TCore API

Documentation for app developers

2020/09/09

目录

目录 1

1.简介 3

2.类库文件说明 3

3.系统架构 4

3.1.整体架构 4

3.2. API的内部架构 4

4.API介面开发规则 5

4.1 .开发流程 5

4.2. API Spi介面 5

4.3.RequestID栏位 5

4.4.连接断开 6

4.5. IsLast栏位 6

4.5. ICERspInfoField异常讯息结构 6

4.6. API错误码 6

5.RTCTradeAPI介面使用说明 7

5.1. RTCTradeAPI介面 7

5.1.1. CreateRTCTradeAPI 7

5.1.2. Release 7

5.1.3. Join 7

5.1.4. RegisterSpi 7

5.1.5. RegisterFront 8

5.1.6. ReqOrderInsert 8

5.1.7. ReqOrderAction 9

5.1.8. ReqQryOrder 10

5.1.9. ReqQryTrade 10

5.1.10. ReqQryInvestorPosition 11

5.1.11. ReqQryTradingAccount 11

5.1.12. ReqQryInvestorPositionDetail 12

5.1.13. ReqQryExecOrder 12

5.1.14. ReqQryInstrument 13

5.2. RTCTradeAPISpi介面 13

5.2.1. OnFrontConnected 13

5.2. 2. OnFrontDisconnected 13

5.2.3. OnRspUserLogin 14

5.2.4. OnRspUserLogout 14

5.2.5. OnRspOrderInsert 14

5.2.6. OnRspOrderAction 16

5.2.7. OnRspQryOrder 16
5.2.8. OnRspQryTrade 18
5.2.9. OnRspQryInvestorPosition 19
5.2.10. OnRspQryTradingAccount 21
5.2.11. OnRspQryInvestorPositionDetail 23
5.2.12. OnRtnOrder 24
5.2.1 3. OnRtnTrade 25
5.2.14. OnRspQryInstrument 26

6.RTCQuoteAPI介面使用说明 27
6.1. RTCQuoteAPI介面 27
6.1.1. CreateRTCQuoteAPI 27
6.1.2. Release 28
6.1.3. Join 28
6.1.4. RegisterFront 28
6.1.5. RegisterSpi 28
6.1.6. SubscribeMarketData 29
6.1.7. UnSubscribeMarketData 29
6.2. RTCQuoteAPISpi介面 29
6.1.1. OnFrontConnected 29
6.1.2. OnFrontDisconnected 29
6.1.3. OnRtnDepthMarketData 30

1.简介

RTCTradeAPI和RTCQuoteAPI介面是基于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

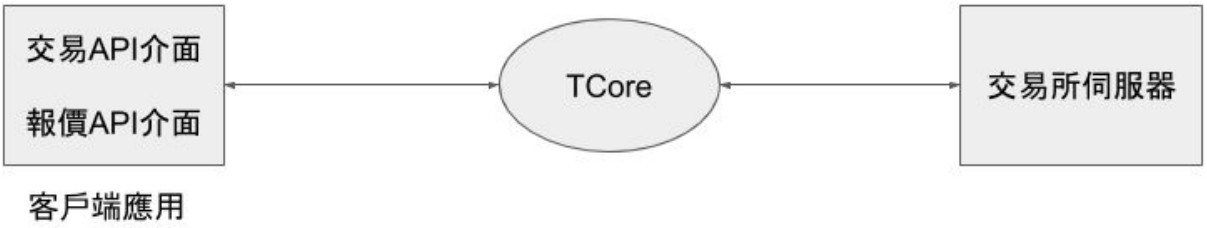
文件包含如下：

文件名	文件描述
RTCTradeAPI.h	交易介面
RTCQuoteAPI.h	报价介面
RTCUserApiStruct.h	定义介面所需要的资料结构
RTCUserApiDataType.h	定义介面所需要的资料类型
RTCErrorCode.h	回传的错误码讯息
RTCQuoteAPI.dll	报价的lib
RTCTradeAPI.dll	交易的lib

支援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主要用来获取行情资讯。

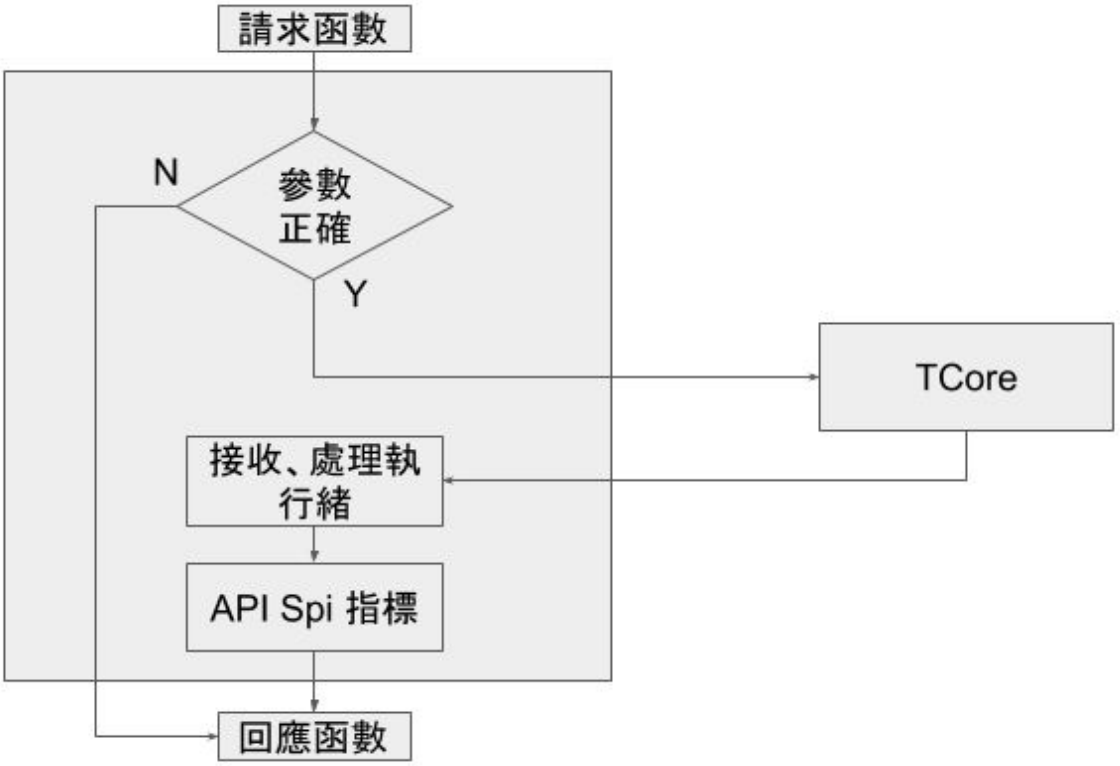
3.系统架构

3.1.整体架构



API介面会把查询、下单资讯送到TCore后会透过adapter送到交易所，后台返回的资料adapter会送到TCore进行处理过后才会送到API介面

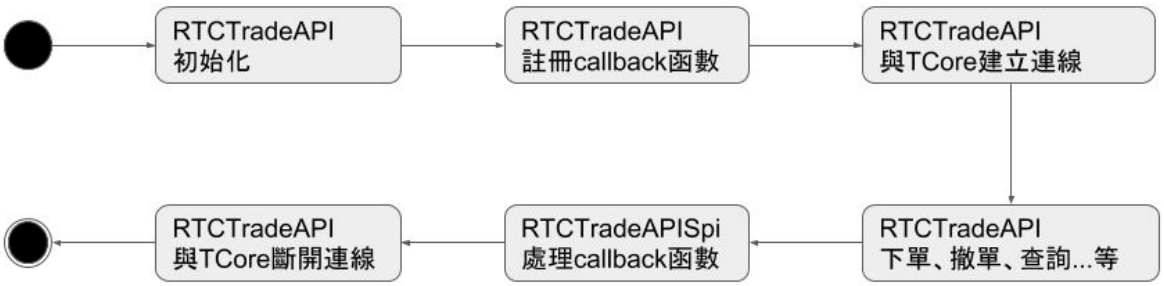
3.2. API的内部架构



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

4.API介面开发规则

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是在不同的执行绪，且收到callback时建议自行用到另一个执行绪中处理，以避免后面资料阻塞。
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

4.2. API Spi介面

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

4.3.RequestID栏位

由于查询资讯都是在不同的执行绪进行处理，介面定义了每次请求与回应资讯的唯一识别ID。

4.4.连接断开

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

4.5. IsLast栏位

在callback中，当资料有多笔的时候，读取此栏位可以得知是否是最后一笔资料。

4.5. ICERspInfoField异常讯息结构

ICERspInfoField	
ICEErrorIDType	错误码(请参考RTCErrorCode.h)
ICEErrorMsgType	错误的讯息

4.6. API错误码

RTCErrorCode.h	
错误码	**说明**
0	正确
-1000	不支援的API
-1001	资料元件初始化失败
-1002	取资料失败
-1003	商品名称错误
-1004	查询失败

5.RTCTradeAPI介面使用说明

5.1. RTCTradeAPI介面

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会自动在档案结尾加上日期，未设定预设不产生
EX:CreateRTCTradeAPI(“C:\\RTCTradeAPI.txt”)

返回值：
返回一个RTCTradeAPI实例指标

5.1.2. Release

不再使用本介面对象时，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介面类的实例

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位址
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 R eqOrderInsert(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;
        ///报单引用
        ICEOrderRefType      OrderRef;
        ///TCore报单编号
        ICEOrderSysIDType    ReportID;
};
```

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. Re qQryTradingAccount

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

5.2. RTCTradeAPISpi介面

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 bIsLast)

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

登出请求响应

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下单帐号,不可下此交易所商品
-14下单错误,不支援的价格或OrderType或TimeInForce
-15不支援证券下单
-20连线未建立
-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 bIsLast)
```

```
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,不支援sprea d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, ICERspInfoField *pRspInfo, int nRequestID,
bool bIsLast)
```

```
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     Sta tusMsg;
        ///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, ICERspInfoField *pRspInfo, int nRequestID,
bool bIsLast)
```

```
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 OnRspQryInvestorPositio n(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
ICEPosiDirectionTypePosiDirection;
///上日持仓
ICEVolumeType	YdPosition;
///今日持仓
ICEVolumeType	Position;
///多头冻结
ICEVolumeType	LongFrozen;
///空头冻结
ICEVolumeType	ShortFrozen;
///开仓冻结金额
ICEMoneyType	LongFrozenAmount;
///开仓冻结金额
ICEMoneyType	ShortFrozenAmount;
///开仓量
ICEVolumeType	OpenVolume;
///平仓量
ICEVolumeType	CloseVolume;
///开仓金额
ICEMoneyType	OpenAmount;
///平仓金额
ICEMoneyType	CloseAmount;
///持仓成本
ICEMoneyType	PositionCost;
///上次占用的保证金
ICEMoneyType	PreMargin;
///占用的保证金
ICEMoneyType	UseMargin;
///冻结的保证金
ICEMoneyType	FrozenMargin;
///冻结的资金
ICEMoneyType	FrozenCash;
///资金差额
ICEMoneyType	CashIn;
///手续费
ICEMoneyType	Commission;
///平仓盈亏
ICEMoneyType	CloseProfit;
///上次结算价
ICEPriceType PreSettlementPrice;
///本次结算价
```

```
        ICEPriceType SettlementPrice;
        ///开仓成本
        ICEMoneyT ype       OpenCost;
        ///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;
        ///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;
        ///今日持仓
        ICEVolumeType       TodayPosition;
};
```

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;
        ///投资者帐号
        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        Interest;
///入金金额
ICEMoneyType        Deposit;
///出金金额
ICEMoneyType        Withdraw;
///冻结的保证金
ICEMoneyType        FrozenMargin;
///冻结的资金
ICEMoneyType        FrozenCash;
///冻结的手续费
ICEMoneyType        FrozenCommission;
///当前保证金总额
ICEMoneyType        CurrMargin;
///资金差额
ICEMoneyType        CashIn;
///手续费
ICEMoneyType        Commission;
///平仓盈亏
ICEMoneyType        CloseProfit;
///持仓盈亏
ICEMoneyType        PositionProfit;
///期货结算准备金
ICEMoneyType        Balance;
/ //可用资金
ICEMoneyType        Available;
///可取资金
ICEMoneyType        WithdrawQuota;
///基本准备金
ICEMoneyType        Reserve;
///信用额度
ICEMoneyType        Credit;
///质押金额
ICEMoneyType        Mortgage;
```

```
        ///交易所保证金
        ICEMoneyType        ExchangeMargin;
        / //投资者交割保证金
        ICEMoneyType        DeliveryMargin;
        ///交易所交割保证金
        ICEMoneyType        ExchangeDeliveryMargin;
        ///保底期货结算准备金
        ICEMoneyType        ReserveBalance;
        ///币种代码
        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
        ICEMone yType       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        CloseProfitByTr ade;
        ///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   R eportID;
};
```

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;
};
```

6.RTCQuoteAPI介面使用说明

6.1. RTCQuoteAPI介面

6.1.1. CreateRTCQuoteAPI

创建RTCQuoteAPI

函数原型：
RTCQuoteAPI *CreateRTCQuoteAPI()

返回值：
返回一个RTCQuoteAPI 实例指标

6.1.2. Release

不再使用本介面对象时，调用该函数删除对象

函数原型：
void Release()

6.1.3. Join

等待介面执行绪结束运行

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：
strHos tAddress：TCore位址
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介面类的实例

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介面

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 Ask Price3;
///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;
};
```