SUNGARD 金仕达

Futures Trading API Programming Manual

Document Description

Document Name	Kingstar Futures Trading API Programming Manual
Version	<V1.5>

Document Revision History

version	date of change	memo
V1.0	<2011-09-15>	1. Be compatible with the CTP api； 2. Using namespace “KingstarAPI”； 3. Open interfaces of “ReqQueryInvestorOpenPosition” and “ReqQueryInvestorOpenCombinePosition”
V1.1	<2012-05-18>	1. Standardization of document name
V1.2	<2012-09-05>	1. Open interfaces of “SetWritablePath” to set the local file save path
V1.3	<2012-11-27>	1. support Gold night market 2. Linux-development based API
V1.4	<2013-01-07>	Open interfaces of “RegisterNameServer”
V1.5	<2013-03-06>	1. support bulk order-cancelling interface 2. support conditional order interface

Content

Chapter1、 Introduction

1.1 Brief

Kingstar,a future trade and broker information management system,contains trade server,risk management server,settlement information management subsystem. The API is used to communicate with the Kingstar trade server. From the API, investor can receive quotation data from SHFE, DCE, CZCE and CFFEX, send trading directive to the four exchanges, receive corresponding response and trade status return. Kingstar API will be compatible with the CTP API。

1.2 Introduction of API files

The API of Kingstar trade server is based on C++ library and carrys out the communication between trade client and Kingstar trade server.Trade clients includes Kingstar standard trade client free used by all investor of Kingstar,and trade tools only used personally (developed by investors or their partners). By using the API, trade client could insert or cancel common order and condition order, contract status fire order, query order or trade record and get the current account and position status.The files of API library are differeced by windows and linux platform

Files of windows-version API :

File Name	File Description
KSTraderApiEx.h	Trading interface c++ head file
KSTradeAPI.h	
KSMdApiEx.h	Quotation interface c++ head file
KSMarketDataAPI.h	
KSUserApiDataTypeEx.h	Defines all data type
KSUserApiStructEx.h	Defines all data structure
KSCosApi.h	Condition order interface c++ head file
KSCosApiDataType.h	Defines data type for condition order interface

KSCosApiStruct.h	Defines data struct for condition order interface
KSTradeAPI.lib、KSTradeAPI.dll	The dynamic link library of trading interface
KSMarketDataAPI.lib、KSMarketDataAPI.dll	The dynamic link library of quotation interface
lkcdll.dll	The dynamic link library of authorization file
ksPortalAPI.dll	The dynamic link library of portal interface
SSPXEncode.dll	The dynamic link library of sspx protocol interface
KSInterB2C.lkc	The authorization file of client api

Files of Linux-version API :

File Name	File Description
KSTraderApiEx.h	Trading interface c++ head file
KSTradeAPI.h	
KSMdApiEx.h	Quotation interface c++ head file
KSMarketDataAPI.h	
KSUserApiDataTypeEx.h	Defines all data type
KSUserApiStructEx.h	Defines all data structure
KSCosApi.h	Condition order interface c++ head file
KSCosApiDataType.h	Defines data type for condition order interface
KSCosApiStruct.h	Defines data struct for condition order interface
libkstradeapi.so	The dynamic link library of trading interface
libksmarketdataapi.so	The dynamic link library of quotation interface
libkslkc64r.so, libkslkc32r.so	The 64 bit and 32 bit license authentication dynamic link library
KSInterB2C.lkc	The authorization file of client api

Note: Users of compilers MS VC 6.0，MS VC.NET 2003,etc, need toturn on the multi-thread option in compile setting,using namespace “KingstarAPI”。

The prefix of condition order head files which is “KSCos” stands for “kingstar condition order system”.

Chapter2、 Archetecture

2.1 Communication Mode

The communication protocol between Kingstar API and Kingstar trade server is futures TradingData Exchange protocol(FTD), an information exchange protocol based on TCP.

In FTD protocol, communication mode includes the following three modes:

1、 Dialog mode, client submits a request to Kingstar, and Kingstar will return corresponding results.

2、 Private mode, Kingstar sends private messages to specific client those messages are all private notify message such as order status or trade confirmation.

3、 Broadcast mode, Kingstar publishs common information to all clients registerd to Kingstar.

Each communication mode is not confined to one network connection. That means, with one network connection, the client can use all the three communication modes, or several different client connection can use the same communication mode. For example, the client can use broadcast mode to receive instrument status change message, and at the same time receive its own private message such as order confirmation message.

2.2 Data Stream

Kingstar support dialog, private and broadcast communication mode.

With dialog communication mode, dialog data stream and query data stream could be transmitted. Dialog and query data stream are both bi-direction data stream, the client application submit request and Kingstar server return response. Kingstar server doesn't maintain the status of dialog and query data stream.when problems occurs, for example reconnect happens, the dialog and query data stream will be reset after the communication rebuilding and data on fly will lost.

With private communication mode, private data stream is transmitted.Private data stream is a unidirectional data stream, using it, the Kingstar server send private message to the corresponding client

application. Private message includes risk notice, order status, order confirmation, trade confirmation.The private data stream is reliable, when the client application lost connection with Kingstar server, at any time in the same trading day, the client application can reconnect the Kingstar server with specified sequence number of its own private data flow and without any risk of lost those private trading data.

With the broadcast communication mode, public data stream is transmitted.It is a unidirectional and reliable data stream just like the private data stream, the only difference between them is the broad cast communication data will broadcast to all connecting client application. Its main useage is pulic instrument status or any public important message.

Chapter3、 Progamming Interface Types

Kingstar trade API provides the two interfaces, CThostFtdcTraderApi and CThostFtdcTraderSpi.The Kingstar quotation API provides CThostFtdcMdApi and CThostFtdcMdSpi. The four interfaces implement FTD protocol; the client could submit requests by invoking functions of the CThostFtdcXXXApi and receive the Kingstar response with reloaded callback functions of their own object inherited from CThostFtdcXXXSpi.。

3.1 Dialog mode programming interface

Communication functions of the interface with dialog mode is usually defined as the following:

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

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

void CThostFtdcMDSpi::OnRspXXX(

*CThostFtdcXXXField *pRspXXX,*

CThostFtdcRspInfoField *pRspInfo,

int nRequestID,

bool bIsLast)

The first parameter of request functions is request content and should not be empty.

The second parameter is the request Id, which should be maintained by client trade application, and within one session the ID is strongly recommended be unique, when the client receive the response from the Kingstar server, the client could relate request and response with same request ID.

When the client receive any response from Kingstar server,the reloaded callback function of CThostFtdcXXXSpi will be invoked,if the response has more than one records,the reloaded callback function would be invoked repeatly until the whole message is received.

The first parameter of response functions is the data of the reponse, which usually includes the original request data.If something wrong happened or Kingstar can not find any record for the request,the parameter will be NULL.The second parameter is a flag used by Kingstar to show whether this response is one successful response.When the callback function is invoked more than one time,except the first time of the callback being invoked,this second parameter may be NULL in the following callback action.The third parameter is request ID which is same as the corresponding request.The last parameter is the end marker of the response, the value "true" manifest the current response is the last one related with the same request.

3.2 Private mode programming interface

The following example shows the usual way of defining the private interface:

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

*void CThostFtdcTraderSpi::OnErrRtnXXX(CThostFtdcXXXField *pXXX,*

*CThostFtdcRspInfoField *pRspInfo)*

There is no function of the quotation API interface to communicate with Kingstar server in private mode. When Kingstar server issue return data with private data stream, the reloaded callback function of the object inherited from CThostFtdcTradeSpi will be

invoked. The first parameter of all callback functions is the return content from Kingstar server, the second parameter of the OnErrRtn CThostFtdcTradeSpi functions is detail error information when something is wrong.

3.3 Boadcast mode programming interface

The client application can use the following two fuctions to communication with Kingstar server with broadcast mode:

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

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

The callback function "OnRtnInstrumentStatus" is used to notify client application the status change of instruments.

The callback function "OnRtnDepthMarketData" is used by Kingstar to public the updated market quotation data from exchanges.

Chapter4、 Kingstar API specification

4.1 Working thread

The Kingstar client process need two kind of thread, one is the application main thread and the othe is trade API working thread, if the client want to receive quotation data, another quotation API working thread is needed. API working thread links trade client and Kingstar server.

The trade and quotation API interface is thread-safe, the client application can use two or more working thread at the same time without need to concern about the thread conflict, the client application should process the callback message as quickly as possible to avoid any unporocessed callback message blocking this working thread. To avoid any blocked communication, the client application should use buffer layer to store all the messages received from Kingstar. The client application can also use such buffer to keep its own data model independence from Kingstar API data model.

4.2 General rules

The client trade application follows two steps to connect and communicate with the Kingstar server: initialization and fuction call.

To use trade API,client trade application should program the following steps::

1. Create a "CThostFtdcTraderApi" instance.
2. Create an event hangdle instance inherited from "CThostFtdcTraderSpi" interface, and registering this instance with the "RegisterSpi" function of the "CThostFtdcTraderApi".
3. Subscribe private stream with the "SubscribePrivateTopic" function of the "CThostFtdcTraderApi".
4. Subscribe public stream with the "SubscribePublicTopic" function of the "CThostFtdcTraderApi".
5. Register the trade front addresses of the Kingstar server with the "RegisterFront" function of the "CThostFtdcTraderApi". The client could call the function several times, in order to establish more reliable communication; this kind of function usage is strongly recommended.
6. Start connection with Kingstar server using the "Init" function of the "CThostFtdcTraderApi".
7. After the Kingstar server confirmed the connection, the callback function "OnFrontConnected" of the "CThostFtdcTraderSpi" interface will be invoked. In the function implementation, the client application can submit the "login" request using the "ReqUserLogin" function of the "CThostFtdcTraderApi".
8. After the Kingstar server confirmed the login, the callback function "OnRspUserLogin" of the "CThostFtdcTraderSpi" interface will be invoked.
9. Now, the communication between the client and Kingstar server is estabilished successfully, and the client trade application can use other Kingstar API to communicate with Kingstar server.

If client trade application want to use quotation API, the client application can use those steps which illustrated previous segments, except subscribing private and public stream.

If client trade application wants to use conditional order API, it should program the following steps:

1. First implement callback interfaces of conditional order system through the “CTKSCosSpi” interfaces which are defined in the head file of condition order: KSCosApi.h

2. Should create a "CThostFtdcTraderApi" instance because the register of condition order needs the existence of Trader API.
3. Define the instance of response-callback class of condition order.
4. Get the pointer which points to request API instance of condition order by calling the “LoadExtApi” function which is declared in the Trade API.
5. Call the request API of conditional order system (Note: Instructions of condition order should be called after the Login instruction of trade API, otherwise the client would get the error message prompting “client has not loginned” .

There are several programming rules:

1. The parameters of all request functions should not be NULL.
2. In case the type of functions' return value is "int", value "0" means functions’ return normally, other values represent error returns.

4.3 CThostFtdcTraderSpi

Kingstar use CThostFtdcTraderSpi as its event interface. Client trade application can inherit the function of CThostFtdcTraderSpi to receive the notification from Kingstar server.

4.3.1 OnFrontConnected

This function is invoked after client finished the connection with Kingstar server, then by inherit this function, the client could use "ReqUserLogin" to send login request.

definition:

void OnFrontConnected();

4.3.2 OnFrontDisconnected

When the connection ended or disconnected, this function is called. If the message is left unprocessed, then the API instance will automatically reconnect with Kingstar server using one of the front addresses from the registed front address list.

definition:

void OnFrontDisconnected (int nReason);

parameters:

nReason: *the reason of disconnecion*

0x1001 network reading failed

0x1002 network writing failed

0x2001 heartbeat receiing timeout

0x2002 heartbeat sending timeout

0x2003 received a error message

4.3.3 OnHeartBeatWarning

This function is used to indicate the long used connection is still available.

definition:

void OnHeartBeatWarning(int nTimeLapse);

parameters:

nTimeLapse: *Length of time elapsed since the last received message.*

4.3.4 OnRspUserLogin

Kingstar server use the callback function "OnRspUserLogin" to notify the client whether the login function "OnRspUserLogin" was accepted by the server.

definition:

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

parameters:

pRspUserLogin: *The pointer of the structure for user's login response. The following is definition of the structure,*

```
struct CThostFtdcRspUserLoginField
{
    ///trading day
```

```
    TThostFtdcDateType  TradingDay;
    ///time of login
    TThostFtdcTimeType  LoginTime;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///trade system name
    TThostFtdcSystemNameType    SystemName;
    ///front id
    TThostFtdcFrontIDType   FrontID;
    ///session id
    TThostFtdcSessionIDType SessionID;
    ///max orderref
    TThostFtdcOrderRefType  MaxOrderRef;
    ///time of SHFE
    TThostFtdcTimeType  SHFETime;
    ///time of DCE
    TThostFtdcTimeType  DCETime;
    ///time of CZCE
    TThostFtdcTimeType  CZCETime;
    ///time of FFEX
    TThostFtdcTimeType  FFEXTime;
};
```

pRspInfo: *Pointer of the structure for system response. The following is definition of the structure,*

```
struct CThostFtdcRspInfoField
{
    ///error id
    TThostFtdcErrorIDType   ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.3.5 OnRspUserLogout

Kingstar server use this callback function to notify the client application whether the function "OnRspUserLogout" was succeeded.

definition:

```
void OnRspUserLogout(
        CThostFtdcUserLogoutField *pUserLogout,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

parameters:

pRspUserLogout: *Pointer of the structure for user's logout response. The following is definition of the structure,*

```
struct CThostFtdcUserLogoutField
{
    ///broker id
    TThostFtdcBrokerIDType BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
};
```

4.3.6 OnRspUserPasswordUpdate

Kingstar server use this callback function to notify the client application whether the function "ReqUserPasswordUpdate" was succeeded.

definition:

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

parameters:

pUserPasswordUpdate: *Pointer of the structure for the response of user's password modification. The following is definition of the structure,*

```
struct CThostFtdcUserPasswordUpdateField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///old password
    TThostFtdcPasswordType  OldPassword;
    ///new password
    TThostFtdcPasswordType  NewPassword;
};
```

4.3.7 OnRspTradingAccountPasswordUpdate

Kingstar server use this callback function to notify the client application whether the function "ReqTradingAccountPasswordUpdate" has been succeeded.

definition:

```
void OnRspTradingAccountPasswordUpdate(
    CThostFtdcTradingAccountPasswordUpdateField *pTradingAccountPasswordUpdate,
  CThostFtdcRspInfoField *pRspInfo,
  int nRequestID,
  bool bIsLast);
```

parameters:

pTradingAccountPasswordUpdate: *Pointer of the structure for the response of trading account password modification. The following is definition of the structure,*

```
struct CThostFtdcTradingAccountPasswordUpdateField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///account id
```

```
    TThostFtdcAccountIDType AccountID;
    ///old password
    TThostFtdcPasswordType  OldPassword;
    ///new password
    TThostFtdcPasswordType  NewPassword;
};
```

4.3.8 OnRspError

Kingstar server uses this callback function to notify something is wrong in the client application's request.

definition:

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

parameters:

pRspInfo: *Pointer of the structure for system response. The following is definition of the structure,*

```
struct CThostFtdcRspInfoField
{
    ///error id
    TThostFtdcErrorIDType   ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.3.9 OnRspOrderInsert

Kingstar server use this callback function to response to the client's "ReqOrderInsert" request.

definition:

```
void OnRspOrderInsert(
        CThostFtdcInputOrderField *pInputOrder,
```

```
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

parameters:

pInputOrder: *Pointer of the structure for the response of order inserting. The following is definition of the structure,*

```
struct CThostFtdcInputOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///price type of condition order
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///order direction
    TThostFtdcDirectionType Direction;
    ///combination order's offset flag
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///combination or hedge flag
    TThostFtdcCombHedgeFlagType CombHedgeFlag;
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///valid date
    TThostFtdcTimeConditionType TimeCondition;
    ///GTD DATE
```

```
    TThostFtdcDateType  GTDDate;
    ///volume type
    TThostFtdcVolumeConditionType  VolumeCondition;
    ///min volume
    TThostFtdcVolumeType  MinVolume;
    ///trigger condition
    TThostFtdcContingentConditionType  ContingentCondition;
    ///stop price
    TThostFtdcPriceType StopPrice;
    ///force close reason
    TThostFtdcForceCloseReasonType  ForceCloseReason;
    ///auto suspend flag
    TThostFtdcBoolType  IsAutoSuspend;
    ///business unit
    TThostFtdcBusinessUnitType  BusinessUnit;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    /// force close flag
    TThostFtdcBoolType  UserForceClose;
};
```

4.3.10 OnRspOrderAction

Kingstar server use this callback function to response to the client's "ReqOrderAction" request.

definition:

```
void OnRspOrderAction(
    CThostFtdcInputOrderActionField  *pInputOrderAction,
    CThostFtdcRspInfoField  *pRspInfo,
    int  nRequestID,
    bool  bIsLast);
```

parameters:

pInputOrderAction: *Pointer of the structure for the response of order action. The*

following is definition of the structure,

```
struct CThostFtdcInputOrderActionField
{
    /// broker id
    TThostFtdcBrokerIDType  BrokerID;
    /// investor id
    TThostFtdcInvestorIDType    InvestorID;
    /// order action reference
    TThostFtdcOrderActionRefType    OrderActionRef;
    /// order reference
    TThostFtdcOrderRefType  OrderRef;
    /// request ID
    TThostFtdcRequestIDType RequestID;
    /// front ID
    TThostFtdcFrontIDType   FrontID;
    /// session ID
    TThostFtdcSessionIDType SessionID;
    /// exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    /// order system ID
    TThostFtdcOrderSysIDType    OrderSysID;
    /// action flag
    TThostFtdcActionFlagType    ActionFlag;
    /// price
    TThostFtdcPriceType LimitPrice;
    /// volume change
    TThostFtdcVolumeType    VolumeChange;
    /// user id
    TThostFtdcUserIDType    UserID;
    /// Instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
};
```

4.3.11 OnRspQueryMaxOrderVolume

Kingstar server use this callback function to response to the client application's "ReqQueryMaxOrderVolume" request.

definition:

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

parameters:

pQueryMaxOrderVolume **:** *Pointer of the structure for the response of ReqQueryMaxOrderVolume.The following is definition of the structure,*

```
struct CThostFtdcQueryMaxOrderVolumeField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor id
    TThostFtdcInvestorIDType    InvestorID;
    /// instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///direction
    TThostFtdcDirectionType Direction;
    /// offset flag
    TThostFtdcOffsetFlagType    OffsetFlag;
    ///hedge flag
    TThostFtdcHedgeFlagType HedgeFlag;
    ///max volume
    TThostFtdcVolumeType    MaxVolume;
};
```

4.3.12 OnRspSettlementInfoConfirm

Kingstar server uses this callback function to response to the client application's "ReqSettlementInfoConfirm" request.

definition:

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

parameters:

pSettlementInfoConfirm: *Pointer of the structure for the response of ReqSettlementInfoConfirm. The following is definition of the structure,*

```
struct CThostFtdcSettlementInfoConfirmField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///confirm date
    TThostFtdcDateType  ConfirmDate;
    ///confirm time
    TThostFtdcTimeType  ConfirmTime;
};
```

4.3.13 *OnRspFromBankToFutureByFuture*

Kingstar server uses this callback function to response to the client application's " *ReqFromBankToFutureByFuture* " request.

definition:

```
void OnRspFromBankToFutureByFuture (
    CThostFtdcTransferBankToFutureRspField *pTransferBankToFutureRsp,
    CThostFtdcRspInfoField *pRspInfo,
```

```
    int nRequestID,
    bool bIsLast);
```

parameters:

pTransferBankToFutureRsp : *Pointer of the structure for the response of ReqFromBankToFutureByFuture.The following is definition of the structure,*

```
struct CThostFtdcTransferBankToFutureRspField
{
    ///response code
    TThostFtdcRetCodeType    RetCode;
    ///response info
    TThostFtdcRetInfoType    RetInfo;
    ///future account
    TThostFtdcAccountIDType FutureAccount;
    ///trade amount
    TThostFtdcMoneyType TradeAmt;
    ///customer fee
    TThostFtdcMoneyType CustFee;
    ///currency code
    TThostFtdcCurrencyCodeType  CurrencyCode;
};
```

4.3.14 OnRspFromFutureToBankByFuture

Kingstar server uses this callback function to response to the client application's " *ReqFromFutureToBankByFuture* " request.

definition:

```
void OnRspFromFutureToBankByFuture (
    CThostFtdcTransferFutureToBankRspField *pTransferFutureToBankRsp,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

parameters:

pTransferFutureToBankRsp : *Pointer of the structure for the response of*

ReqFromFutureToBankByFuture. The following is definition of the structure,

```
struct CThostFtdcTransferFutureToBankRspField
{
    /// response code
    TThostFtdcRetCodeType    RetCode;
    /// response info
    TThostFtdcRetInfoType    RetInfo;
    ///future account
    TThostFtdcAccountIDType FutureAccount;
    ///trade amount
    TThostFtdcMoneyType TradeAmt;
    ///customer fee
    TThostFtdcMoneyType CustFee;
    ///currency code
    TThostFtdcCurrencyCodeType  CurrencyCode;
};
```

4.3.15 OnRspTransferQryBank

Kingstar server uses this callback function to response to the client application's "ReqTransferQryBank" request.

definition:

```
void OnRspTransferQryBank(
    CThostFtdcTransferQryBankRspField *pTransferQryBankRsp,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

parameters:

pTransferQryBankRsp *: Pointer of the structure for the response of ReqTransferQryBank. The following is definition of the structure,*

```
struct CThostFtdcTransferQryBankRspField
{
    ///response code
```

```
    TThostFtdcRetCodeType    RetCode;
    ///response info
    TThostFtdcRetInfoType    RetInfo;
    ///future account
    TThostFtdcAccountIDType FutureAccount;
    ///trade amount
    TThostFtdcMoneyType TradeAmt;
    ///use amount
    TThostFtdcMoneyType UseAmt;
    ///fetch amount
    TThostFtdcMoneyType FetchAmt;
    ///currency code
    TThostFtdcCurrencyCodeType  CurrencyCode;
};
```

4.3.16 OnRspQryTransferSerial

Kingstar server uses this callback function to response to the client application's "ReqQryTransferSerial" request.

definition:

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

parameters:

pTransferSerial *: Pointer of the structure for the response of ReqQryTransferSerial. The following is definition of the structure,*

```
struct CThostFtdcTransferSerialField
{
    ///plate serial
    TThostFtdcPlateSerialType    PlateSerial;
    ///trade date
```

```
TThostFtdcTradeDateType TradeDate;
///trading day
TThostFtdcDateType  TradingDay;
///trade time
TThostFtdcTradeTimeType TradeTime;
///trade code
TThostFtdcTradeCodeType TradeCode;
///session id
TThostFtdcSessionIDType SessionID;
///bank id
TThostFtdcBankIDType    BankID;
///bank branch id
TThostFtdcBankBrchIDType    BankBranchID;
///bank account type
TThostFtdcBankAccTypeType   BankAccType;
///bank account
TThostFtdcBankAccountType   BankAccount;
///bank serial
TThostFtdcBankSerialType    BankSerial;
///broker id
TThostFtdcBrokerIDType  BrokerID;
///broker branch id
TThostFtdcFutureBranchIDType    BrokerBranchID;
///future account type
TThostFtdcFutureAccTypeType FutureAccType;
///account id
TThostFtdcAccountIDType AccountID;
///investor id
TThostFtdcInvestorIDType    InvestorID;
///future serial
TThostFtdcFutureSerialType  FutureSerial;
///identified card type
TThostFtdcIdCardTypeType    IdCardType;
```

```
    ///identified card NO.
    TThostFtdcIdentifiedCardNoType  IdentifiedCardNo;
    ///currency id
    TThostFtdcCurrencyIDType    CurrencyID;
    ///trade amount
    TThostFtdcTradeAmountType    TradeAmount;
    ///customer fee
    TThostFtdcCustFeeType   CustFee;
    ///broker fee
    TThostFtdcFutureFeeType BrokerFee;
    ///availability flag
    TThostFtdcAvailabilityFlagType  AvailabilityFlag;
    ///operator code
    TThostFtdcOperatorCodeType  OperatorCode;
    ///bank new account
    TThostFtdcBankAccountType   BankNewAccount;
    ///error id
    TThostFtdcErrorIDType   ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.3.17 OnRspTransferQryDetail

Kingstar server uses this callback function to response to the client application's "ReqTransferQryDetail" request.

definition:

```
void OnRspTransferQryDetail(
    CThostFtdcTransferQryDetailRspField *pTransferQryDetailRsp,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast);
```

parameters:

pTransferQryDetailRsp : *Pointer of the structure for the response of ReqTransferQryDetail. The following is definition of the structure,*

```
struct CThostFtdcTransferQryDetailRspField
{
    ///trade date
    TThostFtdcDateType  TradeDate;
    ///trade time
    TThostFtdcTradeTimeType TradeTime;
    ///trade code
    TThostFtdcTradeCodeType TradeCode;
    ///future serial
    TThostFtdcTradeSerialNoType FutureSerial;
    ///future id
    TThostFtdcFutureIDType  FutureID;
    ///future account
    TThostFtdcFutureAccountType FutureAccount;
    ///bank serial
    TThostFtdcTradeSerialNoType BankSerial;
    ///bank id
    TThostFtdcBankIDType    BankID;
    ///bank branch id
    TThostFtdcBankBrchIDType    BankBrchID;
    ///bank account
    TThostFtdcBankAccountType   BankAccount;
    ///cert code
    TThostFtdcCertCodeType  CertCode;
    ///currency code
    TThostFtdcCurrencyCodeType  CurrencyCode;
    ///transfer amount
    TThostFtdcMoneyType TxAmount;
    ///transfer valid flag
    TThostFtdcTransferValidFlagType Flag;
};
```

4.3.18 OnRspQryOrder

Kingstar server uses this callback function to response to the client application's "ReqQryOrder" request.

definition:

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

parameters:

pOrder: *Pointer of the structure for the response of ReqQryOrder. The following is definition of the structure,*

```
struct CThostFtdcOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///order price type
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///direction
    TThostFtdcDirectionType Direction;
    ///combination order's offset flag
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///combination or hedge flag
    TThostFtdcCombHedgeFlagType CombHedgeFlag;
```

```
///price
TThostFtdcPriceType LimitPrice;
///volume
TThostFtdcVolumeType    VolumeTotalOriginal;
///valid date type
TThostFtdcTimeConditionType TimeCondition;
///GTD DATE
TThostFtdcDateType  GTDDate;
///volume condition
TThostFtdcVolumeConditionType   VolumeCondition;
///min volume
TThostFtdcVolumeType    MinVolume;
///trigger condition
TThostFtdcContingentConditionType   ContingentCondition;
///stop price
TThostFtdcPriceType StopPrice;
///force close reason
TThostFtdcForceCloseReasonType  ForceCloseReason;
///auto suspend flag
TThostFtdcBoolType  IsAutoSuspend;
///business unit
TThostFtdcBusinessUnitType  BusinessUnit;
///request ID
TThostFtdcRequestIDType RequestID;
///order local ID
TThostFtdcOrderLocalIDType  OrderLocalID;
///exchange ID
TThostFtdcExchangeIDType    ExchangeID;
///participant ID
TThostFtdcParticipantIDType ParticipantID;
///trading code
TThostFtdcClientIDType  ClientID;
///exchange instrument ID
```

```
TThostFtdcExchangeInstIDType    ExchangeInstID;
///trader ID
TThostFtdcTraderIDType  TraderID;
///install ID
TThostFtdcInstallIDType InstallID;
///order submit status
TThostFtdcOrderSubmitStatusType OrderSubmitStatus;
///order notify sequence
TThostFtdcSequenceNoType    NotifySequence;
///trading day
TThostFtdcDateType  TradingDay;
///settlement ID
TThostFtdcSettlementIDType  SettlementID;
///order system ID
TThostFtdcOrderSysIDType    OrderSysID;
///order source
TThostFtdcOrderSourceType   OrderSource;
///order status
TThostFtdcOrderStatusType   OrderStatus;
///order type
TThostFtdcOrderTypeType OrderType;
///volume traded
TThostFtdcVolumeType    VolumeTraded;
///total volume
TThostFtdcVolumeType    VolumeTotal;
///insert date
TThostFtdcDateType  InsertDate;
///insert time
TThostFtdcTimeType  InsertTime;
///active time
TThostFtdcTimeType  ActiveTime;
///suspend time
TThostFtdcTimeType  SuspendTime;
```

```
    ///update time
    TThostFtdcTimeType  UpdateTime;
    ///cancel time
    TThostFtdcTimeType  CancelTime;
    ///active trader ID
    TThostFtdcTraderIDType  ActiveTraderID;
    ///clear participant ID
    TThostFtdcParticipantIDType ClearingPartID;
    ///sequence No.
    TThostFtdcSequenceNoType    SequenceNo;
    ///front ID
    TThostFtdcFrontIDType   FrontID;
    ///session ID
    TThostFtdcSessionIDType SessionID;
    ///user product information
    TThostFtdcProductInfoType   UserProductInfo;
    ///status message
    TThostFtdcErrorMsgType  StatusMsg;
    ///force close flag
    TThostFtdcBoolType  UserForceClose;
    ///user id
    TThostFtdcUserIDType    ActiveUserID;
    ///broker order sequence
    TThostFtdcSequenceNoType    BrokerOrderSeq;
    ///relative order system id
    TThostFtdcOrderSysIDType    RelativeOrderSysID;
};
```

4.3.19 OnRspQryTrade

Kingstar server uses this callback function to response to the client application's "ReqQryTrade"request.

definition:

```
void OnRspQryTrade(
        CThostFtdcTradeField *pTrade,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

parameters:

pTrade: *Pointer of the structure for the response of ReqQryTrade. The following is definition of the structure,*

```
struct CThostFtdcTradeField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///trade ID
    TThostFtdcTradeIDType   TradeID;
    ///direction
    TThostFtdcDirectionType Direction;
    ///order system ID
    TThostFtdcOrderSysIDType    OrderSysID;
    ///participant ID
    TThostFtdcParticipantIDType ParticipantID;
    ///trading code
    TThostFtdcClientIDType  ClientID;
    ///trading role
```

```
TThostFtdcTradingRoleType    TradingRole;
///exchange instrument ID
TThostFtdcExchangeInstIDType    ExchangeInstID;
/// offset flag
TThostFtdcOffsetFlagType    OffsetFlag;
/// hedge flag
TThostFtdcHedgeFlagType HedgeFlag;
///price
TThostFtdcPriceType Price;
///volume
TThostFtdcVolumeType    Volume;
///trade date
TThostFtdcDateType  TradeDate;
///trade time
TThostFtdcTimeType  TradeTime;
///trade type
TThostFtdcTradeTypeType TradeType;
///price source
TThostFtdcPriceSourceType    PriceSource;
///trader ID
TThostFtdcTraderIDType  TraderID;
///order local ID
TThostFtdcOrderLocalIDType  OrderLocalID;
///clear participant ID
TThostFtdcParticipantIDType ClearingPartID;
///business unit
TThostFtdcBusinessUnitType    BusinessUnit;
///sequence No.
TThostFtdcSequenceNoType    SequenceNo;
///trading day
TThostFtdcDateType  TradingDay;
///settlement ID
TThostFtdcSettlementIDType  SettlementID;
```

```
    ///broker order sequence
    TThostFtdcSequenceNoType    BrokerOrderSeq;
    ///trade source
    TThostFtdcTradeSourceType   TradeSource;
};
```

4.3.20 OnRspQryInvestor

Kingstar server uses this callback function to response to the client application's "ReqQryInvestor"request.

definition:

```
void OnRspQry Investor (
      CThostFtdcInvestorField *pInvestor,
      CThostFtdcRspInfoField *pRspInfo,
      int nRequestID,
      bool bIsLast);
```

parameters:

pInvestor: *Pointer of the structure for the response of ReqQryInvestor. The following is definition of the structure,*

```
struct CThostFtdcInvestorField
{
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor group ID
    TThostFtdcInvestorIDType    InvestorGroupID;
    ///investor name
    TThostFtdcPartyNameType InvestorName;
    ///Identified Card Type
    TThostFtdcIdCardTypeType    IdentifiedCardType;
    ///Identified Card No.
    TThostFtdcIdentifiedCardNoType  IdentifiedCardNo;
```

```
    ///is active
    TThostFtdcBoolType  IsActive;
    ///telephone
    TThostFtdcTelephoneType Telephone;
    ///address
    TThostFtdcAddressType    Address;
    ///open date
    TThostFtdcDateType  OpenDate;
    ///mobile
    TThostFtdcMobileType    Mobile;
    ///commissionrate  model  id
    TThostFtdcInvestorIDType    CommModelID;
};
```

4.3.21 OnRspQryInvestorPosition

Kingstar server uses this callback function to response to the client application's "ReqQryInvestorPosition"request.

definition:

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

parameters:

pInvestorPosition *: Pointer of the structure for the response of ReqQryInvestorPosition. The following is definition of the structure,*

```
struct CThostFtdcInvestorPositionField
{
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
```

```
///investor ID
TThostFtdcInvestorIDType    InvestorID;
///position direction
TThostFtdcPosiDirectionType PosiDirection;
///hedge flag
TThostFtdcHedgeFlagType HedgeFlag;
///position date
TThostFtdcPositionDateType  PositionDate;
///position of last trading day
TThostFtdcVolumeType    YdPosition;
///position
TThostFtdcVolumeType    Position;
///long frozen
TThostFtdcVolumeType    LongFrozen;
///short frozen
TThostFtdcVolumeType    ShortFrozen;
///long frozen amount
TThostFtdcMoneyType  LongFrozenAmount;
///short frozen amount
TThostFtdcMoneyType  ShortFrozenAmount;
///open volume
TThostFtdcVolumeType    OpenVolume;
///close volume
TThostFtdcVolumeType    CloseVolume;
///open amount
TThostFtdcMoneyType OpenAmount;
///close amount
TThostFtdcMoneyType CloseAmount;
///position cost
TThostFtdcMoneyType PositionCost;
///previous margin
TThostFtdcMoneyType PreMargin;
///used margin
```

```
TThostFtdcMoneyType UseMargin;
///frozen margin
TThostFtdcMoneyType FrozenMargin;
///frozen cash
TThostFtdcMoneyType FrozenCash;
///frozen commission
TThostFtdcMoneyType FrozenCommission;
///cash in
TThostFtdcMoneyType CashIn;
///commission
TThostFtdcMoneyType Commission;
///close profit
TThostFtdcMoneyType CloseProfit;
///position profit
TThostFtdcMoneyType PositionProfit;
///previous settlement price
TThostFtdcPriceType PreSettlementPrice;
///settlement price
TThostFtdcPriceType SettlementPrice;
///trading day
TThostFtdcDateType  TradingDay;
///settlement ID
TThostFtdcSettlementIDType  SettlementID;
///open cost
TThostFtdcMoneyType OpenCost;
///exchange margin
TThostFtdcMoneyType ExchangeMargin;
///combine position
TThostFtdcVolumeType    CombPosition;
///combine long frozen
TThostFtdcVolumeType     CombLongFrozen;
/// combine short frozen
TThostFtdcVolumeType    CombShortFrozen;
```

```
    ///closeprofit by date
    TThostFtdcMoneyType CloseProfitByDate;
    ///closeprofit by trade
    TThostFtdcMoneyType  CloseProfitByTrade;
    ///today position
    TThostFtdcVolumeType     TodayPosition;
    ///marginrate by money
    TThostFtdcRatioType MarginRateByMoney;
    ///marginrate by volume
    TThostFtdcRatioType MarginRateByVolume;
};
```

4.3.22 OnRspQryTradingAccount

Kingstar server uses this callback function to response to the client application's "ReqQryTradingAccount" request.

definition:

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

parameters:

pTradingAccount : Pointer of the structure for the response of ReqQryTradingAccount. The following is definition of the structure,

```
 struct CThostFtdcTradingAccountField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///account id
    TThostFtdcAccountIDType AccountID;
    ///previous mortgage
    TThostFtdcMoneyType PreMortgage;
```

```
///previous credit
TThostFtdcMoneyType PreCredit;
///previous deposit
TThostFtdcMoneyType PreDeposit;
///previous balance
TThostFtdcMoneyType PreBalance;
///premargin
TThostFtdcMoneyType PreMargin;
///interest base
TThostFtdcMoneyType InterestBase;
///interest
TThostFtdcMoneyType Interest;
///deposit
TThostFtdcMoneyType Deposit;
///withdraw
TThostFtdcMoneyType Withdraw;
///frozen margin
TThostFtdcMoneyType FrozenMargin;
///frozen cash
TThostFtdcMoneyType FrozenCash;
///frozen commission
TThostFtdcMoneyType FrozenCommission;
///current margin
TThostFtdcMoneyType CurrMargin;
///cash in
TThostFtdcMoneyType CashIn;
///commission
TThostFtdcMoneyType Commission;
///close profit
TThostFtdcMoneyType CloseProfit;
///position profit
TThostFtdcMoneyType PositionProfit;
///balance
```

```
    TThostFtdcMoneyType Balance;
    ///available
    TThostFtdcMoneyType Available;
    ///withdraw quota
    TThostFtdcMoneyType WithdrawQuota;
    ///reserve
    TThostFtdcMoneyType Reserve;
    ///trading day
    TThostFtdcDateType  TradingDay;
    ///settlement ID
    TThostFtdcSettlementIDType  SettlementID;
    ///credit
    TThostFtdcMoneyType Credit;
    ///Mortgage
    TThostFtdcMoneyType Mortgage;
    ///excahnge margin
    TThostFtdcMoneyType ExchangeMargin;
    ///delivery margin
    TThostFtdcMoneyType DeliveryMargin;
    ///exchange delivery margin
    TThostFtdcMoneyType ExchangeDeliveryMargin;
};
```

4.3.23 OnRspQryTradingCode

Kingstar server uses this callback function to response to the client application's "ReqQryTradingCode"request.

definition:

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

parameters:

pTradingCode: *Pointer of the structure for the response of ReqQryTradingCode. The following is definition of the structure,*

```
struct CThostFtdcTradingCodeField
{
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///trading code
    TThostFtdcClientIDType  ClientID;
    ///is active
    TThostFtdcBoolType  IsActive;
    ///trading code type
    TThostFtdcClientIDTypeType  ClientIDType;
};
```

4.3.24 OnRspQryExchange

Kingstar server uses this callback function to reponse to the client application's "ReqQryExchange" request.

definition:

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

parameters:

pExchange: *Pointer of the structure for the response of ReqQryExchange. The following is definition of the structure,*

```
struct CThostFtdcExchangeField
```

```
{
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///exchange name
    TThostFtdcExchangeNameType  ExchangeName;
    ///exchange property
    TThostFtdcExchangePropertyType  ExchangeProperty;
};
```

4.3.25 OnRspQryInstrument

Kingstar server uses this callback function to reponse to the client application's "ReqQryInstrument" request.

definition:

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

parameters:

pInstrument: *Pointer of the structure for the response of ReqQryInstrument. The following is definition of the structure,*

```
struct CThostFtdcInstrumentField
{
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///instrument name
    TThostFtdcInstrumentNameType    InstrumentName;
    ///exchange instrument ID
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///product ID
```

```
TThostFtdcInstrumentIDType  ProductID;
///product class
TThostFtdcProductClassType  ProductClass;
///delivery year
TThostFtdcYearType  DeliveryYear;
///delivery month
TThostFtdcMonthType DeliveryMonth;
///max volume for market order
TThostFtdcVolumeType    MaxMarketOrderVolume;
///min volume for market order
TThostFtdcVolumeType    MinMarketOrderVolume;
///max volume for limit order
TThostFtdcVolumeType    MaxLimitOrderVolume;
///min volume for limit order
TThostFtdcVolumeType    MinLimitOrderVolume;
///volume multiple of instrument
TThostFtdcVolumeMultipleType    VolumeMultiple;
///price tick
TThostFtdcPriceType PriceTick;
///create date
TThostFtdcDateType  CreateDate;
///open date
TThostFtdcDateType  OpenDate;
///expire date
TThostFtdcDateType  ExpireDate;
///start delivery date
TThostFtdcDateType  StartDelivDate;
///end delivery date
TThostFtdcDateType  EndDelivDate;
///instrument life phase
TThostFtdcInstLifePhaseType InstLifePhase;
///is trading
TThostFtdcBoolType  IsTrading;
```

```
    ///position type
    TThostFtdcPositionTypeType  PositionType;
    ///position date type
    TThostFtdcPositionDateTypeType  PositionDateType;
    ///long margin ratio
    TThostFtdcRatioType LongMarginRatio;
    ///short margin ratio
    TThostFtdcRatioType ShortMarginRatio;
};
```

4.3.26 OnRspQryDepthMarketData

Kingstar server uses this callback function to reponse the client application's "ReqQryDepthMarketData" request.

definition:

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

parameters:

pDepthMarketData : *Pointer of the structure for the response of ReqQryDepthMarketData. The following is definition of the structure,*

```
struct CThostFtdcDepthMarketDataField
{
    ///trading day
    TThostFtdcDateType  TradingDay;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///exchange instrument ID
    TThostFtdcExchangeInstIDType    ExchangeInstID;
```

```
///last price
TThostFtdcPriceType LastPrice;
///previous settlement price
TThostFtdcPriceType PreSettlementPrice;
///previous close price
TThostFtdcPriceType PreClosePrice;
///previous open volume
TThostFtdcLargeVolumeType   PreOpenInterest;
///open price
TThostFtdcPriceType OpenPrice;
///highest price
TThostFtdcPriceType HighestPrice;
///lowest price
TThostFtdcPriceType LowestPrice;
///trade volume
TThostFtdcVolumeType    Volume;
///turnover
TThostFtdcMoneyType Turnover;
///open interest
TThostFtdcLargeVolumeType   OpenInterest;
///close Price
TThostFtdcPriceType ClosePrice;
///settlement price
TThostFtdcPriceType SettlementPrice;
///upper limit price
TThostFtdcPriceType UpperLimitPrice;
///lower limit price
TThostFtdcPriceType LowerLimitPrice;
///pre-delta
TThostFtdcRatioType PreDelta;
///current delta
TThostFtdcRatioType CurrDelta;
///update time
```

```
TThostFtdcTimeType  UpdateTime;
///Update Millisecond
TThostFtdcMillisecType  UpdateMillisec;
///the first bid price
TThostFtdcPriceType BidPrice1;
///the first bid volume
TThostFtdcVolumeType    BidVolume1;
///the first ask price
TThostFtdcPriceType AskPrice1;
///the first ask volume
TThostFtdcVolumeType    AskVolume1;
///the second bid price
TThostFtdcPriceType BidPrice2;
///the second bid volume
TThostFtdcVolumeType    BidVolume2;
///the second ask price
TThostFtdcPriceType AskPrice2;
///the second ask volume
TThostFtdcVolumeType    AskVolume2;
///the third bid price
TThostFtdcPriceType BidPrice3;
///the third bid volume
TThostFtdcVolumeType    BidVolume3;
///the third ask price
TThostFtdcPriceType AskPrice3;
///the third ask volume
TThostFtdcVolumeType    AskVolume3;
///the fourth bid price
TThostFtdcPriceType BidPrice4;
///the fourth bid volume
TThostFtdcVolumeType    BidVolume4;
///the fourth ask price
TThostFtdcPriceType AskPrice4;
```

```
    ///the fourth ask volume
    TThostFtdcVolumeType    AskVolume4;
    ///the fifth bid price
    TThostFtdcPriceType BidPrice5;
    ///the fifth bid volume
    TThostFtdcVolumeType    BidVolume5;
    ///the fifth ask price
    TThostFtdcPriceType AskPrice5;
    ///the fifth ask volume
    TThostFtdcVolumeType    AskVolume5;
    ///average price
    TThostFtdcPriceType AveragePrice;
};
```

4.3.27 OnRspQryInstrumentMarginRate

Kingstar server uses this callback function to reponse the client application's "ReqQryInstrumentMarginRate" request.

definition:

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

parameters:

pInstrumentMarginRate : *Pointer of the structure for the response of ReqQryInstrumentMarginRate.The following is definition of the structure,*

```
struct CThostFtdcInstrumentMarginRateField
{
    ///instrument id
    TThostFtdcInstrumentIDType  InstrumentID;
    ///investor range
    TThostFtdcInvestorRangeType InvestorRange;
```

```
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor id
    TThostFtdcInvestorIDType    InvestorID;
    ///hedge flag
    TThostFtdcHedgeFlagType HedgeFlag;
    ///long margin ratio by money
    TThostFtdcRatioType LongMarginRatioByMoney;
    ///long margin ratio by volume
    TThostFtdcMoneyType LongMarginRatioByVolume;
    ///short margin ratio by money
    TThostFtdcRatioType ShortMarginRatioByMoney;
    ///short margin ratio by volume
    TThostFtdcMoneyType ShortMarginRatioByVolume;
    ///is relative
    TThostFtdcBoolType  IsRelative;
};
```

4.3.28 OnRspQryInstrumentCommissionRate

Kingstar server uses this callback function to reponse the client application's "ReqQryInstrumentCommissionRate" request.

definition:

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

parameters:

pInstrumentCommissionRate: *Pointer of the structure for the response of ReqQryInstrumentCommissionRate. The following is definition of the structure,*

```
struct CThostFtdcInstrumentCommissionRateField
{
```

```
    ///instrument id
    TThostFtdcInstrumentIDType  InstrumentID;
    ///investor range
    TThostFtdcInvestorRangeType InvestorRange;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor id
    TThostFtdcInvestorIDType    InvestorID;
    ///open ratio by money
    TThostFtdcRatioType OpenRatioByMoney;
    /// open ratio by volume
    TThostFtdcRatioType OpenRatioByVolume;
    ///close ratio by money
    TThostFtdcRatioType CloseRatioByMoney;
    ///close ratio by volume
    TThostFtdcRatioType CloseRatioByVolume;
    ///close today ratio by money
    TThostFtdcRatioType CloseTodayRatioByMoney;
    /// close today ratio by volume
    TThostFtdcRatioType CloseTodayRatioByVolume;
};
```

4.3.29 OnRspQryCFMMCTradingAccountKey

Kingstar server uses this callback function to reponse the client application's "ReqQryCFMMCTradingAccountKey" request.

definition:

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

parameters:

pCFMMCTradingAccountKey : *Pointer of the structure for the response of ReqQryCFMMCTradingAccountKey. The following is definition of the structure,*

```
struct CThostFtdcCFMMCTradingAccountKeyField
{
///broker id
TThostFtdcBrokerIDType  BrokerID;
///Participant id
TThostFtdcParticipantIDType ParticipantID;
///account id
TThostFtdcAccountIDType AccountID;
///key id
TThostFtdcSequenceNoType    KeyID;
///current key
TThostFtdcCFMMCKeyType  CurrentKey;
};
```

4.3.30 OnRspQrySettlementInfo

Kingstar server uses this callback function to response to the client application's "ReqQrySettlementInfo" request.

definition:

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

parameters:

pSettlementInfo : *Pointer of the structure for the response of ReqQrySettlementInfo. The following is definition of the structure,*

```
struct CThostFtdcSettlementInfoField
{
    ///trading day
    TThostFtdcDateType  TradingDay;
```

```
    ///settlement ID
    TThostFtdcSettlementIDType  SettlementID;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///sequence No.
    TThostFtdcSequenceNoType    SequenceNo;
    ///content
    TThostFtdcContentType   Content;
};
```

4.3.31 OnRspQryTransferBank

Kingstar server uses this callback function to response to the client application's "ReqQryTransferBank" request.

definition:

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

parameters:

pTransferBank: *Pointer of the structure for the response of ReqQryTransferBank. The following is definition of the structure,*

```
struct CThostFtdcTransferBankField
{
    ///bank id
    TThostFtdcBankIDType    BankID;
    ///bank branch id
    TThostFtdcBankBrchIDType    BankBrchID;
    ///bank name
    TThostFtdcBankNameType  BankName;
```

```
    ///is active
    TThostFtdcBoolType  IsActive;
};
```

4.3.32 OnRspQryInvestorPositionDetail

Kingstar server uses this callback function to response to the client application's "ReqQryInvestorPositionDetail" request.

definition:

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

parameters:

pInvestorPositionDetail *: Pointer of the structure for the response of ReqQryInvestorPositionDetail. The following is definition of the structure,*

```
struct CThostFtdcInvestorPositionDetailField
{
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///hedge flag
    TThostFtdcHedgeFlagType HedgeFlag;
    ///direction
    TThostFtdcDirectionType Direction;
    ///open date
    TThostFtdcDateType  OpenDate;
    ///trade ID
    TThostFtdcTradeIDType   TradeID;
```

```
///volume
TThostFtdcVolumeType    Volume;
///open price
TThostFtdcPriceType OpenPrice;
///trading day
TThostFtdcDateType  TradingDay;
///settlement ID
TThostFtdcSettlementIDType   SettlementID;
///trade type
TThostFtdcTradeTypeType TradeType;
///combination instrument ID
TThostFtdcInstrumentIDType  CombInstrumentID;
///exchange id
TThostFtdcExchangeIDType    ExchangeID;
///closeprofit by date
TThostFtdcMoneyType CloseProfitByDate;
///closeprofit by trade
TThostFtdcMoneyType CloseProfitByTrade;
///positionprofit by date
TThostFtdcMoneyType PositionProfitByDate;
///positionprofit by trade
TThostFtdcMoneyType PositionProfitByTrade;
///margin
TThostFtdcMoneyType Margin;
///exchange margin
TThostFtdcMoneyType ExchMargin;
///marginrate by money
TThostFtdcRatioType MarginRateByMoney;
///marginrate by volume
TThostFtdcRatioType MarginRateByVolume;
///last settlement price
TThostFtdcPriceType LastSettlementPrice;
///settlement price
```

```
    TThostFtdcPriceType SettlementPrice;
    ///close volume
    TThostFtdcVolumeType    CloseVolume;
    ///close amount
    TThostFtdcMoneyType  CloseAmount;
};
```

4.3.33 OnRspQryNotice

Kingstar server uses this callback function to reponse to the client application's "ReqQryNotice" request.

definition:

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

parameters:

pNotice: *Pointer of the structure for the response of ReqQryNotice. The following is definition of the structure,*

```
struct CThostFtdcNoticeField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///content
    TThostFtdcContentType       Content;
    ///Sequence Label of broker notice
    TThostFtdcSequenceLabelType SequenceLabel;
};
```

4.3.34 OnRtnTrade

Kingstar server uses this callback function to notify the client application when

trade has been finished.

definition:

*void OnRtnTrade(CThostFtdcTradeField *pTrade);*

parameters:

pTrade: *Pointer of the structure for the trade information.The following is definition of the structure,*

```
struct CThostFtdcTradeField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///trade ID
    TThostFtdcTradeIDType   TradeID;
    ///direction
    TThostFtdcDirectionType Direction;
    ///order system ID
    TThostFtdcOrderSysIDType    OrderSysID;
    ///participant ID
    TThostFtdcParticipantIDType ParticipantID;
    ///trading code
    TThostFtdcClientIDType  ClientID;
    ///trading role
    TThostFtdcTradingRoleType   TradingRole;
    ///exchange instrument ID
```

```
TThostFtdcExchangeInstIDType    ExchangeInstID;
///offset flag
TThostFtdcOffsetFlagType    OffsetFlag;
///hedge flag
TThostFtdcHedgeFlagType HedgeFlag;
///price
TThostFtdcPriceType Price;
///volume
TThostFtdcVolumeType    Volume;
///trade date
TThostFtdcDateType  TradeDate;
///trade time
TThostFtdcTimeType  TradeTime;
///trade type
TThostFtdcTradeTypeType TradeType;
///price source
TThostFtdcPriceSourceType   PriceSource;
///trader ID
TThostFtdcTraderIDType  TraderID;
///order local ID
TThostFtdcOrderLocalIDType  OrderLocalID;
///clear participant ID
TThostFtdcParticipantIDType ClearingPartID;
///business unit
TThostFtdcBusinessUnitType    BusinessUnit;
///sequence No.
TThostFtdcSequenceNoType    SequenceNo;
///trading day
TThostFtdcDateType  TradingDay;
///settlement ID
TThostFtdcSettlementIDType  SettlementID;
///broker order sequence
TThostFtdcSequenceNoType    BrokerOrderSeq;
```

```
    ///trade source
    TThostFtdcTradeSourceType    TradeSource;
};
```

4.3.35 OnRtnOrder

Kingstar server uses this callback function to notify the client application about change of order status.

definition:

*void OnRtnOrder(CThostFtdcOrderField *pOrder);*

parameters:

pOrder: *Pointer of the structure for the order information.The following is definition of the structure,*

```
struct CThostFtdcOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///order price type
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///direction
    TThostFtdcDirectionType Direction;
    ///combination order's offset flag
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///combination or hedge flag
    TThostFtdcCombHedgeFlagType CombHedgeFlag;
```

```
///price
TThostFtdcPriceType LimitPrice;
///volume
TThostFtdcVolumeType    VolumeTotalOriginal;
///valid date
TThostFtdcTimeConditionType TimeCondition;
///GTD DATE
TThostFtdcDateType  GTDDate;
///volume condition
TThostFtdcVolumeConditionType       VolumeCondition;
///min volume
TThostFtdcVolumeType        MinVolume;
///trigger condition
TThostFtdcContingentConditionType   ContingentCondition;
///stop price
TThostFtdcPriceType StopPrice;
///force close reason
TThostFtdcForceCloseReasonType  ForceCloseReason;
///auto suspend flag
TThostFtdcBoolType  IsAutoSuspend;
///business unit
TThostFtdcBusinessUnitType      BusinessUnit;
///request ID
TThostFtdcRequestIDType RequestID;
///order local ID
TThostFtdcOrderLocalIDType  OrderLocalID;
///exchange ID
TThostFtdcExchangeIDType    ExchangeID;
///participant ID
TThostFtdcParticipantIDType ParticipantID;
///trading code
TThostFtdcClientIDType  ClientID;
///exchange instrument ID
```

```
TThostFtdcExchangeInstIDType    ExchangeInstID;
///trader ID
TThostFtdcTraderIDType  TraderID;
///install ID
TThostFtdcInstallIDType InstallID;
///order submit status
TThostFtdcOrderSubmitStatusType OrderSubmitStatus;
///notify sequence
TThostFtdcSequenceNoType    NotifySequence;
///trading day
TThostFtdcDateType  TradingDay;
///settlement ID
TThostFtdcSettlementIDType      SettlementID;
///order system ID
TThostFtdcOrderSysIDType    OrderSysID;
///order source
TThostFtdcOrderSourceType   OrderSource;
///order status
TThostFtdcOrderStatusType   OrderStatus;
///order type
TThostFtdcOrderTypeType OrderType;
///volume traded
TThostFtdcVolumeType        VolumeTraded;
///volume total
TThostFtdcVolumeType        VolumeTotal;
///insert date
TThostFtdcDateType  InsertDate;
///insert time
TThostFtdcTimeType  InsertTime;
///active time
TThostFtdcTimeType  ActiveTime;
///suspend time
TThostFtdcTimeType  SuspendTime;
```

```
    ///update time
    TThostFtdcTimeType  UpdateTime;
    ///cancel time
    TThostFtdcTimeType  CancelTime;
    ///active trader ID
    TThostFtdcTraderIDType  ActiveTraderID;
    ///clear participant ID
    TThostFtdcParticipantIDType ClearingPartID;
    ///sequence No.
    TThostFtdcSequenceNoType    SequenceNo;
    ///front ID
    TThostFtdcFrontIDType   FrontID;
    ///session ID
    TThostFtdcSessionIDType SessionID;
    ///user product information
    TThostFtdcProductInfoType   UserProductInfo;
    ///status message
    TThostFtdcErrorMsgType  StatusMsg;
    ///user force close flag
    TThostFtdcBoolType  UserForceClose;
    ///active user id
    TThostFtdcUserIDType        ActiveUserID;
    ///broker order sequence
    TThostFtdcSequenceNoType        BrokerOrderSeq;
    ///relative order system id
    TThostFtdcOrderSysIDType    RelativeOrderSysID;
};
```

4.3.36 OnErrRtnOrderInsert

This callback function is used to notify the client application about the failure of the validation of Kingstar server or exchange.

definition:

void OnErrRtnOrderInsert(

*CThostFtdcInputOrderField *pInputOrder,*

*CThostFtdcRspInfoField *pRspInfo);*

parameters:

pInputOrder: *Pointer of the structure for the order insertion information including the response from server. The following is definition of the structure,*

```
struct CThostFtdcInputOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///order price type
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///direction
    TThostFtdcDirectionType Direction;
    ///combination order's offset flag
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///combination or hedge flag
    TThostFtdcCombHedgeFlagType CombHedgeFlag;
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///valid date
    TThostFtdcTimeConditionType TimeCondition;
    ///GTD DATE
```

```
    TThostFtdcDateType  GTDDate;
    ///volume condition
    TThostFtdcVolumeConditionType  VolumeCondition;
    ///min volume
    TThostFtdcVolumeType  MinVolume;
    ///trigger condition
    TThostFtdcContingentConditionType  ContingentCondition;
    ///stop price
    TThostFtdcPriceType StopPrice;
    ///force close reason
    TThostFtdcForceCloseReasonType  ForceCloseReason;
    ///auto suspend flag
    TThostFtdcBoolType  IsAutoSuspend;
    ///business unit
    TThostFtdcBusinessUnitType  BusinessUnit;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///user force close flag
    TThostFtdcBoolType  UserForceClose;
};
```

4.3.37 OnErrRtnOrderAction

This callback function is used to notify the client application about the failure of the validation of Kingstar server or exchange.

definition:

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

parameters:

pOrderAction: *Pointer of the structure for the order action information including the response from server. The following is definition of the structure,*

```
struct CThostFtdcOrderActionField
```

```
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///order action reference
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///front ID
    TThostFtdcFrontIDType   FrontID;
    ///session ID
    TThostFtdcSessionIDType SessionID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///order system ID
    TThostFtdcOrderSysIDType    OrderSysID;
    ///action flag
    TThostFtdcActionFlagType    ActionFlag;
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume change
    TThostFtdcVolumeType    VolumeChange;
    ///action date
    TThostFtdcDateType  ActionDate;
    ///action time
    TThostFtdcTimeType  ActionTime;
    ///trader ID
    TThostFtdcTraderIDType  TraderID;
    ///install ID
    TThostFtdcInstallIDType InstallID;
```

```
    ///order local ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///action local ID
    TThostFtdcOrderLocalIDType  ActionLocalID;
    ///participant ID
    TThostFtdcParticipantIDType ParticipantID;
    ///trading code
    TThostFtdcClientIDType  ClientID;
    ///business unit
    TThostFtdcBusinessUnitType      BusinessUnit;
    ///order action status
    TThostFtdcOrderActionStatusType OrderActionStatus;
    /// user id
    TThostFtdcUserIDType    UserID;
    ///status message
    TThostFtdcErrorMsgType  StatusMsg;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
};
```

4.3.38 OnRspQrySettlementInfoConfirm

Kingstar server uses this callback function to notify the client application the sucess of "ReqQrySettlementInfoConfirm".

definition:

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

parameters:

pSettlementInfoConfirm **:** *Pointer of the structure for the response of ReqQrySettlementInfoConfirm. The following is definition of the structure,*

```
struct CThostFtdcSettlementInfoConfirmField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///confirm date
    TThostFtdcDateType  ConfirmDate;
    ///confirm time
    TThostFtdcTimeType  ConfirmTime;
};
```

4.3.39 OnRspQryContractBank

Kingstar server uses this callback function to notify the client application the sucess of "ReqQryContractBank".

definition:

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

parameters:

pContractBank : Pointer of the structure for the response of ReqQryContractBank. The following is definition of the structure,

```
struct CThostFtdcContractBankField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///bank id
    TThostFtdcBankIDType    BankID;
    ///bank branch id
    TThostFtdcBankBrchIDType    BankBrchID;
```

```
    ///bank name
    TThostFtdcBankNameType  BankName;
};
```

4.3.40 OnRspQryParkedOrder

Kingstar server uses this callback function to response to parked order query.

definition:

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

parameters:

pParkedOrder: *Pointer of the structure for the response of ReqQryParkedOrder. The following is definition of the structure,*

```
struct CThostFtdcParkedOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///order price type
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///direction
    TThostFtdcDirectionType Direction;
    ///combination order's offset flag
```

```
TThostFtdcCombOffsetFlagType    CombOffsetFlag;
///combination or hedge flag
TThostFtdcCombHedgeFlagType CombHedgeFlag;
///price
TThostFtdcPriceType LimitPrice;
///volume
TThostFtdcVolumeType        VolumeTotalOriginal;
///valid date
TThostFtdcTimeConditionType TimeCondition;
///GTD DATE
TThostFtdcDateType  GTDDate;
///volume condition
TThostFtdcVolumeConditionType       VolumeCondition;
///min volume
TThostFtdcVolumeType        MinVolume;
///trigger condition
TThostFtdcContingentConditionType   ContingentCondition;
///stop price
TThostFtdcPriceType StopPrice;
///force close reason
TThostFtdcForceCloseReasonType  ForceCloseReason;
///auto suspend flag
TThostFtdcBoolType  IsAutoSuspend;
///business unit
TThostFtdcBusinessUnitType      BusinessUnit;
///request ID
TThostFtdcRequestIDType RequestID;
///user force close flag
TThostFtdcBoolType  UserForceClose;
///exchange ID
TThostFtdcExchangeIDType        ExchangeID;
///parked order system ID
TThostFtdcParkedOrderIDType ParkedOrderID;
```

```
    ///user type
    TThostFtdcUserTypeType  UserType;
    ///parked order status
    TThostFtdcParkedOrderStatusType Status;
    ///error id
    TThostFtdcErrorIDType       ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.3.41 OnRspQryParkedOrderAction

Kingstar server use this callback function to response to the query of "RspQryParkedOrderAction".

definition:

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

parameters:

pParkedOrderAction *: Pointer of the structure for the response of ReqQryParkedOrderAction. The following is definition of the structure,*

```
struct CThostFtdcParkedOrderActionField
{
    ///broker ID
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///order action reference
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
```

```
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///front ID
    TThostFtdcFrontIDType       FrontID;
    ///session ID
    TThostFtdcSessionIDType SessionID;
    ///exchange ID
    TThostFtdcExchangeIDType        ExchangeID;
    ///order system ID
    TThostFtdcOrderSysIDType    OrderSysID;
    ///action flag
    TThostFtdcActionFlagType    ActionFlag;
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume change
    TThostFtdcVolumeType        VolumeChange;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///parked order action ID
    TThostFtdcParkedOrderActionIDType       ParkedOrderActionID;
    ///user type
    TThostFtdcUserTypeType  UserType;
    ///parked order action status
    TThostFtdcParkedOrderStatusType Status;
    ///error id
    TThostFtdcErrorIDType       ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.3.42 OnRspQryInvestorPositionCombineDetail

Kingstar server uses this callback function to response to the query of investor combination instrument 's position.

definition:

```
void OnRspQryInvestorPositionCombineDetail(
        CThostFtdcInvestorPositionCombineDetailField
       *pInvestorPositionCombineDetail,
       CThostFtdcRspInfoField *pRspInfo,
       int nRequestID,
       bool bIsLast);
```

parameters:

pInvestorPositionCombineDetail:*Pointer of the structure for the response of ReqQryInvestorPositionCombineDetail. The following is definition of the structure,*

```
struct CThostFtdcInvestorPositionCombineDetailField
{
    ///trading day
    TThostFtdcDateType  TradingDay;
    ///open date
    TThostFtdcDateType  OpenDate;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///settlement ID
    TThostFtdcSettlementIDType  SettlementID;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///combination trade ID
    TThostFtdcTradeIDType   ComTradeID;
    ///trade ID
    TThostFtdcTradeIDType   TradeID;
```

```
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///hedge flag
    TThostFtdcHedgeFlagType HedgeFlag;
    ///direction
    TThostFtdcDirectionType Direction;
    ///total amount
    TThostFtdcVolumeType        TotalAmt;
    ///margin
    TThostFtdcMoneyType     Margin;
    ///excahnge margin
    TThostFtdcMoneyType     ExchMargin;
    ///margin rate by money
    TThostFtdcRatioType MarginRateByMoney;
    ///margin rate by volume
    TThostFtdcRatioType MarginRateByVolume;
    ///leg id
    TThostFtdcLegIDType LegID;
    ///leg multiple
    TThostFtdcLegMultipleType   LegMultiple;
    ///combination instrument ID
    TThostFtdcInstrumentIDType  CombInstrumentID;
};
```

4.3.43 OnRspParkedOrderInsert

Kingstar server use this callback function to notify the client application about the sucess of "ReqParkedOrderInsert".

definition:

```
void OnRspParkedOrderInsert(
    CThostFtdcParkedOrderField *pParkedOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
```

bool bIsLast);

parameters:

pParkedOrder : *Pointer of the structure for the response of ReqParkedOrderInsert.*

The following is definition of the structure,

```
struct CThostFtdcParkedOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///order price type
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///direction
    TThostFtdcDirectionType Direction;
    ///combinationorder's offset flag
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///combination or hedge flag
    TThostFtdcCombHedgeFlagType CombHedgeFlag;
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume
    TThostFtdcVolumeType    Volume  TotalOriginal;
    ///Valid date
    TThostFtdcTimeConditionType TimeCondition;
    ///GTD DATE
    TThostFtdcDateType  GTDDate;
    ///volume condition
```

```
    TThostFtdcVolumeConditionType       VolumeCondition;
    ///min volume
    TThostFtdcVolumeType        MinVolume;
    ///trigger condition
    TThostFtdcContingentConditionType   ContingentCondition;
    ///stop price
    TThostFtdcPriceType StopPrice;
    ///force close reason
    TThostFtdcForceCloseReasonType  ForceCloseReason;
    ///auto suspend flag
    TThostFtdcBoolType  IsAutoSuspend;
    ///business unit
    TThostFtdcBusinessUnitType  BusinessUnit;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///user force close flag
    TThostFtdcBoolType  UserForceClose;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///parked order system ID
    TThostFtdcParkedOrderIDType ParkedOrderID;
    ///user type
    TThostFtdcUserTypeType  UserType;
    ///parked order status
    TThostFtdcParkedOrderStatusType Status;
    ///error id
    TThostFtdcErrorIDType   ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.3.44 OnRspParkedOrderAction

Kingstar server uses this callback function to notify the client application the success of "ReqParkedOrderAction".

definition:

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

parameters:

pParkedOrderAction : Pointer of the structure for the response of ReqParkedOrderAction. The following is definition of the structure,

```
struct CThostFtdcParkedOrderActionField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///order action reference
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///front ID
    TThostFtdcFrontIDType   FrontID;
    ///session ID
    TThostFtdcSessionIDType SessionID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///order system ID
    TThostFtdcOrderSysIDType    OrderSysID;
```

```
    ///action flag
    TThostFtdcActionFlagType    ActionFlag;
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume change
    TThostFtdcVolumeType    VolumeChange;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///parked order action ID
    TThostFtdcParkedOrderActionIDType   ParkedOrderActionID;
    ///user type
    TThostFtdcUserTypeType  UserType;
    ///parked order action status
    TThostFtdcParkedOrderStatusType Status;
    ///error id
    TThostFtdcErrorIDType   ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.3.45 OnRspRemoveParkedOrder

Kingstar server use this callback function to notify the client application whether the success of "ReqRemoveParkedOrder".

definition:

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

parameters:

pRemoveParkedOrder *: Pointer of the structure for the response of ReqRemoveParkedOrder. The following is definition of the structure,*

```
struct CThostFtdcRemoveParkedOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///parked order system ID
    TThostFtdcParkedOrderIDType ParkedOrderID;
};
```

4.3.46 OnRspRemoveParkedOrderAction

Kingstar server use this callback function to notify the client application about the success of "ReqRemoveParkedOrderAction".

definition:

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

parameters:

pRemoveParkedOrderAction *: Pointer of the structure for the response of ReqRemoveParkedOrderAction. The following is definition of the structure,*

```
struct CThostFtdcRemoveParkedOrderActionField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///parked order action trade ID
    TThostFtdcParkedOrderActionIDType   ParkedOrderActionID;
```

```
};
```

4.3.47 OnRspQryInvestorOpenPosition

Kingstar server uses this callback function to response to the client application's "ReqQryInvestorOpenPosition" request.

definition:

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

parameters:

pInvestorPositionDetail *: Pointer of the structure for the response of ReqQryInvestorOpenPosition. The following is definition of the structure,*

```
struct CThostFtdcInvestorPositionDetailField
{
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///hedge flag
    TThostFtdcHedgeFlagType HedgeFlag;
    ///direction
    TThostFtdcDirectionType Direction;
    ///open date
    TThostFtdcDateType  OpenDate;
    ///trade ID
    TThostFtdcTradeIDType   TradeID;
    ///volume
    TThostFtdcVolumeType    Volume;
```

```
///open price
TThostFtdcPriceType OpenPrice;
///trading day
TThostFtdcDateType  TradingDay;
///settlement ID
TThostFtdcSettlementIDType  SettlementID;
///trade type
TThostFtdcTradeTypeType TradeType;
///combination instrument ID
TThostFtdcInstrumentIDType  CombInstrumentID;
///exchange id
TThostFtdcExchangeIDType  ExchangeID;
///closeprofit by date
TThostFtdcMoneyType CloseProfitByDate;
///closeprofit by trade
TThostFtdcMoneyType CloseProfitByTrade;
///positionprofit by date
TThostFtdcMoneyType PositionProfitByDate;
///positionprofit by trade
TThostFtdcMoneyType PositionProfitByTrade;
///margin
TThostFtdcMoneyType Margin;
///exchange margin
TThostFtdcMoneyType ExchMargin;
///marginrate by money
TThostFtdcRatioType MarginRateByMoney;
///marginrate by volume
TThostFtdcRatioType MarginRateByVolume;
///last settlement price
TThostFtdcPriceType LastSettlementPrice;
///settlement price
TThostFtdcPriceType SettlementPrice;
///close volume
```

```
    TThostFtdcVolumeType    CloseVolume;
    ///close amount
    TThostFtdcMoneyType  CloseAmount;
};
```

4.3.48 OnRspQryInvestorOpenCombinePosition

Kingstar server uses this callback function to response to the client application's "ReqQryInvestorOpenCombinePosition" request.

definition:

```
void OnRspQryInvestorPositionCombineDetail(
        CThostFtdcInvestorPositionCombineDetailField
*pInvestorPositionCombineDetail,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

parameters:

pInvestorPositionCombineDetail: *Pointer of the structure for the response of ReqQryInvestorOpenCombinePosition. The following is definition of the structure,*

```
struct CThostFtdcInvestorPositionCombineDetailField
{
    ///trading day
    TThostFtdcDateType  TradingDay;
    ///open date
    TThostFtdcDateType  OpenDate;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///settlement ID
    TThostFtdcSettlementIDType  SettlementID;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
```

```
    ///combination trade ID
    TThostFtdcTradeIDType   ComTradeID;
    ///trade ID
    TThostFtdcTradeIDType   TradeID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///hedge flag
    TThostFtdcHedgeFlagType HedgeFlag;
    ///direction
    TThostFtdcDirectionType Direction;
    ///total amount
    TThostFtdcVolumeType        TotalAmt;
    ///margin
    TThostFtdcMoneyType     Margin;
    ///excahnge margin
    TThostFtdcMoneyType     ExchMargin;
    ///margin rate by money
    TThostFtdcRatioType MarginRateByMoney;
    ///margin rate by volume
    TThostFtdcRatioType MarginRateByVolume;
    ///leg id
    TThostFtdcLegIDType LegID;
    ///leg multiple
    TThostFtdcLegMultipleType       LegMultiple;
    ///combination instrument ID
    TThostFtdcInstrumentIDType      CombInstrumentID;
};
```

4.3.49 OnRspQryBrokerTradingAlgos

Kingstar server uses this callback function to response the trading algorithm of brokerage firms.

definition:

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

parameters:

pBrokerTradingAlgos: *Pointer of the structure for the response of ReqQryBrokerTradingAlgos. The following is definition of the structure:*

```
struct CThostFtdcBrokerTradingAlgosField
{
    /// broker ID
    TThostFtdcBrokerIDType  BrokerID;
    /// exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    /// instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    /// position algo ID
    TThostFtdcHandlePositionAlgoIDType  HandlePositionAlgoID;
    ///find marginrate algo ID
    TThostFtdcFindMarginRateAlgoIDType  FindMarginRateAlgoID;
    ///fund handle algo ID
    TThostFtdcHandleTradingAccountAlgoIDType    HandleTradingAccountAlgoID;
};
```

4.3.50 OnRspBulkCancelOrder

Kingstar server uses this callback function to response the request of bulk orders cancel.

definition:

```
void  OnRspBulkCancelOrder(
    CThostFtdcBulkCancelOrderField *pBulkCancelOrder,
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
```

bool bIsLast)

parameters:

pBulkCancelOrder: *Pointer of the structure for the response of* "ReqBulkCancelOrder".

The following is definition of the structure:

```
struct CThostFtdcBulkCancelOrderField
{
    ///Broker ID
    TThostFtdcBrokerIDType      BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///User ID
    TThostFtdcUserIDType        UserID;
    ///Order Type
    TThostFtdcOrderTypeType     OrderType;
    ///Count of Order
    TThostFtdcVolumeType nCount;
    ///Sets of Order
    CThostOrderKeyField OrderKey[MAX_ORDER_COUNT];
}
```

4.4 CthostFtdcTraderApi

CThostFtdcTraderApi interface's functions include order insertion, order action, order and trade query, and other information query such as client information, investor account, and investor position, instrument information, instrument status, exchange publication, etc..

4.4.1 CreateFtdcTraderApi

The Kingstar client application uses this function to create a CThostFtdcTradeApi instance. Please note that do not use "new" to create any instance.

definition:

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

parameters:

pszFlowPath: Pointer of a constant string, point to one special file directory which used to store notified information sent from Kingstar server, if not specified,the cuurent file directory is the default one.

return value:

A pointer of an instance of CThostFtdcTradeApi.

4.4.2 Release

The Kingstar client application uses this function to delete a CThostFtdcTradeApi instance, but please do not use "delete" to delete any instance.

definition:

void Release();

4.4.3 SetWritablePath

The Kingstar client application uses this function to set the local file save path.

definition:

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

parameters:

szpath: Pointer of a constant string, point to one special file directory which used to store loacl information, if not specified,the cuurent file directory is the default one.

4.4.4 init

The Kingstar client application uses this function to create the connection with Kingstar server, after this user can login in.

definition:

void Init();

4.4.5 join

The Kingstar client application uses this function to waiting the close of a CThostFtdcTradeApi instance.

definition:

void Join();

4.4.6 GetTradingDay

The Kingstar client application uses this function to get the current trading day, the return value will be valid only when the connection between client and Kingstar server is created successfully.

definition:

*const char *GetTradingDay();*

return value:

A pointer of a constant string identifies the current trading date.

4.4.7 RegisterSpi

The Kingstar client application uses this function to register an instance inherited from the CThostFtdcTraderSpi interface.

definition:

*void RegisterSpi(CThostFtdcTraderSpi *pSpi) ;*

parameters:

pSpi: *the pointer of the CThostFtdcTraderSpi instance.*

4.4.8 RegisterFront

The Kingstar client application uses this function to register the front address of the Kingstar server,the function could be invocated more than one times to register more front addresses,and the API would selected one until the connection is created successfully.

definition:

*void RegisterFront(char *pszFrontAddress);*

parameters:

pszFrontAddress: *Pointer of the structure for the front address of the Kingstar server. The address format just like :"protocol://ipaddress:port", for example,"tcp://127.0.0.1:17993","tcp" means the communication protocol , "127.0.0.1" identifies the front address."17993" identifies the server port.*

4.4.9 SubscribePrivateTopic

The Kingstar client application uses this function to subscribe the private topic from Kingstar server. The function must be called before the invocation of "init" function;otherwise the client application wouldn't receive its private stream.

definition:

void SubscribePrivateTopic(TE_RESUME_TYPE nResumeType);

parameters:

nResumeType: *the re-transmit mode of the private stream.*

TERT_RESTART: re-transmit from the begin of the current trading day.

TERT_RESUME: resume transmitting from the last received data.

TERT_QUICK: transmitting the new public stream data from the login time.

4.4.10 SubscribePublicTopic

The Kingstar client application uses this function to subscribe the public topic from Kingstar server. The function must be called before the invocation of "init" function;otherwise the client application wouldn't receive its public stream.

definition:

void SubscribePublicTopic(TE_RESUME_TYPE nResumeType);

parameters:

nResumeType: *the re-transmit mode of the public stream.*

TERT_RESTART: re-transmit from the begin of the current trading day.

TERT_RESUME: resume transmitting from the last received data.

TERT_QUICK: transmitting the new public stream data from the login time.

4.4.11 ReqUserLogin

The Kingstar client application uses this function to send the login in request to the Kingstar server.

definition:

int ReqUserLogin(

*CThostFtdcReqUserLoginField *pReqUserLoginField,*

int nRequestID);

parameters:

pReqUserLoginField: *The pointer of the structure for user's login request. The following is definition of the structure,*

```
struct CThostFtdcReqUserLoginField
{
    ///trading day
    TThostFtdcDateType  TradingDay;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///password
    TThostFtdcPasswordType  Password;
    ///user product information
    TThostFtdcProductInfoType   UserProductInfo;
    ///interface product information
    TThostFtdcProductInfoType   InterfaceProductInfo;
    ///protocol information
    TThostFtdcProtocolInfoType  ProtocolInfo;
    ///Mac address
    TThostFtdcMacAddressType    MacAddress;
    ///one time password
    TThostFtdcPasswordType  OneTimePassword;
    ///client IP address
    TThostFtdcIPAddressType ClientIPAddress;
```

```
};
```

return value:

0, success.

-1, net connection failure.

-2, over the max quantity of unhandled requests.

-3, over the max requests per second.

4.4.12 ReqUserLogout

The Kingstar client application uses this function to send the login out request to the Kingstar server.

definition:

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

parameters:

pReqUserLogout: *Pointer of the structure for user's logout request. The following is definition of the structure,*

```
struct CThostFtdcUserLogoutField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
};
```

4.4.13 ReqUserPasswordUpdate

The Kingstar client application uses this function to send the user password update request to the Kingstar server.

definition:

```
int ReqUserPasswordUpdate(
        CThostFtdcUserPasswordUpdateField
```

*pUserPasswordUpdate,
int nRequestID);

parameters:

pUserPasswordUpdate： *Pointer of the structure for user password updation request. The following is definition of the structure,*

```
struct CThostFtdcUserPasswordUpdateField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///old password
    TThostFtdcPasswordType  OldPassword;
    ///new password
    TThostFtdcPasswordType  NewPassword;
};
```

4.4.14 ReqTradingAccountPasswordUpdate

The Kingstar client application uses this function to send the account password update request to the Kingstar server.

definition:

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

parameters:

pUserPasswordUpdate： *Pointer of the structure for account password updation request. The following is definition of the structure,*

```
struct CThostFtdcTradingAccountPasswordUpdateField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
```

```
    ///account id
    TThostFtdcAccountIDType AccountID;
    ///old password
    TThostFtdcPasswordType  OldPassword;
    ///new password
    TThostFtdcPasswordType  NewPassword;
};
```

4.4.15 ReqOrderInsert

The Kingstar client application uses this function to send the order insertion request to the Kingstar server.

definition:

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

parameters:

pInputOrder: *Pointer of the structure for order insertion request. The following is definition of the structure,*

```
struct CThostFtdcInputOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///order price type
    TThostFtdcOrderPriceTypeType    OrderPriceType;
```

```
    ///direction
    TThostFtdcDirectionType Direction;
    ///combination order's offset flag
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///combination or hedge flag
    TThostFtdcCombHedgeFlagType CombHedgeFlag;
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///valid date
    TThostFtdcTimeConditionType TimeCondition;
    ///GTD DATE
    TThostFtdcDateType  GTDDate;
    ///volume condition
    TThostFtdcVolumeConditionType       VolumeCondition;
    ///min volume
    TThostFtdcVolumeType    MinVolume;
    ///trigger condition
    TThostFtdcContingentConditionType   ContingentCondition;
    ///stop price
    TThostFtdcPriceType StopPrice;
    ///force close reason
    TThostFtdcForceCloseReasonType  ForceCloseReason;
    ///auto suspend flag
    TThostFtdcBoolType  IsAutoSuspend;
    ///business unit
    TThostFtdcBusinessUnitType  BusinessUnit;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///user force close
    TThostFtdcBoolType  UserForceClose;
};
```

OrderRef: *order reference, which should increase monotonically. In the response of eachOnRspUserLogin, the client application could get the MaxOrderRef. Other worth mention, the Kingstar server compares the orderref as string, so staffing all placet of TThostFtdcOrderRefType is needed.*

4.4.16 ReqOrderAction

The Kingstar client application uses this function to send the order cancellation request to the Kingstar server.

definition:

```
int ReqOrderAction(
      CThostFtdcOrderActionField *pOrderAction,
      int nRequestID);
```

parameters:

pOrderAction: *Pointer of the structure for order delettion request. The following is definition of the structure,*

```
struct CThostFtdcOrderActionField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType     InvestorID;
    ///order action reference
    TThostFtdcOrderActionRefType     OrderActionRef;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///front ID
    TThostFtdcFrontIDType       FrontID;
    ///session ID
    TThostFtdcSessionIDType SessionID;
    ///exchange ID
```

```
TThostFtdcExchangeIDType    ExchangeID;
///order system ID
TThostFtdcOrderSysIDType    OrderSysID;
///action flag
TThostFtdcActionFlagType    ActionFlag;
///price
TThostFtdcPriceType LimitPrice;
///volume change
TThostFtdcVolumeType    VolumeChange;
///action date
TThostFtdcDateType  ActionDate;
///action time
TThostFtdcTimeType  ActionTime;
///trader ID
TThostFtdcTraderIDType  TraderID;
///install ID
TThostFtdcInstallIDType InstallID;
///order local ID
TThostFtdcOrderLocalIDType  OrderLocalID;
///action local ID
TThostFtdcOrderLocalIDType  ActionLocalID;
///participant ID
TThostFtdcParticipantIDType ParticipantID;
///trading code
TThostFtdcClientIDType      ClientID;
///business unit
TThostFtdcBusinessUnitType      BusinessUnit;
///order action status
TThostFtdcOrderActionStatusType     OrderActionStatus;
///user id
TThostFtdcUserIDType        UserID;
///status message
TThostFtdcErrorMsgType  StatusMsg;
```

```
    ///instrument id
    TThostFtdcInstrumentIDType      InstrumentID;
};
```

4.4.17 ReqQueryMaxOrderVolume

The Kingstar client application uses this function to send the request of query the max order volume to the Kingstar server.

definition:

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

parameters:

pQueryMaxOrderVolume: *Pointer of the structure for the request of query the max order volume.The following is definition of the structure,*

```
struct CThostFtdcQueryMaxOrderVolumeField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///direction
    TThostFtdcDirectionType Direction;
    ///offset flag
    TThostFtdcOffsetFlagType    OffsetFlag;
    ///hedge flag
    TThostFtdcHedgeFlagType HedgeFlag;
    ///max volume
    TThostFtdcVolumeType    MaxVolume;
};
```

4.4.18 ReqSettlementInfoConfirm

The Kingstar client application uses this function to confirm the settlement information from the Kingstar server.

definition:

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

parameters:

pSettlementInfoConfirm: *Pointer of the structure for settlement information confirmation request.The following is definition of the structure,*

```
struct CThostFtdcSettlementInfoConfirmField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///confirm date
    TThostFtdcDateType  ConfirmDate;
    ///confirm time
    TThostFtdcTimeType  ConfirmTime;
};
```

4.4.19 ReqFromBankToFutureByFuture

The Kingstar client application uses this function to transfer bank to future request to the Kingstar server.

definition:

```
int ReqFromBankToFutureByFuture(
    CThostFtdcTransferBankToFutureReqField *pTransferBankToFutureReq,
    int nRequestID);
```

parameters:

pTransferBankToFutureReq: *Pointer of the structure for transfer bank to future*

request. The following is definition of the structure,

```
struct CThostFtdcTransferBankToFutureReqField
{
    ///future account
    TThostFtdcAccountIDType FutureAccount;
    ///future password flag
    TThostFtdcFuturePwdFlagType FuturePwdFlag;
    ///future account password
    TThostFtdcFutureAccPwdType  FutureAccPwd;
    ///trade amount
    TThostFtdcMoneyType TradeAmt;
    ///customer fee
    TThostFtdcMoneyType CustFee;
    ///currency code
    TThostFtdcCurrencyCodeType  CurrencyCode;
};
```

4.4.20 ReqFromFutureToBankByFuture

The Kingstar client application uses this function to transfer future to bank request to the Kingstar server.

definition:

```
int ReqFromFutureToBankByFuture (
    CThostFtdcTransferFutureToBankReqField *pTransferFutureToBankReq,
    int nRequestID);
```

parameters:

pTransferFutureToBankReq: *Pointer of the structure for transfer future to bank request. The following is definition of the structure,*

```
struct CThostFtdcTransferFutureToBankReqField
{
    ///future account
    TThostFtdcAccountIDType FutureAccount;
    ///future password flag
```

```
    TThostFtdcFuturePwdFlagType FuturePwdFlag;
    ///future account password
    TThostFtdcFutureAccPwdType  FutureAccPwd;
    ///trade amount
    TThostFtdcMoneyType TradeAmt;
    ///customer fee
    TThostFtdcMoneyType CustFee;
    ///currency code
    TThostFtdcCurrencyCodeType  CurrencyCode;
};
```

4.4.21 ReqTransferQryBank

The Kingstar client application uses this function to send the transfer bank account query request to the Kingstar server.

definition:

```
int ReqTransferQryBank(
    CThostFtdcTransferQryBankReqField *pTransferQryBankReq,
    int nRequestID);
```

parameters:

pTransferQryBankReq: *Pointer of the structure for transfer bank account query request. The following is definition of the structure,*

```
struct CThostFtdcTransferQryBankReqField
{
    ///future account
    TThostFtdcAccountIDType FutureAccount;
    ///future password flag
    TThostFtdcFuturePwdFlagType FuturePwdFlag;
    ///future account password
    TThostFtdcFutureAccPwdType  FutureAccPwd;
    ///currency code
    TThostFtdcCurrencyCodeType  CurrencyCode;
};
```

4.4.22 ReqQryTransferSerial

The Kingstar client application uses this function to send the transfer serial query request to the Kingstar server.

definition:

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

parameters:

pQryTransferSerial: *Pointer of the structure for transfer serial query request.*

The following is definition of the structure,

```
struct CThostFtdcQryTransferSerialField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///account id
    TThostFtdcAccountIDType AccountID;
    ///bank id
    TThostFtdcBankIDType    BankID;
};
```

4.4.23 ReqTransferQryDetail

The Kingstar client application uses this function to send the transfer detail query request to the Kingstar server.

definition:

```
int ReqTransferQryDetail(
    CThostFtdcTransferQryDetailReqField *pTransferQryDetailReq,
    int nRequestID);
```

parameters:

pTransferQryDetailReq: *Pointer of the structure for transfer detail query request.*

The following is definition of the structure,

```
struct CThostFtdcTransferQryDetailReqField
```

```
{
    ///future account
    TThostFtdcAccountIDType    FutureAccount;
};
```

4.4.24 ReqQryOrder

The Kingstar client application uses this function to send the order query request to the Kingstar server.

definition:

```
int ReqQryOrder(
      CThostFtdcQryOrderField *pQryOrder,
      int nRequestID);
```

parameters:

pQryOrder: *Pointer of the structure for order query request. The following is definition of the structure,*

```
struct CThostFtdcQryOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///order system ID
    TThostFtdcOrderSysIDType    OrderSysID;
    ///insert time start
    TThostFtdcTimeType  InsertTimeStart;
    ///insert time end
    TThostFtdcTimeType  InsertTimeEnd;
};
```

4.4.25 ReqQryTrade

The Kingstar client application uses this function to send the trade query request to the Kingstar server.

definition:

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

parameters:

pQryTrade: *Pointer of the structure for trade query request. The following is definition of the structure,*

```
struct CThostFtdcQryTradeField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///trade ID
    TThostFtdcTradeIDType   TradeID;
    ///trade time start
    TThostFtdcTimeType  TradeTimeStart;
    ///trade time end
    TThostFtdcTimeType  TradeTimeEnd;
};
```

4.4.26 ReqQryInvestor

The Kingstar client application uses this function to send the investor query request to the Kingstar server.

definition:

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

parameters:

pQryInvestor: *Pointer of the structure for investor query request. The following is definition of the structure,*

```
struct CThostFtdcQryInvestorField
{
   ///broker id
   TThostFtdcBrokerIDType  BrokerID;
   ///investor ID
   TThostFtdcInvestorIDType     InvestorID;
};
```

4.4.27 ReqQryInvestorPosition

The Kingstar client application uses this function to send the investor position query request to the Kingstar server.

definition:

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

parameters:

pQryInvestorPosition : *Pointer of the structure for investor position queryrequest. The following is definition of the structure,*

```
struct CThostFtdcQryInvestorPositionField
{
   ///broker id
   TThostFtdcBrokerIDType  BrokerID;
   ///investor ID
   TThostFtdcInvestorIDType     InvestorID;
   ///instrument id
```

```
    TThostFtdcInstrumentIDType  InstrumentID;
};
```

4.4.28 ReqQryTradingAccount

The Kingstar client application uses this function to send the trading account query request to the Kingstar server

definition:

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

parameters:

pQryTradingAccount: *Pointer of the structure for trading account query request. The following is definition of the structure,*

```
struct CThostFtdcQryTradingAccountField
{
   ///broker id
   TThostFtdcBrokerIDType  BrokerID;
   ///investor ID
   TThostFtdcInvestorIDType    InvestorID;
};
```

4.4.29 ReqQryTradingCode

The Kingstar client application uses this function to send the trading code query request to the Kingstar server.

definition:

```
int ReqQryTradingCode(
      CThostFtdcQryTradingCodeField *pQryTradingCode,
      int nRequestID);
```

parameters:

pQryTradingCode: *Pointer of the structure for trading code query request. The following is definition of the structure,*

```
struct CThostFtdcQryTradingCodeField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///trading code
    TThostFtdcClientIDType  ClientID;
    ///trading code type
    TThostFtdcClientIDTypeType  ClientIDType;
};
```

4.4.30 ReqQryExchange

The Kingstar client application uses this function to send the exchange query request to the Kingstar server.

definition:

```
int ReqQryExchange(
      CThostFtdcQryExchangeField *pQryExchange,
      int nRequestID);
```

parameters:

pQryExchange: *Pointer of the structure for exchange query request. The following is definition of the structure,*

```
struct CThostFtdcQryExchangeField
{
    ///exchange id
    TThostFtdcExchangeIDType        ExchangeID;
};
```

4.4.31 ReqQryInstrument

The Kingstar client application uses this function to send the instrument query request to the Kingstar server.

definition:

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

parameters:

pQryInstrument: *Pointer of the structure for instrument query request. The following is definition of the structure,*

```
struct CThostFtdcQryInstrumentField
{
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///exchange instrument ID
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///product ID
    TThostFtdcInstrumentIDType  ProductID;
};
```

4.4.32 ReqQryDepthMarketData

The Kingstar client application uses this function to send the market quotation query request to the Kingstar server.

definition:

```
int ReqQryDepthMarketData(
        CThostFtdcQryDepthMarketDataField *pQryDepthMarketData,
        int nRequestID);
```

parameters:

pQryDepthMarketData: *Pointer of the structure for market quotation query request.*

The following is definition of the structure,

```
struct CThostFtdcQryDepthMarketDataField
{
    ///instrument id
    TThostFtdcInstrumentIDType  InstrumentID;
};
```

4.4.33 ReqQryInstrumentMarginRate

The Kingstar client application uses this function to send the instrument marginrate query request to the Kingstar server.

definition:

```
int ReqQryInstrumentMarginRate(
        CThostFtdcQryInstrumentMarginRateField  *pQryInstrumentMarginRate,
         int nRequestID);
```

parameters:

pQryInstrumentMarginRate: *Pointer of the structure for instrument marginrate query request. The following is definition of the structure,*

```
struct CThostFtdcQryInstrumentMarginRateField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///hedge flag
    TThostFtdcHedgeFlagType HedgeFlag;
};
```

4.4.34 ReqQryInstrumentCommissionRate

The Kingstar client application uses this function to send the instrument

commissionrate query request to the Kingstar server.

definition:

```
int ReqQryInstrumentCommissionRate(
    CThostFtdcQryInstrumentCommissionRateField
   *pQryInstrumentCommissionRate,
    int nRequestID);
```

parameters:

pQryInstrumentCommissionRate : *Pointer of the structure for instrument commissionrate query request. The following is definition of the structure,*

```
struct CThostFtdcQryInstrumentCommissionRateField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType      InstrumentID;
};
```

4.4.35 ReqQryCFMMCTradingAccountKey

The Kingstar client application uses this function to send the CFMMC trading account key query request to the Kingstar server.

definition:

```
int ReqQryCFMMCTradingAccountKey(
    CThostFtdcQryCFMMCTradingAccountKeyField  *pQryCFMMCTradingAccountKey,
    int nRequestID);
```

parameters:

pQryCFMMCTradingAccountKey: *Pointer of the structure for the CFMMC trading account key query request. The following is definition of the structure,*

```
struct CThostFtdcQryCFMMCTradingAccountKeyField
{
    ///broker id
```

```
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
};
```

4.4.36 ReqQrySettlementInfo

The Kingstar client application uses this function to send the settlement information query request to the Kingstar server.

definition:

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

parameters:

pQrySettlementInfo: *Pointer of the structure for settlement information query request. The following is definition of the structure,*

```
struct CThostFtdcQrySettlementInfoField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///trading day
    TThostFtdcDateType  TradingDay;
};
```

4.4.37 ReqQryTransferBank

The Kingstar client application uses this function to send the transfer bank query request to the Kingstar server.

definition:

```
int ReqQryTransferBank(
    CThostFtdcQryTransferBankField *pQryTransferBank,
```

int nRequestID);

parameters:

pQryTransferBank: *Pointer of the structure for transfer bank query request. The following is definition of the structure,*

```
struct CThostFtdcQryTransferBankField
{
    ///bank id
    TThostFtdcBankIDType    BankID;
    ///bank branch id
    TThostFtdcBankBrchIDType    BankBrchID;
};
```

4.4.38 ReqQryInvestorPositionDetail

The Kingstar client application uses this function to send the investor position detail query request to the Kingstar server.

definition:

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

parameters:

pQryInvestorPositionDetail: *Pointer of the structure for investor position detail query request. The following is definition of the structure,*

```
struct CThostFtdcQryInvestorPositionDetailField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor id
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument id
    TThostFtdcInstrumentIDType  InstrumentID;
};
```

4.4.39 ReqQryNotice

The Kingstar client application uses this function to send the notice query request to the Kingstar server.

definition:

```
int ReqQryNotice(
      CThostFtdcQryNoticeField *pQryNotice,
      int nRequestID);
```

parameters:

pQryNotice: *Pointer of the structure fornotice query request. The following is definition of the structure,*

```
struct CThostFtdcQryNoticeField
{
   ///broker id
   TThostFtdcBrokerIDType  BrokerID;
};
```

4.4.40 ReqQrySettlementInfoConfirm

The Kingstar client application uses this function to send the settlement information confirmation query request to the Kingstar server.

definition:

```
int ReqQrySettlementInfoConfirm(
      CThostFtdcQrySettlementInfoConfirmField
      *pQrySettlementInfoConfirm,
       int nRequestID);
```

parameters:

pQrySettlementInfoConfirm : *Pointer of the structure for settlement information confirmation query request. The following is definition of the structure,*

```
struct CThostFtdcQrySettlementInfoConfirmField
{
   ///broker id
   TThostFtdcBrokerIDType  BrokerID;
```

```
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
};
```

4.4.41 ReqQryContractBank

The Kingstar client application uses this function to send the contract bank query request to the Kingstar server.

definition:

```
int ReqQryContractBank(
    CThostFtdcQryContractBankField *pQryContractBank,
    int nRequestID);
```

parameters:

pQryContractBank: *Pointer of the structure for market contract bank request. The following is definition of the structure,*

```
struct CThostFtdcQryContractBankField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///bank id
    TThostFtdcBankIDType    BankID;
    ///bank branch id
    TThostFtdcBankBrchIDType    BankBrchID;
};
```

4.4.42 ReqQryParkedOrder

The Kingstar client application uses this function to send the parked order query request to the Kingstar server.

definition:

```
int ReqQryParkedOrder(
    CThostFtdcQryParkedOrderField *pQryParkedOrder,
    int nRequestID);
```

parameters:

pQryParkedOrder: *Pointer of the structure for parked order query request. The following is definition of the structure,*

```
struct CThostFtdcQryParkedOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
};
```

4.4.43 ReqQryParkedOrderAction

The Kingstar client application uses this function to send the parked order action query request to the Kingstar server.

definition:

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

parameters:

pQryParkedOrderAction: *Pointer of the structure for parked order action query request. The following is definition of the structure,*

```
struct CThostFtdcQryParkedOrderActionField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument id
```

```
    TThostFtdcInstrumentIDType  InstrumentID;
    ///exchange id
    TThostFtdcExchangeIDType        ExchangeID;
};
```

4.4.44 ReqQryInvestorPositionCombineDetail

The Kingstar client application uses this function to send the investor combination position detail query request to the Kingstar server.

definition:

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

parameters:

pQryInvestorPositionCombineDetail: *Pointer of the structure for investor combination position detail query request.The following is definition of the structure,*

```
struct CThostFtdcQryInvestorPositionCombineDetailField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///combination instrument ID
    TThostFtdcInstrumentIDType  CombInstrumentID;
};
```

4.4.45 ReqParkedOrderInsert

The Kingstar client application uses this function to send the parked order insertion request to the Kingstar server.

definition:

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

parameters:

pParkedOrder: *Pointer of the structure for parked order insertion request. The following is definition of the structure,*

```
struct CThostFtdcParkedOrderField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///order price type
    TThostFtdcOrderPriceTypeType    OrderPriceType;
    ///direction
    TThostFtdcDirectionType Direction;
    ///combination offset flag
    TThostFtdcCombOffsetFlagType    CombOffsetFlag;
    ///combination hedge flag
    TThostFtdcCombHedgeFlagType CombHedgeFlag;
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume
    TThostFtdcVolumeType        VolumeTotalOriginal;
    ///valid date
    TThostFtdcTimeConditionType TimeCondition;
    ///GTD DATE
```

```
    TThostFtdcDateType  GTDDate;
    ///volume condition
    TThostFtdcVolumeConditionType       VolumeCondition;
    ///min volume
    TThostFtdcVolumeType    MinVolume;
    ///trigger condition
    TThostFtdcContingentConditionType   ContingentCondition;
    ///stop price
    TThostFtdcPriceType StopPrice;
    ///force close reason
    TThostFtdcForceCloseReasonType  ForceCloseReason;
    ///is auto suspend
    TThostFtdcBoolType  IsAutoSuspend;
    ///business unit
    TThostFtdcBusinessUnitType      BusinessUnit;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///user force close flag
    TThostFtdcBoolType  UserForceClose;
    ///exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///parked order system ID
    TThostFtdcParkedOrderIDType ParkedOrderID;
    ///user type
    TThostFtdcUserTypeType  UserType;
    ///parked order status
    TThostFtdcParkedOrderStatusType Status;
    ///error id
    TThostFtdcErrorIDType   ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.4.46 ReqParkedOrderAction

The Kingstar client application uses this function to send the parked order action request to the Kingstar server.

definition:

int ReqParkedOrderAction(

*CThostFtdcParkedOrderActionField *pParkedOrderAction,*

int nRequestID);

parameters:

pParkedOrderAction: *Pointer of the structure for parked order action request. The following is definition of the structure,*

```
struct CThostFtdcParkedOrderActionField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///order action reference
    TThostFtdcOrderActionRefType    OrderActionRef;
    ///order reference
    TThostFtdcOrderRefType  OrderRef;
    ///request ID
    TThostFtdcRequestIDType RequestID;
    ///front ID
    TThostFtdcFrontIDType   FrontID;
    ///session ID
    TThostFtdcSessionIDType SessionID;
    ///exchange ID
    TThostFtdcExchangeIDType        ExchangeID;
    ///order system ID
    TThostFtdcOrderSysIDType    OrderSysID;
    ///action flag
    TThostFtdcActionFlagType    ActionFlag;
```

```
    ///price
    TThostFtdcPriceType LimitPrice;
    ///volume change
    TThostFtdcVolumeType        VolumeChange;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///parked order action ID
    TThostFtdcParkedOrderActionIDType   ParkedOrderActionID;
    ///user type
    TThostFtdcUserTypeType  UserType;
    ///parked order action status
    TThostFtdcParkedOrderStatusType Status;
    ///error id
    TThostFtdcErrorIDType   ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.4.47 ReqRemoveParkedOrder

The Kingstar client application uses this function to send the parked ordercancel request to the Kingstar server.

definition:

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

parameters:

pRemoveParkedOrder: *Pointer of the structure for parked order removing request. The following is definition of the structure,*

```
struct CThostFtdcRemoveParkedOrderField
{
```

```
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///parked order system ID
    TThostFtdcParkedOrderIDType ParkedOrderID;
};
```

4.4.48 ReqRemoveParkedOrderAction

The Kingstar client application uses this function to send the parked order actioncancel request to the Kingstar server.

definition:

```
int ReqRemoveParkedOrderAction(
    CThostFtdcRemoveParkedOrderActionField *pRemoveParkedOrderAction,
    int nRequestID);
```

parameters:

pRemoveParkedOrderAction: *Pointer of the structure for parked order removing request. The following is definition of the structure,*

```
struct CThostFtdcRemoveParkedOrderActionField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///parked order action trade ID
    TThostFtdcParkedOrderActionIDType   ParkedOrderActionID;
};
```

4.4.49 ReqQueryInvestorOpenPosition

The Kingstar client application uses this function to send the investor open position detail query request to the Kingstar server.

definition:

int ReqQueryInvestorOpenPosition (

*CThostFtdcQryInvestorPositionDetailField *pQryInvestorPositionDetail,*

int nRequestID);

parameters:

pQryInvestorPositionDetail: *Pointer of the structure for investor open position detail query request. The following is definition of the structure,*

```
 struct CThostFtdcQryInvestorPositionDetailField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///investor id
    TThostFtdcInvestorIDType    InvestorID;
    ///instrument id
    TThostFtdcInstrumentIDType  InstrumentID;
};
```

4.4.50 ReqQueryInvestorOpenCombinePosition

Pointer of the structure for investor open combination position detail query request. The following is definition of the structure,

definition:

int ReqQueryInvestorOpenCombinePosition (

CThostFtdcQryInvestorPositionCombineDetailField

**pQryInvestorPositionCombineDetail,*

int nRequestID);

parameters:

pQryInvestorPositionCombineDetail: *Pointer of the structure for investor open combination position detail query request. The following is definition of the structure,*

```
struct CThostFtdcQryInvestorPositionCombineDetailField
{
    ///broker id
```

```
    TThostFtdcBrokerIDType  BrokerID;
    ///investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///combination instrument ID
    TThostFtdcInstrumentIDType      CombInstrumentID;
};
```

4.4.51 ReqQryBrokerTradingAlgos

Query the trading algorithm of brokerage firms.

definition:

```
int  ReqQryBrokerTradingAlgos(
        CThostFtdcQryBrokerTradingAlgosField * pQryBrokerTradingAlgos,
        int  nRequestID);
```

parameters:

pQryBrokerTradingAlgos: *Pointer of the structure for* trading algorithm of brokerage firms query request. *The following is definition of the structure*

```
struct CThostFtdcQryBrokerTradingAlgosField
{
    /// broker id
    TThostFtdcBrokerIDType  BrokerID;
    /// Exchange ID
    TThostFtdcExchangeIDType        ExchangeID;
    /// investor ID
    TThostFtdcInstrumentIDType      InstrumentID;
};
```

4.4.52 RegisterNameServer

The Kingstar client application uses this function to register nameserver which can acquire the optimal gateway for fast login. Kingstar API support registerring several nameservers, that is, this function can be called many times with different

nameservers and every time of the register will get the first gateway address from gateway lists. One thing need to pay attention to is that once register several times, the responded gateway for final registering will replace the last gateway returned.

This interface only support windows version of Kingstar API at present.

definition:

*void RegisterNameServer(char *pszNameServerAddress);*

parameters:

pszNameServerAddress: *Pointer of the structure for the name server address (also be called portal server address) of the Kingstar servers. The name server address format is just like :" protocol://ipaddress:port/proxyuser:proxypass@proxyipaddress:proxyport/gatewayflag/clientid", proxy related fields are optional. For example,"tcp://127.0.0.1:11000/A/80001","tcp" means the communication protocol, "127.0.0.1" identifies the name server address."11000" identifies the server port."A" identifies the gatewayflag, "80001" identifies the ID of the client.*

Gatewayflag field is as follows:

A	Da zhi hui
B	Wen hua yi jian tong
C	Peng bo shan dian shou
D	Tuo rui bang ze
E	Kuai qi
F	Jin zi ta
G	Da qian

4.4.53 ReqBulkCancelOrder

Kingstar server uses this function to request to cancel bulk orders

definition:

int ReqBulkCancelOrder (

*CThostFtdcBulkCancelOrderField *pBulkCancelOrder,*

int nRequestID)

parameters:

pBulkCancelOrder: *Pointer of the structure for Bulk orders.*

The following is definition of the structure

```
struct CThostFtdcBulkCancelOrderField
{
    ///Broker ID
    TThostFtdcBrokerIDType      BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType    InvestorID;
    ///User ID
    TThostFtdcUserIDType        UserID;
    ///Order Type
    TThostFtdcOrderTypeType     OrderType;
    ///Count of Order
    TThostFtdcVolumeType nCount;
    ///Sets of Order
    CThostOrderKeyField OrderKey[MAX_ORDER_COUNT];
}
```

4.4.54 LoadExtApi

Register conditional order instance

definition:

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

parameters:

Spi:Pointer of instance of spi

ExtApiName: name of api

4.5 CthostFtdcMdSpi

Kingstar use CThostFtdcMdSpi as its event interface. Client quotation application can

inherit the function of CThostFtdcMdSpi to receive the notification from Kingstar server.

4.5.1 OnFrontConnected

This function is invoked after client finished the connection with Kingstar server, then by inherit this function, the client could use "ReqUserLogin" to send login request.

definition:

void OnFrontConnected();

4.5.2 OnFrontDisconnected

When the connection ended or disconnected, this function is called. If the message is left unprocessed, then the API instance will automatically reconnect with Kingstar server using one of the front addresses from the registed front address list.

definition:

void OnFrontDisconnected (int nReason);

parameters:

nReason: *the reason of disconnecion*

0x1001 network reading failed

0x1002 network writing failed

0x2001 heartbeat receiing timeout

0x2002 heartbeat sending timeout

0x2003 received a error message

4.5.3 OnHeartBeatWarning

This function is used to indicate the long used connection is still available.

definition:

void OnHeartBeatWarning(int nTimeLapse);

parameters:

nTimeLapse: *Length of time elapsed since the last received message.*

4.5.4 OnRspUserLogin

Kingstar server use the callback function "OnRspUserLogin" to notify the client whether the login function "OnRspUserLogin" was accepted by the server.

definition:

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

parameters:

pRspUserLogin: *The pointer of the structure for user's login response. The following is definition of the structure,*

```
struct CThostFtdcRspUserLoginField
{
    ///trading day
    TThostFtdcDateType  TradingDay;
    ///time of login
    TThostFtdcTimeType  LoginTime;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///trade system name
    TThostFtdcSystemNameType    SystemName;
    ///front id
    TThostFtdcFrontIDType   FrontID;
    ///session id
    TThostFtdcSessionIDType SessionID;
    ///max orderref
    TThostFtdcOrderRefType  MaxOrderRef;
    ///time of SHFE
    TThostFtdcTimeType  SHFETime;
```

```
    ///time of DCE
    TThostFtdcTimeType  DCETime;
    ///time of CZCE
    TThostFtdcTimeType  CZCETime;
    ///time of FFEX
    TThostFtdcTimeType  FFEXTime;
};
```

pRspInfo: *Pointer of the structure for system response. The following is definition of the structure,*

```
struct CThostFtdcRspInfoField
{
    ///error id
    TThostFtdcErrorIDType    ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.5.5 OnRspUserLogout

Kingstar server use this callback function to notify the client application whether the function "OnRspUserLogout" was succeeded.

definition:

```
void OnRspUserLogout(
        CThostFtdcUserLogoutField *pUserLogout,
        CThostFtdcRspInfoField *pRspInfo,
        int nRequestID,
        bool bIsLast);
```

parameters:

pRspUserLogout: *Pointer of the structure for user's logout response. The following is definition of the structure,*

```
struct CThostFtdcUserLogoutField
{
    ///broker id
```

```
    TThostFtdcBrokerIDType BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
};
```

4.5.6 OnRspError

Kingstar server uses this callback function to notify something is wrong in the client application's request.

definition:

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

parameters:

pRspInfo: *Pointer of the structure for system response. The following is definition of the structure,*

```
struct CThostFtdcRspInfoField
{
    ///error id
    TThostFtdcErrorIDType   ErrorID;
    ///error information
    TThostFtdcErrorMsgType  ErrorMsg;
};
```

4.5.7 OnRspSubMarketData

Kingstar server uses this callback function to reponse the client application's "SubscribeMarketData" request.

definition:

```
void OnRspSubMarketData(
    CThostFtdcSpecificInstrumentField *pSpecificInstrument,
    CThostFtdcRspInfoField *pRspInfo,
```

int nRequestID,

bool bIsLast)

parameters:

pSpecificInstrument : *Pointer of the structure for the response of SubscribeMarketData. The following is definition of the structure,*

```
struct CThostFtdcSpecificInstrumentField
{
  ///instrument id
  TThostFtdcInstrumentIDType  InstrumentID;
};
```

4.5.8 OnRspUnSubMarketData

Kingstar server uses this callback function to reponse the client application's "UnSubscribeMarketData" request.

definition:

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

parameters:

pSpecificInstrument : *Pointer of the structure for the response of UnSubscribeMarketData. The following is definition of the structure,*

```
struct CThostFtdcSpecificInstrumentField
{
  ///instrument id
  TThostFtdcInstrumentIDType  InstrumentID;
};
```

4.5.9 OnRtnDepthMarketData

Kingstar server uses this callback function to notify the client application about

change of subscribed contracts.

definition:

*void OnRtnDepthMarketData(CThostFtdcDepthMarketDataField *pDepthMarketData);*

parameters:

pDepthMarketData: *Pointer of the structure for the subscribed contracts information.The following is definition of the structure,*

```
struct CThostFtdcDepthMarketDataField
{
    ///trading day
    TThostFtdcDateType  TradingDay;
    ///instrument id
    TThostFtdcInstrumentIDType  InstrumentID;
    ///exchange id
    TThostFtdcExchangeIDType        ExchangeID;
    ///exchange instrument id
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///last price
    TThostFtdcPriceType LastPrice;
    ///presettlement price
    TThostFtdcPriceType PreSettlementPrice;
    ///preclose price
    TThostFtdcPriceType PreClosePrice;
    ///preopen interest
    TThostFtdcLargeVolumeType   PreOpenInterest;
    ///open price
    TThostFtdcPriceType OpenPrice;
    ///highest price
    TThostFtdcPriceType HighestPrice;
    ///lowest price
    TThostFtdcPriceType LowestPrice;
    ///volume
    TThostFtdcVolumeType    Volume;
    ///turn over
```

```
TThostFtdcMoneyType Turnover;
///open interest
TThostFtdcLargeVolumeType   OpenInterest;
///close price
TThostFtdcPriceType ClosePrice;
///settlement price
TThostFtdcPriceType SettlementPrice;
///upper limit price
TThostFtdcPriceType UpperLimitPrice;
///lower limit price
TThostFtdcPriceType LowerLimitPrice;
///predelta
TThostFtdcRatioType PreDelta;
///currdelta
TThostFtdcRatioType CurrDelta;
///update time
TThostFtdcTimeType  UpdateTime;
///update millisec
TThostFtdcMillisecType    UpdateMillisec;
///bid price 1
TThostFtdcPriceType BidPrice1;
///bid volume 1
TThostFtdcVolumeType     BidVolume1;
///ask price 1
TThostFtdcPriceType AskPrice1;
///ask volume 1
TThostFtdcVolumeType     AskVolume1;
///bid price 2
TThostFtdcPriceType BidPrice2;
///bid volume 2
TThostFtdcVolumeType     BidVolume2;
///ask price 2
TThostFtdcPriceType AskPrice2;
```

```
    ///ask volume 2
    TThostFtdcVolumeType    AskVolume2;
    ///bid price 3
    TThostFtdcPriceType BidPrice3;
    ///bid volume 3
    TThostFtdcVolumeType    BidVolume3;
    ///ask price 3
    TThostFtdcPriceType AskPrice3;
    ///ask volume 3
    TThostFtdcVolumeType    AskVolume3;
    ///bid price 4
    TThostFtdcPriceType BidPrice4;
    ///bid volume 4
    TThostFtdcVolumeType    BidVolume4;
    ///ask price 4
    TThostFtdcPriceType AskPrice4;
    ///ask volume 4
    TThostFtdcVolumeType    AskVolume4;
    ///bid price 5
    TThostFtdcPriceType BidPrice5;
    ///bid volume 5
    TThostFtdcVolumeType    BidVolume5;
    ///ask price 5
    TThostFtdcPriceType AskPrice5;
    ///ask volume 5
    TThostFtdcVolumeType    AskVolume5;
    ///average price
    TThostFtdcPriceType AveragePrice;
};
```

4.6 CthostFtdcMdApi

CthostFtdcMdApi interface's functions mainly include subscribe marketdata,

unsubscribe marketdata, etc..

4.6.1 CreateFtdcMdApi

The Kingstar client application uses this function to create a CthostFtdcMdApi instance. Please note that do not use "new" to create any instance.

definition:

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

parameters:

pszFlowPath: *Pointer of a constant string, point to one special file directory which used to store notified information sent from Kingstar server, if not specified,the cuurent file directory is the default one.*

return value:

A pointer of an instance of CThostFtdcMdApi.

4.6.2 Release

The Kingstar client application uses this function to delete a CThostFtdcMdApi instance, but please do not use "delete" to delete any instance.

definition:

void Release();

4.6.3 SetWritablePath

The Kingstar client application uses this function to set the local file save path.

definition:

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

parameters:

szpath: Pointer of a constant string, point to one special file directory which used to store loacl information, if not specified,the cuurent file directory is the default one.

4.6.4 Init

The Kingstar client application uses this function to create the connection with Kingstar server, after this user can login in.

definition:

void Init();

4.6.5 Join

The Kingstar client application uses this function to waiting the close of a CThostFtdcMdApi instance.

definition:

void Join();

4.6.6 GetTradingDay

The Kingstar client application uses this function to get the current trading day, the return value will be valid only when the connection between client and Kingstar server is created successfully.

definition:

*const char *GetTradingDay();*

return value:

A pointer of a constant string identifies the current trading date.

4.6.7 RegisterFront

The Kingstar client application uses this function to register the front address of the Kingstar server,the function could be invocated more than one times to register more front addresses,and the API would selected one until the connection is created successfully.

definition:

*void RegisterFront(char *pszFrontAddress);*

parameters:

pszFrontAddress: *Pointer of the structure for the front address of the Kingstar server. The address format just like :"protocol://ipaddress:port", for example,"tcp://127.0.0.1:17993","tcp" means the communication protocol , "127.0.0.1" identifies the front address."17993" identifies the server port.*

4.6.8 RegisterSpi

The Kingstar client application uses this function to register an instance inherited from the CThostFtdcMdSpi interface.

definition:

*void RegisterSpi(CthostFtdcMdSpi *pSpi) ;*

parameters:

pSpi: *the pointer of the CThostFtdcMdSpi instance.*

4.6.9 SubscribeMarketData

The Kingstar client application uses this function to send subscribe marketdata request to the Kingstar server.

definition:

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

parameters:

ppInstrumentID[]: *Pointer of the structure for marketdata subscribe request.*

nCount: *The number of subscription contracts.*

4.6.10 UnSubscribeMarketData

The Kingstar client application uses this function to send unsubscribe marketdata request to the Kingstar server.

definition:

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

parameters:

ppInstrumentID[]: *Pointer of the structure for marketdata unsubscribe request.*

nCount: *The number of unsubscription contracts.*

4.6.11 ReqUserLogin

The Kingstar client application uses this function to send the login in request to the Kingstar server.

definition:

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

parameters:

pReqUserLoginField: *The pointer of the structure for user's login request. The following is definition of the structure,*

```
struct CThostFtdcReqUserLoginField
{
    ///trading day
    TThostFtdcDateType  TradingDay;
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
    ///password
    TThostFtdcPasswordType  Password;
    ///user product information
    TThostFtdcProductInfoType   UserProductInfo;
    ///interface product information
    TThostFtdcProductInfoType   InterfaceProductInfo;
    ///protocol information
    TThostFtdcProtocolInfoType  ProtocolInfo;
    ///Mac address
    TThostFtdcMacAddressType    MacAddress;
```

```
    ///one time password
    TThostFtdcPasswordType  OneTimePassword;
    ///client IP address
    TThostFtdcIPAddressType ClientIPAddress;
};
```

return value:

0, success.

-1, net connection failure.

-2, over the max quantity of unhandled requests.

-3, over the max requests per second.

4.6.12 ReqUserLogout

The Kingstar client application uses this function to send the login out request to the Kingstar server.

definition:

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

parameters:

pReqUserLogout: *Pointer of the structure for user's logout request. The following is definition of the structure,*

```
struct CThostFtdcUserLogoutField
{
    ///broker id
    TThostFtdcBrokerIDType  BrokerID;
    ///user id
    TThostFtdcUserIDType    UserID;
};
```

4.6.13 RegisterNameServer

The Kingstar client application uses this function to register nameserver which

can acquire the optimal gateway for fast login. Kingstar API support registerring several nameservers, that is, this function can be called many times with different nameservers and every time of the register will get the first gateway address from gateway lists. One thing need to pay attention to is that once register several times, the responded gateway for final registering will replace the last gateway returned.

definition:

*void RegisterNameServer(char *pszNameServerAddress);*

parameters:

pszNameServerAddress: *Pointer of the structure for the name server address (also be called portal server address) of the Kingstar servers. The name server address format is just like :" protocol://ipaddress:port/proxyuser:proxypass@proxyipaddress:proxyport/gatewayflag/clientid", proxy related fields are optional. For example,"tcp://127.0.0.1:11000/A/80001","tcp" means the communication protocol, "127.0.0.1" identifies the name server address."11000" identifies the server port."A" identifies the gatewayflag, "80001" identifies the ID of the client.*

Gatewayflag field is as follows:

A	Da zhi hui
B	Wen hua yi jian tong
C	Peng bo shan dian shou
D	Tuo rui bang ze
E	Kuai qi
F	Jin zi ta
G	Da qian

4.7 CTKSCosSpi

4.7.1 OnRspInitInsertConditionalOrder

Response to order conditional order

definition:

```
void OnRspInitInsertConditionalOrder(
    CTKSConditionalOrderOperResultField *pInitInsertConditionalOrder,
```

```
    CThostFtdcRspInfoField *pRspInfo,
    int nRequestID,
    bool bIsLast)
```

parameters:

pInitInsertConditionalOrder: Pointer of structure for the response to conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderOperResultField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///ConditionalOrder ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///LocalOrder ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///OrderStatus
    TThostFtdcOrderStatusType   OrderStatus;
    ///CombOffsetFlag
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///CombHedgeFlag
    TThostFtdcHedgeFlagType CombHedgeFlag;
    ///Buy or Sale direction
    TThostFtdcDirectionType Direction;
    ///Order Price
    TThostFtdcPriceType LimitPrice;
    ///Order Volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///Revocation User ID
    TThostFtdcUserIDType    UserID;
    ///Revocation Time
    TThostFtdcTimeType  CancelTime;
    ///Client ID
    TThostFtdcClientIDType  ClientID;
```

```
    ///Conditional Order Status
    TTKSConditionalOrderStatusType ConditionalOrderStatus;
    ///Error Message
    TThostFtdcErrorMsgType  ErrorMsg;
    ///Order Price Type
    TThostFtdcOrderPriceTypeType OrderPriceType;
    ///Triggered Times
    TThostFtdcVolumeType TriggeredTimes;
    ///Conditional Order Type
    TTKSConditionalOrderType OrderType;
    ///Memo
    TThostFtdcMemoType  Memo;
    ///Active Time
    TThostFtdcTimeType  ActiveTime;
    ///Inactive Time
    TThostFtdcTimeType InActiveTime;
};
```

4.7.2 OnRspQueryConditionalOrder

Response to query conditional order

definition:

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

parameters:

pQueryConditionalOrder: Pointer of structure for the response to conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderOperResultField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///ConditionalOrder ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
```

```
    ///LocalOrder ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///OrderStatus
    TThostFtdcOrderStatusType   OrderStatus;
    ///CombOffsetFlag
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///CombHedgeFlag
    TThostFtdcHedgeFlagType CombHedgeFlag;
    ///Buy or Sale direction
    TThostFtdcDirectionType Direction;
    ///Order Price
    TThostFtdcPriceType LimitPrice;
    ///Order Volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///Revocation User ID
    TThostFtdcUserIDType    UserID;
    ///Revocation Time
    TThostFtdcTimeType  CancelTime;
    ///Client ID
    TThostFtdcClientIDType  ClientID;
    ///Conditional Order Status
    TTKSConditionalOrderStatusType ConditionalOrderStatus;
    ///Error Message
    TThostFtdcErrorMsgType  ErrorMsg;
    ///Order Price Type
    TThostFtdcOrderPriceTypeType OrderPriceType;
    ///Triggered Times
    TThostFtdcVolumeType TriggeredTimes;
    ///Conditional Order Type
    TTKSConditionalOrderType OrderType;
    ///Memo
    TThostFtdcMemoType  Memo;
    ///Active Time
    TThostFtdcTimeType  ActiveTime;
    ///Inactive Time
    TThostFtdcTimeType InActiveTime;
};
```

4.7.3 OnRspModifyConditionalOrder

Response to modify conditional order

definition:

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

parameters:

pModifyConditionalOrder: Pointer of structure for the response to conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderOperResultField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///ConditionalOrder ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///LocalOrder ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///OrderStatus
    TThostFtdcOrderStatusType   OrderStatus;
    ///CombOffsetFlag
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///CombHedgeFlag
    TThostFtdcHedgeFlagType CombHedgeFlag;
    ///Buy or Sale direction
    TThostFtdcDirectionType Direction;
```

```
    ///Order Price
    TThostFtdcPriceType LimitPrice;
    ///Order Volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///Revocation User ID
    TThostFtdcUserIDType    UserID;
    ///Revocation Time
    TThostFtdcTimeType  CancelTime;
    ///Client ID
    TThostFtdcClientIDType  ClientID;
    ///Conditional Order Status
    TTKSConditionalOrderStatusType ConditionalOrderStatus;
    ///Error Message
    TThostFtdcErrorMsgType  ErrorMsg;
    ///Order Price Type
    TThostFtdcOrderPriceTypeType OrderPriceType;
    ///Triggered Times
    TThostFtdcVolumeType TriggeredTimes;
    ///Conditional Order Type
    TTKSConditionalOrderType OrderType;
    ///Memo
    TThostFtdcMemoType  Memo;
    ///Active Time
    TThostFtdcTimeType  ActiveTime;
    ///Inactive Time
    TThostFtdcTimeType InActiveTime;
};
```

4.7.4 OnRspPauseConditionalOrder

Response to pause or active conditional order.

definition:

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

parameters:

pPauseConditionalOrder: Pointer of structure for the response to conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderOperResultField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///ConditionalOrder ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///LocalOrder ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///OrderStatus
    TThostFtdcOrderStatusType   OrderStatus;
    ///CombOffsetFlag
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///CombHedgeFlag
    TThostFtdcHedgeFlagType CombHedgeFlag;
    ///Buy or Sale direction
    TThostFtdcDirectionType Direction;
    ///Order Price
    TThostFtdcPriceType LimitPrice;
    ///Order Volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///Revocation User ID
    TThostFtdcUserIDType    UserID;
    ///Revocation Time
    TThostFtdcTimeType  CancelTime;
    ///Client ID
    TThostFtdcClientIDType  ClientID;
    ///Conditional Order Status
    TTKSConditionalOrderStatusType ConditionalOrderStatus;
    ///Error Message
    TThostFtdcErrorMsgType  ErrorMsg;
    ///Order Price Type
    TThostFtdcOrderPriceTypeType OrderPriceType;
```

```
    ///Triggered Times
    TThostFtdcVolumeType TriggeredTimes;
    ///Conditional Order Type
    TTKSConditionalOrderType OrderType;
    ///Memo
    TThostFtdcMemoType  Memo;
    ///Active Time
    TThostFtdcTimeType  ActiveTime;
    ///Inactive Time
    TThostFtdcTimeType InActiveTime;
};
```

4.7.5 OnRspRemoveConditionalOrder

Response to remove conditional order

definition:

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

parameters:

pRemoveConditionalOrder: Pointer of structure for the handle result of conditional order。

The following is definition of the structure:

```
struct CTKSConditionalOrderRspResultField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
};
```

4.7.6 OnRspSelectConditionalOrder

Response to select conditional order

definition:

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

parameters:

pSelectConditionalOrder: Pointer of structure for the handle result of conditional order。

The following is definition of the structure:

```
struct CTKSConditionalOrderRspResultField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
};
```

4.7.7 OnRspInsertProfitAndLossOrder

Response to order the profit and loss order

definition:

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

parameters:

pInsertProfitAndLossOrder: Pointer of structure for handle result of profit and loss order.

The following is definition of the structure:

```
struct CTKSProfitAndLossOrderOperResultField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Profit and Loss Order ID
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///Operator ID
    TThostFtdcUserIDType    UserID;
    ///Investor Name
    TThostFtdcPartyNameType InvestorName;
    ///Local Order ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///StopLoss Price
    TThostFtdcPriceType  StopLossPrice;
    ///TakeProfit Price
    TThostFtdcPriceType  TakeProfitPrice;
    ///Close Mode
    TTKSCloseModeType CloseMode;
    ///Figures
    TThostFtdcPriceType Figures;
    ///Last Price for market data triggers
    TThostFtdcPriceType LastPrice;
    ///Profit and loss Order Creation Time
    TThostFtdcTimeType  ProfitAndLossOrderFormTime;
    ///Conditonal Order Creation Time
    TThostFtdcTimeType  ConditionalOrderFormTime;
    ///Creation time for Order
    TThostFtdcTimeType  OrderFormTime;
    ///Profit and Loss Order Status
    TTKSConditionalOrderStatusType ProfitAndLossOrderStatus;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Client ID
    TThostFtdcClientIDType  ClientID;
    ///Instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
```

```
    ///CombOffsetFlag
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///CombHedgeFlag
    TThostFtdcHedgeFlagType CombHedgeFlag;
    ///Buy or Sale direction
    TThostFtdcDirectionType Direction;
    ///Order Price
    TThostFtdcPriceType LimitPrice;
    ///Order Volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///Profit and Loss Price OffsetValue
    TTKSOffsetValueType OffsetValue;
    ///Business Unit
    TThostFtdcBusinessUnitType  BusinessUnit;
    ///Conditional Order Spring Price Type
    TTKSSpringTypeType SpringType;
    ///FloatLimitPrice
    TThostFtdcPriceType FloatLimitPrice;
    ///OpenTradePrice
    TThostFtdcPriceType OpenTradePrice;
};
```

4.7.8 OnRspModifyProfitAndLossOrder

Response to modify the profit and loss order

definition:

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

parameters:

pModifyProfitAndLossOrder: Pointer of structure for handle result of profit and loss order.

The following is definition of the structure:

```
struct CTKSProfitAndLossOrderOperResultField
{
    ///Broker ID
```

```
TThostFtdcBrokerIDType BrokerID;
///Investor ID
TThostFtdcInvestorIDType InvestorID;
///Profit and Loss Order ID
TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
///Operator ID
TThostFtdcUserIDType    UserID;
///Investor Name
TThostFtdcPartyNameType InvestorName;
///Local Order ID
TThostFtdcOrderLocalIDType  OrderLocalID;
///StopLoss Price
TThostFtdcPriceType  StopLossPrice;
///TakeProfit Price
TThostFtdcPriceType  TakeProfitPrice;
///Close Mode
TTKSCloseModeType CloseMode;
///Figures
TThostFtdcPriceType Figures;
///Last Price for market data triggers
TThostFtdcPriceType LastPrice;
///Profit and loss Order Creation Time
TThostFtdcTimeType  ProfitAndLossOrderFormTime;
///Conditonal Order Creation Time
TThostFtdcTimeType  ConditionalOrderFormTime;
///Creation time for Order
TThostFtdcTimeType  OrderFormTime;
///Profit and Loss Order Status
TTKSConditionalOrderStatusType ProfitAndLossOrderStatus;
///Conditional Order ID
TTKSConditionalOrderIDType  ConditionalOrderID;
///Exchange ID
TThostFtdcExchangeIDType    ExchangeID;
///Client ID
TThostFtdcClientIDType  ClientID;
///Instrument ID
TThostFtdcInstrumentIDType  InstrumentID;
///CombOffsetFlag
TThostFtdcOffsetFlagType    CombOffsetFlag;
///CombHedgeFlag
TThostFtdcHedgeFlagType CombHedgeFlag;
```

```
    ///Buy or Sale direction
    TThostFtdcDirectionType Direction;
    ///Order Price
    TThostFtdcPriceType LimitPrice;
    ///Order Volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///Profit and Loss Price OffsetValue
    TTKSOffsetValueType OffsetValue;
    ///Business Unit
    TThostFtdcBusinessUnitType  BusinessUnit;
    ///Conditional Order Spring Price Type
    TTKSSpringTypeType SpringType;
    ///FloatLimitPrice
    TThostFtdcPriceType FloatLimitPrice;
    ///OpenTradePrice
    TThostFtdcPriceType OpenTradePrice;
};
```

4.7.9 OnRspRemoveProfitAndLossOrder

Response to remove the profit and loss order

definition:

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

parameters:

pRemoveProfitAndLossOrder: Pointer of structure for remove profit and loss order

The following is definition of the structure:

```
struct CTKSProfitAndLossOrderRemoveField
{
     ///Broker ID
     TThostFtdcBrokerIDType BrokerID;
     ///Investor ID
     TThostFtdcInvestorIDType InvestorID;
```

```
        ///Profit and Loss Order ID
        TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
};
```

4.7.10 OnRspQueryProfitAndLossOrder

Response to query the profit and loss order

definition:

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

parameters:

pQueryProfitAndLossOrder: Pointer of structure for handle result of profit and loss order.

The following is definition of the structure:

```
struct CTKSProfitAndLossOrderOperResultField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Profit and Loss Order ID
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///Operator ID
    TThostFtdcUserIDType    UserID;
    ///Investor Name
    TThostFtdcPartyNameType InvestorName;
    ///Local Order ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///StopLoss Price
    TThostFtdcPriceType  StopLossPrice;
    ///TakeProfit Price
    TThostFtdcPriceType  TakeProfitPrice;
    ///Close Mode
    TTKSCloseModeType CloseMode;
```

```
    ///Figures
    TThostFtdcPriceType Figures;
    ///Last Price for market data triggers
    TThostFtdcPriceType LastPrice;
    ///Profit and loss Order Creation Time
    TThostFtdcTimeType  ProfitAndLossOrderFormTime;
    ///Conditonal Order Creation Time
    TThostFtdcTimeType  ConditionalOrderFormTime;
    ///Creation time for Order
    TThostFtdcTimeType  OrderFormTime;
    ///Profit and Loss Order Status
    TTKSConditionalOrderStatusType ProfitAndLossOrderStatus;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Client ID
    TThostFtdcClientIDType  ClientID;
    ///Instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///CombOffsetFlag
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///CombHedgeFlag
    TThostFtdcHedgeFlagType CombHedgeFlag;
    ///Buy or Sale direction
    TThostFtdcDirectionType Direction;
    ///Order Price
    TThostFtdcPriceType LimitPrice;
    ///Order Volume
    TThostFtdcVolumeType    VolumeTotalOriginal;
    ///Profit and Loss Price OffsetValue
    TTKSOffsetValueType OffsetValue;
    ///Business Unit
    TThostFtdcBusinessUnitType  BusinessUnit;
    ///Conditional Order Spring Price Type
    TTKSSpringTypeType SpringType;
    ///FloatLimitPrice
    TThostFtdcPriceType FloatLimitPrice;
    ///OpenTradePrice
    TThostFtdcPriceType OpenTradePrice;
};
```

4.7.11 OnRtnCOSAskSelect

Notification to selection request of conditional order

definition:

void OnRtnCOSAskSelect(CTKSCOSAskSelectField *pCOSAskSelect)

parameters:

pCOSAskSelect: Pointer of structure for selection request of conditional order

The following is definition of the structure:

```
struct CTKSCOSAskSelectField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Operator ID
    TThostFtdcUserIDType    UserID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Sequence Number
    TThostFtdcSequenceNoType    SequenceNo;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///Memo
    TThostFtdcMemoType  Memo;
    ///Select Type
    TTKSConditionalOrderSelectTypeType SelectType;
 };
```

4.7.12 OnRtnCOSStatus

Notification to status of conditional order.

definition:

void OnRtnCOSStatus(CTKSCOSStatusField *pCOSStatus)

parameters:

pCOSStatus: Pointer of structure for status of conditional order

The following is definition of the structure:

```
struct CTKSCOSStatusField
{
```

```
///Broker ID
TThostFtdcBrokerIDType BrokerID;
///Operator ID
TThostFtdcUserIDType UserID;
///Investor ID
TThostFtdcInvestorIDType InvestorID;
///Sequence Number
TThostFtdcSequenceNoType SequenceNo;
///Conditional Order ID
TTKSConditionalOrderIDType   ConditionalOrderID;
///Status of Conditional Order
TTKSConditionalOrderStatusType ConditionalOrderStatus;
///Memo
TThostFtdcMemoType   Memo;
///Local Order ID
TThostFtdcOrderLocalIDType   OrderLocalID;
///Exchange ID
TThostFtdcExchangeIDType ExchangeID;
///Instrument ID
TThostFtdcInstrumentIDType   InstrumentID;
///Order Status
TThostFtdcOrderStatusTypeOrderStatus;
///CombOffsetFlag
TThostFtdcOffsetFlagType CombOffsetFlag;
///CombHedgeFlag
TThostFtdcHedgeFlagType  CombHedgeFlag;
///Buy or Sale Direction
TThostFtdcDirectionType  Direction;
///Order Price
TThostFtdcPriceType  LimitPrice;
///Order Volume
TThostFtdcVolumeType VolumeTotalOriginal;
///Trading Day
```

```
    TThostFtdcTradeDateType  TradingDay;
    ///Revocation User ID
    TThostFtdcUserIDType CancelUserID;
    ///Revocation Time
    TThostFtdcTimeType   CancelTime;
    ///Client ID
    TThostFtdcClientIDType   ClientID;
    /// Business Unit
    TThostFtdcBusinessUnitType   BusinessUnit;
    ///Order System ID
    TThostFtdcOrderSysIDType OrderSysID;
    ///Traded Volume of Today
    TThostFtdcVolumeType VolumeTraded;
    ///Remainder Volume
    TThostFtdcVolumeType VolumeTotal;
    ///Order Time
    TThostFtdcTimeType   InsertTime;
    ///Active Time
    TThostFtdcTimeType   ActiveTime;
    ///Trading Price
    TThostFtdcPriceType  TradePrice;
    ///Currency ID
    TThostFtdcCurrencyIDType    CurrencyID;
};
```

4.7.13 OnRtnPLStatus

Notification to status of profit and loss order.

definition:

void OnRtnPLStatus(CTKSPLStatusField *pPLStatus)

parameters:

pPLStatus: Pointer of structure for status of profit and loss order

The following is definition of the structure:

```
struct CTKSPLStatusField
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Operator ID
    TThostFtdcUserIDType    UserID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Sequence Number
    TThostFtdcSequenceNoType    SequenceNo;
    ///Profit and Loss Order ID
    TTKSProfitAndLossOrderIDType    ProfitAndLossOrderID;
    ///StopLoss Order ID
    TTKSConditionalOrderIDType  StopLossOrderID;
    ///TakeProfit Order ID
    TTKSConditionalOrderIDType  TakeProfitOrderID;
    ///Status of Profit and Loss Order
    TTKSConditionalOrderStatusType ProfitAndLossOrderStatus;
    ///StopLoss Price
    TThostFtdcPriceType  StopLossPrice;
    ///TakeProfit Price
    TThostFtdcPriceType  TakeProfitPrice;
    ///Profit and Loss Price’s Offset Value
    TTKSOffsetValueType OffsetValue;
    ///OpenTradePrice
    TThostFtdcPriceType OpenTradePrice;
    ///Memo
    TThostFtdcMemoType  Memo;
    ///Local Order ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Instrument ID
```

```
TThostFtdcInstrumentIDType  InstrumentID;
///Order Status
TThostFtdcOrderStatusType   OrderStatus;
///CombOffset Flag
TThostFtdcOffsetFlagType    CombOffsetFlag;
///CombHedgeFlag
TThostFtdcHedgeFlagType CombHedgeFlag;
///Buy or Sale Direction
TThostFtdcDirectionType Direction;
///Order Price
TThostFtdcPriceType LimitPrice;
///Order Volume
TThostFtdcVolumeType    VolumeTotalOriginal;
///Trading Day
TThostFtdcTradeDateType TradingDay;
///Revocation User ID
TThostFtdcUserIDType    CancelUserID;
///Revocation Time
TThostFtdcTimeType  CancelTime;
///Client ID
TThostFtdcClientIDType  ClientID;
/// Business Unit
TThostFtdcBusinessUnitType   BusinessUnit;
///Order System ID
TThostFtdcOrderSysIDType    OrderSysID;
///Traded Volume of Today
TThostFtdcVolumeType    VolumeTraded;
///Remainder Volume
TThostFtdcVolumeType    VolumeTotal;
///Order Time
TThostFtdcTimeType  InsertTime;
///Active Time
TThostFtdcTimeType  ActiveTime;
```

```
    ///Trading Price
    TThostFtdcPriceType TradePrice;
    ///Currency ID
    TThostFtdcCurrencyIDType    CurrencyID;
};
```

4.8 CTKSCosApi

4.8.1 ReqInitInsertConditionalOrder

Request of placing conditional order

definition:

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

parameters:

pConditionalOrderInitInsert: Pointer of structure for placing conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderInitInsert
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Instrument ID
    TThostFtdcInstrumentIDType  InstrumentID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Client ID
    TThostFtdcClientIDType  ClientID;
    ///Buy or Sale Direction
    TThostFtdcDirectionType Direction;
    ///CombOffset Flag
```

```
    TThostFtdcOffsetFlagType    CombOffsetFlag;
    ///CombHedge Flag
    TThostFtdcHedgeFlagType CombHedgeFlag;
    ///Order Volume
    TThostFtdcVolumeType     VolumeTotalOriginal;
    ///Order Price
    TThostFtdcPriceType LimitPrice;
    ///Order Price Type
    TTKSOrderPriceTypeType OrderPriceType;
    ///Conditional Type
    TTKSConditionalTypeType  ConditionalType;
    ///Conditional Price
    TThostFtdcPriceType ConditionalPrice;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///Triggered Times
    TThostFtdcVolumeType TriggeredTimes;
    ///Conditional Order Type
    TTKSConditionalOrderType OrderType;
    ///Active Time
    TThostFtdcTimeType  ActiveTime;
    ///Inactive Time
    TThostFtdcTimeType InActiveTime;
    ///Currency ID
    TThostFtdcCurrencyIDType    CurrencyID;
};
```

4.8.2 ReqQueryConditionalOrder

Request of querying conditional order

definition:

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

int nRequestID)

parameters:

pConditionalOrderQuery: Pointer of structure for querying conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderQuery
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
};
```

4.8.3 ReqModifyConditionalOrder

Request of modifying conditional order

definition:

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

parameters:

pConditionalOrderModify: Pointer of structure for modifying conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderModify
{
///Broker ID
TThostFtdcBrokerIDType BrokerID;
///Investor ID
TThostFtdcInvestorIDType InvestorID;
///Instrument ID
TThostFtdcInstrumentIDType  InstrumentID;
///Exchange ID
```

```
TThostFtdcExchangeIDType    ExchangeID;
///Client ID
TThostFtdcClientIDType  ClientID;
///Buy or Sale Direction
TThostFtdcDirectionType Direction;
///CombOffset Flag
TThostFtdcOffsetFlagType    CombOffsetFlag;
///CombHedge Flag
TThostFtdcHedgeFlagType CombHedgeFlag;
///Order Volume
TThostFtdcVolumeType    VolumeTotalOriginal;
///Order Price
TThostFtdcPriceType LimitPrice;
///Order Price Type
TTKSOrderPriceTypeType OrderPriceType;
///Conditional Type
TTKSConditionalTypeType  ConditionalType;
///Conditional Price
TThostFtdcPriceType ConditionalPrice;
///Conditional Order ID
TTKSConditionalOrderIDType  ConditionalOrderID;
///Triggered Times
TThostFtdcVolumeType TriggeredTimes;
///Conditional Order Type
TTKSConditionalOrderType OrderType;
///Active Time
TThostFtdcTimeType  ActiveTime;
///Inactive Time
TThostFtdcTimeType InActiveTime;
///Currency ID
TThostFtdcCurrencyIDType    CurrencyID;
};
```

4.8.4 ReqRemoveConditionalOrder

Request of removing conditional order

definition:

```
int ReqRemoveConditionalOrder(
    CTKSConditionalOrderRemove *pConditionalOrderRemove,
    int nRequestID)
```

parameters:

pConditionalOrderRemove: Pointer of structure for removing conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderRemove
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
};
```

4.8.5 ReqStateAlterConditionalOrder

Request of pausing or activing conditional order

definition:

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

parameters:

pConditionalOrderStateAlter: Pointer of structure for pausing or activing conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderStateAlter
{
```

```
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Conditional Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///Pause or Active Conditional Order Flag
    TTKSConditionalOrderStateAlterType  ConditionalOrderStateAlter;
};
```

4.8.6 ReqSelectConditionalOrder

Request of selecting conditional order

definition:

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

parameters:

pConditionalOrderSelect: Pointer of structure for selecting conditional order

The following is definition of the structure:

```
struct CTKSConditionalOrderSelect
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Conditonal Order ID
    TTKSConditionalOrderIDType  ConditionalOrderID;
    ///Select Result
    TTKSConditionalOrderSelectResultType SelectResult;
};
```

4.8.7 ReqInsertProfitAndLossOrder

Request of placing profit and loss order

definition:

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

parameters:

pProfitAndLossOrderInsert: Pointer of structure for placing profit and loss order

The following is definition of the structure:

```
struct CTKSProfitAndLossOrderInsert
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Local Order ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///StopLoss Price
    TThostFtdcPriceType  StopLossPrice;
    ///TakeProfit Price
    TThostFtdcPriceType  TakeProfitPrice;
    ///Close Mode
    TTKSCloseModeType CloseMode;
    //FiguresPrice
    TThostFtdcPriceType FiguresPrice;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///BusinessUnit
    TThostFtdcBusinessUnitType   BusinessUnit;
    ///Profit and Loss Price Offset Value
    TTKSOffsetValueType OffsetValue;
    ///Conditional Order Spring Price Type
    TTKSSpringTypeType SpringType;
    ///Float Limit Price
    TThostFtdcPriceType FloatLimitPrice;
    ///TriggeredTimes
    TThostFtdcVolumeType TriggeredTimes;
```

};

4.8.8 ReqModifyProfitAndLossOrder

Request of modifying profit and loss order

definition:

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

parameters:

pProfitAndLossOrderModify: Pointer of structure for modifying profit and loss order

The following is definition of the structure:

```
struct CTKSProfitAndLossOrderModify
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Profit And Loss Order ID
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///StopLoss Price
    TThostFtdcPriceType  StopLossPrice;
    ///TakeProfit Price
    TThostFtdcPriceType  TakeProfitPrice;
    ///CloseMode
    TTKSCloseModeType CloseMode;
    ///Figures Price
    TThostFtdcPriceType FiguresPrice;
    ///Profit and Loss Price Offset Value
    TTKSOffsetValueType OffsetValue;
    ///Conditional Order Spring Price Type
    TTKSSpringTypeType SpringType;
```

```
    ///Float Limit Price
    TThostFtdcPriceType FloatLimitPrice;
    ///Triggered Times
    TThostFtdcVolumeType TriggeredTimes;
};
```

4.8.9 ReqRemoveProfitAndLossOrder

Request of removing profit and loss order

definition:

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

parameters:

pProfitAndLossOrderRemove: Pointer of structure for removing profit and loss order

The following is definition of the structure:

```
struct CTKSProfitAndLossOrderRemove
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Profit and Loss Order ID
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///Local Order ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///Business Unit
    TThostFtdcBusinessUnitType BusinessUnit;
};
```

4.8.10 ReqQueryProfitAndLossOrder

Request of querying profit and loss order。

definition:

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

parameters:

pProfitAndLossOrderQuery: Pointer of structure for querying profit and loss order

The following is definition of the structure:

```
struct CTKSProfitAndLossOrderQuery
{
    ///Broker ID
    TThostFtdcBrokerIDType BrokerID;
    ///Investor ID
    TThostFtdcInvestorIDType InvestorID;
    ///Profit and Loss Order ID
    TTKSProfitAndLossOrderIDType ProfitAndLossOrderID;
    ///Local Order ID
    TThostFtdcOrderLocalIDType  OrderLocalID;
    ///Exchange ID
    TThostFtdcExchangeIDType    ExchangeID;
    ///BusinessUnit
    TThostFtdcBusinessUnitType   BusinessUnit;
};
```

Chapter5、 Sample code

See the Demo kit folder.

Chapter6、 Feedback

If you have any problems using the interface in the process, please submit your detailed feedback to the e-mail:Mingming.shen@sungard.com.

Thanks!