



# TCore API

Documentation for app developers

# 目录

目录	1
1.简介	3
2.类库文件说明	3
3.系统架构	3
3.1. 整体架构	3
3.2. API 的内部架构	4
4.API 接口开发规则	4
4.1. 开发流程	4
4.2. API Spi 界面	5
4.3.RequestID 字段	5
4.4.连接断开	5
4.5. IsLast 字段	5
4.5. ICERspInfoField 异常讯息结构	5
4.6. API 错误码	6
5.RTCTradeAPI 接口使用说明	6
5.1. RTCTradeAPI 界面	6
5.1.1. CreateRTCTradeAPI	6
5.1.2. Release	6
5.1.3. Join	7
5.1.4. RegisterSpi	7
5.1.5. RegisterFront	7
5.1.6. ReqOrderInsert	7
5.1.7. ReqOrderAction	8
5.1.8. ReqQryOrder	9
5.1.9. ReqQryTrade	10
5.1.10. ReqQryInvestorPosition	10
5.1.11. ReqQryTradingAccount	11
5.1.12. ReqQryInvestorPositionDetail	11
5.1.13. ReqQryExecOrder	12
5.1.14. ReqQryInstrument	12
5.1.15. ReqSmartOrderInsert	13
5.1.16. ReqQryInvestorPositionAnalysis	15
5.1.17. ReqCombActionInsert	16
5.1.18. ReqQryCombAction	17
5.2. RTCTradeAPISpi 界面	17
5.2.1. OnFrontConnected	17
5.2.2. OnFrontDisconnected	17

5.2.3. OnRspUserLogin	18
5.2.4. OnRspUserLogout	18
5.2.5. OnRspOrderInsert	18
5.2.6. OnRspOrderAction	20
5.2.7. OnRspQryOrder	20
5.2.8. OnRspQryTrade	22
5.2.9. OnRspQryInvestorPosition	23
5.2.10. OnRspQryTradingAccount	27
5.2.11. OnRspQryInvestorPositionDetail	29
5.2.12. OnRtnOrder	30
5.2.13. OnRtnTrade	31
5.2.14. OnRspQryInstrument	32
5.2.15. OnRspSmartOrderInsert	33
5.2.16. OnRspQryInvestorPositionAnalysis	34
5.2.17. OnRspQryCombAction	37
<b>6.RTCQuoteAPI 接口使用说明</b>	<b>38</b>
6.1. RTCQuoteAPI 界面	38
6.1.1. CreateRTCQuoteAPI	38
6.1.2. Release	38
6.1.3. Join	38
6.1.4. RegisterFront	38
6.1.5. RegisterSpi	39
6.1.6. SubscribeMarketData	39
6.1.7. UnSubscribeMarketData	39
6.2. RTCQuoteAPISpi 界面	40
6.1.1. OnFrontConnected	40
6.1.2. OnFrontDisconnected	40
6.1.3. OnRtnDepthMarketData	40

## 1.简介

RTCTradeAPI 和 RTCQuoteAPI 接口是基于 C++类实现且仿 CTP 交易系统，可下单、撤单、资金查询、持仓查询、委托查询、成交查询...等功能。

## 2.类库文件说明

文件包含如下：

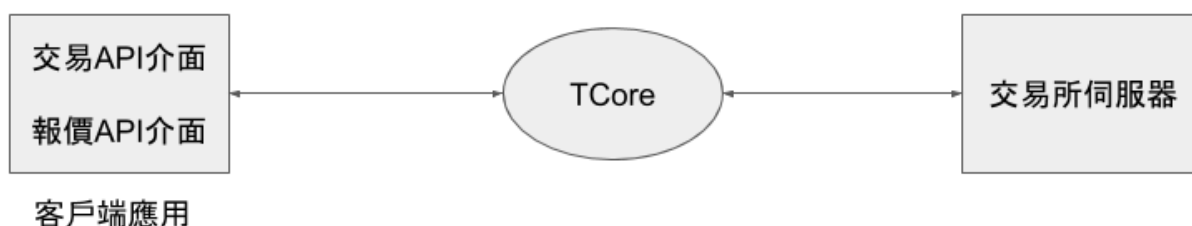
文件名	文件描述
RTCTradeAPI.h	交易接口
RTCQuoteAPI.h	报价界面
RTCUserApiStruct.h	定义接口所需要的数据结构
RTCUserApiDataType.h	定义接口所需要的数据类型
RTCErrCode.h	回传的错误码讯息
RTCQuoteAPI.dll	报价的 lib
RTCTradeAPI.dll	交易的 lib

支持 Microsoft Visual studio 2010 开发环境

API 包括两部分，交易 API(RTCTradeAPI)和报价 API(RTCQuoteAPI)，交易 API 主要用来下单、删单、查询资金、查询持仓...等功能，报价 API 主要用来获取行情信息。

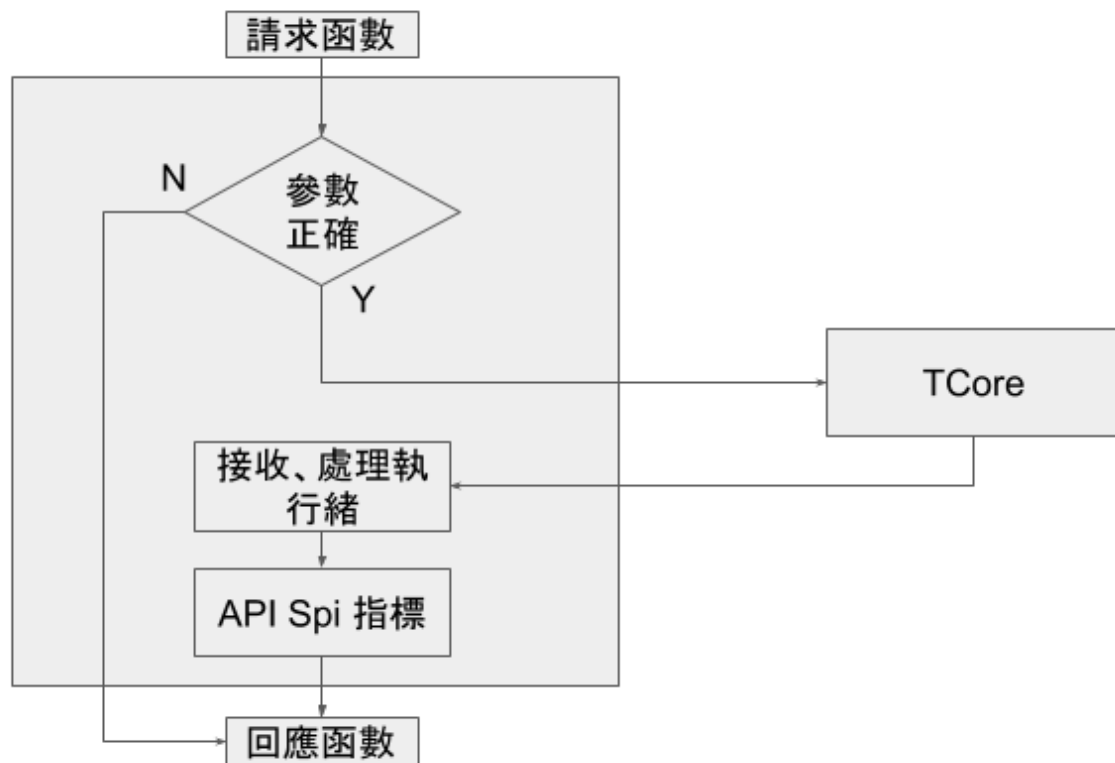
## 3.系统架构

### 3.1. 整体架构



API 接口会把查询、下单信息送到 TCore 后会透过 adapter 送到交易所，后台返回的数据 adapter 会送到 TCore 进行处理过后才会送到 API 接口

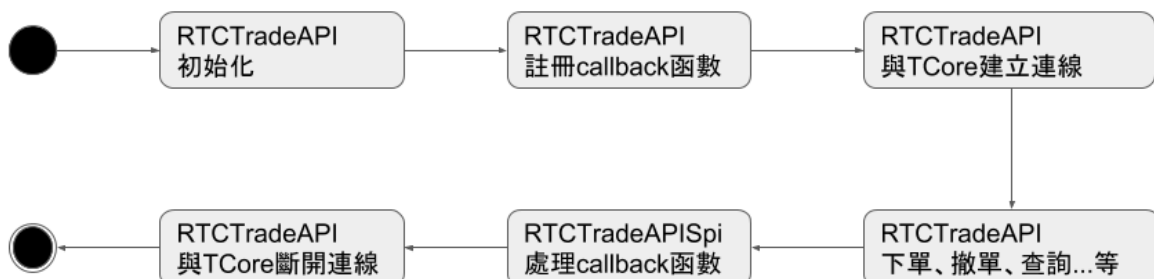
## 3.2. API 的内部架构



RTCTradeAPI 和 RTCQuoteAPI 内部架构都是一样的，都有个 callback 线程在处理 TCore 的回传数据，而客户端在接收每一个 callback 时建议自行维护一个队列，以避免影响后续数据。

## 4.API 接口开发规则

### 4.1. 开发流程



说明:

- 1.在使用 API 功能前, 请务必进行 TCore 连接 RegisterFront()。
- 2.请求和 callback 是在不同的线程, 且收到 callback 时建议自行用到另一个线程中处理, 以避免后面数据阻塞。
- 3.每一个函数入参所需的结构强烈建议清空, 以避免未知的数值出现。
- 4.由于 RTCTradeAPI 和 RTCQuoteAPI 是仿 CTP, 并不是每一个功能都有实现, 而没有实现的功能 API 一律返回-1000, 请依照第五节、第六节有说明的函数进行调用。
- 5.使用查询类型函数时, 参数结构如果带空则全查。
- 6.交易和报价 API 必须要收到 OnFrontConnected() callback 才可做后续的功能。
- 7.使用 API 前, 务必启动 TCore
- 8.中文编码都采用 UTF8
- 9.各请求所需的 InvestorID 同等于 ICERspUserLoginField 结构中 UserID
- 10.不再使用 API 时, 务必调用 Release()

## 4.2. API Spi 界面

RTCTradeAPISpi 和 RTCQuoteAPISpi 接口定义了事件通知, 开发人员务必继承此类, 编写对应的事件处理。

## 4.3.RequestID 字段

由于查询信息都是在不同的线程进行处理, 接口定义了每次请求与响应信息的唯一识别 ID。

## 4.4.连接断开

当与 TCore 联机断开时, OnFrontDisconnected() callback 会通知客户端, 该 API 并不会自动重连。

## 4.5. IsLast 字段

在 callback 中, 当资料有多笔的时候, 读取此字段可以得知是否是最后一笔数据。

## 4.5. ICERspInfoField 异常讯息结构

ICERspInfoField	
ICEErrorIDType	错误码(请参考 RTCErrCode.h)

ICEErrorMsgType	错误的讯息
-----------------	-------

## 4.6. API 错误码

RTCErrCode.h	
错误码	说明
0	正确
-1000	不支援的 API
-1001	数据组件初始化失败
-1002	取资料失败
-1003	商品名称错误
-1004	查询失败

## 5. RTCTradeAPI 接口使用说明

### 5.1. RTCTradeAPI 界面

#### 5.1.1. CreateRTCTradeAPI

创建出的 RTCTradeAPI

函数原型:

```
static RTCTradeAPI *CreateRTCTradeAPI(char *strLogPath)
```

参数:

**strLogPath:** 存放 LOG 档案路径, API 会自动在文件尾加上日期, 未设定预设不产生  
EX:CreateRTCTradeAPI("C:\\RTCTradeAPI.txt")

返回值:

返回一个 RTCTradeAPI 实例指针

### 5.1.2. Release

不再使用本接口对象时，调用该函数删除对象

函数原型：

```
void Release()
```

### 5.1.3. Join

等待接口线程结束运行

函数原型：

```
int Join()
```

返回值：

成功为 0，否则失败

### 5.1.4. RegisterSpi

继承自 callback 接口类的实例

函数原型：

```
void RegisterSpi(RTCTradeAPISpi *pSpi)
```

参数：

pSpi: 指向 callback 指标

### 5.1.5. RegisterFront

连接 TCore

函数原型：

```
LONG RegisterFront(char *strHostAddress, char *strSystemName, char *strServiceKey, int  
iConnectType)
```

参数：

strHostAddress: TCore 地址

strSystemName: TCore 系统名称

strServiceKey: 连结 TCore 所需要的 APP KEY

iConnectType: 固定带 1

返回值：

CONNECT\_RETURN\_FAIL

0



CONNECT_RETURN_CONNECTING	1
CONNECT_RETURN_CONNECTED	2

### 5.1.6. ReqOrderInsert

报单录入请求

函数原型:

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

参数:

```
struct ICEInputOrderField
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType  InvestorID;
    ///合约代码
    ICEInstrumentIDType InstrumentID;
    ///报单价格条件
    ICEOrderPriceTypeType  OrderPriceType;
    ///买卖方向
    ICEDirectionType    Direction;
    ///组合开平标志
    ICECombOffsetFlagType  CombOffsetFlag;
    ///组合投机套保标志
    ICECombHedgeFlagType  CombHedgeFlag;
    ///价格
    ICEPriceType LimitPrice;
    ///数量
    ICEVolumeType    VolumeTotalOriginal;
    ///有效期类型
    ICETimeConditionType    TimeCondition;
    ///止损价
    ICEPriceType StopPrice;
    ///报单引用
    ICEOrderRefType    OrderRef;
    ///成交量类型
    ICEVolumeConditionType  VolumeCondition;
    ///自适应
    ICEBoolType            IsFitOrderFreq;(0 代表不启用, 否则启用)
};
```

返回值:

成功为 0, 否则失败

### 5.1.7. ReqOrderAction

报单操作请求

函数原型:

```
int ReqOrderAction(ICEInputOrderActionField *pInputOrderAction, int nRequestID)
```

参数:

```
struct ICEInputOrderActionField
{
    ///操作标志
    ICEActionFlagType  ActionFlag;
    ///价格
    ICEPriceType LimitPrice;
    ///数量变化
    ICEVolumeType      VolumeChange;
    ///TCORE 报单编号
    ICEOrderSysIDType  ReportID;
    ///改价改量标志
    ICEReplaceExecType ReplaceExecType;
};
```

返回值:

成功为 0，否则失败

备注:

删单的话只需要带入 **ReportID**

### 5.1.8. ReqQryOrder

请求查询报单

函数原型:

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

参数:

```
struct ICEQryOrderField
{
    ///经纪公司代码
    ICEBrokerIDType  BrokerID;
    ///投资者代码
    ICEInvestorIDType InvestorID;
    ///合约代码
```

```
ICEInstrumentIDType InstrumentID;  
///交易所代码  
ICEExchangeIDType ExchangeID;  
///报单编号  
ICEOrderSysIDType OrderSysID;  
};
```

返回值:  
成功为 0, 否则失败

### 5.1.9. ReqQryTrade

请求查询成交

函数原型:  
`int ReqQryTrade(ICEQryTradeField *pQryTrade, int nRequestID)`

参数:  
`struct ICEQryTradeField`  
{  
    ///经纪公司代码  
    ICEBrokerIDType BrokerID;  
    ///投资者代码  
    ICEInvestorIDType InvestorID;  
    ///合约代码  
    ICEInstrumentIDType InstrumentID;  
    ///交易所代码  
    ICEExchangeIDType ExchangeID;  
};

返回值:  
成功为 0, 否则失败

### 5.1.10. ReqQryInvestorPosition

请求查询投资者持仓

函数原型:  
`int ReqQryInvestorPosition(ICEQryInvestorPositionField *pQryInvestorPosition, int nRequestID)`

参数:

```
struct ICEQryInvestorPositionField
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType  InvestorID;
    ///合约代码
    ICEInstrumentIDType InstrumentID;
    ///交易所代码
    ICEExchangeIDType  ExchangeID;
};
```

返回值:

成功为 0，否则失败

### 5.1.11. ReqQryTradingAccount

请求查询资金账户

函数原型:

```
int ReqQryTradingAccount(ICEQryTradingAccountField *pQryTradingAccount, int
nRequestID)
```

参数:

```
struct ICEQryTradingAccountField
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType  InvestorID;
    ///币种代码
    ICECurrencyIDType  CurrencyID;
};
```

返回值:

成功为 0，否则失败

### 5.1.12. ReqQryInvestorPositionDetail

请求查询投资者持仓明细

函数原型:

```
int ReqQryInvestorPositionDetail(ICEQryInvestorPositionDetailField
*pQryInvestorPositionDetail, int nRequestID)
```

参数:

```
struct ICEQryInvestorPositionDetailField
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType  InvestorID;
    ///合约代码
    ICEInstrumentIDType InstrumentID;
    ///交易所代码
    ICEExchangeIDType ExchangeID;
};
```

返回值:

成功为 0，否则失败

### 5.1.13. ReqQryExecOrder

请求查询执行宣告

函数原型:

```
int ReqQryExecOrder(ICEQryExecOrderField *pQryExecOrder, int nRequestID)
```

参数:

```
struct ICEQryExecOrderField
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType  InvestorID;
    ///合约代码
    ICEInstrumentIDType InstrumentID;
    ///交易所代码
    ICEExchangeIDType ExchangeID;
};
```

返回值:

成功为 0，否则失败

### 5.1.14. ReqQryInstrument

请求查询合约

函数原型:

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

参数:

```
struct ICEQryInstrumentField
{
    ///合约代码
    ICEInstrumentIDType InstrumentID;
    ///交易所代码
    ICEExchangeIDType ExchangeID;
    ///合约在交易所的代码
    ICEExchangeInstIDType ExchangeInstID;
};
```

返回值:

成功为 0，否则失败

### 5.1.15. ReqSmartOrderInsert

智慧报单录入请求

函数原型:

```
int ReqSmartOrderInsert(ICEInputSmartOrderField *pInputSmartOrder, int iRequestID)
```

参数:

```
struct ICEInputSmartOrderField
{
    ///经纪公司代码
    ICEBrokerIDType BrokerID;
    ///投资者代码
    ICEInvestorIDType InvestorID;
    ///合约代码
    ICEInstrumentIDType InstrumentID;
    ///报单引用
    ICEOrderRefType OrderRef;
};
```

```

///报单价格条件
ICEOrderPriceTypeType    OrderPriceType;
///买卖方向
ICEDirectionType    Direction;
///组合开平标志
ICECombOffsetFlagType    CombOffsetFlag;
///组合投机套保标志
ICECombHedgeFlagType    CombHedgeFlag;
///价格
ICEPriceType LimitPrice;
///数量
ICEVolumeType    VolumeTotalOriginal;
///有效期类型
ICETimeConditionType    TimeCondition;
///成交量类型
ICEVolumeConditionType    VolumeCondition;
///自适应
ICEBoolType    IsFitOrderFreq;
///洗价
ICEMarkToMarketField    MarkToMarket;
///合成
ICEGroupField    Group;
///大单拆解
ICESliceField Slice;
///追价
ICEChasePriceField ChasePrice;
};

```

返回值：  
成功为 0， 否则失败

```

struct ICEMarkToMarketField
{
    ///洗价类型
    ICEMarkToMarketType    MarkToMarketType;
    ///移动类型
    ICEOrderPriceTypeType    TrailingField;
    ///移动停损跳数
    ICEVolumeType    TrailingTickSize;
    ///触发价
    ICEPriceType TouchPrice;
    ///触发类型

```

```
ICEOrderPriceTypeType    TouchField;
///停损价
ICEPriceType StopPrice;
///追踪的合约
ICEInstrumentIDType ContingentInstrumentID;
};
```

说明:

1.MarkToMarketType 有设定的话, ICEInputSmartOrderField 中 OrderPriceType 可不填写

2.MarkToMarketType 设定为 ICE\_StopOrder、ICE\_StopLimitOrder  
需设定 StopPrice、TouchField, 当 TouchField 价格超过 StopPrice, 即触发下单

3.MarkToMarketType 设定为 ICE\_TrailingStop、ICE\_TrailingStopLimit  
需设定 TrailingTickSize、TrailingField

4.MarkToMarketType 设定为 ICE\_MarketIfTouchedOrder、ICE\_LimitIfTouchedOrder  
需设定 ContingentInstrumentID、TouchPrice、TouchField, 当 ContingentInstrumentID 合约的 TouchField 价格超过 TouchPrice, 即触发下单

5.TrailingField、TouchField 只支持以下设定

///最新价

ICE\_OPT\_LastPrice

///卖一价

ICE\_OPT\_AskPrice1

///买一价

ICE\_OPT\_BidPrice1

```
struct ICEGroupField
{
    ///合成单
    ICEGroupType    GroupType;
    ///合成单编号
    ICEGroupID  GroupID;
};
```

说明:

设定合成单时, 需要下单多笔, 且 GroupID 要一样的编号, 才会有效果

```
struct ICESliceField
```



```
{
    ///拆解类型
    ICESlicedTypeType    Type;
    ///逐笔揭露数量
    ICEVolumeType        DiscloseQty;
    ///逐笔变动比例%
    ICEVolumeType        Variance;
};
```

```
struct ICEChasePriceField
{
    ///每次追价跳数
    ICEVolumeType        ChaseTicks;
    ///追价几次
    ICEVolumeType        ChaseTimes;
    ///每次追价几秒
    ICEVolumeType        ChaseSeconds;
    ///最后下单类型
    ICEOrderBehaviorType LastOrderBehavior;
};
```

### 5.1.16. ReqQryInvestorPositionAnalysis

请求查询投资者持仓分析

函数原型:

```
int ReqQryInvestorPositionAnalysis(ICEQryInvestorPositionAnalysisField
*pQryInvestorPositionAnalysis, int iRequestID)
```

参数:

```
struct ICEQryInvestorPositionAnalysisField
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType   InvestorID;
    ///合约代码
    ICEInstrumentIDType InstrumentID;
};
```

返回值:

成功为 0，否则失败

### 5.1.17. ReqCombActionInsert

申请组合录入请求

函数原型:

```
int ReqCombActionInsert(ICEInputCombActionField *pInputCombAction, int iRequestID)
```

参数:

```
struct ICEInputCombActionField
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType  InvestorID;
    ///合约代码
    ICEInstrumentIDType InstrumentID;
    ///买卖方向
    ICEDirectionType   Direction;
    ///数量
    ICEVolumeType       Volume;
    ///组合指令方向
    ICECombDirectionType CombDirection;
    ///组合类型
    ICECombinationTypeType CombinationType;
};
```

返回值:

成功为 0，否则失败

备注:

组合合约:中间加入&

EX:TC.O.SSE.510050A.202103.C.3.746&TC.O.SSE.510050A.202103.C.3.844

### 5.1.18. ReqQryCombAction

请求查询申请组合

函数原型:

```
int ReqQryCombAction(ICEQryCombActionField *pQryCombAction, int nRequestID)
```

参数:

```
struct ICEQryCombActionField
```

```
{  
    ///经纪公司代码  
    ICEBrokerIDType   BrokerID;  
    ///投资者代码  
    ICEInvestorIDType InvestorID;  
    ///合约代码  
    ICEInstrumentIDType InstrumentID;  
    ///交易所代码  
    ICEExchangeIDType ExchangeID;  
};
```

返回值:  
成功为 0, 否则失败

## 5.2. RTCTradeAPISpi 界面

### 5.2.1. OnFrontConnected

当客户端与 TCore 通信连接时, 该方法被调用

函数原型:  
void OnFrontConnected()

### 5.2.2. OnFrontDisconnected

当客户端与 TCore 通信连接断开时, 该方法被调用。

函数原型:  
void OnFrontDisconnected()

### 5.2.3. OnRspUserLogin

登录请求响应

函数原型:  
void OnRspUserLogin(ICERspUserLoginField \*pRspUserLogin, ICERspInfoField \*pRspInfo, int nRequestID, bool blsLast)

参数:  
struct ICERspUserLoginField  
{

```
///登录成功时间
ICETimeType LoginTime;
///经纪公司代码
ICEBrokerIDType BrokerID;
///用户代码
ICEUserIDType UserID;
///交易系统名称
ICESystemNameType SystemName;
};
```

#### 5.2.4. OnRspUserLogout

注销请求响应

函数原型:

```
void OnRspUserLogout(ICEUserLogoutField *pUserLogout, ICERspInfoField *pRspInfo, int
nRequestID, bool bIsLast)
```

参数:

```
struct ICEUserLogoutField
{
    ///经纪公司代码
    ICEBrokerIDType BrokerID;
    ///用户代码
    ICEUserIDType UserID;
};
```

#### 5.2.5. OnRspOrderInsert

报单录入请求响应

函数原型:

```
void OnRspOrderInsert(ICEInputOrderField *pInputOrder, ICERspInfoField *pRspInfo, int
nRequestID, bool bIsLast)
```

参数:

```
struct ICEInputOrderField
{
    ///经纪公司代码
    ICEBrokerIDType BrokerID;
    ///投资者代码
    ICEInvestorIDType InvestorID;
    ///合约代码
```

```

ICEInstrumentIDType InstrumentID;
///报单价格条件
ICEOrderPriceTypeType    OrderPriceType;
///买卖方向
ICEDirectionType    Direction;
///组合开平标志
ICECombOffsetFlagType    CombOffsetFlag;
///组合投机套保标志
ICECombHedgeFlagType    CombHedgeFlag;
///价格
ICEPriceType LimitPrice;
///数量
ICEVolumeType    VolumeTotalOriginal;
///有效期类型
ICETimeConditionType    TimeCondition;
///止损价
ICEPriceType StopPrice;
///报单引用
ICEOrderRefType    OrderRef;
};

```

备注:

下单失败的错误码

-10 Unknow Error

-11 买卖别不对

-12 复式单商品代码解晰错误

-13 下单账号, 不可下此交易所商品

-14 下单错误, 不支持的价格 或 OrderType 或 TimeInForce

-15 不支援证券下单

-20 联机未建立

-22 价格的 TickSize 错误

-23 下单数量超过该商品的上下限

-24 下单数量错误

-25 价格不能小于和等于 0 (市价类型不会去检查)

## 5.2.6. OnRspOrderAction

报单操作请求响应

函数原型:

```

void OnRspOrderAction(ICEInputOrderActionField *pInputOrderAction, ICERspInfoField
*pRspInfo, int nRequestID, bool blsLast)

```

参数:

```
struct ICEInputOrderActionField
{
    ///操作标志
    ICEActionFlagType  ActionFlag;
    ///价格
    ICEPriceType  LimitPrice;
    ///数量变化
    ICEVolumeType  VolumeChange;
    ///报单引用
    ICEOrderRefType  OrderRef;
    ///TCore 报单编号
    ICEOrderSysIDType  ReportID;
};
```

备注:

删单错误码

- 16 群组虚拟单, 不可删改单
- 17 改单错误, 追价单 不可改量改价
- 18 改单错误, Trailing 不可改量改价

改价改量错误码

- 16 群组虚拟单, 不可删改单
- 17 改单错误, 追价单 不可改量改价
- 18 改单错误, Trailing 不可改量改价
- 19 改单错误, TimeInForce 参数错误
- 21 改单错误, 不支持 spread 改价改量
- 22 价格的 TickSize 错误
- 23 下单数量超过该商品的上下限
- 24 下单数量错误
- 25 价格不能小于和等于 0 (市价类型不会去检查)

### 5.2.7. OnRspQryOrder

请求查询报单响应

函数原型:

```
void OnRspQryOrder(ICEOrderField *pOrder, ICERsplInfoField *pRsplInfo, int nRequestID,
bool blsLast)
```

参数:

```
struct ICEOrderField
{
```

```
///经纪公司代码
ICEBrokerIDType    BrokerID;
///投资者代码
ICEInvestorIDType  InvestorID;
///合约代码
ICEInstrumentIDType InstrumentID;
///用户代码
ICEUserIDType      UserID;
///报单价格条件
ICEOrderPriceTypeType OrderPriceType;
///买卖方向
ICEDirectionType   Direction;
///组合开平标志
ICECombOffsetFlagType CombOffsetFlag;
///组合投机套保标志
ICECombHedgeFlagType CombHedgeFlag;
///价格
ICEPriceType LimitPrice;
///数量
ICEVolumeType      VolumeTotalOriginal;
///有效期类型
ICETimeConditionType TimeCondition;
///止损价
ICEPriceType StopPrice;
///交易所代码
ICEExchangeIDType ExchangeID;
///交易日
ICEDateType TradingDay;
///报单编号
ICEOrderSysIDType OrderSysID;
///报单状态
ICEOrderStatusType OrderStatus;
///今成交数量
ICEVolumeType      VolumeTraded;
///剩余数量
ICEVolumeType      VolumeTotal;
///报单日期
ICEDateType InsertDate;
///委托时间
ICETimeType InsertTime;
///状态信息
ICEErrorMsgType    StatusMsg;
///报单引用
```

```
ICEOrderRefType    OrderRef;
///成交量类型
ICEVolumeConditionType    VolumeCondition;
///报单提交状态
ICEOrderSubmitStatusType    OrderSubmitStatus;
///合约在交易所的代码
ICEExchangeInstIDType    ExchangeInstID;
///TCORE 报单编号
ICEOrderSysIDType    ReportID;
};
```

### 5.2.8. OnRspQryTrade

请求查询成交响应

函数原型:

```
void OnRspQryTrade(ICETradeField *pTrade, ICERsplInfoField *pRsplInfo, int nRequestID,
bool blsLast)
```

参数:

struct ICETradeField

```
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType    InvestorID;
    ///合约代码
    ICEInstrumentIDType    InstrumentID;
    ///用户代码
    ICEUserIDType    UserID;
    ///交易所代码
    ICEExchangeIDType    ExchangeID;
    ///买卖方向
    ICEDirectionType    Direction;
    ///报单编号
    ICEOrderSysIDType    OrderSysID;
    ///开平标志
    ICEOffsetFlagType    OffsetFlag;
    ///价格
    ICEPriceType    Price;
    ///数量
    ICEVolumeType    Volume;
    ///成交时期
```



```
ICEDateType TradeDate;  
///成交时间  
ICETimeType TradeTime;  
///报单引用  
ICEOrderRefType OrderRef;  
///合约在交易所的代码  
ICEExchangeInstIDType ExchangeInstID;  
///TCORE 报单编号  
ICEOrderSysIDType ReportID;  
};
```

### 5.2.9. OnRspQryInvestorPosition

请求查询投资者持仓响应

函数原型:

```
void OnRspQryInvestorPosition(ICEInvestorPositionField *pInvestorPosition,  
ICERspInfoField *pRspInfo, int nRequestID, bool bIsLast)
```

参数:

struct ICEInvestorPositionField

```
{  
    ///合约代码  
    ICEInstrumentIDType InstrumentID;  
    ///经纪公司代码  
    ICEBrokerIDType BrokerID;  
    ///投资者代码  
    ICEInvestorIDType InvestorID;  
    ///持仓多空方向  
    ICEPosiDirectionType PosiDirection;  
    ///投机套保标志  
    ICEHedgeFlagType HedgeFlag; //Unsupported  
    ///持仓日期  
    ICEPositionDateType PositionDate; //Unsupported  
    ///上日持仓  
    ICEVolumeType YdPosition;  
    ///今日持仓  
    ICEVolumeType Position;  
    ///多头冻结  
    ICEVolumeType LongFrozen;  
    ///空头冻结  
    ICEVolumeType ShortFrozen;  
    ///多仓冻结金额
```

ICEMoneyType ///空仓冻结金额	LongFrozenAmount;
ICEMoneyType ///开仓量	ShortFrozenAmount;
ICEVolumeType ///平仓量	OpenVolume;
ICEVolumeType ///开仓金额	CloseVolume;
ICEMoneyType ///平仓金额	OpenAmount;
ICEMoneyType ///持仓成本	CloseAmount;
ICEMoneyType ///上次占用的保证金	PositionCost;
ICEMoneyType ///占用的保证金	PreMargin;
ICEMoneyType ///冻结的保证金	UseMargin;
ICEMoneyType ///冻结的资金	FrozenMargin;
ICEMoneyType ///冻结的手续费	FrozenCash;
ICEMoneyType ///资金差额	FrozenCommission;//Unsupported
ICEMoneyType ///手续费	CashIn;
ICEMoneyType ///平仓盈亏	Commission;
ICEMoneyType ///持仓盈亏	CloseProfit;
ICEMoneyType ///上次结算价	PositionProfit;//Unsupported
ICEPriceType ///本次结算价	PreSettlementPrice;
ICEPriceType ///交易日	SettlementPrice;
ICEDateType ///结算编号	TradingDay;//Unsupported
ICESettlementIDType ///开仓成本	SettlementID;//Unsupported
ICEMoneyType ///交易所保证金	OpenCost;
ICEMoneyType	ExchangeMargin;//Unsupported

```

///组合成交形成的持仓
ICEVolumeType      CombPosition;
///组合多头冻结
ICEVolumeType      CombLongFrozen;
///组合空头冻结
ICEVolumeType      CombShortFrozen;
///逐日盯市平仓盈亏
ICEMoneyType       CloseProfitByDate;
///逐笔对冲平仓盈亏
ICEMoneyType       CloseProfitByTrade;
///今日持仓
ICEVolumeType      TodayPosition;
///保证金率
ICERatioType MarginRateByMoney;
///保证金率(按手数)
ICERatioType MarginRateByVolume;
///执行冻结
ICEVolumeType      StrikeFrozen;
///执行冻结金额
ICEMoneyType       StrikeFrozenAmount;
///放弃执行冻结
ICEVolumeType      AbandonFrozen;
///交易所代码
ICEExchangeIDType  ExchangeID;
///执行冻结的昨仓
ICEVolumeType      YdStrikeFrozen;//Unsupported
///多头持仓数量
ICEVolumeType      SumLongQty;
///空头持仓数量
ICEVolumeType      SumShortQty;
///多头今日持仓
ICEVolumeType      TodayLongQty;
///空头今日持仓
ICEVolumeType      TodayShortQty;
///多头昨日持仓
ICEVolumeType      YdLongQty;
///空头昨日持仓
ICEVolumeType      YdShortQty;
///多头可平仓量
ICEVolumeType      LongAvailable;
///空头可平仓量
ICEVolumeType      ShortAvailable;
///成本均价

```

ICEMoneyType ///开仓价	AvgPrice;
ICEMoneyType ///多头成本均价	OpenPrice;
ICEMoneyType ///空头成本均价	LongAvgPrice;
ICEMoneyType ///多头开仓均价	ShortAvgPrice;
ICEMoneyType ///空头开仓均价	LongOpenPrice;
ICEMoneyType ///逐日浮动盈亏	ShortOpenPrice;
ICEMoneyType ///逐笔浮动盈亏	FloatProfitByDate;
ICEMoneyType ///多逐日浮动盈亏	FloatProfitByTrade;
ICEMoneyType ///空逐日浮动盈亏	LongFloatProfitByDate;
ICEMoneyType ///多逐笔浮动盈亏	ShortFloatProfitByDate;
ICEMoneyType ///空逐笔浮动盈亏	LongFloatProfitByTrade;
ICEMoneyType ///当日盈亏	ShortFloatProfitByTrade;
ICEMoneyType ///市值权益	TodayProfit;
ICEMoneyType ///持仓 Delta	MarketPrice;
ICEGreeksType ///Delta	PosDelta;
ICEGreeksType ///1%Gamma	Delta;
ICEGreeksType ///Theta	Gamma;
ICEGreeksType ///Vega	Theta;
ICEGreeksType ///Rho	Vega;
ICEGreeksType ///Charm	Rho;
ICEGreeksType ///Vanna	Charm;
ICEGreeksType ///Vomma	Vanna;

```

    ICEGreeksType      Vomma;
    ///Speed
    ICEGreeksType      Speed;
    ///Zomma
    ICEGreeksType      Zomma;
    ///时间价值
    ICEGreeksType      TimeValue;
    ///损益
    ICEMoneyType        PNL;
    ///理论损益
    ICEMoneyType        TheoPNL;
    ///组合类型
    ICECombinationTypeType  CombinationType;
};

```

### 5.2.10. OnRspQryTradingAccount

请求查询资金账户响应

函数原型:

```

void OnRspQryTradingAccount(ICETradingAccountField *pTradingAccount,
ICERspInfoField *pRspInfo, int nRequestID, bool bIsLast)

```

参数:

```

struct ICETradingAccountField
{
    ///经纪公司代码
    ICEBrokerIDType      BrokerID;
    ///投资者账号
    ICEAccountIDType     AccountID;
    ///上次质押金额
    ICEMoneyType          PreMortgage;
    ///上次信用额度
    ICEMoneyType          PreCredit;
    ///上次存款额
    ICEMoneyType          PreDeposit;
    ///上次结算准备金
    ICEMoneyType          PreBalance;
    ///上次占用的保证金
    ICEMoneyType          PreMargin;
    ///利息基数
    ICEMoneyType          InterestBase;
    ///利息收入

```

ICEMoneyType ///入金金额	Interest;
ICEMoneyType ///出金金额	Deposit;
ICEMoneyType ///冻结的保证金	Withdraw;
ICEMoneyType ///冻结的资金	FrozenMargin;
ICEMoneyType ///冻结的手续费	FrozenCash;
ICEMoneyType ///当前保证金总额	FrozenCommission;
ICEMoneyType ///资金差额	CurrMargin;
ICEMoneyType ///手续费	CashIn;
ICEMoneyType ///平仓盈亏	Commission;
ICEMoneyType ///持仓盈亏	CloseProfit;
ICEMoneyType ///期货结算准备金	PositionProfit;
ICEMoneyType ///可用资金	Balance;
ICEMoneyType ///可取资金	Available;
ICEMoneyType ///基本准备金	WithdrawQuota;
ICEMoneyType ///信用额度	Reserve;
ICEMoneyType ///质押金额	Credit;
ICEMoneyType ///交易所保证金	Mortgage;
ICEMoneyType ///投资者交割保证金	ExchangeMargin;
ICEMoneyType ///交易所交割保证金	DeliveryMargin;
ICEMoneyType ///保底期货结算准备金	ExchangeDeliveryMargin;
ICEMoneyType ///币种代码	ReserveBalance;
ICECurrencyIDType	CurrencyID;

```

///上次货币质入金额
ICEMoneyType    PreFundMortgageIn;
///上次货币质出金额
ICEMoneyType    PreFundMortgageOut;
///货币质入金额
ICEMoneyType    FundMortgageIn;
///货币质出金额
ICEMoneyType    FundMortgageOut;
///货币质押余额
ICEMoneyType    FundMortgageAvailable;
///可质押货币金额
ICEMoneyType    MortgageableFund;
///特殊产品占用保证金
ICEMoneyType    SpecProductMargin;
///特殊产品冻结保证金
ICEMoneyType    SpecProductFrozenMargin;
///特殊产品手续费
ICEMoneyType    SpecProductCommission;
///特殊产品冻结手续费
ICEMoneyType    SpecProductFrozenCommission;
///特殊产品持仓盈亏
ICEMoneyType    SpecProductPositionProfit;
///特殊产品平仓盈亏
ICEMoneyType    SpecProductCloseProfit;
///根据持仓盈亏算法计算的特殊产品持仓盈亏
ICEMoneyType    SpecProductPositionProfitByAlg;
///特殊产品交易所保证金
ICEMoneyType    SpecProductExchangeMargin;
};

```

### 5.2.11. OnRspQryInvestorPositionDetail

请求查询投资者持仓明细响应

函数原型:

```

void OnRspQryInvestorPositionDetail(ICEInvestorPositionDetailField
*pInvestorPositionDetail, ICERspInfoField *pRspInfo, int nRequestID, bool bIsLast)

```

参数:

```

struct ICEInvestorPositionDetailField
{
    ///合约代码
    ICEInstrumentIDType InstrumentID;

```

```

    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType  InvestorID;
    ///买卖
    ICEDirectionType   Direction;
    ///开仓日期
    ICEDateType    OpenDate;
    ///数量
    ICEVolumeType   Volume;
    ///开仓价
    ICEPriceType    OpenPrice;
    ///交易所代码
    ICEExchangeIDType ExchangeID;
    ///逐日盯市平仓盈亏
    ICEMoneyType    CloseProfitByDate;
    ///逐笔对冲平仓盈亏
    ICEMoneyType    CloseProfitByTrade;
    ///逐日盯市持仓盈亏
    ICEMoneyType    PositionProfitByDate;
    ///逐笔对冲持仓盈亏
    ICEMoneyType    PositionProfitByTrade;
    ///保证金率
    ICERatioType    MarginRateByMoney;
    ///保证金率(按手数)
    ICERatioType    MarginRateByVolume;
    ///昨结算价
    ICEPriceType    LastSettlementPrice;
    ///结算价
    ICEPriceType    SettlementPrice;
    ///平仓量
    ICEVolumeType    CloseVolume;
};

```

### 5.2.12. OnRtnOrder

报单通知

函数原型:

```
void OnRtnOrder(ICEOrderField *pOrder)
```

参数:

```
struct ICEOrderField
```



```
{  
    ///经纪公司代码  
    ICEBrokerIDType    BrokerID;  
    ///投资者代码  
    ICEInvestorIDType  InvestorID;  
    ///合约代码  
    ICEInstrumentIDType InstrumentID;  
    ///用户代码  
    ICEUserIDType      UserID;  
    ///报单价格条件  
    ICEOrderPriceTypeType OrderPriceType;  
    ///买卖方向  
    ICEDirectionType   Direction;  
    ///组合开平标志  
    ICECombOffsetFlagType CombOffsetFlag;  
    ///组合投机套保标志  
    ICECombHedgeFlagType CombHedgeFlag;  
    ///价格  
    ICEPriceType LimitPrice;  
    ///数量  
    ICEVolumeType      VolumeTotalOriginal;  
    ///有效期类型  
    ICETimeConditionType TimeCondition;  
    ///止损价  
    ICEPriceType StopPrice;  
    ///交易所代码  
    ICEExchangeIDType ExchangeID;  
    ///交易日  
    ICEDateType TradingDay;  
    ///报单编号  
    ICEOrderSysIDType OrderSysID;  
    ///报单状态  
    ICEOrderStatusType OrderStatus;  
    ///今成交数量  
    ICEVolumeType      VolumeTraded;  
    ///剩余数量  
    ICEVolumeType      VolumeTotal;  
    ///报单日期  
    ICEDateType InsertDate;  
    ///委托时间  
    ICETimeType InsertTime;  
    ///状态信息  
    ICEErrorMsgType    StatusMsg;
```

```
///报单引用
ICEOrderRefType    OrderRef;
///成交量类型
ICEVolumeConditionType    VolumeCondition;
///报单提交状态
ICEOrderSubmitStatusType    OrderSubmitStatus;
///合约在交易所的代码
ICEExchangeInstIDType    ExchangeInstID;
///TCORE 报单编号
ICEOrderSysIDType    ReportID;
};
```

### 5.2.13. OnRtnTrade

成交通知

函数原型:

```
void OnRtnTrade(ICETradeField *pTrade)
```

参数:

struct ICETradeField

```
{
    ///经纪公司代码
    ICEBrokerIDType    BrokerID;
    ///投资者代码
    ICEInvestorIDType    InvestorID;
    ///合约代码
    ICEInstrumentIDType    InstrumentID;
    ///用户代码
    ICEUserIDType    UserID;
    ///交易所代码
    ICEExchangeIDType    ExchangeID;
    ///买卖方向
    ICEDirectionType    Direction;
    ///报单编号
    ICEOrderSysIDType    OrderSysID;
    ///开平标志
    ICEOffsetFlagType    OffsetFlag;
    ///价格
    ICEPriceType    Price;
    ///数量
    ICEVolumeType    Volume;
    ///成交时期
```

```
ICEDateType TradeDate;  
///成交时间  
ICETimeType TradeTime;  
///报单引用  
ICEOrderRefType OrderRef;  
///合约在交易所的代码  
ICEExchangeInstIDType ExchangeInstID;  
///TCORE 报单编号  
ICEOrderSysIDType ReportID;  
};
```

#### 5.2.14. OnRspQryInstrument

请求查询合约响应

函数原型:

```
void OnRspQryInstrument(ICEInstrumentField *pInstrument, ICERspInfoField *pRspInfo, int  
nRequestID, bool bIsLast)
```

参数:

```
struct ICEInstrumentField  
{  
    ///合约代码  
    ICEInstrumentIDType InstrumentID;  
    ///交易所代码  
    ICEExchangeIDType ExchangeID;  
    ///合约名称  
    ICEInstrumentNameType InstrumentName;  
    ///期权类型  
    ICEOptionsTypeType OptionsType;  
    ///到期日  
    ICEDateType ExpireDate;  
    ///合约数量乘数  
    ICEVolumeMultipleType VolumeMultiple;  
    ///合约在交易所的代码  
    ICEExchangeInstIDType ExchangeInstID;  
    ///合约标识码  
    ICEInstrumentCodeType InstrumentCode;  
    ///执行价  
    ICEPriceType StrikePrice;  
    ///基础商品代码  
    ICEInstrumentIDType UnderlyingInstrID;  
};
```

## 5.2.15. OnRspSmartOrderInsert

智慧报单录入请求响应

函数原型:

```
void OnRspSmartOrderInsert(ICEInputSmartOrderField *pInputSmartOrder,  
ICERspInfoField *pRspInfo, int nRequestID, bool bIsLast)
```

参数:

struct ICEInputSmartOrderField

```
{  
    ///经纪公司代码  
    ICEBrokerIDType    BrokerID;  
    ///投资者代码  
    ICEInvestorIDType  InvestorID;  
    ///合约代码  
    ICEInstrumentIDType InstrumentID;  
    ///报单引用  
    ICEOrderRefType    OrderRef;  
    ///报单价格条件  
    ICEOrderPriceTypeType    OrderPriceType;  
    ///买卖方向  
    ICEDirectionType    Direction;  
    ///组合开平标志  
    ICECombOffsetFlagType    CombOffsetFlag;  
    ///组合投机套保标志  
    ICECombHedgeFlagType    CombHedgeFlag;  
    ///价格  
    ICEPriceType    LimitPrice;  
    ///数量  
    ICEVolumeType    VolumeTotalOriginal;  
    ///有效期类型  
    ICETimeConditionType    TimeCondition;  
    ///成交量类型  
    ICEVolumeConditionType    VolumeCondition;  
    ///自适应  
    ICEBoolType          IsFitOrderFreq;  
    ///洗价  
    ICEMarkToMarketField    MarkToMarket;  
    ///合成  
    ICEGroupField          Group;  
    ///大单拆解  
    ICESliceField    Slice;  
    ///追价
```

```
ICEChasePriceField ChasePrice;  
};
```

## 5.2.16. OnRspQryInvestorPositionAnalysis

请求查询投资者持仓分析响应

函数原型:

```
void OnRspQryInvestorPositionAnalysis(ICEInvestorPositionAnalysisField  
*pInvestorPositionAnalysis, ICERsplInfoField *pRsplInfo, int nRequestID, bool blsLast)
```

参数:

```
struct ICEInvestorPositionAnalysisField  
{  
    ///经纪公司代码  
    ICEBrokerIDType BrokerID;  
    ///投资者代码  
    ICEInvestorIDType InvestorID;  
    ///合约代码  
    ICEInstrumentIDType InstrumentID;  
    ///年份  
    ICEYearType Year;  
    ///月  
    ICEMonthType Month;  
    ///Delta  
    ICEGreeksType Delta;  
    ///当日交易 Delta  
    ICEGreeksType TdDelta;  
    ///昨仓 Delta  
    ICEGreeksType YdDelta;  
    ///Gamma  
    ICEGreeksType Gamma;  
    ///当日交易 Gamma  
    ICEGreeksType TdGamma;  
    ///昨仓 Gamma  
    ICEGreeksType YdGamma;  
    ///Theta  
    ICEGreeksType Theta;  
    ///当日交易 Theta  
    ICEGreeksType TdTheta;  
    ///昨仓 Theta  
    ICEGreeksType YdTheta;  
    ///Vega  
    ICEGreeksType Vega;
```

```

///当日交易 Vega
ICEGreeksType      TdVega;
///昨仓 Vega
ICEGreeksType      YdVega;
///Rho
ICEGreeksType      Rho;
///当日交易 Rho
ICEGreeksType      TdRho;
///昨仓 Rho
ICEGreeksType      YdRho;
///Charm
ICEGreeksType      Charm;
///当日交易 Charm
ICEGreeksType      TdCharm;
///Vanna
ICEGreeksType      Vanna;
///当日交易 Vanna
ICEGreeksType      TdVanna;
///Vomma
ICEGreeksType      Vomma;
///当日交易 Vomma
ICEGreeksType      TdVomma;
///Speed
ICEGreeksType      Speed;
///当日交易 Speed
ICEGreeksType      TdSpeed;
///Zomma
ICEGreeksType      Zomma;
///当日交易 Zomma
ICEGreeksType      TdZomma;
///时间价值
ICEGreeksType      TimeValue;
///PnL
ICEGreeksType      PnL;
///当日交易 PnL
ICEGreeksType      TdPnL;
///昨仓 PnL
ICEGreeksType      YdPnL;
///理论 PnL
ICEGreeksType      TheoPnL;
///当日交易理论 PnL
ICEGreeksType      TdTheoPnL;
///昨仓理论 PnL
ICEGreeksType      YdTheoPnL;

```

```

    ///浮动 PnL
    ICEGreeksType      FloatPnL;
    ///平仓 PnL
    ICEGreeksType      ClosePnL;
    ///认购净仓
    ICEVolumeType      CallOI;
    ///认沽净仓
    ICEVolumeType      PutOI;
    ///多净仓
    ICEVolumeType      LongOI;
    ///空净仓
    ICEVolumeType      ShortOI;
    ///当日交易开仓手数
    ICEVolumeType      TdOpenQty;
    ///当日交易平仓手数
    ICEVolumeType      TdCloseQty;
    ///当日交易买卖净成交
    ICEVolumeType      NetFill;
    ///当日交易总成交
    ICEVolumeType      TotalFill;
    ///昨净留仓
    ICEVolumeType      YdNetPosition;
};

```

### 5.2.17. OnRspQryCombAction

请求查询投资者持仓分析响应

函数原型:

```

void OnRspQryCombAction(ICECombActionField *pCombAction, ICERspInfoField
*pRspInfo, int nRequestID, bool bIsLast)

```

参数:

```

struct ICECombActionField
{
    ///经纪公司代码
    ICEBrokerIDType      BrokerID;
    ///投资者代码
    ICEInvestorIDType    InvestorID;
    ///合约代码
    ICEInstrumentIDType  InstrumentID;
    ///买卖方向
    ICEDirectionType     Direction;
}

```

```
///数量
ICEVolumeType      Volume;
///组合指令方向
ICECombDirectionType CombDirection;
///本地申请组合编号
ICEOrderLocalIDType ActionLocalID;
///交易所代码
ICEExchangeIDType ExchangeID;
///组合状态
ICEOrderActionStatusType ActionStatus;
///交易日
ICEDateType TradingDay;//Unsupported
///组合编号
ICETradeIDType ComTradeID;
///组合类型
ICECombinationTypeType CombinationType;
};
```

## 6. RTCQuoteAPI 接口使用说明

### 6.1. RTCQuoteAPI 界面

#### 6.1.1. CreateRTCQuoteAPI

创建 RTCQuoteAPI

函数原型:

```
RTCQuoteAPI *CreateRTCQuoteAPI(char *strLogPath)
```

参数:

**strLogPath:** 存放 LOG 档案路径, API 会自动在文件尾加上日期, 未设定预设不产生  
EX:CreateRTCQuoteAPI("C:\\RTCQuoteAPI.txt")

返回值:

返回一个 RTCQuoteAPI 实例指针



### 6.1.2. Release

不再使用本接口对象时，调用该函数删除对象

函数原型：

void Release()

### 6.1.3. Join

等待接口线程结束运行

函数原型：

int Join()

返回值：

成功为 0，否则失败

### 6.1.4. RegisterFront

连接 TCore

函数原型：

LONG RegisterFront(char \*strHostAddress, char \*strSystemName, char \*strServiceKey, int iConnectType)

参数：

strHostAddress: TCore 地址

strSystemName: TCore 系统名称

strServiceKey: 连结 TCore 所需要的 APP KEY

iConnectType: 固定带 1

返回值：

成功为 0，否则失败

### 6.1.5. RegisterSpi

继承自 callback 接口类的实例

函数原型：

void RegisterSpi(RTCQuoteAPISpi \*pSpi)

参数：

pSpi: 指向 callback 指标

### 6.1.6. SubscribeMarketData

订阅行情

函数原型:

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

参数:

ppInstrumentID: 要订阅的 TCore symbol

nCount: 订阅 TCore symbol 的数量

返回值:

成功为 0, 否则失败

### 6.1.7. UnSubscribeMarketData

退订行情

函数原型:

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

参数:

ppInstrumentID: 要退订的 TCore symbol

nCount: 退订 TCore symbol 的数量

返回值:

成功为 0, 否则失败

## 6.2. RTCQuoteAPISpi 界面

### 6.1.1. OnFrontConnected

当客户端与 TCore 通信连接时, 该方法被调用

函数原型:

```
void OnFrontConnected()
```

### 6.1.2. OnFrontDisconnected

当客户端与 TCore 通信连接断开时, 该方法被调用。

函数原型:

```
void OnFrontDisconnected()
```

### 6.1.3. OnRtnDepthMarketData

行情通知

函数原型:

```
void OnRtnDepthMarketData(ICEDepthMarketDataField *pDepthMarketData)
```

参数:

struct ICEDepthMarketDataField

```
{  
    ///交易日  
    ICEDateType TradingDay;  
    ///合约代码  
    ICEInstrumentIDType InstrumentID;  
    ///交易所代码  
    ICEExchangeIDType ExchangeID;  
    ///最新价  
    ICEPriceType LastPrice;  
    ///昨收盘  
    ICEPriceType PreClosePrice;  
    ///昨持仓量  
    ICELargeVolumeType PreOpenInterest;  
    ///今开盘  
    ICEPriceType OpenPrice;  
    ///最高价  
    ICEPriceType HighestPrice;  
    ///最低价  
    ICEPriceType LowestPrice;  
    ///数量  
    ICEVolumeType Volume;  
    ///持仓量  
    ICELargeVolumeType OpenInterest;  
    ///今收盘  
    ICEPriceType ClosePrice;  
    ///涨停板价  
    ICEPriceType UpperLimitPrice;  
    ///跌停板价  
    ICEPriceType LowerLimitPrice;  
    ///申买价一  
    ICEPriceType BidPrice1;  
    ///申买量一
```

```
ICEVolumeType      BidVolume1;
///申卖价一
ICEPriceType AskPrice1;
///申卖量一
ICEVolumeType      AskVolume1;
///申买价二
ICEPriceType BidPrice2;
///申买量二
ICEVolumeType      BidVolume2;
///申卖价二
ICEPriceType AskPrice2;
///申卖量二
ICEVolumeType      AskVolume2;
///申买价三
ICEPriceType BidPrice3;
///申买量三
ICEVolumeType      BidVolume3;
///申卖价三
ICEPriceType AskPrice3;
///申卖量三
ICEVolumeType      AskVolume3;
///申买价四
ICEPriceType BidPrice4;
///申买量四
ICEVolumeType      BidVolume4;
///申卖价四
ICEPriceType AskPrice4;
///申卖量四
ICEVolumeType      AskVolume4;
///申买价五
ICEPriceType BidPrice5;
///申买量五
ICEVolumeType      BidVolume5;
///申卖价五
ICEPriceType AskPrice5;
///申卖量五
ICEVolumeType      AskVolume5;
///业务日期
ICEDateType ActionDay;
///上次结算价
ICEPriceType PreSettlementPrice;
///成交金额
ICEMoneyType      Turnover;
```

```
///本次结算价  
ICEPriceType SettlementPrice;  
///最后修改时间  
ICETimeType UpdateTime;  
///合约在交易所的代码  
ICEExchangeInstIDType ExchangeInstID;  
///合约交易状态  
ICEInstrumentStatusType InstrumentStatus;  
};
```