一次返回给用户数据记录数量过多，目前证券API 可以支持分页查询，每页查询上限的条目可以在系统参数表中进行配置。当此参数的值较大时，注意同时配置查询间隔，建议将查询间隔也同时调整（例如，每页查询值大于10000时，将查询间隔调整至5ms一次）

reqQryTrade 方法

功能：成交查询请求

函数原形：

```
int reqQryTrade(CXeleReqQryTradeField &inputField, int nRequestID)
```

参数：

- inputField：成交查询请求，其结构体CXeleReqQryTradeField如下：

```
struct CXeleReqQryTradeField {  
    ///可选，柜台报单编号int类型，不为0表示精确查询，本地报单编号、合约和时间条件都无效  
    TXeleOrderIDType          OrderSysID;  
    ///可选，证券代码 范围查询，配合分页查询变量使用  
    TXeleSecuritiesIDType      SecuritiesID;  
    ///可选，开始时间 格式：HHMMSSmmm(时分秒毫秒) 范围查询，配合分页查询变量使用  
    TXeleShortTimeType         TimeStart;  
    ///可选，结束时间 格式：HHMMSSmmm(时分秒毫秒) 范围查询，配合分页查询变量使用  
    TXeleShortTimeType         TimeEnd;  
    ///可选，用户本地报单编号，不为0表示精确查询，合约和时间条件无效  
    TXeleOrderIDType           UserLocalID;  
    ///可选，分页查询起始值不填，默认从第一条开始  
    TXeleVolumeType            StartNum;  
    ///可选，单次分页查询数量不填，使用系统参数表配置分页数量  
    TXeleVolumeType            Num;  
    ///可选，排序类型  
    TXeleSortType              SortType;  
    ///预留  
    char                        Reserved[115];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryTrade进行成交查询请求，可以在onRspQryTrade成交查询回报中查看相关信息。

【注意】

[查询方式]

1. 如果 OrderSysID条件有效，只以OrderSysID为条件查询，则其他条件不生效
2. 如果 OrderSysID 条件无效UserLocalID条件有效，只以UserLocalID为条件查询，则其他条件不生效
3. 如果 OrderSysID和 UserLocalID都无效，则可以按SecuritiesID和时间条件联合查询

[分页查询]

为避免API一次返回给用户数据记录数量过多，目前证券API 可以支持分页查询，每页查询上限的条目可以在系统参数表中进行配置。当此参数的值较大时，注意同时配置查询间隔，建议将查询间隔也同时调整（例如，每页查询值大于10000时，将查询间隔调整至5ms一次）

reqBTFundTransfer方法

功能：跨柜台资金调拨请求

函数原形：

```
int reqBTFundTransfer (CXeleReqBTCapTransferManagerField &inputField, int  
nRequestID)
```

参数：

- inputField：Manager 柜台间资金调拨请求，其结构体CXeleReqBTCapTransferManagerField如下：

```
struct CXeleReqBTCapTransferManagerField {  
    ///可选，机构代码  
    TXeleOrgIDType          OrgID;  
    ///必选，资金账号  
    TXeleUserIDType         AccountID;  
    ///预留，币种，默认使用人民币  
    TXeleCurrencyType       Currency;  
    ///必选，调拨金额  
    TXeleMoneyType          Fundamt;  
    ///必选，操作方向'1'：上交柜台调往深交柜台；'2'：深交柜台调往上交柜台  
    TXeleDirectionType      Direction;  
    ///预留  
    char                    Reserved[63];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqBTCapFundTransfer进行跨柜台间资金调拨请求，可以在onRspBTCapFundTransfer回报中查看相关信息。

reqQryBTCapFundTransferRecord方法

功能：跨柜台资金调拨记录查询请求

函数原形：

```
int reqQryBTCapFundTransferRecord (CXeleReqQryBTCapTransferManagerField  
&inputField, int nRequestID);
```

参数：

- inputField：跨柜台资金调拨记录查询请求，其结构体CXeleReqQryBTCapTransferField如下：

```
struct CXeleReqQryBTCapTransferManagerField {  
    ///机构代码  
    TXeleOrgIDType          OrgID;  
    ///资金账号  
    TXeleUserIDType         AccountID;  
    ///预留  
    char                    Reserved[63];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqBTCapTransferRecord进行跨柜台资金调拨记录查询请求，可以在onRspBTCapTransferRecord回报中查看相关信息。

reqInFund方法

功能：集中交易资金调入艾科柜台请求

函数原形：

```
int reqInFund (CXeleReqCapTransferField &inputField, int nRequestID)
```

参数：

- inputField：集中交易资金调入艾科柜台请求域，其结构体CXeleReqCapTransferField如下：

```
struct CXeleReqCapTransferField {  
    ///预留，机构代码（客户不填）  
    TXeleOrgIDType          OrgID;  
    ///必选，资金账号  
    TXeleUserIDType        AccountID;  
    ///预留，币种（客户不填）  
    TXeleCurrencyType      Currency;  
    ///预留，返还日期（客户不填）  
    TXeleRtnDateType       RtnDate;  
    ///必选，冻结、解冻金额  
    TXeleMoneyType         Fundamt;  
    ///预留，备注（客户不填）  
    TXeleRemarkType        RemarkMsg;  
    ///预留，只执行一次集中柜台冻结（客户不填）  
    TXeleCallonceFlag      Callonce;  
    ///分支机构（客户不填）  
    TXeleBranchNoType      BranchNo;  
    ///客户代码（客户不填）  
    TXeleUserIDType        CustID;  
    ///调入/调出类型,0:可用资金, 1:RTGS额度, 2:可取资金(部分券商使用)  
    TXeleTransferType      TransferType;  
    ///预留  
    char                   Reserved[43];  
}
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户可以使用reqInFund进行集中交易资金调入艾科柜台请求操作，可以在onRspInFund回报中查看相关信息。

reqOutFund方法

功能：艾科柜台资金调出集中交易请求

函数原形：

```
int reqOutFund (CXeleReqCapTransferField &inputField, int nRequestID)
```

参数：

- inputField：艾科柜台资金调出集中交易请求域，其结构体CXeleReqCapTransferField为：

```
struct CXeleReqCapTransferField {  
    ///预留，机构代码（客户不填）  
    TXeleOrgIDType          OrgID;  
    ///必选，资金账号  
    TXeleUserIDType         AccountID;  
    ///预留，币种（客户不填）  
    TXeleCurrencyType       Currency;  
    ///返还日期（客户不填）  
    TXeleRtnDateType        RtnDate;  
    ///必选，冻结、解冻金额  
    TXeleMoneyType          Fundamt;  
    ///预留，备注（客户不填）  
    TXeleRemarkType         RemarkMsg;  
    ///只执行一次集中柜台冻结（客户不填）  
    TXeleCallonceFlag       Callonce;  
    ///分支机构（客户不填）  
    TXeleBranchNoType       BranchNo;  
    ///客户代码（客户不填）  
    TXeleUserIDType         CustID;  
    ///预留  
    TXeleReserved1Type      Reserved[44];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户可以使用reqOutFund进行艾科柜台资金调出集中交易请求操作，可以在onRspOutFund回报中查看相关信息。

reqQryInOutFundRecord 方法

功能：集中交易资金调拨艾科柜台明细查询请求

函数原形：

```
int reqQryInOutFundRecord (CXeleReqQryCapTransferRecordField &inputField, int  
nRequestID)
```

参数：

- inputField：集中交易资金调拨艾科柜台明细查询请求域，其结构体CXeleReqQryCapTransferRecordField为：

```

struct CXeleReqQryCapTransferRecordField {
    ///机构代码(QFII中暂不使用)
    TXeleOrgIDType          OrgID;
    ///资金账号
    TXeleUserIDType         AccountID;
    ///预留
    TXeleReserved1Type      Reserved;
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryInOutFundRecord进行集中交易资金调拨艾科柜台明细查询请求，可以在onRspQryInOutFundRecord回报中查看相关信息。

SPI类接口

onRspInsertOrder方法

功能：报单录入应答

函数原形：

```

void onRspInsertOrder (CXeleRspOrderInsertField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField: 报单应答，其结构体CXeleRspOrderInsertField如下：

```

struct CXeleRspOrderInsertField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType     SecuritiesID;
    ///买卖方向
    TXeleDirectionType        Direction;
    ///开平标志（股票不填）
    TXeleOffsetFlagType       CmbOffsetFlag;
    ///限价价格
    TXelePriceType            LimitPrice;
    ///报单数量
    TXeleVolumeType           Volume;
    ///报单类型
    TXeleOrderTypeType        OrderType;
    ///有效期类型
    TXeleTimeConditionType     TimeCondition;
    ///交易所类型
    TXeleMarketType           Market;
    ///预埋单标记 '0':非预埋单 '1':预埋单
    TXelePreOrderFlag         PreOrderFlag;
};

```

```

    ///错误编号，拒单使用
    TXeleErrorIDType                ErrorID;
    ///客户端登录子节点
    TXeleSubClientIndexType          SubClientIndex;
    ///业务单元(用户定义)
    TXeleBusinessUnitType            BusinessUnit;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType           StrOrderSysID;
    ///投资者账号
    TXeleUserIDType                  AccountID;
    ///交易所报单编号(暂不使用)
    TXeleOrderExchangeIDType         OrderExchangeID;
    ///交易前置id(暂不使用)
    TXeleExchangeIDType              ExchangeFrontID;
    ///备兑标志(期权使用)
    TXeleCoveredFlagType              CoveredOrUncovered;
    ///要约收购人编码(该字段在onRspInsertOrder()接口中无效)
    TXeleTendererIDType              TendererID;
    ///预留
    char                             Reserved[6];
};

```

onErrRtnInsertOrder 方法

功能：报单录入错误回报

函数原形：

```

void onErrRtnInsertOrder (CXeleRspOrderInsertField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：报单错误回报，其结构体CXeleRspOrderInsertField如下：

```

struct CXeleRspOrderInsertField {
    ///柜台报单编号int类型
    TXeleOrderIDType                OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType                UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType           SecuritiesID;
    ///买卖方向
    TXeleDirectionType              Direction;
    ///开平标志（股票不填）
    TXeleOffsetFlagType             CmbOffsetFlag;
    ///限价价格
    TXelePriceType                   LimitPrice;
    ///报单数量
    TXeleVolumeType                 Volume;
    ///报单类型
    TXeleOrderTypeType              OrderType;
    ///有效期类型
    TXeleTimeConditionType           TimeCondition;
}

```



```

TXeleErrorIDType           ErrorID;
///客户端登录子节点
TXeleSubClientIndexType    SubClientIndex;
///业务单元(用户定义)
TXeleBusinessUnitType      BusinessUnit;
///订单所有类型
TXeleTradeOwnerType        OwnerType;
///柜台报单编号str类型
TXeleStrOrderSysIDType     StrOrderSysID;
///投资者账号
TXeleUserIDType            AccountID;
///交易所类型
TXeleMarketType            Market;
///交易所报单编号(暂未使用)
TXeleOrderExchangeIDType   OrderExchangeID;
///交易前置id(暂未使用)
TXeleExchangeIDType        ExchangeFrontID;
///要约收购人编码(上交所要约该字段有效)
TXeleTendererIDType        TendererID;
///预留
char                        Reserved1[5];
};

```

onErrRtnCancelOrder方法

功能：撤单错误回报

函数原形：

```

void onErrRtnCancelOrder (CXeleRspOrderActionField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：报单错误回报，其结构体CXeleRspOrderActionField如下：

```

struct CXeleRspOrderActionField {
///柜台报单编号int类型
TXeleOrderIDType           OrderSysID;
///用户本地报单编号
TXeleOrderIDType           UserLocalID;
///撤单报单编号
TXeleOrigSysIDType         OrigSysID;
///被撤单用户本地报单编号(柜台2.5以及2.5以上版本才支持该字段)
TXeleOrderIDType           OrigUserLocalID;
///被撤单柜台报单编号str类型(柜台2.5以及2.5以上版本才支持该字段)
TXeleStrOrderSysIDType     OrigStrOrderSysID;
///预埋单标记 '0':非预埋单 '1':预埋单
TXelePreOrderFlag          PreOrderFlag;
///预留
char                        Reserved0[7];
///错误编号
TXeleErrorIDType           ErrorID;
///客户端登录子节点
TXeleSubClientIndexType    SubClientIndex;
};

```

```

    ///业务单元(用户定义)
    TXeleBusinessUnitType          BusinessUnit;
    ///订单所有类型
    TXeleTradeOwnerType            OwnerType;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType         StrOrderSysID;
    ///投资者账号
    TXeleUserIDType                AccountID;
    ///交易所类型
    TXeleMarketType                Market;
    ///交易所报单编号
    TXeleOrderExchangeIDType       OrderExchangeID;
    ///交易前置id(暂未使用)
    TXeleExchangeIDType            ExchangeFrontID;
    ///要约收购人编码(上交所要约该字段有效)
    TXeleTendererIDType            TendererID;
    ///预留
    char                            Reserved1[5];
};

```

onRtnOrder 方法

功能：报单回报

函数原形：

```

void onRtnOrder(CXeleRtnOrderField *pRspField, CXeleRspInfo *pRspInfo, int
nRequestID, bool bIsLast);

```

参数：

- pRspField：报单回报。其结构体CXeleRtnOrderField如下：

```

struct CXeleRtnOrderField {
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType     SecuritiesID;
    ///买卖方向
    TXeleDirectionType        Direction;
    ///开平标志（股票暂未使用）
    TXeleOffsetFlagType        CmbOffsetFlag;
    ///价格(期权精度0.0001，股票精度0.001)
    TXelePriceType             LimitPrice;
    ///报单数量
    TXeleVolumeType            Volume;
    ///报单类型
    TXeleOrderTypeType         OrderType;
    ///有效期类型
    TXeleTimeConditionType     TimeCondition;
    ///交易所类型
    TXeleMarketType            Market;
    ///备兑标志(期权使用)
    TXeleCoveredFlagType        CoveredOrUncovered;
    ///柜台报单编号int类型

```

```

TXeleOrderIDType          OrderSysID;
///交易所报单编号(撤单不使用)
TXeleOrderExchangeIDType  OrderExchangeID;
///接受请求时间
TXeleTimeType             TransactTime;
///对应申报市价转限价的订单，这里填写转为限价订单的价格，
///单位：元（期权精度0.0001，股票精度0.001）
TXelePriceType            DiscretionPrice;
///累计成交数量
TXeleVolumeType           TradeVolume;
///未成交手数
///如果状态是撤单或者部撤，leavesVolume是已经成功撤单的数量；
///如果报单状态是部分成交，leavesVolume 表示未成交数量 = 报单数量 - 累计成交数量）
TXeleVolumeType           LeavesVolume;
///订单状态
TXeleOrderStatusType      OrderStatus;
///保证金(暂未使用)
TXeleMoneyType            Margin;
///冻结权利金(暂未使用)
TXeleMoneyType            FrozenPremium;
///冻结手续费(暂未使用)
TXeleMoneyType            FrozenFee;
///客户端登录子节点
TXeleSubClientIndexType   SubClientIndex;
///业务单元(用户定义)
TXeleBusinessUnitType     BusinessUnit;
///流水重构报文标记
TXeleRecoveryFlagType     RecoveryFlag;
///柜台报单编号str类型
TXeleStrOrderSysIDType    StrOrderSysID;
///投资者账号
TXeleUserIDType           AccountID;
///交易前置id,多网关版本使用
///返回值为实际报单的网关id + 1
///返回值为1代表是0号网关,2代表1号网关
TXeleExchangeIDType       ExchangeFrontID;
///被撤单用户本地报单编号(柜台2.5以及2.5以上版本才支持该字段)
TXeleOrderIDType          OrigUserLocalID;
///被撤单柜台报单编号int类型(柜台2.5以及2.5以上版本才支持该字段)
TXeleOrigSysIDType        OrigOrderSysID;
///被撤单柜台报单编号str类型(带号段)(柜台2.5以及2.5以上版本才支持该字段)
TXeleStrOrderSysIDType    OrigStrOrderSysID;
///要约收购人编码(上交所要约该字段有效)
TXeleTendererIDType       TendererID;
///交易所执行编号
TXeleOrderExchangeIDType  ExecID;
///预留
char                      Reserved[52];
};

```

onRtnTrade 方法

功能：成交回报

函数原形：

```
void onRtnTrade(CXeleRtnTradeField *pRspField CXeleRspInfo *pRspInfo, int  
nRequestID, bool bIsLast);
```

参数：

- pRspField：成交回报，其结构体CXeleRtnTradeField如下：

```
struct CXeleRtnTradeField {  
    ///用户本地报单编号  
    TXeleOrderIDType          UserLocalID;  
    ///证券代码  
    TXeleSecuritiesIDType     SecuritiesID;  
    ///买卖方向  
    TXeleDirectionType        Direction;  
    ///开平标志，股票不填  
    TXeleOffsetFlagType        CombOffsetFlag;  
    ///限价价格(期权精度0.0001，股票精度0.001)  
    TXelePriceType             LimitPrice;  
    ///数量  
    TXeleVolumeType            Volume;  
    ///报单类型  
    TXeleOrderTypeType         OrderType;  
    ///有效期类型  
    TXeleTimeConditionType      TimeCondition;  
    ///交易所类型  
    TXeleMarketType            Market;  
    ///备兑标志(期权使用)  
    TXeleCoveredFlagType        CoveredOrUncovered;  
    ///柜台报单编号int类型  
    TXeleOrderIDType            OrderSysID;  
    ///交易所报单编号  
    TXeleOrderExchangeIDType    OrderExchangeID;  
    ///暂不使用  
    TXeleOrderIDType            TradeID;  
    ///成交价格(期权精度0.0001，股票精度0.001)  
    TXelePriceType              TradePrice;  
    ///成交数量  
    TXeleVolumeType             TradeVolume;  
    ///未成交手数(报单数量-累计成交数量)  
    TXeleVolumeType             LeavesVolume;  
    ///订单执行时间  
    TXeleTimeType               TransactTime;  
    ///原有订单接受时间(上交使用)  
    TXeleTimeType               OrigTime;  
    ///订单状态  
    TXeleOrderStatusType        OrderStatus;  
    ///成交金额，精度0.01  
    TXeleMoneyType              TotalValueTraded;  
    ///佣金
```

```

TXeleSecuritiesCommissionType    Commission;
///印花税
TXeleStampTaxRateType            StampTax;
///过户费
TXeleTransferFeeRateType         Transfer;
///总手续费
TXeleTotalFeeType                TotalFee;
///保证金，暂未使用
TXeleMoneyType                   Margin;
///权利金，暂未使用
TXeleMoneyType                   Premium;
///客户端登录子节点
TXeleSubClientIndexType          SubClientIndex;
///业务单元(用户定义)
TXeleBusinessUnitType            BusinessUnit;
// 流水重构报文标记
TXeleRecoveryFlagType            RecoveryFlag;
///柜台报单编号str类型
TXeleStrOrderSysIDType           StrOrderSysID;
///投资者账号
TXeleUserIDType                  AccountID;
/// 交易所执行编号
TXeleOrderExchangeIDType         ExecID;
///交易前置id,多网关版本使用
///返回值为实际报单的网关id + 1
///返回值为1代表是0号网关,2代表1号网关
TXeleExchangeIDType              ExchangeFrontID;
/// 累计成交手数
TXeleVolumeType                  CumQty;
/// 本市场成份证券数
TXeleRecordNumType               RecordNum;
/// 成份证券代码
TXeleSecuritiesIDType            ComponentSecurityID;
/// 成份证券来源
TXeleSecuritySourceType           ComponentSecuritySource;
/// 使用老仓数量
TXeleQuantityType                UseOldQty;
/// 成份股份成交数量
TXeleQuantityType                ComponentTradeQty;
/// 现金替代金额
TXelePriceType                   ContAmtValue;
///预留
char                              Reserved[45];
};

```

onRspQryOrder 方法

功能：报单查询应答

函数原形：

```

void onRspQryOrder(CXeleRspQryOrderField *pRspField, CXeleRspInfo *pRspInfo, int
nRequestID, bool bIsLast) ;

```

参数：

- pRspField：报单查询应答，其结构体CXeleRspQryOrderField如下：

```
struct CXeleRspQryOrderField {
    ///用户报单编号
    TXeleOrderIDType          UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType     SecuritiesID;
    ///买卖方向
    TXeleDirectionType        Direction;
    ///开平标志
    TXeleOffsetFlagType        CmbOffsetFlag;
    ///价格
    TXelePriceType             LimitPrice;
    ///数量
    TXeleVolumeType            Volume;
    ///报单类型
    TXeleOrderTypeType         OrderType;
    ///有效期类型
    TXeleTimeConditionType     TimeCondition;
    ///交易所类型
    TXeleMarketType            Market;
    ///委托方式
    TXeleOperwayType           Operway;
    ///柜台报单编号int类型
    TXeleOrderIDType           OrderSysID;
    ///交易所报单编号
    TXeleOrderExchangeIDType   OrderExchangeID;
    ///接受请求时间
    TXeleTimeType              TransactTime;
    ///对应申报市价转限价订单，这里填写转为限价订单的价格，单位：元
    TXelePriceType              DiscretionPrice;
    ///累计成交数量
    TXeleVolumeType             TradeVolume;
    ///未成交手数
    ///如果状态是撤单或者部撤，leavesVolume是已经成功撤单的数量；
    ///如果报单状态是部分成交，leavesVolume表示未成交数量 = 报单数量 - 累计成交数量
    TXeleVolumeType             LeavesVolume;
    ///订单状态
    TXeleOrderStatusType        OrderStatus;
    ///佣金
    TXeleSecuritiesCommissionType Commission;
    ///印花税
    TXeleStampTaxRateType       StampTax;
    ///过户费
    TXeleTransferFeeRateType     Transfer;
    ///流量费
    TXeleTrafficFeeType          TrafficFee;
    ///总手续费
    TXeleTotalFeeType           TotalFee;
    ///保证金(暂未使用)
    TXeleMoneyType              Margin;
    ///冻结权利金(暂未使用)
    TXeleMoneyType              FrozenPremium;
```

```

///投资者账号
TXeleUserIDType AccountID;
///客户端登录子节点
TXeleSubClientIndexType SubClientIndex;
///业务单元(用户定义)
TXeleBusinessUnitType BusinessUnit;
///柜台报单编号str类型
TXeleStrOrderSysIDType StrOrderSysID
///报单类型
TXeleMessageType OrderMessageId;
///证券类别代码
TXeleSecuritiesType SecuritiesType;
///预埋单标记 '0':非预埋单 '1':预埋单
TXelePreOrderFlag PreOrderFlag;
///报单错误编号
TXeleErrorIDType ErrorId;
///分页查询结束值 用来作为下次查询的起始值, 来实现分页
TXeleVolumeType EndNum;
///分页查询条件下, 是否可以再次进行查询 只在最后一条回报中有效
TXeleIsLastType QryAgain;
///当前条件下查询到的回报总数
TXeleVolumeType TotalNum;
///投资者股东账号
TXeleInvestorIDType InvestorID;
///客户代码
TXeleUserIDType CustID;
///成交金额
TXelePriceType TradeAmount;
///主柜台未同步到备机的委托类型
TXeleUnSyncOrderFlag UnSyncOrderFlag;
///备兑标志(期权使用)
TXeleCoveredFlagType CoveredOrUncovered;
///被撤单柜台报单编号str类型(带号段)
TXeleStrOrderSysIDType OrigStrOrderSysID;
///报单来源,支持接入网关的柜台,此字段有效
TXeleOrderSourceType OrderSource;
///交易前置id
///返回值为实际报单的网关id + 1
///例如: 返回值为1代表是0号网关,2代表1号网关
TXeleExchangeIDIntType ExchangeFrontID;
///委托方式扩展字段,部分券商支持
char OperWayEx[3];
///预留
char Reserved[27];
};

```

onRspQryTrade 方法

功能：成交单查询应答

函数原形：

```
void onRspQryTrade(CXeleRspQryTradeField *pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);
```

参数：

- pRspField：成交单查询应答，其结构体CXeleRspQryTradeField如下：

```
struct CXeleRspQryTradeField {
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///交易日
    TXeleDateType             TradingDay;
    ///成交数量
    TXeleVolumeType           TradeVolume;
    ///未成交手数（报单数量-累计成交数量）
    TXeleVolumeType           LeavesVolume;
    ///证券代码
    TXeleSecuritiesIDType     SecuritiesID;
    ///申报时间
    TXeleShortTimeType        OrderTime;
    ///成交时间
    TXeleShortTimeType        TradeTime;
    ///成交价格
    TXelePriceType            TradePrice;
    ///成交金额
    TXelePriceType            TradeAmount;
    ///暂不使用
    TXeleOrderIDType          TradeID;
    ///交易所类型
    TXeleMarketType           Market;
    ///备兑标志(期权使用)
    TXeleCoveredFlagType      CoveredOrUncovered;
    ///柜台报单编号int类型
    TXeleOrderIDType          OrdersSysID;
    ///交易所报单编号
    TXeleOrderExchangeIDType  OrderExchangeID;
    ///买卖方向
    TXeleDirectionType        Direction;
    ///开平标志，股票不填
    TXeleOffsetFlagType       CmbOffsetFlag;
    ///佣金
    TXeleSecuritiesCommissionType Commission;
    ///印花税
    TXeleStampTaxRateType     StampTax;
    ///过户费
    TXeleTransferFeeRateType  Transfer;
    ///总手续费
    TXeleTotalFeeType         TotalFee;
    ///保证金，暂未使用
    TXeleMoneyType            Margin;
    ///权利金，暂未使用
    TXeleMoneyType            Premium;
    ///投资者账号
    TXeleUserIDType           AccountID;
```

```

///订单状态
TXeleOrderStatusType          OrderStatus;
///柜台报单编号str类型
TXeleStrOrderSysIDType        StrOrderSysID;
/// 交易所执行编号
TXeleOrderExchangeIDType      ExecID;
///客户端登录子节点
TXeleSubClientIndexType       SubClientIndex;
///分页查询结束值 用来作为下次查询的起始值，来实现分页
TXeleVolumeType               EndNum;
///分页查询条件下，是否可以再次进行查询 只在最后一条回报中有效
TXeleIsLastType               QryAgain;
///当前条件下查询到的回报总数
TXeleVolumeType               TotalNum;
///投资者股东账号
TXeleInvestorIDType           InvestorID;
///客户代码
TXeleUserIDType               CustID;
///报单来源,支持接入网关的柜台,此字段有效
TXeleOrderSourceType           OrderSource;
///报单类型
TXeleOrderTypeType            OrderType;
/// 本市场成份证券数
TXeleRecordNumType            RecordNum;
/// 成份证券代码
TXeleSecuritiesIDType         ComponentSecurityID;
/// 成份证券来源
TXeleSecuritySourceType        ComponentSecuritySource;
/// 使用老仓数量
TXeleQuantityType             UseOldQty;
/// 成份股份成交数量
TXeleQuantityType             ComponentTradeQty;
/// 现金替代金额
TXelePriceType                ContAmtValue;
///预留
char                           Reserved[14];
};

```

- blsLast：此次请求的响应的最后一次回调标志（若使用报单编号查询只会返回一条信息，可不判断 blsLast；若使用时间范围查询，回报就是一批信息，需使用blsLast来判断结束）

onRspBTFundTransfer 方法

功能：跨柜台资金调拨应答

函数原形：

```

void onRspBTFundTransfer (CXeleRspBTCapTransferManagerField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool blsLast) ;

```

参数：

- pRspField：跨柜台资金调拨应答，其结构体CXeleRspBTCapTransferManagerField如下：

```

struct CXeleRspBTCapTransferManagerField {

```

```

    ///全局唯一消息编号 从1开始递增
    TXeleUniqueNumberType          UniqueNumber;
    ///机构代码
    TXeleOrgIDType                  OrgID;
    ///资金账号
    TXeleUserIDType                 AccountID;
    ///币种
    TXeleCurrencyType               Currency;
    ///调拨金额
    TXeleMoneyType                  Fundamt;
    ///操作方向, '1': 上交柜台调往深交柜台; '2': 深交柜台调往上交柜台
    TXeleDirectionType              Direction;
    ///预留
    char                             Reserved[63];
};

```

onRtnBTFundTransfer方法

功能：跨柜台资金调拨结果返回

函数原形：

```

void onRtnBTFundTransfer (CXeleRtnBtCapTransferManagerField *pRspField
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：跨柜台资金调拨结果返回

其结构体CXeleRtnBtCapTransferManagerField如下：

```

struct CXeleRtnBtCapTransferManagerField {
    ///全局唯一消息编号
    TXeleUniqueNumberType          UniqueNumber;
    ///机构代码
    TXeleOrgIDType                  OrgID;
    ///资金账号
    TXeleUserIDType                 AccountID;
    ///币种
    TXeleCurrencyType               Currency;
    ///调拨金额
    TXeleMoneyType                  Fundamt;
    ///操作方向, '1': 上交柜台调往深交柜台; '2': 深交柜台调往上交柜台
    TXeleDirectionType              Direction;
    ///操作结果, '0': 调拨失败, '1': 调拨成功
    TXeleOrderStatusType            Status;
    ///预留
    char                             Reserved[63];
};

```

onRspQryBTFundTransferRecord方法

功能：跨柜台资金调拨记录查询应答

函数原形：

```
void onRspQryBTFundTransferRecord(CXeleRspQryBTCapTransferManagerField
*pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) ;
```

参数：

- pRspField：跨柜台资金调拨记录查询应答，其结构体CXeleRspQryBTCapTransferManagerField如下：

```
struct CXeleRspQryBTCapTransferManagerField {
    ///全局唯一消息编号
    TXeleUniqueNumberType        UniqueNumber;
    ///机构代码
    TXeleOrgIDType                OrgID;
    ///资金账号
    TXeleUserIDType              AccountID;
    ///币种
    TXeleCurrencyType            Currency;
    ///调拨金额
    TXeleMoneyType               Fund;
    ///操作方向，'1'： 上交柜台调往深交柜台；'2'： 深交柜台调往上交柜台
    TXeleDirectionType           Direction;
    ///操作时间
    TXeleShortTimeType           Time;
    ///错误编号
    TXeleErrorIDType             ErrorID;
    ///内部调用过程，不用关注：'1'接受到调拨消息，'2'发出调拨请求结束，'3'接受到调出结束消息，'4'发出调入消息请求结束，'5'接受到调入结束消息，'6'操作结束
    TXeleOrderStatusType         Status;
    ///资金冲正操作标记 '0'： 非资金冲正操作，'1'： 资金冲正操作
    TXeleIsFundReversalType      IsFundReversal;
    ///资金冲正柜台 SS表示上交柜台，SZ表示深交柜台 '--'表示无效
    TXeleReversalCounterType     ReversalCounter;
    ///资金冲正结果标记 '1'： 正在处理中 '2'： 处理成功 '3'： 处理失败 '-'表示无效
    TXeleReversalResultType      ReversalResult;
    ///预留
    char                          Reserved[62];
};
```

onRspInFund 方法

功能：集中交易资金调入艾科柜台应答

函数原形：

```
void onRspInFund (CXeleRspCapTransferField *pRspField, CXeleRspInfo *pRspInfo,
int nRequestID, bool bIsLast);
```

参数：

- pRspField: 集中交易资金调入艾科柜台应答，其结构体CXeleRspCapTransferField如下：

```
struct CXeleRspCapTransferField {
    ///机构代码
    TXeleOrgIDType          OrgID;
    ///资金账号
    TXeleUserIDType         AccountID;
    ///币种
    TXeleCurrencyType       Currency;
    ///操作流水号
    TXeleSnoType            Sno;
    ///调取前可用(恒生柜台：该字段暂未使用)
    TXeleMoneyType          BefFundavl;
    ///资金余额
    TXeleMoneyType          Fundbal;
    ///调整后可用(恒生柜台：该字段暂未使用)
    TXeleMoneyType          AftFundavl;
    ///调整前可取(恒生柜台：该字段暂未使用)
    TXeleMoneyType          BefCashbal;
    ///调整后可取(恒生柜台：该字段暂未使用)
    TXeleMoneyType          AftCashbal;
    ///调入资金,调取资金
    TXeleMoneyType          Fundamt;
    ///集中交易柜台响应错误码
    TXeleCentralTradingErrorIDType  ctErrorId;
    ///集中交易柜台响应错误信息
    TXeleCentralTradingErrorMsgType ctErrorMsg;
    ///fpga资金是否更新
    TXeleIsupdateFpgaFundType  IsupdateFpgaFund;
    ///调入/调出类型,0:可用资金,1:RTGS额度,2:可取资金(部分券商使用)
    TXeleTransferType         TransferType;
    ///预留
    char                      Reserved[10];
}
```

onRspOutFund方法

功能：艾科柜台资金调出集中交易应答

函数原形：

```
void onRspUnfreezeCap(CXeleRspCapTransferField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) ;
```

参数：

- pRspField：艾科柜台资金调出集中交易应答，其结构体CXeleRspCapTransferField如下：

```
struct CXeleRspCapTransferField {
    ///机构代码
    TXeleOrgIDType          OrgID;
    ///资金账号
```

```

TXeleUserIDType      AccountID;
///币种
TXeleCurrencyType    Currency;
///操作流水号
TXeleSnoType         Sno;
///调取前可用(恒生柜台：该字段暂未使用)
TXeleMoneyType       BefFundavl;
///资金余额
TXeleMoneyType       Fundbal;
///调整后可用(恒生柜台：该字段暂未使用)
TXeleMoneyType       AftFundavl;
///调整前可取(恒生柜台：该字段暂未使用)
TXeleMoneyType       BefCashbal;
///调整后可取(恒生柜台：该字段暂未使用)
TXeleMoneyType       AftCashbal;
///调入资金,调取资金
TXeleMoneyType       Fundamt;
///集中交易柜台响应错误码
TXeleCentralTradingErrorIdType  ctErrorId;
///集中交易柜台响应错误信息
TXeleCentralTradingErrorMsgType  ctErrorMsg;
///预留
char                  Reserved[12]; };

```

onRspQryInOutFundRecord方法

功能：集中交易资金调拨艾科柜台明细查询应答

函数原形：

```

void onRspQryInOutFundRecord (CXeleRspQryCapTransferRecordField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast)

```

参数：

- pRspField：集中交易资金调拨艾科柜台明细查询域，其结构体 CXeleRspQryCapTransferRecordField如下：

```

struct CXeleRspQryCapTransferRecordField {
///机构代码(QFII中暂不使用)
TXeleOrgIDType      OrgID;
///资金账号
TXeleUserIDType     AccountID;
///冻结/解冻金额
TXeleMoneyType       Fundamt;
///操作方向，'1':冻结，'2':解冻
TXeleDirectionType  Direction;
///操作流水号
TXeleSnoType         Sno;
///操作时间
TXeleShortTimeType   Time;
///交易所类型
TXeleMarketType      Market;

```

```

    ///操作来源
    TXeleActionSourceType          ActionSource;
    ///预留
    char                            Reserved[62];
};

```

证券相关

API类接口

reqQryRate 方法

功能：费率(印花税率、过户费率、佣金率、流量费)查询请求

函数原形：

```

int reqQryRate (CXeleReqQryStockFeeField &inputField, int nRequestID)

```

参数：

- inputField：费率(印花税率、过户费率、佣金率、流量费)查询，其结构体 CXeleReqQryStockFeeField如下：

```

struct CXeleReqQryStockFeeField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///证券代码
    TXeleSecuritiesIDType    SecuritiesID;
    ///营业部代码(暂未使用)
    TXeleDepartmentIDType    DepartID;
    ///证券类型
    TXeleSecuritiesType       SecuritiesType;
    ///预留
    char                      Reserved[127];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryRate进行费率(印花税率、过户费率、佣金率、流量费、最小佣金、多冻值、ETF申赎相关费率)查询请求，可以在onRspQryRate响应中查看相关信息。

reqQryCentralTradingFund方法

功能：集中交易资金查询请求

函数原形：

```

int reqQryCentralTradingFund (CXeleReqQryCentralTradingFundField &inputField,
int nRequestID)

```

参数：

- inputField：集中交易柜台资金查询请求域，其结构体CXeleReqQryCentralTradingFundField如下：

```
/// 查询集中交易资金请求
struct CXeleReqQryCentralTradingFundField{
    ///机构代码（客户不填）
    TXeleOrgIDType          OrgID;
    ///客户代码（客户不填）
    TXeleUserIDType         CustId;
    ///资金账号
    TXeleUserIDType         AccountID;
    ///币种（客户选填）
    TXeleCurrencyType       Currency;
    ///备注
    TXeleRemarkType         RemarkMsg;
    ///预留
    TXeleReservedType       Reserved;
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryCentralTradingFund进行证券集中交易柜台资金明细查询请求，可以在onRspQryCentralTradingFund回报中查看相关信息。

reqInPosition方法

功能：集中交易持仓调入艾科柜台请求(中信柜台专用，其他券商不支持)

函数原形：

```
int reqInPosition(CXeleReqPosTransferField &inputField , int nRequestID)
```

参数：

- inputField：持仓划拨请求域，其结构体CXeleReqPosTransferField如下：

/// 持仓划拨请求(中信柜台专用，其他券商不支持)

```
/// 持仓划拨请求(中信柜台专用，其他券商不支持)
struct CXeleReqPosTransferField {
    ///机构代码(可选)
    TXeleOrgIDType          OrgID;
    ///资金账号
    TXeleUserIDType         AccountID;
    ///股东账户(不使用)
    TXeleInvestorIDType     InvestorID;
    ///市场代码(可选)
    TXeleMarketIDType       MarketID;
    ///证券代码
    TXeleSecuritiesIDType   SecuritiesID;
    ///股份发生数
```

```

TXeleSignedVolumeType      Volume;
///划拨方向 :1 冻结 , 2 解冻(可选)
TXeleDirectionType         Direction;
///备注(不使用)
TXeleRemarkType            RemarkMsg;
///只执行一次集中柜台冻结(不使用)
TXeleCallOnceFlag          CallOnce;
///分支机构(不使用)
TXeleBranchNoType          BranchNo;
///客户代码(不使用)
TXeleUserIDType            CustID;
///预留
char                        Reserved[44];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqInPosition划拨集中柜台持仓到艾科柜台，可以在onRspInPosition回报中查看相关信息。

reqOutPosition方法

功能：集中交易持仓调出艾科柜台请求(中信柜台专用，其他券商不支持)

函数原形：

```
int reqOutPosition (CXeleReqPosTransferField &inputField , int nRequestID)
```

参数：

- inputField：持仓划拨请求域，其结构体CXeleReqPosTransferField参见 reqInPosition中描述
- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqOutPosition划拨集中柜台持仓从艾科柜台调出，可以在onRspOutPosition回报中查看相关信息。

reqQryInOutPositionRecord方法

功能：集中交易持仓调拨艾科柜台明细查询请求(中信柜台专用，其他券商不支持)

函数原形：

```
int reqQryInOutPositionRecord (CXeleReqQryPositionTransferRecordField
&inputField, int nRequestID)
```

参数：

- inputField：集中交易柜台资金查询请求域，其结构体CXeleReqQryPositionTransferRecordField如下：

///集中交易持仓调拨艾科柜台明细查询请求(中信柜台专用，其他券商不支持)

```
struct CXeleReqQryPositionTransferRecordField {  
    ///资金账号  
    TXeleUserIDType      AccountID;  
    ///证券代码  
    TXeleSecuritiesIDType SecuritiesID;  
    ///预留  
    TXeleReservedType     Reserved;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户后持仓划拨后，可以使用reqQryInOutPositionRecord查询持仓划拨记录，可以在onRspQryInOutPositionRecord回报中查看相关信息。

reqInOutPosition方法

功能：集中交易持仓调入/调出艾科柜台请求(部分券商使用)

函数原形：

```
int reqInOutPosition (CXeleReqInOutPositionField&inputField , int nRequestID)
```

参数：

- inputField：持仓划拨请求域，其结构体CXeleReqInOutPositionField如下：

```
/// 持仓划拨请求(部分券商使用)  
struct CXeleReqInOutPositionField {  
    ///资金账号(必填)  
    TXeleUserIDType      AccountID;  
    ///证券代码(必填)  
    TXeleSecuritiesIDType SecuritiesID;  
    ///划拨数量(必填)  
    TXeleVolumeType      Volume;  
    ///划拨方向：'1' 划入，'2' 划出(必填)  
    TXeleInOutDirectionType InOutDirection;  
    ///预留  
    char                  Reserved[128];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqInOutPosition把集中柜台持仓调入/调出到艾科柜台，可以在onRspInOutPosition回报中查看相关信息。

reqQryInOutPositionDetails方法

功能：集中交易持仓调拨艾科柜台明细查询请求(部分券商使用)

函数原形：

```
int reqQryInOutPositionDetails (CXeleReqQryInOutPositionDetailsField  
&inputField, int nRequestID)
```

参数：

- inputField：集中交易持仓调拨艾科柜台明细查询请求域，其结构体CXeleReqQryInOutPositionDetailsField如下：

```
/// CXeleReqQryInOutPositionDetailsField（部分券商使用）  
struct CXeleReqQryInOutPositionDetailsField {  
    ///资金账号（必填）  
    TXeleUserIDType      AccountID;  
    ///证券代码（可选）  
    TXeleSecuritiesIDType SecuritiesID;  
    ///预留  
    TXeleReservedType     Reserved;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户后持仓划拨后，可以使用reqQryInOutPositionDetails查询持仓划拨记录，可以在onRspQryInOutPositionDetails回报中查看相关信息。

reqQryGateWaysRecord方法

功能：账号的可用网关信息查询请求

函数原形：

```
int reqQryGateWaysRecord (CXeleReqQryAvailableGatewayRecordField &inputField,  
int nRequestID)
```

参数：

- inputField 股东账户可用网关信息查询请求域，其结构体CXeleReqQryAvailableGatewayRecordField如下：

```

struct CXeleReqQryAvailableGatewayRecordField{
    ///必选，资金账户
    TXeleUserIDType          AccountID;
    ///预留
    char                      Reserved0;
    ///预留
    TXeleReservedType        Reserved;
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryGateWaysRecord查询自己的可用网关信息请求，可以在onRspQryGateWaysRecord回报中查看相关信息。

该接口不再更新，推荐使用交易所网关查询请求reqQryExChangeGateWay接口。

reqQryBtFund方法

功能：沪深柜台资金账户查询请求

函数原形：

```

int reqQryBtFund (CXeleReqQryClientAccountFundManagerField &inputField, int
nRequestID)

```

参数：

- inputField：沪深柜台资金账户查询请求，其结构体CXeleReqQryClientAccountField如下：

```

struct CXeleReqQryClientAccountFundManagerField{
    ///必选，资金账号
    TXeleUserIDType          AccountID;
    ///必选，交易所类型
    TXeleMarketType          Market;
    ///预留
    char                      Reserved[48];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryBtFund进行沪深柜台资金账户查询请求，可以在onRspQryBtFund回报中查看相关信息。

reqQryFund方法

功能：证券资金查询请求

函数原形：

```
int reqQryFund(CXeleReqQryClientAccountField &inputField, int nRequestID)
```

参数：

- inputField：沪深柜台资金账户查询请求，其结构体CXeleReqQryClientAccountField如下：

```
struct CXeleReqQryClientAccountField {  
    ///必选，资金账户  
    TXeleUserIDType          AccountID;  
    ///预留  
    TXeleReservedType        Reserved;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryFund进行证券资金查询请求，可以在onRspQryFund回报中查看相关信息。

reqQryPosition方法

功能：证券持仓查询请求

函数原形：

```
int reqQryPosition(CXeleReqQryPositionField &inputField, int nRequestID)
```

参数：

- inputField：证券持仓查询请求域（SecuritiesID查询，该字段不填查所有），其结构体CXeleReqQryPositionField为：

```
struct CXeleReqQryPositionField {  
    ///必选，资金账户  
    TXeleUserIDType          AccountID;  
    ///可选，证券代码，不填查所有  
    TXeleSecuritiesIDType    SecuritiesID;  
    ///交易所类型,不填查询登录的市场，填写查询对应的市场，'a' 查询全部  
    TXeleMarketType          Market;  
    ///预留  
    char                      Reserved[127];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryPosition进行证券持仓查询，可以在onRspQryPosition回报中查看相关信息。

reqQrySecurities 方法

功能：证券信息查询请求

```
int reqQrySecurities(CXeleReqQrySecuritiesField &inputField, int nRequestID);
```

参数：

- inputField：证券合约查询请求，其结构体CXeleReqQrySecuritiesField如下：

```
struct CXeleReqQrySecuritiesField {  
    ///可选，证券代码，不填查所有  
    TXeleSecuritiesIDType      SecuritiesID;  
    ///预留  
    TXeleReservedType          Reserved;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQrySecurities进行证券信息查询请求，可以在onRspQrySecurities回报中查看相关信息。

reqQryRightsAndInterests方法

功能：权益查询-新股额度查询请求

函数原形：

```
int reqQryRightsAndInterests (CXeleReqQryStockQuotaField &inputField, int  
nRequestID)
```

参数：

- inputField：权益查询-新股额度查询域，其结构体CXeleReqQryStockQuotaField如下：

```
struct CXeleReqQryStockQuotaField {  
    ///资金账号  
    TXeleUserIDType      AccountID;  
    ///合约类型  
    TXeleSecuritiesType  SecuritiesType;  
    ///证券代码  
    TXeleSecuritiesIDType SecuritiesID;  
    ///预留  
    TXeleReservedType     Reserved;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryRightsAndInterests进行权益查询-新股额度查询请求，可以在onRspQryRightsAndInterests回报中查看相关信息。

reqQryInvestorInfo方法

功能：股东账号信息查询请求

函数原形：

```
int reqQryInvestorInfo (CXeleReqQryInvestorInfoField &inputField, int  
nRequestID)
```

参数：

- inputField：权益查询股东账号信息查询请求域，其结构体CXeleReqQryInvestorInfoField如下：

```
struct CXeleReqQryInvestorInfoField {  
    ///必选，资金账户  
    TXeleUserIDType          AccountID;  
    ///预留  
    TXeleReservedType        Reserved;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryInvestorInfo查询自己的股东账号等其他相关信息请求，可以在onRspQryInvestorInfo回报中查看相关信息。

reqQryPrcProgramInfo方法

功能：程序化风控信息查询请求

函数原形：

```
int reqQryPrcProgramInfo(CXeleReqQryPrcProgramInfoField &inputField, int  
nRequestID)
```

参数：

- inputField 程序化风控信息查询请求域，其结构体CXeleReqQryPrcProgramInfoField如下：

```
struct CXeleReqQryPrcProgramInfoField {  
    ///必选，资金账户  
    TXeleUserIDType          AccountID;  
    ///程序化风控类别  
    ///0 返回全部，1 全天报撤笔数限制风控，2流速风控  
    TXelePrcProgramType      PrcProgramType;  
    ///预留  
    char                      Reserved[20];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryPrcProgramInfo查询自己的程序化风控信息，可以在onRspQryPrcProgramInfo回报中查看相关信息

reqQryETFCreationRedemptionSecurities方法

功能：ETF申赎母基金信息查询请求

函数原形：

```
reqQryETFCreationRedemptionSecurities(CXeleReqQryETFCreationRedemptionSecuritiesField &inputField, int nRequestID)
```

参数：

- inputField **ETF** 申赎母基金信息查询请求域，其结构体CXeleReqQryETFCreationRedemptionSecuritiesField 如下：

```
struct CXeleReqQryETFCreationRedemptionSecuritiesField {
    ///证券代码选填，不填查所有
    ///母基金证券代码
    TXeleSecuritiesIDType SecuritiesID;
    ///预留
    char Reserved[120];
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryETFCreationRedemptionSecurities查询ETF申赎母基金信息，可以在回报onRspQryETFCreationRedemptionSecurities中查看相关信息

reqQryETFCreationRedemptionComponentSecurities方法

功能：ETF申赎成股信息查询请求

函数原形：

```
reqQryETFCreationRedemptionComponentSecurities(CXeleReqQryETFCreationRedemptionComponentSecuritiesField &inputField, int nRequestID)
```

参数：

- inputField **ETF** 申赎成股信息查询请求域，其结构体CXeleReqQryETFCreationRedemptionComponentSecuritiesField 如下：

```

struct CXeleReqQryETFCreationRedemptionComponentSecuritiesField {
    ///选填，证券代码，指母基金证券代码，不填默认返回所有
    ///如果不填，则查询所有ETF基金的成份股信息；如果填了证券代码，则只返回此证券代码对应的成份
    股信息
    TXeleSecuritiesIDType    SecuritiesID;
    ///成份证券代码 选填 不填查所有
    TXeleSecuritiesIDType    ComponentSecurityId;
    ///预留
    char                      Reserved[112];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryETFCreationRedemptionComponentSecurities查询ETF申赎成股信息，可以在回报onRspQryETFCreationRedemptionSecurities中查看相关信息

reqQryExChangeGateWay方法

功能：交易所网关查询请求

函数原形：

```

int reqQryExChangeGateWay(CXeleReqQryExChangeGateWayField &inputField, int
nRequestID)

```

参数：

- inputField 交易所网关信息查询请求域，其结构体CXeleReqQryExChangeGateWayField如下：

```

struct CXeleReqQryPrcProgramInfoField {
    ///必选，资金账户
    TXeleUserIDType    AccountID;
    ///预留
    char                Reserved[49];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryExChangeGateWay查询交易所网关信息，可以在onRspQryExChangeGateWay回报中查看相关信息

reqQryQFIOrder 方法

功能：QFI报单查询请求

函数原形：

```

int reqQryQFIOrder(CXeleReqQryQFIOrderField &inputField, int nRequestID)

```

参数：

- **inputField** : QFI报单查询请求，其结构体CXeleReqQryQFIOrderField为：

```

struct CxleReqQryQFIOrderField {
    ///可选，柜台报单编号int类型，不为0表示精确查询，本地报单编号、合约和时间条件都无效
    TXleOrderIDType                OrdersSysID;
    ///可选，证券代码 范围查询，配合分页查询变量使用
    TXleSecuritiesIDType           SecuritiesID;
    ///可选，本地报单编号，不为0表示精确查询，合约和时间条件无效
    TXleQFIOrderIDType             UserLocalID;
    ///可选，开始时间 格式：HHMMSSmmm(时分秒毫秒) 范围查询
    TXleShortTimeType              TimeStart;
    ///可选，结束时间 格式：HHMMSSmmm(时分秒毫秒) 范围查询
    TXleShortTimeType              TimeEnd;
    ///可选，分页查询起始值不填，默认从第一条开始
    TXleVolumeType                  StartNum;
    ///可选，单次分页查询数量不填，使用系统参数表配置分页数量
    TXleVolumeType                  Num;
    ///可选，排序类型
    TXleSortType                    SortType;
    ///订单状态位图(支持 XTS-3.1.1149-1104dd4_7.9 之后版本)
    ///默认为0，全量查询
    ///支持指定状态
    ///示例： 查询正报、已报
    ///QryOrderStatus=QRYSTAT_REPORTING | QRYSTAT_REPORTED;
    TXleQryStatusType              QryOrderStatus;
    ///预留
    char                            Reserved[89];
};

```

- nRequestID : 请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryQFIOrder进行QFI报单查询请求，可以在onRspQryQFIOrder回报中查看相关信息。

【注意】

[查询方式]

1. 如果 OrderSysID条件有效，只以OrderSysID为条件查询，则其他条件不生效
2. 如果 OrderSysID 条件无效UserLocalID条件有效，只以UserLocalID为条件查询，则其他条件不生效，此处的UserLocalID为字符型，最大支持32个字节
3. 如果 OrderSysID和 UserLocalID都无效，则可以按SecuritiesID和时间条件联合查询

[分页查询]

为避免API一次返回给用户数据记录数量过多，目前证券API 可以支持分页查询，每页查询上限的条目可以在系统参数表中进行配置。当此参数的值较大时，注意同时配置查询间隔，建议将查询间隔也同时调整（例如，每页查询值大于10000时，将查询间隔调整至5ms一次）

reqQryQFIITrade 方法

功能：QFII成交查询请求

函数原形：

```
int reqQryQFIITrade(CXeleReqQryQFIITradeField &inputField, int nRequestID)
```

参数：

- inputField：QFII成交查询请求，其结构体CXeleReqQryQFIITradeField如下：

```
struct CXeleReqQryQFIITradeField {  
    ///可选，柜台报单编号int类型，不为0表示精确查询，本地报单编号、合约和时间条件都无效  
    TXeleOrderIDType          OrdersSysID;  
    ///可选，证券代码 范围查询，配合分页查询变量使用  
    TXeleSecuritiesIDType      SecuritiesID;  
    ///可选，开始时间 格式：HHMMSSmmm(时分秒毫秒) 范围查询，配合分页查询变量使用  
    TXeleShortTimeType         TimeStart;  
    ///可选，结束时间 格式：HHMMSSmmm(时分秒毫秒) 范围查询，配合分页查询变量使用  
    TXeleShortTimeType         TimeEnd;  
    ///可选，用户本地报单编号，不为0表示精确查询，合约和时间条件无效  
    TXeleQFIIDOrderIDType      UserLocalID;  
    ///可选，分页查询起始值不填，默认从第一条开始  
    TXeleVolumeType            StartNum;  
    ///可选，单次分页查询数量不填，使用系统参数表配置分页数量  
    TXeleVolumeType            Num;  
    ///可选，排序类型  
    TXeleSortType              SortType;  
    ///Etf成交记录查询标志，qfii柜台使用  
    TXeleQryEtfTradeFlag       QryEtfTradeFlag;  
    ///预留  
    char                        Reserved[114];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryQFIITrade进行QFII成交查询请求，可以在onRspQryQFIITrade回报中查看相关信息。

reqQryQFIIPosition 方法

功能：QFII证券持仓查询请求

函数原形：

```
int reqQryQFIIPosition(CXeleReqQryQFIIPositionField& inputField, int nRequestID)
```

参数：

- inputField：QFII证券持仓查询请求域（SecuritiesID查询，该字段不填查所有），其结构体CXeleReqQryQFIIPositionField如下：

```
struct CXeleReqQryQFIIPositionField {
    ///必选，资金账户
    TXeleUserIDType          AccountID;
    ///可选，证券代码，不填查所有
    TXeleSecuritiesIDType    SecuritiesID;
    ///可选，不填查询登录接口中对应的市场的持仓记录
    TXeleMarketType          Market;
    ///预留
    char                      Reserved[104];
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryQFIIPosition进行证券持仓查询，可以在onRspQryQFIIPosition回报中查看相关信息。

reqTenderOfferOrder 方法

功能：要约收购请求

函数原形：

```
int reqTenderOfferOrder(CXeleReqTenderOfferOrderField &inputField, int
nRequestID);
```

参数：

- inputField：要约收购请求域，其结构体CXeleReqTenderOfferOrderField如下：

```
///要约收购请求（48字节）
struct CXeleReqTenderOfferOrderField {
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType    SecuritiesID;
    ///买卖方向：'7'-要约预受，'8'-要约撤销
    TXeleDirectionType        Direction;
    ///报单数量
    TXeleVolumeType           Volume;
    ///委托方式
    TXeleOperwayType          Operway;
    ///交易前置id，与买卖指定前置逻辑一致，只允许通过软网关报单；
    ///查询接口返回的网关需要转换成整型，如'1'转换成1后填充该字段
    ///指定软件网关的有效值范围是：[33，250]，且实际使用的报单网关编号为x-1；
    ///例如，假配置配置的网关编号为 x=33，则实际使用的是 33-1=32 号网关；
    ///该字段为必填字段。其中，255 表示自动轮询。
    TXeleExchangeIDType       ExchangeFrontID;
    ///要约收购人编码
    TXeleTendererIDType        TendererID;
    ///委托方式扩展字段，部分券商支持
    char                      OperwayEx[3];
};
```

```

    ///预留
    char Reserved[20];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：

1. 当用户成功登录柜台，获取交易权限后，使用reqTenderOfferOrder，可以进行要约收购操作；
2. 无论报单录入是否通过风控，都会返回一个onRspTenderOfferOrder回报；
3. 如果未通过交易所风控，则还会返回onErrRtnInsertOrder回报；
4. 仅上交所支持该接口，深交所不支持；

reqWarrantExerciseOrder方法

功能：权证行权请求

函数原形：

```

int reqWarrantExerciseOrder(CXeleReqWarrantExerciseOrderField &inputField, int nRequestID);

```

参数：

- inputField：权证行权请求域，其结构体CXeleReqWarrantExerciseOrderField如下：

```

///权证行权请求（48字节）
struct CXeleReqWarrantExerciseOrderField {
    ///用户本地报单编号
    TXeleOrderIDType UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType SecuritiesID;
    ///买卖方向：固定为'1'，其他值无效
    TXeleDirectionType Direction;
    ///限价价格
    TXelePriceType LimitPrice;
    ///报单数量
    TXeleVolumeType Volume;
    ///报单类型：固定为'2'-限价
    TXeleOrderTypeType OrderType;
    ///委托方式
    TXeleOperwayType Operway;
    ///交易前置id，与买卖指定前置逻辑一致，只允许通过软网关报单；
    ///查询接口返回的网关需要转换成整型，如'1'转换成1后填充该字段
    ///指定软件网关的有效值范围是：[33，250]，且实际使用的报单网关编号为x-1；
    ///例如，假设置配的网关编号为 x=33，则实际使用的是 33-1=32 号网关；
    ///该字段为必填字段。其中，255 表示自动轮询。
    TXeleExchangeIDType ExchangeFrontID;
    ///委托方式扩展字段，部分券商支持
    char OperWayEx[3];
    ///预留
    char Reserved[17];
};

```

```
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：

1. 当用户成功登录柜台，获取交易权限后，使用reqWarrantExerciseOrder，可以进行权证行权操作；
2. 无论报单录入是否通过风控，都会返回一个onRspWarrantExerciseOrder回报；
3. 如果未通过交易所风控，则还会返回onErrRtnInsertOrder回报；
4. 仅深交所支持该接口，上交所不支持；

SPI类接口

onRspQryRate方法

功能：费率(印花税率、过户费率、佣金率、流量费)查询应答

函数原形：

```
void onRspQryRate (CXeleRspQryStockFeeField *pRspField, CXeleRspInfo *pRspInfo,
int nRequestID, bool bIsLast);
```

参数：

- pRspField：费率(印花税率、过户费率、佣金率、流量费)查询应答，其结构体CXeleRspQryStockFeeField为：

```
struct CXeleRspQryStockFeeField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///证券代码
    TXeleSecuritiesIDType    SecuritiesID;
    ///营业部代码
    TXeleDepartmentIDType    DepartID;
    ///印花税率
    TXeleStampTaxRateType     StampTaxRate;
    ///过户费率
    TXeleTransferFeeRateType  TransferFee;
    ///佣金率
    TXeleSecuritiesCommissionType CommRate;
    ///流量费
    TXeleTrafficFeeType       TrafficFee;
    ///交易所类型
    TXeleMarketType           Market;
    ///证券类型
    TXeleSecuritiesType       SecuritiesType;
    ///最小佣金
    TXeleMinCommissionType    MinCommission;
    ///多冻值
    TXeleMoreFreeze           MoreFreeze;
```

```

    ///ETF申赎附加费,若该合约不支持申赎, 值为0
    TXeleETFCreationRedemptionExtraFee          ETFCreationRedemptionExtraFee;
    ///ETF申赎佣金费率,若该合约不支持申赎, 值为0
    TXeleETFCreationRedemptionCommFeeRate        ETFCreationRedemptionCommFeeRate;
    ///ETF申赎过户费,若该合约不支持申赎, 值为0
    TXeleETFCreationRedemptionTransferFeeRate    ETFCreationRedemptionTransferFeeRate;
    ///预留
    char                                           Reserved[86]
};

```

onRspQryCentralTradingFund

功能：集中交易柜台资金查询应答

函数原形：

```

void onRspQryCentralTradingFund(CXeleRspQryCentralTradingFundField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) { }

```

参数：

- pRspField：集中交易柜台资金查询应答域，其结构体CXeleRspQryCentralTradingFundField如下：

```

struct CXeleRspQryCentralTradingFundField {
    ///机构代码
    TXeleOrgIDType      OrgID;
    ///客户代码
    TXeleUserIDType     CustId;
    ///资金账号
    TXeleUserIDType     AccountID;
    ///币种
    TXeleCurrencyType   Currency;
    ///资金余额
    TXeleMoneyType      Fundbal;
    ///资金可用
    TXeleMoneyType      Fundavl;
    ///资金资产
    TXeleMoneyType      Fund;
    ///主资金标志
    TXeleFundType       Fundseq;
    ///预留128字节
    TXeleReservedType   Reserved;
};

```

onRspInPosition方法

功能：集中交易持仓调入艾科柜台应答(中信柜台专用，其他券商不支持)

函数原形：

```
void onRspInPosition (CxeleRspPosTransferField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast)
```

参数：

- pRspField：持仓划拨应答域，其结构体CXeleRspPosTransferField如下：

```
struct CXeleRspPosTransferField{
    ///机构代码
    TXeleOrgIDType          OrgID;
    ///资金账号
    TXeleUserIDType         AccountID;
    ///股东账户
    TXeleInvestorIDType     InvestorID;
    ///证券代码
    TXeleSecuritiesIDType   SecuritiesID;
    ///fpga仓位是否更新
    TXeleIsUpdateFpgaFundType IsUpdateFpgaStock;
    ///操作流水号
    TXeleSnoType            Sno;
    ///股份发生数
    TXeleSignedVolumeType   stkeffect;
    ///错误信息
    TXeleCentralTradingErrorMsgType ptErrorMsg;
    ///预留
    char                    Reserved[44];
};
```

onRspOutPosition方法

功能：集中交易持仓调出艾科柜台应答(中信柜台专用，其他券商不支持)

函数原形：

```
void onRspOutPosition (CxeleRspPosTransferField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast)
```

参数：

- pRspField：持仓划拨应答域，其结构体CXeleRspPosTransferField参见onRspInPosition

onRspQryInOutPositionRecord方法

功能：集中交易持仓调拨艾科柜台明细查询应答(中信柜台专用，其他券商不支持)

函数原形：

```
void onRspQryInOutPositionRecord (CxeleRspQryPositionTransferRecordField
*pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast)
```

参数：

- pRspField：明细应答域，其结构体CXeleRspQryPositionTransferRecordField如下：

```
struct CXeleRspQryPositionTransferRecordField {
    ///机构代码
    TXeleOrgIDType          OrgID;
    ///资金账号
    TXeleUserIDType         AccountID;
    ///证券代码
    TXeleSecuritiesIDType   SecuritiesID;
    ///冻结/解冻仓位
    TXeleVolumeType         Stkeffect;
    ///操作方向，'1': 冻结，'2':解冻
    TXeleDirectionType      Direction;
    ///fpga仓位是否更新
    TXeleIsUpdateFpgaFundType IsUpdateFpgaFund;
    ///执行信息
    TXeleCentralTradingErrorMsgType postTransMsg;
    ///操作时间
    TXeleShortTimeType       Time;
    ///操作流水号
    TXeleSnoType             Sno;
    ///预留
    char                    Reserved[59];
};
```

onRspInOutPosition方法

功能：集中交易持仓调入/调出艾科柜台应答(部分券商使用)

函数原形：

```
void onRspInOutPosition (CXeleRspInOutPositionField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast)
```

参数：

- pRspField：持仓划拨应答域，其结构体CXeleRspInOutPositionField如下：

```
struct CXeleRspInOutPositionField{
    ///机构代码
    TXeleOrgIDType          OrgID;
    ///资金账号
    TXeleUserIDType         AccountID;
    ///证券代码
    TXeleSecuritiesIDType   SecuritiesID;
    ///股东账户
    TXeleInvestorIDType      InvestorID;
    ///fpga仓位是否更新,暂不使用
    TXeleIsUpdateFpgaFundType IsUpdateFpgaStock;
    ///操作流水号
    TXeleSnoType             Sno;
    ///划拨数量
    TXeleVolumeType         Volume;
};
```

```

    ///划拨方向 : '1' 划入 , '2' 划出
    TXeleInOutDirectionType          InOutDirection;
    ///集中交易柜台响应错误码,若没有错误码时, 返回-1, 代表无效值
    TXeleCentralTradingErrorIdType    ctErrorId;
    ///集中交易柜台响应错误信息
    TXeleCentralTradingErrorMsgLongType ctErrorMsg;
    ///预留
    char                               Reserved[44];
};

```

响应头中的pRspInfo. ErrorID为55055时，响应体中的集中柜台错误信息ctErrorId、ctErrorMsg才有意义。

onRspQryInOutPositionDetails方法

功能：集中交易持仓调拨艾科柜台明细查询应答(部分券商使用)

函数原形：

```

void onRspQryInOutPositionDetails (CXeleRspQryInOutPositionDetailsField
*pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast)

```

参数：

- pRspField：明细应答域，其结构体CXeleRspQryInOutPositionDetailsField如下：

```

struct CXeleRspQryInOutPositionDetailsField {
    ///机构代码
    TXeleOrgIDType          OrgID;
    ///资金账号
    TXeleUserIDType         AccountID;
    ///证券代码
    TXeleSecuritiesIDType   SecuritiesID;
    ///仓位变动数量
    TXeleVolumeType         Stkeffect;
    ///划拨方向 : '1' 划入 , '2' 划出
    TXeleInOutDirectionType InOutDirection;
    ///fpga仓位是否更新
    TXeleIsUpdateFpgaFundType IsUpdateFpgaStock;
    ///艾科柜台响应错误码
    TXeleErrorIDType         ErrorId;
    ///艾科柜台响应错误信息
    TXeleErrorMsgType        ErrorMsg;
    ///集中交易柜台响应错误码,若没有错误码时, 返回-1, 代表无效值
    TXeleCentralTradingErrorIdType ctErrorId;
    ///集中交易柜台响应错误信息,
    TXeleCentralTradingErrorMsgLongType ctErrorMsg;
    ///操作时间
    TXeleShortTimeType       Time;
    ///操作来源
    TXeleActionSourceType     ActionSource;
    ///操作流水号
    TXeleSnoType              Sno;
}

```

```

    ///预留
    char                                     Reserved[123];
};

```

onRspQryGateWaysRecord方法

功能：用户可用网关信息查询应答

函数原形：

```

void onRspQryGateWaysRecord(CXeleRspQryAvailableGatewayRecordField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：用户可用网关信息查询应答，其结构体CXeleRspQryAvailableGatewayRecordField如下：

```

struct CXeleRspQryAvailableGatewayRecordField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///交易所类型
    TXeleMarketType          Market;
    ///竞价网关,报单时需要转换成整型,如'1'转换成1后填充指定网关字段
    TXeleGateway              BidGateways;
    ///债券网关,报单时需要转换成整型,如'1'转换成1后填充指定网关字段
    TXeleGateway              BondGateways;
    ///预留
    TXeleReserved3Type        Reserved;
};

```

onRspQryBtFund方法

功能：沪深柜台资金账户查询应答

函数原形：

```

void onRspQryBtFund (CXeleRspQryClientAccountFundManagerField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：沪深柜台资金账户查询应答，其结构体CXeleRspQryClientAccountFundManagerField如下：

```

struct CXeleRspQryClientAccountFundManagerField{
    ///资金账号
    TXeleUserIDType          AccountID;
    ///可用余额
    TXeleAvailableFundType    AvailableFund;
    ///可取余额 暂不使用
    TXeleAvailableCashType    AvailableCash;
};

```

```

    ///冻结资金
    TXeleFrozeMarginType      FrozeCapital;
    ///冻结手续费
    TXeleFrozenFeeType        FrozenFee;
    ///已付手续费
    TXeleUsedFeeType          UsedFee;
    ///初始上场资金（不变）
    TXeleTotalFundType        InitTotalFund;
    ///总卖出
    TXeleSellFund             SellFund;
    ///总买入
    TXeleBuyFund              BuyFund;
    ///交易所类型
    TXeleMarketType           Market;
    ///预留
    char                       Reserved[48];
};

```

onRspQryFund方法

功能：证券资金查询应答

函数原形：

```

void onRspQryFund (CXeleRspQryStockClientAccountField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：证券资金查询应答，其结构体CXeleRspQryStockClientAccountField如下：

```

struct CXeleRspQryStockClientAccountField {
    ///资金账户
    TXeleUserIDType      AccountID;
    ///交易账户状态(暂不使用)
    TXeleAcctStatusType  AcctStatus;
    ///冻结资金
    TXeleFrozeMarginType FrozeCapital;
    ///冻结手续费
    TXeleFrozenFeeType    FrozenFee;
    ///已付手续费
    TXeleUsedFeeType      UsedFee;
    ///初始上场资金
    TXeleTotalFundType    InitTotalFund;
    ///上场资金（可变）； 初始上场资金 + 出入金额，可能为负（建议客户不使用）
    TXeleTotalFundType    TotalFund;
    ///总卖出
    TXeleSellFund         SellFund;
    ///总买入
    TXeleBuyFund          BuyFund;
    ///当日盈亏(暂不使用)
    TXeleCurrGainLossType CurrGainLoss;
    ///总盈亏(暂不使用)

```

```

TXeleTotalGainLossType      TotalGainLoss;
///可用资金
TXeleAvailableFundType      AvailableFund;
///交易所类型
TXeleMarketType             Market;
///投资者股东账号
TXeleInvestorIDType          InvestorID;
///客户代码
TXeleUserIDType              CustID;
///被证券公司冻结的资金
TXeleFrozenFundType          FrozenFund;
///可取金额(部分券商使用)
TXeleWithdrawableFundType    WithdrawableFund;
///参考资产
TXeleMoneyType                ReferenceAsset;
///总参考市值
TXeleMoneyType                TotalReferenceMarketCap;
///日初可取金额(部分券商使用)
TXeleMoneyType                OrigwithdrawableFund;
///预留
char                          Reserved[62];
};

```

onRspQryPosition 方法

功能：证券持仓查询应答

函数原形：

```

void onRspQryPosition (CXeleRspQryStockPositionField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField: 证券持仓查询应答，其结构体CXeleRspQryStockPositionField如下：

```

struct CXeleRspQryStockPositionField {
    ///资金账户
    TXeleUserIDType      AccountID;
    ///证券代码
    TXeleSecuritiesIDType SecuritiesID;
    ///证券类别代码
    TXeleSecuritiesType   SecuritiesType;
    ///预留
    char                   Reserved1[2];
    ///证券子类别代码(暂未使用)
    TXeleSecuritiesSubTypeType SecuritiesSubType;
    ///今买仓
    TXeleTdBuyPositionType TdBuyPosition;
    ///今卖仓
    TXeleTdSellPositionType TdSellPosition;
    ///在途买仓
    TXeleUnTdPositionType  UnTdBuyPosition;
}

```

```

///在途卖仓
TXeleUnTdPositionType          UnTdSellPosition;
///在途冻结资金，暂未使用
TXeleUnTdFrozenCapType          UnTdFrozenCap;
///昨持仓（不变），最大数量仅支持2147483647
TXeleYdPositionType             YdPosition;
///买入成本，暂未使用
TXeleYdPositionCostType         YdTotalCost;
///昨持仓剩余
TXeleYdPositionLeftType         YdPositionLeft;
///总持仓 = 老仓 + 今买仓 - 今卖仓
TXeleTotalPositionType          TotalPosition;
///持仓成本
TXeleTotalCostType              TotalCost;
///持仓均价
TXeleAvgPxType                  AvgPrice;
///备兑锁定持仓，部分券商使用
TXeleCoveredFrozenPositionType  CoveredFrozenPosition;
///交易所类型
TXeleMarketType                 Market;
///现有持仓数量（含未卖持仓）=老仓 + 今买仓 (-+)出入仓 - 今卖仓
TXeleRemainingPosition          RemainingPosition;
///可卖持仓数量
TXeleAvailablePosition          AvailablePosition;
///日内可转交易类型,Y表示T+0，N表示T+1
TXeleDayTradingType             DayTrading;
///股东账户
TXeleInvestorIDType             InvestorID;
///客户代码
TXeleUserIDType                 CustID;
///被证券公司冻结的持仓
TXeleFrozenPositionType         FrozenPosition;
///可备兑锁定持仓,允许备兑锁定的持仓数量，部分券商使用
TXeleCoveredFrozenPositionType  AvailableCoverLockPosition;
///可申购数量
TXeleQtyType                     AvailableCreationQty;
///可赎回数量
TXeleQtyType                     AvailableRedemptionQty;
///参考市值
TXeleMoneyType                  ReferenceMarketCap;
///当日申购数量
TXeleQtyType                     TotalCreationQty;
///当日赎回数量
TXeleQtyType                     TotalRedemptionQty;
///预留
char                             Reserved[33];
};

```

onRspQrySecurities 方法

功能：证券信息查询应答

函数原形：

```
void onRspQrySecurities(CXeleRspQryStockSecuritiesField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) ;
```

参数：

- pRspField：证券信息查询应答，其结构体CXeleRspQryStockSecuritiesField如下：

```
struct CXeleRspQryStockSecuritiesField {
    ///证券代码
    TXeleSecuritiesIDType      SecuritiesID;
    ///合约名称
    TXeleSecuritiesNameType    SecuritiesName;
    ///证券类别代码
    TXeleSecuritiesType        SecuritiesType;
    ///证券子类别代码
    TXeleSecuritiesSubType     SecuritiesSubType;
    ///最小变动价位
    TXeleMinTickPriceType      TickPrice;
    ///币种(暂未使用)
    TXeleCurrencyType          Currency;
    ///证券面值
    TXeleSecuritiesParValue    ParValue;
    ///昨结算价
    TXelePreSettlePriceType     PreSettlePrice;
    ///涨跌幅限制类型
    TXeleLimitPriceClassType    LimitPriceClass;
    ///限价单最大买单量
    TXeleMaxLimitOrderVolumeType MaxLimitBuyVolume;
    ///限价单最大卖单量
    TXeleMaxLimitOrderVolumeType MaxLimitSellVolume;
    ///限价单最小买单量
    TXeleMinLimitOrderVolumeType MinLimitBuyVolume;
    ///限价单最小卖单量
    TXeleMinLimitOrderVolumeType MinLimitSellVolume;
    ///市价单最大买单量
    TXeleMaxMarketOrderVolumeType MaxMarketBuyVolume;
    ///市价单最大卖单量
    TXeleMaxMarketOrderVolumeType MaxMarketSellVolume;
    ///市价单最小买单量
    TXeleMinMarketOrderVolumeType MinMarketBuyVolume;
    ///市价单最小卖单量
    TXeleMinMarketOrderVolumeType MinMarketSellVolume;
    ///跌停板价
    TXeleLowerPriceType         LowerPrice;
    ///涨停板价
    TXeleUpperPriceType         UpperPrice;
    ///交易所类型
    TXeleMarketType             Market;
    ///日内可转交易类型,Y表示T+0,N表示T+1
    TXeleDayTradingType         DayTrading;
    ///股票风险级别 , '0' 正常, '1' 风险警示, '2' 退市整理期
    TXeleStockLevel             StockLevel;
    ///最小报单数量单位
    TXeleLotSize                LotSize;
```

```

    ///发行方式
    TXeleIssueType          IssueType;
    ///是否注册制, Y-是-Yes, N-否-No
    TXeleIsRegistration      IsRegistration;
    ///是否非交易, 0-否-No, 1-是-Yes
    TXeleIsNoTrading         IsNoTrading;
    ///昨收盘价
    TXelePriceType           PreClosePrice;
    ///预留
    char                      Reserved[110];
};

```

onRspQryRightsAndInterests方法

功能：权益查询-新股额度查询响应

函数原形：

```

void onRspQryRightsAndInterests (CXeleRspQryStockQuotaField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：证券新股额度查询响应，其结构体CXeleRspQryStockQuotaField为：

```

struct CXeleRspQryStockQuotaField {
    ///资金账号
    TXeleUserIDType          AccountID;
    ///合约类型
    TXeleSecuritiesType      SecuritiesType;
    ///证券代码
    TXeleSecuritiesIDType    SecuritiesID;
    ///新股额度
    TXeleStockQuotaType       StockQuota;
    ///新股已申购额度
    TXeleStockQuotaType       StockHoldQuota;
    ///新股剩余额度
    TXeleStockQuotaType       StockAvlQuota;
    ///交易所类型
    TXeleMarketType          Market;
    ///账户级主板新股总额度
    TXeleStockQuotaType       MainBoardStockQuota;
    ///账户级科创板新股总额度
    TXeleStockQuotaType       TechBoardStockQuota;
    ///预留
    TXeleReservedType         Reserved [111];
};

```

onRspQryInvestorInfo方法

功能：股东账户信息查询应答

函数原形：

```
void onRspQryInvestorInfo CXeleRspQryInvestorInfoField *pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) {};
```

参数：

- pRspField：用户查询自己的股东账号等其他相关信息应答，其结构体CXeleRspQryInvestorInfoField为：

```
///用户查询自己的股东账号等其他相关信息应答
struct CXeleRspQryInvestorInfoField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///市场
    TXeleMarketType          Market;
    ///股东账户
    TXeleInvestorIDType       InvestorID;
    ///股东姓名
    TXeleInvestorNameType     InvestorName;
    ///股东权限，0.表示普通股票交易权限，1.SS:科创板权限 SZ:核准制创业板权限，2为可买入可转债，3.SZ:为可买入注册制创业板，4.为可买入退市整理期股票'
    TXeleInvestorRightsType    InvestorRights;
    ///席位编号
    TXeleInvestorPbuType       InvestorPbu;
    ///客户代码
    TXeleUserIDType           CustID;
    ///专业投资者标志
    TXeleInvestorFlag          InvestorFlag;
    ///预留
    char                      Reserved[112];
};
```

onRspQryPrcProgramInfo方法

功能：程序化风控信息查询应答

函数原形：

```
void onRspQryPrcProgramInfo(CXeleRspQryPrcProgramInfoField *pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) {};
```

参数：

- pRspField：用户查询自己的程序化风控信息应答，其相关结构体如下为：

```
struct CXeleRspQryRateLimitInfoField {
    ///单位时间(毫秒)
```

```

TXeleTimeUnitType          timeUnit;
///单位时间流速阈值
TXelePrcProgramTresholdType RateLimit;
///证券类别
///'1' 股票, '2' 基金, '3' 债券 (若柜台多种类型合并配置, 则以逗号分隔返回, 如'2','3')
TXelePrcSecuritiesType     PrcSecuritiesType;
///预留
char                        Reserved[30];
};

struct CXeleRspQryDailyLimitInfoField {
    ///全天委托笔数阈值
    TXelePrcProgramTresholdType OrderCountLimit;
    ///当前委撤笔数 (程序化风控类别为全天报撤笔数限制风控时有效)
    TXeleCurrentOrderCountType CurrentOrderCount;
    ///证券类别
    ///'1' 股票, '2' 基金, '3' 债券 (若柜台多种类型合并配置, 则以逗号分隔返回, 如'2','3')
    TXelePrcSecuritiesType     PrcSecuritiesType;
    ///预留
    char                        Reserved[30];
};

///查询程序化风控信息应答
struct CXeleRspQryPrcProgramInfoField {
    ///资金账户
    TXeleUserIDType            AccountID;
    ///市场
    TXeleMarketType            Market;
    ///程序化风控类别
    ///1 全天报撤笔数限制风控, 2 流速风控
    TXelePrcProgramType        PrcProgramType;
    ///风控开关
    ///'0' 关, '1' 开
    TXelePrcProgramEnableType  Enable;
    union {
        CXeleRspQryRateLimitInfoField orderRateLimit;
        CXeleRspQryDailyLimitInfoField dailyLimit;
    } prcProgrmData;
    ///预留
    char                        Reserved[60];
};

```

onRspQryExChangeGateWay方法

功能：交易所网关信息查询应答

函数原形：

```

void onRspQryExChangeGateway(CXeleRspQryExChangeGatewayField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：用户查询交易所网关信息应答，其相关结构体如下为：

```

struct CXeleRspQryExChangeGatewayField {
    ///资金账号，必填
    TXeleUserIDType          AccountID;
    ///通道类型，必填
    TXeleChannelType         ChannelType;
    ///网关编号，必填
    TXeleGatewayID          GatewayID;
    ///网关类型，必填
    TXeleGatewayType         GatewayType;
    ///网关状态，必填
    TXeleGatewayStatus       GatewayStatus;
    ///最近一次网关状态的更新时间，必填
    TXeleEnterTime           EnterTime;
    ///市场，必填
    TXeleMarketType          Market;
    ///预留，必填
    char                      Reserved[100];
};

```

onRtnInOutPosition方法

功能：仓位变化回报通知(部分券商使用)

函数原形：

```

void onRtnInOutPosition(CXeleRtnInOutPositionField * pRtnField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast)

```

参数：

- pRtnField：仓位变化回报通知域，其结构体CXeleRtnInOutPositionField如下：

```

struct CXeleRtnInOutPositionField {
    ///资金账号
    TXeleUserIDType          AccountID;
    ///证券代码
    TXeleSecuritiesIDType    SecuritiesID;
    ///市场类型
    TXeleMarketType          Market;
    ///仓位变化数量
    TXeleVolumeType          Volume;
    ///划拨方向：'1' 划入，'2' 划出，'5' 备兑锁定，'6' 备兑解锁
    TXeleInOutDirectionType  InOutDirection;
    ///流水号
    Long                      SerialNum;
    ///操作时间
    ///格式：HHMMSSsssuuunnn,表示柜台接收到请求后处理完请求的时间
    TXeleEnterTime           Time;
    ///划拨模式，1 经过集中柜台 2 不经过集中柜台
    TXeleInOutMode           InOutMode;
    ///预留
    char                      Reserved[64];
};

```

onRspQryETFCreationRedemptionComponentSecurities方法

功能：ETF申赎成分股信息查询

函数原形：

```
void  
onRspQryETFCreationRedemptionComponentSecurities(CXeleRspQryETFCreationRedemptionComponentSecuritiesField * pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast)
```

参数：

- pRtnField：仓位变化回报通知域，其结构体 CXeleRspQryETFCreationRedemptionComponentSecuritiesField 如下：

```
struct CXeleRspQryETFCreationRedemptionComponentSecuritiesField {  
    ///证券代码  
    TXeleSecuritiesIDType      SecuritiesID;  
    ///成份证券代码  
    TXeleSecuritiesIDType      ComponentSecurityId;  
    ///成份证券名称  
    TXeleSecurityName          ComponentSecurityName;  
    ///成份证券来源  
    TXeleSecuritySourceType     ComponentSecuritySource;  
    ///现金替代标记  
    TXeleSubstituteFlag        SubstituteFlag;  
    ///成份证券数  
    TXeleQuantityType          ComponentShare;  
    ///溢价比例  
    TXelePriceType              PremiumRatio;  
    ///折价比例  
    TXelePriceType              DiscountRatio;  
    ///申购替代金额  
    TXelePriceType              CreationCashSubstitute;  
    ///赎回替代金额  
    TXelePriceType              RedemptionCashSubstitute;  
    ///预留  
    char                        Reserved[160];  
};
```

onRspQryETFCreationRedemptionSecurities方法

功能：ETF申赎母基金信息查询

函数原形：

```
void  
onRspQryETFCreationRedemptionSecurities(CXeleRspQryETFCreationRedemptionSecuritiesField * pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast)
```

参数：

- pRtnField：仓位变化回报通知域，其结构体 CXeleRspQryETFCreationRedemptionSecuritiesField 如下：

```

struct CXeleRspQryETFCreationRedemptionSecuritiesField {
    ///证券代码
    TXeleSecuritiesIDType      SecuritiesID;
    ///合约名称
    TXeleSecurityName          SecurityName;
    ///成份证券来源
    TXeleSecuritySourceType     SecuritySource;
    ///最小申赎单位
    TXeleQuantityType          CreationRedemptionUnit;
    ///预估现金差额
    TXelePriceType              EstimateCashComponent;
    ///最大现金替代比例
    TXelePriceType              MaxCashRatio;
    ///申购权限
    TXeleRightFlag              CreationRight;
    ///赎回权限
    TXeleRightFlag              RedemptionRight;
    ///本市场成份证券数
    TXeleRecordNumType          RecordNum;
    ///所有成份证券数
    TXeleTotalRecordNum         TotalRecordNum;
    ///现金余额
    TXelePriceType              CashComponent;
    ///申赎基准单位净值
    TXelePriceType              NAVperCU;
    ///单位净值
    TXelePriceType              NAV;
    ///红利金额
    TXelePriceType              DividendPerCU;
    ///账户ETF申购控制
    TXeleRightFlag              CustomerCreationDisallows;
    ///账户ETF赎回控制
    TXeleRightFlag              CustomerRedemptionDisallows;
    ///预留
    char                         Reserved[149];
};

```

onRtnExChangeGateway方法

功能：交易所网关信息通知，在交易所网关状态发生变化时，自动向api端推送当前状态

函数原形：

```

void onRtnExChangeGateway (CXeleRtnExChangeGatewayField* pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast)

```

参数：

- pRspField：交易所网关信息通知域，其结构体CXeleRtnExChangeGatewayField如下：

```

struct CXeleRtnExChangeGatewayField {

```

```

    ///资金账号，必填
    TXeleUserIDType          AccountID;
    ///通道类型，必填
    TXeleChannelType         ChannelType;
    ///网关编号，必填
    TXeleGatewayID           GatewayID;
    ///网关类型，必填
    TXeleGatewayType         GatewayType;
    ///网关状态，必填
    TXeleGatewayStatus       GatewayStatus;
    ///最近一次网关状态的更新时间，必填
    TXeleEnterTime           EnterTime;
    ///市场，必填
    TXeleMarketType          Market;
    ///预留，必填
    char                      Reserved[100];
};

```

onRspQryQFIIOrder 方法

功能：QFII报单查询应答

函数原形：

```

void onRspQryQFIIOrder(CXeleRspQryQFIIOrderField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：QFII报单查询应答，其结构体CXeleRspQryQFIIOrderField如下：

```

struct CXeleRspQryQFIIOrderField {
    ///QFII用户报单编号
    TXeleQFIIOrderIDType      UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType     SecuritiesID;
    ///买卖方向
    TXeleDirectionType        Direction;
    ///开平标志
    TXeleOffsetFlagType       CmbOffsetFlag;
    ///价格
    TXelePriceType            LimitPrice;
    ///数量
    TXeleVolumeType           Volume;
    ///报单类型
    TXeleOrderTypeType        OrderType;
    ///有效期类型
    TXeleTimeConditionType    TimeCondition;
    ///交易所类型
    TXeleMarketType          Market;
    ///委托方式
    TXeleOperwayType          Operway;
    ///柜台报单编号int类型

```

TXeleOrderIDType	OrderSysID;
///交易所报单编号	
TXeleOrderExchangeIDType	OrderExchangeID;
///接受请求时间	
TXeleTimeType	TransactTime;
///对应申报市价转限价的订单，这里填写转为限价订单的价格，单位：元	
TXelePriceType	DiscretionPrice;
///累计成交数量	
TXeleVolumeType	TradeVolume;
///未成交手数	
///如果状态是撤单或者部撤，leavesVolume是已经成功撤单的数量;	
///如果报单状态是部分成交，leavesVolume 表示未成交数量 = 报单数量 - 累计成交数量)	
TXeleVolumeType	LeavesVolume;
///订单状态	
TXeleOrderStatusType	OrderStatus;
///佣金	
TXeleSecuritiesCommissionType	Commission;
///印花税	
TXeleStampTaxRateType	StampTax;
///过户费	
TXeleTransferFeeRateType	Transfer;
///流量费	
TXeleTrafficFeeType	TrafficFee;
///总手续费	
TXeleTotalFeeType	TotalFee;
///保证金(暂未使用)	
TXeleMoneyType	Margin;
///冻结权利金(暂未使用)	
TXeleMoneyType	FrozenPremium;
///投资者账号	
TXeleUserIDType	AccountID;
///客户端登录子节点	
TXeleSubClientIndexType	SubClientIndex;
///业务单元(用户定义)	
TXeleBusinessUnitType	BusinessUnit;
///柜台报单编号str类型	
TXeleStrOrderSysIDType	StrOrderSysID
///报单类型	
TXeleMessageIDType	OrderMessageId;
///证券类别代码	
TXeleSecuritiesType	SecuritiesType;
///预埋单标记 '0':非预埋单 '1':预埋单	
TXelePreOrderFlag	PreOrderFlag;
///报单错误编号	
TXeleErrorIDType	ErrorId;
///分页查询结束值 用来作为下次查询的起始值，来实现分页	
TXeleVolumeType	EndNum;
///分页查询条件下，是否可以再次进行查询 只在最后一条回报中有效	
TXeleIsLastType	QryAgain;
///当前条件下查询到的回报总数	
TXeleVolumeType	TotalNum;
///投资者股东账号	
TXeleInvestorIDType	InvestorID;
///客户代码	
TXeleUserIDType	CustID;

```

    ///成交金额
    TxelePriceType                TradeAmount;
    ///主柜台未同步到备机的委托类型
    TxeleUnSyncOrderFlag          UnSyncOrderFlag;
    ///备兑标志(期权使用)
    TxeleCoveredFlagType          CoveredOrUncovered;
    ///被撤单柜台报单编号str类型(带号段)
    TxeleStrOrderSysIDType        OrigStrOrderSysID;
    ///报单来源,支持接入网关的柜台,此字段有效
    TxeleOrderSourceType          OrderSource;
    ///交易前置id(QFII中暂不使用)
    ///返回值为实际报单的网关id + 1
    ///例如: 返回值为1代表是0号网关,2代表1号网关
    TxeleExchangeIDIntType        ExchangeFrontID;
    ///预留
    char                           Reserved[64];
};

```

onRspQryQFIITrade 方法

功能：QFII成交单查询应答

函数原形：

```

void onRspQryQFIITrade(CXeleRspQryQFIITradeField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：QFII成交单查询应答，其结构体CXeleRspQryQFIITradeField如下：

```

struct CXeleRspQryQFIITradeField {
    ///QFII用户本地报单编号
    TxeleQFIIOrderIDType        UserLocalID;
    ///交易日
    TxeleDateType                TradingDay;
    ///成交数量
    TxeleVolumeType              TradeVolume;
    ///未成交手数(报单数量-累计成交数量)
    TxeleVolumeType              LeavesVolume;
    ///证券代码
    TxeleSecuritiesIDType        SecuritiesID;
    ///申报时间
    TxeleShortTimeType           OrderTime;
    ///成交时间
    TxeleShortTimeType           TradeTime;
    ///成交价格
    TxelePriceType                TradePrice;
    ///成交金额
    TxelePriceType                TradeAmount;
    ///暂未使用
    TxeleOrderIDType             TradeID;
    ///交易所类型

```

TXeleMarketType	Market;
///备兑标志(期权使用)	
TXeleCoveredFlagType	CoveredOrUncovered;
///柜台报单编号int类型	
TXeleOrderIDType	OrderSysID;
///交易所报单编号	
TXeleOrderExchangeIDType	OrderExchangeID;
///买卖方向	
TXeleDirectionType	Direction;
///开平标志, 股票不填	
TXeleOffsetFlagType	CmbOffsetFlag;
///佣金	
TXeleSecuritiesCommissionType	Commission;
///印花税	
TXeleStampTaxRateType	StampTax;
///过户费	
TXeleTransferFeeRateType	Transfer;
///总手续费	
TXeleTotalFeeType	TotalFee;
///保证金(暂未使用)	
TXeleMoneyType	Margin;
///权利金(暂未使用)	
TXeleMoneyType	Premium;
///投资者账号	
TXeleUserIDType	AccountID;
///订单状态	
TXeleOrderStatusType	OrderStatus;
///柜台报单编号str类型	
TXeleStrOrderSysIDType	StrOrderSysID;
/// 交易所执行编号	
TXeleOrderExchangeIDType	ExecID;
///客户端登录子节点	
TXeleSubClientIndexType	SubClientIndex;
///分页查询结束值 用来作为下次查询的起始值, 来实现分页	
TXeleVolumeType	EndNum;
///分页查询条件下, 是否可以再次进行查询 只在最后一条回报中有效	
TXeleIsLastType	QryAgain;
///当前条件下查询到的回报总数	
TXeleVolumeType	TotalNum;
///投资者股东账号	
TXeleInvestorIDType	InvestorID;
///客户代码	
TXeleUserIDType	CustID;
///报单来源, 支持接入网关的柜台, 此字段有效	
TXeleOrderSourceType	OrderSource;
///报单类型	
TXeleOrderTypeType	OrderType;
/// 本市场成份证券数	
TXeleRecordNumType	RecordNum;
/// 成份证券代码	
TXeleSecuritiesIDType	ComponentSecurityID;
/// 成份证券来源	
TXeleSecuritySourceType	ComponentSecuritySource;
/// 使用老仓数量	
TXeleQuantityType	UseOldQty;

```

    /// 成份股份成交数量
    TXeleQuantityType                ComponentTradeQty;
    /// 现金替代金额
    TXelePriceType                    ContAmtValue;
    ///ETF成交记录查询标志, QFII柜台使用
    TXeleQryEtfTradeFlag              QryEtfTradeFlag;
    ///预留
    char                              Reserved[64];
};

```

- blsLast：此次请求的响应的最后一次回调标志（若使用报单编号查询只会返回一条信息，可不判断blsLast；若使用时间范围查询，回报就是一批信息，需使用blsLast来判断结束）

onRspQryQFIIPosition方法

功能：QFII证券持仓查询应答

函数原形：

```

void onRspQryQFIIPosition (CXeleRspQryQFIIStockPositionField
*pRspField, CXeleRspInfo *pRspInfo, int nRequestID, bool blsLast);

```

参数：

- pRspField: QFII证券持仓查询应答，其结构体CXeleRspQryQFIIStockPositionField如下：

```

struct CXeleRspQryQFIIStockPositionField {
    ///资金账户
    TXeleUserIDType                AccountID;
    ///证券代码
    TXeleSecuritiesIDType          SecuritiesID;
    ///证券类别代码
    TXeleSecuritiesType            SecuritiesType;
    ///今买仓
    TXeleQFIIPositionType          TdBuyPosition;
    ///今卖仓
    TXeleTdSellPositionType        TdSellPosition;
    ///在途买仓
    TXeleQFIIPositionType          UntdBuyPosition;
    ///在途卖仓
    TXeleQFIIPositionType          UntdSellPosition;
    ///昨持仓（不变）
    TXeleQFIIPositionType          YdPosition;
    ///昨持仓剩余
    TXeleQFIIPositionType          YdPositionLeft;
    ///持仓成本
    TXeleTotalCostType             TotalCost;
    ///交易所类型
    TXeleMarketType                Market;
    ///日内可转交易类型, Y表示T+0, N表示T+1
    TXeleDayTradingType            DayTrading;
    ///股东账户
    TXeleInvestorIDType            InvestorID;
};

```

```

    ///客户代码
    TXeleUserIDType          CustID;
    ///当日申购持仓数量
    TXeleQFIIPositionType    TdCreationQty;
    ///当日赎回成分股数量
    TXeleQFIIPositionType    TdRedemptionQty;
    ///在途申购数量
    TXeleQFIIPositionType    UnTdCreationQty;
    ///出入仓数量
    TXeleQFIIPositionType    InOutPosition;
    ///在途赎回数量
    TXeleQFIIPositionType    UnTdRedemptionQty;
    ///在途卖来自赎回成分股数量
    TXeleQFIIPositionType    UnTdSellFromRedemptionQty;
    ///在途卖来自申购数量
    TXeleQFIIPositionType    UnTdSellFromCreationQty;
    ///预留
    char                      Reserved[141];
};

```

- blsLast：此次请求的响应的最后一次回调标志（若使用报单编号查询只会返回一条信息，可不判断blsLast；若使用时间范围查询，回报就是一批信息，需使用blsLast来判断结束）

onRspTenderOfferOrder方法

功能：要约录入应答

函数原形：

```

void onRspTenderOfferOrder(CXeleRspTenderOfferOrderField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool blsLast);

```

参数：

- pRspField: 要约录入应答域，其结构体CXeleRspTenderOfferOrderField如下：

```

///要约收购应答
struct CXeleRspTenderOfferOrderField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrdersSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType     SecuritiesID;
    ///买卖方向
    TXeleDirectionType        Direction;
    ///报单数量
    TXeleVolumeType           Volume;
    ///交易所类型
    TXeleMarketType           Market;
    ///预埋单标记 '0':非预埋单 '1':预埋单
    TXelePreOrderFlag         PreOrderFlag;
    ///错误编号，拒单使用

```

```

TXeleErrorIdType           ErrorId;
///客户端登录子节点
TXeleSubClientIndexType    SubClientIndex;
///业务单元(用户定义)
TXeleBusinessUnitType      BusinessUnit;
///柜台报单编号str类型
TXeleStrOrderSysIDType     StrOrderSysID;
///投资者账号
TXeleUserIDType            AccountID;
///交易所报单编号
TXeleOrderExchangeIDType   OrderExchangeID;
///交易前置id(暂不使用)
TXeleExchangeIDType        ExchangeFrontID;
///要约收购人编码
TXeleTendererIDType        TendererID;
///预留
char                        Reserved[14];
};

```

onRspWarrantExerciseOrder方法

功能：权证录入应答

函数原形：

```

void onRspWarrantExerciseOrder(CXeleRspWarrantExerciseOrderField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField: 权证录入应答域，其结构体CXeleRspWarrantExerciseOrderField如下：

```

///权证行权应答
struct CXeleRspWarrantExerciseOrderField {
    ///柜台报单编号int类型
    TXeleOrderIDType           OrdersSysID;
    ///用户本地报单编号
    TXeleOrderIDType           UserLocalID;
    ///证券代码
    TXeleSecuritiesIDType      SecuritiesID;
    ///买卖方向
    TXeleDirectionType         Direction;
    ///限价价格
    TXelePriceType             LimitPrice;
    ///报单数量
    TXeleVolumeType            Volume;
    ///报单类型：固定为'2'-限价
    TXeleOrderTypeType         OrderType;
    ///交易所类型
    TXeleMarketType            Market;
    ///预埋单标记 '0':非预埋单 '1':预埋单
    TXelePreOrderFlag           PreOrderFlag;
    ///错误编号，拒单使用
    TXeleErrorIdType           ErrorId;
};

```

```

    ///客户端登录子节点
    TXeleSubClientIndexType      SubClientIndex;
    ///业务单元(用户定义)
    TXeleBusinessUnitType        BusinessUnit;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType       StrOrderSysID;
    ///投资者账号
    TXeleUserIDType              AccountID;
    ///交易所报单编号
    TXeleOrderExchangeIDType     OrderExchangeID;
    ///交易前置id(暂不使用)
    TXeleExchangeIDType          ExchangeFrontID;
    ///预留
    char                          Reserved[11];
};

```

期权相关

API类接口

reqInsertCombOrder 方法

功能：期权组合报单请求，组合策略构成若有权利仓，其必须在前

函数原形：

```

int reqInsertCombOrder (CXeleReqOptionCombOrderField &inputField, int
nRequestID)

```

参数：

- inputField：期权组合报单请求，其结构体CXeleReqOptionCombOrderField如下:

```

struct CXeleReqOptionCombOrderField {
    ///用户本地报单编号
    TXeleOrderIDType      UserLocalID;
    ///订单所有类型
    TXeleTradeOwnerType    OwnerType;
    ///申报数量
    TXeleShortVolumeType   CombVolume;
    ///申报类型，'1':组合，'2':拆分，行权不填
    TXeleCombOrderType     CmbOrderType;
    ///组合策略类型，行权不填
    TXeleStrategyCombType   StgyCmbType;
    ///组合策略流水号 拆分时填写
    TXeleSecondaryOrderType SecondaryOrderID;
    ///成分合约个数,当前仅支持2腿,固定填2
    TXeleSumLegsType        SumLegs;
    ///成分合约扩展结构体，深交拆分可不填写
    CXeleCombLegField       CombLeg[4];
    ///业务单元，供客户自身业务使用
    TXeleBusinessUnitType   BusinessUnit;
}

```

```

    ///委托方式
    TXeleOperwayType          Operway;
    ///交易前置id,默认或者未填写为0网关轮询,查询接口返回的网关需要转换成整型,如'1'转换成1后填充该字段
    TXeleExchangeIDType       ExchangeFrontID;
    ///预留
    char                       Reserved[28];
};

```

其中期权成分扩展结构体CXeleCombLegField如下：

```

struct CXeleCombLegField {
    ///成份合约代码
    TXeleSecuritiesIDType     LegSecuritiesID;
    ///成份合约方向,行权不填
    TXeleLegSideType          LegSide;
    ///预留
    char                       Reserved;
    ///成分合约数量
    TXeleShortVolumeType       Volume;
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用CXeleReqOptionCombOrderField进行期权组合报单请求，可以在onRsplInsertCombOrder回报中查看相关信息。

reqInsertExercise方法

功能：期权单腿行权报单请求

函数原形：

```

int reqInsertExercise (CXeleReqOptionExerciseAndLockField &inputField, int
nRequestID)

```

参数：

- inputField：期权单腿行权报单请求域，其结构体CXeleReqOptionExerciseAndLockField为：

```

struct CXeleReqOptionExerciseAndLockField {
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///合约编码(证券锁定/证券解锁时表示标的代码)
    TXeleSecuritiesIDType     SecuritiesID;
    ///请求数量
    TXeleVolumeType           Volume;
    ///锁定/解锁,行权不填
    TXeleDirectionType        Side;
    ///订单所有类型
    TXeleTradeOwnerType        OwnerType;
};

```

```

    ///业务单元(用户定义)
    TXeleBusinessUnitType          BusinessUnit;
    ///委托方式
    TXeleOperwayType               Operway;
    ///交易前置id,默认或者未填写为0网关轮询,查询接口返回的网关需要转换成整型,如'1'转换成1后填充该字段
    TXeleExchangeIDType            ExchangeFrontID;
    ///预留
    char                            Reserved[20];
};

```

- nRequestID : 请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用CXeleReqOptionExerciseAndLockField进行期权单腿行权报单请求，可以在onRsplInsertExercise回报中查看相关信息。

reqCancelExercise 方法

功能：期权单腿行权撤单请求

函数原形：

```
int reqCancelExercise (CXeleReqOrderActionField &inputField, int nRequestID)
```

参数：

- inputField：期权单腿行权报单请求域，其结构体CXeleReqOrderActionField为：

```

struct CXeleReqOrderActionField {
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///撤单报单编号
    TXeleOrigSysIDType        OrigSysID;
    ///预留
    char                       Reserved1[24];
    ///业务单元, (用户定义)
    TXeleBusinessUnitType     BusinessUnit;
    ///订单所有类型
    TXeleTradeOwnerType       OwnerType;
    ///委托方式
    TXeleOperwayType          Operway;
    ///交易前置id(暂未使用)
    TXeleExchangeIDType       ExchangeFrontID;
    ///预留
    char                       Reserved2[3];
    ///错误编号, 拒单使用
    TXeleErrorIDType          ErrorID;
};

```

- nRequestID : 请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用ReqExerciseCancel进行期权单腿行权撤单请求，可以在onRspExerciseCancel回报中查看相关信息。

reqInsertExerciseComb 方法

功能：组合行权请求

函数原形：

```
int reqInsertExerciseComb (CXeleReqOptionCombOrderField &inputField, int nRequestID)
```

参数：

- inputField：组合行权 请求域，其结构体CXeleReqOptionCombOrderField如下：

```
struct CXeleReqOptionCombOrderField {  
    ///用户本地报单编号  
    TXeleOrderIDType          UserLocalID;  
    ///订单所有类型  
    TXeleTradeOwnerType       OwnerType;  
    ///申报数量  
    TXeleShortVolumeType      CombVolume;  
    ///申报类型，'1':组合，'2':拆分，行权不填  
    TXeleCombOrderType        CmbOrderType;  
    ///组合策略类型，行权不填  
    TXeleStrategyCombType     StgyCmbType;  
    ///组合策略流水号 拆分时填写  
    TXeleSecondaryOrderType    SecondaryOrderID;  
    ///成分合约个数,当前仅支持2腿,固定填2  
    TXeleSumLegsType           SumLegs;  
    ///成分合约扩展结构体，深交拆分可不填写  
    CXeleCombLegField          CombLeg[4];  
    ///业务单元，供客户自身业务使用  
    TXeleBusinessUnitType      BusinessUnit;  
    ///委托方式  
    TXeleOperwayType           Operway;  
    ///交易前置id,默认或者未填写为0网关轮询,查询接口返回的网关需要转换成整型，如'1'转换成1后填充该字段  
    TXeleExchangeIDType        ExchangeFrontID;  
    ///预留  
    char                       Reserved[28];  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqInsertExerciseComb进行行权组合录入请求，可以在onRspInsertExerciseComb回报中查看相关信息。

reqCancelExerciseComb 方法

功能：组合行权撤单请求

函数原形：

```
int reqCancelExerciseComb (CXeleReqOrderActionField &inputField, int nRequestID)
```

参数：

- inputField：组合行权请求域，其结构体CXeleReqOrderActionField如下：

```
struct CXeleReqOrderActionField {  
    ///用户本地报单编号  
    TXeleOrderIDType          UserLocalID;  
    ///撤单报单编号  
    TXeleOrigSysIDType        OrigSysID;  
    ///预留  
    char                      Reserved1[24];  
    ///业务单元，(用户定义)  
    TXeleBusinessUnitType     BusinessUnit;  
    ///订单所有类型  
    TXeleTradeOwnerType       OwnerType;  
    ///委托方式  
    TXeleOperwayType          Operway;  
    ///交易前置id(暂未使用)  
    TXeleExchangeIDType       ExchangeFrontID;  
    ///预留  
    char                      Reserved2[3];  
    ///错误编号，拒单使用  
    TXeleErrorIDType          ErrorID;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqCancelExerciseComb进行组合行权撤单请求，可以在onRspCancelExerciseComb回报中查看相关信息。

reqQryOptionPosition方法

功能：期权持仓查询请求

函数原形：

```
int reqQryOptionPosition (CXeleReqQryPositionField &inputField, int nRequestID);
```

参数：

- inputField：持仓查询请求域（SecuritiesID查询，该字段不填查所有），其结构体CXeleReqQryPositionField如下：

```

struct CXeleReqQryPositionField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///证券代码
    TXeleSecuritiesIDType    SecuritiesID;
    ///交易所类型,不填查询登录的市场,填写查询对应的市场, 'a' 查询全部,期权无效
    TXeleMarketType          Market;
    ///预留
    char                      Reserved[127];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryOptionPosition进行期权持仓查询请求，可以在onRspQryOptionPosition期权持仓查询回报中查看相关信息。

reqQryOptionFund 方法

功能：期权资金查询请求

函数原形：

```

int reqQryOptionFund (CXeleReqQryClientAccountField &inputField, int
nRequestID);

```

参数：

- inputField：客户资金查询请求域，其结构体CXeleReqQryClientAccountField如下：

```

struct CXeleReqQryClientAccountField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///预留
    TXeleReservedType        Reserved;
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryOptionFund进行期权资金查询请求，可以在onRspQryOptionFund回报中查看相关信息。

reqQryOptionSecurities方法

功能：期权合约查询请求

函数原形：

```
int reqQryOptionSecurities (CXeleReqQrySecuritiesField &inputField, int  
nRequestID)
```

参数：

- inputField：合约查询请求域，其结构体CXeleReqQrySecuritiesField如下：

```
struct CXeleReqQrySecuritiesField {  
    ///证券代码  
    TXeleSecuritiesIDType    SecuritiesID;  
    ///预留  
    TXeleReservedType        Reserved;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryOptionSecurities进行期权合约查询请求，可以在onRspQryOptionSecurities中查看相关信息。

reqQryOptionRate方法

功能：期权佣金费率、保证金率查询请求

函数原形：

```
int reqQryOptionRate (CXeleReqQryOptionMarginFeeField &inputField, int  
nRequestID);
```

参数：

- inputField：期权佣金费率、保证金率查询请求，其结构体CXeleReqQryOptionMarginFeeField为：

```
struct CXeleReqQryOptionMarginFeeField {  
    ///资金账户  
    TXeleUserIDType        AccountID;  
    ///证券代码  
    TXeleSecuritiesIDType    SecuritiesID;  
    ///营业部代码(暂未使用)  
    TXeleDepartmentIDType    DepartID;  
    ///预留  
    TXeleReservedType        Reserved;  
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryOptionRate进行 期权佣金费率、保证金率查询请求，可以在onRspQryOptionRate回报中查看相关信息。

reqQryOptionCombPosition 方法

功能：期权组合持仓查询请求

函数原形：

```
int reqQryOptionCombPosition (CXeleReqQryOptionCombPositionField &inputField,
int nRequestID)
```

参数：

- inputField：期权组合持仓查询请求域，其结构体CXeleReqQryOptionCombPositionField为：

```
struct CXeleReqQryOptionCombPositionField {
    ///资金账号
    TXeleUserIDType          AccountID;
    ///证券代码
    TXeleSecuritiesIDType    SecuritiesID;
    ///预留
    char                      Reserved[120];
};
```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

说明：当用户登录柜台系统后，用户使用reqQryOptionCombPosition进行期权组合持仓查询请求，可以在onRspQryOptionCombPosition回报中查看相关信息。

reqOTU方法

功能：证券锁定与解锁请求

函数原形：

```
void reqOTU(CXeleReqOptionExerciseAndLockField *pRspField, int nRequestID);
```

参数：

- pRspField：证券锁定与解锁请求，其结构体CXeleReqOptionExerciseAndLockField为：

```
struct CXeleReqOptionExerciseAndLockField {
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///合约编码(证券锁定/证券解锁时表示标的代码)
    TXeleSecuritiesIDType    SecuritiesID;
    ///请求数量
    TXeleVolumeType           Volume;
    ///锁定/解锁,行权不填
    TXeleDirectionType        Side;
```

```

    ///订单所有类型
    TXeleTradeOwnerType          OwnerType;
    ///业务单元(用户定义)
    TXeleBusinessUnitType        BusinessUnit;
    ///委托方式
    TXeleOperwayType             Operway;
    ///交易前置id,默认或者未填写为0网关轮询,查询接口返回的网关需要转换成整型,如'1'转换成1后填充该字段
    TXeleExchangeIDType          ExchangeFrontID;
    ///预留
    char                          Reserved[20];
};

```

- nRequestID：请求ID

返回值：0表示成功，其他值异常；

SPI类接口

onRspInsertCombOrder 方法

功能：期权组合报单应答

函数原形：

```

void onRspInsertCombOrder (CXeleRspOptionCombOrderField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：期权组合报单应答，其结构体CXeleRspOptionCombOrderField如下：

```

struct CXeleRspOptionCombOrderField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///订单所有类型
    TXeleTradeOwnerType        OwnerType;
    ///申报数量
    TXeleShortVolumeType       CombVolume;
    ///申报类型, '1':组合, '2':拆分, 行权无效
    TXeleCombOrderType         CmbOrderType;
    ///组合策略类型, 行权无效
    TXeleStrategyCombType      StgyCmbType;
    ///组合策略流水号 拆分时有效
    TXeleSecondaryOrderType     SecondaryOrderID;
    ///成分合约个数
    TXeleSumLegsType           SumLegs;
    ///成分合约扩展结构体
    CXeleCombLegField          CombLeg[4];
    ///客户端登录子节点
    TXeleSubClientIndexType     SubClientIndex;
    ///业务单元, 供客户自身业务使用
}

```

```

TXeleBusinessUnitType      BusinessUnit;
///柜台报单编号str类型
TXeleStrOrderSysIDType     StrOrderSysID;
///交易所类型
TXeleMarketType           Market;
///投资者账号
TXeleUserIDType           AccountID;
///预留
char                       Reserved[32];
};

```

其中期权成分扩展结构体CXeleCombLegField如下：

```

struct CXeleCombLegField {
    ///成份合约代码
    TXeleSecuritiesIDType    LegSecuritiesID;
    ///成份合约方向,行权不填
    TXeleLegSideType         LegSide;
    ///预留
    char                     Reserved;
    ///成分合约数量
    TXeleShortVolumeType     Volume;
};

```

onErrRtnInsertCombOrder方法

功能：期权组合报单录入错误回报

函数原形：

```

void onErrRtnInsertCombOrder (CXeleRspOptionCombOrderField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：组合报单错误回报，其回报结构体CXeleRspOptionCombOrderField如下：

```

struct CXeleRspOptionCombOrderField {
    ///柜台报单编号int类型
    TXeleOrderIDType         OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType         UserLocalID;
    ///订单所有类型
    TXeleTradeOwnerType      OwnerType;
    ///申报数量
    TXeleShortVolumeType     CombVolume;
    ///申报类型, '1':组合, '2':拆分, 行权无效
    TXeleCombOrderType       CmbOrderType;
    ///组合策略类型, 行权无效
    TXeleStrategyCombType    StgyCmbType;
    ///组合策略流水号 拆分时有效
    TXeleSecondaryOrderType  SecondaryOrderID;
};

```

```

    ///成分合约个数
    TXeleSumLegsType                SumLegs;
    ///成分合约扩展结构体
    CXeleCombLegField              CombLeg[4];
    ///客户端登录子节点
    TXeleSubClientIndexType        SubClientIndex;
    ///业务单元，供客户自身业务使用
    TXeleBusinessUnitType          BusinessUnit;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType         StrOrderSysID;
    ///交易所类型
    TXeleMarketType                Market;
    ///投资者账号
    TXeleUserIDType                AccountID;
    ///预留
    char                            Reserved[32];
};

```

其中期权成分扩展结构体CXeleCombLegField如下：

```

struct CXeleCombLegField {
    ///成份合约代码
    TXeleSecuritiesIDType          LegSecuritiesID;
    ///成份合约方向,行权不填
    TXeleLegSideType               LegSide;
    ///预留
    char                            Reserved;
    ///成分合约数量
    TXeleShortVolumeType           Volume;
};

```

onRtnCombOrder方法

功能：期权组合报单回报

函数原形：

```

void onRtnCombOrder (CXeleRtnOptionCombOrderField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：期权组合报单回报，其结构体CXeleRtnOptionCombOrderField为：

```

struct CXeleRtnOptionCombOrderField {
    ///柜台报单编号int类型
    TXeleOrderIDType              OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType              UserLocalID;
    ///订单所有类型
    TXeleTradeOwnerType           OwnerType;
    ///申报数量

```

```

TxleShortVolumeType      CombVolume;
///申报类型, '1':组合, '2':拆分, 行权无效
TxleCombOrderType        CmbOrderType;
///组合策略类型, 行权无效
TxleStrategyCombType      StgyCmbType;
///组合策略流水号 拆分时有效
TxleSecondaryOrderType    SecondaryOrderID;
///成分合约个数
TxleSumLegsType           SumLegs;
///成分合约扩展结构体
CxeleCombLegField         CombLeg[4];
///接受请求时间
TxleTimeType              TransactTime;
///订单状态
TxleOrderStatusType       OrderStatus;
///客户端登录子节点
TxleSubClientIndexType    SubClientIndex;
///业务单元, 供客户自身业务使用
TxleBusinessUnitType      BusinessUnit;
///柜台报单编号str类型
TxleStrOrderSysIDType     StrOrderSysID;
///交易所类型
TxleMarketType            Market;
///交易前置id
TxleExchangeIDType        ExchangeFrontID;
/// 交易所报单编号
TxleOrderExchangeIDType   OrderExchangeID;
///投资者账号
TxleUserIDType            AccountID;
///预留
char                       Reserved1;
///被撤单用户本地报单编号
TxleOrderIDType           OrigUserLocalID;
///被撤单柜台报单编号int类型
TxleOrigSysIDType         OrigOrderSysID;
///被撤单柜台报单编号str类型(带号段)
TxleStrOrderSysIDType     OrigStrOrderSysID;
///预留
char                       Reserved[38];
};

```

其中期权多腿组合信息结构体CxeleCombOrderField如下：

```

struct CxeleCombOrderField {
    ///成份合约代码
    TxleSecuritiesIDType    LegSecuritiesID;
    ///成份合约方向, 行权不填
    TxleLegSideType         LegSide;
    ///预留
    char                     Reserved;
    ///成分合约数量
    TxleShortVolumeType     Volume;
};

```

onRtnCombTrade方法

功能：期权组合成交回报

函数原形：

```
void onRtnCombTrade(CXeleRtnOptionCombTradeField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast);
```

参数：

- pRspField：期权组合成交回报，其结构体CXeleRtnOptionCombTradeField如下：

```
struct CXeleRtnOptionCombTradeField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrdersSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///订单所有类型
    TXeleTradeOwnerType        OwnerType;
    ///申报数量
    TXeleShortVolumeType       CombVolume;
    ///申报类型，'1':组合，'2':拆分，行权无效
    TXeleCombOrderType         CmbOrderType;
    ///组合策略类型，行权无效
    TXeleStrategyCombType      StgyCmbType;
    ///组合策略流水号 行权无效
    TXeleSecondaryOrderType     SecondaryOrderID;
    ///成分合约个数
    TXeleSumLegsType            SumLegs;
    ///成分合约扩展结构体
    CXeleCombLegField           CombLeg[4];
    ///成交时间
    TXeleTimeType               TransactTime;
    ///成交数量
    TXeleVolumeType             TradeVolume;
    ///未成交手数（报单数量-累计成交数量）
    TXeleVolumeType             LeavesVolume;
    ///订单状态
    TXeleOrderStatusType        OrderStatus;
    ///客户端登录子节点
    TXeleSubClientIndexType     SubClientIndex;
    ///业务单元，供客户自身业务使用
    TXeleBusinessUnitType       BusinessUnit;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType      StrOrderSysID;
    ///交易所类型
    TXeleMarketType             Market;
    ///交易前置id
    TXeleExchangeIDType         ExchangeFrontID;
    ///投资者账号
    TXeleUserIDType             AccountID;
    ///预留
    char                        Reserved[57];
};
```

```
};
```

其中期权多腿组合信息结构体CXeleCombOrderField如下：

```
struct CXeleCombOrderField {  
    ///成份合约代码  
    TXeleSecuritiesIDType      LegSecuritiesID;  
    ///成份合约方向,行权不填  
    TXeleLegSideType           LegSide;  
    ///预留  
    char                        Reserved;  
    ///成分合约数量  
    TXeleShortVolumeType       Volume;  
};
```

onRspInsertExercise方法

功能：单腿行权报单应答

函数原形：

```
void onRspInsertExercise (CXeleRspOptionExerciseAndLockField *pRspField,  
    CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);
```

参数：

- pRspField：行权报单应答，其结构体CXeleRspOptionExerciseAndLockField如下：

```
struct CXeleRspOptionExerciseAndLockField {  
    ///柜台报单编号int类型  
    TXeleOrderIDType      OrderSysID;  
    ///用户本地报单编号  
    TXeleOrderIDType      UserLocalID;  
    ///合约编码(证券锁定/证券解锁时表示标的代码)  
    TXeleSecuritiesIDType SecuritiesID;  
    ///请求数量  
    TXeleVolumeType       Volume;  
    ///锁定/解锁,行权无效  
    TXeleDirectionType    Side;  
    ///订单所有类型  
    TXeleTradeOwnerType    OwnerType;  
    ///客户端登录子节点  
    TXeleSubClientIndexType SubClientIndex;  
    ///业务单元(用户定义)  
    TXeleBusinessUnitType  BusinessUnit;  
    ///柜台报单编号str类型  
    TXeleStrOrderSysIDType StrOrderSysID;  
    ///交易所类型  
    TXeleMarketType        Market;  
    ///投资者账号  
    TXeleUserIDType        AccountID;  
    ///预留
```

```
char Reserved[56];  
};
```

onErrRtnInsertExercise

功能：单腿行权错误回报

函数原形：

```
void onErrRtnInsertExercise (CXeleRspOptionExerciseAndLockField *pRspField,  
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);
```

参数：

- pRspField：单腿行权错误回报，其回报结构体CXeleRspOptionExerciseAndLockField如下：

```
struct CXeleRspOptionExerciseAndLockField {  
    ///柜台报单编号int类型  
    TXeleOrderIDType OrdersSysID;  
    ///用户本地报单编号  
    TXeleOrderIDType UserLocalID;  
    ///合约编码(证券锁定/证券解锁时表示标的代码)  
    TXeleSecuritiesIDType SecuritiesID;  
    ///请求数量  
    TXeleVolumeType Volume;  
    ///锁定/解锁,行权无效  
    TXeleDirectionType Side;  
    ///订单所有类型  
    TXeleTradeOwnerType OwnerType;  
    ///客户端登录子节点  
    TXeleSubClientIndexType SubClientIndex;  
    ///业务单元(用户定义)  
    TXeleBusinessUnitType BusinessUnit;  
    ///柜台报单编号str类型  
    TXeleStrOrderSysIDType StrOrderSysID;  
    ///交易所类型  
    TXeleMarketType Market;  
    ///投资者账号  
    TXeleUserIDType AccountID;  
    ///预留  
    char Reserved[56];  
};
```

onRspCancelExercise方法

功能：单腿行权撤单应答

函数原形：

```
void onRspCancelExercise (CXeleRspOrderActionField *pRspField, CXeleRspInfo  
*pRspInfo, int nRequestID, bool bIsLast);
```

参数：

- pRspField：单腿行权撤单应答，其结构体CXeleRspOrderActionField如下：

```
struct CXeleRspOrderActionField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///被撤单柜台报单编号
    TXeleOrigSysIDType         OrigSysID;
    ///被撤单用户本地报单编号(柜台2.5以及2.5以上版本才支持该字段)
    TXeleOrderIDType          OrigUserLocalID;
    ///被撤单柜台报单编号str类型(柜台2.5以及2.5以上版本才支持该字段)
    TXeleStrOrderSysIDType     OrigStrOrderSysID;
    ///预埋单标记 '0':非预埋单 '1':预埋单
    TXelePreOrderFlag          PreOrderFlag;
    ///预留
    char                       Reserved0[7];
    ///错误编号
    TXeleErrorIDType           ErrorID;
    ///客户端登录子节点
    TXeleSubClientIndexType     SubClientIndex;
    ///业务单元(用户定义)
    TXeleBusinessUnitType       BusinessUnit;
    ///订单所有类型
    TXeleTradeOwnerType         OwnerType;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType      StrOrderSysID;
    ///投资者账号
    TXeleUserIDType             AccountID;
    ///交易所类型
    TXeleMarketType             Market;
    ///交易所报单编号(暂不使用)
    TXeleOrderExchangeIDType    OrderExchangeID;
    ///交易前置id(暂不使用)
    TXeleExchangeIDType         ExchangeFrontID;
    ///要约收购人编码(上交所要约该字段有效)
    TXeleTendererIDType         TendererID;
    ///预留
    char                       Reserved1[5];
};
```

onErrRtnCancelExercise方法

功能：单腿行权撤单错误回报

函数原形：

```
void onErrRtnCancelExercise (CXeleRspOrderActionField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast);
```

参数：

- pRspField：单腿行权操作错误回报，其结构体CXeleRspOrderActionField如下：

```
struct CXeleRspOrderActionField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///被撤单柜台报单编号
    TXeleOrigSysIDType         OrigSysID;
    ///被撤单用户本地报单编号(柜台2.5以及2.5以上版本才支持该字段)
    TXeleOrderIDType          OrigUserLocalID;
    ///被撤单柜台报单编号str类型(柜台2.5以及2.5以上版本才支持该字段)
    TXeleStrOrderSysIDType     OrigStrOrdersSysID;
    ///预埋单标记 '0':非预埋单 '1':预埋单
    TXelePreOrderFlag          PreOrderFlag;
    ///预留
    char                       Reserved0[7];
    ///错误编号
    TXeleErrorIDType           ErrorID;
    ///客户端登录子节点
    TXeleSubClientIndexType     SubClientIndex;
    ///业务单元(用户定义)
    TXeleBusinessUnitType       BusinessUnit;
    ///订单所有类型
    TXeleTradeOwnerType         OwnerType;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType      StrOrderSysID;
    ///投资者账号
    TXeleUserIDType             AccountID;
    ///交易所类型
    TXeleMarketType             Market;
    ///交易所报单编号(暂不使用)
    TXeleOrderExchangeIDType    OrderExchangeID;
    ///交易前置id(暂不使用)
    TXeleExchangeIDType         ExchangeFrontID;
    ///要约收购人编码(上交所要约该字段有效)
    TXeleTendererIDType         TendererID;
    ///预留
    char                       Reserved1[5];
};
```

onRtnExerciseOrder方法

功能：单腿行权报单回报

函数原形：

```
void onRtnExerciseOrder (CXeleRtnOptionExerciseAndLockField *pRspField,
    CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);
```

参数：

- pRspField：单腿行权报单回报，其结构体CXeleRtnOptionExerciseAndLockField如下：

```

struct CXeleRtnOptionExerciseAndLockField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///合约编码
    TXeleSecuritiesIDType     SecuritiesID;
    ///请求数量
    TXeleVolumeType           Volume;
    ///锁定/解锁,行权无效
    TXeleDirectionType        Side;
    ///订单所有类型
    TXeleTradeOwnerType        OwnerType;
    ///订单状态
    TXeleOrderStatusType       OrderStatus;
    ///客户端登录子节点
    TXeleSubClientIndexType    SubClientIndex;
    ///业务单元(用户定义)
    TXeleBusinessUnitType      BusinessUnit;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType     StrOrderSysID;
    ///交易所类型
    TXeleMarketType           Market;
    ///接受请求时间
    TXeleTimeType              TransactTime;
    ///交易所报单编号
    TXeleOrderExchangeIDType   OrderExchangeID;
    ///交易前置id
    TXeleExchangeIDType        ExchangeFrontID;
    ///投资者账号
    TXeleUserIDType            AccountID;
    ///预留
    char                       Reserved1;
    ///被撤单用户本地报单编号
    TXeleOrderIDType           OrigUserLocalID;
    ///被撤单柜台报单编号int类型
    TXeleOrigSysIDType          OrigOrderSysID;
    ///被撤单柜台报单编号str类型(带号段)
    TXeleStrOrderSysIDType      OrigStrOrderSysID;
    ///预留
    char                       Reserved[38];
};

```

onRspInsertExerciseComb方法

功能：组合行权报单应答

函数原形：

```

void onRspInsertExerciseComb (CXeleRspOptionCombOrderField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：组合行权报单应答，其结构体CXeleRspOptionCombOrderField如下：

```

struct CXeleRspOptionCombOrderField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///订单所有类型
    TXeleTradeOwnerType       OwnerType;
    ///申报数量
    TXeleShortvolumeType      CombVolume;
    ///申报类型，'1':组合，'2':拆分，行权无效
    TXeleCombOrderType        CmbOrderType;
    ///组合策略类型，行权无效
    TXeleStrategyCombType     StgyCmbType;
    ///组合策略流水号 拆分时有效
    TXeleSecondaryOrderType   SecondaryOrderID;
    ///成分合约个数
    TXeleSumLegsType          SumLegs;
    ///成分合约扩展结构体
    CXeleCombLegField         CombLeg[4];
    ///客户端登录子节点
    TXeleSubClientIndexType   SubClientIndex;
    ///业务单元，供客户自身业务使用
    TXeleBusinessUnitType     BusinessUnit;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType    StrOrderSysID;
    ///交易所类型
    TXeleMarketType           Market;
    ///投资者账号
    TXeleUserIDType           AccountID;
    ///预留
    char                       Reserved[32];
};

```

onErrRtnInsertExerciseComb方法

功能：组合行权错误回报

函数原形：

```

void onErrRtnInsertExerciseComb(CXeleRspOptionCombOrderField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：组合行权应答，其结构体CXeleRspOptionCombOrderField如下：

```

struct CXeleRspOptionCombOrderField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;

```

```

///订单所有类型
TXeleTradeOwnerType      OwnerType;
///申报数量
TXeleShortVolumeType     CombVolume;
///申报类型, '1':组合, '2':拆分, 行权无效
TXeleCombOrderType       CmbOrderType;
///组合策略类型, 行权无效
TXeleStrategyCombType    StgyCmbType;
///组合策略流水号 拆分时有效
TXeleSecondaryOrderType   SecondaryOrderID;
///成分合约个数
TXeleSumLegsType          SumLegs;
///成分合约扩展结构体
CXeleCombLegField        CombLeg[4];
///客户端登录子节点
TXeleSubClientIndexType   SubClientIndex;
///业务单元, 供客户自身业务使用
TXeleBusinessUnitType     BusinessUnit;
///柜台报单编号str类型
TXeleStrOrderSysIDType    StrOrderSysID;
///交易所类型
TXeleMarketType           Market;
///投资者账号
TXeleUserIDType           AccountID;
///预留
char                      Reserved[32];
};

```

onRspCancelExerciseComb方法

功能：组合行权撤单应答

函数原形：

```

void onRspCancelExerciseComb (CXeleRspOrderActionField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：撤单应答，其结构体CXeleRspOrderActionField如下：

```

struct CXeleRspOrderActionField {
    ///柜台报单编号int类型
    TXeleOrderIDType      OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType      UserLocalID;
    ///被撤单柜台报单编号
    TXeleOrigSysIDType     OrigSysID;
    ///被撤单用户本地报单编号(柜台2.5以及2.5以上版本才支持该字段)
    TXeleOrderIDType       OrigUserLocalID;
    ///被撤单柜台报单编号str类型(柜台2.5以及2.5以上版本才支持该字段)
    TXeleStrOrderSysIDType OrigStrOrderSysID;
    ///预埋单标记 '0':非预埋单 '1':预埋单
};

```

```

TXelePreOrderFlag          PreOrderFlag;
///预留
char                       Reserved0[7];
///错误编号
TXeleErrorIDType           ErrorID;
///客户端登录子节点
TXeleSubClientIndexType    SubClientIndex;
///业务单元(用户定义)
TXeleBusinessUnitType      BusinessUnit;
///订单所有类型
TXeleTradeOwnerType        OwnerType;
///柜台报单编号str类型
TXeleStrOrderSysIDType     StrOrderSysID;
///投资者账号
TXeleUserIDType            AccountID;
///交易所类型
TXeleMarketType            Market;
///交易所报单编号(暂不使用)
TXeleOrderExchangeIDType   OrderExchangeID;
///交易前置id(暂不使用)
TXeleExchangeIDType        ExchangeFrontID;
///要约收购人编码(上交所要约该字段有效)
TXeleTendererIDType        TendererID;
///预留
char                       Reserved1[5];
};

```

onErrRtnCancelExerciseComb方法

功能：组合行权撤单错误回报

函数原形：

```

void onErrRtnCancelExerciseComb (CXeleRspOrderActionField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：撤单应答，其结构体CXeleRspExerciseCombInsertField如下：

```

struct CXeleRspOrderActionField {
///柜台报单编号int类型
TXeleOrderIDType          OrderSysID;
///用户本地报单编号
TXeleOrderIDType          UserLocalID;
///被撤单柜台报单编号
TXeleOrigSysIDType         OrigSysID;
///被撤单用户本地报单编号(柜台2.5以及2.5以上版本才支持该字段)
TXeleOrderIDType          OrigUserLocalID;
///被撤单柜台报单编号str类型(柜台2.5以及2.5以上版本才支持该字段)
TXeleStrOrderSysIDType     OrigStrOrderSysID;
///预埋单标记 '0':非预埋单 '1':预埋单
TXelePreOrderFlag          PreOrderFlag;
};

```

```

    ///预留
    char Reserved0[7];
    ///错误编号
    TXeleErrorIDType ErrorID;
    ///客户端登录子节点
    TXeleSubClientIndexType SubClientIndex;
    ///业务单元(用户定义)
    TXeleBusinessUnitType BusinessUnit;
    ///订单所有类型
    TXeleTradeOwnerType OwnerType;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType StrOrderSysID;
    ///投资者账号
    TXeleUserIDType AccountID;
    ///交易所类型
    TXeleMarketType Market;
    ///交易所报单编号(暂不使用)
    TXeleOrderExchangeIDType OrderExchangeID;
    ///交易前置id(暂不使用)
    TXeleExchangeIDType ExchangeFrontID;
    ///要约收购人编码(上交所要约该字段有效)
    TXeleTendererIDType TendererID;
    ///预留
    char Reserved1[5];
};

```

onRtnExerciseCombOrder方法

功能：组合行权报单回报

函数原形：

```

void onRtnExerciseCombOrder (CXeleRtnOptionCombOrderField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：组合行权报单回报，其结构体CXeleRtnOptionCombOrderField如下：

```

struct CXeleRtnOptionCombOrderField {
    ///柜台报单编号int类型
    TXeleOrderIDType OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType UserLocalID;
    ///订单所有类型
    TXeleTradeOwnerType OwnerType;
    ///申报数量
    TXeleShortVolumeType CombVolume;
    ///申报类型，'1':组合，'2':拆分，行权无效
    TXeleCombOrderType CombOrderType;
    ///组合策略类型，行权无效
    TXeleStrategyCombType StgyCmbType;
    ///组合策略流水号 拆分时有效
};

```

```

TxleSecondaryOrderType      SecondaryOrderID;
///成分合约个数
TxleSumLegsType             SumLegs;
///成分合约扩展结构体
CxleCombLegField            CombLeg[4];
///接受请求时间
TxleTimeType                TransactTime;
///订单状态
TxleOrderStatusType         OrderStatus;
///客户端登录子节点
TxleSubClientIndexType      SubClientIndex;
///业务单元，供客户自身业务使用
TxleBusinessUnitType        BusinessUnit;
///柜台报单编号str类型
TxleStrOrderSysIDType       StrOrderSysID;
///交易所类型
TxleMarketType              Market;
///交易前置id
TxleExchangeIDType          ExchangeFrontID;
///交易所报单编号
TxleOrderExchangeIDType     OrderExchangeID;
///投资者账号
TxleUserIDType              AccountID;
///预留
char                         Reserved1;
///被撤单用户本地报单编号
TxleOrderIDType             OrigUserLocalID;
///被撤单柜台报单编号int类型
TxleOrigSysIDType           OrigOrderSysID;
///被撤单柜台报单编号str类型(带号段)
TxleStrOrderSysIDType       OrigStrOrderSysID;
///预留
char                         Reserved[38];
};

```

onRspQryOptionPosition方法

功能：期权持仓查询应答

函数原形：

```

void onRspQryOptionPosition (CxleRspQryOptionPositionField *pRspField,
CxleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：期权持仓查询应答，其结构体CxleRspQryOptionPositionField如下：

```

struct CxleRspQryOptionPositionField {
///资金账户
TxleUserIDType            AccountID;
///证券代码
TxleSecuritiesIDType      SecuritiesID;

```

```

    ///权利仓日初持仓
    TXelePositionType          InitLongYdPosition;
    ///义务仓日初持仓
    TXelePositionType          InitShortYdPosition;
    ///权利仓日初持仓剩余
    TXelePositionType          LongYdPosition;
    ///权利仓今日持仓
    TXelePositionType          LongPosition;
    ///义务仓日初持仓剩余
    TXelePositionType          ShortYdPosition;
    ///义务仓今日持仓
    TXelePositionType          ShortPosition;
    ///在途开仓数量
    TXeleUnTdPositionType      UnTdOpenPosition;
    ///在途平仓数量
    TXeleUnTdPositionType      UnTdClosePosition;
    ///交易所类型
    TXeleMarketType            Market;
    ///成交持仓比(标的物查询使用)
    TXeleSpeculationRatioType   SpecRatio;
    ///权力仓成本价
    TXeleAveragePriceType       AvgBoughtPrice;
    ///义务仓成本价
    TXeleAveragePriceType       AvgSoldPrice;
    ///在途买开数量
    TXelePositionType          UnTdOpenBoughtPos;
    ///在途卖平数量
    TXelePositionType          UnTdCloseSellPos;
    ///组合权利仓
    TXelePositionType          LongCombPosition;
    ///组合义务仓
    TXelePositionType          ShortCombPosition;
    ///备兑持仓
    TXelePositionType          CoveredPos;
    ///在途备兑开仓
    TXelePositionType          UnTdOpenCoveredPos;
    ///在途备兑平仓
    TXelePositionType          UnTdCloseCoveredPos;
    ///预留
    char                        Reserved[128];
};

```

onRspQryOptionFund方法

功能：期权资金查询应答

函数原形：

```

void onRspQryOptionFund (CXeleRspQryOptionClientAccountField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：期权资金查询应答，其结构体CXeleRspQryOptionClientAccountField如下：

```

struct CXeleRspQryOptionClientAccountField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///交易账户状态
    TXeleAcctStatusType      AcctStatus;
    ///保证金
    TXeleMoneyType           Margin;
    ///冻结保证金
    TXeleMoneyType           FrozenMargin;
    ///权利金
    TXeleMoneyType           Premium;
    ///冻结权利金
    TXeleMoneyType           FrozenPremium;
    ///手续费
    TXeleMoneyType           Fee;
    ///冻结手续费
    TXeleMoneyType           FrozenFee;
    ///持仓盈亏
    TXeleMoneyType           PosGainLoss;
    ///平仓盈亏
    TXeleMoneyType           CloseGainLoss;
    ///可用资金
    TXeleMoneyType           Available;
    ///出金
    TXeleMoneyType           Withdraw;
    ///入金
    TXeleMoneyType           Deposit;
    ///初始上场资金（不变）
    TXeleTotalFundType       InitTotalFund;
    ///上场资金（可变）
    TXeleTotalFundType       TotalFund;
    ///初始保证金
    TXeleMoneyType           DayMargin;
    ///交易所类型
    TXeleMarketType          Market;
    ///总限额
    TXeleMoneyType           TotalQuota;
    ///已用额度
    TXeleMoneyType           UsedQuota;
    ///预留
    char                      Reserved[112];
};

```

onRspQryOptionSecurities方法

功能：期权合约查询应答

函数原形：

```

void onRspQryOptionSecurities(CXeleRspQryOptionSecuritiesField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast) ;

```

参数：

- pRspField：期权合约查询应答，其结构体CXeleRspQryOptionSecuritiesField如下：

```

struct CXeleRspQryOptionSecuritiesField {
    ///证券代码
    TXeleSecuritiesIDType          SecuritiesID;
    ///合约交易代码
    TXeleTradeOptionCodeType       TradeOptionCode;
    ///合约名称
    TXeleSecuritiesNameType        SecuritiesName;
    ///期权类型
    TXeleOptionsTypeType           OptionType;
    ///标的证券代码
    TXeleSecuritiesIDType          UnderlySecuritiesID;
    ///标的证券名称
    TXeleSecuritiesNameType        UnderlySecuritiesName;
    ///标的交易品种
    TXeleUnderlyingType            UnderlySecuritiesType;
    ///涨跌标志（C/P）
    TXeleCPFlagType                CPFlag;
    ///期权行权价
    TXeleExercisePriceType         ExercisePrice;
    ///期权行权日
    TXeleShortTimeType             ExerciseDate;
    ///期权交割日
    TXeleShortTimeType             DeliveryDate;
    ///合约乘数
    TXeleMultiUnitType             MultiUnit;
    ///最小变动价位
    TXeleMinTickPriceType          TickPrice;
    ///昨结算价
    TXelePreSettlePriceType         PreSettlePrice;
    ///标的证券前收盘价
    TXeleUnderlyPreClosePriceType  UnderlyPreClosePrice;
    ///单位保证金
    TXeleMarginUnitType            MarginUnit;
    ///限价单最大买单量
    TXeleMaxLimitOrderVolumeType   MaxLimitBuyVolume;
    ///限价单最大卖单量
    TXeleMaxLimitOrderVolumeType   MaxLimitSellVolume;
    ///限价单最小买单量
    TXeleMinLimitOrderVolumeType   MinLimitBuyVolume;
    ///限价单最小卖单量
    TXeleMinLimitOrderVolumeType   MinLimitSellVolume;
    ///市价单最大买单量
    TXeleMaxMarketOrderVolumeType  MaxMarketBuyVolume;
    ///市价单最大卖单量
    TXeleMaxMarketOrderVolumeType  MaxMarketSellVolume;
    ///市价单最小买单量
    TXeleMinMarketOrderVolumeType  MinMarketBuyVolume;
    ///市价单最小卖单量
    TXeleMinMarketOrderVolumeType  MinMarketSellVolume;
    ///昨收盘价
    TXeleYdClosePxType             YdClosePrice;
    ///昨结算价
    TXeleYdSettlePxType            YdSettlePrice;

```

```

    ///跌停板价
    TXeleLowerPriceType          LowerPrice;
    ///涨停板价
    TXeleUpperPriceType          UpperPrice;
    ///首交易日
    TXeleDateType                StartDate;
    ///最后交易日
    TXeleDateType                EndDate;
    ///过期日
    TXeleDateType                ExpireDate;
    ///交易所类型
    TXeleMarketType              Market;
    ///资金账号
    TXeleUserIDType              AccountID;
    ///预留
    char                          Reserved[128];
};

```

onRspQryOptionRate 方法

功能：期权佣金费率、保证金率查询应答

函数原形：

```

void onRspQryOptionRate (CXeleRspQryOptionMarginFeeField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);

```

参数：

- pRspField：期权佣金费率、保证金率查询应答，其结构体CXeleRspQryOptionMarginFeeField如下：

```

struct CXeleRspQryOptionMarginFeeField {
    ///资金账户
    TXeleUserIDType          AccountID;
    ///证券代码
    TXeleSecuritiesIDType    SecuritiesID;
    ///营业部代码
    TXeleDepartmentIDType    DepartID;
    ///佣金费率
    TXeleMoneyType            CommissionRate;
    ///TXeleEntrustTypeType委托类型
    TXeleEntrustTypeType      EntrustType;
    ///TXeleChargingTypeType收费方式
    TXeleChargingTypeType     ChangingType;
    ///交易所类型
    TXeleMarketType           Market;
    ///预留
    char                      Reserved[156];
};

```

onRspQryOptionCombPosition 方法

功能：期权组合持仓查询应答

函数原形：

```
void onRspQryOptionCombPosition (CXeleRspQryOptionCombPositionField *pRspField,
CXeleRspInfo *pRspInfo, int nRequestID, bool bIsLast);
```

参数：

- pRspField：期权组合持仓查询应答，其结构体CXeleRspQryOptionCombPositionField如下：

```
struct CXeleRspQryOptionCombPositionField {
    ///资金账号
    TXeleUserIDType          AccountID;
    ///组合策略流水号,若为空,表示组合未成交
    TXeleSecondaryOrderType   SecondaryOrderID;
    ///组合策略类型
    TXeleStrategyCombType     StgyCmbType;
    ///组合成功数量
    TXeleVolumeType           CombVolume;
    ///解组成功数量
    TXeleVolumeType           UnCombVolume;
    ///初始组合数量(仅老仓有值,可用来标识新老仓)
    TXeleVolumeType           InitCombVolume;
    ///成分合约个数(最多4腿,暂时支持2腿)
    TXeleSumLegsType          SumLegs;
    ///成份合约扩展
    CXeleCombLegField         CombLeg[4];
    ///交易所类型
    TXeleMarketType          Market;
    ///在途组合数量
    TXeleVolumeType           UnTdCombVolume;
    ///在途解组数量
    TXeleVolumeType           UnTdUnCombVolume;
    ///柜台报单编号str类型,老仓时,该字段无效
    TXeleStrOrderSysIDType    StrOrderSysID;
    ///预留
    char                      Reserved[128];
};
```

其中期权成分扩展结构体CXeleCombLegField如下：

```

struct CXeleCombLegField {
    ///成份合约代码
    TXeleSecuritiesIDType      LegSecuritiesID;
    ///成份合约方向,行权不填
    TXeleLegSideType          LegSide;
    ///预留
    char                        Reserved;
    ///成分合约数量
    TXeleShortVolumeType      Volume;
};

```

onRspOTU 方法

功能：（期权）证券锁定/解锁响应

函数原形：

```

void onRspOTU(CXeleRspOptionExerciseAndLockField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast)

```

参数：

- pRspField：证券锁定/解锁响应，其结构体CXeleRspOptionExerciseAndLockField为：

```

struct CXeleRspOptionExerciseAndLockField {
    ///柜台报单编号int类型
    TXeleOrderIDType      OrderSysID;
    ///用户本地报单编号
    TXeleOrderIDType      UserLocalID;
    ///合约编码(证券锁定/证券解锁时表示标的代码)
    TXeleSecuritiesIDType SecuritiesID;
    ///请求数量
    TXeleVolumeType       Volume;
    ///锁定/解锁,行权无效
    TXeleDirectionType    Side;
    ///订单所有类型
    TXeleTradeOwnerType    OwnerType;
    ///客户端登录子节点
    TXeleSubClientIndexType SubClientIndex;
    ///业务单元(用户定义)
    TXeleBusinessUnitType  BusinessUnit;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType StrOrderSysID;
    ///交易所类型
    TXeleMarketType       Market;
    ///投资者账号
    TXeleUserIDType       AccountID;
    ///预留
    char                  Reserved[56];
};

```

onRtnOTU方法

功能：（期权）证券锁定/解锁回报

函数原形：

```
void onRtnOTU(CXeleRtnOptionExerciseAndLockField *pRspField, CXeleRspInfo
*pRspInfo, int nRequestID, bool bIsLast) ;
```

参数：

- pRspField：证券锁定/解锁回报，其结构体CXeleRtnOptionExerciseAndLockField为：

```
struct CXeleRtnOptionExerciseAndLockField {
    ///柜台报单编号int类型
    TXeleOrderIDType          OrdersSysID;
    ///用户本地报单编号
    TXeleOrderIDType          UserLocalID;
    ///合约编码
    TXeleSecuritiesIDType     SecuritiesID;
    ///请求数量
    TXeleVolumeType           Volume;
    ///锁定/解锁,行权无效
    TXeleDirectionType        Side;
    ///订单所有类型
    TXeleTradeOwnerType        OwnerType;
    ///订单状态
    TXeleOrderStatusType       OrderStatus;
    ///客户端登录子节点
    TXeleSubClientIndexType    SubClientIndex;
    ///业务单元(用户定义)
    TXeleBusinessUnitType      BusinessUnit;
    ///柜台报单编号str类型
    TXeleStrOrderSysIDType     StrOrderSysID;
    ///交易所类型
    TXeleMarketType           Market;
    ///接受请求时间
    TXeleTimeType             TransactTime;
    ///交易所报单编号
    TXeleOrderExchangeIDType   OrderExchangeID;
    ///交易前置id
    TXeleExchangeIDType        ExchangeFrontID;
    ///投资者账号
    TXeleUserIDType            AccountID;
    ///预留
    char                       Reserved1;
    ///被撤单用户本地报单编号
    TXeleOrderIDType          OrigUserLocalID;
    ///被撤单柜台报单编号int类型
    TXeleOrigSysIDType         OrigOrderSysID;
    ///被撤单柜台报单编号str类型(带号段)
    TXeleStrOrderSysIDType     OrigStrOrderSysID;
    ///预留
    char                       Reserved[38];
};
```

```
};
```

onErrRtnOTU 方法

功能：（期权）证券锁定/解锁错误回报

函数原形：

```
void onErrRtnOTU(CXeleRspOptionExerciseAndLockField *pRspField, CXeleRspInfo  
*pRspInfo, int nRequestID, bool bIsLast) ;
```

参数：

- pRspField：（期权）证券锁定/解锁错误回报，其结构体CXeleRspOptionExerciseAndLockField为：

```
struct CXeleRspOptionExerciseAndLockField {  
    ///柜台报单编号int类型  
    TXeleOrderIDType          OrdersSysID;  
    ///用户本地报单编号  
    TXeleOrderIDType          UserLocalID;  
    ///合约编码(证券锁定/证券解锁时表示标的代码)  
    TXeleSecuritiesIDType      SecuritiesID;  
    ///请求数量  
    TXeleVolumeType           Volume;  
    ///锁定/解锁,行权无效  
    TXeleDirectionType        Side;  
    ///订单所有类型  
    TXeleTradeOwnerType        OwnerType;  
    ///客户端登录子节点  
    TXeleSubClientIndexType     SubClientIndex;  
    ///业务单元(用户定义)  
    TXeleBusinessUnitType       BusinessUnit;  
    ///柜台报单编号str类型  
    TXeleStrOrderSysIDType      StrOrderSysID;  
    ///交易所类型  
    TXeleMarketType            Market;  
    ///投资者账号  
    TXeleUserIDType            AccountID;  
    ///预留  
    char                        Reserved[56];  
};
```

其他

Trader-API中各线程作用说明

心跳处理线程

API核柜台查询链路处理回报线程

API和manager链路处理回报线程

API交易链路发送和接收数据线程

注: 建议绑核api_config.txt中CpuCore的第一个参数

API交易链路处理交易回调线程

注: 建议绑核api_config.txt中CpuCore的第二个参数

LOG日志线程

字段类型

字段类型总表

序号	数据类型名	数据类型	数据类型说明
1	DWORD	unsigned int	DWORD双字类型
2	TXeleFairProtocolType Fair	unsigned char	协议号类型
3	TXeleMessageIDType	unsigned char	消息编号类型
4	TXeleMessageLenType	unsigned short	消息长度类型
5	TXeleRequestIDType	int	请求编号类型
6	TXeleSeqNoType	unsigned int	序列号类型
7	TXeleSeqSeriesType	unsigned char	流类型
8	TXeleOrigSysIDType	unsigned int	原报单编号类型
9	TXeleOrderExchangeIDType	字符数组	交易所报单编号类型
10	TXeleOrderIDType	unsigned int	报单编号类型
11	TXelePriceType	double	价格类型
12	TXeleVolumeType	unsigned int	数量类型
13	TXeleOrderTypeType	char	报单类型类型
14	TXeleSecuritiesIDType	字符数组	证券代码类型

序号	数据类型名	数据类型	数据类型说明
15	TXeleMarketType	char	交易所类型
16	TXeleDirectionType	char	交易方向类型
17	TXeleTimeConditionType	char	时间条件类型
18	TXeleTimeType	字符数组	长时间类型
19	TXeleOffsetFlagType	char	开平标志类型
20	TXeleErrorIDType	unsigned int	错误编号类型
21	TXeleErrorMsgType	字符数组	错误信息类型
22	TXeleOrderStatusType	char	报单状态类型
23	TXeleMoneyType	double	资金类型
24	TXeleUserIDType	字符数组	交易用户代码类型
25	TXeleUserPasswordType	字符数组	交易用户密码类型
26	TXeleAppIDType	字符数组	终端软件AppID类型
27	TXeleAuthCodeType	字符数组	终端软件授权码类型
28	TXeleParticipantIDTypes	字符数组	会员代码类型
29	TXeleIPType	IP类型	字符数组
30	TXeleErrorId	unsigned short	错误码类型
31	TXeleErrorLenType	unsigned char	错误信息长度类型
32	TXeleIsLastType	bool	结束标志类型
33	TXeleDateType	字符数组	日期类型
34	TXeleShortTimeType	字符数组	短时间类型
35	TXeleSessionIDType	short int	会话编号类型
36	TXeleTokenType	unsigned short int	令牌类型
37	TXeleSendBuffType	字符数组	发送缓冲区类型
38	TXeleLockedSecuritiesIDType	字符数组	锁定证券代码类型
39	TXeleTransferFeeRateType	double	交易所过户费率类型

序号	数据类型名	数据类型	数据类型说明
40	TXeleStampTaxRateType	double	交易所过户费率类型
41	TXeleAcctStatusType	char	交易账户状态类型
42	TXeleFrozeMarginType	double	冻结保证金（股票叫冻结资金）
43	TXeleTotalFeeType	double	总手续费类型
44	TXeleCurrGainLossType	double	当日盈亏类型
45	TXeleTotalGainLossType	double	总盈亏类型
46	TXeleFrozenFeeType	double	冻结手续费类型
47	TXeleUsedFeeType	double	已付手续费类型
48	TXeleAvailableFundType	double	可用资金类型
49	TXeleTotalFundType	double	总资金类型
50	TXeleSellFund	double	卖出资金类型
51	TXeleBuyFund	double	买入资金类型
52	TXeleAveragePriceType	double	成交均价（加权平均）类型
53	TXeleTdBuyPositionType	unsigned long	今买持仓类型
54	TXeleTdSellPositionType	unsigned long	今卖持仓类型
55	TXeleYdPositionType	unsigned long	昨持仓类型
56	TXeleTotalPositionType	unsigned long	总持仓类型
57	TXeleTotalCostType	double	持仓成本类型
58	TXeleSecuritiesNameType	字符数组	合约名称类型
59	TXeleCPFlagType	char	涨跌标志（期权C/P）类型
60	TXeleExercisePriceType	double	交易所过户费率类型
61	TXeleUncoveredPositionType	unsigned long	当前合约未平仓数（期权）类型

序号	数据类型名	数据类型	数据类型说明
62	TXelePreSettlePriceType	double	昨结算价类型
63	TXeleUnderlyPreClosePriceType	double	标的证券前收盘价
64	TXeleMarginUnitType	double	单位保证金（期权）类型
65	TXeleSetIDType	unsigned char	产品集类型
66	TXeleLimitPriceClassType	char	涨跌幅限制类型类型
67	TXeleMaxLimitOrderVolumeType	unsigned long	限价单最大下单量类型
68	TXeleMinLimitOrderVolumeType	unsigned long	限价单最小下单量类型
69	TXeleMaxMarketOrderVolumeType	unsigned long	市价单最大下单量类型
70	TXeleMinMarketOrderVolumeType	unsigned long	市价单最小单量类型
71	TXeleLowerPriceType	double	跌停板价类型
72	TXeleUpperPriceType	double	涨停板价类型
73	TXeleAvgPxType	double	持仓均价类型
74	TXeleTradeVolumeType	unsigned long	成交数量类型
75	TXeleUpRatioType	double	上浮比率类型
76	TXeleExerciseDayType	字符数组	行权日类型
77	TXeleSumLegsType	unsigned char	合约数量
78	TXeleSecondaryOrderType	字符数组	组合流水号类型
79	TXeleLegSideType	char	成份合约方向
80	TXeleStrategyCombCodeType	字符数组	策略组合编码类型
81	TXeleStrategyCombType	unsigned char	策略组合类型
82	TXeleCombOrderType	char	组合申报类型
83	TXeleCoveredFlagType	char	备兑标志类型

序号	数据类型名	数据类型	数据类型说明
84	TXeleCoveredFrozenPositionType	unsigned long	备兑冻结量类型
85	TXeleTrafficFeeType	double	交易所流量费率
86	TXeleYdPositionLeftType	long long	昨持仓剩余类型
87	TXeleUnTdFrozenCapType	double	在途冻结资金类型
88	TXeleUnTdFrozenPremiumType	double	在途冻结权利金类型
89	TXeleYdPositionCostType	double	买入成本类型
91	TXeleSecuritiesSubTypeType	字符数组	合约子品种类型
92	TXeleOptionsTypeType	字符数组	期权类型
93	TXelePositionType	unsigned long	持仓类型
94	TXeleClientIpType	字符数组	客户端IP类型
95	TXeleClientMacType	字符数组	客户端MAC类型
96	TXeleReservedType	字符数组	预留类型
97	TXeleReserved1Type	字符数组	预留类型
98	TXeleReserved2Type	字符数组	预留类型
99	TXeleDepartmentIDType	字符数组	营业部类型
100	TXeleTradeSystemIDType	unsigned char	柜台系统节点ID类型
101	TXeleUnTdPositionType	unsigned long	在途持仓类型
102	TXeleTradeOptionCodeType	字符数组	期权标识代码类型
103	TxeleBusinessUnitType	字符数组	客户业务处理单元类型
104	TXeleSubClientIndexType	unsigned char	客户端登录字节点类型
105	TxeleTradeOwnerType	unsigned char	订单所有类型
107	TXelePartyIdType	字符数组	席位类型
108	TXeleExchangeIDType	unsigned char	交易前置ID类型

序号	数据类型名	数据类型	数据类型说明
109	TxeleFlowRebuildType	unsigned char	流水重构类型
110	TXeleSecuritiesType	int	合约类型
111	TXeleDayTradingType	char	日内可转交易类型
112	TXeleOperwayType	char	委托方式
113	TXeleTransferDirectionType	char	出入金调拨方向类型
114	TXeleCurrencyType	char	币种类型
115	TXeleMarketIDType	char	市场代码类型
116	TXeleOrderMode	char	报单模式类型
117	TXeleEntrustTypeType	char	委托类型
118	TXeleChargingTypeType	char	收费方式类型
119	TXeleRemarkType	字符数组	备注类型
120	TXeleFundType	int	集中交易柜台主资金标志类型
121	TXeleStockQuotaType	long	新股配股额度类型
122	TXeleFlowCountType	unsigned int	流水数量类型
123	TXeleRecoveryFlagType	char	流水重构报文标记类型
124	TXeleInvestorIDType	字符数组	投资者ID类型
125	TXeleTimeIntervalType	int	心跳间隔(s) 类型
126	TXeleTimeOutType	int	超时时间(s)
127	TXeleUniqueNumberType	int	全局唯一消息编号类型,从1开始递增
128	TXeleIsFundReversalType	char	是否进行了资金冲正操作标记类型
129	TXeleReversalCounterType	字符数组	资金冲正柜台标记类型
130	TXeleReversalResultType	char	资金冲正结果标记类型
131	TXeleIpAddrType	字符数组	客户端ip地址
132	TXeleMacAddrType	字符数组	客户端mac地址

序号	数据类型名	数据类型	数据类型说明
133	TXeleHostNameType	字符数组	客户端主机名称
134	TXeleCpuSerialType	字符数组	客户端cpu序列号
135	TXeleHardDiskSerialType	字符数组	客户端硬盘序列号
136	TXeleCentralTradingErrorIdType	int	集中交易响应错误编码类型
137	TXeleCentralTradingErrorIdType	集中交易响应错误信息类型	集中交易响应错误信息类型
138	TXeleCommandNum	字符数组	通用接口命令号
139	TXeleExchangeIDIntType	int	交易前置ID类型(int类型)
140	TXeleFrozenPositionType	Unsigned int	冻结持仓类型
141	TXeleCoveredFrozenPositionType	Unsigned long	备兑冻结量类型
142	TXeleQtyType	long	长数量类型
143	TXeleETFCreationRedemptionExtraFee	double	ETF申赎附加费类型
144	TXeleETFCreationRedemptionCommFeeRate	double	ETF申赎佣金费率类型
145	TXeleETFCreationRedemptionTransferFeeRate	double	ETF申赎过户费类型
146	TXeleActionSourceType	unsigned char	操作来源
147	TXeleSystemFeaturesType	unsigned int	柜台支持的功能特性
148	TXeleTendererIDType	字符数组	要约收购人编码

附录

附录A 错误代码表

详见独立的错误码说明文件

附录B 数据字典

[Fair协议号类型]

字典子项	子项含义
0x5f	Fair协议号

[流水重构报文标记类型] TXeleRecoveryFlagType

字典子项	子项含义
0	正常回报
1	流水重构回报

[报单消息编号类型] TXeleMessageIDType

字典子项	子项含义
101	普通报单类型消息ID
103	普通撤单类型消息ID

[流类型] TXeleSeqSeriesType

字典子项	子项含义
1	对话流
2	会有私有流
3	公共流
4	查询流
5	交易员私有流
6	询价流

[报单类型类型] TXeleOrderTypeType

字典子项	子项含义
'1'	对手方最优剩余转限价(沪深)
'2'	限价(沪深)
'3'	本方最优(沪深)
'4'	市价立即成交剩余撤销(深交)
'5'	市价全额成交或撤销(深交)

字典子项	子项含义
'6'	最优五档立即成交剩余撤销(沪深)
'7'	最优五档立即成交剩余转限价(上交)
'8'	限价全额成交或撤销(股票期权)

[交易所交易前置描述符类型] TXeleMarketType

字典子项	子项含义	交互对象
'1'	上交股票	柜台
'2'	深交股票	柜台
'3'	上交期权	柜台
'4'	深交期权	柜台
'1'	上交柜台	Manager (管理中心)
'2'	深交柜台	Manager (管理中心)
'a'	沪深柜台资金查询汇总	Manager、柜台 (只为沪深柜台资金查询接口使用，其他接口不能使用)

[委托方式类型] TXeleOperwayType

字典子项	子项含义
'1'	API报单
'2'	其他方式报单

[交易方向类型] TXeleDirectionType

字典子项	子项含义
'1'	买
'2'	卖
'3'	ETF申购
'4'	ETF赎回
'7'	要约预受 (仅上交所支持)
'8'	要约撤销 (仅上交所支持)

[出入金调拨方向类型] TXeleTransferDirectionType

字典子项	子项含义
'1'	调出
'2'	调入

[时间条件类型] TXeleTimeConditionType

字典子项	子项含义
'0'	GFD当日有效
'3'	IOC 即时成交剩余自动撤销
'4'	FOK即时全部成交否则撤销

[合约类型] TXeleSecuritiesType

股票柜台2.5及以上版本支持

字典子项	上交股票	深交股票
1	股票	股票
2	基金	基金
3	可转债	可转债
4	新股	新股
5	配股	配股
6	新债申购	新债申购
7	配债	配债
8	科创板	核准制创业板
'9	科创板新股申购	核准制创业板申购
10	科创板配股	核准制创业板配股
11	非交易类型	
12		
13		
14		注册制创业板
15		注册制创业板申购
16		注册制创业板配股
17	逆回购	逆回购

字典子项	上交股票	深交股票
18	ETF基金	ETF基金
19	货币ETF基金	货币ETF基金
20	国债ETF基金	国债ETF基金
23	LOF基金	LOF基金
24	封闭式基金	封闭式基金

[报单状态类型] TXeleOrderStatusType

字典子项	子项含义
'0'	未报
'1'	正报
'2'	已报
'3'	已报待撤
'4'	部成待撤
'5'	部撤
'6'	已撤
'7'	部成
'8'	已成
'9'	废单

[流水重构类型] TXeleFlowRebuildType

字典子项	子项含义
0	不进行流水重构
1	只进行资金流水重构
2	只进行报文流水重构
3	资金和报文流水重构同时进行

[币种类型] TXeleCurrencyType

字典子项	子项含义
'0'	CNY人民币
'1'	USD 美元

字典子项	子项含义
'2'	HKD 港币

[开平标志类型] TXeleOffsetFlagType

字典子项	子项含义
'C'	平仓
'O'	开仓

[备兑标志类型] TXeleCoveredFlagType

字典子项	子项含义
'0'	深交备兑
'1'	深交非备兑
''	上交非备兑
'1'	上交备兑

[组合申报类型] TXeleCombOrderType

字典子项	子项含义
'1'	组合
'2'	解组

[策略组合类型] TXeleStrategyCombType

字典子项	子项含义
1	认购牛市价差策略
2	认沽熊市价差策略
3	认沽牛市价差策略
4	认购熊市价差策略
5	跨式空头策略
6	宽跨式空头策略
7	认购期权保证金开仓转备兑开仓

[合约方向类型] TXeleLegSideType

字典子项	子项含义
'1'	买开，权利仓
'2'	卖开，义务仓

[市场代码类型] TXeleMarketIDType

字典子项	子项含义
'1'	上交A股
'2'	上交B股
'3'	深交A股
'4'	深交B股
'5'	深交三板

[报单模式类型] TXeleOrderMode

字典子项	子项含义
'0'	用户模式
'1'	柜台模式

[订单所有类型] TXeleTradeOwnerType

字典子项	子项含义
1	个人投资者发起
101	交易所发起
102	会员发起
103	机构投资者发起
104	自营交易发起
105	流动性服务提供商发起

[委托类型] TXeleEntrustTypeType

字典子项	子项含义
'1'	买开
'2'	卖开
'3'	买平

字典子项	子项含义
'4'	卖平

[收费方式类型] TXeleChargingTypeType

字典子项	子项含义
'1'	按金额收取
'2'	按手数收取

[排序类型] TXeleSortType

字典子项	子项含义
0	默认排序
1	按时间正排序 先报的单先查到
2	按时间倒排序 新报的单先查到

[批量报单类型]TXeleBatchType

字典子项	子项含义
'1'	等量拆单
'2'	递减拆单
'3'	多证券委托组合

[查询订单状态位图类型] TxeleQryStatusType

字典子项	子项含义
QRYSTAT_UNREPORT	未报
QRYSTAT_REPORTING	正报
QRYSTAT_REPORTED	已报
QRYSTAT_REPORTED_ACTION	已报待撤
QRYSTAT_PTRADE_ACTION	部成待撤
QRYSTAT_PACTION	部撤
QRYSTAT_ACTIONED	已撤
QRYSTAT_PTRADE	部成
QRYSTAT_TRADED	已成
QRYSTAT_ERROR	废单

[报单来源类型] TXeleOrderSourceType

字典子项	子项含义
ORDERSOURCEFPGA	api硬件通道报单
ORDERSOURCESOFT	API软件通道报单
ORDERSOURCEWEB	web撤单
ORDERSOURCEGATEWAYSELF	接入网关报单(本平台报单)
ORDERSOURCEGATEWAYOTHER	接入网关报单(其他平台报单)

[通用接口命令号] TXeleCommandNum

字典子项	子项含义
AccountInit	用户密码初始化

[程序化风控类别] TXelePrcProgramType

字典子项	子项含义
0	默认返回所有
1	全天报撤笔数限制风控
2	流速风控

[持仓划拨方向类别] TXeleInOutDirectionType

字典子项	子项含义
'1'	划入：从集中交易把持仓划入到Xele柜台
'2'	划出：从Xele柜台把持仓划出到集中交易
'5'	备兑锁定-CoverLock
'6'	备兑解锁-CoverUnlock

[报单通道类型] TXeleChannelType

字典子项	子项含义
1	FPGA硬件通道
2	软件通道

[网关类型] TXeleGateWayType

字典子项	子项含义
1	深交所现货集中竞价交易平台
2	深交所综合金融服务平台
3	深交所非交易处理平台
4	深交所衍生品集中竞价交易平台
5	深交所国际市场互联平台
6	深交所固定收益交易平台
7	深交所行情
21	上交所竞价撮合平台
22	上交所综合业务平台
23	上交所期权业务平台
24	上交所港股通平台
25	上交所新债券交易平台
26	上交所固定收益平台
27	上交所互联网交易平台
28	上交所行情

[网关状态类型] TXeleGateWayStatus

字典子项	子项含义
1	未开放
2	预开放
3	开放
4	暂停
5	关闭
6	离线
7	禁用

[划拨模式类型] TXeleInOutMode

字典子项	子项含义
1	经过集中柜台
2	不经过集中柜台

[专业投资者标志类型] TXeleInvestorFlag

字典子项	子项含义
1	普通投资者
2	专业投资者

[股票风险级别] TXeleStockLevel

字典子项	子项含义
'0'	正常
'1'	风险警示
'2'	退市整理期

[发行方式] TXeleIssueType

字典子项	子项含义
1	配股配债
2	增发
3	新股申购
4	新债申购
5	基金申购

[证券子类别代码] TxeleSecuritiesSubType

字典子项	子项含义
1	主板 A 股
2	创业板股票
3	主板 B 股
4	国债（含地方债）
5	企业债
6	公司债

字典子项	子项含义
7	可转债
8	私募债
9	可交换私募债
10	证券公司次级债
11	质押式回购
12	资产支持证券
13	本市场股票ETF
14	跨市场股票ETF
15	跨境ETF
16	本市场实物债券ETF
17	现金债券ETF
18	黄金ETF
19	货币ETF
20	杠杆ETF
21	商品期货ETF
22	标准LOF
23	分级子基金
24	封闭式基金
25	仅申赎基金
26	权证
27	个股期权
28	ETF期权
29	优先股
30	证券公司短期债
31	可交换公司债
32	主板存托凭证
33	创业板存托凭证
34	基础设施基金
35	定向可转债

字典子项	子项含义
36	跨银行间实物债券ETF
37	科创板
38	科创板存托凭证
39	控制指令
40	无类别

[操作来源] TXeleActionSourceType

字典子项	子项含义
1	API发起的
2	monitor发起的
3	manager发起的

[调入调出类型] TXeleTransferType

字典子项	子项含义
0	可用资金
1	RTGS额度

[涨跌幅限制类型] TXeleLimitPriceClassType

字典子项	子项含义
'N'	表示交易规则（2013修订版）3.4.13规定的有涨跌幅限制类型或者权证管理办法第22条规定
'R'	表示交易规则（2013修订版）3.4.15和3.4.16规定的无涨跌幅限制类型
'S'	表示回购涨跌幅控制类型
'F'	表示基于参考价格的涨跌幅控制
'P'	表示IPO上市首日的涨跌幅控制类型
'U'	表示无任何价格涨跌幅控制类型

[柜台支持的功能特性类型] TXeleSystemFeaturesType

字典子项	子项含义
bit0	1表示支持UDP报单，0表示不支持；
bit1~31	保留扩展；

FAQ

- 【问题1】用户每个账户都会创建一个对象，10个账号就有10个对象，这10个账户都绑定在一个CPU（进程）上，还是需要绑不同的CPU（进程）？

答：可以绑一个CPU也可以绑多个CPU。一个CPU（进程）可以处理多个API。每个API只能处理一个账户，即一个CPU（进程）可以绑定多个账号。如果需要多线程使用可以创建多个API实例，每个实例用不同的节点 SubClientIndex登录

- 【问题2】发单的时候需要指定交易所吗？

答：不需要，由于深交和上交属于2个柜台，客户需要用2个API对接2个不同的柜台，暂不支持一个API同时连接两个柜台。

- 【问题3】在onRtnOrder回报字段中，交易所报单ID（OrderExchangeID）在生产实盘中是不是真的有交易所报单编号？

答：有的。

- 【问题4】怎么实现Api断线重连功能（异常处理）？

答：目前Api可以支持连接艾科管理中心、柜台，当完成连接后，如果发生了链路断连时，有以下两种场景。

场景一：管理中心连接断了，这时不会影响到Api的交易和查询功能，如果不需要进行柜台间资金调拨时，无需处理。

场景二：柜台的查询或者交易链路断连，这时整个Api的连接都会断掉，无法进行任何操作，需要重新发送登录请求来进行重新建连。

当发生异常断连时会收到以下对应的回调函数：

onFrontManagerQueryDisconnected(int nReason) 对应管理中心连接断连

onFrontQueryDisconnected(int nReason)对应柜台查链路断连

onFrontTradeDisconnected(int nReason)对应柜台交易链路断连

考虑到编程实现复杂情况，可以在收到以上任意一种断连回调后，先调用登出reqUserLogout接口，等待一秒后再调用登录reqUserLogin接口来重新建连。

- 【问题5】api需要配置几个URL？

答：如果配置了ManagerURL，则其他URL可以都不配置，因为manager会返回其他URL，但是如果配置了其他URL，则配置的URL优先级最高。

- 【问题6】哪些配置是确认全部支持的？

答：如果配置里面有强调部分券商支持的，那要跟券商确认是否支持，如果未标明是部分券商支持的，则表示是通用支持功能。

- 【问题7】FpgaRspRcv的设计初衷是什么？

答：硬件通道默认接收当前账户其他节点的回报，但是如果交易客户不需要其他节点的回报，那会需要额外的资源处理这些数据，且链路回报性能也会影响。如果配置不接收，则可以节省客户资源和链路资源。

- 【问题8】SuperLog打开后会影响到报单或回报性能吗？

答：会的。

- 【问题9】CreateRtnOrderByRtnTrade是干什么用的？

答：这个参数和柜台参数有重合，是否打开要和券商确认清楚。如果和柜台同时打开，在成交回报接收的同时，会收到两个一样的报单回报，若api版本配置文件中无此参数，请忽略。

- 【问题10】RecvSendDetach参数有什么效果，如何使用？

答：如果这个参数配置了1。首先，就意味着api内部没有单独的报单线程了，只有客户自己的报单线程了。那么CpuCore这个参数的第一个核配置就可以配置成-1，只需要绑定客户自己的报单线程即可，需要客户自己绑定。其次，报单性能会更好，如果抓网卡出去的报文时间戳-调用报单接口时的时间戳，会发现时间缩短了。但是要注意，api调用报单接口的时间增加了，这是正常现象，因为调用时间包括了发送到网卡的时间，而原来是不包括的。相当于做了整体的穿透优化。

- 【问题11】AllSuperviseInfo和SuperviseExtraInfo什么区别？

答：前者如果配置了，正确性由客户保证，api和柜台会不做任何更改直接落库到柜台表中。后者只是补充了公网IP和IPORT的补充信息，柜台会和自动获取的部分进行拼接落库。

- 【问题12】登录报错55086具体是什么信息错误？

答：55086的报错对应信息是Api supervise information check fail，意思是api传输到柜台的监管信息不符合交易所要求。柜台会对以下重要构成元素进行校验，LIP，MAC，HD，PCN，CPU，若监管信息整体为空或以上元素为空（空即NA）则柜台会认为此监管信息无效，不允许登录，若出现该报错，请交易客户详细检查监管信息格式。