

速子信息极速交易系统 接口使用说明书

2024- 01 - 03



一、TITD接口简介

1.1 TITD发布文件概览

TITD发布接口包括4个头文件、linux库文件，examples。

发布头文件	TiTdApi.h	TITD头文件，包括两个类CTiTdApi和CTiTdSpi
	TiErrCode.h	TITD_SERVER定义的错误码，所有的错误码为负数。 交易所平台错误对应的错误信息请参考TITD_ERRCODE.xls
	TiDataType.h	TITD定义的数据类型
	TiDataStruct.h	TITD定义的结构体
TITD API库	titdapi.so	
示例代码	titdapi.ini	初始化配置文件(init函数中用配置文件初始化时使用)
	TiExample.h	主启动头文件
	TiExample.cpp	主启动源文件
	TiExampleInit.cpp	api初始化相关示例
	TiExampleLogin.cpp	客户登录相关示例
	TiExampleQuery.cpp	异步查询相关示例
	TiExampleGet.cpp	同步查询相关示例
	TiExampleOrder.cpp	报单相关示例
	TiExampleQuote.cpp	报价、询价相关示例
	TiExampleExercise.cpp	行权相关示例
	TiExampleOther.cpp	其他示例

1.2 接口版本更新说明

更新日期	更新内容
2022-03-02	“更新密码”、“登出”功能更新。
2022-03-10	交易编码类型状态，新增“禁止开仓”、无权限。
2022-03-22	查询通知时，如果没填入具体的userid，则通过clientid来获取其中的一个userid来查询
2022-03-24	get方法获取资金，获取的是登录时的“实时资金”，盘中不再变化。
2022-08-03	登录配置中的接收请求的网卡端口RecvNicPort限定为51100-51200，不设置即为从中随机选择。
2022-08-19	新增tcp方式连接的库libtitdapi_tcp.so。
2022-09-06	新增库libtitdapi_sf.so、libtitdapi_exa.so，新增看穿式信息采集。
2022-11-04	更新了头文件和库，加了股票非交易业务。
2022-11-15	更新了库，交易相关的请求发送正确统一返回0，不再返回数据包大小。
2022-11-23	行权撤销、组合行权撤销、组合、解组合，若成功时，统一调整为不推送OnRsp***的响应。
2022-04-19	完善报价功能。
2022-05-31	init时增加配置LockFree，1:无锁线程不安全; 其他:有锁线程安全，默认为其他线程安全。
2023-12-20	原4个api库（libtitdapi_tcp.so、libtitdapi.so、libtitdapi_sf.so、libtitdapi_exa.so），整合成统一使用titdapi.so。

1.3 TITD接口概述

速子信息极速交易系统（以下简称TITD）是一套由速子信息技术（杭州）有限公司开发的交易软件。

客户应用TITD，可以连接上海期货交易所、中国金融期货交易所、大连期货交易所、郑州期货交易所、上海证券交易所、深圳证券交易所等国内交易所进行场内交易。

TITD接口分CTiTdApi请求接口类、CTiTdSpi回报接口类两类。客户通过创建CTiTdApi对象，发送初始化、客户登录、查询、报单、行权等一系列操作，通过CTiTdSpi的相关函数接收TITD服务端给出的回报、应答。

TITD的运行步骤为：

一、客户登录模块。

此模块为之后其后相关交易的先决条件，需按顺序逐个进行。

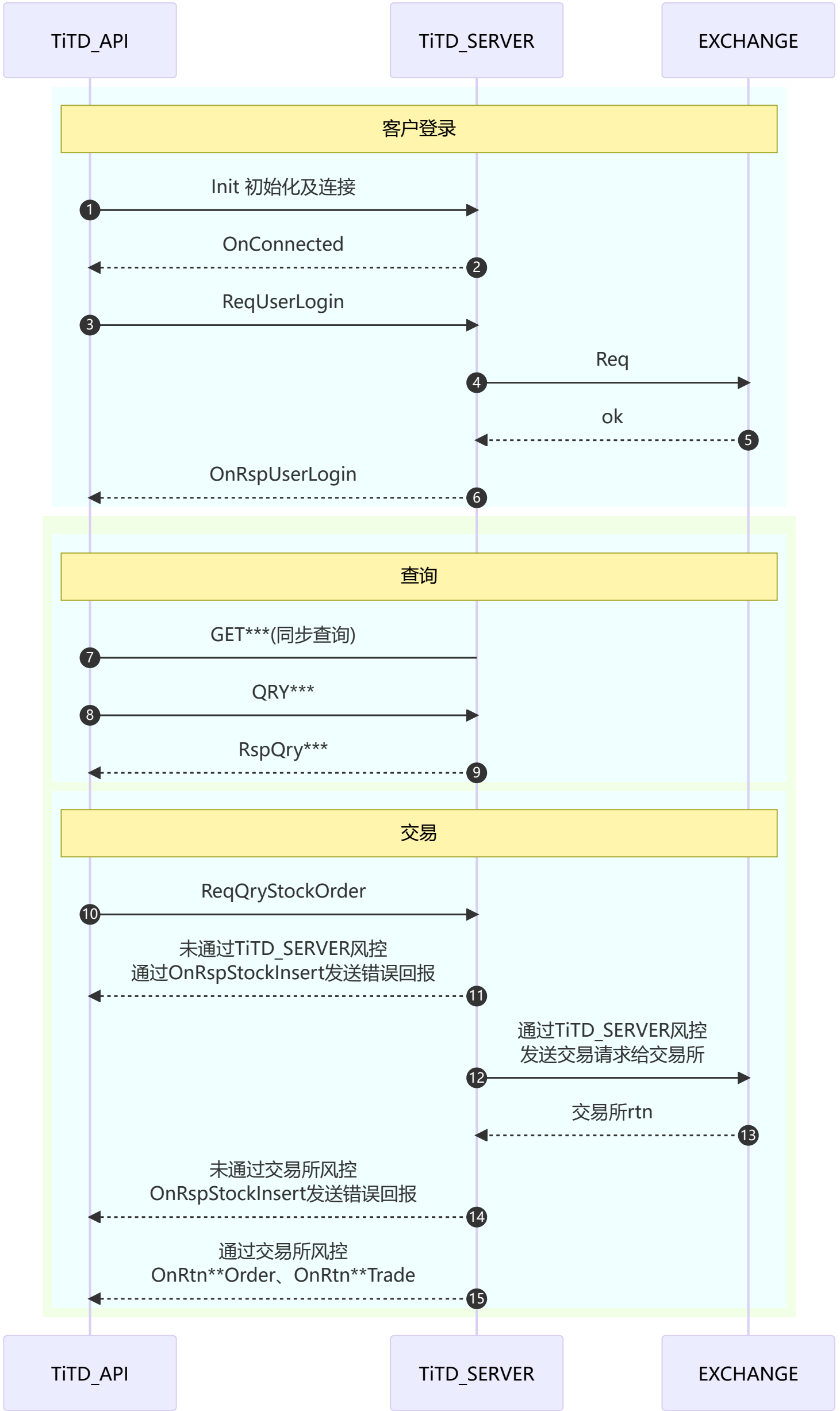
顺序为：

- ①创建api，设置回调函数(CreateTiTdApi、RegisterSpi)
- ②初始化并请求连接(Init);
- ③等待连接成功OnConnected;
- ④请求客户登录ReqUserLogin;
- ⑤等待客户登录成功OnRspUserLogin。

二、交易查询交易模块。

客户登录成功后，可按需要发送各种请求，包括：同步查询，异步查询，报单，报价，行权，申请组合等。

TITD的业务逻辑(部分)如下图所示：



1.4 TITD接口函数汇总

- 以下表格为TITD包含的所有函数，列出了api请求函数，和与之对应的OnRsp开头的应答，与OnRtn为开头的回报。
- OnRtn开头的回报包含：
 - 一、连接回报：OnConnected、OnDisconnected。
 - 二、私有流回报：OnRtnStockOrder、OnRtnStockTrade、OnRtnOptionsOrder、OnRtnOptionsTrade、OnRtnQuoteOrder、OnRtnExercise、OnRtnMarginCombAction、OnRtnBusinessOrder、OnRtnBusinessTrade、OnRtnWithdrawDeposit。
 - 三、公有流通知：OnRtnInstrumentStatus、OnRtnForQuote。

- 关于私有流的一些说明：

私有流是针对单一客户的回报。

TITD登录所需的客户号为UserId, UserId为类似于交易员账号的一个ID,一个UserId下可带多个ClientId（交易编码\股东账号），同一交易编码也可以归属于不同的UserId。用户报单请求、报价请求、询价请求、行权请求以及申请组合时要填的都是ClientId，所有UserId下的ClientId有回报时，都会通过OnRtn***函数收到。OnRsp开头的为应答，只有本次登录后发送的请求才能收到对应的应答。

举例：有两个UserId，A和B，A交易员下带交易编码a和b，B交易员下也带交易编码a。此时A、B交易员都处于登录状态。

当A交易员，发送交易编码ReqOptionsInsert，那么A交易员能收到OnRspOptionsInsert，也能收到OnRtnOptionsOrder、OnRtnOptionsTrade。

B交易员只能收到通知OnRtnOptionsOrder、OnRtnOptionsTrade。

- 关于公有流的一些说明：

公有流是不针对单一客户的回报，柜台发出回报后所有订阅公有流的用户都能收到。

	api格式	对应的spi	备注
api初始化类	CreateTiTdApi		创建api接口
	GetApiVersion		获取api版本号
	Release		删除接口对象本身
	RegisterSpi		注册回调接口
	Init	OnConnected、OnDisconnected	初始化api配置，并连接交易柜台
	~CTiTdApi		
客户登录类	ReqUserLogin	OnRspUserLogin	客户登录
	ReqUserLogout	OnRspUserLogout	客户登出
	ReqUserPasswordUpdate	OnRspUserPasswordUpdate	密码更新
查询类	Get***	为同步查询	此类api查询的数据是从本地数据库获取，登录后当天数据不会改变。
	ReqQry***	OnRspQry***	此类api查询的数据是从服务端获取，盘中会变化
报单类	ReqStockInsert	OnRspStockInsert OnRtnStockOrder OnRtnStockTrade	股票类合约下单
	ReqStockCancel	OnRspStockCancel	股票类合约撤单
	ReqOptionsInsert	OnRspOptionsInsert OnRtnOptionsOrder OnRtnOptionsTrade	期货、期权类合约下单
	ReqOptionsCancel	OnRspOptionsCancel	期货、期权类合约撤单
	ReqBusinessInsert	OnRspBusinessInsert OnRtnBusinessOrder OnRtnBusinessTrade	现货非交易业务报单
	ReqBusinessCancel	OnRspBusinessCancel	现货非交易业务撤单
报价、询价	ReqQuoteInsert	OnRspQuoteInsert OnRtnQuoteOrder	报价
	ReqQuoteCancel	OnRspQuoteCancel	报价撤销
	ReqForQuote	OnRspForQuote	询价（暂不支持）
行权	ReqExercise	OnRspExercise OnRtnExercise	行权
	ReqCombExercise	OnRspCombExercise	组合行权
	ReqExerciseCancel	OnRspExerciseCancel	行权撤销、组合行权撤销
其他	ReqStockLock	OnRspStockLock	锁仓
	ReqMarginCombAction	OnRspMarginCombAction OnRtnMarginCombAction	申请组合、解组合
	ReqUserAuthentication	OnRspUserAuthentication	用户密码验证（现货系统，第三方用户使用）
其他-柜台主动推送流	无请求，服务端主动推送	OnRtnWithdrawDeposit	私有流 - 出入金通知
	无请求，服务端主动推送	OnRtnInstrumentStatus	公有流 - 合约状态变化的通知（目前只推送交易所状态变化通知）
	无请求，服务端主动推送	OnRtnForQuote	公有流 - 询价通知

1.5 TITD通用字段说明

字段	字段名称	类型	说明
UserId	交易用户、登录用户	char[16]	由TITD维护，登录时用该字段，一个UserId可带多个ClientId
ClientId	交易编码\股东代码	char[11]	----
ClientNo	交易编码序号	int	由TITD维护，每个交易日唯一，每天交易前用GetClientNo等获取当日交易编码序号
AccountId	资金账户	char[16]	----
InstrumentId	合约编码	char[31]	----
InstrumentNo	合约序号	int	由TITD维护，每个交易日固定，每天交易前用GetInstrumentNo等获取当日合约序号
ExchangeId	交易所编号	char	固定对应值，请参考头文件
FrontNo	席位序号	char	由TITD维护，而用GetFrontList获取当前可用席位，席位编号1-N顺序
SessionNo	Session号	int	由TITD维护，每次UserLogin后对应不同的SessionNo
RequestID	请求ID	int	由用户自行维护，TITD不做重复性和递增性要求
OrderSysID	交易所报单编号	char[17]	由交易所维护，交易所发送的报单编号，保证了在所有平台内唯一性
SequenceNo	私有流序号	int	由TITD维护，每个交易日连续递增，用户可用该字段判断是否有私有流丢包
LocalOrderNo	会员内部订单编号	int	由用户维护，同一Session内要求递增且不重复。用户也可以不填，直接置0

二、TITD接口详解

2.1 api初始化

相关字段解析请见：[2.1 api初始化相关字段](#)

```
static CTiTdApi* CreateTiTdApi();
static const char* GetApiVersion();           //获取api版本号
virtual void Release() = 0;
virtual void RegisterSpi(CTiTdSpi* pSpi) = 0;
virtual int Init(CTdConfigInfoField& cfgInfo) = 0; //用变量初始化
virtual int Init(const char* cfgFile) = 0;       //用配置文件初始化

virtual void OnConnected() {};
virtual void OnDisconnected(int nReason) {};
```

注意点：

- ① Init为初始化和连接函数，分为用变量初始化和用配置文件初始化，两者所设的参数一模一样，用户只需任选其一，用户调用Init即表示发起与TiTD_SERVER的连接请求。
- ② 与TiTD_SERVER连接成功后接口响应OnConnected，与TiTD_SERVER断开连接会响应OnDisconnected，断开连接后，API会自动重新连接，客户端可不做处理。但是**重连成功后，API需重新发起客户登录请求。**
- ③ GetApiVersion返回api版本号，以库编译时间作为版本标识，如2022-08-03 06:48:26。

参考代码：

```
simplespi spi;           //simplespi为CTiTdSpi子类，类中有接口成员变量CTiTdApi* _api;
spi._api = CTiTdApi::CreateTiTdApi();
spi._api->RegisterSpi(&spi);
spi._api->Init("titdapi.ini");
spi._api->Release();       //使用结束后用release删除接口
```

2.1.1 Init函数1

```
virtual int Init(CTdConfigInfoField& cfgInfo) = 0;
```

当使用结构体初始化时，init函数调用参考如下

```
CTdConfigInfoField config= {0};
strcpy(config.UserId, m_UserID);           // 交易用户代码(该用户名和登录时用的用户名相同)
config.NicType = 1;                         // 网卡类型；目前系统支持以下几种；1:普通socket;2:Solarflare；3:Exablaze；
strcpy(config.SendNicName, "");            // 发送请求的网卡名称(可以与RecvNicName一样)
```

```
strcpy(config.RecvNicName, "");          // 接收应答的网卡名称(可以与SendNicName一样)
strcpy(config.RecvNicIp, "172.18.18.131");      // 接收应答的网卡IP
config.RecvNicPort = 0;                      // 接收应答的网卡端口(请设置成51100-51200之间的端口, 0表示不设置系统会自动分配端口进行数据接收)
config.SubPrivateType = 2;                   // 私有流订阅模式
                                           ///  0:不订阅私有流
                                           ///  1:从本交易日开始重传
                                           ///  2:从上次收到的续传
                                           ///  3:只传送登录后私有流的内容

config.SubPublicType = 0;                   // 公共流订阅模式
                                           ///  0:不订阅公有流
                                           ///  3:订阅公有流

strcpy(config.FrontAddress, "tcp://110.0.0.1:7567");    // 交易服务器网络地址
                                           ///  网络地址的格式为: “protocol://ipaddress:port”,
                                           ///  如: "tcp://127.0.0.1:17001" 或 "udp://127.0.0.1:17001"。
                                           ///  “tcp”代表传输协议, “127.0.0.1”代表服务器地址。"17001”代表服务器端口号。

config.RecvDataCpuId = -1;                  // 接收网络数据的线程cpuid(-1不绑定cpu)
config.DealDataCpuId = -1;                  // 处理网络数据的线程cpuid(-1不绑定cpu)
config.MaxRecvDataSize = 512;               // 接收数据缓存的大小(单位为M,如果不填入默认大小为512M)
config.IsLog = 1;                           // 是否记录日志
config.timeOut = 3;                         // 超时时间默认为3秒
config.LockFree = 1;                        // 兼容2023.12.20前版本配置, 2023.12.20后版本可不填, 都为线程安全模式
                                           // 是否为无锁多线程不安全模式:1:无锁线程不安全; 其他:有锁线程安全

m_pTraderApi->Init(config);
```

配置更改说明:

字段	字段含义	是否需更改	字段说明
UserId	交易用户代码	需更改	用于API自动产生跟用户名相关的三个文件testPrivate.con、testPublic.con、testTraderApiresume.con, 记录交易中需要本地记录的内容。一般设置成跟ReqUserLogin中客户登录的userid一致。
NicType	网卡类型	需更改	使用udp报单模式时, 需填写此项, tcp报单模式时, 不需要填写。
FrontAddress	交易服务器网络地址	必须更改	交易服务器地址, 由经纪商提供。
RecvNicIp	接收应答的网卡IP	udp模式报单时更改。	接收应答的网卡IP
SendNicName	发送请求的网卡名称	udp报单且使用Solarflare或Exablaze网卡发单时必须写网卡名。	如:enp1s0
RecvNicName	接收应答的网卡名称	udp报单且使用Solarflare或Exablaze网卡收回报时必须写网卡名。	如:enp1s0d1

- 除上述需更改的字段外, 其余字段按说明按需更改。
- 若选择记录日志, API将会在运行目录下自动生成log文件夹, 并记录日志名格式为titdapi_yyyy-mm-dd_hh-mm-ss.n.log的日志。
- 公有流有两个: OnRtnForQuote和OnRtnInstrumentStatus, 分别为询价通知和合约状态变化的通知。用户可通过设置选择接收或不接收公有流。
- 表格中所述tcp报单模式为测试时所用, 生产所用的都为udp报单模式, 当选择udp报单模式时, 通过NicType选择使用的网卡类型。

2.1.2 Init函数2

```
virtual int Init(const char* cfgFile) = 0;          //用配置文件初始化
```

ini配置文件参考

```
;基础信息
[BaseInfo]
;交易用户代码(该用户名和登录时用的用户名相同)
UserId=test
;网卡类型; 目前系统支持以下几种; 1:普通socket;2:Solarflare; 3:Exablaze;
NicType=1
;发送请求的网卡名称(可以与RecvNicName一样)
SendNicName=ss
;接收请求的网卡名称(可以与SendNicName一样)
RecvNicName=
;接收应答的网卡IP
RecvNicIp=
;接收应答的网卡端口(请设置成51100-51200之间的端口, 0表示不设置系统会自动分配端口进行数据接收)
RecvNicPort=0
;私有流订阅模式
;0:不订阅私有流
;1:从本交易日开始重传
;2:从上次收到的续传
```



```
;3:只传送登录后私有流的内容
SubPrivateType=1
;公共流订阅模式
;0:不订阅公有流
;3:订阅公有流
SubPublicType=3
;交易服务器网络地址
;网络地址的格式为：“protocol://ipaddress:port”，如: "tcp://127.0.0.1:17001" "udp://127.0.0.1:17001"。
;“tcp”代表传输协议，“127.0.0.1”代表服务器地址。”17001”代表服务器端口号。
FrontAddress=udp://172.18.18.11:4662
;接收网络数据的线程cpuid(-1不绑定cpu)
RecvDataCpuId=1
;处理网络数据的线程cpuid(-1不绑定cpu)
DealDataCpuId=2
;接收数据缓存的大小(单位为M,如果不填入默认大小为512M)
MaxRecvDataSize=512
;是否记录日志:1:记录;0:不记录
IsLog=1
;超时时间默认为3秒
timeOut=3
;LockFree兼容2023.12.20前版本配置，2023.12.20后版本可不填，都为线程安全模式
;是否为无锁多线程不安全模式:1:无锁线程不安全；其他:有锁线程安全
LockFree=1
```

init调用参考如下：

```
m_pTraderApi->Init("./titdapi.ini");
```

注意：

当使用ini文件初始化时，且不需要设置某项配置时，可以参数置空，但请不要删除key。

2.2 客户登录

```
virtual int ReqUserLogin(CTdReqUserLoginField* pReqUserLogin, int nRequestID) = 0;
virtual int ReqUserLogout(int nRequestID) = 0;
virtual int ReqUserPasswordUpdate(CTdUserPasswordUpdateField* pUserPasswordUpdate, int nRequestID) = 0;
```

```
virtual void OnRspUserLogin(CTdRspUserLoginField& pRspUserLogin, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
virtual void OnRspUserLogout(CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
virtual void OnRspUserPasswordUpdate(CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

客户登录成功后，才能做同步查询、异步查询、报单、询报价等操作。

2.2.1 ReqUserLogin

```
virtual int ReqUserLogin(CTdReqUserLoginField* pReqUserLogin, int nRequestID) = 0;
```

```
///用户登录请求
struct CTdReqUserLoginField
{
    TdUserIDType      UserId;           // 交易用户代码
    TdPasswordType     Password;         // 密码
    TdProtocolInfoType ProtocolInfo;     // 协议信息
    TdProductInfoType  UserProductInfo; // 用户端产品信息
    TdAppIdType         AppId;           // appid
    TdAppIdIdentifyType AppIdIdentify;   // appid验证码
};
```

注意点：

① ReqUserLoginUserId为客户交易时真正用到的UserId。Init初始化中的UserId为生成.con文件的UserId。

客户使用时，ReqUserLogin和Init中的UserId建议使用同一个，以防收到的私有流有错误。

② nRequestID由用户维护，TITD柜台不做唯一性、递增性等要求，所有请求对应的响应中，nRequestID按请求中填写的值原样返回，以下所有的nRequestID用法都一样，不再做赘述。

2.2.2 OnRspUserLogin

```
virtual void OnRspUserLogin(CTDRspUserLoginField& prspUserLogin, CTDRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
///用户登录应答
struct CTDRspUserLoginField
{
    TdSessionNoType      SessionNo;          // 本次登录的session
    TdDateType            TradingDay;          // 交易日
    TdIntTimeType         LoginTime;           // 登录成功时间
    TdLocalOrderNoType    MaxLocalOrderNo;     // 最大会员内部订单编号
    TdUserIDType          UserId;              // 交易用户代码
    TdTradingSystemNameType TradingSystemName; // 交易系统名称
    TdSequenceNoType      SequenceNo;          // 当前用户接收的私有流最新序号
    TdDateType            ActionDay;           // 业务发生日期
};
```

- ProtocolInfo、UserProductInfo为用户需要用客户端通过TITD报单时要填的字段。
- **SequenceNo**：返回当前登录的交易用户接收的私有流最新序号，若当天该交易用户还没有收到过私有流，该值为0。
私有流序号**每交易日（TradingDay）**清空，每日从1开始**连续递增**，用户可通过该值是否连续来判断有私有流丢包。
- SessionNo： 每次登录成功即为一个新的session。
- **MaxLocalOrderNo**： 会员内部订单编号LocalOrderNo为整形，客户可填0，表示不维护该字段。若要维护该字段，要求在本session内递增且不重复。MaxLocalOrderNo为本次session的最大会员内部订单编号，要求在此基础上递增。

2.2.3 ReqUserLogout

```
virtual int ReqUserLogout(int nRequestID) = 0;
```

ReqUserLogout为客户登出请求。若用户未调用ReqUserLogout就直接退出程序，在n秒后用户会自动登出。

用户超时时间在Init函数中的timeOut中设置，默认为3秒。

2.2.4 OnRspUserLogout

```
virtual void OnRspUserLogout(CTDRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

OnRspUserLogout为用户登出响应。

2.2.5 ReqUserPasswordUpdate

```
virtual int ReqUserPasswordUpdate(CTDUserPasswordUpdateField* puserPasswordUpdate, int nRequestID) = 0;
```

ReqUserPasswordUpdate为更新交易用户密码请求。

2.2.6 OnRspUserPasswordUpdate

```
virtual void OnRspUserPasswordUpdate(CTDRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

OnRspUserPasswordUpdate为ReqUserPasswordUpdate对应的响应。

2.3 同步查询

同步查询以Get为开头，用户登录时，会初始化一批数据到本地，以Get开头的查询都为查询此类数据，故Get得到的数据，登录后时不会再有变化的。

同步查询函数前面0-n个参数为查询过滤参数，**最后一个引用参数为查询输出参数。**

Get类函数不会对输出参数做初始化，客户在获取各类信息之前，请对要存放输出的变量做初始化，以防获取到错误数据。

同步查询一共有6类，分别为[获取客户信息](#)、[获取交易前置](#)、[获取合约信息](#)、[获取手续费信息](#)、[获取保证金](#)和[获取资金信息](#)。

所有的同步查询查询到的相关数据，都是用户登陆时加载的信息，不会根据用户的交易变化。

2.3.1 查询客户信息

GetClient开头的相关函数，用于获取客户信息，有以下四个函数。

```
virtual int GetClientList(TdRspQryClientList& output) = 0;
virtual int GetClientInfo(char* clientId, CTdRspQryClientField& pClient) = 0;
virtual int GetClientInfo(int clientNo, CTdRspQryClientField& pClient) = 0;
virtual int GetClientNo(char *clientId,int &clientNo) = 0;
```

GetClientList：获取当前USERID所带的所有客户号详情。

GetClientInfo：用clientId交易编码查询特定客户号详情。clientId为必填项。

GetClientNo：只查询clientNo。clientId为必填项。

查询到的客户信息对应的结构体为：

```
//查询交易编码应答
struct CTdRspQryClientField
{
    TdClientNoType      ClientNo;
    TdClientIdType      ClientId;           // 交易编码
    TdClientType         ClientType;        // 交易编码类型
    TdExchangeIdType     ExchangeId;        // 交易所
    TdTradeStatusType    Status;            // 交易编码交易状态
};
```

2.3.2 查询前置

GetFrontList获取交易前置。

```
virtual int GetFrontList(TdRspQryFrontFieldList& output) = 0;
```

GetFrontList：获取所有可用的交易前置列表。

```
//查询前置应答
struct CTdRspQryFrontField
{
    TdFrontNoType        FrontNo;           //席位序号
    TdExchangeIdType     ExchangeId;        //交易所
    TdPartyTypeType      PartyType;         //席位类型
    TdPartyIdType        PartyId;           //席位信息
};
```

2.3.3 查询合约

GetInstrument开头的相关函数，用于获取合约信息。

```
virtual int GetInstrumentList(TdRspQryInstrumentFieldList& output) = 0;
virtual int GetInstrumentInfo(char* instrumentId, CTdRspQryInstrumentField& pInstrument) = 0;
virtual int GetInstrumentInfo(int instrumentNo, CTdRspQryInstrumentField& pInstrument) = 0;
virtual int GetInstrumentNo(char* instrumentId, int &instrumentNo) = 0;
```

GetInstrumentList：获取所有的合约详情列表。

GetInstrumentInfo：根据instrumentId/instrumentNo获取合约详情。instrumentId、instrumentNo为必填项。

GetInstrumentNo：根据instrumentId获取instrumentNo。instrumentId为必填项。

```
///查询合约应答
struct CTdRspQryInstrumentField
{
    TdInstrumentNoType    InstrumentNo;
    TdInstrumentIdType     InstrumentId;     //合约代码
    TdExchangeIdType       ExchangeId;       //交易所代码
    TdInstrumentNameType    InstrumentName;   //合约名称
    TdProductIdType        ProductId;        //产品代码
    TdInstrumentIdType      UnderlyingInstrId; //标的商品代码
    TdProductType          ProductType;      //产品类型
    TdCallorPutType        OptionsType;      //期权类型
    TdIntType              Volumemultiple;   //合约数量乘数
    TdIntType              DeliveryYear;     //交割年份
    TdIntType              DeliveryMonth;    //交割月
    TdIntType              AdvanceMonth;     //提前月份
    TdTradeStatusType      TradeStatus;      //交易权限
    TdDateType             CreateDate;       //创建日
};
```

```
TdDateType      OpenDate;           //上市日
TdDateType      ExpireDate;          //到期日
TdDateType      StartdelivDate;      //开始交割日
TdDateType      EnddelivDate;        //最后交割日
TdPriceType     BasisPrice;          //挂牌基准价
TdPriceType     StrikePrice;         //行权价格
TdVolumeType    MaxMarketOrdervolume; //市价单最大下单量
TdVolumeType    MinMarketOrdervolume; //市价单最小下单量
TdVolumeType    MaxLimitOrdervolume;  //限价单最大下单量
TdVolumeType    MinLimitOrdervolume;  //限价单最小下单量
TdPriceType     PriceTick;           //最小变动价位
//以下为行情信息
TdPriceType     LastPrice;           //最新价
TdPriceType     SettlementPrice;     //昨日结算价
TdPriceType     UpperLimitPrice;     //涨停板价格
TdPriceType     LowerLimitPrice;     //跌停板价格
};
```

2.3.4 查询费率

GetRate开头的相关函数，用于获取手续费率信息。

```
virtual int GetRateList(TdRspQryRateFieldList& output) = 0;
virtual int GetRateInfo(char* clientId, char* instrumentId, CTdRspQryRateField& pRate) = 0;
virtual int GetRateInfo(int clientNo, int instrumentNo, CTdRspQryRateField& pRate) = 0;
```

GetRateList：获取当前USERID所带的所有客户对应的所有合约的手续费率。

GetRateInfo：查询特定clientId、instrumentId对应的合约的手续费率。clientId、instrumentId为必填项。

GetRateInfo：查询特定clientNo、instrumentNo对应的合约的手续费率。clientNo、instrumentNo为必填项。

```
///查询手续费应答
struct CTdRspQryRateField
{
    TdClientNoType    ClientNo;
    TdClientIdType    ClientId;           //交易编码
    TdInstrumentNoType InstrumentNo;
    TdInstrumentIdType InstrumentId;       //合约代码
    TdMoneyType        OpenByMoney;       //开仓手续费率
    TdMoneyType        OpenByVolume;      //开仓手续费
    TdMoneyType        CloseByMoney;      //平仓手续费率
    TdMoneyType        CloseByVolume;     //平仓手续费
    TdMoneyType        CloseTodayByMoney; //平今手续费率
    TdMoneyType        CloseTodayByVolume; //平今手续费
    TdMoneyType        OpenMax;           //开仓单笔最高手续费
    TdMoneyType        OpenMin;           //开仓单笔最低手续费
    TdMoneyType        CloseMax;          //平仓单笔最高手续费
    TdMoneyType        CloseMin;          //平仓单笔最低手续费
};
```

2.3.5 查询期权保证金

GetOptMargin开头的相关函数，用于获取期权保证金率信息。

```
virtual int GetOptMarginList(TdRspQryOptMarginFieldList& output) = 0;
virtual int GetOptMarginInfo(char* clientId, char* instrumentId, CTdRspQryOptMarginField& pMargin) = 0;
virtual int GetOptMarginInfo(int clientNo, int instrumentNo, CTdRspQryOptMarginField& pMargin) = 0;
```

查询保证金区分期货接口和期权接口。

如果查询期货相关品种的保证金，请使用GetFut相关函数。如果查询期权相关品种的保证金，请使用GetOpt相关函数。

Get***MarginList：获取当前登录用户所带的所有客户对应的所有合约的保证金率。

Get***MarginInfo：查询特定clientId、instrumentId对应的合约的保证金率。clientId、instrumentId为必填项。

Get***MarginInfo：查询特定clientNo、instrumentNo对应的合约的保证金率。clientNo、instrumentNo为必填项。

GetOptMarginInfo获取到的期权保证金参数：

当交易的为上交所、深交所股票期权时，Param1，Param2为百分比数，在使用时请乘以0.01。

当交易的为中金所期权时，Param1调整系数，Param2保障系数，如调整系数为12%，保障系数为0.5，收到的Param1为12，Param2为0.5。

```
///期权保证金应答
struct CTdRspQryOptMarginField
{
    TdClientNoType      ClientNo;
    TdClientIdType       ClientId;           //交易编码
    TdInstrumentNoType    InstrumentNo;
    TdInstrumentIdType    InstrumentId;       //合约代码
    TdMoneyType           Param1;            //参数1
    TdMoneyType           Param2;            //参数2
    TdMoneyType           MarginRate;         //在交易所基础上收取的比例
};
```

2.3.6 查询期货保证金

```
virtual int GetFutMarginList(TdRspQryFutMarginFieldList& output) = 0;
virtual int GetFutMarginInfo(char* clientId, char* instrumentId, CTdRspQryFutMarginField& pMargin) = 0;
virtual int GetFutMarginInfo(int clientNo, int instrumentNo, CTdRspQryFutMarginField& pMargin) = 0;
```

```
///期货保证金应答
struct CTdRspQryFutMarginField
{
    TdClientNoType      ClientNo;
    TdClientIdType       ClientId;           //交易编码
    TdInstrumentNoType    InstrumentNo;
    TdInstrumentIdType    InstrumentId;       //合约代码
    TdMoneyType           LongbyMoney;        //多头保证金率
    TdMoneyType           LongbyVolume;       //多头保证金费
    TdMoneyType           ShortbyMoney;       //空头保证金率
    TdMoneyType           ShortbyVolume;      //空头保证金率
};
```

2.3.7 查询初始资金

GetAccount用于获取初始资金信息（用户登录时的资金）。

```
virtual int GetAccount(char* accountId, CTdRspQryAccountField& pRspPartAccount) = 0;
```

accountId：输入。填写要查询的资金账号，必填项。

pRspPartAccount：输出。查询到的资金信息。

同步查询出的资金信息，为用户登录时的初始资金信息，**如果想查询实时资金信息，请用ReqQryAccount。**

```
///资金查询应答
struct CTdRspQryAccountField
{
    TdAccountIDType      AccountId;          /// 资金帐号
    TdMoneyType           Prebalance;         /// 昨权益(为资金账号昨日结算后的权益)
    TdMoneyType           DistribFund;        /// 本系统分配资金(本系统日初初始分配的权益,即日初时的可用+保证金占用)
    TdMoneyType           Balance;            /// 结算准备金
    TdMoneyType           Commi;              /// 手续费
    TdMoneyType           FutMargin;          /// 当前期货保证金总额
    TdMoneyType           OptMargin;          /// 当前期权保证金总额
    TdMoneyType           CombMargin;         /// 当前保证金优惠总额
    TdMoneyType           CloseProfit;        /// 平仓盈亏
    TdMoneyType           PosiProfit;         /// 持仓盈亏(浮动盈亏)
    TdMoneyType           Premium;           /// 期权权利金收支(如果是现货账户则保存开仓时的持仓金额)
    TdMoneyType           Deposit;           /// 入金金额
    TdMoneyType           Withdraw;          /// 出金金额
    TdMoneyType           FrozenMargin;       /// 冻结的保证金
    TdMoneyType           FrozenPremium;     /// 冻结的权利金
    TdMoneyType           FrozenCommi;       /// 冻结的手续费
    TdMoneyType           EntryFees;         /// 当日申报费用
    TdMoneyType           BuyPremium;        /// 权力仓当前占用的权利金(如果是现货账户则表示持仓占用的金额)
};
```

字段与异步查询OnRspQryAccount获取的资金字段一样，相应的资金计算方式请参考异步资金查询中的说明。[异步查询相关字段 - 资金查询](#)。

2.4 异步查询

TITD所有的异步查询请求都以ReqQry开头，查询应答都以OnRspQry开头。异步查询一共有8类，分别为[报单查询](#)、[报价查询](#)、[行权查询](#)、[成交查询](#)、[持仓查询](#)、[资金查询](#)、[行情查询](#)和[通知查询](#)。

所有的异步查询应答，都有四个参数。nRequestID为对应的的查询请求发出的nRequestID，blsLast为true代表最后一条数据，除查询资金外**blsLast为true时第一个参数为空，仅告诉用户此查询应答发送结束。**

所有的异步查询查询到的相关数据，都时根据交易实时变动的。

2.4.1 报单查询

报单查询分Stock接口和Options接口。

Stock接口用于查询股票现货相关的报单。

Options接口用于查询期货交易所的期货、期权，证券交易所的股票期权。

```
virtual int ReqQryStockOrder(CTdQryStockOrderField* pQryOrder, int nRequestID) = 0;
virtual int ReqQryOptionsOrder(CTdQryOptionsOrderField* pQryOrder, int nRequestID) = 0;

virtual void OnRspQryStockOrder(CTdRspQryStockOrderField& pOrder, CTdRspInfoField& pRspInfo, int nRequestID, bool blsLast) {};
virtual void OnRspQryOptionsOrder(CTdRspQryOptionsOrderField& pOrder, CTdRspInfoField& pRspInfo, int nRequestID, bool blsLast) {};
```

1、ReqQryStockOrder

该方法为查询股票现货报单时使用。

```
virtual int ReqQryStockOrder(CTdQryStockOrderField* pQryOrder, int nRequestID) = 0;
```

2、OnRspQryStockOrder

该方法为查询股票现货报单的响应。

```
virtual void OnRspQryStockOrder(CTdRspQryStockOrderField& pOrder, CTdRspInfoField& pRspInfo, int nRequestID, bool blsLast) {};
```

当blsLast为true时，pOrder中字段为空。

3、ReqQryOptionsOrder

该方法为查询期权、期货报单时使用。

```
virtual int ReqQryOptionsOrder(CTdQryOptionsOrderField* pQryOrder, int nRequestID) = 0;
```

```
///期权,期货报单查询请求
struct CTdQryOptionsOrderField
{
    TdClientIDType      ClientID;          /// 所有过滤条件都不填,就查询当前登录的userid下所带客户的所有报单
    TdInstrumentIDType  InstrumentID;      /// [交易编码]
    TdOrderSysIDType    OrderSysID;        /// [合约编码]
    TdIntTimeType       BegTime;           /// [交易所报单编号] 默认值:0或空格表示不过滤报单编号
    TdIntTimeType       EndTime;           /// [起始时间] 0表示不设起始时间. 默认值:0
    TdSideType          Side;              /// [结束时间] 0表示不设结束时间. 默认值:0
    TdOrdStatusType     OrdStatus;         /// [买卖方向]
    TdQryLimitType      LimitSession;      /// [订单状态] 默认值:0表示不过滤订单状态
    TdQryLimitType      LimitUserId;       /// [限制当前session] 默认:不限制
    TdQryLimitType      LimitUserId;       /// [限制当前UserId] 默认:不限制
};
```

注意点:

① 查询报单，哪些是必填字段？

头文件中，查询请求用[]括出的为选填字段。

查询报单中所有的都为选填字段，所有过滤条件都不填,就查询当前登录的userid下所带客户的所有状态的报单。

② 怎样查询所有未成交的单子？

查询所有未成交报单需要发送两遍查询请求，请求的“报单状态”过滤条件中一个填已报入，一个填部分成交。

③ 查询未成交的单子能看出该笔报单的未成交量吗？

看不出。查询未成交得委托，里面是看不出还有多少已成交多少未成交的，Volume为报单时的报单量。

要想知道未成交和成交数量，有两个方法：

方法1，重新收委托回报和成交回报并计算；

方法2，查询这些未成交的委托对应的成交记录。

④“限制当前session”是什么意思？

若“LimitSession”填限制，表示只查询当前此次登录的session下符合条件的报单。

若“LimitSession”填不限制，表示查询所有符合条件的报单。

⑤ “限制当前UserId”是什么意思？

若“LimitUserId”填限制，表示只查询原始报单对应的UserId为当前登录的UserId所下的单子。

比如UserId A, A1都带用交易编码a。若此次登录的为A，且查询报单“LimitUserId”填“限制当前UserId”，则查询报单时通过A1下的属于a的单子，无法被查询到。

4、OnRspQryOptionsOrder

该方法为查询期权、期货报单对应的响应。

```
virtual void OnRspQryOptionsorder(CTdRspQryOptionsOrderField& porder, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///期权,期货报单查询应答
struct CTdRspQryOptionsorderField
{
    TdRequestNoType      RequestNo;           //报单请求编号
    TdClientIdType       ClientId;
    TdInstrumentIdType   InstrumentId;
    TdClientNoType       ClientNo;           //交易编码序号
    TdInstrumentNoType   InstrumentNo;        //合约序号
    TdLocalOrderNoType   LocalOrderNo;       //会员内部订单编号
    TdPriceTypeType      PriceType;          //报单价格条件限价单,事价单等
    TdSideType           Side;               //买卖方向
    TdOffsetFlagType     OffsetFlag;          //开平标记
    TdHedgeFlagType      HedgeFlag;          //投机套保标记
    TdTimeInForceType    TimeInForce;        //订单有效时间类型
    TdCoveredOrUncoveredType CoveredOrUncovered; //备兑标签
    TdVolumeConditionType VolumeCondition;    //成交量类型
    TdTrigConditionType  TrigCondition;       //触发条件
    TdVolumeType          Volume;             //数量
    TdVolumeType          MinVolume;          //最小成交量
    TdPriceType           Price;              //价格
    TdPriceType           StopPrice;          //止损价
    TdOwnerTypeType       OwnerType;          //订单所有类型
    TdExchangeIdType      ExchangeId;        //交易所代
    TdFrontNoType         FrontNo;            //处理本次报单的实际席位序号
    TdSessionNoType       SessionNo;         //发送该报单的session
    TdOrdStatusType       OrdStatus;         //订单状态
    TdIntTimeType         TransTime;         //发生时间
    TdOrderSysIDType      OrderSysId;        //交易所订单编号
    TdUserIDType          UserId;            //原始报单交易用户代码
};
```

当bIsLast为true时，pOrder中字段为空。

2.4.2 报价查询

1、ReqQryQuoteOrder

该方法为查询报价时使用。

```
virtual int ReqQryQuoteOrder(CTdQryQuoteOrderField* pQryQuote, int nRequestID) = 0;
```

```
///报价查询
struct CTdQryQuoteOrderField
{
    // 所有过滤条件都不填,就查询当前登录的userid下所带客户的所有报单
    TdClientIdType      ClientId;           /// 【交易编码】
    TdInstrumentIdType   InstrumentId;       /// 【合约编码】
    TdOrderSysIDType     OrderSysID;        /// 【交易所报单编号】 默认值:0或空格表示不过滤报单编号
    TdOrderSysIDType     BidOrderId;        /// 【交易所报单编号】 默认值:0或空格表示不过滤报单编号
    TdOrderSysIDType     AskOrderId;        /// 【交易所报单编号】 默认值:0或空格表示不过滤报单编号
    TdIntTimeType        BegTime;           /// 【起始时间】      0表示不设起始时间. 默认值:0
    TdIntTimeType        EndTime;          /// 【结束时间】      0表示不设结束时间. 默认值:0
    TdOrdStatusType      BidStatus;         /// 【买方状态】      默认值:0 0表示不过滤订单状态
```

```
TdOrdStatusType      AskStatus;          /// [卖方状态]      默认值:0 0表示不过滤订单状态
TdQryLimitType       LimitSession;       /// [限制当前session] 默认:不限制
TdQryLimitType       LimitUserId;        /// [限制当前UserId]  默认:不限制
};
```

2、OnRspQryQuoteOrder

该方法为查询报价的响应。

```
virtual void OnRspQryQuoteOrder(CTdRspQryQuoteOrderField& pQuote, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///报价查询应答
struct CTdRspQryQuoteOrderField
{
    TdRequestNoType      RequestNo;          ///报单请求编号
    TdClientIdType       ClientId;
    TdInstrumentIdType   InstrumentId;
    TdClientNoType       ClientNo;          ///交易编码序号
    TdInstrumentNoType   InstrumentNo;       ///合约序号
    TdLocalOrderNoType   LocalOrderNo;      ///会员内部订单编号
    TdQuoteReqIDType     QuoteReqId;        ///报价请求ID,预留字段,当报价是对请求的响应时, 填写报价请求的ID
    TdPriceType          BidPx;             ///买报价
    TdPriceType          AskPx;             ///卖报价
    TdVolumeType         BidSize;           ///买数量
    TdVolumeType         AskSize;           ///卖数量
    TdOffsetFlagType     BidOffsetFlag;     ///平仓标识
    TdOffsetFlagType     AskOffsetFlag;     ///平仓标识
    TdOwnerTypeType      OwnerType;         ///订单所有类型
    TdExchangeIdType     ExchangeId;        ///交易所代码
    TdFrontNoType        FrontNo;           ///处理本次报单的实际席位序号
    TdSessionNoType      SessionNo;         ///发送该报单的session
    TdOrderSysIDType     OrderSysId;        ///交易所订单编号
    TdOrderSysIDType     BidOrderId;        ///买方交易所订单编号
    TdOrderSysIDType     AskOrderId;        ///卖方交易所订单编号
    TdOrdStatusType      BidStatus;         ///买方状态
    TdOrdStatusType      AskStatus;         ///卖方状态
    TdIntTimeType        TransTime;         ///发生时间
    TdUserIDType         UserId;            ///原始报单交易用户代码
};
```

当bIsLast为true时, pQuote中字段为空。

2.4.3 行权查询

TITD查询行权分查询普通行权和查询组合行权。

```
virtual int ReqQryExercise(CTdQryExerciseField* pExercise, int nRequestID) = 0;
virtual int ReqQryCombExercise(CTdQryCombExerciseField* pExercise, int nRequestID) = 0;
```

```
virtual void OnRspQryExercise(CTdRspQryExerciseField& pExercise, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
virtual void OnRspQryCombExercise(CTdRspQryCombExerciseField& pExercise, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

1、ReqQryExercise

ReqQryExercise为查询普通行权的请求。对应的应答 为OnRspQryExercise。

```
virtual int ReqQryExercise(CTdQryExerciseField* pExercise, int nRequestID) = 0;
```

```
///行权查询请求
struct CTdQryExerciseField
{
    TdClientIdType       ClientId;          /// [交易编码]
    TdInstrumentIdType   InstrumentId;       /// [合约编码]      默认值:0 0或空表示查询所有合约
    TdOrderSysIDType     OrderSysID;        /// [交易所报单编号]默认值:0或空表示不过滤报单编号
    TdIntTimeType        BegTime;           /// [起始时间]      默认值:0 0表示不设起始时间
    TdIntTimeType        EndTime;           /// [结束时间]      默认值:0 0表示不设结束时间
    TdOrdStatusType      OrdStatus;         /// [订单状态]      默认值:0 0表示不过滤订单状态
};
```


2、OnRspQryExercise

```
virtual void OnRspQryExercise(CTdRspQryExerciseField& pExercise, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
///  
///行权查询应答  
struct CTdRspQryExerciseField  
{  
    TdClientNoType      ClientNo;          ///  
    TdInstrumentNoType   InstrumentNo;       ///  
    TdClientIdType       ClientId;          ///  
    TdInstrumentIdType   InstrumentId;       ///  
    TdLocalOrderNoType   LocalOrderNo;      ///  
    TdVolumeType         Volume;           ///  
    TdOwnerTypeType      OwnerType;         ///  
    TdExchangeIdType     ExchangeId;        ///  
    TdFrontNoType        FrontNo;           ///  
    TdSessionNoType      SessionNo;         ///  
    TdOrdStatusType      OrdStatus;         ///  
    TdIntTimeType        TransTime;         ///  
    TdOrderSysIDType     OrderSysId;        ///  
    TdUserIDType         UserId;            ///  
};
```

当bIsLast为true时，pExercise中字段为空。

3、ReqQryCombExercise

ReqQryCombExercise为查询组合行权的请求。对应的应答 为OnRspQryCombExercise。

```
virtual int ReqQryCombExercise(CTdQryCombExerciseField* pExercise, int nRequestID) = 0;
```

```
///  
///组合行权查询请求  
struct CTdQryCombExerciseField  
{  
    TdClientIdType      ClientId;          ///  
    TdInstrumentIdType   InstrumentId1;     ///  
    TdInstrumentIdType   InstrumentId2;     ///  
    TdOrderSysIDType     OrderSysID;        ///  
    TdIntTimeType        BegTime;           ///  
    TdIntTimeType        EndTime;           ///  
    TdOrdStatusType      OrdStatus;         ///  
};
```

4、OnRspQryCombExercise

该方法为查询组合行权对应的响应

```
virtual void OnRspQryCombExercise(CTdRspQryCombExerciseField& pExercise, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
///  
///组合行权查询应答  
struct CTdRspQryCombExerciseField  
{  
    TdClientNoType      ClientNo;          ///  
    TdClientIdType       ClientId;          ///  
    TdLocalOrderNoType   LocalOrderNo;      ///  
    TdVolumeType         Volume;           ///  
    TdInstrumentNoType    LegInstrumentNo1;  ///  
    TdInstrumentIdType   InstrumentId1;     ///  
    TdVolumeType         LegVolume1;        ///  
    TdInstrumentNoType    LegInstrumentNo2;  ///  
    TdInstrumentIdType   InstrumentId2;     ///  
    TdVolumeType         LegVolume2;        ///  
    TdOwnerTypeType      OwnerType;         ///  
    TdExchangeIdType     ExchangeId;        ///  
    TdFrontNoType        FrontNo;           ///  
    TdSessionNoType      SessionNo;         ///  
    TdOrdStatusType      OrdStatus;         ///  
    TdIntTimeType        TransTime;         ///  
    TdOrderSysIDType     OrderSysId;        ///  
    TdUserIDType         UserId;            ///  
};
```

当bisLast为true时， pExercise中字段为空。

2.4.4 成交查询

成交查询分Stock接口和Options接口。

Stock接口用于查询股票现货相关的成交单。

Options接口用于查询期货交易所的期货、期权，证券交易所的股票期权的成交单。

```
virtual int ReqQryStockTrade(CTdQryStockTradeField* pQryTrade, int nRequestID) = 0;
virtual int ReqQryOptionsTrade(CTdQryOptionsTradeField* pQryTrade, int nRequestID) = 0;

virtual void OnRspQryStockTrade(CTdRspQryStockTradeField& pTrade, CTdRspInfoField& prspInfo, int nRequestID, bool bisLast) {};
virtual void OnRspQryOptionsTrade(CTdRspQryOptionsTradeField& pTrade, CTdRspInfoField& prspInfo, int nRequestID, bool bisLast) {};
```

1、ReqQryStockTrade

```
virtual int ReqQryStockTrade(CTdQryStockTradeField* pQryTrade, int nRequestID) = 0;

///股票成交查询
struct CTdQryStockTradeField
{
    TdClientIdType      ClientId;          /// 所有过滤条件都不填,就查询当前登录的userid下所带客户的所有成交
    TdInstrumentIdType   InstrumentId;      /// [交易编码]
    TdOrderSysIDType     OrderSysID;        /// [交易所报单编号] 默认值:0或空表示不过滤报单编号
    TdTradeIDType        TradeId;          /// [成交编号]
    TdSequenceNoType     BegSeqNo;          /// [起始回报流序号] 默认值:0 0表示不设起始要求
    TdSequenceNoType     EndSeqNo;          /// [结束回报流序号] 默认值:0 0表示不设结束要求
    TdIntTimeType        BegTime;           /// [起始时间] 默认值:0 0表示不设起始时间
    TdIntTimeType        EndTime;           /// [结束时间] 默认值:0 0表示不设结束时间
    TdSideType           Side;              /// [买卖方向]
    TdQryLimitType       LimitSession;      /// [限制当前session] 默认值:0 0表示不限制
    TdQryLimitType       LimitUserId;       /// [限制当前UserId] 默认值:0 0表示不限制
};
```

2、OnRspQryStockTrade

```
virtual void OnRspQryStockTrade(CTdRspQryStockTradeField& pTrade, CTdRspInfoField& prspInfo, int nRequestID, bool bisLast) {};
```

```
///股票成交查询应答
struct CTdRspQryStockTradeField
{
    TdFrontNoType        FrontNo;           //处理本次报单的实际席位序号
    TdSequenceNoType     SequenceNo;        //回报序号(私有流序号每个交易日连续)
    TdRequestNoType       RequestNo;        //报单请求编号
    TdSessionNoType       SessionNo;        //原始报单session
    TdClientNoType        ClientNo;
    TdInstrumentNoType    InstrumentNo;
    TdLocalOrderNoType    LocalOrderNo;
    TdClientIdType        ClientId;
    TdInstrumentIdType     InstrumentId;
    TdExchangeIdType      ExchangeId;       //交易所代码
    TdSideType            Side;              //买卖方向
    TdPriceType           TradePrice;       //成交价格
    TdVolumeType          TradeVolume;      //成交数量
    TdVolumeType          LeavesVolume;     //本次成交后申报余额数量
    TdIntTimeType         TradeTime;        //成交时间
    TdOrderSysIDType      OrderSysId;       //交易所报单编号
    TdTradeIDType         TradeId;          //成交编号
    TdUserIDType          UserId;           //原始报单交易用户代码
};
```

3、ReqQryOptionsTrade

```
virtual int ReqQryOptionsTrade(CTdQryOptionsTradeField* pQryTrade, int nRequestID) = 0;
```

```
///期货、期权成交查询请求
struct CTdQryOptionsTradeField
{
    // 所有过滤条件都不填,就查询当前登录的userid下所带客户的所有成交
```

```
TdClientIdType      ClientId;          /// [交易编码]
TdInstrumentIdType   InstrumentId;      /// [合约编码]
TdOrdersSysIDType    OrdersSysID;      /// [交易所报单编号]默认值:0或空格表示不过滤报单编号
TdTradeIDType        TradeId;          /// [成交编号]
TdSequenceNoType     BegSeqNo;         /// [起始回报流序号] 0表示不设起始要求 默认值:0
TdSequenceNoType     EndSeqNo;         /// [结束回报流序号] 0表示不设结束要求 默认值:0
TdIntTimeType        BegTime;          /// [起始时间] 0表示不设起始时间. 默认值:0
TdIntTimeType        EndTime;          /// [结束时间] 0表示不设结束时间. 默认值:0
TdSideType           Side;             /// [买卖方向]
TdQryLimitType       LimitSession;     /// [限制当前session] 默认:不限制
TdQryLimitType       LimitUserId;      /// [限制当前UserId] 默认:不限制

};
```

4、OnRspQryOptionsTrade

```
virtual void OnRspQryOptionsTrade(CTdRspQryOptionsTradeField& pTrade, CTdRspInfoField& prspInfo, int nRequestID, bool bisLast) {};
```

```
///期货、期权成交查询应答
struct CTdRspQryOptionsTradeField
{
    TdFrontNoType      FrontNo;
    TdSequenceNoType   SequenceNo;      ///回报序号
    TdRequestNoType     RequestNo;       ///报单请求编号
    TdLocalOrderNoType  LocalOrderNo;
    TdSessionNoType     SessionNo;       ///原始报单session
    TdClientNoType      ClientNo;
    TdInstrumentNoType  InstrumentNo;
    TdClientIdType      ClientId;
    TdInstrumentIdType  InstrumentId;
    TdExchangeIdType    ExchangeId;      ///交易所代码
    TdPriceTypeType     PriceType;        ///报单价格条件限价单市价单等
    TdSideType          Side;            ///买卖方向
    TdOffsetFlagType    OffsetFlag;       ///开平标记
    TdHedgeFlagType     HedgeFlag;        ///投机套保标记
    TdTimeInForceType   TimeInForce;      ///有效期类型
    TdCoveredOrUncoveredType CoveredOrUncovered; ///备兑标签
    TdVolumeConditionType VolumeCondition; ///成交量类型
    TdTrigConditionType TrigCondition;    ///触发条件
    TdOwnerTypeType     OwnerType;        ///订单所有类型
    TdPriceType          TradePrice;      ///成交价格
    TdVolumeType         TradeVolume;     ///成交数量
    TdVolumeType         LeavesVolume;    ///本次成交后申报余额数量
    TdIntTimeType        TransTime;       ///成交时间
    TdOrdersSysIDType    OrdersSysId;     ///交易所报单编号
    TdTradeIDType        TradeId;         ///成交编号
    TdUserIDType         UserId;          ///原始报单交易用户代码
};
```

当blsLast为true时，pTrade中字段为空。

2.4.5 持仓查询

持仓查询三种。

ReqQryStcokPosition：用于查询股票现货的持仓查询。

ReqQryOptionsPosition：用于查询股期权、期货的持仓查询。

ReqQryCombPosition：用于被组合的持仓。在查询普通持仓时，也会体现被组合的合约的持仓。

```
virtual int ReqQryStcokPosition(CTdQryStockPositionField* pQryPosition, int nRequestID) = 0;
virtual int ReqQryOptionsPosition(CTdRspQryOptionsPositionField* pQryPosition, int nRequestID) = 0;
virtual int ReqQryCombPosition(CTdQryCombPositionField* pQryPosi, int nRequestID) = 0;
```

```
virtual void OnRspQryOptionsPosition(CTdRspQryOptionsPositionField& pPosition, CTdRspInfoField& prspInfo, int nRequestID, bool bisLast) {};
```

```
virtual void OnRspQryStockPosition(CTdRspQryStockPositionField& pPosition, CTdRspInfoField& prspInfo, int nRequestID, bool bisLast) {};
```

```
virtual void OnRspQrySseCombPosition(CTdRspQryCombPositionField& pPosi, CTdRspInfoField& prspInfo, int nRequestID, bool bisLast) {};
```

1、ReqQryStockPosition

ReqQryStockPosition为现货持仓查询。

- 当连接现货系统时，该接口正常查询现货（即股票持仓、ETF持仓、债券持仓等）持仓。

- 当连接期权系统(sse)时，该接口可查询现货锁仓量，即期权系统中通过ReqStockLock锁仓，ReqOptionsInsert备兑开仓后，可通过ReqQryStockPosition当前已锁定的现货量、当前已被备兑开仓占用的现货量。

此时通过该接口查询到的现货持仓，不是客户真正持有的现货持仓，只表示已锁定的现货量。

具体查询到OnRspQryStockPosition中的锁仓量等字段说明，详见ReqStockLock。

```
virtual int ReqQryStockPosition(CTdQryStockPositionField* pQryPosition, int nRequestID) = 0;
```

```
//股票持仓查询请求
struct CTdQryStockPositionField
{
    TdClientIdType      ClientId;          ///<交易编码>    必填项:不填返回错误提醒
    TdInstrumentIdType  InstrumentId;      ///<合约编码>    默认值:0 0或空表示查询所有合约
};
```

2、OnRspQryStockPosition

该方法为ReqQryStockPosition查询现货持仓对应的应答。当blsLast为true时，pPosition中字段为空。

```
virtual void OnRspQryStockPosition(CTdRspQryStockPositionField& pPosition, CTdRspInfoField& prspInfo, int nRequestID, bool blsLast) {};
```

```
//股票持仓查询应答
struct CTdRspQryStockPositionField
{
    TdClientNoType      ClientNo;
    TdInstrumentNoType  InstrumentNo;
    TdClientIdType      ClientId;
    TdInstrumentIdType  InstrumentId;
    TdExchangeIdType    ExchangeId;        ///<交易所代码>
    TdVolumeType        YdPosition;        ///<上日持仓>
    TdVolumeType        Position;          ///<总持仓>
    TdVolumeType        TodayPosition;     ///<今日持仓>
    TdVolumeType        FrozenVolume;      ///<冻结数量>
    TdVolumeType        BuyTradeVolume;    ///<当日买成交量>
    TdVolumeType        SellTradeVolume;   ///<当日卖成交量>
    TdPriceType          Price;            ///<持仓价格>
    TdVolumeType        LockVolume;        ///<锁仓数量(已经被锁定的可用于备兑开仓的数量)>
    TdVolumeType        FrozenLock;        ///<锁仓冻结数量(解锁时冻结的数量)>
};
```

各字段解析如下：

YdPosition： 昨日结算时仓位。

Position： 当前持仓数量。

TodayPosition： 当前持仓中属于今仓的。

FrozenVolume： 被冻结的仓位。

BuyTradeVolume： 当日买入成交数量。

SellTradeVolume： 当日卖出成交数量。

当OnRspQryStockPosition是做为“连接期权系统时查询现货锁仓量”时，各字段解析请参考ReqStockLock中的说明。

3、ReqQryOptionsPosition

```
///  
virtual int ReqQryOptionsPosition(CTdQryOptionsPositionField* pQryPosition, int nRequestID) = 0;
```

```
///  
struct CTdQryOptionsPositionField
{
    TdClientIdType      ClientId;          ///<交易编码>    必填项:不填返回错误提醒
    TdInstrumentIdType  InstrumentId;      ///<合约编码>    默认值:0 0或空表示查询所有合约
    TdSideType          Side;              ///<买卖方向>    默认值:0 0表示不限制买卖方向
    TdCoveredOrUncoveredType CoveredOrUn;  ///<备兑标记>    默认值:0 0表示不限制备兑标记
};
```

4、OnRspQryOptionsPosition

该方法为查询期权持仓（期货系统为期货持仓） 应答。当blsLast为true时， pTrade中字段为空。

```
///期权, 期货持仓查询应答
virtual void OnRspQryOptionsPosition(CTdRspQryOptionsPositionField& pPosition, CTdRspInfoField& pRspInfo, int nRequestID, bool blsLast) {};
```

```
//////期货、期权持仓查询应答
struct CTdRspQryOptionsPositionField
{
    TdClientNoType      ClientNo;
    TdInstrumentNoType  InstrumentNo;
    TdClientIdType      ClientId;
    TdInstrumentIdType  InstrumentId;
    TdExchangeIdType    ExchangeId;          /// 交易所代码
    TdCoveredOrUncoveredType  Converd;        /// 备兑标记
    TdSideType          Side;                 /// 买卖方向
    TdVolumeType        YdPosition;           /// 上日持仓
    TdVolumeType        Position;             /// 总持仓
    TdVolumeType        TodayPosition;        /// 今日持仓
    TdVolumeType        FrozenVolume;         /// 冻结数量
    TdVolumeType        BuyTradeVolume;       /// 当日买成交量
    TdVolumeType        SellTradeVolume;      /// 当日卖成交量
    TdPriceType         Price;                /// 持仓价格
    TdMoneyType         UseMargin;            /// 占用的保证金(权利金)
};
```

注意点:

① 已被组合的持仓，在查询持仓中还能查出吗？

能，已经申请组合的单腿，在用ReqQryOptionsPosition查询持仓中，仍旧**能查出**这些持仓，对被已申请组合持仓单腿数在查询持仓应答中的FrozenVolume字段中标识。

② 查到某些合约显示的持仓为0时怎么回事？

只要合约在该方向上当日曾有过持仓(有历史仓，或者今天做过交易)，那么不管该合约在此方向上的仓位有没有被清空，查询都是能查到的。

③ 各查询到的字段能做下详细解释吗？

YdPosition: 上日持仓。此字段为上一交易日结算后，该合约的持仓。

Position: 总持仓。 此字段为该合约当前真实的仓位。

TodayPosition: 今日持仓。 此字段为该合约当天新开仓的持仓数。

FrozenVolume: 冻结数量。此字段为已被组合的合约单腿数量。

为方便理解，以下为持仓数量的理解：

合约A为期权，昨日结算后有10手权利仓。 以下都为合约A权力仓持仓。

step1：未做任何操作时， 查询结果为：YdPosition=10， Position=10， TodayPosition=0， BuyTradeVolume=0， SellTradeVolume=0。

step2：买开5手合约A， 查询结果为：YdPosition=10， Position=15， TodayPosition=5， BuyTradeVolume=5， SellTradeVolume=0。

step3：卖开5手合约A， 查询结果同step2 。此时合约持仓的变化体现在合约A的义务仓持仓里。

step4：卖平5手合约A， 查询结果为：YdPosition=10， Position=10， TodayPosition=5， BuyTradeVolume=5， SellTradeVolume=5。

step5：卖平10手合约A， 查询结果为：YdPosition=10， Position=0， TodayPosition=0， BuyTradeVolume=5， SellTradeVolume=15。

Price: 为开仓价的加权平均，若该仓位有平仓，不改变此价格。

5、ReqQryCombPosition

```
virtual int ReqQryCombPosition(CTdQryCombPositionField* pQryPosi, int nRequestID) = 0;
```

```
//组合持仓查询请求
struct CTdQryCombPositionField
{
    TdClientIdType      ClientId;          /// <交易编码> 必填项:不填返回错误提醒
    TdCombIDType        CombID;            /// [组合类型] 默认值:0 0或空格表示查询所有组合类型， 目前为7种组合策略：CNSJC、PXSJC、PNSJC、CXSJC、KS、KKS、ZBD
    TdOrdersSysIDType   CombInstId;        /// [组合编码] 默认值:0 0或空格表示查询所有组合持仓
};
```


6、OnRspQrySseCombPosition

该方法为查询组合持仓应答。

当blsLast为true时，pPosi中字段为空。

```
virtual void OnRspQryCombPosition(CTdRspQryCombPositionField& pPosi, CTdRspInfoField& prspInfo, int nRequestID, bool blsLast) {};
```

```
//组合持仓查询应答
struct CTdRspQryCombPositionField
{
    TdClientNoType      ClientNo;
    TdClientIdType      ClientId;
    TdExchangeIdType    ExchangeId;          /// 交易所代码
    TdInstrumentIdType   CombId;              /// 组合策略编码:目前为7种组合策略: CNSJC、PXSJC、PNSJC、CXSJC、KS、KKS、ZBD
    TdOrderSysIdType     CombInstId;          /// 组合编码。组合申报时，该字段为空格；拆分申报时，填写拟拆分组合的组合编码
    TdVolumeType         Volume;              /// 持仓数量
    TdVolumeType         FrozenVolume;        /// 冻结数量
    TdPriceType          ComMargin;           /// 优惠的保证金
    int NoLeges;          /// 成分合约数，取值不超过4，后接重复组
    OmItemRtn item[4];
};
```

注意点:

- ① 查询到的组合持仓，腿1，腿2可能跟CTP会不一样，详细区别请参考本文档2.9.1申请组合。

2.4.6 资金查询

查询资金相关的结构体字段解析请参考：[2.4 异步查询相关字段-查询资金字段](#)

```
virtual int ReqQryAccount(CTdQryAccountField* pQryPartAccount, int nRequestID) = 0;
```

```
virtual void OnRspQryAccount(CTdRspQryAccountField& pAccount, CTdRspInfoField& prspInfo, int nRequestID, bool blsLast) {};
```

1、ReqQryAccount

```
virtual int ReqQryAccount(CTdQryAccountField* pQryPartAccount, int nRequestID) = 0;
```

```
///资金查询
struct CTdQryAccountField
{
    TdAccountIDType      AccountId;          /// <资金帐号>      必填项:不填返回错误提醒
};
```

2、OnRspQryAccount

该方法为查询客户资金应答。当blsLast为true时，pAccount中字段有值。

```
virtual void OnRspQryAccount(CTdRspQryAccountField& pAccount, CTdRspInfoField& prspInfo, int nRequestID, bool blsLast) {};
```

```
///资金查询应答
struct CTdRspQryAccountField
{
    TdAccountIDType      AccountId;          /// 资金帐号
    TdMoneyType          Prebalance;         /// 昨权益(为资金账号昨日结算后的权益)
    TdMoneyType          DistribFund;        /// 本系统分配资金(本系统日初初始分配的权益,即日初时的可用+保证金占用)
    TdMoneyType          Balance;            /// 结算准备金
    TdMoneyType          Commi;               /// 手续费
    TdMoneyType          FutMargin;           /// 当前期货保证金总额
    TdMoneyType          OptMargin;           /// 当前期权保证金总额
    TdMoneyType          CombMargin;          /// 当前保证金优惠总额
    TdMoneyType          CloseProfit;         /// 平仓盈亏
    TdMoneyType          PosiProfit;          /// 持仓盈亏(浮动盈亏)
    TdMoneyType          Premium;             /// 期权权利金收支(如果是现货账户则保存开仓时的持仓金额)
    TdMoneyType          Deposit;             /// 入金金额
    TdMoneyType          Withdraw;            /// 出金金额
    TdMoneyType          FrozenMargin;         /// 冻结的保证金
    TdMoneyType          FrozenPremium;       /// 冻结的权利金
    TdMoneyType          FrozenCommi;         /// 冻结的手续费
    TdMoneyType          EntryFees;           /// 当日申报费用
    TdMoneyType          BuyPremium;          /// 权力仓当前占用的权利金(如果是现货账户则表示持仓占用的金额)
```

```
};
```

当连接的系统为期权、期货系统时，相关字段说明如下：

- 客户当前可用资金（Balance）= 本系统分配资金 - 手续费 - 当前期货保证金总额 - 当前期权保证金总额 + 当前保证金优惠总额 + 平仓盈亏 + 权利金收支 + 入金 - 出金 - 冻结保证金 - 冻结权利金 - 冻结手续费 - 当日申报费用 - 浮动亏损
- 当日申报费用（EntryFees）期货交易系统中报单收取的申报费。
- API中浮动盈亏字段为PosiProfit，当为浮动亏损时，PosiProfit为负值。期权系统中，浮动盈亏不做计算。
- 期权的保证金计算时，取的期权和标的价可以在交易系统中设置，如果取值用到最新价OptMargin会随着价格变动而变动，查询持仓中的期权保证金也会变动。具体期权和标的价可详询经济商。
- 期权**冻结的保证金**取报单请求发出时行情计算出的保证金，冻结的保证金**不**随合约价格变动而变动。
- **权力仓当前占用的权利金(BuyPremium)** 期权账户对持仓占用权利金的计算采用先开先平的方式计算当前持仓占用的权利金,按开仓金额计算，不等同于权力仓市值。

当连接的系统为现货系统时，相关字段说明如下：

- Balance字段为可用资金。
- 可用资金(Balance) = 本系统分配资金(DistribFund) - 手续费(Commi) + 资金收支(Premium) + 入金(Deposit) - 出金(Withdraw) - 冻结的保证金(FrozenMargin) - 冻结的手续费(FrozenCommi)
- DistribFund：本系统分配资金为日初可用资金。
- PosiProfit：titd不记录浮动盈亏
- CloseProfit：titd记录客户平仓盈亏
- FrozenMargin：现货系统中，非交易业务中的国债逆回购业务所占用的资金，记录在FrozenMargin。
- FrozenCommi：记录冻结的保证金，包含在途非交易业务的保证金。
- 现货系统中查询资金是否有总权益？
现货系统“查询资金”方法没有总权益项，若客户需要计算总权益，可首先按持仓计算出持仓市值。再用公式“总权益 = 本系统分配资金(DistribFund) + 入金(Deposit) - 出金(Withdraw) + 资金收支(Premium) - 手续费(Commi) + 持仓市值(按持仓计算)”计算。

2.4.7 行情查询

```
virtual int ReqQryMarketData(CTdQryMarketDataField* pQryMarketData, int nRequestID) = 0;
```

```
virtual void OnRspQryMarketData(CTdRspQryMarketDataField& pMarketData, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast);
```

1、ReqQryMarketData

```
virtual int ReqQryMarketData(CTdQryMarketDataField* pQryMarketData, int nRequestID) = 0;
```

```
///查询行情请求
struct CTdQryMarketDataField
{
    TdInstrumentNoType    InstrumentNo;    //【合约No】、【合约代码】任选其一，两者都填，以合约No为准
    TdInstrumentIdType    InstrumentId;    //【合约No】
    TdInstrumentIdType    InstrumentId;    //【合约代码】
};
```

2、OnRspQryMarketData

```
virtual void OnRspQryMarketData(CTdRspQryMarketDataField& pMarketData, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast);
```

```
///查询行情应答
struct CTdRspQryMarketDataField
{
    TdInstrumentNoType    InstrumentNo;    //合约No
    TdInstrumentIdType    InstrumentId;    //合约代码
    TdPriceType           LastPrice;      //最新价
    TdPriceType           SettlementPrice; //昨日结算价
    TdPriceType           UpperlimitPrice; //涨停板价格
    TdPriceType           LowerlimitPrice; //跌停板价格
};
```

注意：
LastPrice为最新价，该值会实时更新。

2.4.8 回报查询

```
virtual int ReqQryNotify(CTdQryNotifyField* pQryNotify, int nRequestID) = 0;
```

```
virtual void OnRspQryNotify(CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast);
```

1、ReqQryNotify

ReqQryNotify为查询回报请求，查询后api会收到OnRspQryNotify来表示查询回报请求是否成功发送。

查询回报成功发送后，api会重新收到查询的所有回报。

如查询时BegSeqNo和EndSeqNo都设0，那么api将重新收到当日所有的私有六推送。

```
virtual int ReqQryNotify(CTdQryNotifyField* pQryNotify, int nRequestID) = 0;
```

查询回报发出后，将重新推送要查询的回报。

```
///查询回报请求
struct CTdQryNotifyField
{
    TdSequenceNoType    BegSeqNo;          /// 【起始回报流序号】 0表示不设起始要求 默认值:0
    TdSequenceNoType    EndSeqNo;          /// 【结束回报流序号】 0表示不设结束要求 默认值:0
};
```

BegSeqNo为0，表示查询EndSeqNo之前所有的私有流。

EndSeqNo为0，表示查询BegSeqNo之后所有的私有流。

BegSeqNo、EndSeqNo都为0，表示查询当日所有私有流。

2、OnRspQryNotify

```
virtual void OnRspQryNotify(CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast);
```

OnRspQryNotify仅表示查询回报请求是否成功发送。

2.4.9 股票非交易业务委托查询

```
///股票非交易业务报单查询请求
virtual int ReqQryBusinessOrder(CTdQryBusinessOrderField* pQryOrder, int nRequestID) = 0;
```

```
///股票非交易报单查询应答
virtual void OnRspQryBusinessOrder(CTdRspQryBusinessOrderField& pOrder, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

1、ReqQryBusinessOrder

该方法为股票非交易业务委托查询请求。

```
///股票非交易业务报单查询请求
virtual int ReqQryBusinessOrder(CTdQryBusinessOrderField* pQryOrder, int nRequestID) = 0;
```

```
///股票非交易业务报单查询请求
struct CTdQryBusinessOrderField
{
    TdClientIDType      ClientId;          /// 所有过滤条件都不填,就查询当前登录的userid下所带客户的所有报单
    TdInstrumentIDType  InstrumentId;      /// 【交易编码】
    TdOrderSysIDType    OrderSysID;        /// 【合约编码】
    TdIntTimeType       BegTime;           /// 【交易所报单编号】 默认值:0或空 表示不过滤报单编号
    TdIntTimeType       EndTime;           /// 【起始时间】      默认值:0 0表示不设起始时间
    TdSideType          Side;              /// 【结束时间】      默认值:0 0表示不设结束时间
    TdOrdStatusType     OrdStatus;         /// 【买卖方向】
    TdQryLimitType      LimitSession;      /// 【订单状态】      默认值:0 表示不过滤订单状态
    TdQryLimitType      LimitUserId;       /// 【限制当前session】 默认值:0 0表示不限制
};
```

2、OnRspQryBusinessOrder

该方法为股票非交易业务委托查询应答。

```
///股票非交易报单查询应答
virtual void OnRspQryBusinessOrder(CTdRspQryBusinessOrderField& porder, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
///股票非交易业务报单查询应答
struct CTdRspQryBusinessOrderField
{
    TdBusinessType      BusinessType;           //业务类型
    TdRequestNoType     RequestNo;              //报单请求编号
    TdClientIdType      ClientId;
    TdInstrumentIdType  InstrumentId;
    TdClientNoType      ClientNo;              //交易编码序号
    TdLocalOrderNoType  LocalOrderNo;          //会员内部订单编号
    //TdPriceTypeType    PriceType;            //报单价格条件
    TdSideType          Side;                  //买卖方向
    //TdTimeInForceType  TimeInForce;          //订单有效时间类型
    TdVolumeType        Volume;                //数量
    TdPriceType          Price;                //价格
    //TdOwnerTypeType    OwnerType;            //订单所有类型
    TdExchangeIdType    ExchangeId;            //交易所代码
    TdFrontNoType       FrontNo;                //处理本次报单的实际席位序号
    TdSessionNoType     SessionNo;             //发送该报单的session
    TdOrdStatusType     OrdStatus;              //订单状态
    TdIntTimeType        TransTime;             //发生时间
    TdOrdersSysIDType    OrdersSysId;           //交易所订单编号
    TdUserIDType         UserId;                //原始报单交易用户代码
    TdInstrumentIdType   DestInstrumentId;      //目标合约代码
};
```

2.5 报单

TITD报单、撤单区分Stock接口，Options接口。

Stock接口，用于上交所、深交所股票现货报单；

Options接口，用于四大期货交易所的期货、期权品种，以及上交所、深交所期权品种。

```
virtual int ReqStockInsert(CTdStockInsertReqField* pStockInsert, int nRequestID) = 0;
virtual int ReqStockCancel(CTdOrderCancelReqField* pStockCancel, int nRequestID) = 0;
virtual int ReqOptionsInsert(CTdOrderInsertReqField* pOptionsInsert, int nRequestID) = 0;
virtual int ReqOptionsCancel(CTdOrderCancelReqField* pOptionsCancel, int nRequestID) = 0;
```

```
virtual void OnRspStockInsert(CTdStockInsertRspField& pStockInsert, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
virtual void OnRspStockCancel(CTdOrderCancelRspField& pStockCancel, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
virtual void OnRspOptionsInsert(CTdOrderInsertRspField& pOptionsInsert, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
virtual void OnRspOptionsCancel(CTdOrderCancelRspField& pOptionsCancel, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
virtual void OnRtnStockOrder(CTdStockOrderRtnField& pOrder) {};
```

```
virtual void OnRtnOptionsOrder(CTdOptionsOrderRtnField& pOrder) {};
```

```
virtual void OnRtnOptionsTrade(CTdOptionsTraderRtnField& pTrade) {};
```

```
virtual void OnRtnStockTrade(CTdStockTradeRtnField& pTrade) {};
```

2.5.1 报单常用序号维护表

报单中的一些常用序号的维护及规律请见下表：

编号名称	发送要求	返回
requestID	TITD不做非重复性与递增要求，该值由客户自行维护	Rsp或Rtn中原样返回请求中的值 Rtn中对应名字为RequestNo
LocalOrderNo	LocalOrderNo有两种选择，客户维护该字段或不维护该字段。 ①若维护该字段，可以用LocalOrderNo撤单，在请求到达交易所之前就能撤单。要求在本 session内唯一且递增 ，OnRspUserLogin中返回的MaxLocalOrderNo为本次登录后的最小LocalOrderNo值，要求在此基础上递增。 ②若不维护该字段，则直接填0。此时 不能用LocalOrderNo撤单 。	Rsp或Rtn中原样返回请求中的值。若非本系统订单，该值返回给客户统一为0。
FrontNo	FrontNo发送选择 ① 指定席位。从GetFrontList获取当前TITD系统可连接席位，从中选择一个席位，下单时指定。 ② 不指定席位。填0或者不填该值，TITD分配报单席位。	返回实际处理席位。 特殊情况：处理的席位非当前TITD系统所用的席位，如非TITD系统下单席位统一为1。

编号名称	发送要求	返回
SequenceNo	私有流序号，报单请求中不指定	私有流号统一编码， 当日交易日内针对UserID 从1开始递增且连续，若不连续表示丢包。SequenceNo每交易日清空，OnRspUserLogin中会返回上一私有流序号。
SessionNo	请求中不用填	Rtn中返回对应的原始报单请求发出的SessionNo，若非TITD系统报单，则统一为0。
UserId	请求中不用填，默认为登录的UserId	Rtn中返回对应的原始报单或撤单请求发出的UserId，若非TITD系统报单，则统一为0。 注意：若当前为“已撤单”委托回报，且报单和撤单非同一UserId，则“ 已撤单 ”状态的Rtn中返回的为报单请求发出的 UserId 。

2.5.2 各种报单类型如何填参数？

各类型的报单类型填写，请参考下表：

SSE - 期权：

交易所支持的订单种类	PriceType	TimeInForce
普通限价申报	Limit	GFD
市价剩余转限价申报	Market2Limit	GFD
市价剩余撤销申报(普通市价单)	Market	IOC
全额即时限价申报	Limit	FOK
全额即时市价申报	Market	FOK

SSE - 股票：

交易所支持的订单种类	PriceType	TimeInForce
普通限价	Limit	---
最优5档即时成交剩余撤销的市价订单	Best1	---
最优5档即时成交剩余转限价申报的市价订单	Best2	---
本方最优价格申报的市价订单	Best3	---
对手方最优价格申报的市价订单	Best4	---

SZSE - 期权：

交易所支持的订单种类	PriceType	TimeInForce
普通限价申报	Limit	GFD
对手方最优价格	Best4	GFD
本方最优价格申报剩余转限价	Best3	GFD
最优五档即时成交剩余撤销申报	Best1	IOC
市价即时成交剩余撤销申报	Market	IOC
市价全额成交或撤销申报	Market	FOK
限价全额成交或撤销申报	Limit	FOK

SZSE - 股票：

交易所支持的订单种类	PriceType	TimeInForce
普通限价申报	Limit	GFD
对手方最优价格	Best4	GFD
本方最优价格申报剩余转限价	Best3	GFD
最优五档即时成交剩余撤销申报	Best1	IOC
市价即时成交剩余撤销申报	Market	IOC

交易所支持的订单种类	PriceType	TimeInForce
市价全额成交或撤销申报	Market	FOK

CFFEX - 期货：

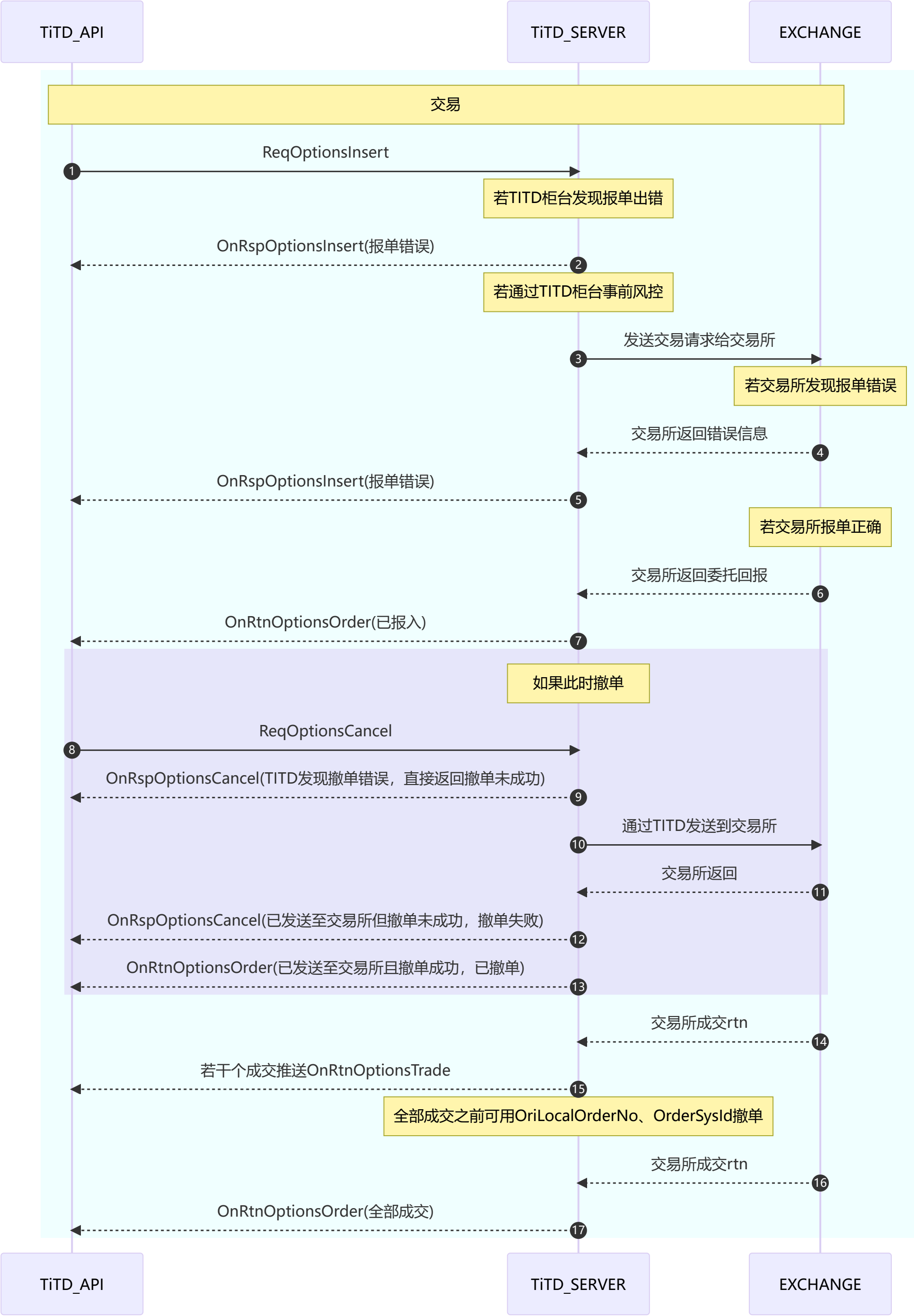
交易所支持的订单种类	PriceType	TimeInForce	成交量类型
最优一档即时成交剩余撤销指令	Best3	IOC	TD_VC_AV 任何数量
最优一档即时成交剩余转限价指令	Best3	GFD	TD_VC_AV 任何数量
最优五档即时成交剩余撤销指令	Best1	IOC	TD_VC_AV 任何数量
最优五档即时成交剩余转限价指令	Best1	GFD	TD_VC_AV 任何数量
普通限价申请	Limit	GFD	TD_VC_AV 任何数量
限价即时全部成交或撤销	Limit	IOC	TD_VC_CV 全部数量
限价即时成交剩余撤销	Limit	IOC	TD_VC_AV 任何数量
限价即时成交剩余撤销（附加最小成交量）	Limit	IOC	TD_VC_MV 最小数量

CFFEX - 期权：

交易所支持的订单种类	PriceType	TimeInForce	成交量类型
普通限价申请	Limit	GFD	TD_VC_AV 任何数量
限价即时全部成交或撤销	Limit	IOC	TD_VC_CV 全部数量
限价即时成交剩余撤销	Limit	IOC	TD_VC_AV 任何数量
限价即时成交剩余撤销（附加最小成交量）	Limit	IOC	TD_VC_MV 最小数量

2.5.3 报单处理流程

以Options接口为例，TITD交易处理流程如下。



2.5.4 报单相关函数

1、ReqStockInsert

ReqStockInsert为股票现货报单请求接口。

报单错误时响应：OnRspStockInsert。

报单正确时响应：OnRtnStockOrder、OnRtnStockTrade。

```
virtual int ReqStockInsert(CTdStockInsertReqField* pStockInsert, int nRequestID) = 0;

///股票报单请求
struct CTdStockInsertReqField
{
    TdClientNoType      ClientNo;           //交易编码序号
    TdInstrumentNoType   InstrumentNo;        //合约序号
    TdLocalOrderNoType   LocalOrderNo;       //会员内部订单编号
    TdPriceTypeType      PriceType;          //报单价格条件
    TdSideType           Side;               //买卖方向
    TdTimeInForceType    TimeInForce;        //订单有效时间类型
    TdVolumeType         Volume;             //数量
    TdPriceType           Price;             //价格
    TdOwnerTypeType      OwnerType;          //订单所有类型
    TdExchangeIdType     ExchangeId;         //交易所代码
    ///处理本次报单的席位序号,当所填的序号在系统中不存在时系统会优选最佳席位进行报单,否则,根据用户所选席位进行报单
    TdFrontNoType        FrontNo;
};
```

股票报单请求接口，现支持的报单类型及说明可见下表：

报单品种	所属类别	数量单位	价格单位	买入报单最小报单量
股票	股票	股	每股的价格（元）	100 (科创板200)
ETF	基金	份	每份的价格（元）	100
可转债	债券	张	每百元面额的价格	10

注意点：

① ClientNo是什么？

ClientNo为交易编码序号，由TITD柜台维护，每交易日唯一。

当天交易前，用户需通过同步查询函数中的GetClient系列函数获取要下单的交易编码对应的ClientNo，下单时通过ClientNo下单。

② InstrumentNo是什么？

InstrumentNo为合约对应的合约序号，由TITD柜台维护，每交易日唯一。

用户需通过同步查询函数中的GetInstrument系列函数获取要下单的合约编码对应的InstrumentNo，下单时通过InstrumentNo下单。

③ LocalOrderNo是什么？是否必填？

由用户选择维护与否。

若维护LocalOrderNo，则要求同一Session内递增且不重复。OnRspUserLogin中返回的MaxLocalOrderNo为本次登录后的最小LocalOrderNo值，要求在此基础上递增。

若不维护LocalOrderNo，直接LocalOrderNo置0。

TITD柜台对于用户订单的维护有两种方式：

- LocalOrderNo + SessionNo
- OrderSysId

用户可选以上任一——组进行报单的维护(如撤单)，若选择LocalOrderNo + SessionNo，则LocalOrderNo不能填0。

④ LocalOrderNo在回报中是否有返回？非本系统报单(不是通过TITD下单)怎样返回该值？

Rsp或Rtn中原样返回请求中的值。若非本系统订单，该值返回给客户统一为0。

⑤ nRequestID有递增性要求吗？

nRequestID由用户维护，TITD柜台不做唯一性、递增性等要求，所有请求对应的响应中，nRequestID按请求中填写的值原样返回，以下所有的nRequestID用法都一样，不再做赘述。

⑥ PriceType、TimeInForce怎么填？

见2.5.2 各种报单类型如何填写。

2、OnRspStockInsert

OnRspStockInsert为股票现货报单请求对应的错误响应。

```
virtual void OnRspStockInsert(CTdStockInsertRspField& pStockInsert, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///股票报单应答
struct CTdStockInsertRspField
{
    TdClientNoType      ClientNo;           //交易编码序号
    TdInstrumentNoType   InstrumentNo;       //合约序号
    TdLocalOrderNoType   LocalOrderNo;      //会员内部订单编号
    TdPriceTypeType      PriceType;         //报单价格条件
    TdSideType           Side;              //买卖方向
    TdTimeInForceType    TimeInForce;       //订单有效时间类型
    TdVolumeType         Volume;            //数量
    TdPriceType           Price;            //价格
    TdOwnerTypeType      OwnerType;         //订单所有类型
    TdExchangeIdType     ExchangeId;        //交易所代码
    TdFrontNoType        FrontNo;           //处理本次报单的实际席位序号
    TdSessionNoType      SessionNo;        //发送该报单的session
    TdOrderSysIDType     OrderSysId;        //交易所订单编号
};
```

3、ReqStockCancel

ReqStockCancel为股票现货报单撤销请求接口。

报单错误时响应：OnRspStockCancel。

报单正确时响应：OnRtnStockOrder。

```
virtual int ReqStockCancel(CTdOrderCancelReqField* pStockCancel, int nRequestID) = 0;
```

```
///撤单请求
struct CTdOrderCancelReqField
{
    TdSessionNoType      OriSessionNo;      //原始报单的sessionno如果是通过LocalOrderNo撤单，需要填写，不填写则默认撤销本次登录后的LocalOrderNo
    TdLocalOrderNoType    OriLocalOrderNo;  //原始交易客户方订单编号
    TdOrderSysIDType      OrderSysId;        //交易所订单编号(OrigLocalOrderNo和OrderSysId任意一个即可，当OriLocalOrderNo、OrderSysId都填写时，以OriLocalOrderNo为准)
};
```

- 注意点:
- ① 撤单用OriLocalOrderNo时，请求需附加原始报单时的OriSessionNo，若不填写OriSessionNo，默认为本次登录的Session。
 - ② 当撤单OriLocalOrderNo、OrderSysId都填写时，以OriLocalOrderNo为准。

4、OnRspStockCancel

OnRspStockCancel为股票现货报单撤销请求对应的错误响应。

```
virtual void OnRspStockCancel(CTdOrderCancelRspField& pstockCancel, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///撤单应答
struct CTdOrderCancelRspField
{
    TdSessionNoType      OriSessionNo;      //原始报单的sessionno
    TdLocalOrderNoType    OriLocalOrderNo;  //原始交易客户方订单编号
    TdOrderSysIDType      OrderSysId;        //交易所订单编号
    TdSessionNoType      SessionNo;        //发送该报单的session
};
```

5、OnRtnStockOrder

OnRtnStockOrder为股票现货报单委托回报。

```
virtual void OnRtnStockorder(CTdStockOrderRtnField& porder) {};
```



```
//股票委托回报
struct CTdStockOrderRtnField
{
    TdFrontNoType      FrontNo;           //处理本次报单的实际席位序号
    TdSequenceNoType   SequenceNo;        //回报序号(私有流序号每个交易日连续)
    TdRequestNoType     RequestNo;         //报单请求编号
    TdSessionNoType     SessionNo;         //原始报单session
    TdClientNoType      ClientNo;
    TdInstrumentNoType  InstrumentNo;
    TdLocalOrderNoType  LocalOrderNo;
    TdClientIdType      ClientId;
    TdUserIDType        UserId;            //原始报单交易用户代码
    TdInstrumentIdType  InstrumentId;
    TdExchangeIdType    ExchangeId;        //交易所代码
    TdPriceTypeType     PriceType;         //报单价格条件
    TdSideType          Side;              //买卖方向
    TdTimeInForceType   TimeInForce;       //有效期类型
    TdOwnerTypeType     OwnerType;         //订单所有类型
    TdOrdStatusType     OrdStatus;         //订单状态
    TdPriceType         Price;             //价格
    TdVolumeType        Volume;            //数量
    TdIntTimeType       OrderTime;         //委托时间
    TdOrderSysIDType    OrdersSysId;       //交易所报单编号
    TdVolumeType        LeavesVolume;      //订单剩余数量(交易所返回字段,每个交易所的含义不同)
    TdVolumeType        CancelVolume;      //订单撤销数量
};
```

6、OnRtnStockTrade

OnRtnStockOrder为股票现货报单成交回报。

```
virtual void OnRtnStockTrade(CTdStockTradeRtnField& pTrade) {};
```

```
//股票成交回报
struct CTdStockTradeRtnField
{
    TdFrontNoType      FrontNo;           //处理本次报单的实际席位序号
    TdSequenceNoType   SequenceNo;        //回报序号(私有流序号每个交易日连续)
    TdRequestNoType     RequestNo;         //报单请求编号
    TdSessionNoType     SessionNo;         //原始报单session
    TdClientNoType      ClientNo;
    TdInstrumentNoType  InstrumentNo;
    TdLocalOrderNoType  LocalOrderNo;
    TdClientIdType      ClientId;
    TdInstrumentIdType  InstrumentId;
    TdExchangeIdType    ExchangeId;        //交易所代码
    TdSideType          Side;              //买卖方向
    TdPriceType         TradePrice;        //成交价格
    TdVolumeType        TradeVolume;       //成交数量
    TdVolumeType        LeavesVolume;      //本次成交后申报余额数量
    TdIntTimeType       TradeTime;         //成交时间
    TdOrderSysIDType    OrdersSysId;       //交易所报单编号
    TdTradeIDType       TradeId;           //成交编号
    TdUserIDType        UserId;            //原始报单交易用户代码
};
```

7、ReqOptionsInsert

ReqOptionsInsert为期权、期货报单请求接口。

报单错误时响应：OnRspOptionsInsert。

报单正确时响应：OnRtnOptionsOrder、OnRtnOptionsTrade。

```
virtual int ReqOptionsInsert(CTdOrderInsertReqField* pOptionsInsert, int nRequestID) = 0;
```

```
///期权,期货报单请求
struct CTdOrderInsertReqField
{
    TdClientNoType      ClientNo;          // 交易编码序号
    TdInstrumentNoType   InstrumentNo;      // 合约序号
    TdLocalOrderNoType   LocalOrderNo;      // 会员内部订单编号
    TdPriceTypeType      PriceType;         // 报单价格条件
    TdSideType           Side;              // 买卖方向
    TdOffsetFlagType     OffsetFlag;        // 开平标记
    TdHedgeFlagType      HedgeFlag;         // 投机套保标记
};
```

```
TdTimeInForceType      TimeInForce;          // 订单有效时间类型
TdCoveredOrUncoveredType  CoveredOrUncovered; //备兑标签
TdVolumeConditionType     VolumeCondition;     //成交量类型
TdTrigConditionType       TrigCondition;        //触发条件
TdVolumeType              Volume;              //数量
TdVolumeType              MinVolume;           //最小成交量
TdPriceType               Price;               //价格
TdPriceType               StopPrice;           //止损价
TdOwnerTypeType           OwnerType;           //订单所有类型
TdExchangeIdType          ExchangeId;         //交易所代码
TdFrontNoType             FrontNo;             // 处理本次报单的席位序号
                        ///  当所填的序号在系统中不存在时系统会优选最佳席位进行报单,否则,根据用户所选席位进行报单

};
```

注意:

① ClientNo是什么?

ClientNo为交易编码序号，由TITD柜台维护，每交易日唯一。

当天交易前，用户需通过同步查询函数中的GetClient系列函数获取要下单的交易编码对应的ClientNo，下单时通过ClientNo下单。

② InstrumentNo是什么?

InstrumentNo为合约对应的合约序号，由TITD柜台维护，每交易日唯一。

用户需通过同步查询函数中的GetInstrument系列函数获取要下单的合约编码对应的InstrumentNo，下单时通过InstrumentNo下单。

③ LocalOrderNo是什么? 是否必填?

由用户选择维护与否。

若维护LocalOrderNo，则要求同一Session内递增且不重复。OnRspUserLogin中返回的MaxLocalOrderNo为本次登录后的最小LocalOrderNo值，要求在此基础上递增。

若不维护LocalOrderNo，直接LocalOrderNo置0。

TITD柜台对于用户订单的维护有两种方式：

- LocalOrderNo + SessionNo
- OrderSysId

用户可选以上任一——组进行报单的维护(如撤单)，若选择LocalOrderNo + SessionNo，则LocalOrderNo不能填0。

④ LocalOrderNo在回报中是否有返回? 非本系统报单(不是通过TITD下单)怎样返回该值?

Rsp或Rtn中原样返回请求中的值。若非本系统订单，该值返回给客户统一为0。

⑤ MinVolume、PriceType、TimeInForce怎么填写?

期货限价即时成交剩余撤销附加最小成交量时需要使用。交易所支持的报单方式及各字段的填写请参考“2.5.2 各种报单类型如何填写参数”。

⑥ OffsetFlag值怎么填?

OffsetFlag： 只有上期所区分平今仓和平历史仓。

⑦ nRequestID有递增性要求吗?

nRequestID由用户维护，TITD柜台不做唯一性、递增性等要求，所有请求对应的响应中，nRequestID按请求中填写的值原样返回，以下所有的nRequestID用法都一样，不再做赘述。

8、OnRspOptionsInsert

OnRspOptionsInsert为期权、期货报单请求对应的错误响应。

```
virtual void OnRspOptionsInsert(CTdOrderInsertRspField& pOptionsInsert, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///期权,期货报单应答
struct CTdOrderInsertRspField
{
    TdClientNoType      ClientNo;          //交易编码序号
    TdInstrumentNoType   InstrumentNo;      //合约序号
    TdLocalOrderNoType   LocalOrderNo;     //会员内部订单编号
    TdPriceTypeType      PriceType;        //报单价格条件
    TdSideType           Side;             //买卖方向
    TdOffsetFlagType     OffsetFlag;        //开平标记
    TdHedgeFlagType      HedgeFlag;        //投机套保标记
```

```
TdTimeInForceType      TimeInForce;      //订单有效时间类型
TdCoveredOrUncoveredType CoveredOrUncovered; //备兑标签
TdVolumeConditionType   VolumeCondition;   //成交量类型
TdTrigConditionType     TrigCondition;     //触发条件
TdVolumeType            Volume;            //数量
TdVolumeType            MinVolume;         //最小成交量
TdPriceType             Price;             //价格
TdPriceType             StopPrice;         //止损价
TdOwnerTypeType         OwnerType;         //订单所有类型
TdExchangeIdType        ExchangeId;        //交易所代
TdFrontNoType           FrontNo;           //处理本次报单的实际席位序号
TdSessionNoType         SessionNo;         //发送该报单的session
TdOrderSysIDType        OrdersSysId;       //交易所订单编号

};
```

9、ReqOptionsCancel

ReqOptionsCancel为期权、期货报单撤销请求接口。

撤单错误时响应：OnRspOptionsCancel。

撤单成功时响应：OnRtnOptionsOrder。

```
virtual int ReqOptionsCancel(CTdOrderCancelReqField* pOptionsCancel, int nRequestID) = 0;
```

```
///撤单请求
struct CTdOrderCancelReqField
{
    TdSessionNoType      OriSessionNo;      //原始报单的sessionno如果是通过LocalOrderNo撤单，需要填写，不填写则默认撤销本次登录后的
        LocalOrderNo
    TdLocalOrderNoType   OriLocalOrderNo;    //原始交易客户方订单编号
    TdOrderSysIDType     OrdersSysId;        //交易所订单编号(OrigLocalOrderNo和OrdersSysId任意一个即可，当OriLocalOrderNo、OrdersSysId
        都填写时，以OriLocalOrderNo为准)
};
```

注意点:

- ① 撤单用OriLocalOrderNo时，请求需附加原始报单时的OriSessionNo，若不填写OriSessionNo，默认为本次登录的Session。
- ② 当撤单OriLocalOrderNo、OrderSysId都填写时，以OriLocalOrderNo为准。

10、OnRspOptionsCancel

OnRspOptionsCancel为期权、期货报单撤销请求对应的错误响应接口。

```
virtual void OnRspOptionsCancel(CTdOrderCancelRspField& pOptionsCancel, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///撤单应答
struct CTdOrderCancelRspField
{
    TdSessionNoType      OriSessionNo;      //原始报单的sessionno
    TdLocalOrderNoType   OriLocalOrderNo;    //原始交易客户方订单编号
    TdOrderSysIDType     OrdersSysId;        //交易所订单编号
    TdSessionNoType      SessionNo;         //发送该报单的session
};
```

11、OnRtnOptionsOrder

委托回报，订单成功报入交易所后，推送委托回报。

```
virtual void OnRtnOptionsOrder(CTdOptionsOrderRtnField& pOrder) {};
```

```
///期权委托回报
struct CTdOptionsOrderRtnField
{
    TdFrontNoType        FrontNo;            //处理本次报单的实际席位序号
    TdSequenceNoType     SequenceNo;         //回报序号(私有流序号每个交易日连续)
    TdRequestNoType      RequestNo;          //报单请求编号
    TdSessionNoType      SessionNo;          //原始报单session
    TdClientNoType       ClientNo;
    TdInstrumentNoType   InstrumentNo;
    TdLocalOrderNoType   LocalOrderNo;
    TdClientIdType       ClientId;
    TdInstrumentIdType    InstrumentId;
```

```
TdExchangeIdType      ExchangeId;           //交易所代码
TdPriceTypeType       PriceType;            //报单价格条件
TdSideType            Side;                 //买卖方向
TdTimeInForceType     TimeInForce;          //订单有效时间类型
TdOwnerTypeType       OwnerType;            //订单所有类型
TdOrdStatusType       OrdStatus;            //当前订单的状态
TdPriceType           Price;                //价格
TdVolumeType          volume;               //数量
TdIntTimeType         OrderTime;            //委托时间
TdOrderSysIDType      OrdersSysId;          //交易所订单编号
TdVolumeType          LeavesVolume;         //订单剩余数量(交易所返回字段,每个交易所的含义不同)
TdVolumeType          CancelVolume;         //订单撤销数量
TdVolumeType          MinVolume;            //最小成交量
TdPriceType           StopPrice;            //止损价
TdOffsetFlagType      OffsetFlag;           //开平标记
TdHedgeFlagType       HedgeFlag;            //投机套保标记
TdCoveredOrUncoveredType CoveredOrUncovered; //备兑标签
TdVolumeConditionType VolumeCondition;      //成交量类型
TdTrigConditionType   TrigCondition;        //触发条件
TdUserIDType          UserId;               //原始报单交易用户代码

};
```

注意:

① 怎样在OnRtnOptionsOrder里维护当前已成交的总数量？

- **当您交易的为上交所、深交所期权时**，委托回报的推送机制为：推送两次委托回报，第一次为OrdStatus是“已报入”状态的委托回报，第二次委托回报推送机制为：该笔委托状态不会再发生改变，比如该笔报单已撤销或全部成交。中间状态下不维护OnRtnOptionsOrder，如有部分成交，但未撤单时，不推送OnRtnOptionsOrder。所以**OnRtnOptionsOrder里维护当前已成交的总数量不可靠，请以成交回报OnRtnOptionsTrade中的成交数量为准维护当前已成交的总数量。**
- **当您交易的为中金所期货、期权时**，第一个委托回报根据报单状态的不同，状态为“已撤单”、“已报入”、“全部成交”等都有可能。

②OnRtnOptionsOrder中LeavesVolume剩余数量的含义

- **当您交易上交所期权**，报单状态为“已报入”且报单类为非全额的市价单时，LeavesVolume为已撤销或已转成限价单的数量。
如“市价剩余转限价单”在“已报入”状态下为转成限价单的量；
“普通市价单”在“已报入”状态下为未能成交撤单的量；
- **当您交易深交所期权**，“已报入”状态下，不维护LeavesVolume字段，都为该委托单的报单数量。

③市价单无对手方时的回报不一样。

上交所期权，无对手方时，直接返回OnRspOrderinsert错单。

深交所期权，无对手方时，是“已报入”、“已撤单”两个委托回报。

12、OnRtnOptionsTrade

成交回报，订单成功成交时，推送成交回报。

```
virtual void OnRtnOptionsTrade(CTdOptionsTradeRtnField& pTrade) {};
```

```
//期权委托回报
struct CTdOptionsOrderRtnField
{
    TdFrontNoType      FrontNo;               //处理本次报单的实际席位序号
    TdSequenceNoType   SequenceNo;            //回报序号(私有流序号每个交易日连续)
    TdRequestNoType     RequestNo;            //报单请求编号
    TdSessionNoType     SessionNo;            //原始报单session
    TdClientNoType      ClientNo;
    TdInstrumentNoType  InstrumentNo;
    TdLocalOrderNoType  LocalOrderNo;
    TdClientIdType      ClientId;
    TdInstrumentIdType  InstrumentId;
    TdExchangeIdType    ExchangeId;           //交易所代码
    TdPriceTypeType     PriceType;            //报单价格条件
    TdSideType          Side;                 //买卖方向
    TdTimeInForceType   TimeInForce;          //订单有效时间类型
    TdOwnerTypeType     OwnerType;            //订单所有类型
    TdOrdStatusType     OrdStatus;            //当前订单的状态
    TdPriceType         Price;                //价格
    TdVolumeType        volume;               //数量
    TdIntTimeType       OrderTime;            //委托时间
    TdOrderSysIDType    OrdersSysId;          //交易所订单编号
    TdVolumeType        LeavesVolume;         //订单剩余数量(交易所返回字段,每个交易所的含义不同)
    TdVolumeType        CancelVolume;         //订单撤销数量
    TdVolumeType        MinVolume;            //最小成交量
    TdPriceType         StopPrice;            //止损价
    TdOffsetFlagType    OffsetFlag;           //开平标记
    TdHedgeFlagType     HedgeFlag;            //投机套保标记
    TdCoveredOrUncoveredType CoveredOrUncovered; //备兑标签
```

```
TdVolumeConditionType    VolumeCondition;    //成交量类型
TdTrigConditionType      TrigCondition;      //触发条件
TdUserIDType             UserId;             //原始报单交易用户代码

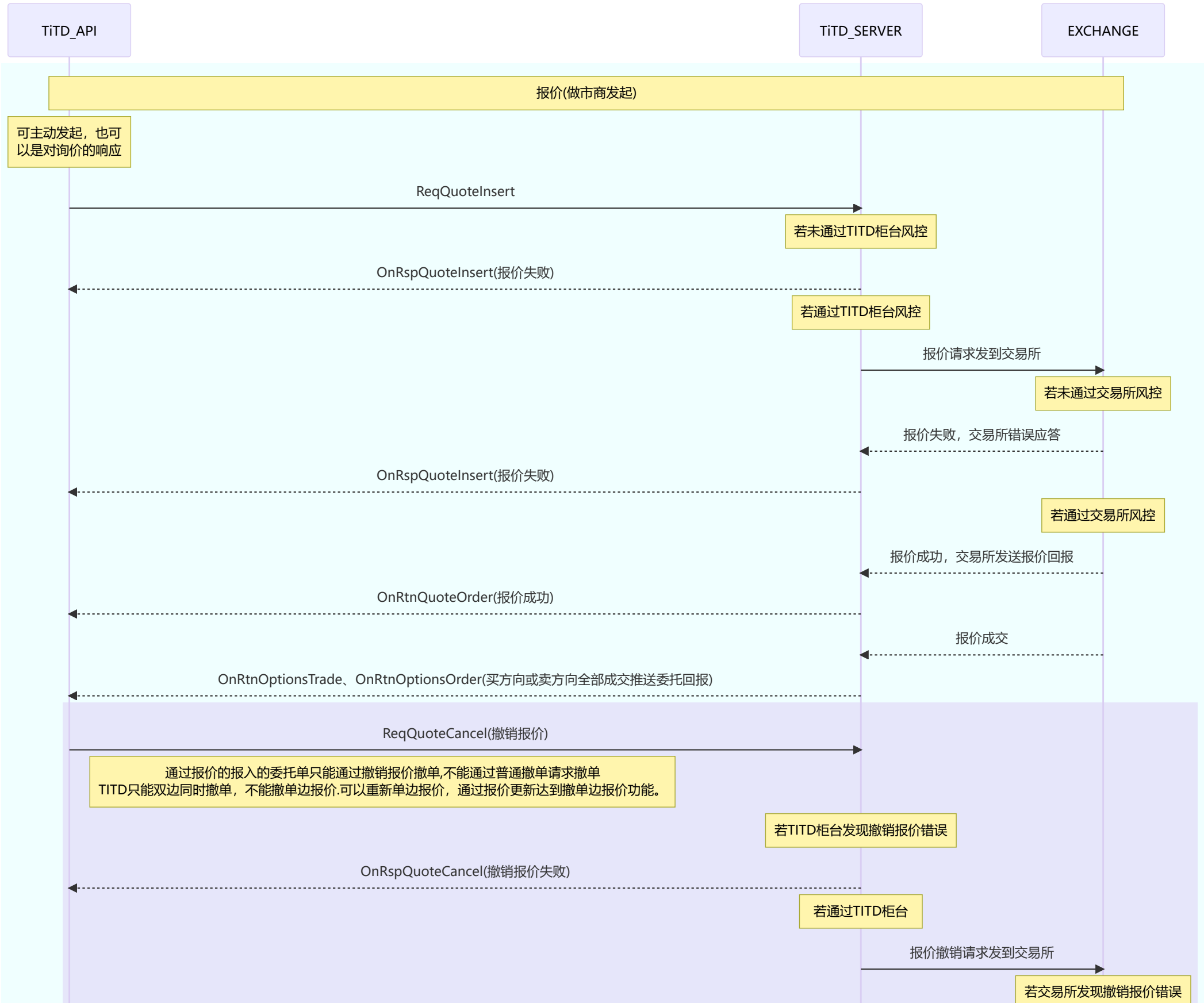
};
```

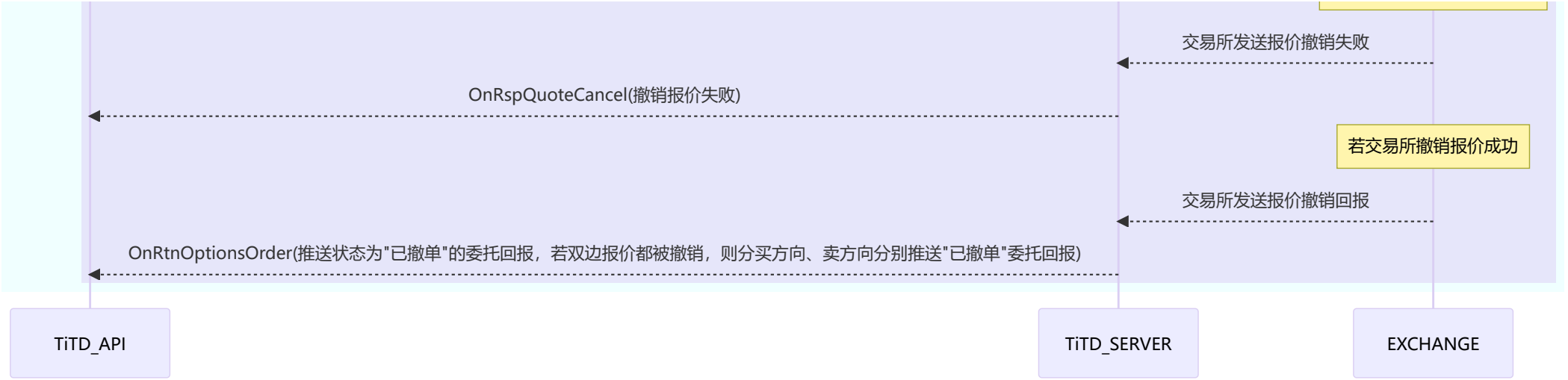
2.6 报价

```
virtual int ReqQuoteInsert(CTdQuoteInsertReqField* pQuoteInsert, int nRequestID) = 0;    //报价
virtual int ReqQuoteCancel(CTdOrderCancelReqField* pQuoteCancel, int nRequestID) = 0;    //报价撤销
```

```
//报价应答(报价失败时应答)
virtual void OnRspQuoteInsert(CTdQuoteInsertRspField& pQuoteInsert, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
//报价撤销应答(报价撤销失败时应答)
virtual void OnRspQuoteCancel(CTdOrderCancelRspField& pQuoteCancel, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
//报价回报
virtual void OnRtnQuoteOrder(CTdQuoteOrderRtnField& pQuote) {};
```

2.6.1 报价处理流程





2.6.2 报价相关函数

1、ReqQuoteInsert

ReqQuoteInsert为报价请求接口。

报价错误时响应：OnRspQuoteInsert。

报价正确时响应：OnRtnQuoteOrder、OnRtnOptionsOrder、OnRtnOptionsTrade。详细推送模式可参考2.6.1报价处理流程。

```
virtual int ReqQuoteInsert(CTdQuoteInsertReqField* pQuoteInsert, int nRequestID) = 0;    //报价
```

```
///期权,期货报价请求
struct CTdQuoteInsertReqField
{
    TdClientNoType      ClientNo;           //交易编码序号
    TdInstrumentNoType   InstrumentNo;       //合约序号
    TdLocalOrderNoType   LocalOrderNo;      //会员内部订单编号
    TdQuoteReqIDType     QuoteReqId;        //报价请求ID,预留字段,当报价是对请求的响应时, 填写报价请求的ID
    TdPriceType          BidPx;             //买报价
    TdPriceType          AskPx;             //卖报价
    TdVolumeType         BidSize;           //买数量
    TdVolumeType         AskSize;           //卖数量
    TdOffsetFlagType     BidOffsetFlag;     //平仓标识
    TdOffsetFlagType     AskOffsetFlag;     //平仓标识
    TdOwnerTypeType      OwnerType;         //订单所有类型
    TdExchangeIdType     ExchangeId;        //交易所代码
    ///处理本次报单的席位序号,当所填的序号在系统中不存在时系统会优选最佳席位进行报单,否则,根据用户所选席位进行报单
    TdFrontNoType        FrontNo;
};
```

2、OnRspQuoteInsert

OnRspQuoteInsert报价错误时响应。

```
virtual void OnRspQuoteInsert(CTdQuoteInsertRspField& pQuoteInsert, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///期权,期货报价应答
struct CTdQuoteInsertRspField
{
    TdClientNoType      ClientNo;           //交易编码序号
    TdInstrumentNoType   InstrumentNo;       //合约序号
    TdLocalOrderNoType   LocalOrderNo;      //会员内部订单编号
    TdQuoteReqIDType     QuoteReqId;        //报价请求ID,预留字段,当报价是对请求的响应时, 填写报价请求的ID
    TdPriceType          BidPx;             //买报价
    TdPriceType          AskPx;             //卖报价
    TdVolumeType         BidSize;           //买数量
    TdVolumeType         AskSize;           //卖数量
    TdOffsetFlagType     BidOffsetFlag;     //平仓标识
```

```
TdOffsetFlagType      AskOffsetFlag;          //平仓标识
TdOwnerTypeType       OwnerType;              //订单所有类型
TdExchangeIdType      ExchangeId;             //交易所代码
TdFrontNoType         FrontNo;                //处理本次报单的实际席位序号
TdSessionNoType       SessionNo;              //发送该报单的session
TdOrderSysIDType      OrderSysId;             //交易所订单编号
TdOrderSysIDType      BidOrderId;              //买方交易所订单编号
TdOrderSysIDType      AskOrderId;             //卖方交易所订单编号
};
```

3、OnRtnQuoteOrder

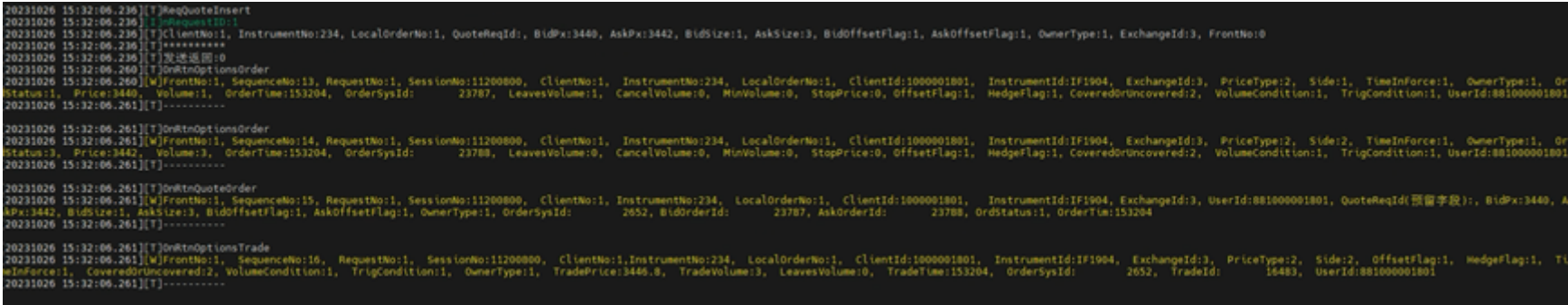
OnRtnQuoteOrder为报价回报，到报价正确报入交易所时，推送OnRtnQuoteOrder。

除报价回报OnRtnQuoteOrder外，报价的单腿的委托回报OnRtnOptionsOrder和成交回报OnRtnOptionsTrade也会推送给客户，标识订单的委托情况和成交情况。

```
virtual void OnRtnQuoteOrder(CTdQuoteOrderRtnField& pQuote) {};
```

```
//报价回报
struct CTdQuoteOrderRtnField
{
    TdFrontNoType      FrontNo;                //处理本次报单的实际席位序号
    TdSequenceNoType   SequenceNo;              //回报序号(私有流序号每个交易日连续)
    TdRequestNoType    RequestNo;              //报单请求编号
    TdSessionNoType    SessionNo;              //原始报单session
    TdClientNoType     ClientNo;
    TdInstrumentNoType InstrumentNo;
    TdLocalOrderNoType LocalOrderNo;
    TdClientIdType     ClientId;
    TdInstrumentIdType InstrumentId;
    TdExchangeIdType   ExchangeId;              //交易所代码
    TdUserIDType        UserId;                  //原始报单交易用户代码
    TdQuoteReqIDType    QuoteReqId;             //报价请求ID,预留字段,当报价是对请求的响应时，填写报价请求的ID
    TdPriceType         BidPx;                  //买报价
    TdPriceType         AskPx;                  //卖报价
    TdVolumeType        BidSize;                //买数量
    TdVolumeType        AskSize;                //卖数量
    TdOffsetFlagType    BidOffsetFlag;          //平仓标识
    TdOffsetFlagType    AskOffsetFlag;          //平仓标识
    TdOwnerTypeType     OwnerType;              //订单所有类型
    TdOrderSysIDType    OrderSysId;             //交易所订单编号
    TdOrderSysIDType    BidOrderId;             //买方交易所订单编号
    TdOrderSysIDType    AskOrderId;            //卖方交易所订单编号
    TdOrdStatusType     OrdStatus;              //当前订单状态
    TdIntTimeType       OrderTime;              //委托时间
};
```

下图为测试中金所报价时，一腿马上成交，一腿已报入时的委托回报，先收到两个OnRtnOptionsOrder委托回报推送，标识当前报价单买卖双方的委托状态；再收到OnRtnQuoteOrder，标识当前报价已成功报入交易所；最后收到OnRtnOptionsTrade成交回报推送，标识报价某一腿已成交。



4、ReqQuoteCancel

ReqQuoteCancel为报价撤销请求。

报价撤销错误时响应：OnRspQuoteCancel。

报价撤销正确时响应：OnRtnOptionsOrder

```
virtual int ReqQuoteCancel(CTdOrderCancelReqField* pQuoteCancel, int nRequestID) = 0;    //报价撤销
```



```
///撤单请求
struct CTdOrderCancelReqField
{
    TdSessionNoType      OriSessionNo;      //原始报单的sessionno如果是通过LocalOrderNo撤单，需要填写，不填写则默认撤销本次登录后的LocalOrderNo
    TdLocalOrderNoType    OriLocalOrderNo;    //原始交易客户方订单编号
    TdOrderSysIDType      OrderSysId;         //交易所订单编号(OrigLocalOrderNo和OrderSysId任意一个即可，当OriLocalOrderNo、OrderSysId都填写时，以OriLocalOrderNo为准)
};
```

注意点:

- ① 撤单用OriLocalOrderNo时，请求需附加原始报单时的OriSessionNo，若不填写OriSessionNo，默认为本次登录的Session。
- ② 当撤单OriLocalOrderNo、OrderSysId都填写时，以OriLocalOrderNo为准。

5、OnRspQuoteCancel

OnRspQuoteCancel为报价撤销错误时的响应。

```
virtual void OnRspQuoteCancel(CTdOrderCancelRspField& pQuoteCancel, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///撤单应答
struct CTdOrderCancelRspField
{
    TdSessionNoType      OriSessionNo;      //原始报单的sessionno
    TdLocalOrderNoType    OriLocalOrderNo;    //原始交易客户方订单编号
    TdOrderSysIDType      OrderSysId;         //交易所订单编号
    TdSessionNoType      SessionNo;         //发送该报单的session
};
```

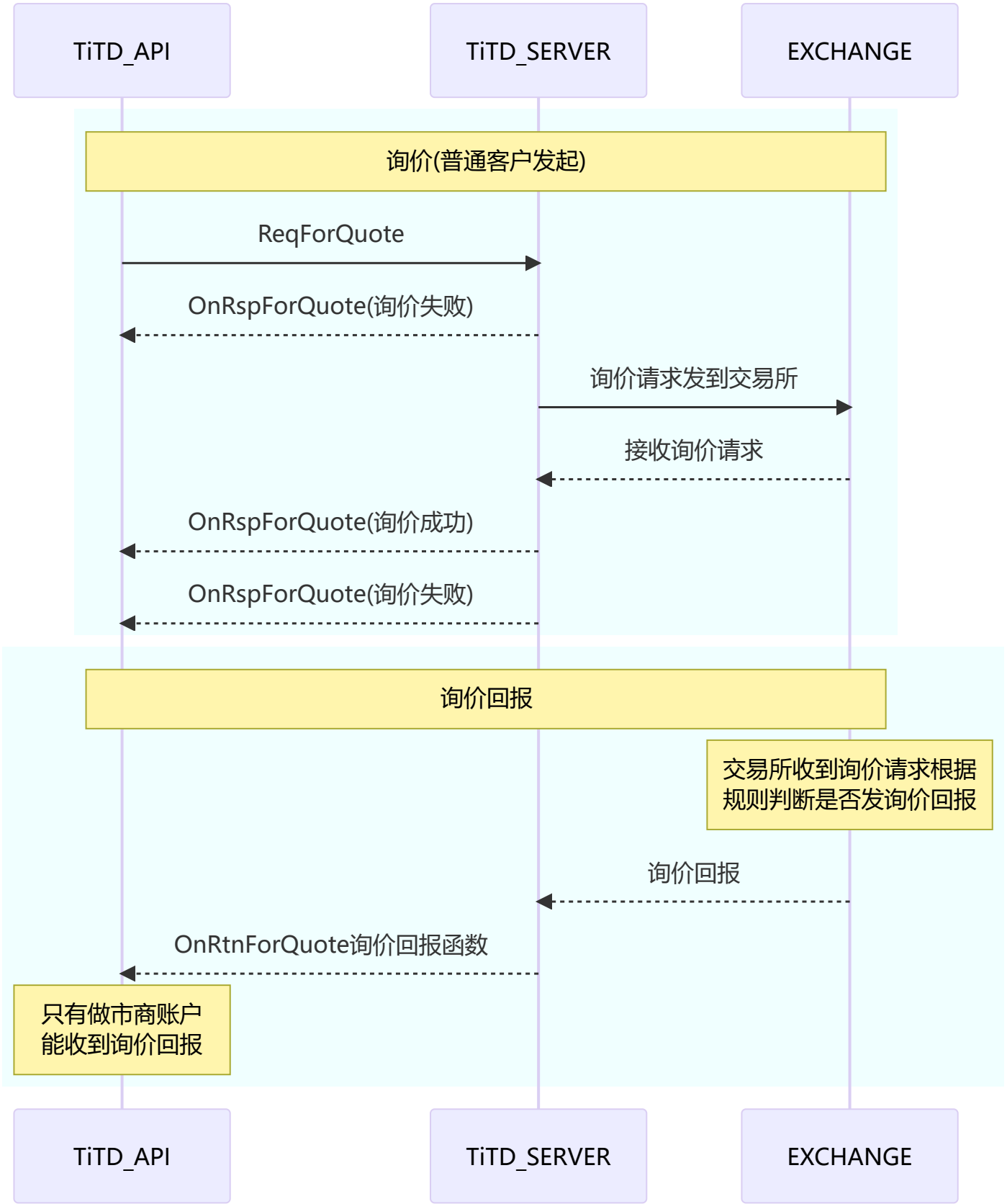
2.7 询价

询价功能暂未支持

```
virtual int ReqForQuote(CTdForQuoteReqField* pForQuote, int nRequestID) = 0; //询价
```

```
virtual void OnRspForQuote(CTdForQuoteRspField& pForQuote, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {}; //询价应答
virtual void OnRtnForQuote(CTdForQuoteRtnField& pForQuote) {}; //询价通知
```

2.7.1询价处理流程



- 通过询价应答OnRspForQuote告知客户有没有询价成功。
- 询价回报为公有流，只有做市商账号，且在初始化订阅公有流接收方式时设置为3"接收公有流"才能接收到询价通知。

2.7.2 询价相关函数

询价功能暂未支持

```
virtual int ReqForQuote(CTdForQuoteReqField* pForQuote, int nRequestID) = 0; //询价
virtual void OnRspForQuote(CTdForQuoteRspField& pForQuote, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {}; //询价应答
virtual void OnRtnForQuote(CTdForQuoteRtnField& pForQuote) {}; //询价通知
```

1、ReqForQuote

该方法为询价请求，对应的应答为OnRspForQuote，OnRspForQuote用于告知客户询价有没有正确发送至交易所。

```
virtual int ReqForQuote(CTdForQuoteReqField* pForQuote, int nRequestID) = 0; //询价
```

2、OnRspForQuote

该方法为询价应答，用于告知客户询价有没有正确发送。

```
virtual void OnRspForQuote(CTdForQuoteRspField& pForQuote, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {}; //询价应答
```

```
///询价应答
struct CTdForQuoteRspField
{
    TdClientNoType      ClientNo;           ///交易编码序号
    TdInstrumentNoType   InstrumentNo;       ///合约序号
    TdLocalOrderNoType   LocalOrderNo;      ///会员内部订单编号
    TdOwnerTypeType      OwnerType;         ///订单所有类型
    TdExchangeIdType     ExchangeId;        ///交易所代码
    TdFrontNoType        FrontNo;           ///处理本次报单的实际席位序号
    TdSessionNoType      SessionNo;         ///发送该报单的session
    TdQuoteReqIDType     QuoteReqId;        ///报价请求ID
};
```

3、OnRtnForQuote

该方法为询价通知，需客户订阅共有流。

```
virtual void OnRtnForQuote(CTdForQuoteRtnField& pForQuote) {};//询价通知
```

```
//询价通知
struct CTdForQuoteRtnField
{
    TdInstrumentNoType   InstrumentNo;
    TdInstrumentIdType   InstrumentId;
    TdExchangeIdType     ExchangeId;        ///交易所代码
    TdQuoteReqIDType     QuoteReqId;        ///报价请求ID
};
```

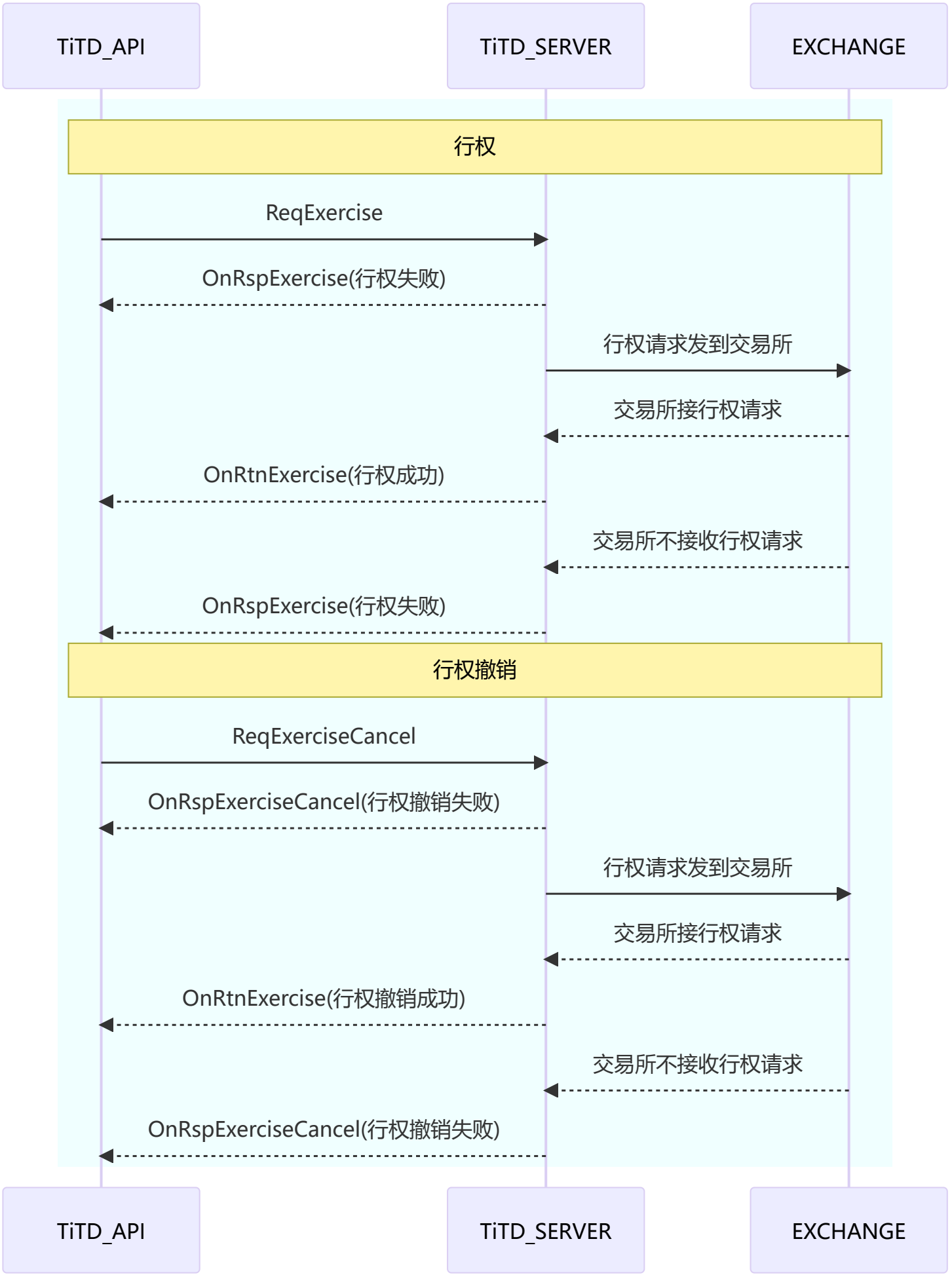
2.8 行权

```
virtual int ReqExercise(CTdExerciseReqField* pExercise, int nRequestID) = 0;           ///行权
virtual int ReqCombExercise(CTdCombExerciseReqField* pExercise, int nRequestID) = 0;  ///组合行权
virtual int ReqExerciseCancel(CTdOrderCancelReqField* pExerciseCancel, int nRequestID) = 0;  ///行权撤销
```

```
//行权应答(行权失败时应答)
virtual void OnRspExercise(CTdExerciseRspField& pExercise, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};  
//组合行权应答(组合行权失败时应答)
virtual void OnRspCombExercise(CTdCombExerciseRspField& pExercise, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};  
//行权撤销应答
virtual void OnRspExerciseCancel(CTdOrderCancelRspField& pExercise, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};  
//行权回报
virtual void OnRtnExercise(CTdExerciseRtnField& pExercise) {};
```

2.8.1行权处理流程

普通行权ReqExercise，交易逻辑如下：



组合行权交易处理的逻辑如下：



2.8.2 行权相关函数

1、ReqExercise

ReqExercise为单腿行权请求，组合行权请用ReqCombExercise。

行权错误时响应：OnRspExercise。

行权正确时响应：OnRtnExercise。

```
virtual int ReqExercise(CTdExerciseReqField* pExercise, int nRequestID) = 0; //行权
```

```
///  
struct CTdExerciseReqField  
{  
    TdClientNoType      ClientNo;           ///  
    TdInstrumentNoType   InstrumentNo;        ///  
    TdLocalOrderNoType   LocalOrderNo;       ///  
    TdVolumeType         Volume;             ///  
    TdOwnerTypeType      OwnerType;          ///  
    TdExchangeIdType     ExchangeId;         ///  
};
```

- 注意:
- ① 组合行权不冻结资金。
 - ② 行看涨期权时，行权的所冻结资金加在资金中的“冻结保证金”中。

2、OnRspExercise

OnRspExercise为行权错误响应。

```
virtual void OnRspExercise(CTdExerciseRspField& pExercise, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///  
struct CTdExerciseRspField  
{  
    TdClientNoType      ClientNo;           ///  
    TdInstrumentNoType   InstrumentNo;        ///  
    TdLocalOrderNoType   LocalOrderNo;       ///  
    TdVolumeType         Volume;             ///  
    TdOwnerTypeType      OwnerType;          ///  
    TdExchangeIdType     ExchangeId;         ///  
    TdFrontNoType        FrontNo;            ///  
    TdSessionNoType      SessionNo;          ///  
    TdOrderSysIDType     OrderSysId;         ///  
};
```

3、ReqCombExercise

ReqCombExercise为组合行权请求。

行权错误时响应：OnRspCombExercise。

行权正确时响应：OnRtnExercise(分两次，每个单腿都对应一次OnRtnExercise)。

```
virtual int ReqCombExercise(CTdCombExerciseReqField* pExercise, int nRequestID) = 0;           ///  

```

```
///  
struct CTdCombExerciseReqField  
{  
    TdClientNoType      ClientNo;           ///  
    TdLocalOrderNoType   LocalOrderNo;       ///  
    TdVolumeType         Volume;             ///  
    TdInstrumentNoType   LegInstrumentNo1;    ///  
    TdVolumeType         LegVolume1;         ///  
    TdInstrumentNoType   LegInstrumentNo2;    ///  
    TdVolumeType         LegVolume2;         ///  
    TdOwnerTypeType      OwnerType;          ///  
    TdExchangeIdType     ExchangeId;         ///  
};
```

4、OnRspCombExercise

OnRspCombExercise为组合行权错误响应。

```
virtual void OnRspCombExercise(CTdCombExerciseRspField& pExercise, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///  
struct CTdCombExerciseRspField  
{  
    TdClientNoType      ClientNo;           ///  
    TdLocalOrderNoType   LocalOrderNo;       ///  
    TdVolumeType         Volume;             ///  
    TdInstrumentNoType   LegInstrumentNo1;    ///  

```



```
TdVolumeType      LegVolume1;           //腿1申报数量
TdInstrumentNoType LegInstrumentNo2;      //腿2合约序号
TdVolumeType      LegVolume2;           //腿2申报数量
TdOwnerTypeType   OwnerType;             //订单所有类型
TdExchangeIdType  ExchangeId;            //交易所代码
TdFrontNoType     FrontNo;               //处理本次报单的实际席位序号
TdSessionNoType   SessionNo;            //发送该报单的session
};
```

5、ReqExerciseCancel

ReqExerciseCancel为行权撤销、组合行权撤销请求。

行权撤销错误时响应：OnRspExerciseCancel。

行权撤销正确时响应：OnRtnExercise(若为组合行权的撤销，则推送分两次，每个单腿都对应一次OnRtnExercise)。

```
virtual int ReqExerciseCancel(CTdOrderCancelReqField* pExerciseCancel, int nRequestID) = 0;           //行权撤销
```

```
///期权,期货撤单请求
struct CTdOrderCancelReqField
{
    TdSessionNoType      OriSessionNo;           //原始报单的sessionno如果是通过LocalOrderNo撤单，需要填写，
                                                    ///不填写则默认撤销本次登录后的LocalOrderNo
    TdLocalOrderNoType    OriLocalOrderNo;       //原始交易客户方订单编号
    TdOrderSysIDType      OrderSysId;            //交易所订单编号(OrigLocalOrderNo和OrderSysId任意一个即可)
};
```

注意点:

- ① 组合行权撤销与普通行权共用函数ReqExerciseCancel。
- ② 组合行权分推送两笔行权通知OnRtnExercise。两笔OnRtnExercise中的OrderSysId一样，组合行权撤销时，组合行权的两个单腿一起撤销。

6、OnRspExerciseCancel

OnRspExerciseCancel为行权撤销、组合行权撤销错误响应。

```
virtual void OnRspExerciseCancel(CTdOrderCancelRspField& pExercise, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
///期权,期货撤单应答
struct CTdOrderCancelRspField
{
    TdSessionNoType      OriSessionNo;           //原始报单的sessionno
    TdLocalOrderNoType    OriLocalOrderNo;       //原始交易客户方订单编号
    TdOrderSysIDType      OrderSysId;            //交易所订单编号
    TdSessionNoType      SessionNo;            //发送该报单的session
};
```

7、OnRtnExercise

该方法为客户行权、组合行权正确报入后的行权通知，属于客户私有流。

若为组合行权对应的推送，两个单腿对应的OrderSysId是一样的。

```
virtual void OnRtnExercise(CTdExerciseRtnField& pExercise) {};
```

```
///行权回报
struct CTdExerciseRtnField
{
    TdFrontNoType      FrontNo;               //处理本次报单的实际席位序号
    TdSequenceNoType   SequenceNo;            //回报序号(私有流序号每个交易日连续)
    TdRequestNoType     RequestNo;             //报单请求编号
    TdSessionNoType     SessionNo;            //原始报单session
    TdClientNoType      ClientNo;
    TdInstrumentNoType   InstrumentNo;
    TdLocalOrderNoType   LocalOrderNo;
    TdClientIdType       ClientId;
    TdInstrumentIdType   InstrumentId;
    TdExchangeIdType     ExchangeId;           //交易所代码
    TdVolumeType         Volume;               //申报数量
    TdOwnerTypeType     OwnerType;            //订单所有类型
    TdOrdStatusType     OrdStatus;
```

```
TdIntTimeType      ExerciseTime;          //行权成功时间
TdOrderSysIDType    OrderSysId;           //交易所订单编号
TdUserIDType        UserId;               //原始报单交易用户代码
};
```

2.9 其他

2.9.1 申请组合

```
virtual int ReqMarginCombAction(CTdMarginCombActionField* pMarginCombAction, int nRequestID) = 0;
```

```
virtual void OnRspMarginCombAction(CTdMarginCombActionRspField& pMarginCombAction, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {}; //
组合申请错误应答
virtual void OnRtnMarginCombAction(CTdMarginCombActionRtnField& pMarginCombAction) {}; //组合报入回报
```

1、ReqMarginCombAction

ReqMarginCombAction为申请组合。

申请组合错误时响应：OnRspMarginCombAction。

申请组合正确时响应：OnRtnMarginCombAction。

```
virtual int ReqMarginCombAction(CTdMarginCombActionField* pMarginCombAction, int nRequestID) = 0;
```

```
///客户申请组合请求
struct CTdMarginCombActionField
{
    TdClientNoType      ClientNo;          //交易编码序号
    TdLocalOrderNoType   LocalOrderNo;     //会员内部订单编号
    TdVolumeType         Volume;           //申报数量
    TdCombType           Side;             //组合与拆分组组合标记,组合策略为“ZBD”时，组合与拆分方向只能为组合
    TdCombIDType         CombId;           //组合策略编码:目前为7种组合策略：CNSJC、PXSJC、PNSJC、CXSJC、KS、KKS、ZBD
    TdOrderSysIDType     CombInstId;       //组合编码。组合申报时，该字段为空格；拆分申报时，填写拟拆分组组合的组合编码
    TdOwnerTypeType      OwnerType;        //订单所有类型
    TdExchangeIDType     ExchangeId;       //交易所代码
    int NoLeges;          //成分合约数，取值不超过4，后接重复组
    OmItem item[4];
};
```

注意点：

- ① 各item中成分合约的先后顺序不能调换，否则会报错。
- ② 上交所组合策略类型和下单方式以及和ctp组合方法对比

组合策略	策略名称	组合保证金算法	头寸1	头寸2	Ctp头寸1	Ctp头寸2	Ctp买卖方向
CNSJC	认购牛市价差	0	低价C,权利	高价C,义务	低价C,权利	高价C,义务	买
PXSJC	认沽熊市价差	0	高价P,权利	低价P,义务	高价P,权利	低价P,义务	买
PNSJC	认沽牛市价差	行权价之差	低价P,权利	高价P,义务	高价P,义务	低价P,权利	卖
CXSJC	认购熊市价差	行权价之差	高价C,权利	低价C,义务	低价C,义务	高价C,权利	卖
KS	跨式空头	表下①	同价C,义务	同价P,义务	同价C,义务	同价P,义务	卖
KKS	宽跨式空头	表下②	高价C,义务	低价P,义务	高价C,,义务	低价P,义务	卖

注① KS组合保证金算法： Max(认购期权开仓保证金，认沽期权开仓保证金) + 保证金较低方合约的前结算价×合约单位

注② KKS组合保证金算法： Max(认购期权开仓保证金，认沽期权开仓保证金) + 保证金较低方合约的前结算价×合约单位

注③ 上表中高价、低价是期权行权价的高低。

注④ 深交所上表中上交所相关类型方法一样。

注⑤ 上表中组合保证金按公式计算后，客户实收保证金需乘以MarginRate(查询保证金中可查)。

③ 转备兑（ZBD）目前只支持上交所。做转备兑时Side填写'B'。

2、OnRspMarginCombAction

OnRspMarginCombAction为申请组合错误时响应

```
virtual void OnRspMarginCombAction(CTdMarginCombActionRspField& pMarginCombAction, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

组合申请应答

```
///客户申请组合应答
struct CTdMarginCombActionRspField
{
    TdClientNoType      ClientNo;           //交易编码序号
    TdLocalOrderNoType  LocalOrderNo;       //会员内部订单编号
    TdVolumeType        Volume;             //申报数量
    TdCombType          Side;               //组合与拆分组合标记
    TdCombIDType        CombId;             //组合策略编码:目前为7种组合策略: CNSJC、PXSJC、PNSJC、CXSJC、KS、KKS、ZBD
    TdOrdersSysIDType   CombInstId;         //组合编码。组合申报成功时返回组合的组合编码
    TdExchangeIDType    ExchangeId;        //交易所代码
    TdFrontNoType       FrontNo;            //处理本次报单的实际席位序号
    TdSessionNoType     SessionNo;          //发送该报单的session
    int NoLeges;         //成分合约数，取值不超过4，后接重复组
    OmItem item[4];
};
```

ClientNo、LocalOrderNo、Volume、Side、Combld、CombInstld、Exchangeld、NoLeges、item根据请求字段原样返回。

FrontNo、SessionNo返回实际处理的席位号和发送报单的session。

3、OnRtnMarginCombAction

OnRtnMarginCombAction为组合回报，当申请组合正确且成功报入交易所推送给客户。

```
virtual void OnRtnMarginCombAction(CTdMarginCombActionRtnField& pMarginCombAction) {};
```

```
///客户申请组合回报
struct CTdMarginCombActionRtnField
{
    TdFrontNoType      FrontNo;           //处理本次报单的实际席位序号
    TdSequenceNoType   SequenceNo;       //回报序号(私有流序号每个交易日连续)
    TdRequestNoType    RequestNo;        //报单请求编号
    TdSessionNoType    SessionNo;        //原始报单session
    TdClientNoType     ClientNo;         //交易编码序号
    TdLocalOrderNoType LocalOrderNo;     //会员内部订单编号
    TdClientIDType     ClientId;         //交易编码
    TdUserIDType       UserId;           //原始报单交易用户代码
    TdExchangeIDType   ExchangeId;      //交易所代码
    TdCombType         Side;             //组合与拆分组合标记,组合策略为“ZBD”时，组合与拆分方向只能为组合
    int NoLeges;       //成分合约数，取值不超过4，后接重复组
    TdVolumeType       Volume;          //申报数量(拆分组合的时候需要填入拆分的数量，组合的时候该字段无效)
    TdCombIDType       CombId;          //组合策略编码:目前为7种组合策略: CNSJC、PXSJC、PNSJC、CXSJC、KS、KKS、ZBD
    TdOrdersSysIDType  CombInstId;      //组合编码。组合申报成功时返回组合的组合编码
    TdOwnerTypeType    OwnerType;       //订单所有类型
    TdIntTimeType      CombActionTime;  //组合、解组合成功时间
    TdOrdersSysIDType  OrderSysId;      //交易所报单编号
    TdPriceType        ComMargin;       //组合或拆分后保证金的变化值(组合为正拆分为负)
    OmItemRtn          item[4];
};
```

2.9.2 申请锁仓

当客户需要备兑开仓时，客户需要先使用ReqStockLock进行现货锁仓。

注意：当客户行权，行看跌期权时，客户不需要主动发送现货锁仓，柜台后台会自动对行权所需要的现货进行锁定。

```
virtual int ReqStockLock(CTdStockLockReqField* pStockLock, int nRequestID) = 0;
```

```
virtual void OnRspStockLock(CTdStockLockRspField& pStockLock, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

1、ReqStockLock

该方法为证券锁定(上交所)请求。

当上交所需要做备兑开仓，需要先使用该方法锁定现货仓位。

当深交所需要做备兑开仓，不需要先锁定现货仓位，可直接用ReqOptionsInsert报备兑仓。

证券锁定错误时响应：OnRspStockLock。

证券锁定正确时响应：OnRspStockLock。

```
virtual int ReqStockLock(CTdStockLockReqField* pStockLock, int nRequestID) = 0;

///证券锁定请求
struct CTdStockLockReqField
{
    TdClientNoType      ClientNo;           //交易编码序号
    TdInstrumentNoType   InstrumentNo;       //合约序号
    TdLocalOrderNoType   LocalOrderNo;      //会员内部订单编号
    TdLockType           Locked;             //锁定标志撤销
    TdVolumeType         Volume;            //数量
    TdExchangeIdType     ExchangeId;        //交易所代码
    ///处理本次报单的席位序号,当所填的序号在系统中不存在时系统会优选最佳席位进行报单,否则,根据用户所选席位进行报单
    TdFrontNoType        FrontNo;
};
```

证券锁定数量单位为股。

证券锁定后，可以用ReqQryStockPosition查询锁定的持仓。

股票持仓查询应答的结构体如下

```
///股票持仓查询应答
struct CTdRspQryStockPositionField
{
    TdClientNoType      ClientNo;
    TdInstrumentNoType   InstrumentNo;
    TdClientIdType       ClientId;
    TdInstrumentIdType   InstrumentId;
    TdExchangeIdType     ExchangeId;        /// 交易所代码
    TdVolumeType         YdPosition;        /// 上日持仓
    TdVolumeType         Position;          /// 总持仓
    TdVolumeType         TodayPosition;     /// 今日持仓
    TdVolumeType         FrozenVolume;      /// 冻结数量
    TdVolumeType         BuyTradeVolume;    /// 当日买成交量
    TdVolumeType         SellTradeVolume;   /// 当日卖成交量
    TdPriceType          Price;             /// 持仓价格
    TdVolumeType         LockVolume;        /// 锁仓数量(已经被锁定的可用于备兑开仓的数量)
    TdVolumeType         FrozenLock;        /// 锁仓冻结数量(解锁时冻结的数量)
};
```

当期权系统中做现货锁定，查询锁定的现货时，OnRspQryStockPosition中各字段的含义如下：

YdPosition： 昨日结算时现货被锁定的仓位。

Position： 当前现货锁仓的数量。

TodayPosition： 当日申请冻结 - 当日申请解冻。

FrozenVolume： 备兑开仓被冻结的仓位。

BuyTradeVolume： 当日申请冻结的数量。

SellTradeVolume： 当日申请解冻的数量。

以上所有数量单位为股。

当前可用于备兑开仓的现货量计算公式为：Position - FrozenVolume。

其余常见问题：

- 行权行看跌期权时，需不需要主动锁定现货？
不需要。发送行权后，柜台会主动冻结现货仓位。
- 行权冻结的仓位，可以通过ReqQryStockPosition查询到吗？
不可以。ReqQryStockPosition只能查询到通过ReqStockLock主动锁定的仓位。

2、OnRspStockLock

```
virtual void OnRspStockLock(CTdStockLockRspField& pStockLock, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
///证券锁定应答
struct CTdStockLockRspField
{
    TdClientNoType      ClientNo;           ///交易编码序号
    TdInstrumentNoType   InstrumentNo;        ///合约序号
    TdLocalOrderNoType   LocalOrderNo;       ///会员内部订单编号
    TdLockType           Locked;              ///锁定标志
    TdVolumeType         Volume;              ///数量
    TdExchangeIdType     ExchangeId;          ///交易所代码
    TdFrontNoType        FrontNo;             ///处理本次报单的实际席位序号
    TdSessionNoType      SessionNo;           ///发送该报单的session
    TdOrdersSysIDType    OrdersSysId;         ///交易所订单编号
};
```

证券锁定应答无OrderSysId。

2.9.3 出入金通知

1、OnRtnWithdrawDeposit

```
virtual void OnRtnWithdrawDeposit(CTdWithdrawDepositRtnField& pwithdrawDeposit) {};
```

出入金推送为服务端主动发起的私有流回报。当柜台发起客户出入金时，api如果订阅了私有流，则会收到出入金通知。

```
///出入金通知
struct CTdWithdrawDepositRtnField
{
    TdFrontNoType        FrontNo;
    TdSequenceNoType     SequenceNo;          ///回报序号(私有流序号每个交易日连续)
    TdAccountIDType      AccountId;           ///资金帐户
    TdClientIdType       ClientId;            ///交易编码
    TdPriceType          Deposit;             ///入金金额
    TdPriceType          Withdraw;            ///出金金额
};
```

2.9.4 合约状态变化通知

1、OnRtnInstrumentStatus

合约状态变化为公有流回报，可以通过Init函数中，设置公有流订阅方式（SubPublicType），来选择接收或者不接收合约状态变化的回报。

```
virtual void OnRtnInstrumentStatus(CTdInstrumentStatusRtnField& pStatus) {};
```

```
///合约状态通知
struct CTdInstrumentStatusRtnField
{
    TdInstrumentNoType    InstrumentNo;
    TdInstrumentIdType     InstrumentId;        ///合约代码
    TdExchangeIdType      ExchangeId;          ///交易所代码
    TdProductIdType       ProductId;           ///产品代码
    TdInstStatusType      InstStatus;          ///合约状态
};
```

2.9.5 股票非交易业务

股票非交易业务相关的方法如下：

```
///股票非交易业务报单请求
virtual int ReqBusinessInsert(CTdBusinessInsertReqField* pStockInsert, int nRequestID) = 0;
///股票非交易业务撤单请求
virtual int ReqBusinessCancel(CTdOrderCancelReqField* pStockCancel, int nRequestID) = 0;
///股票非交易业务报单应答
virtual void OnRspBusinessInsert(CTdBusinessInsertRspField& pstockInsert, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///股票非交易业务撤单应答
virtual void OnRspBusinessCancel(CTdOrderCancelRspField& pstockCancel, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///股票非交易业务委托回报
virtual void OnRtnBusinessOrder(CTdBusinessOrderRtnField& pOrder) {};
```

```
///股票非交易业务成交回报
virtual void OnRtnBusinessTrade(CTdBusinessTradeRtnField& pTrade) {};
```

股票非交易业务支持的业务类型详见ReqBusinessInsert方法说明。

1、ReqBusinessInsert

该方法为股票非交易业务的报单请求。

非交易业务报单错误应答：OnRspBusinessInsert

非交易业务报单正确应答：OnRtnBusinessOrder、OnRtnBusinessTrade

```
///股票非交易业务报单请求
virtual int ReqBusinessInsert(CTdBusinessInsertReqField* pStockInsert, int nRequestID) = 0;
```

```
///股票非交易业务报单请求
struct CTdBusinessInsertReqField
{
    TdBusinessType      BusinessType;           //业务类型
    TdClientNoType      ClientNo;               //交易编码序号
    TdInstrumentIdType   InstrumentId;           //合约代码
    TdLocalOrderNoType   LocalOrderNo;          //会员内部订单编号
    TdSideType           Side;                   //买卖方向
    TdVolumeType         Volume;                 //数量
    TdPriceType          Price;                  //价格
    TdExchangeIdType     ExchangeId;            //交易所代码
    TdInstrumentIdType    DestInstrumentId;      //目标合约代码
};
```

当前支持的非交易业务报单类型：国债逆回购。

各非交易业务的数量单位、价格单位说明如下表，未说明的非交易业务类型表示api暂不支持。

业务类型	业务类型选项	买卖方向	数量单位	价格单位
新股发行	TD_Business_SseIn	-	-	-
配股、科创板配售	TD_Business_SseR1	-	-	-
配转债	TD_Business_SseR4	-	-	-
要约预受	TD_Business_SseFS	-	-	-
要约撤销	TD_Business_SseFC	-	-	-
开放式基金申购	TD_Business_SseOC	-	-	-
开放式基金赎回	TD_Business_SseOR	-	-	-
开放式基金认购	TD_Business_SseOS	-	-	-
开放式基金转托管	TD_Business_SseOT	-	-	-
开放式基金分红设置	TD_Business_SseOD	-	-	-
开放式基金转换	TD_Business_SseOV	-	-	-
余券划转	TD_Business_SseST	-	-	-
还券划转	TD_Business_SseSR	-	-	-
担保品划入	TD_Business_SseCI	-	-	-
担保品划出	TD_Business_SseCO	-	-	-
券源划入	TD_Business_SseSI	-	-	-
券源划出	TD_Business_SseSO	-	-	-
国债逆回购	TD_Business_SseCRP	卖	元	每百元资金到期年收益率

2、ReqBusinessCancel

该方法为股票非交易业务的撤单请求。

非交易业务撤单错误应答：OnRspBusinessCancel

非交易业务撤单正确应答：OnRtnBusinessOrder(推送“已撤单”状态的回报通知)

```
///股票非交易业务撤单请求
virtual int ReqBusinessCancel(CTdOrderCancelReqField* pStockCancel, int nRequestID) = 0;
```

```
///撤单请求
struct CTdOrderCancelReqField
{
    TdSessionNoType      OriSessionNo;          //原始报单的sessionno如果是通过LocalOrderNo撤单，需要填写，不填写则默认撤销本次登录后的LocalOrderNo
    TdLocalOrderNoType    OriLocalOrderNo;        //原始交易客户方订单编号
    TdOrderSysIDType      OrderSysId;             //交易所订单编号(OrigLocalOrderNo和OrderSysId任意一个即可，当OriLocalOrderNo、OrderSysId都填写时，以OriLocalOrderNo为准)
};
```

3、OnRspBusinessInsert

该方法为股票非交易业务报单错误时的应答。

```
///股票非交易业务报单应答
virtual void OnRspBusinessInsert(CTdBusinessInsertRspField& pStockInsert, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///股票非交易业务报单应答
struct CTdBusinessInsertRspField
{
    TdBusinessType      BusinessType;            //业务类型
    TdClientNoType      ClientNo;                //交易编码序号
    TdInstrumentIDType   InstrumentId;            //合约代码
    TdLocalOrderNoType   LocalOrderNo;           //会员内部订单编号
    TdSideType           Side;                   //买卖方向
    TdVolumeType         Volume;                 //数量
    TdPriceType          Price;                  //价格
    TdExchangeIDType     ExchangeId;             //交易所代码
    TdSessionNoType      SessionNo;              //发送该报单的session
    TdOrderSysIDType     OrderSysId;             //交易所订单编号
    TdInstrumentIDType   DestInstrumentId;        //目标合约代码
};
```

4、OnRspBusinessCancel

该方法为股票非交易业务撤单错误时的应答。

```
///股票非交易业务撤单应答
virtual void OnRspBusinessCancel(CTdOrderCancelRspField& pStockCancel, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast) {};
```

```
///撤单应答
struct CTdOrderCancelRspField
{
    TdSessionNoType      OriSessionNo;          //原始报单的sessionno
    TdLocalOrderNoType    OriLocalOrderNo;        //原始交易客户方订单编号
    TdOrderSysIDType      OrderSysId;             //交易所订单编号
    TdSessionNoType      SessionNo;              //发送该报单的session
};
```

5、OnRtnBusinessOrder

该方法为股票非交易业务委托回报，订单成功报入交易所后，推送非交易业务委托回报。

```
///股票非交易业务委托回报
virtual void OnRtnBusinessOrder(CTdBusinessOrderRtnField& pOrder) {};
```

```
//股票非交易业务委托回报
struct CTdBusinessOrderRtnField
{
    TdBusinessType      BusinessType;            //业务类型
    TdSequenceNoType     SequenceNo;              //回报序号(私有流序号每个交易日连续)
    TdRequestNoType      RequestNo;               //报单请求编号
    TdSessionNoType      SessionNo;              //原始报单session
    TdClientNoType      ClientNo;
    TdLocalOrderNoType   LocalOrderNo;
    TdClientIDType       ClientId;
    TdUserIDType         UserId;                  //原始报单交易用户代码
    TdInstrumentIDType    InstrumentId;
    TdExchangeIDType     ExchangeId;              //交易所代码
    TdSideType           Side;                   //买卖方向
    TdOrdStatusType      OrdStatus;              //订单状态
    TdPriceType          Price;                  //价格
    TdVolumeType         Volume;                 //数量
};
```

```
TdIntTimeType      OrderTime;           //委托时间
TdOrderSysIDType   OrdersSysId;         //交易所报单编号
TdVolumeType       LeavesVolume;        //订单剩余数量(交易所返回字段,每个交易所的含义不同)
TdVolumeType       CancelVolume;        //订单撤销数量
TdInstrumentIDType DestInstrumentId;     //目标合约代码
};
```

6、OnRtnBusinessTrade

该方法为股票非交易业务成交回报。

```
///股票非交易业务成交回报
virtual void OnRtnBusinessTrade(CTdBusinessTradeRtnField& pTrade) {};
```

```
///股票非交易业务成交回报
struct CTdBusinessTradeRtnField
{
    TdBusinessType      BusinessType;           //业务类型
    TdSequenceNoType    SequenceNo;             //回报序号(私有流序号每个交易日连续)
    TdRequestNoType     RequestNo;              //报单请求编号
    TdSessionNoType     SessionNo;              //原始报单session
    TdClientNoType      ClientNo;
    TdLocalOrderNoType  LocalOrderNo;
    TdClientIdType      ClientId;
    TdInstrumentIDType  InstrumentId;
    TdExchangeIDType    ExchangeId;             //交易所代码
    TdSideType          Side;                   //买卖方向
    TdPriceType          TradePrice;             //成交价格
    TdVolumeType         TradeVolume;            //成交数量
    TdVolumeType         LeavesVolume;           //本次成交后申报余额数量
    TdIntTimeType        TradeTime;              //成交时间
    TdOrderSysIDType     OrdersSysId;            //交易所报单编号
    TdTradeIDType        TradeId;                //成交编号
    TdUserIDType         UserId;                  //原始报单交易用户代码
    TdInstrumentIDType   DestInstrumentId;       //目标合约代码
};
```

2.9.6 用户验证

用户验证方法为现货系统密码验证功能，第三方系统使用。

```
///用户验证请求
virtual int ReqUserAuthentication(CTdReqUserAuthenField* pReqUserAuthen, int nRequestID) = 0;
```

```
///用户验证应答
virtual void OnRspUserAuthentication(CTdRspUserAuthenField& pUserInfo, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

1、ReqUserAuthentication

ReqUserAuthentication用户验证请求，通过OnRspUserAuthentication来告知用户密码验证是否通过。

```
///用户验证请求
virtual int ReqUserAuthentication(CTdReqUserAuthenField* pReqUserAuthen, int nRequestID) = 0;
```

```
///用户验证请求
struct CTdReqUserAuthenField
{
    TdUserIDType      UserId;           // 交易用户代码
    TdPasswordType    Password;        // 密码
};
```

2、OnRspUserAuthentication

OnRspUserAuthentication为用户密码验证响应，pRspInfo中ErrorId为0标识密码验证通过。

```
///用户验证应答
virtual void OnRspUserAuthentication(CTdRspUserAuthenField& pUserInfo, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast) {};
```

```
///用户登录应答
struct CTdRspUserAuthenField
{
    TdUserIDType      UserId;           // 交易用户代码
};
```

三、demo

下面通过一个简单的代码示例，带大家快速了解下交易API的使用方法。该示例演示了API的初始化、连接、登录、报单和查询。

```
/// 以下为main.cpp

#include "tradeHandler.h"
#include <unistd.h>

int main()
{
    CTraderHandler *pTrade = new CTraderHandler;
    pTrade->Init();           //初始化和连接请求
    sleep(3);                //等待连接
    pTrade->ReqUserLogin();    //用户登录
    sleep(3);
    pTrade->ReqOptionsInsert(); //报单
    // pTrade->ReqStockInsert(); //报单
    pTrade->ReqQryAccount();   //查询资金
    sleep(3);
    pTrade->Release();         //release
    sleep(3);

    return 0;
}

/// 以下为tradeHandler.h

#include "TiTdApi.h"
#include <stdio.h>
#include <string.h>
using namespace titd;

class CTraderHandler : public CTiTdSpi
{
private:
    CTiTdApi *m_pApi;

public:
    // 初始化并连接
    void Init()
    {
        m_pApi = CTiTdApi::CreateTiTdApi();
        m_pApi->RegisterSpi(this);

        CTdConfigInfoField config= {0};
        strcpy(config.UserId, "8810000243");
        config.NicType = 1;
        strcpy(config.SendNicName, "");
        strcpy(config.RecvNicName, "");
        strcpy(config.RecvNicIp, "172.18.18.131");
        config.RecvNicPort = 0;
        config.SubPrivateType = 2;
        config.SubPublicType = 0;
        strcpy(config.FrontAddress, "udp://172.18.18.112:4662");
        config.RecvDataCpuId = -1;
        config.DealDataCpuId = -1;
        config.MaxRecvDataSize = 512;
        config.IsLog = 1;
        config.timeOut = 3;
        m_pApi->Init(config);           //调用init即表示初始化并发送连接请求
    }

    void OnConnected()
    {
        printf("OnConnected\n");
    }

    void OnDisconnected(int nReason)
    {
    }
```

```

    printf("OnDisconnected:%0x\n", nReason);
}

//释放
void Release()
{
    m_pApi->Release();
}

//用户登陆
void ReqUserLogin()
{
    CTdReqUserLoginField req = {0};
    strcpy(req.UserId, "8810000243");
    strcpy(req.Password, "123456");
    strcpy(req.ProtocolInfo, "");
    strcpy(req.UserProductInfo, "");
    strcpy(req.AppId, "client_zqsy_v231009");
    strcpy(req.AppIdIdentify, "GLT1x8SS79");
    m_pApi->ReqUserLogin(&req, 0); //OnConnected连接成功后，再发送客户登录请求
};

///用户登录应答
void OnRspUserLogin(CTdRspUserLoginField& prspUserLogin, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast)
{
    printf("OnRspUserLogin\n");
}

//报单
void ReqOptionsInsert()
{
    int clientno, instrumentno;
    m_pApi->GetClientNo("10000243", clientno);
    m_pApi->GetInstrumentNo("IF1904", instrumentno);

    CTdOrderInsertReqField req = {0};
    req.ClientNo = clientno;
    req.LocalOrderNo = 0;
    req.ExchangeId = TD_Exchange_Cffex;
    req.InstrumentNo = instrumentno;
    req.Side = TD_SIDE_Buy;
    req.OffsetFlag = TD_OC_Open;
    req.Volume = 1;
    req.PriceType = TD_PriceType_Limit;
    req.TimeInForce = TD_TimeInForce_GFD;
    req.HedgeFlag = TD_HF_Speculation;
    req.CoveredOrUncovered = TD_Covered_Uncovered;
    req.VolumeCondition = TD_VC_AV;
    req.TrigCondition = TD_CC_Immediately;
    req.MinVolume = 0;
    req.Price = 3440;
    req.OwnerType = TD_OWNER_Per;
    m_pApi->ReqOptionsInsert(&req, 0);
}

void OnRspOptionsInsert(CTdOrderInsertRspField& poptionsInsert, CTdRspInfoField& prspInfo, int nRequestID, bool bIsLast)
{
    printf("OnRspOptionsInsert - order error!\n");
    printf("errorid:%d errormsg:%s\n",
        prspInfo.ErrorId,
        prspInfo.ErrorMessage);
}

void OnRtnOptionsOrder(CTdOptionsOrderRtnField& porder)
{
    printf("OnRtnOptionsOrder.....\n");
}

void OnRtnOptionsTrade(CTdOptionsTradeRtnField& pTrade)
{
    printf("OnRtnOptionsTrade.....\n");
}

//报单
void ReqStockInsert()
{
    int clientno, instrumentno;
    m_pApi->GetClientNo("10000243", clientno);
    m_pApi->GetInstrumentNo("600000", instrumentno);

    CTdStockInsertReqField req = {0};

```



```
    req.ClientNo = clientno;
    req.InstrumentNo = instrumentno;
    req.LocalOrderNo = 0;
    req.PriceType = TD_PriceType_Limit;
    req.Side = TD_SIDE_Buy;
    req.TimeInForce = TD_TimeInForce_GFD;
    req.Volume = 100;
    req.Price = 2;
    req.OwnerType = TD_OWNER_Per;
    req.ExchangeId = TD_Exchange_Sse;
    req.FrontNo = 0;
    m_pApi->ReqStockInsert(&req, 0);
}

void OnRspStockInsert(CTdStockInsertRspField& pstockInsert, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast)
{
    printf("OnRspStockInsert - order error!\n");
    printf("errorid:%d errormsg:%s\n",
        pRspInfo.ErrorId,
        pRspInfo.ErrorMsg);
}

void OnRtnStockOrder(CTdStockOrderRtnField& porder)
{
    printf("OnRtnStockOrder.....\n");
}

void OnRtnStockTrade(CTdStockTradeRtnField& pTrade)
{
    printf("OnRtnStockTrade.....\n");
}

//查询资金
void ReqQryAccount()
{
    CTdQryAccountField req = {0};
    strcpy(req.AccountId, "8810000243");
    m_pApi->ReqQryAccount(&req, 0);
}

void OnRspQryAccount(CTdRspQryAccountField& pAccount, CTdRspInfoField& pRspInfo, int nRequestID, bool bIsLast)
{
    printf("OnRspQryAccount.....\n");
    printf("accountid:%s, errorid:%d errormsg:%s\n",
        pAccount.AccountId,
        pRspInfo.ErrorId,
        pRspInfo.ErrorMsg);
}

};
```