SUNGARD 金仕达

期 货 交 易
API 编程手册

文档标识

文档名称	金仕达期货交易 API 编程手册
版本号	<V1.6>
状况	○ 草案 ○ 评审过的 ○ 更新过的 ◉ 定为基线的

文档修订历史

版本	日期	描述	文档所有者
V1.0	<2011-08-15>	1. 兼容 CTP api 接口的相关字段 2. 头文件请使用命名空间 namespace KingstarAPI 3. 增加查投资者开盘前持仓、组合持仓明细查询接口	沈明明
V1.1	<2012-05-08>	1. 标出“保留字段”(紫色字体) 2. 组合单暂不支持、预埋单暂不支持、UDP 行情暂不支持、银行推送暂不支持	沈明明
V1.2	<2012-05-18>	1. 文档名称标准化	沈明明
V1.3	<2012-09-06>	1. 支持 UDP 行情 2. 增加指定本地文件路径接口	沈明明
V1.4	<2012-11-26>	1， 支持黄金夜市、仓单折抵 2， API 支持 linux 环境下开发	胡旭东
V1.5	<2013-01-07>	支持 RegisterNameServer 接口	胡旭东
V1.6	<2013-03-06>	1，支持批量撤单接口 2，支持条件单接口	胡旭东

此版本文档的正式核准

姓名	签字	日期

分发控制

副本	接受人	机构

目 录

1. 第一章 介绍

1.1 概述

本手册为 Kingstar API 与 Kingstar 服务器之间的通讯数据包接口规范。通过 API，投资者可以接收来自上交所，大商所、郑商所和中金所的行情数据，并发送交易指令，接收相应的反馈和交易状态等信息。Kingstar API 将实现兼容上期技术综合交易平台 CTP 的 API。

1.2 API 文件

Kingstar 平台上使用的 API 是一个基于 C++的类库，通过使用和扩展类库提供的接口来实现实现客户端和 Kingstar 服务器之间的数据传输。通过 API，客户端可以发出或撤销普通单、查询委托或交易状态、查询账户实时信息和交易头寸。API 程序库根据不同的操作系统平台所包含文件有所不同：

Windows 平台 API 包含文件如下：

库文件名	库文件描述
KSTraderApiEx.h	交易接口头文件
KSTradeAPI.h	
KSMdApiEx.h	行情接口头文件
KSMarketDataAPI.h	
KSUserApiDataTypeEx.h	定义了 API 所需的一系列数据类型的头文件
KSUserApiStructEx.h	定义了一系列业务相关的数据结构的头文件
KSCosApi.h	条件单接口头文件
KSCosApiDataType.h	定义了条件单 API 所需的一系列数据类型的头文件
KSCosApiStruct.h	定义了一系列条件单业务相关的数据结构的头文件
KSTradeAPI.lib	交易 API 导入库文件

KSTradeAPI.dll	交易 API 动态链接库
KSMarketDataAPI.lib	行情 API 导入库文件
KSMarketDataAPI.dll	行情 API 动态链接库
lkcdll.dll	交易行情 API 运行需附加的动态链接库
ksPortalAPI.dll	门户集成功能接口库
SSPXEncode.dll	Sspx 协议编码接口库
KSInterB2C.lkc	客户端授权文件

Linux 平台 API 包含文件如下：

库文件名	库文件描述
KSTraderApiEx.h	交易接口头文件
KSTradeAPI.h	
KSMdApiEx.h	行情接口头文件
KSMarketDataAPI.h	
KSUserApiDataTypeEx.h	定义了 API 所需的一系列数据类型的头文件
KSUserApiStructEx.h	定义了一系列业务相关的数据结构的头文件
KSCosApi.h	条件单接口头文件
KSCosApiDataType.h	定义了条件单 API 所需的一系列数据类型的头文件
KSCosApiStruct.h	定义了一系列条件单业务相关的数据结构的头文件
libkstradeapi.so	交易 API 动态链接库
libksmarketdataapi.so	行情 API 动态链接库
Libkslkc64r.so 、libkslkc32r.so	授权文件验证动态链接库
KSInterB2C.lkc	客户端授权文件

注：编程请使用头文件命名空间 namespace KingstarAPI。

条件单头文件前缀“KSCos”术语解释为： kingstar condition order system

2. 第二章 体系结构

2.1 通讯模式

金仕达的 API 和 Kingstar 服务器之间使用的通讯协议是期货交易数据协议（futures TradingData Exchange protocol ，FTD），它基于 TCP 协议。

在 FTD 协议中，通讯模式包括以下三种模式：

对话模式，客户端给 Kingstar 服务器发送请求，该请求被交易所端接收和处理，并给予响应。例如报单、查询等。这种通讯模式与普通的客户/服务器模式相同。

私有模式，Kingstar 服务器把特定的私人信息发送给对应的客户端，包括持仓信息、交易确认信息等。

广播模式，Kingstar 服务器将把公告等信息发送给所有的注册用户。

通讯模式和网络的连接不一定存在简单的一对一的关系。也就是说，一个网络连接中可能传送多种不同通讯模式的报文，一种通讯模式的报文也可以在多个不同的连接中传送。

2.2 数据流

Kingstar 服务器提供了对话、私有、广播等三种通讯模式。

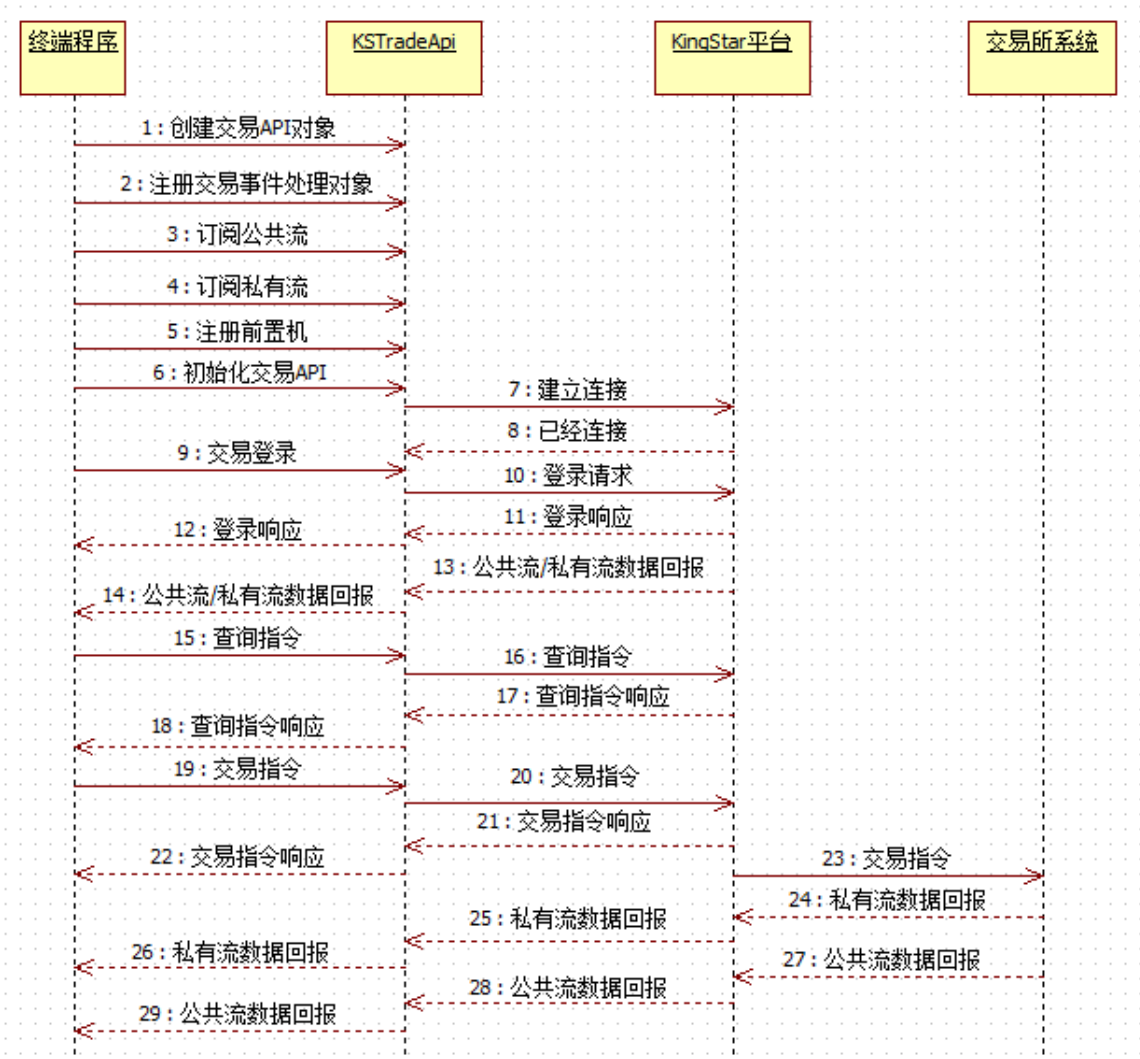
在对话模式中，传输的是对话数据流和查询数据流。对话和查询数据流是双向数据流，客户端发出请求，Kingstar 服务器返回结果。Kingstar 服务器并不保存对话和查询数据流。当故障发生时，比如连接中断后又重新连接，对话和查询数据流将会重置，通讯途中的数据可能会丢失。

在私人通讯模式中，传输的是私人数据流。私人数据流是单向数据流，Kingstar 服务器就是利用它来把相应的私人信息发给提出申请的客户端。私人信息包括，风险提示、指令状态、指令确认、交易确认等。私人数据流是可靠的。当客户端和 Kingstar 服务器失去连接后，在同一交易日的任何时间，客户端都可重新连上 Kingstar 服务器，获取一系列指定的私人信息而不用担心这些数据会丢失。

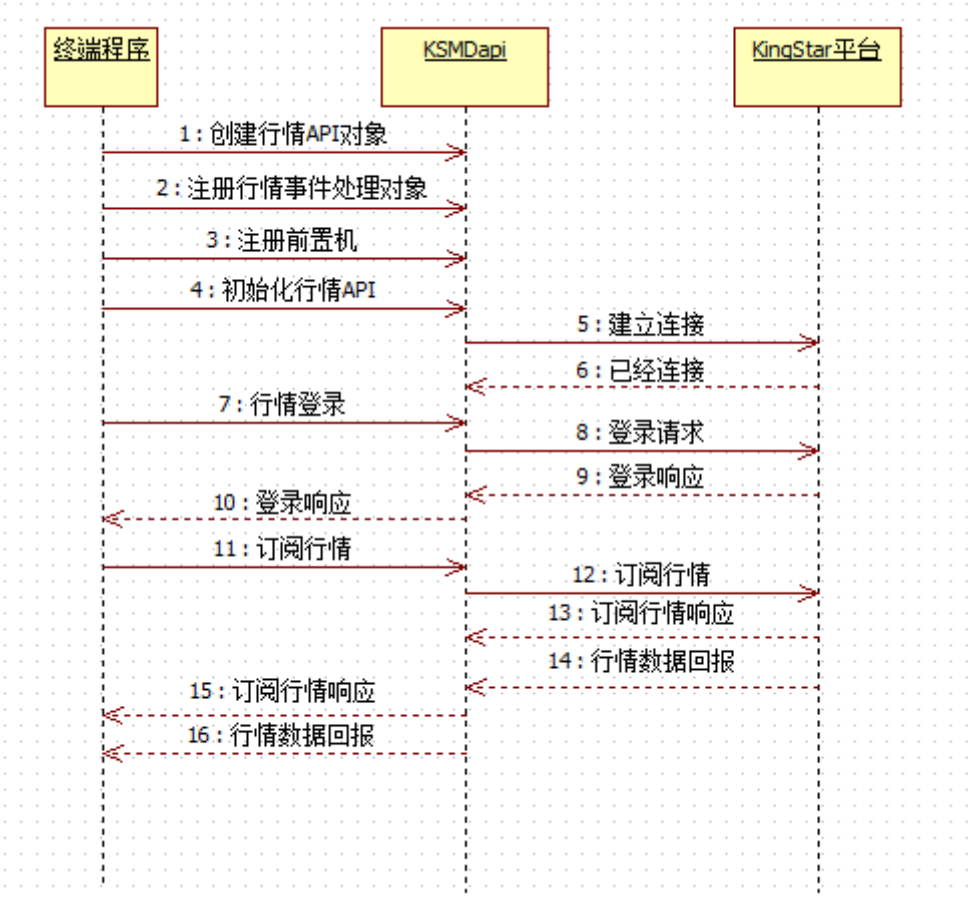
在广播通讯模式中，传输的是公共数据流。和私人数据流一样，它是单向数据流，而且可靠。区别在于，广播通讯数据会发送到所有连接的客户端上。它的主要用途就是发布公共设备的状态信息或重要的公共信息。

2.3 通讯流程

2.3.1 交易流程

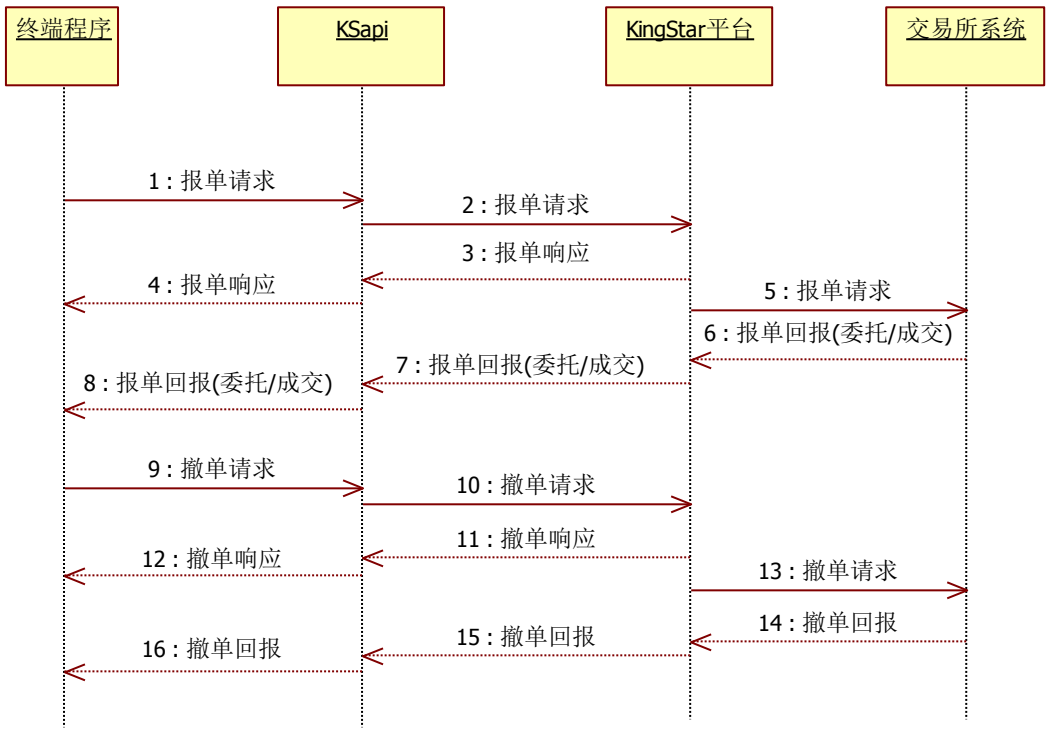


2.3.2 行情流程


终端程序
KSMDapi
KingStar平台
1：创建行情API对象
2：注册行情事件处理对象
3：注册前置机
4：初始化行情API
5：建立连接
6：已经连接
7：行情登录
8：登录请求
9：登录响应
10：登录响应
11：订阅行情
12：订阅行情
13：订阅行情响应
14：行情数据回报
15：订阅行情响应
16：行情数据回报

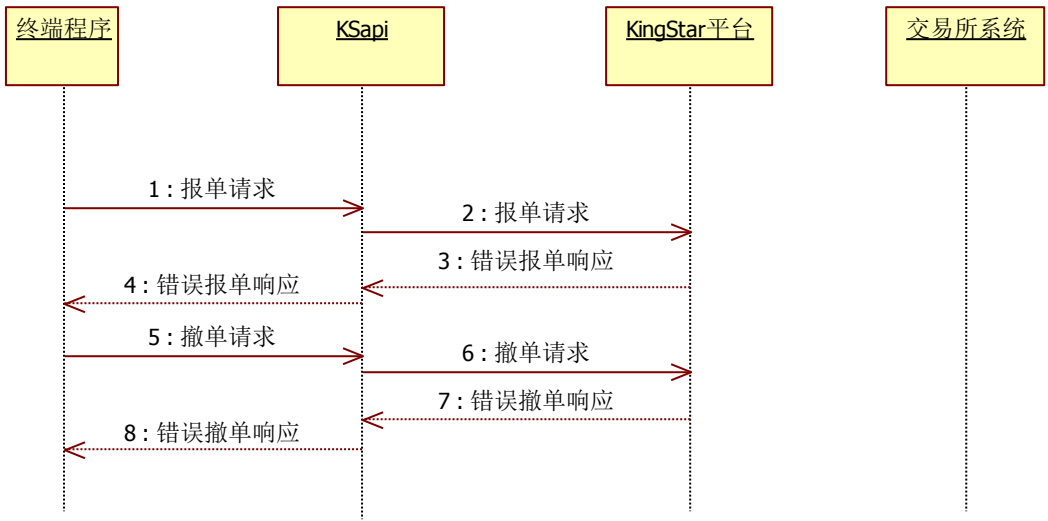
2.4 报单指令（特别说明）

2.4.1 指令正常

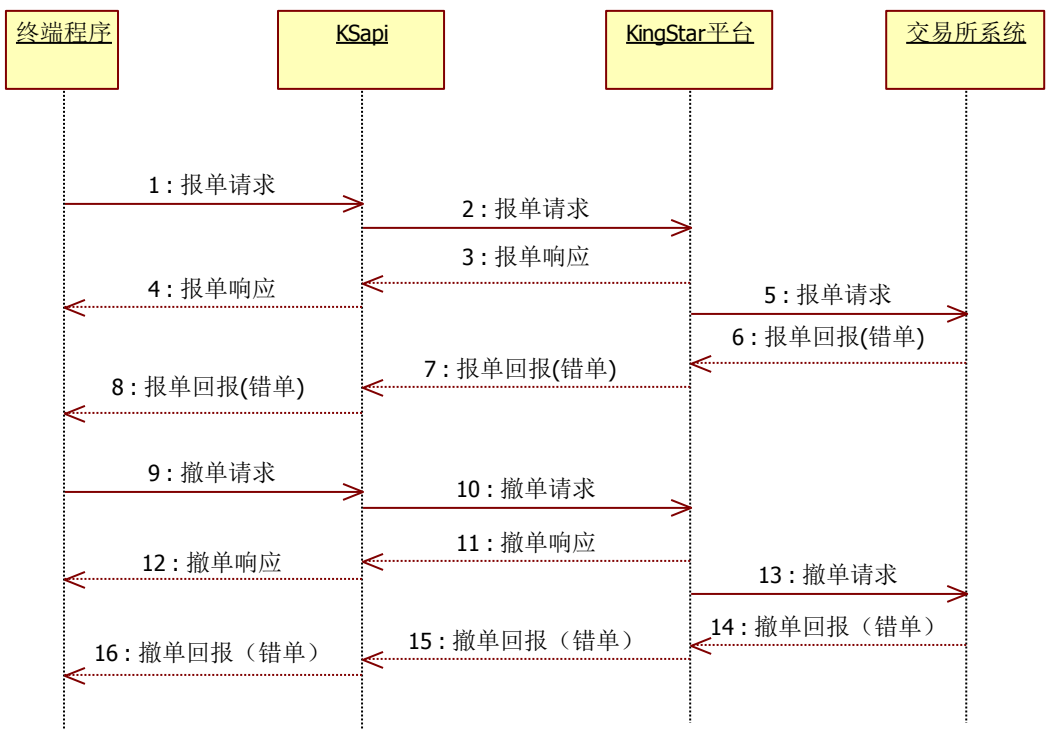


2.4.2 指令异常

当终端程序发出报单、撤单指令错误时，分两种情况：图（一）显示指令填写错误，Kingstar 平台会在送交交易所之前进行检测给出错误提示；图（二）显示交易所系统对所发出的指令进行处理，当不符合交易条件时，交易所给出错误提示。


终端程序
KSapi
KingStar平台
交易所系统
1：报单请求
2：报单请求
3：错误报单响应
4：错误报单响应
5：撤单请求
6：撤单请求
7：错误撤单响应
8：错误撤单响应

（一）


终端程序
KSapi
KingStar平台
交易所系统
1：报单请求
2：报单请求
3：报单响应
4：报单响应
5：报单请求
6：报单回报(错单)
7：报单回报(错单)
8：报单回报(错单)
9：撤单请求
10：撤单请求
11：撤单响应
12：撤单响应
13：撤单请求
14：撤单回报（错单）
15：撤单回报（错单）
16：撤单回报（错单）

（二）

3. 第三章 接口模式

Kingstar 的交易 API 提供了两种编程接口：CThostFtdcTraderApi 与 CThostFtdcTraderSpi。行情 API 提供的是 CThostFtdcMdApi 与 CThostFtdcMdSpi 等两种编程接口。这四个接口对 FTD 协议进行了封装，方便客户端应用程序的开发。客户端可使用 CThostFtdcXXXApi 来发出操作请求，并通过继承 CThostFtdcXXXSpi 并重载回调函数来处理 Kingstar 服务器的响应。

3.1 对话模式的编程接口

对话模式的通讯函数常常这样定义：

请求: *int CThostFtdcTraderApi::ReqXXX(*
*CThostFtdcXXXField *pReqXXX,*
int nRequestID)
int CThostFtdcMDApi::ReqXXX(
*CThostFtdcXXXField *pReqXXX,*
int nRequestID)

响应: *void CThostFtdcTraderSpi::OnRspXXX(*
*CThostFtdcXXXField *pRspXXX,*
*CthostFtdcRspInfoField *pRspInfo,*
int nRequestID,
bool bIsLast)
void CThostFtdcMDSpi::OnRspXXX(
*CThostFtdcXXXField *pRspXXX,*
*CthostFtdcRspInfoField *pRspInfo,*
int nRequestID,
bool bIsLast)

其中请求接口第一个参数为请求的内容，不能为空。

第二个参数为请求号。请求号由客户端应用程序负责维护，正常情况下每个请求的请求号不要重复。在接收 Kingstar 服务器的响应时，可以得到当时发出请求时填写的请求号，从而可以将响应与请求对应起来。

当客户端收到 Kingstar 服务器发出的反馈时，回调函数 *CThostFtdcXXXSpi* 被激活。如果反馈多于一条，回调函数 *CThostFtdcXXXSpi* 会反复被激活，直到数据全部接收完毕。

回调函数的第一个参数为响应的具体数据，如果出错或没有结果有可能为 NULL。

第二个参数为处理结果，表明本次请求的处理结果是成功还是失败。在发生多次回调时，除了第一次回调，其它的回调该参数都可能为 NULL。

第三个参数为请求号，即原来发出请求时填写的请求号。

最后一个参数是结束标识，其值为“true”时表示进行的是该请求的最后一个反馈。

3.2 私有模式的编程接口

下面的例子为私有模式的常用接口：

*void CThostFtdcTraderSpi::OnRtnXXX(CThostFtdcXXXField *pXXX)*

*void CThostFtdcTraderSpi::OnErrRtnXXX(CThostFtdcXXXField *pXXX,*
*CThostFtdcRspInfoField *pRspInfo)*

在私有模式里并没有连接 API 与 Kingstar 服务器的行情函数。当 Kingstar 服务器发出私有数据流时，CThostFtdcTradeSpi 的回调函数将被激活。所有回调函数的第一个参数都是 Kingstar 服务器返回的具体内容。*OnErrRtnCThostFtdcTradeSpi* 函数的第二个参数是报错时详细的错误信息。

3.3 广播模式的编程接口

使用广播模式时，客户端可以用以下两种方式与 Kingstar 服务器进行通讯：

void CThostFtdcTraderSpi::OnRtnInstrumentStatus(
*CThostFtdcInstrumentStatusField *pInstrumentStatus)*

void CThostFtdcTraderSpi::OnRtnDepthMarketData(
*CThostFtdcDepthMarketDataField *pDepthMarketData)*

回调函数 *OnRtnInstrumentStatus* 用于通知客户端合约状态的变化。

回调函数 *OnRtnDepthMarketData* 用于公布最新的交易所行情数据。

4. 第四章 开发接口

4.1 工作线程

客户端进程需要两种线程，一是应用程序主线程，另一种是交易 API 工作线程。如果客户端想要接收行情数据，那么也需要行情 API 工作线程。API 工作线程连接了客户端和 Kingstar 服务器。

交易 API 和行情 API 是安全线程，客户端的应用程序可以同时使用两种或多种的工作线程，而无须担心线程冲突。客户端的应用程序要能尽快的处理回调信息，这样才能避免未处理的回调信息堵塞工作线程。要避免通讯堵塞，客户端的应用程序需要使用缓冲层来储存从 Kingstar 服务器接收的数据，当然缓冲层也可以用来保护客户端自有数据，以便使之与 API 的数据区分开。

4.2 通用规则

客户端应用程序需要经过两步才能连接到 Kingstar 服务器：初始化与功能启用。

使用交易 API，客户端交易应用程序需要编写：

（1）创建一个“CThostFtdcTraderApi”实例。[可调用 SetWritablePath 指定本地文件存储路径]

（2）创建一个处理来自“CThostFtdcTraderSpi”接口的事件处理器，然后使用“CThostFtdcTraderApi”的“RegisterSpi”函数记录下这些事件。

（3）使用“CThostFtdcTraderApi”的“SubscribePrivateTopic”函数处理私有数据流。

（4）使用“CThostFtdcTraderApi”的“SubscribePublicTopic”函数处理公共数据流。

（5）使用“CThostFtdcTraderApi”的“RegisterFront”函数记录 Kingstar 服务器的前端地址。客户端多运行几次这种函数，以便与服务器建立更可靠的联系。强烈建议。

（6）使用“CThostFtdcTraderApi”的“Init“函数来连接 Kingstar 服务器。

（7）Kingstar 服务器连上之后，“CThostFtdcTraderSpi”接口的回调函数“OnFrontConnected”将被激活。函数运行过程中，客户端的应用程序需要使用“CThostFtdcTraderApi”的“ReqUserLogin”函数来提交“login”请求。

(8）当 Kingstar 服务器确认登陆成功后，“CThostFtdcTraderSpi”接口回调函数“OnRspUserLogin”将被激活。

（9）这样，客户端与 Kingstar 服务器的通讯就建立起来了。客户端交易应用程序可以使用其他金仕达 API 业务来与 Kingstar 服务器进行通讯。

使用行情 API，客户端行情应用程序需要编写：

（1）创建一个“CThostFtdcMdApi”实例。[可调用 SetWritablePath 指定本地文件存储路径]

（2）创建一个处理来自“CThostFtdcMdSpi”接口的事件处理器，然后使用“CThostFtdcMdApi”的“RegisterSpi”函数记录下这些事件。

（3）使用“CThostFtdcTraderApi”的“Init“函数来连接 Kingstar 服务器。

（4）Kingstar 服务器连上之后，“CThostFtdcTraderSpi”接口的回调函数“OnFrontConnected”将被激活。函数运行过程中，客户端的应用程序需要使用“CThostFtdcMdApi”的“ReqUserLogin” 函数来提交“login”请求。

(5）当 Kingstar 服务器确认登陆成功后，“CThostFtdcMdSpi”接口回调函数“OnRspUserLogin”将被激活。

（6）这样，客户端与 Kingstar 服务器的通讯就建立起来了，客户端行情应用程序就可以进行行情的订阅与退订了。

使用条件单 API，客户端交易应用程序需要编写：

（1） 通过继承条件单头文件中的 CTKSCosSpi 接口类，实现条件单回调实例类。

（2） 条件单的注册，需要交易 API 的创建。故首先要创建一个“CThostFtdcTraderApi”实例

（3） 通过条件单回调实例类，定义条件单应答回调实例。

（4） 通过交易 API 的 LoadExtApi 请求接口注册条件单。此函数返回指向条件单 API 实例的指针。

（5） 调用条件单 API 接口中的函数。（注意，条件单指令的调用需要在交易登陆之后进行，否则将会出现客户尚未登陆的报错信息）。

其他注意事项：

（1）API 请求的输入参数不能为 NULL；

（2）API 请求的返回参数，0 表示正确，其他表示返回错误。

4.3 CThostFtdcTraderSpi 接口

CThostFtdcTraderSpi 实现了事件通知接口。用户必需派生 CThostFtdcTraderSpi 接口，编写事件处理方法来处理感兴趣的事件。

4.3.1 OnFrontConnected 方法

当客户端与 Kingstar 服务器建立起通信连接时（还未登录前），该方法被调用。

函数原型：

void OnFrontConnected()；

本方法在完成初始化后调用，可以在其中完成用户登录任务。

4.3.2 OnFrontDisconnected 方法

当客户端与 Kingstar 服务器通信连接断开时，该方法被调用。当发生这个情况后，API 会自动重新连接，客户端可不做处理。自动重连地址，可能是原来注册的地址，也可能是系统支持的其它可用的通信地址，它由程序自动选择。

函数原型：

void OnFrontDisconnected (int nReason)；

参数：

nReason：*连接断开原因*

0x1001　网络读失败

0x1002　网络写失败

0x2001　接收心跳超时

0x2002　发送心跳失败

0x2003　收到错误报文

0x2004　服务器主动断开

4.3.3 OnHeartBeatWarning 方法

心跳超时警告。当长时间未收到报文时，该方法被调用。

函数原型：

void OnHeartBeatWarning(int nTimeLapse)；

参数：

nTimeLapse：*距离上次接收报文的时间*

4.3.4 OnRspUserLogin 方法

当客户端发出登录请求之后，Kingstar 服务器返回响应时，该方法会被调用，通知客户端登录是否成功。

函数原型：

void OnRspUserLogin(

*CThostFtdcRspUserLoginField *pRspUserLogin,*

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)；

参数：

pRspUserLogin：*返回用户登录信息的地址。*

用户登录信息结构：

```
struct CThostFtdcRspUserLoginField
{
    ///交易日
    TThostFtdcDateType      TradingDay;
    ///登录成功时间
    TThostFtdcTimeType      LoginTime;
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///用户代码
    TThostFtdcUserIDType        UserID;
    ///交易系统名称
    TThostFtdcSystemNameType    SystemName;
    ///前置编号
    TThostFtdcFrontIDType     FrontID;
    ///会话编号
    TThostFtdcSessionIDType     SessionID;
    ///最大报单引用
    TThostFtdcOrderRefType      MaxOrderRef;
    ///上期所时间
    TThostFtdcTimeType      SHFETime;
    ///大商所时间
    TThostFtdcTimeType      DCETime;
    ///郑商所时间
    TThostFtdcTimeType      CZCETime;
    ///中金所时间
    TThostFtdcTimeType      FFEXTime;
};
```

pRspInfo：*返回用户响应信息的地址。错误误代码为 0 时，表示操作成功，以下同。*

响应信息结构：

struct CThostFtdcRspInfoField

{

///错误代码

TThostFtdcErrorIDType ErrorID;

///错误信息

TThostFtdcErrorMsgType ErrorMsg;

};

nRequestID：*返回用户登录请求的 ID，该 ID 由用户在登录时指定。*

bIsLast：*指示该次返回是否为针对 nRequestID 的最后一次返回。*

4.3.5 OnRspUserLogout 方法

当客户端发出退出请求之后，Kingstar 返回响应时，该方法会被调用，通知客户端退出是否成功。

函数原型：

void OnRspUserLogout(

*CThostFtdcUserLogoutField *pUserLogout,*

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)；

参数：

pRspUserLogout：*返回用户退出信息的地址。*

用户登出信息结构：

struct CThostFtdcUserLogoutField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///用户代码

TThostFtdcUserIDType UserID;

};

4.3.6 OnRspUserPasswordUpdate 方法

用户密码修改应答。当客户端发出用户密码修改指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspUserPasswordUpdate(
        CThostFtdcUserPasswordUpdateField    *pUserPasswordUpdate,
         CThostFtdcRspInfoField    *pRspInfo,
         int nRequestID,
         bool bIsLast)；
```

参数：

pUserPasswordUpdate：*指向用户密码修改结构的地址，包含了用户密码修改请求的输入数据。*

用户密码修改结构：

```
struct CThostFtdcUserPasswordUpdateField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///原来的口令
    TThostFtdcPasswordType      OldPassword;
    ///新的口令
    TThostFtdcPasswordType      NewPassword;
};
```

4.3.7 OnRspTradingAccountPasswordUpdate 方法

资金账户口令更新应答。当客户端发出资金账户口令更新指令后，Kingstar返回响应时，该方法会被调用。

函数原型：

```
void OnRspTradingAccountPasswordUpdate(
    CThostFtdcTradingAccountPasswordUpdateField *pTradingAccountPasswordUpdate,
```

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)；

参数：

pTradingAccountPasswordUpdate：指向资金账户口令变更域结构的地址，包含了用户密码修改请求的输入数据。

资金账户口令变更域结构：

```
struct CThostFtdcTradingAccountPasswordUpdateField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者帐号
    TThostFtdcAccountIDType     AccountID;
    ///原来的口令
    TThostFtdcPasswordType      OldPassword;
    ///新的口令
    TThostFtdcPasswordType      NewPassword;
};
```

4.3.8 OnRspError 方法

针对用户请求的出错通知。

函数原型：

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

参数：

pRspInfo：*返回用户响应信息的地址。*

响应信息结构：

```
struct CThostFtdcRspInfoField
{
    ///错误代码
```

```
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType    ErrorMsg;
};
```

4.3.9 OnRspOrderInsert 方法

报单录入应答。当客户端发出过报单录入指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspOrderInsert(
        CThostFtdcInputOrderField *pInputOrder,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pInputOrder：*指向报单录入结构的地址，包含了提交报单录入时的输入数据，和后台返回的报单编号。*

输入报单结构：

```
struct CThostFtdcInputOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///报单引用
    TThostFtdcOrderRefType    OrderRef;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///报单价格条件
    TThostFtdcOrderPriceTypeType    OrderPriceType;
```

```
    ///买卖方向
    TThostFtdcDirectionType    Direction;
    ///组合开平标志
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///组合投机套保标志
    TThostFtdcCombHedgeFlagType    CombHedgeFlag;
    ///价格
    TThostFtdcPriceType    LimitPrice;
    ///数量
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///有效期类型
    TThostFtdcTimeConditionType    TimeCondition;
    ///GTD 日期 （保留字段）
    TThostFtdcDateType    GTDDate;
    ///成交量类型
    TThostFtdcVolumeConditionType    VolumeCondition;
    ///最小成交量
    TThostFtdcVolumeType    MinVolume;
    ///触发条件
    TThostFtdcContingentConditionType    ContingentCondition;
    ///止损价
    TThostFtdcPriceType    StopPrice;
    ///强平原因
    TThostFtdcForceCloseReasonType    ForceCloseReason;
    ///自动挂起标志
    TThostFtdcBoolType    IsAutoSuspend;
    ///业务单元
    TThostFtdcBusinessUnitType    BusinessUnit;
    ///请求编号
    TThostFtdcRequestIDType    RequestID;
    ///用户强评标志
    TThostFtdcBoolType    UserForceClose;
};
```

4.3.10 OnRspOrderAction 方法

报单操作应答。当客户端发出过报单操作指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspOrderAction(
        CThostFtdcOrderActionField *pOrderAction,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pOrderAction：*指向报单操作结构的地址，包含了提交报单操作的输入数据，和后台返回的报单编号。*

报单操作结构：

```
struct CThostFtdcOrderActionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///报单操作引用
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///请求编号
    TThostFtdcRequestIDType     RequestID;
    ///前置编号
    TThostFtdcFrontIDType FrontID;
    ///会话编号
    TThostFtdcSessionIDType     SessionID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///报单编号
```

```
TThostFtdcOrderSysIDType    OrderSysID;
///操作标志
TThostFtdcActionFlagType    ActionFlag;
///价格
TThostFtdcPriceType    LimitPrice;
///数量变化
TThostFtdcVolumeType    VolumeChange;
///操作日期
TThostFtdcDateType    ActionDate;
///操作时间
TThostFtdcTimeType    ActionTime;
///交易所交易员代码
TThostFtdcTraderIDType    TraderID;
///安装编号
TThostFtdcInstallIDType    InstallID;
///本地报单编号
TThostFtdcOrderLocalIDType    OrderLocalID;
///操作本地编号
TThostFtdcOrderLocalIDType    ActionLocalID;
///会员代码
TThostFtdcParticipantIDType    ParticipantID;
///客户代码
TThostFtdcClientIDType    ClientID;
///业务单元
TThostFtdcBusinessUnitType    BusinessUnit;
///报单操作状态
TThostFtdcOrderActionStatusType    OrderActionStatus;
///用户代码
TThostFtdcUserIDType    UserID;
///状态信息
TThostFtdcErrorMsgType    StatusMsg;
///合约代码
TThostFtdcInstrumentIDType InstrumentID;
```

};

4.3.11 OnRspQueryMaxOrderVolume 方法

查询最大报单数量应答。当客户端发出查询最大报单数量指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQueryMaxOrderVolume(
        CThostFtdcQueryMaxOrderVolumeField *pQueryMaxOrderVolume,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pQueryMaxOrderVolume：*指向查询最大报单数量结构的地址，包含了最大允许报单数量。*

最大报单数量结构：

```
struct CThostFtdcQueryMaxOrderVolumeField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///买卖方向
    TThostFtdcDirectionType     Direction;
    ///开平标志
    TThostFtdcOffsetFlagType    OffsetFlag;
    ///投机套保标志
    TThostFtdcHedgeFlagType     HedgeFlag;
    ///最大允许报单数量
    TThostFtdcVolumeType  MaxVolume;
};
```

4.3.12 OnRspSettlementInfoConfirm 方法

投资者结算结果确认应答。当客户端发出投资者结算结果确认指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspSettlementInfoConfirm(
        CThostFtdcSettlementInfoConfirmField *pSettlementInfoConfirm,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pSettlementInfoConfirm: *指向投资者结算结果确认信息结构的地址，包含了最大允许报单数量。*

投资者结算结果确认信息结构：

```
struct CThostFtdcSettlementInfoConfirmField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///确认日期
    TThostFtdcDateType      ConfirmDate;
    ///确认时间
    TThostFtdcTimeType      ConfirmTime;
};
```

4.3.13 OnRspFromBankToFutureByFuture 方法

请求银行资金转期货响应。当客户端发出请求银行资金转期货指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspFromBankToFutureByFuture (
    CThostFtdcReqTransferField *pReqTransfer,
```

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)；

参数：

pReqTransfer：*指向银行资金转期货请求响应结构的地址。*

银行资金转期货请求响应结构：

```
struct CThostFtdcReqTransferField
{
    ///业务功能码
    TThostFtdcTradeCodeType    TradeCode;
    ///银行代码
    TThostFtdcBankIDType    BankID;
    ///银行分支机构代码
    TThostFtdcBankBrchIDType  BankBranchID;
    ///期商代码
    TThostFtdcBrokerIDType      BrokerID;
    ///期商分支机构代码
    TThostFtdcFutureBranchIDType    BrokerBranchID;
    ///交易日期
    TThostFtdcTradeDateType     TradeDate;
    ///交易时间
    TThostFtdcTradeTimeType     TradeTime;
    ///银行流水号
    TThostFtdcBankSerialType    BankSerial;
    ///交易系统日期
    TThostFtdcTradeDateType     TradingDay;
    ///银期平台消息流水号
    TThostFtdcSerialType    PlateSerial;
    ///最后分片标志
    TThostFtdcLastFragmentType      LastFragment;
    ///会话号
    TThostFtdcSessionIDType     SessionID;
    ///客户姓名
```

```
TThostFtdcIndividualNameType    CustomerName;
///证件类型
TThostFtdcIdCardTypeType    IdCardType;
///证件号码
TThostFtdcIdentifiedCardNoType  IdentifiedCardNo;
///客户类型
TThostFtdcCustTypeType      CustType;
///银行帐号
TThostFtdcBankAccountType    BankAccount;
///银行密码
TThostFtdcPasswordType      BankPassWord;
///投资者帐号
TThostFtdcAccountIDType    AccountID;
///期货密码
TThostFtdcPasswordType      Password;
///安装编号
TThostFtdcInstallIDType     InstallID;
///期货公司流水号
TThostFtdcFutureSerialType  FutureSerial;
///用户标识
TThostFtdcUserIDType  UserID;
///验证客户证件号码标志
TThostFtdcYesNoIndicatorType    VerifyCertNoFlag;
///币种代码
TThostFtdcCurrencyIDType   CurrencyID;
///转帐金额
TThostFtdcTradeAmountType  TradeAmount;
///期货可取金额
TThostFtdcTradeAmountType    FutureFetchAmount;
///费用支付标志
TThostFtdcFeePayFlagType  FeePayFlag;
///应收客户费用
TThostFtdcCustFeeType    CustFee;
```

```
    ///应收期货公司费用
    TThostFtdcFutureFeeType     BrokerFee;
    ///发送方给接收方的消息
    TThostFtdcAddInfoType     Message;
    ///摘要
    TThostFtdcDigestType    Digest;
    ///银行帐号类型
    TThostFtdcBankAccTypeType     BankAccType;
    ///渠道标志
    TThostFtdcDeviceIDType      DeviceID;
    ///期货单位帐号类型
    TThostFtdcBankAccTypeType     BankSecuAccType;
    ///期货公司银行编码
    TThostFtdcBankCodingForFutureType       BrokerIDByBank;
    ///期货单位帐号
    TThostFtdcBankAccountType     BankSecuAcc;
    ///银行密码标志
    TThostFtdcPwdFlagType       BankPwdFlag;
    ///期货资金密码核对标志
    TThostFtdcPwdFlagType       SecuPwdFlag;
    ///交易柜员
    TThostFtdcOperNoType OperNo;
    ///请求编号
    TThostFtdcRequestIDType     RequestID;
    ///交易 ID
    TThostFtdcTIDType       TID;
    ///转账交易状态
    TThostFtdcTransferStatusType      TransferStatus;
};
```

4.3.14 OnRspFromFutureToBankByFuture 方法

请求期货资金转银行响应。当客户端发出请求期货资金转银行指令后，Kingstar 服务器返

回响应时，该方法会被调用。

函数原型：

void OnRspFromFutureToBankByFuture (

*CThostFtdcReqTransferField *pReqTransfer,*

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)；

参数：

pReqTransfer：*指向期货资金转银行请求响应结构的地址。*

期货资金转银行请求响应结构：

struct CThostFtdcReqTransferField

{

///业务功能码

TThostFtdcTradeCodeType TradeCode;

///银行代码

TThostFtdcBankIDType BankID;

///银行分支机构代码

TThostFtdcBankBrchIDType BankBranchID;

///期商代码

TThostFtdcBrokerIDType BrokerID;

///期商分支机构代码

TThostFtdcFutureBranchIDType BrokerBranchID;

///交易日期

TThostFtdcTradeDateType TradeDate;

///交易时间

TThostFtdcTradeTimeType TradeTime;

///银行流水号

TThostFtdcBankSerialType BankSerial;

///交易系统日期

TThostFtdcTradeDateType TradingDay;

///银期平台消息流水号

TThostFtdcSerialType PlateSerial;

///最后分片标志

```
TThostFtdcLastFragmentType	LastFragment;
///会话号
TThostFtdcSessionIDType	SessionID;
///客户姓名
TThostFtdcIndividualNameType	CustomerName;
///证件类型
TThostFtdcIdCardTypeType	IdCardType;
///证件号码
TThostFtdcIdentifiedCardNoType	IdentifiedCardNo;
///客户类型
TThostFtdcCustTypeType	CustType;
///银行帐号
TThostFtdcBankAccountType	BankAccount;
///银行密码
TThostFtdcPasswordType	BankPassWord;
///投资者帐号
TThostFtdcAccountIDType	AccountID;
///期货密码
TThostFtdcPasswordType	Password;
///安装编号
TThostFtdcInstallIDType	InstallID;
///期货公司流水号
TThostFtdcFutureSerialType	FutureSerial;
///用户标识
TThostFtdcUserIDType	UserID;
///验证客户证件号码标志
TThostFtdcYesNoIndicatorType	VerifyCertNoFlag;
///币种代码
TThostFtdcCurrencyIDType	CurrencyID;
///转帐金额
TThostFtdcTradeAmountType	TradeAmount;
///期货可取金额
TThostFtdcTradeAmountType	FutureFetchAmount;
```

```
    ///费用支付标志
    TThostFtdcFeePayFlagType    FeePayFlag;
    ///应收客户费用
    TThostFtdcCustFeeType    CustFee;
    ///应收期货公司费用
    TThostFtdcFutureFeeType    BrokerFee;
    ///发送方给接收方的消息
    TThostFtdcAddInfoType    Message;
    ///摘要
    TThostFtdcDigestType    Digest;
    ///银行帐号类型
    TThostFtdcBankAccTypeType    BankAccType;
    ///渠道标志
    TThostFtdcDeviceIDType    DeviceID;
    ///期货单位帐号类型
    TThostFtdcBankAccTypeType    BankSecuAccType;
    ///期货公司银行编码
    TThostFtdcBankCodingForFutureType    BrokerIDByBank;
    ///期货单位帐号
    TThostFtdcBankAccountType    BankSecuAcc;
    ///银行密码标志
    TThostFtdcPwdFlagType    BankPwdFlag;
    ///期货资金密码核对标志
    TThostFtdcPwdFlagType    SecuPwdFlag;
    ///交易柜员
    TThostFtdcOperNoType    OperNo;
    ///请求编号
    TThostFtdcRequestIDType    RequestID;
    ///交易 ID
    TThostFtdcTIDType    TID;
    ///转账交易状态
    TThostFtdcTransferStatusType    TransferStatus;
};
```

4.3.15 OnRspQueryBankAccountMoneyByFuture 方法

请求查询银行资金响应。当客户端发出请求查询银行资金指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQueryBankAccountMoneyByFuture (
    CThostFtdcReqQueryAccountField *pReqQueryAccount,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)；
```

参数：

pReqQueryAccount：*指向查询银行资金请求响应结构的地址。*

查询银行资金请求响应结构：

```
struct CThostFtdcReqQueryAccountField
{
    ///业务功能码
    TThostFtdcTradeCodeType    TradeCode;
    ///银行代码
    TThostFtdcBankIDType    BankID;
    ///银行分支机构代码
    TThostFtdcBankBrchIDType  BankBranchID;
    ///期商代码
    TThostFtdcBrokerIDType      BrokerID;
    ///期商分支机构代码
    TThostFtdcFutureBranchIDType    BrokerBranchID;
    ///交易日期
    TThostFtdcTradeDateType    TradeDate;
    ///交易时间
    TThostFtdcTradeTimeType    TradeTime;
    ///银行流水号
    TThostFtdcBankSerialType    BankSerial;
    ///交易系统日期
    TThostFtdcTradeDateType    TradingDay;
```

```
///银期平台消息流水号
TThostFtdcSerialType    PlateSerial;
///最后分片标志
TThostFtdcLastFragmentType      LastFragment;
///会话号
TThostFtdcSessionIDType     SessionID;
///客户姓名
TThostFtdcIndividualNameType    CustomerName;
///证件类型
TThostFtdcIdCardTypeType    IdCardType;
///证件号码
TThostFtdcIdentifiedCardNoType    IdentifiedCardNo;
///客户类型
TThostFtdcCustTypeType      CustType;
///银行帐号
TThostFtdcBankAccountType   BankAccount;
///银行密码
TThostFtdcPasswordType      BankPassWord;
///投资者帐号
TThostFtdcAccountIDType     AccountID;
///期货密码
TThostFtdcPasswordType      Password;
///期货公司流水号
TThostFtdcFutureSerialType      FutureSerial;
///安装编号
TThostFtdcInstallIDType     InstallID;
///用户标识
TThostFtdcUserIDType    UserID;
///验证客户证件号码标志
TThostFtdcYesNoIndicatorType    VerifyCertNoFlag;
///币种代码
TThostFtdcCurrencyIDType    CurrencyID;
///摘要
```

```
    TThostFtdcDigestType    Digest;
    ///银行帐号类型
    TThostFtdcBankAccTypeType    BankAccType;
    ///渠道标志
    TThostFtdcDeviceIDType    DeviceID;
    ///期货单位帐号类型
    TThostFtdcBankAccTypeType    BankSecuAccType;
    ///期货公司银行编码
    TThostFtdcBankCodingForFutureType    BrokerIDByBank;
    ///期货单位帐号
    TThostFtdcBankAccountType    BankSecuAcc;
    ///银行密码标志
    TThostFtdcPwdFlagType    BankPwdFlag;
    ///期货资金密码核对标志
    TThostFtdcPwdFlagType    SecuPwdFlag;
    ///交易柜员
    TThostFtdcOperNoType OperNo;
    ///请求编号
    TThostFtdcRequestIDType    RequestID;
    ///交易 ID
    TThostFtdcTIDType    TID;
};
```

4.3.16 OnRspQryTransferSerial 方法

请求查询转账流水响应。当客户端发出请求查询转账流水指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryTransferSerial(
        CThostFtdcTransferSerialField *pTransferSerial,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
         bool bIsLast)；
```

参数：

pTransferSerial：*指向查询转账流水请求响应结构的地址。*

查询转账流水请求响应结构：

```
struct CThostFtdcTransferSerialField
{
    ///平台流水号
    TThostFtdcPlateSerialType    PlateSerial;
    ///交易发起方日期
    TThostFtdcTradeDateType    TradeDate;
    ///交易日期
    TThostFtdcDateType    TradingDay;
    ///交易时间
    TThostFtdcTradeTimeType    TradeTime;
    ///交易代码
    TThostFtdcTradeCodeType    TradeCode;
    ///会话编号
    TThostFtdcSessionIDType    SessionID;
    ///银行编码
    TThostFtdcBankIDType    BankID;
    ///银行分支机构编码
    TThostFtdcBankBrchIDType  BankBranchID;
    ///银行帐号类型
    TThostFtdcBankAccTypeType  BankAccType;
    ///银行帐号
    TThostFtdcBankAccountType BankAccount;
    ///银行流水号
    TThostFtdcBankSerialType    BankSerial;
    ///期货公司编码
    TThostFtdcBrokerIDType    BrokerID;
    ///期商分支机构代码
    TThostFtdcFutureBranchIDType    BrokerBranchID;
    ///期货公司帐号类型
    TThostFtdcFutureAccTypeType    FutureAccType;
```

```
    ///投资者帐号
    TThostFtdcAccountIDType    AccountID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///期货公司流水号
    TThostFtdcFutureSerialType  FutureSerial;
    ///证件类型
    TThostFtdcIdCardTypeType   IdCardType;
    ///证件号码
    TThostFtdcIdentifiedCardNoType  IdentifiedCardNo;
    ///币种代码
    TThostFtdcCurrencyIDType   CurrencyID;
    ///交易金额
    TThostFtdcTradeAmountType TradeAmount;
    ///应收客户费用
    TThostFtdcCustFeeType CustFee;
    ///应收期货公司费用
    TThostFtdcFutureFeeType    BrokerFee;
    ///有效标志
    TThostFtdcAvailabilityFlagType    AvailabilityFlag;
    ///操作员
    TThostFtdcOperatorCodeType      OperatorCode;
    ///新银行帐号
    TThostFtdcBankAccountType BankNewAccount;
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType     ErrorMsg;
};
```

4.3.17 OnRspQryOrder 方法

报单查询请求。当客户端发出报单查询指令后，Kingstar 服务器返回响应时，该方法会被

调用。

函数原型：

```
void OnRspQryOrder(
        CThostFtdcOrderField *pOrder,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pOrder：*指向报单信息结构的地址。*

报单信息结构：

```
struct CThostFtdcOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///报单价格条件
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///买卖方向
    TThostFtdcDirectionType     Direction;
    ///组合开平标志
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///组合投机套保标志
    TThostFtdcCombHedgeFlagType   CombHedgeFlag;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量
```

```
TThostFtdcVolumeType	VolumeTotalOriginal;
///有效期类型
TThostFtdcTimeConditionType	TimeCondition;
///GTD 日期
TThostFtdcDateType	GTDDate;
///成交量类型
TThostFtdcVolumeConditionType	VolumeCondition;
///最小成交量
TThostFtdcVolumeType	MinVolume;
///触发条件
TThostFtdcContingentConditionType	ContingentCondition;
///止损价
TThostFtdcPriceType	StopPrice;
///强平原因
TThostFtdcForceCloseReasonType	ForceCloseReason;
///自动挂起标志
TThostFtdcBoolType	IsAutoSuspend;
///业务单元
TThostFtdcBusinessUnitType	BusinessUnit;
///请求编号
TThostFtdcRequestIDType	RequestID;
///本地报单编号
TThostFtdcOrderLocalIDType	OrderLocalID;
///交易所代码
TThostFtdcExchangeIDType	ExchangeID;
///会员代码
TThostFtdcParticipantIDType	ParticipantID;
///客户代码
TThostFtdcClientIDType	ClientID;
///合约在交易所的代码
TThostFtdcExchangeInstIDType	ExchangeInstID;
///交易所交易员代码
TThostFtdcTraderIDType	TraderID;
```

```
///安装编号
TThostFtdcInstallIDType	InstallID;
///报单提交状态
TThostFtdcOrderSubmitStatusType	OrderSubmitStatus;
///报单提示序号
TThostFtdcSequenceNoType	NotifySequence;
///交易日
TThostFtdcDateType	TradingDay;
///结算编号
TThostFtdcSettlementIDType	SettlementID;
///报单编号
TThostFtdcOrderSysIDType	OrderSysID;
///报单来源
TThostFtdcOrderSourceType	OrderSource;
///报单状态
TThostFtdcOrderStatusType	OrderStatus;
///报单类型
TThostFtdcOrderTypeType	OrderType;
///今成交数量
TThostFtdcVolumeType	VolumeTraded;
///剩余数量
TThostFtdcVolumeType	VolumeTotal;
///报单日期
TThostFtdcDateType	InsertDate;
///委托时间
TThostFtdcTimeType	InsertTime;
///激活时间
TThostFtdcTimeType	ActiveTime;
///挂起时间
TThostFtdcTimeType	SuspendTime;
///最后修改时间
TThostFtdcTimeType	UpdateTime;
///撤销时间
```

```
    TThostFtdcTimeType      CancelTime;
    ///最后修改交易所交易员代码
    TThostFtdcTraderIDType      ActiveTraderID;
    ///结算会员编号
    TThostFtdcParticipantIDType      ClearingPartID;
    ///序号
    TThostFtdcSequenceNoType  SequenceNo;
    ///前置编号
    TThostFtdcFrontIDType FrontID;
    ///会话编号
    TThostFtdcSessionIDType      SessionID;
    ///用户端产品信息
    TThostFtdcProductInfoType   UserProductInfo;
    ///状态信息
    TThostFtdcErrorMsgType      StatusMsg;
    ///用户强评标志
    TThostFtdcBoolType      UserForceClose;
    ///操作用户代码
    TThostFtdcUserIDType    ActiveUserID;
    ///经纪公司报单编号
    TThostFtdcSequenceNoType  BrokerOrderSeq;
    ///相关报单
    TThostFtdcOrderSysIDType   RelativeOrderSysID;
};
```

4.3.18 OnRspQryTrade 方法

成交单查询应答。当客户端发出成交单查询指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryTrade(
        CThostFtdcTradeField *pTrade,
        CThostFtdcRspInfoField *pRspInfo,
```

int nRequestID,

bool bIsLast)；

参数：

pTrade：*指向成交信息结构的地址。*

成交信息结构：

```
struct CThostFtdcTradeField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType  InstrumentID;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///成交编号
    TThostFtdcTradeIDType  TradeID;
    ///买卖方向
    TThostFtdcDirectionType      Direction;
    ///报单编号
    TThostFtdcOrderSysIDType  OrderSysID;
    ///会员代码
    TThostFtdcParticipantIDType      ParticipantID;
    ///客户代码
    TThostFtdcClientIDType    ClientID;
    ///交易角色
    TThostFtdcTradingRoleType    TradingRole;
    ///合约在交易所的代码
    TThostFtdcExchangeInstIDType    ExchangeInstID;
```

```
///开平标志
TThostFtdcOffsetFlagType    OffsetFlag;
///投机套保标志
TThostFtdcHedgeFlagType    HedgeFlag;
///价格
TThostFtdcPriceType    Price;
///数量
TThostFtdcVolumeType  Volume;
///成交时期
TThostFtdcDateType    TradeDate;
///成交时间
TThostFtdcTimeType    TradeTime;
///成交类型
TThostFtdcTradeTypeType    TradeType;
///成交价来源
TThostFtdcPriceSourceType    PriceSource;
///交易所交易员代码
TThostFtdcTraderIDType    TraderID;
///本地报单编号
TThostFtdcOrderLocalIDType    OrderLocalID;
///结算会员编号
TThostFtdcParticipantIDType    ClearingPartID;
///业务单元
TThostFtdcBusinessUnitType    BusinessUnit;
///序号
TThostFtdcSequenceNoType  SequenceNo;
///交易日
TThostFtdcDateType    TradingDay;
///结算编号
TThostFtdcSettlementIDType  SettlementID;
///经纪公司报单编号
TThostFtdcSequenceNoType  BrokerOrderSeq;
///成交来源
```

TThostFtdcTradeSourceType TradeSource;

};

4.3.19 OnRspQryInvestor 方法

会员客户查询应答。当客户端发出会员客户查询指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

void OnRspQryInvestor (

*CThostFtdcInvestorField *pInvestor,*

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)；

参数：

pInvestor：*指向投资者信息结构的地址。*

投资者信息结构：

struct CThostFtdcInvestorField

{

///投资者代码

TThostFtdcInvestorIDType InvestorID;

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者分组代码

TThostFtdcInvestorIDType InvestorGroupID;

///投资者名称

TThostFtdcPartyNameType InvestorName;

///证件类型

TThostFtdcIdCardTypeType IdentifiedCardType;

///证件号码

TThostFtdcIdentifiedCardNoType IdentifiedCardNo;

///是否活跃

TThostFtdcBoolType IsActive;

///联系电话

```
    TThostFtdcTelephoneType     Telephone;
    ///通讯地址
    TThostFtdcAddressType Address;
    ///开户日期
    TThostFtdcDateType      OpenDate;
    ///手机
    TThostFtdcMobileType   Mobile;
    ///手续费率模板代码
    TThostFtdcInvestorIDType     CommModelID;
};
```

4.3.20 OnRspQryInvestorPosition 方法

投资者持仓查询应答。当客户端发出投资者持仓查询指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryInvestorPosition(
        CThostFtdcInvestorPositionField *pInvestorPosition,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pInvestorPosition ：*指向投资者持仓应答结构的地址。*

投资者持仓应答结构：

```
struct CThostFtdcInvestorPositionField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///持仓多空方向
```

```
TThostFtdcPosiDirectionType    PosiDirection;
///投机套保标志
TThostFtdcHedgeFlagType    HedgeFlag;
///持仓日期
TThostFtdcPositionDateType PositionDate;
///上日持仓
TThostFtdcVolumeType  YdPosition;
///今日持仓
TThostFtdcVolumeType  Position;
///多头冻结
TThostFtdcVolumeType  LongFrozen;
///空头冻结
TThostFtdcVolumeType  ShortFrozen;
///多头开仓冻结金额
TThostFtdcMoneyType   LongFrozenAmount;
///空头开仓冻结金额
TThostFtdcMoneyType   ShortFrozenAmount;
///开仓量
TThostFtdcVolumeType  OpenVolume;
///平仓量
TThostFtdcVolumeType  CloseVolume;
///开仓金额
TThostFtdcMoneyType   OpenAmount;
///平仓金额
TThostFtdcMoneyType   CloseAmount;
///持仓成本
TThostFtdcMoneyType   PositionCost;
///上次占用的保证金
TThostFtdcMoneyType   PreMargin;
///占用的保证金
TThostFtdcMoneyType   UseMargin;
///冻结的保证金
TThostFtdcMoneyType   FrozenMargin;
```

```
///冻结的资金
TThostFtdcMoneyType	FrozenCash;
///冻结的手续费
TThostFtdcMoneyType	FrozenCommission;
///资金差额
TThostFtdcMoneyType	CashIn;
///手续费
TThostFtdcMoneyType	Commission;
///平仓盈亏
TThostFtdcMoneyType	CloseProfit;
///持仓盈亏
TThostFtdcMoneyType	PositionProfit;
///上次结算价
TThostFtdcPriceType	PreSettlementPrice;
///本次结算价
TThostFtdcPriceType	SettlementPrice;
///交易日
TThostFtdcDateType	TradingDay;
///结算编号
TThostFtdcSettlementIDType	SettlementID;
///开仓成本
TThostFtdcMoneyType	OpenCost;
///交易所保证金
TThostFtdcMoneyType	ExchangeMargin;
///组合成交形成的持仓
TThostFtdcVolumeType	CombPosition;
///组合多头冻结
TThostFtdcVolumeType	CombLongFrozen;
///组合空头冻结
TThostFtdcVolumeType	CombShortFrozen;
///逐日盯市平仓盈亏
TThostFtdcMoneyType	CloseProfitByDate;
///逐笔对冲平仓盈亏
```

```
    TThostFtdcMoneyType    CloseProfitByTrade;
    ///今日持仓
    TThostFtdcVolumeType   TodayPosition;
    ///保证金率
    TThostFtdcRatioType    MarginRateByMoney;
    ///保证金率(按手数)
    TThostFtdcRatioType    MarginRateByVolume;
};
```

4.3.21 OnRspQryTradingAccount 方法

请求查询资金账户响应。当客户端发出请求查询资金账户指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryTradingAccount(
        CThostFtdcTradingAccountField *pTradingAccount,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pTradingAccount ：*指向资金账户结构的地址。*

资金账户结构：

```
struct CThostFtdcTradingAccountField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者帐号
    TThostFtdcAccountIDType     AccountID;
    ///上次质押金额
    TThostFtdcMoneyType    PreMortgage;
    ///上次信用额度
    TThostFtdcMoneyType    PreCredit;
    ///上次存款额
```

```
TThostFtdcMoneyType	PreDeposit;
///上次结算准备金
TThostFtdcMoneyType	PreBalance;
///上次占用的保证金
TThostFtdcMoneyType	PreMargin;
///利息基数
TThostFtdcMoneyType	InterestBase;
///利息收入
TThostFtdcMoneyType	Interest;
///入金金额
TThostFtdcMoneyType	Deposit;
///出金金额
TThostFtdcMoneyType	Withdraw;
///冻结的保证金
TThostFtdcMoneyType	FrozenMargin;
///冻结的资金
TThostFtdcMoneyType	FrozenCash;
///冻结的手续费
TThostFtdcMoneyType	FrozenCommission;
///当前保证金总额
TThostFtdcMoneyType	CurrMargin;
///资金差额
TThostFtdcMoneyType	CashIn;
///手续费
TThostFtdcMoneyType	Commission;
///平仓盈亏
TThostFtdcMoneyType	CloseProfit;
///持仓盈亏
TThostFtdcMoneyType	PositionProfit;
///期货结算准备金
TThostFtdcMoneyType	Balance;
///可用资金
TThostFtdcMoneyType	Available;
```

```
    ///可取资金
    TThostFtdcMoneyType    WithdrawQuota;
    ///基本准备金
    TThostFtdcMoneyType    Reserve;
    ///交易日
    TThostFtdcDateType     TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType    SettlementID;
    ///信用额度
    TThostFtdcMoneyType    Credit;
    ///质押金额
    TThostFtdcMoneyType    Mortgage;
    ///交易所保证金
    TThostFtdcMoneyType    ExchangeMargin;
    ///投资者交割保证金
    TThostFtdcMoneyType    DeliveryMargin;
    ///交易所交割保证金
    TThostFtdcMoneyType    ExchangeDeliveryMargin;
};
```

4.3.22 OnRspQryTradingCode 方法

请求查询交易编码响应。当客户端发出请求查询交易编码指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryTradingCode(
        CThostFtdcTradingCodeField *pTradingCode,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
         bool bIsLast)；
```

参数：

pTradingCode：*指向交易编码结构的地址。*

交易编码结构：

```
struct CThostFtdcTradingCodeField
{
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///客户代码
    TThostFtdcClientIDTypeClientID;
    ///是否活跃
    TThostFtdcBoolType      IsActive;
    ///交易编码类型
    TThostFtdcClientIDTypeType ClientIDType;
};
```

4.3.23 OnRspQryExchange 方法

请求查询交易所响应。当客户端发出请求查询交易所指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryExchange(
        CThostFtdcExchangeField *pExchange,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pExchange： *指向交易所结构的地址。*

交易所结构：

```
struct CThostFtdcExchangeField
{
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
```

```
    ///交易所名称
    TThostFtdcExchangeNameType    ExchangeName;
    ///交易所属性
    TThostFtdcExchangePropertyTypeExchangeProperty;
};
```

4.3.24 OnRspQryInstrument 方法

请求查询合约响应。当客户端发出请求查询合约指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryInstrument(
        CThostFtdcInstrumentField *pInstrument,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast) ；
```

参数：

pInstrument： *指向合约结构的地址。*

合约结构：

```
struct CThostFtdcInstrumentField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///合约名称
    TThostFtdcInstrumentNameType    InstrumentName;
    ///合约在交易所的代码
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///产品代码
    TThostFtdcInstrumentIDType   ProductID;
    ///产品类型
    TThostFtdcProductClassType   ProductClass;
```

```
///交割年份
TThostFtdcYearType	DeliveryYear;
///交割月
TThostFtdcMonthType	DeliveryMonth;
///市价单最大下单量
TThostFtdcVolumeType	MaxMarketOrderVolume;
///市价单最小下单量
TThostFtdcVolumeType	MinMarketOrderVolume;
///限价单最大下单量
TThostFtdcVolumeType	MaxLimitOrderVolume;
///限价单最小下单量
TThostFtdcVolumeType	MinLimitOrderVolume;
///合约数量乘数
TThostFtdcVolumeMultipleType	VolumeMultiple;
///最小变动价位
TThostFtdcPriceType	PriceTick;
///创建日
TThostFtdcDateType	CreateDate;
///上市日
TThostFtdcDateType	OpenDate;
///到期日
TThostFtdcDateType	ExpireDate;
///开始交割日
TThostFtdcDateType	StartDelivDate;
///结束交割日
TThostFtdcDateType	EndDelivDate;
///合约生命周期状态
TThostFtdcInstLifePhaseType	InstLifePhase;
///当前是否交易
TThostFtdcBoolType	IsTrading;
///持仓类型
TThostFtdcPositionTypeType	PositionType;
///持仓日期类型
```

```
    TThostFtdcPositionDateTypeType    PositionDateType;
    ///多头保证金率
    TThostFtdcRatioType    LongMarginRatio;
    ///空头保证金率
    TThostFtdcRatioType    ShortMarginRatio;
};
```

4.3.25 OnRspQryDepthMarketData 方法

请求查询行情响应。当客户端发出请求查询行情指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryDepthMarketData(
        CThostFtdcDepthMarketDataField *pDepthMarketData,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pDepthMarketData ：*指向深度行情结构的地址。*

深度行情结构：

```
struct CThostFtdcDepthMarketDataField
{
    ///交易日
    TThostFtdcDateType    TradingDay;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///合约在交易所的代码
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///最新价
    TThostFtdcPriceType    LastPrice;
    ///上次结算价
```

```
TThostFtdcPriceType	PreSettlementPrice;
///昨收盘
TThostFtdcPriceType	PreClosePrice;
///昨持仓量
TThostFtdcLargeVolumeType PreOpenInterest;
///今开盘
TThostFtdcPriceType	OpenPrice;
///最高价
TThostFtdcPriceType	HighestPrice;
///最低价
TThostFtdcPriceType	LowestPrice;
///数量
TThostFtdcVolumeType	Volume;
///成交金额
TThostFtdcMoneyType	Turnover;
///持仓量
TThostFtdcLargeVolumeType OpenInterest;
///今收盘
TThostFtdcPriceType	ClosePrice;
///本次结算价
TThostFtdcPriceType	SettlementPrice;
///涨停板价
TThostFtdcPriceType	UpperLimitPrice;
///跌停板价
TThostFtdcPriceType	LowerLimitPrice;
///昨虚实度
TThostFtdcRatioType	PreDelta;
///今虚实度
TThostFtdcRatioType	CurrDelta;
///最后修改时间
TThostFtdcTimeType	UpdateTime;
///最后修改毫秒
TThostFtdcMillisecType UpdateMillisec;
```

```
///申买价一
TThostFtdcPriceType	BidPrice1;
///申买量一
TThostFtdcVolumeType	BidVolume1;
///申卖价一
TThostFtdcPriceType	AskPrice1;
///申卖量一
TThostFtdcVolumeType	AskVolume1;
///申买价二
TThostFtdcPriceType	BidPrice2;
///申买量二
TThostFtdcVolumeType	BidVolume2;
///申卖价二
TThostFtdcPriceType	AskPrice2;
///申卖量二
TThostFtdcVolumeType	AskVolume2;
///申买价三
TThostFtdcPriceType	BidPrice3;
///申买量三
TThostFtdcVolumeType	BidVolume3;
///申卖价三
TThostFtdcPriceType	AskPrice3;
///申卖量三
TThostFtdcVolumeType	AskVolume3;
///申买价四
TThostFtdcPriceType	BidPrice4;
///申买量四
TThostFtdcVolumeType	BidVolume4;
///申卖价四
TThostFtdcPriceType	AskPrice4;
///申卖量四
TThostFtdcVolumeType	AskVolume4;
///申买价五
```

```
    TThostFtdcPriceType     BidPrice5;
    ///申买量五
    TThostFtdcVolumeType    BidVolume5;
    ///申卖价五
    TThostFtdcPriceType     AskPrice5;
    ///申卖量五
    TThostFtdcVolumeType    AskVolume5;
    ///当日均价
    TThostFtdcPriceType     AveragePrice;
};
```

4.3.26 OnRspQryInstrumentMarginRate 方法

请求查询合约保证金率响应。当客户端发出请求查询合约保证金率指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryInstrumentMarginRate(
        CThostFtdcInstrumentMarginRateField    *pInstrumentMarginRate,
        CThostFtdcRspInfoField    *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pInstrumentMarginRate：*指向合约保证金率结构的地址。*

合约保证金率结构：

```
struct CThostFtdcInstrumentMarginRateField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///投资者范围
    TThostFtdcInvestorRangeType      InvestorRange;
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
```

```
    TThostFtdcInvestorIDType    InvestorID;
    ///投机套保标志
    TThostFtdcHedgeFlagType    HedgeFlag;
    ///多头保证金率
    TThostFtdcRatioType    LongMarginRatioByMoney;
    ///多头保证金费
    TThostFtdcMoneyType    LongMarginRatioByVolume;
    ///空头保证金率
    TThostFtdcRatioType    ShortMarginRatioByMoney;
    ///空头保证金费
    TThostFtdcMoneyType    ShortMarginRatioByVolume;
    ///是否相对交易所收取
    TThostFtdcBoolType    IsRelative;
};
```

4.3.27 OnRspQryInstrumentCommissionRate 方法

请求查询合约手续费率响应。当客户端发出请求查询合约手续费率指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryInstrumentCommissionRate(
        CThostFtdcInstrumentCommissionRateField *pInstrumentCommissionRate,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pInstrumentCommissionRate： *指向合约手续费率结构的地址。*

合约手续费率结构：

```
struct CThostFtdcInstrumentCommissionRateField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///投资者范围
```

```
    TThostFtdcInvestorRangeType     InvestorRange;
    ///经纪公司代码
    TThostFtdcBrokerIDType     BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType     InvestorID;
    ///开仓手续费率
    TThostFtdcRatioType     OpenRatioByMoney;
    ///开仓手续费
    TThostFtdcRatioType     OpenRatioByVolume;
    ///平仓手续费率
    TThostFtdcRatioType     CloseRatioByMoney;
    ///平仓手续费
    TThostFtdcRatioType     CloseRatioByVolume;
    ///平今手续费率
    TThostFtdcRatioType     CloseTodayRatioByMoney;
    ///平今手续费
    TThostFtdcRatioType     CloseTodayRatioByVolume;
};
```

4.3.28 OnRspQryCFMMCTradingAccountKey 方法

请求查询保证金监控中心密钥响应。当客户端发出请求查询保证金监控中心密钥指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryCFMMCTradingAccountKey(
        CThostFtdcCFMMCTradingAccountKeyField     *pCFMMCTradingAccountKey,
        CThostFtdcRspInfoField *pRspInfo,
         int nRequestID,
        bool bIsLast)；
```

参数：

pCFMMCTradingAccountKey：*指向保证金监控中心密钥结构的地址。*

保证金监控中心密钥结构：

```
struct CThostFtdcCFMMCTradingAccountKeyField
```

```
{
///经纪公司代码
TThostFtdcBrokerIDType      BrokerID;
///经纪公司统一编码
TThostFtdcParticipantIDType      ParticipantID;
///投资者帐号
TThostFtdcAccountIDType    AccountID;
///密钥编号
TThostFtdcSequenceNoType  KeyID;
///动态密钥
TThostFtdcCFMMCKeyType CurrentKey;
};
```

4.3.29 OnRspQrySettlementInfo 方法

请求查询投资者结算结果响应。当客户端发出请求查询投资者结算结果指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQrySettlementInfo(
        CThostFtdcSettlementInfoField *pSettlementInfo,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pSettlementInfo ： 指向投资者结算结果结构的地址。

投资者结算结果结构：

```
struct CThostFtdcSettlementInfoField
{
    ///交易日
    TThostFtdcDateType      TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType   SettlementID;
    ///经纪公司代码
```

```
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///序号
    TThostFtdcSequenceNoType    SequenceNo;
    ///消息正文
    TThostFtdcContentType Content;
};
```

4.3.30 OnRspQryTransferBank 方法

请求查询转帐银行响应。当客户端发出请求查询转帐银行指令后，Kingstar 返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryTransferBank(
    CThostFtdcTransferBankField *pTransferBank,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)；
```

参数：

pTransferBank： *指向转帐银行结构的地址。*

转帐银行结构：

```
struct CThostFtdcTransferBankField
{
    ///银行代码
    TThostFtdcBankIDType  BankID;
    ///银行分中心代码
    TThostFtdcBankBrchIDType  BankBrchID;
    ///银行名称
    TThostFtdcBankNameType    BankName;
    ///是否活跃
    TThostFtdcBoolType      IsActive;
};
```

4.3.31 OnRspQryInvestorPositionDetail 方法

请求查询投资者持仓明细响应。当客户端发出请求请求查询投资者持仓明细指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryInvestorPositionDetail(
        CThostFtdcInvestorPositionDetailField *pInvestorPositionDetail,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast) ；
```

参数：

pInvestorPositionDetail ***：*** *指向投资者持仓明细结构的地址。*

投资者持仓明细结构：

```
struct CThostFtdcInvestorPositionDetailField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///投机套保标志
    TThostFtdcHedgeFlagType    HedgeFlag;
    ///买卖
    TThostFtdcDirectionType    Direction;
    ///开仓日期
    TThostFtdcDateType    OpenDate;
    ///成交编号
    TThostFtdcTradeIDType TradeID;
    ///数量
    TThostFtdcVolumeType  Volume;
    ///开仓价
    TThostFtdcPriceType    OpenPrice;
```

```
///交易日
TThostFtdcDateType	TradingDay;
///结算编号
TThostFtdcSettlementIDType	SettlementID;
///成交类型
TThostFtdcTradeTypeType	TradeType;
///组合合约代码
TThostFtdcInstrumentIDType	CombInstrumentID;
///交易所代码
TThostFtdcExchangeIDType	ExchangeID;
///逐日盯市平仓盈亏
TThostFtdcMoneyType	CloseProfitByDate;
///逐笔对冲平仓盈亏
TThostFtdcMoneyType	CloseProfitByTrade;
///逐日盯市持仓盈亏
TThostFtdcMoneyType	PositionProfitByDate;
///逐笔对冲持仓盈亏
TThostFtdcMoneyType	PositionProfitByTrade;
///投资者保证金
TThostFtdcMoneyType	Margin;
///交易所保证金
TThostFtdcMoneyType	ExchMargin;
///保证金率
TThostFtdcRatioType	MarginRateByMoney;
///保证金率(按手数)
TThostFtdcRatioType	MarginRateByVolume;
///昨结算价
TThostFtdcPriceType	LastSettlementPrice;
///结算价
TThostFtdcPriceType	SettlementPrice;
///平仓量
TThostFtdcVolumeType	CloseVolume;
///平仓金额
```

```
    TThostFtdcMoneyType    CloseAmount;
};
```

4.3.32 OnRspQryNotice 方法

请求查询客户通知响应。当客户端发出请求查询客户通知指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryNotice(
        CThostFtdcNoticeField *pNotice,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pNotice： *指向客户通知结构的地址。*

客户通知结构：

```
struct CThostFtdcNoticeField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///消息正文
    TThostFtdcContentType Content;
    ///经纪公司通知内容序列号
    TThostFtdcSequenceLabelType     SequenceLabel;
};
```

4.3.33 OnRtnTrade 方法

成交回报。当发生成交时 Kingstar 服务器会通知客户端，该方法会被调用。

函数原型：

```
void OnRtnTrade(CThostFtdcTradeField *pTrade)；
```

参数：

pTrade： *指向成交信息结构的地址。*

成交信息结构:

```
struct CThostFtdcTradeField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///成交编号
    TThostFtdcTradeIDType   TradeID;
    ///买卖方向
    TThostFtdcDirectionType      Direction;
    ///报单编号
    TThostFtdcOrderSysIDType   OrderSysID;
    ///会员代码
    TThostFtdcParticipantIDType      ParticipantID;
    ///客户代码
    TThostFtdcClientIDType  ClientID;
    ///交易角色
    TThostFtdcTradingRoleType    TradingRole;
    ///合约在交易所的代码
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///开平标志
    TThostFtdcOffsetFlagType    OffsetFlag;
    ///投机套保标志
    TThostFtdcHedgeFlagType    HedgeFlag;
```

```
    ///价格
    TThostFtdcPriceType     Price;
    ///数量
    TThostFtdcVolumeType    Volume;
    ///成交时期
    TThostFtdcDateType      TradeDate;
    ///成交时间
    TThostFtdcTimeType      TradeTime;
    ///成交类型
    TThostFtdcTradeTypeType     TradeType;
    ///成交价来源
    TThostFtdcPriceSourceType   PriceSource;
    ///交易所交易员代码
    TThostFtdcTraderIDType      TraderID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType      OrderLocalID;
    ///结算会员编号
    TThostFtdcParticipantIDType     ClearingPartID;
    ///业务单元
    TThostFtdcBusinessUnitType    BusinessUnit;
    ///序号
    TThostFtdcSequenceNoType    SequenceNo;
    ///交易日
    TThostFtdcDateType      TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType    SettlementID;
    ///经纪公司报单编号
    TThostFtdcSequenceNoType    BrokerOrderSeq;
    ///成交来源
    TThostFtdcTradeSourceType    TradeSource;
};
```

4.3.34 OnRtnOrder 方法

报单回报。当客户端进行报单录入、报单操作及其它原因（部分成交）导致报单状态发生变化时，Kingstar 服务器会主动通知客户端，该方法会被调用。

函数原型：

*void OnRtnOrder(CThostFtdcOrderField *pOrder);*

参数：

pOrder：*指向报单信息结构的地址。*

报单信息结构：

```
struct CThostFtdcOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType  InstrumentID;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///报单价格条件
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///买卖方向
    TThostFtdcDirectionType     Direction;
    ///组合开平标志
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///组合投机套保标志
    TThostFtdcCombHedgeFlagType  CombHedgeFlag;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量
    TThostFtdcVolumeType  VolumeTotalOriginal;
```

```
///有效期类型
TThostFtdcTimeConditionType	TimeCondition;
///GTD 日期
TThostFtdcDateType	GTDDate;
///成交量类型
TThostFtdcVolumeConditionType	VolumeCondition;
///最小成交量
TThostFtdcVolumeType	MinVolume;
///触发条件
TThostFtdcContingentConditionType	ContingentCondition;
///止损价
TThostFtdcPriceType	StopPrice;
///强平原因
TThostFtdcForceCloseReasonType	ForceCloseReason;
///自动挂起标志
TThostFtdcBoolType	IsAutoSuspend;
///业务单元
TThostFtdcBusinessUnitType	BusinessUnit;
///请求编号
TThostFtdcRequestIDType	RequestID;
///本地报单编号
TThostFtdcOrderLocalIDType	OrderLocalID;
///交易所代码
TThostFtdcExchangeIDType	ExchangeID;
///会员代码
TThostFtdcParticipantIDType	ParticipantID;
///客户代码
TThostFtdcClientIDType	ClientID;
///合约在交易所的代码
TThostFtdcExchangeInstIDType	ExchangeInstID;
///交易所交易员代码
TThostFtdcTraderIDType	TraderID;
///安装编号
```

```
TThostFtdcInstallIDType	InstallID;
///报单提交状态
TThostFtdcOrderSubmitStatusType	OrderSubmitStatus;
///报单提示序号
TThostFtdcSequenceNoType	NotifySequence;
///交易日
TThostFtdcDateType	TradingDay;
///结算编号
TThostFtdcSettlementIDType	SettlementID;
///报单编号
TThostFtdcOrderSysIDType	OrderSysID;
///报单来源
TThostFtdcOrderSourceType	OrderSource;
///报单状态
TThostFtdcOrderStatusType	OrderStatus;
///报单类型
TThostFtdcOrderTypeType	OrderType;
///今成交数量
TThostFtdcVolumeType	VolumeTraded;
///剩余数量
TThostFtdcVolumeType	VolumeTotal;
///报单日期
TThostFtdcDateType	InsertDate;
///委托时间
TThostFtdcTimeType	InsertTime;
///激活时间
TThostFtdcTimeType	ActiveTime;
///挂起时间
TThostFtdcTimeType	SuspendTime;
///最后修改时间
TThostFtdcTimeType	UpdateTime;
///撤销时间
TThostFtdcTimeType	CancelTime;
```

```
    ///最后修改交易所交易员代码
    TThostFtdcTraderIDType    ActiveTraderID;
    ///结算会员编号
    TThostFtdcParticipantIDType    ClearingPartID;
    ///序号
    TThostFtdcSequenceNoType  SequenceNo;
    ///前置编号
    TThostFtdcFrontIDType  FrontID;
    ///会话编号
    TThostFtdcSessionIDType    SessionID;
    ///用户端产品信息
    TThostFtdcProductInfoType  UserProductInfo;
    ///状态信息
    TThostFtdcErrorMsgType    StatusMsg;
    ///用户强评标志
    TThostFtdcBoolType    UserForceClose;
    ///操作用户代码
    TThostFtdcUserIDType  ActiveUserID;
    ///经纪公司报单编号
    TThostFtdcSequenceNoType  BrokerOrderSeq;
    ///相关报单
    TThostFtdcOrderSysIDType  RelativeOrderSysID;
};
```

4.3.35 OnErrRtnOrderInsert 方法

报单录入错误回报。由 Kingstar 服务器主动通知客户端，该方法会被调用。

函数原型：

```
void OnErrRtnOrderInsert(
        CThostFtdcInputOrderField *pInputOrder,
        CThostFtdcRspInfoField *pRspInfo)；
```

参数：

pInputOrder：*指向报单录入结构的地址，包含了提交报单录入时的输入数据，和后台返*

回的报单编号。

输入报单结构：

```
struct CThostFtdcInputOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///报单价格条件
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///买卖方向
    TThostFtdcDirectionType     Direction;
    ///组合开平标志
    TThostFtdcCombOffsetFlagType   CombOffsetFlag;
    ///组合投机套保标志
    TThostFtdcCombHedgeFlagType   CombHedgeFlag;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量
    TThostFtdcVolumeType  VolumeTotalOriginal;
    ///有效期类型
    TThostFtdcTimeConditionType     TimeCondition;
    ///GTD 日期
    TThostFtdcDateType      GTDDate;
    ///成交量类型
    TThostFtdcVolumeConditionType  VolumeCondition;
    ///最小成交量
```

```
    TThostFtdcVolumeType  MinVolume;
    ///触发条件
    TThostFtdcContingentConditionType    ContingentCondition;
    ///止损价
    TThostFtdcPriceType    StopPrice;
    ///强平原因
    TThostFtdcForceCloseReasonType    ForceCloseReason;
    ///自动挂起标志
    TThostFtdcBoolType    IsAutoSuspend;
    ///业务单元
    TThostFtdcBusinessUnitType BusinessUnit;
    ///请求编号
    TThostFtdcRequestIDType    RequestID;
    ///用户强评标志
    TThostFtdcBoolType    UserForceClose;
};
```

4.3.36 OnErrRtnOrderAction 方法

报单操作错误回报。由 Kingstar 服务器主动通知客户端，该方法会被调用。

函数原型：

```
void OnErrRtnOrderAction (
        CThostFtdcOrderActionField *pOrderAction,
        CThostFtdcRspInfoField *pRspInfo);
```

参数：

pOrderAction：指向报价操作结构的地址，包含了报价操作请求的输入数据，和后台返回的报价编号。

报价操作结构：

```
struct CThostFtdcOrderActionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者代码
```

```
TThostFtdcInvestorIDType	InvestorID;
///报单操作引用
TThostFtdcOrderActionRefType	OrderActionRef;
///报单引用
TThostFtdcOrderRefType	OrderRef;
///请求编号
TThostFtdcRequestIDType	RequestID;
///前置编号
TThostFtdcFrontIDType FrontID;
///会话编号
TThostFtdcSessionIDType	SessionID;
///交易所代码
TThostFtdcExchangeIDType	ExchangeID;
///报单编号
TThostFtdcOrderSysIDType	OrderSysID;
///操作标志
TThostFtdcActionFlagType	ActionFlag;
///价格
TThostFtdcPriceType	LimitPrice;
///数量变化
TThostFtdcVolumeType	VolumeChange;
///操作日期
TThostFtdcDateType	ActionDate;
///操作时间
TThostFtdcTimeType	ActionTime;
///交易所交易员代码
TThostFtdcTraderIDType	TraderID;
///安装编号
TThostFtdcInstallIDType	InstallID;
///本地报单编号
TThostFtdcOrderLocalIDType	OrderLocalID;
///操作本地编号
TThostFtdcOrderLocalIDType	ActionLocalID;
```

```
    ///会员代码
    TThostFtdcParticipantIDType     ParticipantID;
    ///客户代码
    TThostFtdcClientIDTypeClientID;
    ///业务单元
    TThostFtdcBusinessUnitType BusinessUnit;
    ///报单操作状态
    TThostFtdcOrderActionStatusTypeOrderActionStatus;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///状态信息
    TThostFtdcErrorMsgType     StatusMsg;
    ///合约代码
    TThostFtdcInstrumentIDTypeInstrumentID;
};
```

4.3.37 OnRspQrySettlementInfoConfirm 方法

查询结算确认响应。由 Kingstar 主动通知客户端，该方法会被调用。

函数原型：

```
void OnRspQrySettlementInfoConfirm(
        CThostFtdcSettlementInfoConfirmField *pSettlementInfoConfirm,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

参数：

pSettlementInfoConfirm ：*指向返回的结算确认信息结构。*

结算确认结构：

```
struct CThostFtdcSettlementInfoConfirmField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType     BrokerID;
    ///投资者代码
```

```
    TThostFtdcInvestorIDType    InvestorID;
    ///确认日期
    TThostFtdcDateType    ConfirmDate;
    ///确认时间
    TThostFtdcTimeType    ConfirmTime;
};
```

4.3.38 OnRspQryContractBank 方法

请求查询签约银行响应。

函数原型：

```
void OnRspQryContractBank(
    CThostFtdcContractBankField *pContractBank,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

参数：

pContractBank： *指向查询签约银行响应结构。*

查询签约银行响应结构：

```
struct CThostFtdcContractBankField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///银行代码
    TThostFtdcBankIDType  BankID;
    ///银行分中心代码
    TThostFtdcBankBrchIDType  BankBrchID;
    ///银行名称
    TThostFtdcBankNameType    BankName;
};
```

4.3.39 OnRspQryParkedOrder 方法[暂不支持]

请求查询预埋单响应。

函数原型：

```
void OnRspQryParkedOrder(
    CThostFtdcParkedOrderField *pParkedOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

参数：

pParkedOrder：*指向请求查询预埋单响应结构。*

请求查询预埋单响应结构：

```
struct CThostFtdcParkedOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///报单价格条件
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///买卖方向
    TThostFtdcDirectionType     Direction;
    ///组合开平标志
    TThostFtdcCombOffsetFlagType   CombOffsetFlag;
    ///组合投机套保标志
    TThostFtdcCombHedgeFlagType   CombHedgeFlag;
    ///价格
```

TThostFtdcPriceType LimitPrice;

///数量

TThostFtdcVolumeType VolumeTotalOriginal;

///有效期类型

TThostFtdcTimeConditionType TimeCondition;

///GTD 日期

TThostFtdcDateType GTDDate;

///成交量类型

TThostFtdcVolumeConditionType VolumeCondition;

///最小成交量

TThostFtdcVolumeType MinVolume;

///触发条件

TThostFtdcContingentConditionType ContingentCondition;

///止损价

TThostFtdcPriceType StopPrice;

///强平原因

TThostFtdcForceCloseReasonType ForceCloseReason;

///自动挂起标志

TThostFtdcBoolType IsAutoSuspend;

///业务单元

TThostFtdcBusinessUnitType BusinessUnit;

///请求编号

TThostFtdcRequestIDType RequestID;

///用户强评标志

TThostFtdcBoolType UserForceClose;

///交易所代码

TThostFtdcExchangeIDType ExchangeID;

///预埋报单编号

TThostFtdcParkedOrderIDType ParkedOrderID;

///用户类型

TThostFtdcUserTypeType UserType;

///预埋单状态

TThostFtdcParkedOrderStatusType Status;

```
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType    ErrorMsg;
};
```

4.3.40 OnRspQryParkedOrderAction 方法[暂不支持]

请求查询预埋撤单响应。

函数原型：

```
void OnRspQryParkedOrderAction(
        CThostFtdcParkedOrderActionField *pParkedOrderAction,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

参数：

pParkedOrderAction ：*指向请求查询预埋撤单响应结构。*

请求查询预埋撤单响应结构：

```
struct CThostFtdcParkedOrderActionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///报单操作引用
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///报单引用
    TThostFtdcOrderRefType    OrderRef;
    ///请求编号
    TThostFtdcRequestIDType    RequestID;
    ///前置编号
    TThostFtdcFrontIDType FrontID;
    ///会话编号
```

```
    TThostFtdcSessionIDType     SessionID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///报单编号
    TThostFtdcOrderSysIDType   OrderSysID;
    ///操作标志
    TThostFtdcActionFlagType    ActionFlag;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量变化
    TThostFtdcVolumeType  VolumeChange;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///预埋撤单单编号
    TThostFtdcParkedOrderActionIDType  ParkedOrderActionID;
    ///用户类型
    TThostFtdcUserTypeType      UserType;
    ///预埋撤单状态
    TThostFtdcParkedOrderStatusType      Status;
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType      ErrorMsg;
};
```

4.3.41 OnRspQryInvestorPositionCombineDetail 方法[暂不支持]

请求查询投资者组合持仓明细响应。

函数原型：

```
void OnRspQryInvestorPositionCombineDetail(
        CThostFtdcInvestorPositionCombineDetailField *pInvestorPositionCombineDetail,
```

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast);

参数：

pInvestorPositionCombineDetail：*指向请求查询投资者组合持仓明细响应结构。*

请求查询投资者组合持仓明细响应结构

struct CThostFtdcInvestorPositionCombineDetailField

{

///交易日

TThostFtdcDateType TradingDay;

///开仓日期

TThostFtdcDateType OpenDate;

///交易所代码

TThostFtdcExchangeIDType ExchangeID;

///结算编号

TThostFtdcSettlementIDType SettlementID;

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者代码

TThostFtdcInvestorIDType InvestorID;

///组合编号

TThostFtdcTradeIDType ComTradeID;

///撮合编号

TThostFtdcTradeIDType TradeID;

///合约代码

TThostFtdcInstrumentIDType InstrumentID;

///投机套保标志

TThostFtdcHedgeFlagType HedgeFlag;

///买卖

TThostFtdcDirectionType Direction;

///持仓量

TThostFtdcVolumeType TotalAmt;

///投资者保证金

```
    TThostFtdcMoneyType    Margin;
    ///交易所保证金
    TThostFtdcMoneyType    ExchMargin;
    ///保证金率
    TThostFtdcRatioType    MarginRateByMoney;
    ///保证金率(按手数)
    TThostFtdcRatioType    MarginRateByVolume;
    ///单腿编号
    TThostFtdcLegIDType    LegID;
    ///单腿乘数
    TThostFtdcLegMultipleType   LegMultiple;
    ///组合持仓合约编码
    TThostFtdcInstrumentIDType CombInstrumentID;
};
```

4.3.42 OnRspParkedOrderInsert 方法[暂不支持]

预埋单录入请求响应。

函数原型：

```
void OnRspParkedOrderInsert(
    CThostFtdcParkedOrderField *pParkedOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

参数：

pParkedOrder ：*指向预埋单录入请求响应结构。*

预埋单录入请求响应结构：

```
struct CThostFtdcParkedOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
```

```
///合约代码
TThostFtdcInstrumentIDType InstrumentID;
///报单引用
TThostFtdcOrderRefType	OrderRef;
///用户代码
TThostFtdcUserIDType	UserID;
///报单价格条件
TThostFtdcOrderPriceTypeType	OrderPriceType;
///买卖方向
TThostFtdcDirectionType	Direction;
///组合开平标志
TThostFtdcCombOffsetFlagType	CombOffsetFlag;
///组合投机套保标志
TThostFtdcCombHedgeFlagType	CombHedgeFlag;
///价格
TThostFtdcPriceType	LimitPrice;
///数量
TThostFtdcVolumeType	VolumeTotalOriginal;
///有效期类型
TThostFtdcTimeConditionType	TimeCondition;
///GTD 日期
TThostFtdcDateType	GTDDate;
///成交量类型
TThostFtdcVolumeConditionType	VolumeCondition;
///最小成交量
TThostFtdcVolumeType	MinVolume;
///触发条件
TThostFtdcContingentConditionType	ContingentCondition;
///止损价
TThostFtdcPriceType	StopPrice;
///强平原因
TThostFtdcForceCloseReasonType	ForceCloseReason;
///自动挂起标志
```

```
    TThostFtdcBoolType      IsAutoSuspend;
    ///业务单元
    TThostFtdcBusinessUnitType BusinessUnit;
    ///请求编号
    TThostFtdcRequestIDType     RequestID;
    ///用户强评标志
    TThostFtdcBoolType      UserForceClose;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///预埋报单编号
    TThostFtdcParkedOrderIDType     ParkedOrderID;
    ///用户类型
    TThostFtdcUserTypeType      UserType;
    ///预埋单状态
    TThostFtdcParkedOrderStatusType      Status;
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType      ErrorMsg;
};
```

4.3.43 OnRspParkedOrderAction 方法[暂不支持]

预埋撤单录入请求响应。

函数原型：

```
void OnRspParkedOrderAction(
        CThostFtdcParkedOrderActionField *pParkedOrderAction,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

参数：

pParkedOrderAction ： *指向预埋撤单录入请求响应结构。*

预埋撤单录入请求响应结构：

```
struct CThostFtdcParkedOrderActionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///报单操作引用
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///请求编号
    TThostFtdcRequestIDType     RequestID;
    ///前置编号
    TThostFtdcFrontIDType FrontID;
    ///会话编号
    TThostFtdcSessionIDType     SessionID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///报单编号
    TThostFtdcOrderSysIDType   OrderSysID;
    ///操作标志
    TThostFtdcActionFlagType    ActionFlag;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量变化
    TThostFtdcVolumeType  VolumeChange;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///预埋撤单单编号
    TThostFtdcParkedOrderActionIDType  ParkedOrderActionID;
    ///用户类型
```

```
    TThostFtdcUserTypeType      UserType;
    ///预埋撤单状态
    TThostFtdcParkedOrderStatusType     Status;
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType      ErrorMsg;
};
```

4.3.44 OnRspRemoveParkedOrder 方法[暂不支持]

删除预埋单响应。

函数原型：

```
void OnRspRemoveParkedOrder(
        CThostFtdcRemoveParkedOrderField *pRemoveParkedOrder
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

参数：

pRemoveParkedOrder ：*指向删除预埋单响应结构。*

删除预埋单响应结构：

```
struct CThostFtdcRemoveParkedOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///预埋报单编号
    TThostFtdcParkedOrderIDType     ParkedOrderID;
};
```

4.3.45 OnRspRemoveParkedOrderAction 方法[暂不支持]

删除预埋撤单响应。

函数原型：

```
void OnRspRemoveParkedOrderAction(
    CThostFtdcRemoveParkedOrderActionField *pRemoveParkedOrderAction,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
     bool bIsLast);
```

参数：

pRemoveParkedOrderAction：*指向删除预埋撤单响应结构。*

删除预埋撤单响应结构：

```
struct CThostFtdcRemoveParkedOrderActionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///预埋撤单编号
    TThostFtdcParkedOrderActionIDType  ParkedOrderActionID;
};
```

4.3.46 OnRspQryInvestorOpenPosition 方法

请求查询投资者开盘前持仓明细响应。当客户端发出请求请求查询投资者开盘前持仓明细指令后，Kingstar 服务器返回响应时，该方法会被调用。

函数原型：

```
void OnRspQryInvestorOpenPosition(
        CThostFtdcInvestorPositionDetailField *pInvestorPositionDetail,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pInvestorPositionDetail ***：*** *指向投资者开盘前持仓明细结构的地址。*

投资者开盘前持仓明细结构：

```
struct CThostFtdcInvestorPositionDetailField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///经纪公司代码
    TThostFtdcBrokerIDType     BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType   InvestorID;
    ///投机套保标志
    TThostFtdcHedgeFlagType    HedgeFlag;
    ///买卖
    TThostFtdcDirectionType    Direction;
    ///开仓日期
    TThostFtdcDateType     OpenDate;
    ///成交编号
    TThostFtdcTradeIDType TradeID;
    ///数量
    TThostFtdcVolumeType   Volume;
    ///开仓价
    TThostFtdcPriceType    OpenPrice;
    ///交易日
    TThostFtdcDateType     TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType SettlementID;
    ///成交类型
    TThostFtdcTradeTypeType    TradeType;
    ///组合合约代码
    TThostFtdcInstrumentIDType CombInstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///逐日盯市平仓盈亏
```

```
    TThostFtdcMoneyType    CloseProfitByDate;
    ///逐笔对冲平仓盈亏
    TThostFtdcMoneyType    CloseProfitByTrade;
    ///逐日盯市持仓盈亏
    TThostFtdcMoneyType    PositionProfitByDate;
    ///逐笔对冲持仓盈亏
    TThostFtdcMoneyType    PositionProfitByTrade;
    ///投资者保证金
    TThostFtdcMoneyType    Margin;
    ///交易所保证金
    TThostFtdcMoneyType    ExchMargin;
    ///保证金率
    TThostFtdcRatioType    MarginRateByMoney;
    ///保证金率(按手数)
    TThostFtdcRatioType    MarginRateByVolume;
    ///昨结算价
    TThostFtdcPriceType    LastSettlementPrice;
    ///结算价
    TThostFtdcPriceType    SettlementPrice;
    ///平仓量
    TThostFtdcVolumeType   CloseVolume;
    ///平仓金额
    TThostFtdcMoneyType    CloseAmount;
};
```

4.3.47 OnRspQryInvestorOpenCombinePosition 方法

请求查询投资者开盘前组合持仓明细响应。

函数原型：

```
void OnRspQryInvestorPositionCombineDetail(
        CThostFtdcInvestorPositionCombineDetailField *pInvestorPositionCombineDetail,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
```

bool bIsLast);

参数：

pInvestorPositionCombineDetail：*指向请求查询投资者开盘前组合持仓明细响应结构。*

请求查询投资者开盘前组合持仓明细响应结构

```
struct CThostFtdcInvestorPositionCombineDetailField
{
    ///交易日
    TThostFtdcDateType      TradingDay;
    ///开仓日期
    TThostFtdcDateType      OpenDate;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///结算编号
    TThostFtdcSettlementIDType SettlementID;
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///组合编号
    TThostFtdcTradeIDType ComTradeID;
    ///撮合编号
    TThostFtdcTradeIDType TradeID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///投机套保标志
    TThostFtdcHedgeFlagType    HedgeFlag;
    ///买卖
    TThostFtdcDirectionType      Direction;
    ///持仓量
    TThostFtdcVolumeType  TotalAmt;
    ///投资者保证金
    TThostFtdcMoneyType   Margin;
    ///交易所保证金
```

```
    TThostFtdcMoneyType    ExchMargin;
    ///保证金率
    TThostFtdcRatioType    MarginRateByMoney;
    ///保证金率(按手数)
    TThostFtdcRatioType    MarginRateByVolume;
    ///单腿编号
    TThostFtdcLegIDType    LegID;
    ///单腿乘数
    TThostFtdcLegMultipleType   LegMultiple;
    ///组合持仓合约编码
    TThostFtdcInstrumentIDType CombInstrumentID;
};
```

4.3.48 OnRspQryBrokerTradingAlgos 方法

请求查询经纪公司交易算法响应。

函数原型：

```
void    OnRspQryBrokerTradingAlgos(
            CThostFtdcBrokerTradingAlgosField * pBrokerTradingAlgos,
            CThostFtdcRspInfoField * pRspInfo,
            int    nRequestID,
            bool    bIsLast);
```

参数：

pBrokerTradingAlgos：*指向请求经纪公司交易算法响应结构。*

请求查询经纪公司交易算法响应结构

```
struct CThostFtdcBrokerTradingAlgosField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
```

///持仓处理算法编号

TThostFtdcHandlePositionAlgoIDType HandlePositionAlgoID;

///寻找保证金率算法编号

TThostFtdcFindMarginRateAlgoIDType FindMarginRateAlgoID;

///资金处理算法编号

TThostFtdcHandleTradingAccountAlgoIDType HandleTradingAccountAlgoID;

};

4.3.49 OnRspBulkCancelOrder 方法

批量订单撤销响应

函数原型：

void OnRspBulkCancelOrder(

CThostFtdcBulkCancelOrderField *pBulkCancelOrder,

CThostFtdcRspInfoField *pRspInfo,

int nRequestID,

bool bIsLast)

参数：

pBulkCancelOrder：*指向批量报单结构的地址*

批量报单结构如下：

struct CThostFtdcBulkCancelOrderField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者代码

TThostFtdcInvestorIDType InvestorID;

///用户代码

TThostFtdcUserIDType UserID;

///报单类型

TThostFtdcOrderTypeType OrderType;

///报单个数

TThostFtdcVolumeType nCount;

///报单集合

CThostOrderKeyField OrderKey[MAX_ORDER_COUNT];

}

4.4 CthostFtdcTraderApi 接口

CThostFtdcTraderApi 接口提供给用户的功能包括，报单与报价的录入、报单 与报价的撤销、报单与报价的挂起、报单与报价的激活、报单与报价的修改、报单与报价的查询、成交单查询、会员客户查询、会员持仓查询、客户持仓查询、合约查询、合约交易状态查询、交易所公告查询等功能。

4.4.1 CreateFtdcTraderApi 方法

客户端应用程序使用此函数来创造一个“CThostFtdcTradeApi”实例。注意：创造实例时不要使用“new”。

函数原型：

*static CThostFtdcTradeApi *CreateFtdcTradeApi(const char *pszFlowPath = "");*

参数：

pszFlowPath：*常量字符指针，用于指定一个文件目录来存贮交易托管系统发布消息的状态。默认值代表当前目录。*

返回值：

返回一个指向 CThostFtdcTradeApi 实例的指针。

4.4.2 SetWritablePath 方法

客户端应用程序使用此函数来设置本地文件存放路径

函数原型：

*int SetWritablePath(const char * szpath = "");*

参数：

Szpath：常量字符指针，用于指定一个文件目录来存贮本地文件。默认值代表当前目录。

4.4.3 Release 方法

释放一个 CThostFtdcTradeApi 实例。不能使用 delete 方法

函数原型：

void Release()；

4.4.4 init 方法

使客户端开始与 Kingstar 服务器建立连接，连接成功后可以进行登陆。

函数原型：

void Init()；

4.4.5 join 方法

客户端等待一个接口实例线程的结束。

函数原型：

void Join()；

4.4.6 GetTradingDay 方法

获得当前交易日。只有当与 Kingstar 服务器连接建立后才会取到正确的值。

函数原型：

*const char *GetTradingDay()；*

返回值：

返回一个指向日期信息字符串的常量指针。

4.4.7 RegisterSpi 方法

注册一个派生自 CThostFtdcTraderSpi 接口类的实例，该实例将完成事件处理。

函数原型：

*void RegisterSpi(CThostFtdcTraderSpi *pSpi) ;*

参数：

pSpi：*实现了 CThostFtdcTraderSpi 接口的实例指针。*

4.4.8 RegisterFront 方法

设置交易托管系统的网络通讯地址，Kingstar 服务器拥有多个通信地址，但用户只需要选

择一个通信地址。客户端 RegisterFront 多次时，使用第一个有效的通信地址。

函数原型：

*void RegisterFront(char *pszFrontAddress);*

参数：

pszFrontAddress： *指向后台服务器地址的指针。服务器地址的格式为："protocol://ipaddress:port"，如："tcp://127.0.0.1:17993"。 "tcp"代表传输协议，"127.0.0.1"代表服务器地址，"17993"代表服务器端口号。同时 API 也提供了对代理服务器的支持，包括 socks4、socks5 及 http，客户端开发时只需通过传递给 API 不同的连接字符串就可实现，例如 "socks5://127.0.0.1:17993 /user:pass@127.0.0.1:18993" 。*

4.4.9 SubscribePrivateTopic 方法

订阅私有流。该方法要在 Init 方法前调用。若不调用则不会收到私有流的数据。

函数原型：

void SubscribePrivateTopic(TE_RESUME_TYPE nResumeType);

参数：

nResumeType： *私有流重传方式：*

TERT_RESTART: 从本交易日开始重传；

TERT_RESUME: 从上次收到的续传；

TERT_QUICK: 只传送登录后私有流的内容。

4.4.10 SubscribePublicTopic 方法

订阅公共流。该方法要在 Init 方法前调用。若不调用则不会收到公共流的数据。

函数原型：

void SubscribePublicTopic(TE_RESUME_TYPE nResumeType);

参数：

nResumeType： *私有流重传方式：*

TERT_RESTART: 从本交易日开始重传；

TERT_RESUME: 从上次收到的续传；

TERT_QUICK: 只传送登录后公有流的内容。

4.4.11 ReqUserLogin 方法

用户发出登陆请求。

函数原型：

```
int ReqUserLogin(
        CThostFtdcReqUserLoginField *pReqUserLoginField,
        int nRequestID)；
```

参数：

pReqUserLoginField： *指向用户登录请求结构的地址。*

用户登录请求结构：

```
struct CThostFtdcReqUserLoginField
{
    ///交易日
    TThostFtdcDateType      TradingDay;
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///密码
    TThostFtdcPasswordType      Password;
    ///用户端产品信息
    TThostFtdcProductInfoType   UserProductInfo;
    ///接口端产品信息
    TThostFtdcProductInfoType   InterfaceProductInfo;
    ///协议信息
    TThostFtdcProtocolInfoType  ProtocolInfo;
    ///Mac 地址
    TThostFtdcMacAddressType  MacAddress;
    ///动态密码
    TThostFtdcPasswordType      OneTimePassword;
    ///终端 IP 地址
    TThostFtdcIPAddressType     ClientIPAddress;
};
```

nRequestID： *用户登录请求的 ID，该 ID 由用户指定，管理。用户需要填写 UserProductInfo 字段，即客户端的产品信息，软件开发商、 版本号等，例如：KSTraderV100。 InterfaceProductInfo 和 ProtocolInfo 只须占位，不必有效赋值。*

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

4.4.12 ReqUserLogout 方法

用户发出登出请求。

函数原型：

```
int ReqUserLogout(
        CThostFtdcUserLogoutField *pUserLogout,
        int nRequestID)；
```

参数：

pReqUserLogout： *指向用户登出请求结构的地址。*

用户登出请求结构：

```
struct CThostFtdcUserLogoutField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///用户代码
    TThostFtdcUserIDType  UserID;
};
```

4.4.13 ReqUserPasswordUpdate 方法

用户密码修改请求。

函数原型：

```
int ReqUserPasswordUpdate(
        CThostFtdcUserPasswordUpdateField
```

*pUserPasswordUpdate,

int nRequestID)；

参数:

pUserPasswordUpdate： *指向用户口令修改结构的地址。*

用户口令修改结构：

```
struct CThostFtdcUserPasswordUpdateField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///原来的口令
    TThostFtdcPasswordType      OldPassword;
    ///新的口令
    TThostFtdcPasswordType      NewPassword;
};
```

4.4.14 ReqTradingAccountPasswordUpdate 方法

资金账户口令更新请求。

函数原型：

```
int ReqTradingAccountPasswordUpdate(
        CThostFtdcTradingAccountPasswordUpdateField
        *pTradingAccountPasswordUpdate,
        int nRequestID) ;
```

参数：

pUserPasswordUpdate： *指向资金账户口令修改结构的地址。*

资金账户口令修改结构：

```
struct CThostFtdcTradingAccountPasswordUpdateField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者帐号
```

```
    TThostFtdcAccountIDType    AccountID;
    ///原来的口令
    TThostFtdcPasswordType     OldPassword;
    ///新的口令
    TThostFtdcPasswordType     NewPassword;
};
```

4.4.15 ReqOrderInsert 方法

客户端发出报单录入请求。

函数原型：

```
int ReqOrderInsert(
        CThostFtdcInputOrderField *pInputOrder,
        int nRequestID)；
```

参数：

pInputOrder： *指向输入报单结构的地址。*

输入报单结构：

```
struct CThostFtdcInputOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType     BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///报单引用
    TThostFtdcOrderRefType     OrderRef;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///报单价格条件
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///买卖方向
    TThostFtdcDirectionType     Direction;
```

```
    ///组合开平标志
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///组合投机套保标志
    TThostFtdcCombHedgeFlagType   CombHedgeFlag;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量
    TThostFtdcVolumeType   VolumeTotalOriginal;
    ///有效期类型
    TThostFtdcTimeConditionType      TimeCondition;
    ///GTD 日期
    TThostFtdcDateType      GTDDate;
    ///成交量类型
    TThostFtdcVolumeConditionType   VolumeCondition;
    ///最小成交量
    TThostFtdcVolumeType   MinVolume;
    ///触发条件
    TThostFtdcContingentConditionType    ContingentCondition;
    ///止损价
    TThostFtdcPriceType     StopPrice;
    ///强平原因
    TThostFtdcForceCloseReasonType       ForceCloseReason;
    ///自动挂起标志
    TThostFtdcBoolType      IsAutoSuspend;
    ///业务单元
    TThostFtdcBusinessUnitType BusinessUnit;
    ///请求编号
    TThostFtdcRequestIDType     RequestID;
    ///用户强评标志
    TThostFtdcBoolType      UserForceClose;
};
```

OrderRef *：报单引用，只能单调递增。每次登成功后，可以从 OnRspUserLogin 的输出参数 CThostFtdcRspUserLoginField 中获得上次登录用过的最大 OrderRef:MaxOrderRef。*

4.4.16 ReqOrderAction 方法

客户端发出报单操作请求，包括报单的撤销、报单的挂起、报单的激活、报单的修改。

函数原型：

```
int ReqOrderAction(
        CThostFtdcInputOrderActionField    *pInputOrderAction,
        int    nRequestID)；
```

参数：

pInputOrderAction：*指向报单操作结构的地址。*

报单操作结构：

```
struct CThostFtdcInputOrderActionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///报单操作引用
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///请求编号
    TThostFtdcRequestIDType     RequestID;
    ///前置编号
    TThostFtdcFrontIDType    FrontID;
    ///会话编号
    TThostFtdcSessionIDType     SessionID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///报单编号
    TThostFtdcOrderSysIDType   OrderSysID;
    ///操作标志
    TThostFtdcActionFlagType    ActionFlag;
    ///价格
```

```
    TThostFtdcPriceType     LimitPrice;
    ///数量变化
    TThostFtdcVolumeType    VolumeChange;
    ///用户代码
    TThostFtdcUserIDType    UserID;
    ///合约代码
    TThostFtdcInstrumentIDType    InstrumentID;
};
```

4.4.17 ReqQueryMaxOrderVolume 方法

查询最大报单数量请求。

函数原型：

```
int ReqQueryMaxOrderVolume(
        CThostFtdcQueryMaxOrderVolumeField *pQueryMaxOrderVolume,
        int nRequestID)；
```

参数：

pQueryMaxOrderVolume： *指向查询最大报单数量结构的地址。*

查询最大报单数量结构：

```
struct CThostFtdcQueryMaxOrderVolumeField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///买卖方向
    TThostFtdcDirectionType     Direction;
    ///开平标志
    TThostFtdcOffsetFlagType    OffsetFlag;
    ///投机套保标志
    TThostFtdcHedgeFlagType     HedgeFlag;
```

```
    ///最大允许报单数量
    TThostFtdcVolumeType  MaxVolume;
};
```

4.4.18 ReqSettlementInfoConfirm 方法

投资者结算结果确认。

函数原型：

```
int ReqSettlementInfoConfirm(
        CThostFtdcSettlementInfoConfirmField *pSettlementInfoConfirm,
        int nRequestID)；
```

参数：

pSettlementInfoConfirm： *指向投资者结算结果确认结构的地址。*

投资者结算结果确认结构：

```
struct CThostFtdcSettlementInfoConfirmField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///确认日期
    TThostFtdcDateType      ConfirmDate;
    ///确认时间
    TThostFtdcTimeType      ConfirmTime;
};
```

4.4.19 ReqFromBankToFutureByFuture 方法

请求银行资金转期货。

函数原型：

```
int ReqFromBankToFutureByFuture (
    CThostFtdcReqTransferField *pReqTransfer,
    int nRequestID)；
```

参数：

pReqTransfer：*指向银行资金转期货请求结构的地址。*

银行资金转期货请求结构：

```
struct CThostFtdcReqTransferField
{
    ///业务功能码
    TThostFtdcTradeCodeType    TradeCode;
    ///银行代码
    TThostFtdcBankIDType  BankID;
    ///银行分支机构代码
    TThostFtdcBankBrchIDType  BankBranchID;
    ///期商代码
    TThostFtdcBrokerIDType      BrokerID;
    ///期商分支机构代码
    TThostFtdcFutureBranchIDType    BrokerBranchID;
    ///交易日期
    TThostFtdcTradeDateType    TradeDate;
    ///交易时间
    TThostFtdcTradeTimeType    TradeTime;
    ///银行流水号
    TThostFtdcBankSerialType    BankSerial;
    ///交易系统日期
    TThostFtdcTradeDateType    TradingDay;
    ///银期平台消息流水号
    TThostFtdcSerialType    PlateSerial;
    ///最后分片标志
    TThostFtdcLastFragmentType      LastFragment;
    ///会话号
    TThostFtdcSessionIDType     SessionID;
    ///客户姓名
    TThostFtdcIndividualNameType    CustomerName;
    ///证件类型
    TThostFtdcIdCardTypeType   IdCardType;
```

```
///证件号码
TThostFtdcIdentifiedCardNoType  IdentifiedCardNo;
///客户类型
TThostFtdcCustTypeType      CustType;
///银行帐号
TThostFtdcBankAccountType    BankAccount;
///银行密码
TThostFtdcPasswordType      BankPassWord;
///投资者帐号
TThostFtdcAccountIDType    AccountID;
///期货密码
TThostFtdcPasswordType      Password;
///安装编号
TThostFtdcInstallIDType      InstallID;
///期货公司流水号
TThostFtdcFutureSerialType  FutureSerial;
///用户标识
TThostFtdcUserIDType    UserID;
///验证客户证件号码标志
TThostFtdcYesNoIndicatorType    VerifyCertNoFlag;
///币种代码
TThostFtdcCurrencyIDType   CurrencyID;
///转帐金额
TThostFtdcTradeAmountType    TradeAmount;
///期货可取金额
TThostFtdcTradeAmountType    FutureFetchAmount;
///费用支付标志
TThostFtdcFeePayFlagType  FeePayFlag;
///应收客户费用
TThostFtdcCustFeeType    CustFee;
///应收期货公司费用
TThostFtdcFutureFeeType    BrokerFee;
///发送方给接收方的消息
```

```
    TThostFtdcAddInfoType    Message;
    ///摘要
    TThostFtdcDigestType    Digest;
    ///银行帐号类型
    TThostFtdcBankAccTypeType    BankAccType;
    ///渠道标志
    TThostFtdcDeviceIDType    DeviceID;
    ///期货单位帐号类型
    TThostFtdcBankAccTypeType    BankSecuAccType;
    ///期货公司银行编码
    TThostFtdcBankCodingForFutureType    BrokerIDByBank;
    ///期货单位帐号
    TThostFtdcBankAccountType    BankSecuAcc;
    ///银行密码标志
    TThostFtdcPwdFlagType    BankPwdFlag;
    ///期货资金密码核对标志
    TThostFtdcPwdFlagType    SecuPwdFlag;
    ///交易柜员
    TThostFtdcOperNoType    OperNo;
    ///请求编号
    TThostFtdcRequestIDType    RequestID;
    ///交易 ID
    TThostFtdcTIDType    TID;
    ///转账交易状态
    TThostFtdcTransferStatusType    TransferStatus;
};
```

4.4.20 ReqFromFutureToBankByFuture 方法

请求期货资金转银行。

函数原型：

```
int ReqFromFutureToBankByFuture (
    CThostFtdcReqTransferField *pReqTransfer,
```

int nRequestID)；

参数：

***pReqTransfer：** 指向期货资金转银行请求结构的地址。*

期货资金转银行请求结构：

```
struct CThostFtdcReqTransferField
{
    ///业务功能码
    TThostFtdcTradeCodeType    TradeCode;
    ///银行代码
    TThostFtdcBankIDType  BankID;
    ///银行分支机构代码
    TThostFtdcBankBrchIDType  BankBranchID;
    ///期商代码
    TThostFtdcBrokerIDType      BrokerID;
    ///期商分支机构代码
    TThostFtdcFutureBranchIDType    BrokerBranchID;
    ///交易日期
    TThostFtdcTradeDateType    TradeDate;
    ///交易时间
    TThostFtdcTradeTimeType    TradeTime;
    ///银行流水号
    TThostFtdcBankSerialType    BankSerial;
    ///交易系统日期
    TThostFtdcTradeDateType    TradingDay;
    ///银期平台消息流水号
    TThostFtdcSerialType    PlateSerial;
    ///最后分片标志
    TThostFtdcLastFragmentType      LastFragment;
    ///会话号
    TThostFtdcSessionIDType     SessionID;
    ///客户姓名
    TThostFtdcIndividualNameType    CustomerName;
    ///证件类型
```

TThostFtdcIdCardTypeType IdCardType;

///证件号码

TThostFtdcIdentifiedCardNoType IdentifiedCardNo;

///客户类型

TThostFtdcCustTypeType CustType;

///银行帐号

TThostFtdcBankAccountType BankAccount;

///银行密码

TThostFtdcPasswordType BankPassWord;

///投资者帐号

TThostFtdcAccountIDType AccountID;

///期货密码

TThostFtdcPasswordType Password;

///安装编号

TThostFtdcInstallIDType InstallID;

///期货公司流水号

TThostFtdcFutureSerialType FutureSerial;

///用户标识

TThostFtdcUserIDType UserID;

///验证客户证件号码标志

TThostFtdcYesNoIndicatorType VerifyCertNoFlag;

///币种代码

TThostFtdcCurrencyIDType CurrencyID;

///转帐金额

TThostFtdcTradeAmountType TradeAmount;

///期货可取金额

TThostFtdcTradeAmountType FutureFetchAmount;

///费用支付标志

TThostFtdcFeePayFlagType FeePayFlag;

///应收客户费用

TThostFtdcCustFeeType CustFee;

///应收期货公司费用

TThostFtdcFutureFeeType BrokerFee;

```
    ///发送方给接收方的消息
    TThostFtdcAddInfoType    Message;
    ///摘要
    TThostFtdcDigestType    Digest;
    ///银行帐号类型
    TThostFtdcBankAccTypeType    BankAccType;
    ///渠道标志
    TThostFtdcDeviceIDType    DeviceID;
    ///期货单位帐号类型
    TThostFtdcBankAccTypeType    BankSecuAccType;
    ///期货公司银行编码
    TThostFtdcBankCodingForFutureType    BrokerIDByBank;
    ///期货单位帐号
    TThostFtdcBankAccountType    BankSecuAcc;
    ///银行密码标志
    TThostFtdcPwdFlagType    BankPwdFlag;
    ///期货资金密码核对标志
    TThostFtdcPwdFlagType    SecuPwdFlag;
    ///交易柜员
    TThostFtdcOperNoType    OperNo;
    ///请求编号
    TThostFtdcRequestIDType    RequestID;
    ///交易 ID
    TThostFtdcTIDType    TID;
    ///转账交易状态
    TThostFtdcTransferStatusType    TransferStatus;
};
```

4.4.21 ReqQueryBankAccountMoneyByFuture 方法

请求查询银行资金。

函数原型：

```
int ReqQueryBankAccountMoneyByFuture (
```

*CThostFtdcReqQueryAccountField *pReqQueryAccount，*

int nRequestID)；

参数：

pReqQueryAccount：*指向查询银行资金请求结构的地址。*

查询银行资金请求结构：

```
struct CThostFtdcReqQueryAccountField
{
    ///业务功能码
    TThostFtdcTradeCodeType    TradeCode;
    ///银行代码
    TThostFtdcBankIDType    BankID;
    ///银行分支机构代码
    TThostFtdcBankBrchIDType  BankBranchID;
    ///期商代码
    TThostFtdcBrokerIDType      BrokerID;
    ///期商分支机构代码
    TThostFtdcFutureBranchIDType    BrokerBranchID;
    ///交易日期
    TThostFtdcTradeDateType    TradeDate;
    ///交易时间
    TThostFtdcTradeTimeType    TradeTime;
    ///银行流水号
    TThostFtdcBankSerialType    BankSerial;
    ///交易系统日期
    TThostFtdcTradeDateType    TradingDay;
    ///银期平台消息流水号
    TThostFtdcSerialType    PlateSerial;
    ///最后分片标志
    TThostFtdcLastFragmentType      LastFragment;
    ///会话号
    TThostFtdcSessionIDType     SessionID;
    ///客户姓名
    TThostFtdcIndividualNameType    CustomerName;
```

///证件类型

TThostFtdcIdCardTypeType IdCardType;

///证件号码

TThostFtdcIdentifiedCardNoType IdentifiedCardNo;

///客户类型

TThostFtdcCustTypeType CustType;

///银行帐号

TThostFtdcBankAccountType BankAccount;

///银行密码

TThostFtdcPasswordType BankPassWord;

///投资者帐号

TThostFtdcAccountIDType AccountID;

///期货密码

TThostFtdcPasswordType Password;

///期货公司流水号

TThostFtdcFutureSerialType FutureSerial;

///安装编号

TThostFtdcInstallIDType InstallID;

///用户标识

TThostFtdcUserIDType UserID;

///验证客户证件号码标志

TThostFtdcYesNoIndicatorType VerifyCertNoFlag;

///币种代码

TThostFtdcCurrencyIDType CurrencyID;

///摘要

TThostFtdcDigestType Digest;

///银行帐号类型

TThostFtdcBankAccTypeType BankAccType;

///渠道标志

TThostFtdcDeviceIDType DeviceID;

///期货单位帐号类型

TThostFtdcBankAccTypeType BankSecuAccType;

///期货公司银行编码

```
    TThostFtdcBankCodingForFutureType    BrokerIDByBank;
    ///期货单位帐号
    TThostFtdcBankAccountType    BankSecuAcc;
    ///银行密码标志
    TThostFtdcPwdFlagType    BankPwdFlag;
    ///期货资金密码核对标志
    TThostFtdcPwdFlagType    SecuPwdFlag;
    ///交易柜员
    TThostFtdcOperNoType    OperNo;
    ///请求编号
    TThostFtdcRequestIDType    RequestID;
    ///交易 ID
    TThostFtdcTIDType    TID;
};
```

4.4.22 ReqQryTransferSerial 方法

请求查询转帐流水。

函数原型：

```
int ReqQryTransferSerial(
    CThostFtdcQryTransferSerialField *pQryTransferSerial,
    int nRequestID)；
```

参数：

pQryTransferSerial： *指向查询转账流水请求结构的地址。*

查询转账流水请求结构：

```
struct CThostFtdcQryTransferSerialField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者帐号
    TThostFtdcAccountIDType    AccountID;
    ///银行编码
    TThostFtdcBankIDType    BankID;
```

};

4.4.23 ReqTransferQryDetail 方法

请求查询银行交易明细。

函数原型：

int ReqTransferQryDetail(

*CThostFtdcTransferQryDetailReqField *pTransferQryDetailReq,*

int nRequestID)；

参数：

pTransferQryDetailReq：*指向查询银行交易明细请求结构的地址。*

查询银行交易明细请求结构：

struct CThostFtdcTransferQryDetailReqField

{

///期货资金账户

TThostFtdcAccountIDType FutureAccount;

};

4.4.24 ReqQryOrder 方法

报单查询请求。

函数原型：

int ReqQryOrder(

*CThostFtdcQryOrderField *pQryOrder,*

int nRequestID)；

参数：

pQryOrder：*指向报单查询结构的地址。*

报单查询结构：

struct CThostFtdcQryOrderField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者代码

```
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///报单编号
    TThostFtdcOrderSysIDType   OrderSysID;
    ///开始时间
    TThostFtdcTimeType     InsertTimeStart;
    ///结束时间
    TThostFtdcTimeType     InsertTimeEnd;
};
```

4.4.25 ReqQryTrade 方法

成交单查询请求。

函数原型：

```
int ReqQryTrade(
        CThostFtdcQryTradeField *pQryTrade,
        int nRequestID)；
```

参数：

pQryTrade： *指向成交查询结构的地址。*

成交查询结构：

```
struct CThostFtdcQryTradeField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType     BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
```

```
    ///成交编号
    TThostFtdcTradeIDType TradeID;
    ///开始时间
    TThostFtdcTimeType    TradeTimeStart;
    ///结束时间
    TThostFtdcTimeType    TradeTimeEnd;
};
```

4.4.26 ReqQryInvestor 方法

会员客户查询请求。

函数原型：

```
int ReqQry Investor (
        CThostFtdcQryInvestorField *pQryInvestor,
        int nRequestID)；
```

参数：

pQryInvestor： *指向客户查询结构的地址。*

客户查询结构：

```
struct CThostFtdcQryInvestorField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
};
```

4.4.27 ReqQryInvestorPosition 方法

会员持仓查询请求。

函数原型：

```
int ReqQryInvestorPosition(
        CThostFtdcQryInvestorPositionField *pQryInvestorPosition,
        int nRequestID)；
```

参数：

pQryInvestorPosition：*指向会员持仓查询结构的地址。*

会员持仓查询结构：

```
struct CThostFtdcQryInvestorPositionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
};
```

4.4.28 ReqQryTradingAccount 方法

请求查询资金账户。

函数原型：

```
int ReqQryTradingAccount(
        CThostFtdcQryTradingAccountField *pQryTradingAccount,
        int nRequestID)；
```

参数：

pQryTradingAccount：*指向查询资金账户结构的地址。*

查询资金账户结构：

```
struct CThostFtdcQryTradingAccountField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
};
```

4.4.29 ReqQryTradingCode 方法

请求查询交易编码。

函数原型：

int ReqQryTradingCode(

*CThostFtdcQryTradingCodeField *pQryTradingCode,*

int nRequestID)；

参数：

pQryTradingCode：*指向查询交易编码结构的地址。*

查询交易编码结构：

struct CThostFtdcQryTradingCodeField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者代码

TThostFtdcInvestorIDType InvestorID;

///交易所代码

TThostFtdcExchangeIDType ExchangeID;

///客户代码

TThostFtdcClientIDTypeClientID;

///交易编码类型

TThostFtdcClientIDTypeType ClientIDType;

};

4.4.30 ReqQryExchange 方法

请求查询交易所。

函数原型：

int ReqQryExchange(

*CThostFtdcQryExchangeField *pQryExchange,*

int nRequestID)；

参数：

pQryExchange：指向查询交易编码结构的地址。

查询交易所编码结构：

```
struct CThostFtdcQryExchangeField
{
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
};
```

4.4.31 ReqQryInstrument 方法

请求查询合约。

函数原型：

```
int ReqQryInstrument(
        CThostFtdcQryInstrumentField *pQryInstrument,
        int nRequestID)；
```

参数：

pQryInstrument： *指向查询查询合约结构的地址。*

查询合约结构：

```
struct CThostFtdcQryInstrumentField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///合约在交易所的代码
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///产品代码
    TThostFtdcInstrumentIDType ProductID;
};
```

4.4.32 ReqQryDepthMarketData 方法

请求查询行情。

函数原型：

int ReqQryDepthMarketData(

*CThostFtdcQryDepthMarketDataField *pQryDepthMarketData,*

int nRequestID)；

参数：

pQryDepthMarketData：*指向查询行情结构的地址。*

查询行情结构：

struct CThostFtdcQryDepthMarketDataField

{

///合约代码

TThostFtdcInstrumentIDType InstrumentID;

};

4.4.33 ReqQryInstrumentMarginRate 方法

请求查询合约保证金率。

函数原型：

int ReqQryInstrumentMarginRate(

*CThostFtdcQryInstrumentMarginRateField *pQryInstrumentMarginRate,*

int nRequestID)；

参数：

pQryInstrumentMarginRate：*指向查询合约保证金率结构的地址。*

查询合约保证金率结构：

struct CThostFtdcQryInstrumentMarginRateField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者代码

TThostFtdcInvestorIDType InvestorID;

///合约代码

TThostFtdcInstrumentIDType InstrumentID;

///投机套保标志

TThostFtdcHedgeFlagType HedgeFlag;

};

4.4.34 ReqQryInstrumentCommissionRate 方法

请求查询合约手续费率。

函数原型：

int ReqQryInstrumentCommissionRate(

*CThostFtdcQryInstrumentCommissionRateField *pQryInstrumentCommissionRate,*

int nRequestID)；

参数：

pQryInstrumentCommissionRate： *指向查询合约手续费率结构的地址。*

查询合约手续费率结构：

struct CThostFtdcQryInstrumentCommissionRateField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者代码

TThostFtdcInvestorIDType InvestorID;

///合约代码

TThostFtdcInstrumentIDType InstrumentID;

};

4.4.35 ReqQryCFMMCTradingAccountKey 方法

请求查询保证金监管系统经纪公司资金账户密钥。

函数原型：

int ReqQryCFMMCTradingAccountKey(

*CThostFtdcQryCFMMCTradingAccountKeyField *pQryCFMMCTradingAccountKey,*

int nRequestID)；

参数：

pQryCFMMCTradingAccountKey： *指向查询监控中心密钥结构的地址。*

查询监控中心密钥结构：

struct CThostFtdcQryCFMMCTradingAccountKeyField

{

///经纪公司代码

```
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
};
```

4.4.36 ReqQrySettlementInfo 方法

请求查询投资者结算结果。

函数原型：

```
int ReqQrySettlementInfo(
        CThostFtdcQrySettlementInfoField   *pQrySettlementInfo,
        int nRequestID)；
```

参数：

pQrySettlementInfo： *指向查询投资者结算结果结构的地址。*

查询投资者结算结果结构：

```
struct CThostFtdcQrySettlementInfoField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///交易日
    TThostFtdcDateType      TradingDay;
};
```

4.4.37 ReqQryTransferBank 方法

请求查询转帐银行。

函数原型：

```
int ReqQryTransferBank(
    CThostFtdcQryTransferBankField *pQryTransferBank,
    int nRequestID)；
```

参数：

pQryTransferBank： *指向查询转帐银行结构的地址。*

查询转帐银行结构：

```
struct CThostFtdcQryTransferBankField
{
    ///银行代码
    TThostFtdcBankIDType  BankID;
    ///银行分中心代码
    TThostFtdcBankBrchIDType  BankBrchID;
};
```

4.4.38 ReqQryInvestorPositionDetail 方法

请求查询投资者持仓明细。

函数原型：

```
int ReqQryInvestorPositionDetail(
        CThostFtdcQryInvestorPositionDetailField *pQryInvestorPositionDetail,
        int nRequestID)；
```

参数：

pQryInvestorPositionDetail： *指向查询投资者持仓明细结构的地址。*

查询投资者持仓明细结构：

```
 struct CThostFtdcQryInvestorPositionDetailField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
};
```

4.4.39 ReqQryNotice 方法

请求查询客户通知。

函数原型：

int ReqQryNotice(

*CThostFtdcQryNoticeField *pQryNotice,*

int nRequestID)；

参数：

pQryNotice：*指向查询客户通知结构的地址。*

查询客户通知结构：

struct CThostFtdcQryNoticeField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

};

4.4.40 ReqQrySettlementInfoConfirm 方法

查询结算信息确认。

函数原型：

int ReqQrySettlementInfoConfirm(

CThostFtdcQrySettlementInfoConfirmField

**pQrySettlementInfoConfirm,*

int nRequestID)；

参数：

pQrySettlementInfoConfirm ：*指向查询结算信息确认结构的地址。*

查询结算信息确认：

struct CThostFtdcQrySettlementInfoConfirmField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者代码

TThostFtdcInvestorIDType InvestorID;

};

4.4.41 ReqQryContractBank 方法

请求查询签约银行。

函数原型：

int ReqQryContractBank(

*CThostFtdcQryContractBankField *pQryContractBank,*

int nRequestID)；

参数：

pQryContractBank：*指向查询签约银行请求结构的地址。其中 BrokerID 不能为空，BankID 和 BankBrchID 可以为空。*

查询签约银行请求结构：

struct CThostFtdcQryContractBankField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///银行代码

TThostFtdcBankIDType BankID;

///银行分中心代码

TThostFtdcBankBrchIDType BankBrchID;

};

4.4.42 ReqQryParkedOrder 方法

请求查询预埋单。

函数原型：

int ReqQryParkedOrder(

*CThostFtdcQryParkedOrderField *pQryParkedOrder,*

int nRequestID)；

参数：

pQryParkedOrder： *指向查询预埋单请求结构的地址。*

查询预埋单请求结构：

struct CThostFtdcQryParkedOrderField

{

```
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
};
```

4.4.43 ReqQryParkedOrderAction 方法

请求查询预埋撤单。

函数原型：

```
int ReqQryParkedOrderAction(
        CThostFtdcQryParkedOrderActionField *pQryParkedOrderAction,
        int nRequestID)；
```

参数：

pQryParkedOrderAction：*指向请求查询预埋撤单结构的地址。*

请求查询预埋撤单结构：

```
struct CThostFtdcQryParkedOrderActionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
};
```

4.4.44 ReqQryInvestorPositionCombineDetail 方法

请求查询投资者组合持仓明细。

函数模型：

```
int ReqQryInvestorPositionCombineDetail(
        CThostFtdcQryInvestorPositionCombineDetailField
        *pQryInvestorPositionCombineDetail,
        int nRequestID);；
```

参数：

pQryInvestorPositionCombineDetail： *指向请求查询投资者组合持仓明细请求结构的地址。*

查询组合持仓明细请求结构：

```
struct CThostFtdcQryInvestorPositionCombineDetailField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///组合持仓合约编码
    TThostFtdcInstrumentIDType CombInstrumentID;
};
```

4.4.45 ReqParkedOrderInsert 方法

预埋单录入请求。

函数模型：

```
int ReqParkedOrderInsert (
        CThostFtdcParkedOrderField *pParkedOrder,
        int nRequestID)；
```

参数：

pParkedOrder： *指向预埋单录入请求结构的地址。*

预埋单结构：

```
struct CThostFtdcParkedOrderField
{
```

```
///经纪公司代码
TThostFtdcBrokerIDType	BrokerID;
///投资者代码
TThostFtdcInvestorIDType	InvestorID;
///合约代码
TThostFtdcInstrumentIDType InstrumentID;
///报单引用
TThostFtdcOrderRefType	OrderRef;
///用户代码
TThostFtdcUserIDType	UserID;
///报单价格条件
TThostFtdcOrderPriceTypeType	OrderPriceType;
///买卖方向
TThostFtdcDirectionType	Direction;
///组合开平标志
TThostFtdcCombOffsetFlagType	CombOffsetFlag;
///组合投机套保标志
TThostFtdcCombHedgeFlagType	CombHedgeFlag;
///价格
TThostFtdcPriceType	LimitPrice;
///数量
TThostFtdcVolumeType	VolumeTotalOriginal;
///有效期类型
TThostFtdcTimeConditionType	TimeCondition;
///GTD 日期
TThostFtdcDateType	GTDDate;
///成交量类型
TThostFtdcVolumeConditionType	VolumeCondition;
///最小成交量
TThostFtdcVolumeType	MinVolume;
///触发条件
TThostFtdcContingentConditionType	ContingentCondition;
///止损价
```

```
    TThostFtdcPriceType     StopPrice;
    ///强平原因
    TThostFtdcForceCloseReasonType      ForceCloseReason;
    ///自动挂起标志
    TThostFtdcBoolType      IsAutoSuspend;
    ///业务单元
    TThostFtdcBusinessUnitType BusinessUnit;
    ///请求编号
    TThostFtdcRequestIDType     RequestID;
    ///用户强评标志
    TThostFtdcBoolType      UserForceClose;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///预埋报单编号
    TThostFtdcParkedOrderIDType     ParkedOrderID;
    ///用户类型
    TThostFtdcUserTypeType      UserType;
    ///预埋单状态
    TThostFtdcParkedOrderStatusType      Status;
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType      ErrorMsg;
};
```

4.4.46 ReqParkedOrderAction 方法

预埋撤单录入请求。

函数原型：

```
int ReqParkedOrderAction(
        CThostFtdcParkedOrderActionField *pParkedOrderAction,
        int nRequestID)；
```

参数：

pParkedOrderAction：*指向预埋撤单录入请求结构的地址。*

预埋撤单录入请求结构：

```
struct CThostFtdcParkedOrderActionField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///报单操作引用
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///报单引用
    TThostFtdcOrderRefType      OrderRef;
    ///请求编号
    TThostFtdcRequestIDType     RequestID;
    ///前置编号
    TThostFtdcFrontIDType FrontID;
    ///会话编号
    TThostFtdcSessionIDType     SessionID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///报单编号
    TThostFtdcOrderSysIDType   OrderSysID;
    ///操作标志
    TThostFtdcActionFlagType    ActionFlag;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量变化
    TThostFtdcVolumeType   VolumeChange;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///预埋撤单单编号
```

```
    TThostFtdcParkedOrderActionIDType  ParkedOrderActionID;
    ///用户类型
    TThostFtdcUserTypeType      UserType;
    ///预埋撤单状态
    TThostFtdcParkedOrderStatusType      Status;
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType      ErrorMsg;
};
```

4.4.47 ReqRemoveParkedOrder 方法

请求删除预埋单。

函数原型：

```
int ReqRemoveParkedOrder(
        CThostFtdcRemoveParkedOrderField *pRemoveParkedOrder,
        int nRequestID)；
```

参数：

pRemoveParkedOrder： *指向请求删除预埋单结构的地址。*

请求删除预埋单结构：

```
struct CThostFtdcRemoveParkedOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///预埋报单编号
    TThostFtdcParkedOrderIDType    ParkedOrderID;
};
```

4.4.48 ReqRemoveParkedOrderAction 方法

请求删除预埋撤单。

函数原型：

int ReqRemoveParkedOrderAction(

*CThostFtdcRemoveParkedOrderActionField *pRemoveParkedOrderAction,*

int nRequestID);；

参数：

***pRemoveParkedOrderAction：** 指向请求删除预埋撤单结构的地址。*

请求删除预埋撤单结构：

struct CThostFtdcRemoveParkedOrderActionField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///投资者代码

TThostFtdcInvestorIDType InvestorID;

///预埋撤单编号

TThostFtdcParkedOrderActionIDType ParkedOrderActionID;

};

4.4.49 ReqQueryInvestorOpenPosition 方法

请求查询开盘前的持仓明细

函数原型：

int ReqQueryInvestorOpenPosition (

*CThostFtdcQryInvestorPositionDetailField *pQryInvestorPositionDetail,*

int nRequestID)；

参数：

***pQryInvestorPositionDetail：** 指向查询投资者开盘前持仓明细结构的地址。*

查询投资者开盘前持仓明细结构：

struct CThostFtdcQryInvestorPositionDetailField

{

///经纪公司代码

```
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
};
```

4.4.50 ReqQueryInvestorOpenCombinePosition 方法

查询开盘前的组合持仓明细

函数模型：

```
int ReqQueryInvestorOpenCombinePosition (
        CThostFtdcQryInvestorPositionCombineDetailField
        *pQryInvestorPositionCombineDetail,
        int nRequestID);；
```

参数：

pQryInvestorPositionCombineDetail：*指向请求查询投资者开盘前组合持仓明细请求结构的地址。*

查询开盘前组合持仓明细请求结构：

```
struct CThostFtdcQryInvestorPositionCombineDetailField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///组合持仓合约编码
    TThostFtdcInstrumentIDType CombInstrumentID;
};
```

4.4.51 ReqQryBrokerTradingAlogs 方法

查询经纪公司交易算法

函数模型：

int ReqQryBrokerTradingAlgos(

*CThostFtdcQryBrokerTradingAlgosField * pQryBrokerTradingAlgos,*

int nRequestID)；

参数：

pQryBrokerTradingAlgos：*指向请求查询经纪公司交易算法请求结构的地址。*

查询经纪公司交易算法请求结构：

```
struct CThostFtdcQryBrokerTradingAlgosField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///交易所代码
    TThostFtdcExchangeIDType      ExchangeID;
    ///合约代码
    TThostFtdcInstrumentIDType      InstrumentID;
};
```

4.4.52 RegisterNameServer 方法

注册名字服务器（门户）

客户端可通过调用此方法注册名字服务器从而获取最优网关以实现快速登录。Kingstar API 支持多次注册名字服务器，也就是说，此函数可以被调用多次，每次调用都能实现从此名字服务器相关的最优网关列表中返回一个网关。需要注意的是，一旦多次调用此方法，后续返回的网关地址会替代上次返回的网关地址。

目前此接口只支持 API 的 windows 版本。

函数模型：

*void RegisterNameServer(char *pszNameServerAddress);*

参数：

pszNameServerAddress：*指向名字服务器地址（也叫门户服务器地址）的指针。名字服务器地址格式如下所示：“协议:/ip 地址:端口/代理用户名:代理用户密码@代理服务器 ip 地址:代理服务器端口/网关标识/客户号”。其中代理相关字段是可选的。例如：“tcp://127.0.0.1:11000/A/80001”，“tcp”代表通讯协议，“127.0.0.1”标识名字服务器的 ip 地址，“11000”标识名字服务器的端口，“A”代表网关标识，“80001”标识客户号。*

网关标识字段如下所示：

值	描述
‘A’	大智慧
‘B’	文华一键通
‘C’	彭博闪电手
‘D’	托瑞邦泽
‘E’	快期
‘F’	金字塔
‘G’	达钱

4.4.53 ReqBulkCancelOrder 方法

批量撤单请求

函数模型：

```
int ReqBulkCancelOrder(
    CThostFtdcBulkCancelOrderField *pBulkCancelOrder,
    int nRequestID)
```

参数：

pBulkCancelOrder： 指向批量报单结构的地址。

批量报单结构如下：

```
struct CThostFtdcBulkCancelOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///用户代码
    TThostFtdcUserIDType        UserID;
    ///报单类型
    TThostFtdcOrderTypeType     OrderType;
    ///报单个数
    TThostFtdcVolumeType nCount;
    ///报单集合
    CThostOrderKeyField OrderKey[MAX_ORDER_COUNT];
}
```

4.4.54 LoadExtApi 方法

注册条件单实例

函数模型：

```
void * LoadExtApi(
    void * spi,
    const char *ExtApiName)
```

参数：

Spi:指向 spi 实例的指针

ExtApiName：API 的名称

4.5 CthostFtdcMdSpi 接口

CThostFtdcMdSpi 实现了事件通知接口。用户必需派生 CThostFtdcMdSpi 接口，编写事件处理方法来处理感兴趣的事件。

4.5.1 OnFrontConnected 方法

当客户端与 Kingstar 服务器建立起通信连接时（还未登录前），该方法被调用。

函数原型：

void OnFrontConnected()；

本方法在完成初始化后调用，可以在其中完成用户登录任务。

4.5.2 OnFrontDisconnected 方法

当客户端与 Kingstar 服务器通信连接断开时，该方法被调用。当发生这个情况后，API 会自动重新连接，客户端可不做处理。自动重连地址，可能是原来注册的地址，也可能是系统支持的其它可用的通信地址，它由程序自动选择。

函数原型：

void OnFrontDisconnected (int nReason)；

参数：

nReason：*连接断开原因*

0x1001　网络读失败

0x1002　网络写失败

0x2001　接收心跳超时

0x2002　发送心跳失败

0x2003　收到错误报文

0x2004　服务器主动断开

4.5.3　OnHeartBeatWarning 方法

心跳超时警告。当长时间未收到报文时，该方法被调用。

函数原型：

void OnHeartBeatWarning(int nTimeLapse)；

参数：

nTimeLapse：*距离上次接收报文的时间*

4.5.4　OnRspUserLogin 方法

当客户端发出登录请求之后，Kingstar 服务器返回响应时，该方法会被调用，通知客户端登录是否成功。

函数原型：

```
void OnRspUserLogin(
        CThostFtdcRspUserLoginField *pRspUserLogin,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)；
```

参数：

pRspUserLogin：*返回用户登录信息的地址。*

用户登录信息结构：

```
struct CThostFtdcRspUserLoginField
{
    ///交易日
    TThostFtdcDateType    TradingDay;
    ///登录成功时间
    TThostFtdcTimeType    LoginTime;
    ///经纪公司代码
```

```
    TThostFtdcBrokerIDType      BrokerID;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///交易系统名称
    TThostFtdcSystemNameType  SystemName;
    ///前置编号
    TThostFtdcFrontIDType FrontID;
    ///会话编号
    TThostFtdcSessionIDType     SessionID;
    ///最大报单引用
    TThostFtdcOrderRefType      MaxOrderRef;
    ///上期所时间
    TThostFtdcTimeType      SHFETime;
    ///大商所时间
    TThostFtdcTimeType      DCETime;
    ///郑商所时间
    TThostFtdcTimeType      CZCETime;
    ///中金所时间
    TThostFtdcTimeType      FFEXTime;
};
```

pRspInfo：*返回用户响应信息的地址。错误误代码为 0 时，表示操作成功，以下同。*

响应信息结构：

```
struct CThostFtdcRspInfoField
{
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType      ErrorMsg;
};
```

nRequestID：*返回用户登录请求的 ID，该 ID 由用户在登录时指定。*

bIsLast：*指示该次返回是否为针对 nRequestID 的最后一次返回。*

4.5.5 OnRspUserLogout 方法

当客户端发出退出请求之后，Kingstar 返回响应时，该方法会被调用，通知客户端退出是否成功。

函数原型：

void OnRspUserLogout(

*CThostFtdcUserLogoutField *pUserLogout,*

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)；

参数：

pRspUserLogout：*返回用户退出信息的地址。*

用户登出信息结构：

struct CThostFtdcUserLogoutField

{

///经纪公司代码

TThostFtdcBrokerIDType BrokerID;

///用户代码

TThostFtdcUserIDType UserID;

};

4.5.6 OnRspError 方法

针对用户请求的出错通知。

函数原型：

void OnRspError(

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)

参数：

pRspInfo：*返回用户响应信息的地址。*

响应信息结构：

struct CThostFtdcRspInfoField

```
{
    ///错误代码
    TThostFtdcErrorIDType ErrorID;
    ///错误信息
    TThostFtdcErrorMsgType    ErrorMsg;
};
```

4.5.7 OnRspSubMarketData 方法

当客户端发出订阅行情请求之后，Kingstar 返回响应时，该方法会被调用，通知客户端订阅是否成功。

函数原型：

```
void OnRspSubMarketData(
    CThostFtdcSpecificInstrumentField *pSpecificInstrument,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

参数：

pSpecificInstrument： *返回客户端订阅行情的合约结构地址。*

合约结构：

```
struct CThostFtdcSpecificInstrumentField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
};
```

4.5.8 OnRspUnSubMarketData 方法

当客户端发出退订行情请求之后，Kingstar 返回响应时，该方法会被调用，通知客户端退订是否成功。

函数原型：

```
void OnRspUnSubMarketData (
        CThostFtdcSpecificInstrumentField *pSpecificInstrument,
```

*CThostFtdcRspInfoField *pRspInfo,*

int nRequestID,

bool bIsLast)

参数：

pSpecificInstrument：*返回客户端订阅行情的合约结构地址。*

合约结构：

```
struct CThostFtdcSpecificInstrumentField
{
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
};
```

4.5.9 OnRtnDepthMarketData 方法

行情推送。当发生行情订阅成功时，Kingstar 服务器会通知客户端，该方法会被调用。

函数原型：

```
void OnRtnDepthMarketData(CThostFtdcDepthMarketDataField *pDepthMarketData);
```

参数：

pDepthMarketData：*返回客户端订阅的行情结构地址。*

行情结构：

```
struct CThostFtdcDepthMarketDataField
{
    ///交易日
    TThostFtdcDateType      TradingDay;
    ///合约代码
    TThostFtdcInstrumentIDType InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///合约在交易所的代码
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///最新价
    TThostFtdcPriceType     LastPrice;
    ///上次结算价
```

```
TThostFtdcPriceType	PreSettlementPrice;
///昨收盘
TThostFtdcPriceType	PreClosePrice;
///昨持仓量
TThostFtdcLargeVolumeType PreOpenInterest;
///今开盘
TThostFtdcPriceType	OpenPrice;
///最高价
TThostFtdcPriceType	HighestPrice;
///最低价
TThostFtdcPriceType	LowestPrice;
///数量
TThostFtdcVolumeType	Volume;
///成交金额
TThostFtdcMoneyType	Turnover;
///持仓量
TThostFtdcLargeVolumeType OpenInterest;
///今收盘
TThostFtdcPriceType	ClosePrice;
///本次结算价
TThostFtdcPriceType	SettlementPrice;
///涨停板价
TThostFtdcPriceType	UpperLimitPrice;
///跌停板价
TThostFtdcPriceType	LowerLimitPrice;
///昨虚实度
TThostFtdcRatioType	PreDelta;
///今虚实度
TThostFtdcRatioType	CurrDelta;
///最后修改时间
TThostFtdcTimeType	UpdateTime;
///最后修改毫秒
TThostFtdcMillisecType UpdateMillisec;
```

```
///申买价一
TThostFtdcPriceType	BidPrice1;
///申买量一
TThostFtdcVolumeType	BidVolume1;
///申卖价一
TThostFtdcPriceType	AskPrice1;
///申卖量一
TThostFtdcVolumeType	AskVolume1;
///申买价二
TThostFtdcPriceType	BidPrice2;
///申买量二
TThostFtdcVolumeType	BidVolume2;
///申卖价二
TThostFtdcPriceType	AskPrice2;
///申卖量二
TThostFtdcVolumeType	AskVolume2;
///申买价三
TThostFtdcPriceType	BidPrice3;
///申买量三
TThostFtdcVolumeType	BidVolume3;
///申卖价三
TThostFtdcPriceType	AskPrice3;
///申卖量三
TThostFtdcVolumeType	AskVolume3;
///申买价四
TThostFtdcPriceType	BidPrice4;
///申买量四
TThostFtdcVolumeType	BidVolume4;
///申卖价四
TThostFtdcPriceType	AskPrice4;
///申卖量四
TThostFtdcVolumeType	AskVolume4;
///申买价五
```

```
    TThostFtdcPriceType     BidPrice5;
    ///申买量五
    TThostFtdcVolumeType    BidVolume5;
    ///申卖价五
    TThostFtdcPriceType     AskPrice5;
    ///申卖量五
    TThostFtdcVolumeType    AskVolume5;
    ///当日均价
    TThostFtdcPriceType     AveragePrice;
};
```

4.6 CthostFtdcMdApi 接口

CthostFtdcMdApi 接口提供给用户的功能主要体现在行情的订阅、退订等功能。

4.6.1 CreateFtdcMdApi 方法

客户端应用程序使用此函数来创造一个“CthostFtdcMdApi”实例。注意：创造实例时不要使用“new”。

函数原型：

*static CThostFtdcMdApi *CreateFtdcMdApi (const char *pszFlowPath = "");*

参数：

pszFlowPath：*常量字符指针，用于指定一个文件目录来存贮行情系统发布消息的状态。默认值代表当前目录。*

返回值：

返回一个指向 CThostFtdcMdApi 实例的指针。

4.6.2 SetWritablePath 方法

客户端应用程序使用此函数来设置本地文件存放路径

函数原型：

*int SetWritablePath(const char * szpath = "");*

参数：

Szpath：常量字符指针，用于指定一个文件目录来存贮本地文件。默认值代表当前目录。

4.6.3 Release 方法

释放一个 CThostFtdcMdApi 实例。不能使用 delete 方法

函数原型：

void Release()；

4.6.4 Init 方法

使客户端开始与 Kingstar 服务器建立连接，连接成功后可以进行登陆。

函数原型：

void Init()；

4.6.5 Join 方法

客户端等待一个接口实例线程的结束。

函数原型：

void Join()；

4.6.6 GetTradingDay 方法

获得当前交易日。只有当与 Kingstar 服务器连接建立后才会取到正确的值。

函数原型：

*const char *GetTradingDay()；*

返回值：

返回一个指向日期信息字符串的常量指针。

4.6.7 RegisterFront 方法

设置行情系统的网络通讯地址，Kingstar 服务器拥有多个通信地址，但用户只需要选择一个通信地址。客户端 RegisterFront 多次时，使用第一个有效的通信地址。

函数原型：

*void RegisterFront(char *pszFrontAddress);*

参数：

pszFrontAddress *：指向后台服务器地址的指针。服务器地址的格式为："protocol://ipaddress:port"，如："tcp://127.0.0.1:17993"。 "tcp"代表传输协议，"127.0.0.1"代表服务器地址，"17993"代表服务器端口号。同时 API 也提供了对代理服务器的支持，包括 socks4、socks5 及 http，客户端开发时只需通过传递给 API 不同的连接字符串就可实现，例如 "socks5://127.0.0.1:17993 /user:pass@127.0.0.1:18993"* 。

4.6.8 RegisterSpi 方法

注册一个派生自 CThostFtdcMdSpi 接口类的实例，该实例将完成事件处理。

函数原型：

*void RegisterSpi(CthostFtdcMdSpi *pSpi) ;*

参数：

pSpi*：实现了 CThostFtdcMdSpi 接口的实例指针。*

4.6.9 SubscribeMarketData 方法

用户发出订阅行情的请求。

函数原型：

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

参数：

****ppInstrumentID[]****：*指向用户订阅行情的合约结构的地址。*

nCount*：行情订阅的合约个数。*

4.6.10 UnSubscribeMarketData 方法

用户发出退订行情的请求。

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

参数：

****ppInstrumentID[]****：*指向用户退订行情的合约结构的地址*

nCount：*退订行情的合约个数。*

4.6.11 ReqUserLogin 方法

用户发出登陆请求。

函数原型：

```
int ReqUserLogin(
        CThostFtdcReqUserLoginField *pReqUserLoginField,
        int nRequestID)；
```

参数：

pReqUserLoginField：*指向用户登录请求结构的地址。*

用户登录请求结构：

```
struct CThostFtdcReqUserLoginField
{
    ///交易日
    TThostFtdcDateType      TradingDay;
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///用户代码
    TThostFtdcUserIDType  UserID;
    ///密码
    TThostFtdcPasswordType      Password;
    ///用户端产品信息
    TThostFtdcProductInfoType   UserProductInfo;
    ///接口端产品信息
    TThostFtdcProductInfoType   InterfaceProductInfo;
    ///协议信息
    TThostFtdcProtocolInfoType  ProtocolInfo;
    ///Mac 地址
    TThostFtdcMacAddressType  MacAddress;
    ///动态密码
    TThostFtdcPasswordType      OneTimePassword;
    ///终端 IP 地址
```

```
    TThostFtdcIPAddressType     ClientIPAddress;
};
```

4.6.12 ReqUserLogout 方法

用户发出登出请求。

函数原型：

```
int ReqUserLogout(
        CThostFtdcUserLogoutField *pUserLogout,
        int nRequestID)；
```

参数：

pReqUserLogout： *指向用户登出请求结构的地址。*

用户登出请求结构：

```
struct CThostFtdcUserLogoutField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType      BrokerID;
    ///用户代码
    TThostFtdcUserIDType  UserID;
};
```

4.6.13 RegisterNameServer 方法

注册名字服务器（门户）

客户端可通过调用此方法注册名字服务器从而获取最优网关以实现快速登录。Kingstar API 支持多次注册名字服务器，也就是说，此函数可以被调用多次，每次调用都能实现从此名字服务器相关的最优网关列表中返回一个网关。需要注意的是，一旦多次调用此方法，后续返回的网关地址会替代上次返回的网关地址。

目前此接口只支持 API 的 windows 版本。

函数模型：

```
void RegisterNameServer(char *pszNameServerAddress);
```

参数：

pszNameServerAddress： *指向名字服务器地址（也叫门户服务器地址）的指针。名字服务*

器地址格式如下所示：“协议:/ip 地址:端口/代理用户名:代理用户密码@代理服务器 ip 地址:代理服务器端口/网关标识/客户号”。其中代理相关字段是可选的。例如：“tcp://127.0.0.1:11000/A/80001”，“tcp”代表通讯协议，“127.0.0.1”标识名字服务器的 ip 地址，“11000”标识名字服务器的端口，“A”代表网关标识，“80001”标识客户号。

网关标识字段如下所示：

值	描述
‘A’	大智慧
‘B’	文华一键通
‘C’	彭博闪电手
‘D’	托瑞邦泽
‘E’	快期
‘F’	金字塔
‘G’	达钱

4.7 CTKSCosSpi 接口

4.7.1 OnRspInitInsertConditionalOrder 方法

条件单录入响应

函数模型：

```
void OnRspInitInsertConditionalOrder(
        CTKSConditionalOrderOperResultField *pInitInsertConditionalOrder,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)
```

参数：

pInitInsertConditionalOrder：指向条件单响应结构的指针。

条件单响应结构如下所示：

```
struct CTKSConditionalOrderOperResultField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///条件单编号
    TTKSConditionalOrderIDType     ConditionalOrderID;
```

```
    ///本地报单编号
    TThostFtdcOrderLocalIDType      OrderLocalID;
    ///交易所代码
    TThostFtdcExchangeIDType ExchangeID;
    ///合约代码
    TThostFtdcInstrumentIDType      InstrumentID;
    ///报单状态
    TThostFtdcOrderStatusType  OrderStatus;
    ///开平标志
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///投机套保标志
    TThostFtdcHedgeFlagType    CombHedgeFlag;
    ///买卖方向
    TThostFtdcDirectionType     Direction;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量
    TThostFtdcVolumeType VolumeTotalOriginal;
    ///撤销用户
    TThostFtdcUserIDType  UserID;
    ///撤销时间
    TThostFtdcTimeType     CancelTime;
    ///客户代码
    TThostFtdcClientIDType      ClientID;
    ///条件单状态
    TTKSConditionalOrderStatusType ConditionalOrderStatus;
    ///错误信息
    TThostFtdcErrorMsgType     ErrorMsg;
    ///价格类别
    TThostFtdcOrderPriceTypeType OrderPriceType;
    ///触发次数
    TThostFtdcVolumeType TriggeredTimes;
    ///条件单类型
    TTKSConditionalOrderType OrderType;
    ///备注
    TThostFtdcMemoType   Memo;
    ///有效时间
    TThostFtdcTimeType    ActiveTime;
    ///失效时间
    TThostFtdcTimeType InActiveTime;
};
```

4.7.2 OnRspQueryConditionalOrder 方法

条件单查询返回响应

函数模型：

```
void OnRspQueryConditionalOrder(
    CTKSConditionalOrderOperResultField *pQueryConditionalOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

参数：

pQueryConditionalOrder：指向条件单响应结构的指针

条件单响应结构如下所示：

```
struct CTKSConditionalOrderOperResultField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///条件单编号
    TTKSConditionalOrderIDType    ConditionalOrderID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType    OrderLocalID;
    ///交易所代码
    TThostFtdcExchangeIDType ExchangeID;
    ///合约代码
    TThostFtdcInstrumentIDType    InstrumentID;
    ///报单状态
    TThostFtdcOrderStatusType  OrderStatus;
    ///开平标志
    TThostFtdcOffsetFlagType   CombOffsetFlag;
    ///投机套保标志
    TThostFtdcHedgeFlagType   CombHedgeFlag;
    ///买卖方向
    TThostFtdcDirectionType    Direction;
    ///价格
    TThostFtdcPriceType    LimitPrice;
    ///数量
    TThostFtdcVolumeType VolumeTotalOriginal;
```

```
    ///撤销用户
    TThostFtdcUserIDType  UserID;
    ///撤销时间
    TThostFtdcTimeType    CancelTime;
    ///客户代码
    TThostFtdcClientIDType      ClientID;
    ///条件单状态
    TTKSConditionalOrderStatusType ConditionalOrderStatus;
    ///错误信息
    TThostFtdcErrorMsgType      ErrorMsg;
    ///价格类别
    TThostFtdcOrderPriceTypeType OrderPriceType;
    ///触发次数
    TThostFtdcVolumeType TriggeredTimes;
    ///条件单类型
    TTKSConditionalOrderType OrderType;
    ///备注
    TThostFtdcMemoType   Memo;
    ///有效时间
    TThostFtdcTimeType    ActiveTime;
    ///失效时间
    TThostFtdcTimeType InActiveTime;
};
```

4.7.3 OnRspModifyConditionalOrder 方法

条件单修改响应。

函数模型：

```
void OnRspModifyConditionalOrder(
    CTKSConditionalOrderOperResultField *pModifyConditionalOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

参数：

pModifyConditionalOrder：指向条件单响应结构的指针

条件单响应结构如下所示：

```
struct CTKSConditionalOrderOperResultField
{
```

```
///营业部代码
TThostFtdcBrokerIDType BrokerID;
///投资者代码
TThostFtdcInvestorIDType InvestorID;
///条件单编号
TTKSConditionalOrderIDType     ConditionalOrderID;
///本地报单编号
TThostFtdcOrderLocalIDType     OrderLocalID;
///交易所代码
TThostFtdcExchangeIDType ExchangeID;
///合约代码
TThostFtdcInstrumentIDType     InstrumentID;
///报单状态
TThostFtdcOrderStatusType  OrderStatus;
///开平标志
TThostFtdcOffsetFlagType    CombOffsetFlag;
///投机套保标志
TThostFtdcHedgeFlagType    CombHedgeFlag;
///买卖方向
TThostFtdcDirectionType     Direction;
///价格
TThostFtdcPriceType    LimitPrice;
///数量
TThostFtdcVolumeType VolumeTotalOriginal;
///撤销用户
TThostFtdcUserIDType  UserID;
///撤销时间
TThostFtdcTimeType    CancelTime;
///客户代码
TThostFtdcClientIDType     ClientID;
///条件单状态
TTKSConditionalOrderStatusType ConditionalOrderStatus;
///错误信息
TThostFtdcErrorMsgType     ErrorMsg;
///价格类别
TThostFtdcOrderPriceTypeType OrderPriceType;
///触发次数
TThostFtdcVolumeType TriggeredTimes;
///条件单类型
TTKSConditionalOrderType OrderType;
///备注
```

```
    TThostFtdcMemoType   Memo;
    ///有效时间
    TThostFtdcTimeType    ActiveTime;
    ///失效时间
    TThostFtdcTimeType InActiveTime;
};
```

4.7.4 OnRspPauseConditionalOrder 方法

条件单暂停或激活操作响应。

函数模型：

```
void OnRspPauseConditionalOrder(
    CTKSConditionalOrderOperResultField *pPauseConditionalOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

参数：

pPauseConditionalOrder：指向条件单响应结构的指针

条件单响应结构如下所示：

```
struct CTKSConditionalOrderOperResultField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///条件单编号
    TTKSConditionalOrderIDType     ConditionalOrderID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType     OrderLocalID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///合约代码
    TThostFtdcInstrumentIDType     InstrumentID;
    ///报单状态
    TThostFtdcOrderStatusType   OrderStatus;
    ///开平标志
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///投机套保标志
```

```
    TThostFtdcHedgeFlagType    CombHedgeFlag;
    ///买卖方向
    TThostFtdcDirectionType    Direction;
    ///价格
    TThostFtdcPriceType    LimitPrice;
    ///数量
    TThostFtdcVolumeType VolumeTotalOriginal;
    ///撤销用户
    TThostFtdcUserIDType  UserID;
    ///撤销时间
    TThostFtdcTimeType    CancelTime;
    ///客户代码
    TThostFtdcClientIDType    ClientID;
    ///条件单状态
    TTKSConditionalOrderStatusType ConditionalOrderStatus;
    ///错误信息
    TThostFtdcErrorMsgType    ErrorMsg;
    ///价格类别
    TThostFtdcOrderPriceTypeType OrderPriceType;
    ///触发次数
    TThostFtdcVolumeType TriggeredTimes;
    ///条件单类型
    TTKSConditionalOrderType OrderType;
    ///备注
    TThostFtdcMemoType   Memo;
    ///有效时间
    TThostFtdcTimeType    ActiveTime;
    ///失效时间
    TThostFtdcTimeType InActiveTime;
};
```

4.7.5 OnRspRemoveConditionalOrder 方法

条件单删除响应。

函数模型：

```
void OnRspRemoveConditionalOrder(
    CTKSConditionalOrderRspResultField *pRemoveConditionalOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
```

bool bIsLast)

参数：

pRemoveConditionalOrder：指向条件单处理结果信息的指针。

条件单处理结果信息如下所示：

```
struct CTKSConditionalOrderRspResultField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///条件单编号
    TTKSConditionalOrderIDType    ConditionalOrderID;
};
```

4.7.6 OnRspSelectConditionalOrder 方法

条件单选择响应。

函数模型：

```
void OnRspSelectConditionalOrder(
    CTKSConditionalOrderRspResultField *pSelectConditionalOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

参数：

pSelectConditionalOrder：指向条件单处理结果信息的指针。

条件单处理结果信息如下所示：

```
struct CTKSConditionalOrderRspResultField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///条件单编号
    TTKSConditionalOrderIDType    ConditionalOrderID;
};
```

4.7.7 OnRspInsertProfitAndLossOrder 方法

盈损单下单响应。

函数模型：

```
void OnRspInsertProfitAndLossOrder(
    CTKSProfitAndLossOrderOperResultField *pInsertProfitAndLossOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

参数：

pInsertProfitAndLossOrder：指向盈损单处理结构的指针。

盈损单处理结构如下所示：

```
struct CTKSProfitAndLossOrderOperResultField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///止损止盈单号
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///操作员
    TThostFtdcUserIDType  UserID;
    ///投资者名称
    TThostFtdcPartyNameType InvestorName;
    ///本地报单编号
    TThostFtdcOrderLocalIDType      OrderLocalID;
    ///止损价
    TThostFtdcPriceType    StopLossPrice;
    ///止盈价
    TThostFtdcPriceType    TakeProfitPrice;
    ///平仓方式
    TTKSCloseModeType CloseMode;
    ///平仓反向加减价位数（平仓方式=1或时有效）
    TThostFtdcPriceType Figures;
    ///行情触发时的最新价
    TThostFtdcPriceType LastPrice;
    ///生成止损止盈单时间
    TThostFtdcTimeType     ProfitAndLossOrderFormTime;
```

```
    ///生成条件单时间
    TThostFtdcTimeType    ConditionalOrderFormTime;
    ///生成委托单时间
    TThostFtdcTimeType    OrderFormTime;
    ///止损止盈单状态
    TTKSConditionalOrderStatusType ProfitAndLossOrderStatus;
    ///条件单编号
    TTKSConditionalOrderIDType    ConditionalOrderID;
    ///交易所代码
    TThostFtdcExchangeIDType ExchangeID;
    ///客户代码
    TThostFtdcClientIDType      ClientID;
    ///合约代码
    TThostFtdcInstrumentIDType      InstrumentID;
    ///开平标志
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///投机套保标志
    TThostFtdcHedgeFlagType    CombHedgeFlag;
    ///买卖方向
    TThostFtdcDirectionType      Direction;
    ///价格
    TThostFtdcPriceType    LimitPrice;
    ///数量
    TThostFtdcVolumeType VolumeTotalOriginal;
    ///生成盈损价的方式
    TTKSOffsetValueType OffsetValue;
    ///业务单元
    TThostFtdcBusinessUnitTypeBusinessUnit;
    ///条件单触发价格类型
    TTKSSpringTypeType SpringType;
    ///浮动止损价
    TThostFtdcPriceType    FloatLimitPrice;
    ///开仓成交价格
    TThostFtdcPriceType OpenTradePrice;
};
```

4.7.8 OnRspModifyProfitAndLossOrder 方法

盈损单修改响应。

函数模型：

```
void OnRspModifyProfitAndLossOrder(
        CTKSProfitAndLossOrderOperResultField                   *pModifyProfitAndLossOrder,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast)
```

参数：

pModifyProfitAndLossOrder：指向盈损单处理结构的指针。

盈损单处理结构如下所示：

```
struct CTKSProfitAndLossOrderOperResultField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///止损止盈单号
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///操作员
    TThostFtdcUserIDType  UserID;
    ///投资者名称
    TThostFtdcPartyNameType InvestorName;
    ///本地报单编号
    TThostFtdcOrderLocalIDType      OrderLocalID;
    ///止损价
    TThostFtdcPriceType    StopLossPrice;
    ///止盈价
    TThostFtdcPriceType    TakeProfitPrice;
    ///平仓方式
    TTKSCloseModeType CloseMode;
    ///平仓反向加减价位数（平仓方式=1或时有效）
    TThostFtdcPriceType Figures;
    ///行情触发时的最新价
    TThostFtdcPriceType LastPrice;
    ///生成止损止盈单时间
    TThostFtdcTimeType     ProfitAndLossOrderFormTime;
    ///生成条件单时间
    TThostFtdcTimeType     ConditionalOrderFormTime;
    ///生成委托单时间
    TThostFtdcTimeType     OrderFormTime;
    ///止损止盈单状态
```

```
    TTKSConditionalOrderStatusType ProfitAndLossOrderStatus;
    ///条件单编号
    TTKSConditionalOrderIDType     ConditionalOrderID;
    ///交易所代码
    TThostFtdcExchangeIDType ExchangeID;
    ///客户代码
    TThostFtdcClientIDType       ClientID;
    ///合约代码
    TThostFtdcInstrumentIDType       InstrumentID;
    ///开平标志
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///投机套保标志
    TThostFtdcHedgeFlagType    CombHedgeFlag;
    ///买卖方向
    TThostFtdcDirectionType      Direction;
    ///价格
    TThostFtdcPriceType     LimitPrice;
    ///数量
    TThostFtdcVolumeType VolumeTotalOriginal;
    ///生成盈损价的方式
    TTKSOffsetValueType OffsetValue;
    ///业务单元
    TThostFtdcBusinessUnitTypeBusinessUnit;
    ///条件单触发价格类型
    TTKSSpringTypeType SpringType;
    ///浮动止损价
    TThostFtdcPriceType     FloatLimitPrice;
    ///开仓成交价格
    TThostFtdcPriceType OpenTradePrice;
};
```

4.7.9 OnRspRemoveProfitAndLossOrder 方法

盈损单删除响应。

函数模型：

```
void OnRspRemoveProfitAndLossOrder(
    CTKSProfitAndLossOrderRemoveField                    *pRemoveProfitAndLossOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
```

bool bIsLast)

参数：

pRemoveProfitAndLossOrder：指向止损止盈单删除结构的指针。

止损止盈单删除结构如下所示：

```
struct CTKSProfitAndLossOrderRemoveField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///止损止盈单号
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
};
```

4.7.10 OnRspQueryProfitAndLossOrder 方法

止损止盈单查询响应。

函数模型：

```
void OnRspQueryProfitAndLossOrder(
    CTKSProfitAndLossOrderOperResultField *pQueryProfitAndLossOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

参数：

pQueryProfitAndLossOrder：指向盈损单处理结构的指针。

盈损单处理结构如下所示：

```
struct CTKSProfitAndLossOrderOperResultField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///止损止盈单号
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///操作员
```

```
TThostFtdcUserIDType  UserID;
///投资者名称
TThostFtdcPartyNameType InvestorName;
///本地报单编号
TThostFtdcOrderLocalIDType      OrderLocalID;
///止损价
TThostFtdcPriceType    StopLossPrice;
///止盈价
TThostFtdcPriceType    TakeProfitPrice;
///平仓方式
TTKSCloseModeType CloseMode;
///平仓反向加减价位数（平仓方式=1或时有效）
TThostFtdcPriceType Figures;
///行情触发时的最新价
TThostFtdcPriceType LastPrice;
///生成止损止盈单时间
TThostFtdcTimeType     ProfitAndLossOrderFormTime;
///生成条件单时间
TThostFtdcTimeType     ConditionalOrderFormTime;
///生成委托单时间
TThostFtdcTimeType     OrderFormTime;
///止损止盈单状态
TTKSConditionalOrderStatusType ProfitAndLossOrderStatus;
///条件单编号
TTKSConditionalOrderIDType     ConditionalOrderID;
///交易所代码
TThostFtdcExchangeIDType  ExchangeID;
///客户代码
TThostFtdcClientIDType      ClientID;
///合约代码
TThostFtdcInstrumentIDType      InstrumentID;
///开平标志
TThostFtdcOffsetFlagType    CombOffsetFlag;
///投机套保标志
TThostFtdcHedgeFlagType    CombHedgeFlag;
///买卖方向
TThostFtdcDirectionType      Direction;
///价格
TThostFtdcPriceType     LimitPrice;
///数量
TThostFtdcVolumeType VolumeTotalOriginal;
```

```
    ///生成盈损价的方式
    TTKSOffsetValueType OffsetValue;
    ///业务单元
    TThostFtdcBusinessUnitTypeBusinessUnit;
    ///条件单触发价格类型
    TTKSSpringTypeType SpringType;
    ///浮动止损价
    TThostFtdcPriceType     FloatLimitPrice;
    ///开仓成交价格
    TThostFtdcPriceType OpenTradePrice;
};
```

4.7.11 OnRtnCOSAskSelect 方法

条件单请求选择通知。

函数模型：

void OnRtnCOSAskSelect(CTKSCOSAskSelectField *pCOSAskSelect)

参数：

pCOSAskSelect：指向条件单请求选择信息结构的指针。

条件单请求选择信息结构如下：

```
struct CTKSCOSAskSelectField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///操作员
    TThostFtdcUserIDType  UserID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///序号
    TThostFtdcSequenceNoType SequenceNo;
    ///条件单编号
    TTKSConditionalOrderIDType     ConditionalOrderID;
    ///备注
    TThostFtdcMemoType   Memo;
    ///选择方式
    TTKSConditionalOrderSelectTypeType SelectType;
  };
```

4.7.12 OnRtnCOSStatus 方法

条件单状态通知。

函数模型：

void OnRtnCOSStatus(CTKSCOSStatusField *pCOSStatus)

参数：

pCOSStatus：指向条件单状态结构的指针。

条件单状态结构如下所示：

```
struct CTKSCOSStatusField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///操作员
    TThostFtdcUserIDType    UserID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///序号
    TThostFtdcSequenceNoType    SequenceNo;
    ///条件单编号
    TTKSConditionalOrderIDType      ConditionalOrderID;
    ///条件单状态
    TTKSConditionalOrderStatusType ConditionalOrderStatus;
    ///备注
    TThostFtdcMemoType     Memo;
    ///本地报单编号
    TThostFtdcOrderLocalIDType OrderLocalID;
    ///交易所代码
    TThostFtdcExchangeIDType    ExchangeID;
    ///合约代码
    TThostFtdcInstrumentIDType  InstrumentID;
    ///报单状态
    TThostFtdcOrderStatusType    OrderStatus;
    ///开平标志
```

```
TThostFtdcOffsetFlagType      CombOffsetFlag;
///投机套保标志
TThostFtdcHedgeFlagType       CombHedgeFlag;
///买卖方向
TThostFtdcDirectionType Direction;
///价格
TThostFtdcPriceType       LimitPrice;
///数量
TThostFtdcVolumeType    VolumeTotalOriginal;
///交易日
TThostFtdcTradeDateType       TradingDay;
///撤销用户
TThostFtdcUserIDType    CancelUserID;
///撤销时间
TThostFtdcTimeType        CancelTime;
///客户代码
TThostFtdcClientIDType  ClientID;
///业务单元
TThostFtdcBusinessUnitType   BusinessUnit;
///报单编号
TThostFtdcOrderSysIDType    OrderSysID;
///今成交数量
TThostFtdcVolumeType    VolumeTraded;
///剩余数量
TThostFtdcVolumeType    VolumeTotal;
///委托时间
TThostFtdcTimeType        InsertTime;
///激活时间
TThostFtdcTimeType        ActiveTime;
///成交价格
TThostFtdcPriceType       TradePrice;
///货币代码
TThostFtdcCurrencyIDType        CurrencyID;
```

};

4.7.13 OnRtnPLStatus 方法

盈损单状态通知。

函数模型：

void OnRtnPLStatus(CTKSPLStatusField *pPLStatus)

参数：

pPLStatus：指向盈损单状态结构的指针。

盈损单状态结构如下所示：

```
struct CTKSPLStatusField
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///操作员
    TThostFtdcUserIDType  UserID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///序号
    TThostFtdcSequenceNoType SequenceNo;
    ///止损止盈单编号
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///止损条件单编号
    TTKSConditionalOrderIDType     StopLossOrderID;
    ///止盈条件单编号
    TTKSConditionalOrderIDType     TakeProfitOrderID;
    ///盈损单状态
    TTKSConditionalOrderStatusType ProfitAndLossOrderStatus;
    ///止损价
    TThostFtdcPriceType    StopLossPrice;
    ///止盈价
    TThostFtdcPriceType    TakeProfitPrice;
    ///生成盈损价的方式
```

```
TTKSOffsetValueType OffsetValue;
///开仓成交价格
TThostFtdcPriceType OpenTradePrice;
///备注
TThostFtdcMemoType   Memo;
///本地报单编号
TThostFtdcOrderLocalIDType      OrderLocalID;
///交易所代码
TThostFtdcExchangeIDType ExchangeID;
///合约代码
TThostFtdcInstrumentIDType      InstrumentID;
///报单状态
TThostFtdcOrderStatusType  OrderStatus;
///开平标志
TThostFtdcOffsetFlagType    CombOffsetFlag;
///投机套保标志
TThostFtdcHedgeFlagType    CombHedgeFlag;
///买卖方向
TThostFtdcDirectionType      Direction;
///价格
TThostFtdcPriceType     LimitPrice;
///数量
TThostFtdcVolumeType VolumeTotalOriginal;
///交易日
TThostFtdcTradeDateType     TradingDay;
///撤销用户
TThostFtdcUserIDType  CancelUserID;
///撤销时间
TThostFtdcTimeType     CancelTime;
///客户代码
TThostFtdcClientIDType      ClientID;
///业务单元
TThostFtdcBusinessUnitTypeBusinessUnit;
```

```
    ///报单编号
    TThostFtdcOrderSysIDType  OrderSysID;
    ///今成交数量
    TThostFtdcVolumeType VolumeTraded;
    ///剩余数量
    TThostFtdcVolumeType VolumeTotal;
    ///委托时间
    TThostFtdcTimeType     InsertTime;
    ///激活时间
    TThostFtdcTimeType     ActiveTime;
    ///成交价格
    TThostFtdcPriceType     TradePrice;
    ///货币代码
    TThostFtdcCurrencyIDType       CurrencyID;
};
```

4.8 CTKSCosApi 接口

4.8.1 ReqInitInsertConditionalOrder 方法

条件单录入请求

函数模型：

```
Int ReqInitInsertConditionalOrder(
    CTKSConditionalOrderInitInsert *pConditionalOrderInitInsert,
    int nRequestID)
```

参数：

pConditionalOrderInitInsert：指向条件单录入结构的指针

条件单录入结构如下：

```
struct CTKSConditionalOrderInitInsert
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
```

```
TThostFtdcInvestorIDType InvestorID;
///合约代码
TThostFtdcInstrumentIDType       InstrumentID;
///交易所代码
TThostFtdcExchangeIDType ExchangeID;
///客户代码
TThostFtdcClientIDType       ClientID;
///买卖方向
TThostFtdcDirectionType      Direction;
///开平标志
TThostFtdcOffsetFlagType    CombOffsetFlag;
///投机套保标志
TThostFtdcHedgeFlagType    CombHedgeFlag;
///数量
TThostFtdcVolumeType VolumeTotalOriginal;
///价格
TThostFtdcPriceType     LimitPrice;
///价格类别
TTKSOrderPriceTypeType OrderPriceType;
///条件类型
TTKSConditionalTypeType    ConditionalType;
///条件价
TThostFtdcPriceType ConditionalPrice;
///条件单编号
TTKSConditionalOrderIDType     ConditionalOrderID;
///触发次数
TThostFtdcVolumeType TriggeredTimes;
///条件单类型
TTKSConditionalOrderType OrderType;
///有效时间
TThostFtdcTimeType    ActiveTime;
///失效时间
TThostFtdcTimeType InActiveTime;
```

```
    ///货币代码
    TThostFtdcCurrencyIDType    CurrencyID;
};
```

4.8.2 ReqQueryConditionalOrder 方法

条件单查询请求。

函数模型：

```
int ReqQueryConditionalOrder(
    CTKSConditionalOrderQuery *pConditionalOrderQuery,
    int nRequestID)
```

参数：

pConditionalOrderQuery：指向条件单查询结构的指针

条件单查询结构如下：

```
struct CTKSConditionalOrderQuery
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///条件单编号
    TTKSConditionalOrderIDType    ConditionalOrderID;
};
```

4.8.3 ReqModifyConditionalOrder 方法

条件单修改请求。

函数模型：

```
int ReqModifyConditionalOrder(
    CTKSConditionalOrderModify *pConditionalOrderModify,
    int nRequestID)
```

参数：

pConditionalOrderModify：指向修改条件单结构的指针

修改条件单结构如下：

```
struct CTKSConditionalOrderModify
{
///营业部代码
TThostFtdcBrokerIDType BrokerID;
///投资者代码
TThostFtdcInvestorIDType InvestorID;
///合约代码
TThostFtdcInstrumentIDType	InstrumentID;
///交易所代码
TThostFtdcExchangeIDType ExchangeID;
///客户代码
TThostFtdcClientIDType	ClientID;
///买卖方向
TThostFtdcDirectionType	Direction;
///开平标志
TThostFtdcOffsetFlagType	CombOffsetFlag;
///投机套保标志
TThostFtdcHedgeFlagType	CombHedgeFlag;
///数量
TThostFtdcVolumeType VolumeTotalOriginal;
///价格
TThostFtdcPriceType	LimitPrice;
///价格类别
TTKSOrderPriceTypeType OrderPriceType;
///条件类型
TTKSConditionalTypeType	ConditionalType;
///条件价
TThostFtdcPriceType ConditionalPrice;
///条件单编号
TTKSConditionalOrderIDType	ConditionalOrderID;
///触发次数
TThostFtdcVolumeType TriggeredTimes;
```

///条件单类型
TTKSConditionalOrderType OrderType;
///有效时间
TThostFtdcTimeType ActiveTime;
///失效时间
TThostFtdcTimeType InActiveTime;
///货币代码
TThostFtdcCurrencyIDType CurrencyID;
};

4.8.4 ReqRemoveConditionalOrder 方法

条件单删除请求。
函数模型：
int ReqRemoveConditionalOrder(
CTKSConditionalOrderRemove *pConditionalOrderRemove,
int nRequestID)
参数：
pConditionalOrderRemove：指向条件单删除结构的指针
条件单删除结构如下：
struct CTKSConditionalOrderRemove
{
///营业部代码
TThostFtdcBrokerIDType BrokerID;
///投资者代码
TThostFtdcInvestorIDType InvestorID;
///条件单编号
TTKSConditionalOrderIDType ConditionalOrderID;
};

4.8.5 ReqStateAlterConditionalOrder 方法

条件单暂停或激活请求。

函数模型：

```
int   ReqStateAlterConditionalOrder(
      CTKSConditionalOrderStateAlter *pConditionalOrderStateAlter,
      int nRequestID)
```

参数：

pConditionalOrderStateAlter： 指向条件单暂停或激活结构的指针

条件单暂停或激活结构如下：

```
struct CTKSConditionalOrderStateAlter
{
      ///营业部代码
      TThostFtdcBrokerIDType BrokerID;
      ///投资者代码
      TThostFtdcInvestorIDType InvestorID;
      ///条件单编号
      TTKSConditionalOrderIDType     ConditionalOrderID;
      ///暂停或激活操作标志
      TTKSConditionalOrderStateAlterType  ConditionalOrderStateAlter;
};
```

4.8.6 ReqSelectConditionalOrder 方法

选择条件单请求

函数模型：

```
int ReqSelectConditionalOrder(
      CTKSConditionalOrderSelect *pConditionalOrderSelect,
      int nRequestID)
```

参数：

pConditionalOrderSelect： 指向条件单选择结构的指针。

条件单选择结构如下：

```
struct CTKSConditionalOrderSelect
{
      ///营业部代码
      TThostFtdcBrokerIDType BrokerID;
```

```
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///条件单编号
    TTKSConditionalOrderIDType    ConditionalOrderID;
    ///选择结果
    TTKSConditionalOrderSelectResultType SelectResult;
};
```

4.8.7 ReqInsertProfitAndLossOrder 方法

止损止盈单录入请求

函数模型：

```
int ReqInsertProfitAndLossOrder(
    CTKSProfitAndLossOrderInsert *pProfitAndLossOrderInsert,
    int nRequestID)
```

参数：

pProfitAndLossOrderInsert：指向止损止盈单录入结构的指针。

止损止盈单录入结构如下：

```
struct CTKSProfitAndLossOrderInsert
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType    OrderLocalID;
    ///止损价
    TThostFtdcPriceType    StopLossPrice;
    ///止盈价
    TThostFtdcPriceType    TakeProfitPrice;
    ///平仓方式
    TTKSCloseModeType CloseMode;
    //平仓反向加减价位数（平仓方式=1或时有效）
    TThostFtdcPriceType FiguresPrice;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///业务单元
```

```
    TThostFtdcBusinessUnitTypeBusinessUnit;
    ///生成盈损价的方式
    TTKSOffsetValueType OffsetValue;
    ///条件单触发价格类型
    TTKSSpringTypeType SpringType;
    ///浮动止损价
    TThostFtdcPriceType    FloatLimitPrice;
    ///触发次数
    TThostFtdcVolumeType TriggeredTimes;
};
```

4.8.8 ReqModifyProfitAndLossOrder 方法

止损止盈单修改请求。

函数模型：

```
int ReqModifyProfitAndLossOrder(
    CTKSProfitAndLossOrderModify *pProfitAndLossOrderModify,
    int nRequestID)
```

参数：

pProfitAndLossOrderModify：指向止损止盈单修改结构的指针。

止损止盈单修改结构如下：

```
struct CTKSProfitAndLossOrderModify
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///止损止盈单号
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///止损价
    TThostFtdcPriceType    StopLossPrice;
    ///止盈价
    TThostFtdcPriceType    TakeProfitPrice;
    ///平仓方式
    TTKSCloseModeType CloseMode;
```

```
    //平仓反向价位数（平仓方式=1 或时有效）
    TThostFtdcPriceType FiguresPrice;
    ///生成盈损价的方式
    TTKSOffsetValueType OffsetValue;
    ///条件单触发价格类型
    TTKSSpringTypeType SpringType;
    ///浮动止损价
    TThostFtdcPriceType     FloatLimitPrice;
    ///触发次数
    TThostFtdcVolumeType TriggeredTimes;
};
```

4.8.9 ReqRemoveProfitAndLossOrder 方法

止损止盈单删除请求。

函数模型：

```
int ReqRemoveProfitAndLossOrder(
    CTKSProfitAndLossOrderRemove *pProfitAndLossOrderRemove,
    int nRequestID)
```

参数：

pProfitAndLossOrderRemove：指向止损止盈单删除结构的指针

止损止盈单删除结构如下所示：

```
struct CTKSProfitAndLossOrderRemove
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///止损止盈单号
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType      OrderLocalID;
    ///交易所代码
    TThostFtdcExchangeIDType ExchangeID;
    ///业务单元
    TThostFtdcBusinessUnitTypeBusinessUnit;
```

};

4.8.10 ReqQueryProfitAndLossOrder 方法

止损止盈单查询请求。

函数模型：

```
int ReqQueryProfitAndLossOrder(
        CTKSProfitAndLossOrderQuery *pProfitAndLossOrderQuery,
        int nRequestID)
```

参数：

pProfitAndLossOrderQuery： 指向止损止盈单查询结构的指针。

止损止盈单查询结构如下所述：

```
struct CTKSProfitAndLossOrderQuery
{
    ///营业部代码
    TThostFtdcBrokerIDType BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType InvestorID;
    ///止损止盈单号
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType      OrderLocalID;
    ///交易所代码
    TThostFtdcExchangeIDType  ExchangeID;
    ///业务单元
    TThostFtdcBusinessUnitTypeBusinessUnit;
};
```

5. 第五章 示例代码

参见开发包内的 Demo 文件夹。

6. 第六章 问题反馈

若接口在使用过程有任何问题，请将详细信息反馈至电子邮箱：Mingming.shen@sungard.com。
谢谢。