金牛资产管理系统终端 API

上海览逸信息科技有限公司

二零一二年八月

目　　录

1.介绍

文件属性	内容
文件名称	金牛资产管理系统终端 API
文件版本号	V0.1
作者	上海览逸信息科技有限公司
文档编写日期	2012.08.01

交易托管系统 API 是一个基于 C++的类库, 通过使用和扩展类库提供的接口来实现相关交易功能，包括报单与报价的录入、报单与报价的撤销、报单与报价的挂起、报单与报价的激活、报单与报价的修改、报单与报价的查询、成交单查询、投资者查询、投资者持仓查询、合约查询、交易日获取等。该类库包含以下 5 个文件：

FtdcTraderApi.h	交易接口头文件
FtdcUserApiStruct.h	定义了 API 所需的一系列数据类型的头文件
FtdcUserApiDataType.h	定义了一系列业务相关的数据结构的头文件
thosttraderapi.dll	动态链接库二进制文件
thostraderapi.lib	导入库文件
thostmduserapi.dll	动态链接库二进制文件
thostmduserapi.lib	导入库文件

支持 MS VC 6.0，MS VC.NET 2003 编译器。需要打开多线程编译选项/MT。

2. 接口模式

交易员 API 提供了二个接口， 分别为 CThostFtdcTraderApi 和 CThostFtdcTraderSpi。这两个接口对 FTD 协议进行了封装，方便客户端应用程序的开发。
客户端应用程序可以通过 CThostFtdcTraderApi 发出操作请求，通继承 CThostFtdcTraderSpi 并重载回调函数来处理后台服务的响应。

2.1. 对话流和查询流编程接口

通过对话流进行通讯的编程接口通常如下：

```
请求：int CThostFtdcTraderApi::ReqXXX(
            CThostFtdcXXXField *pReqXXX,
            int nRequestID)
响应：void CThostFtdcTraderSpi::OnRspXXX(
            CThostFtdcXXXField *pRspXXX,
```

CThostFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast)

其中请求接口第一个参数为请求的内容，不能为空。
第二个参数为请求号。请求号由客户端应用程序负责维护，正常情况下每个请求的请求号不要重复。在接收交易托管系统的响应时，可以得到当时发出请求时填写的请求号，从而可以将响应与请求对应起来。
当收到后台服务应答时，CThostFtdcTraderSpi 的回调函数会被调用。如果响应数据不止一个，则回调函数会被多次调用。
回调函数的第一个参数为响应的具体数据，如果出错或没有结果有可能为 NULL。
第二个参数为处理结果，表明本次请求的处理结果是成功还是失败。在发生多次回调时，除了第一次回调，其它的回调该参数都可能为 NULL。
第三个参数为请求号，即原来发出请求时填写的请求号。
第四个参数为响应结束标志，表明是否是本次响应的最后一次回调。

2.2. 私有流编程接口

私有流中的数据中会员的私有信息，包括报单回报、成交回报等。

通过私有流接收回报的编程接口通常如下：

void CThostFtdcTraderSpi::OnRtnXXX(CThostFtdcXXXField *pXXX) 或

void CThostFtdcTraderSpi::OnErrRtnXXX(CThostFtdcXXXField *pXXX,

CThostFtdcRspInfoField *pRspInfo)

当收到交易托管系统通过私有流发布的回报数据时，CThostFtdcTraderSpi 的回调函数会被调用。回调函数的参数为回报的具体内容。

3. 业务与接口对照

业务类型	业务	请求接口	响应接口	数据流
登录	登录	CThostFtdcTraderApi::ReqUserLogin	CThostFtdcTraderSpi::OnRspUserLogin	对话流
	登出	CThostFtdcTraderApi::ReqUserLogout	CThostFtdcTraderSpi::OnRspUserLogout	对话流
	修改用户口令	CThostFtdcTraderApi::ReqUserPasswordUpdate	CThostFtdcTraderSpi::OnRspUserPasswordUpdate	对话流
交易	报单录入	CThostFtdcTraderApi::ReqOrderInsert	CThostFtdcTraderSpi::OnRspOrderInsert	对话流
	报单操作	CThostFtdcTraderApi::R	CThostFtdcTraderSpi::OnR	对话流

		eqOrderAction	spOrderAction	
私有回报	成交回报	N/A	CThostFtdcTraderSpi::OnRtnTrade	对话流
	报单回报	N/A	CThostFtdcTraderSpi::OnRtnOrder	对话流
	报单查询	CThostFtdcTraderApi::ReqQryOrder	CThostFtdcTraderSpi::OnRspQryOrder	查询流
	成交查询	CThostFtdcTraderApi::ReqQryTrade	CThostFtdcTraderSpi::OnRspQryTrade	查询流
	投资者查询	CThostFtdcTraderApi::ReqQry Investor	CThostFtdcTraderSpi::OnRspQry Investor	查询流
	投资者持仓查询	CThostFtdcTraderApi::ReqQry Investor	Position CThostFtdcTraderSpi::OnRspQry Investor Position	查询流
	合约查询	CThostFtdcTraderApi::ReqQryInstrument	CThostFtdcTraderSpi::OnRspQryInstrument	查询流

交易接口和私有流接口会有相互关联，如用户报单录入 ReqOrderInsert，马上会收到报单响应 OnRspOrderInsert，说明交易系统已
经收到报单。报单进入交易系统后，如果报单的交易状态发生变化，就会收到报单回报 OnRtnOrder。如果报单被撮合(部分)成交，就
会收到成交回报 OnRtnTrade。其中，一个用户的报单回报和成交回报也会被所属会员下其他的用户接受到。

4. 开发接口

4.1. 通用规则

客户端和交易托管系统的通讯过程分为 2 个阶段：初始化阶段和功能调用阶段。
在初始化阶段，程序必须完成如下步骤（具体代码请参考开发实例）：
1, 产生一个 CThostFtdcTraderApi 实例
2, 产生一个事件处理的实例
3, 注册一个事件处理的实例
4, 订阅私有流
5, 订阅公共流
6, 设置交易托管服务的地址

在功能调用阶段，程序可以任意调用交易接口中的请求方法，如 ReqOrderInsert 等。同时按照需要响应回调接口中的。

其他注意事项：
1，API 请求的输入参数不能为 NULL。

2，API 请求的返回参数，0 表示正确，其他表示错误，详细错误编码请查表。

4.2. CThostFtdcTraderSpi 接口

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

4.2.1. OnFrontConnected 方法

当客户端与交易托管系统建立起通信连接时（还未登录前），该方法被调用。
函数原形：
void OnFrontConnected()；
本方法在完成初始化后调用，可以在其中完成用户登录任务。

4.2.2. OnFrontDisconnected 方法

当客户端与交易托管系统通信连接断开时，该方法被调用。当发生这个情况后，API 会自动重新连接，客户端可不做处理。自动重连地址，可能是原来注册的地址，也可能是系统支持的其它可用的通信地址，它由程序自动选择。
函数原形：
void OnFrontDisconnected (int nReason)；
参数：
nReason：连接断开原因
0x1001 网络读失败
0x1002 网络写失败
0x2001 接收心跳超时
0x2002 发送心跳失败
0x2003 收到错误报文

4.2.3. OnHeartBeatWarning 方法

心跳超时警告。当长时间未收到报文时，该方法被调用。
函数原形：
void OnHeartBeatWarning(int nTimeLapse)；
参数：
nTimeLapse：距离上次接收报文的时间

4.2.4. OnRspUserLogin 方法

当客户端发出登录请求之后，交易托管系统返回响应时，该方法会被调用，
通知客户端登录是否成功。

函数原形：

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

参数：

pRspUserLogin：返回用户登录信息的地址。

用户登录信息结构：

```
struct CThostFtdcRspUserLoginField
{
  ///交易日
  TThostFtdcDateType  TradingDay;
  ///登录成功时间
  TThostFtdcTimeType  LoginTime;
  ///经纪公司代码
  TThostFtdcBrokerIDType  BrokerID;
  ///用户代码
  TThostFtdcUserIDType UserID;
  ///交易系统名称
  TThostFtdcSystemNameType  SystemName;
};
```

pRspInfo：返回用户响应信息的地址。特别注意在有连续的成功的响应数据
时，中间有可能返回 NULL，但第一次不会，以下同。错误代码为 0 时，表示操
作成功，以下同。

响应信息结构：

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

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

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

4.2.5. OnRspUserLogout 方法

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

函数原形：

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

参数：

pRspUserLogout：返回用户退出信息的地址。

用户登出信息结构：

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

pRspInfo：返回用户响应信息的地址。

响应信息结构：

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

nRequestID：返回用户登出请求的 ID，该 ID 由用户在登出时指定。

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

4.2.6. OnRspUserPasswordUpdate 方法

用户密码修改应答。当客户端发出用户密码修改指令后，交易托管系统返回
响应时，该方法会被调用。

函数原形：

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

参数：

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

用户密码修改结构：

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

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID： 返回用户密码修改请求的 ID， 该 ID 由用户在密码修改时指定。

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

4.2.7. OnRspTradingAccountPasswordUpdate 方法

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

函数原形：

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

参数：

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

资金账户口令变更域结构：

```
struct CThostFtdcTradingAccountPasswordUpdateField
{
```

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

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID： 返回用户密码修改请求的 ID， 该 ID 由用户在密码修改时指定。

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

4.2.8. OnRspOrderInsert 方法

报单录入应答。 当客户端发出过报单录入指令后， 交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

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

输入报单结构：

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

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

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

```
    TThostFtdcErrorMsgType    ErrorMsg;
};
```

nRequestID： 返回报单录入操作请求的 ID， 该 ID 由用户在报单录入时指定。
bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

4.2.9. OnRspOrderAction 方法

报单操作应答。报单操作包括报单的撤销、报单的挂起、报单的激活、报单的修改。当客户端发出过报单操作指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

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

报单操作结构：

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

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

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID： 返回用户报单操作请求的 ID， 该 ID 由用户在报单操作时指定。

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

4.2.10. OnRspSettlementInfoConfirm 方法

投资者结算结果确认应答。当客户端发出投资者结算结果确认指令后，交易

托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

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

投资者结算结果确认信息结构：

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

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID： 返回用户报单操作请求的 ID， 该 ID 由用户在报单操作时指定。

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

4.2.11. OnRspQryOrder 方法

报单查询请求。当客户端发出报单查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

```
void OnRspQryOrder(
CThostFtdcOrderField *pOrder,
CThostFtdcRspInfoField *pRspInfo,
int nRequestID,
```

bool bIsLast);

参数：

pOrder：指向报单信息结构的地址。

报单信息结构：

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

```
///业务单元
TThostFtdcBusinessUnitType    BusinessUnit;
///请求编号
TThostFtdcRequestIDType    RequestID;
///本地报单编号
TThostFtdcOrderLocalIDType    OrderLocalID;
///交易所代码
TThostFtdcExchangeIDType ExchangeID;
///会员代码
TThostFtdcParticipantIDType    ParticipantID;
///客户代码
TThostFtdcClientIDType    ClientID;
///合约在交易所的代码
TThostFtdcExchangeInstIDType    ExchangeInstID;
///交易所交易员代码
TThostFtdcTraderIDType    TraderID;
///安装编号
TThostFtdcInstallIDType    InstallID;
///报单提交状态
TThostFtdcOrderSubmitStatusType    OrderSubmitStatus;
///报单提示序号
TThostFtdcSequenceNoType NotifySequence;
///交易日
TThostFtdcDateType    TradingDay;
///结算编号
TThostFtdcSettlementIDType    SettlementID;
///报单编号
TThostFtdcOrderSysIDType    OrderSysID;
///报单来源
TThostFtdcOrderSourceType OrderSource;
///报单状态
TThostFtdcOrderStatusType    OrderStatus;
///报单类型
TThostFtdcOrderTypeType    OrderType;
///今成交数量
TThostFtdcVolumeType VolumeTraded;
///剩余数量
TThostFtdcVolumeType VolumeTotal;
///报单日期
TThostFtdcDateType    InsertDate;
///插入时间
TThostFtdcTimeType    InsertTime;
///激活时间
TThostFtdcTimeType    ActiveTime;
```

```
    ///挂起时间
    TThostFtdcTimeType    SuspendTime;
    ///最后修改时间
    TThostFtdcTimeType    UpdateTime;
    ///撤销时间
    TThostFtdcTimeType    CancelTime;
    ///最后修改交易所交易员代码
    TThostFtdcTraderIDType    ActiveTraderID;
    ///结算会员编号
    TThostFtdcParticipantIDType    ClearingPartID;
    ///序号
    TThostFtdcSequenceNoType SequenceNo;
    ///前置编号
    TThostFtdcFrontIDType FrontID;
    ///会话编号
    TThostFtdcSessionIDType    SessionID;
    ///用户端产品信息
    TThostFtdcProductInfoType    UserProductInfo;
    ///状态信息
    TThostFtdcErrorMsgType    StatusMsg;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID： 返回用户报单查询请求的 ID， 该 ID 由用户在报单查询时指定。

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

4.2.12. OnRspQryTrade 方法

成交单查询应答。当客户端发出成交单查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

pTrade：指向成交信息结构的地址。

成交信息结构：

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

```
    TThostFtdcTradeTypeType    TradeType;
    ///成交价来源
    TThostFtdcPriceSourceType    PriceSource;
    ///交易所交易员代码
    TThostFtdcTraderIDType    TraderID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType    OrderLocalID;
    ///结算会员编号
    TThostFtdcParticipantIDType    ClearingPartID;
    ///业务单元
    TThostFtdcBusinessUnitType    BusinessUnit;
    ///序号
    TThostFtdcSequenceNoType SequenceNo;
    ///交易日
    TThostFtdcDateType    TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType    SettlementID;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID： 返回用户成交单请求的 ID， 该 ID 由用户在成交单查询时指定。

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

4.2.13. OnRspQryInvestor 方法

会员客户查询应答。当客户端发出会员客户查询指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

p Investor：指向投资者信息结构的地址。

投资者信息结构：

```
struct CThostFtdcInvestorField
{
  ///投资者代码
  TThostFtdcInvestorIDType  InvestorID;
  ///经纪公司代码
  TThostFtdcBrokerIDType  BrokerID;
  ///投资者分组代码
  TThostFtdcInvestorIDType  InvestorGroupID;
  ///投资者名称
  TThostFtdcPartyNameType  InvestorName;
  ///证件类型
  TThostFtdcIdCardTypeType  IdentifiedCardType;
  ///证件号码
  TThostFtdcIdentifiedCardNoType IdentifiedCardNo;
  ///是否活跃
  TThostFtdcBoolType  IsActive;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

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

4.2.14. OnRspQryInvestorPosition 方法

投资者持仓查询应答。当客户端发出投资者持仓查询指令后，后交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

pInvestorPosition：指向投资者持仓应答结构的地址。

投资者持仓应答结构：

```
struct CThostFtdcInvestorPositionField
{
    ///合约代码
    TThostFtdcInstrumentIDType    InstrumentID;
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///持仓多空方向
    TThostFtdcPosiDirectionType    PosiDirection;
    ///投机套保标志
    TThostFtdcHedgeFlagType    HedgeFlag;
    ///持仓日期
    TThostFtdcPositionDateType PositionDate;
    ///上日持仓
    TThostFtdcVolumeType YdPosition;
    ///今日持仓
    TThostFtdcVolumeType Position;
    ///多头冻结
    TThostFtdcVolumeType LongFrozen;
    ///空头冻结
    TThostFtdcVolumeType ShortFrozen;
    ///开仓冻结金额
    TThostFtdcMoneyType    LongFrozenAmount;
    ///开仓冻结金额
    TThostFtdcMoneyType    ShortFrozenAmount;
    ///开仓量
    TThostFtdcVolumeType OpenVolume;
    ///平仓量
    TThostFtdcVolumeType CloseVolume;
    ///开仓金额
    TThostFtdcMoneyType    OpenAmount;
    ///平仓金额
    TThostFtdcMoneyType    CloseAmount;
    ///持仓成本
    TThostFtdcMoneyType    PositionCost;
    ///上次占用的保证金
    TThostFtdcMoneyType    PreMargin;
    ///占用的保证金
    TThostFtdcMoneyType    UseMargin;
    ///冻结的保证金
    TThostFtdcMoneyType    FrozenMargin;
    ///冻结的资金
    TThostFtdcMoneyType    FrozenCash;
```

```
    ///冻结的手续费
    TThostFtdcMoneyType    FrozenCommission;
    ///资金差额
    TThostFtdcMoneyType    CashIn;
    ///手续费
    TThostFtdcMoneyType    Commission;
    ///平仓盈亏
    TThostFtdcMoneyType    CloseProfit;
    ///持仓盈亏
    TThostFtdcMoneyType    PositionProfit;
    ///上次结算价
    TThostFtdcPriceType    PreSettlementPrice;
    ///本次结算价
    TThostFtdcPriceType    SettlementPrice;
    ///交易日
    TThostFtdcDateType    TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType    SettlementID;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID：返回会员持仓查询请求的 ID，该 ID 由用户在会员持仓查询时指定。

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

4.2.15. OnRspQryTradingAccount 方法

请求查询资金账户响应。当客户端发出请求查询资金账户指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

pTradingAccount：指向资金账户结构的地址。
资金账户结构：

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

  ///上次质押金额
  TThostFtdcMoneyType  PreMortgage;
  ///上次信用额度
  TThostFtdcMoneyType  PreCredit;
  ///上次存款额
  TThostFtdcMoneyType  PreDeposit;
  ///上次结算准备金
  TThostFtdcMoneyType  PreBalance;
  ///上次占用的保证金
  TThostFtdcMoneyType  PreMargin;
  ///利息基数
  TThostFtdcMoneyType  InterestBase;
  ///利息收入
  TThostFtdcMoneyType  Interest;
  ///入金金额
  TThostFtdcMoneyType  Deposit;
  ///出金金额
  TThostFtdcMoneyType  Withdraw;
  ///冻结的保证金
  TThostFtdcMoneyType  FrozenMargin;
  ///冻结的资金
  TThostFtdcMoneyType  FrozenCash;
  ///冻结的手续费
  TThostFtdcMoneyType  FrozenCommission;
  ///当前保证金总额
  TThostFtdcMoneyType  CurrMargin;
  ///资金差额
  TThostFtdcMoneyType  CashIn;
  ///手续费
  TThostFtdcMoneyType  Commission;
  ///平仓盈亏
  TThostFtdcMoneyType  CloseProfit;
  ///持仓盈亏
  TThostFtdcMoneyType  PositionProfit;
  ///期货结算准备金
```

```
    TThostFtdcMoneyType    Balance;
    ///可用资金
    TThostFtdcMoneyType    Available;
    ///可取资金
    TThostFtdcMoneyType    WithdrawQuota;
    ///基本准备金
    TThostFtdcMoneyType    Reserve;
    ///交易日
    TThostFtdcDateType    TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType    SettlementID;
    ///信用额度
    TThostFtdcMoneyType    Credit;
    ///质押金额
    TThostFtdcMoneyType    Mortgage;
    ///交易所保证金
    TThostFtdcMoneyType    ExchangeMargin;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

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

4.2.16. OnRspQryInstrument 方法

请求查询合约响应。当客户端发出请求查询合约指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

pInstrument：指向合约结构的地址。

合约结构：

```
struct CThostFtdcInstrumentField
{
    ///合约代码
    TThostFtdcInstrumentIDType    InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType ExchangeID;
    ///合约名称
    TThostFtdcInstrumentNameType    InstrumentName;
    ///合约在交易所的代码
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///产品代码
    TThostFtdcInstrumentIDType    ProductID;
    ///产品类型
    TThostFtdcProductClassType    ProductClass;
    ///交割年份
    TThostFtdcYearType    DeliveryYear;
    ///交割月
    TThostFtdcMonthType    DeliveryMonth;
    ///市价单最大下单量
    TThostFtdcVolumeType MaxMarketOrderVolume;
    ///市价单最小下单量
    TThostFtdcVolumeType MinMarketOrderVolume;
    ///限价单最大下单量
    TThostFtdcVolumeType MaxLimitOrderVolume;
    ///限价单最小下单量
    TThostFtdcVolumeType MinLimitOrderVolume;
    ///合约数量乘数
    TThostFtdcVolumeMultipleType    VolumeMultiple;
    ///最小变动价位
    TThostFtdcPriceType    PriceTick;
    ///创建日
    TThostFtdcDateType    CreateDate;
    ///上市日
    TThostFtdcDateType    OpenDate;
    ///到期日
    TThostFtdcDateType    ExpireDate;
    ///开始交割日
    TThostFtdcDateType    StartDelivDate;
    ///结束交割日
    TThostFtdcDateType    EndDelivDate;
    ///合约生命周期状态
    TThostFtdcInstLifePhaseType    InstLifePhase;
    ///当前是否交易
    TThostFtdcBoolType    IsTrading;
```

```
    ///持仓类型
    TThostFtdcPositionTypeType    PositionType;
    ///持仓日期类型
    TThostFtdcPositionDateTypeType PositionDateType;
    ///多头保证金率
    TThostFtdcRatioType    LongMarginRatio;
    ///空头保证金率
    TThostFtdcRatioType    ShortMarginRatio;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

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

4.2.17. OnRspQrySettlementInfo 方法

请求查询投资者结算结果响应。 当客户端发出请求查询投资者结算结果指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

pSettlementInfo：指向投资者结算结果结构的地址。

投资者结算结果结构：

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

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

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

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

4.2.18. OnRspQryInvestorPositionDetail 方法

请求查询投资者持仓明细响应。 当客户端发出请求请求查询投资者持仓明细指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

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

投资者持仓明细结构：

```
struct CThostFtdcInvestorPositionDetailField
{
    ///合约代码
    TThostFtdcInstrumentIDType    InstrumentID;
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///投机套保标志
    TThostFtdcHedgeFlagType    HedgeFlag;
```

```
    ///买卖方向
    TThostFtdcDirectionType    Direction;
    ///开仓日期
    TThostFtdcDateType    OpenDate;
    ///成交编号
    TThostFtdcTradeIDType    TradeID;
    ///数量
    TThostFtdcVolumeType Volume;
    ///开仓价
    TThostFtdcPriceType    OpenPrice;
    ///交易日
    TThostFtdcDateType    TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType    SettlementID;
    ///成交类型
    TThostFtdcTradeTypeType    TradeType;
    ///组合合约代码
    TThostFtdcInstrumentIDType    CombInstrumentID;
};
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。

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

4.2.19. OnRspQryNotice 方法

请求查询客户通知响应。当客户端发出请求查询客户通知指令后，交易托管系统返回响应时，该方法会被调用。

函数原形：

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

参数：

pNotice：指向客户通知结构的地址。
客户通知结构：
///客户通知
struct CThostFtdcNoticeField
{
  ///经纪公司代码
  TThostFtdcBrokerIDType BrokerID;
  ///消息正文
  TThostFtdcContentType Content;
  ///经纪公司通知内容序列号
  TThostFtdcSequenceLabelType SequenceLabel;
};
pRspInfo：指向响应信息结构的地址。
响应信息结构：
  struct CThostFtdcRspInfoField
{
  ///错误代码
  TThostFtdcErrorIDType ErrorID;
  ///错误信息
  TThostFtdcErrorMsgType ErrorMsg;
};
nRequestID：返回会员客户查询请求的 ID，该 ID 由用户在会员客户查询时指定。
bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

4.2.20. OnRspQryInstrument 方法

合约查询应答。当客户端发出合约查询指令后，交易托管系统返回响应时，该方法会被调用。
函数原形：
void OnRspQryInstrument(
CThostFtdcInstrumentField *pInstrument,
CThostFtdcRspInfoField *pRspInfo,
int nRequestID,
bool bIsLast);
参数：
pRspInstrument：指向合约结构的地址。
  合约结构：
  struct CThostFtdcInstrumentField
{
  ///合约代码
  TThostFtdcInstrumentIDType InstrumentID;

```
///交易所代码
TThostFtdcExchangeIDType ExchangeID;
///合约名称
TThostFtdcInstrumentNameType    InstrumentName;
///合约在交易所的代码
TThostFtdcExchangeInstIDType    ExchangeInstID;
///产品代码
TThostFtdcInstrumentIDType    ProductID;
///产品类型
TThostFtdcProductClassType    ProductClass;
///交割年份
TThostFtdcYearType    DeliveryYear;
///交割月
TThostFtdcMonthType    DeliveryMonth;
///市价单最大下单量
TThostFtdcVolumeType MaxMarketOrderVolume;
///市价单最小下单量
TThostFtdcVolumeType MinMarketOrderVolume;
///限价单最大下单量
TThostFtdcVolumeType MaxLimitOrderVolume;
///限价单最小下单量
TThostFtdcVolumeType MinLimitOrderVolume;
///合约数量乘数
TThostFtdcVolumeMultipleType    VolumeMultiple;
///最小变动价位
TThostFtdcPriceType    PriceTick;
///创建日
TThostFtdcDateType    CreateDate;
///上市日
TThostFtdcDateType    OpenDate;
///到期日
TThostFtdcDateType    ExpireDate;
///开始交割日
TThostFtdcDateType    StartDelivDate;
///结束交割日
TThostFtdcDateType    EndDelivDate;
///合约生命周期状态
TThostFtdcInstLifePhaseType    InstLifePhase;
///当前是否交易
TThostFtdcBoolType    IsTrading;
///持仓类型
TThostFtdcPositionTypeType    PositionType;
///持仓日期类型
TThostFtdcPositionDateTypeType PositionDateType;
```

};
pRspInfo：指向响应信息结构的地址。
响应信息结构：
struct CThostFtdcRspInfoField
{
///错误代码
TThostFtdcErrorIDType ErrorID;
///错误信息
TThostFtdcErrorMsgType ErrorMsg;
};
nRequestID：返回合约查询请求的 ID，该 ID 由用户在合约查询时指定。
bIsLast：指示该次返回是否为针对 nRequestID 的最后一次返回。

4.2.21. OnRtnTrade 方法

成交回报。当发生成交时交易托管系统会通知客户端，该方法会被调用。
函数原形：
void OnRtnTrade(CThostFtdcTradeField *pTrade)；
参数：
pTrade：指向成交信息结构的地址。
成交信息结构：
struct CThostFtdcTradeField
{
///经纪公司代码
TThostFtdcBrokerIDType BrokerID;
///投资者代码
TThostFtdcInvestorIDType InvestorID;
///合约代码
TThostFtdcInstrumentIDType InstrumentID;
///报单引用
TThostFtdcOrderRefType OrderRef;
///用户代码
TThostFtdcUserIDType UserID;
///交易所代码
TThostFtdcExchangeIDType ExchangeID;
///成交编号
TThostFtdcTradeIDType TradeID;
///买卖方向
TThostFtdcDirectionType Direction;
///报单编号
TThostFtdcOrderSysIDType OrderSysID;
///会员代码
TThostFtdcParticipantIDType ParticipantID;

```
    ///客户代码
    TThostFtdcClientIDType    ClientID;
    ///交易角色
    TThostFtdcTradingRoleType TradingRole;
    ///合约在交易所的代码
    TThostFtdcExchangeInstIDType    ExchangeInstID;
    ///开平标志
    TThostFtdcOffsetFlagType    OffsetFlag;
    ///投机套保标志
    TThostFtdcHedgeFlagType    HedgeFlag;
    ///价格
    TThostFtdcPriceType    Price;
    ///数量
    TThostFtdcVolumeType Volume;
    ///成交时期
    TThostFtdcDateType    TradeDate;
    ///成交时间
    TThostFtdcTimeType    TradeTime;
    ///成交类型
    TThostFtdcTradeTypeType    TradeType;
    ///成交价来源
    TThostFtdcPriceSourceType    PriceSource;
    ///交易所交易员代码
    TThostFtdcTraderIDType    TraderID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType    OrderLocalID;
    ///结算会员编号
    TThostFtdcParticipantIDType    ClearingPartID;
    ///业务单元
    TThostFtdcBusinessUnitType    BusinessUnit;
    ///序号
    TThostFtdcSequenceNoType SequenceNo;
    ///交易日
    TThostFtdcDateType    TradingDay;
    ///结算编号
    TThostFtdcSettlementIDType    SettlementID;
};
```

4.2.22. OnRtnOrder 方法

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

函数原形：

void OnRtnOrder(CThostFtdcOrderField *pOrder);

参数：

pOrder：指向报单信息结构的地址。

报单信息结构：

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

```
///业务单元
TThostFtdcBusinessUnitType    BusinessUnit;
///请求编号
TThostFtdcRequestIDType    RequestID;
///本地报单编号
TThostFtdcOrderLocalIDType    OrderLocalID;
///交易所代码
TThostFtdcExchangeIDType ExchangeID;
///会员代码
TThostFtdcParticipantIDType    ParticipantID;
///客户代码
TThostFtdcClientIDType    ClientID;
///合约在交易所的代码
TThostFtdcExchangeInstIDType    ExchangeInstID;
///交易所交易员代码
TThostFtdcTraderIDType    TraderID;
///安装编号
TThostFtdcInstallIDType    InstallID;
///报单提交状态
TThostFtdcOrderSubmitStatusType    OrderSubmitStatus;
///报单提示序号
TThostFtdcSequenceNoType NotifySequence;
///交易日
TThostFtdcDateType    TradingDay;
///结算编号
TThostFtdcSettlementIDType    SettlementID;
///报单编号
TThostFtdcOrderSysIDType    OrderSysID;
///报单来源
TThostFtdcOrderSourceType OrderSource;
///报单状态
TThostFtdcOrderStatusType    OrderStatus;
///报单类型
TThostFtdcOrderTypeType    OrderType;
///今成交数量
TThostFtdcVolumeType VolumeTraded;
///剩余数量
TThostFtdcVolumeType VolumeTotal;
///报单日期
TThostFtdcDateType    InsertDate;
///插入时间
TThostFtdcTimeType    InsertTime;
///激活时间
TThostFtdcTimeType    ActiveTime;
```

```
    ///挂起时间
    TThostFtdcTimeType    SuspendTime;
    ///最后修改时间
    TThostFtdcTimeType    UpdateTime;
    ///撤销时间
    TThostFtdcTimeType    CancelTime;
    ///最后修改交易所交易员代码
    TThostFtdcTraderIDType    ActiveTraderID;
    ///结算会员编号
    TThostFtdcParticipantIDType    ClearingPartID;
    ///序号
    TThostFtdcSequenceNoType SequenceNo;
    ///前置编号
    TThostFtdcFrontIDType FrontID;
    ///会话编号
    TThostFtdcSessionIDType    SessionID;
    ///用户端产品信息
    TThostFtdcProductInfoType    UserProductInfo;
    ///状态信息
    TThostFtdcErrorMsgType    StatusMsg;
};
```

4.2.23. OnRspQrySettlementInfoConfirm 方法

查询结算确认响应。由交易托管系统主动通知客户端，该方法会被调用。

函数原形：

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

参数：

pSettlementInfoConfirm：指向返回的结算确认信息结构。

结算确认结构：

```
///投资者结算结果确认信息
struct CThostFtdcSettlementInfoConfirmField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///确认日期
    TThostFtdcDateType    ConfirmDate;
```

```
    ///确认时间
    TThostFtdcTimeType    ConfirmTime;
};;
```

pRspInfo：指向响应信息结构的地址。

响应信息结构：

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

4.3. CThostFtdcTraderApi 接口

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

4.3.1. CreateFtdcTraderApi 方法

产生一个 CThostFtdcTradeApi 的一个实例，不能通过 new 来产生。

函数原形：

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

参数：

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

返回值：

返回一个指向 CThostFtdcTradeApi 实例的指针。

4.3.2. Release 方法

释放一个 CThostFtdcTradeApi 实例。不能使用 delete 方法

函数原形：

void Release();

4.3.3. Init 方法

使客户端开始与交易托管系统建立连接，连接成功后可以进行登陆。

函数原形：

void Init()；

4.3.4. GetTradingDay 方法

获得当前交易日。只有当与交易托管系统连接建立后才会取到正确的值。

函数原形：

const char *GetTradingDay()；

返回值：

返回一个指向日期信息字符串的常量指针。

4.3.5. RegisterSpi 方法

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

函数原形：

void RegisterSpi(CThostFtdcTraderSpi *pSpi) ;

参数：

pSpi：实现了 CThostFtdcTraderSpi 接口的实例指针。

4.3.6. RegisterFront 方法

设置交易托管系统的网络通讯地址，交易托管系统拥有多个通信地址，但用户只需要选择一个通信地址。

函数原形：

void RegisterFront(char *pszFrontAddress);

参数：

pszFrontAddress：指向后台服务器地址的指针。服务器地址的格式为：“protocol://ipaddress:port”，如：”tcp://127.0.0.1:17001”。 “tcp”代表传输协议，“127.0.0.1”代表服务器地址。”17001”代表服务器端口号。

4.3.7. SubscribePrivateTopic 方法

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

函数原形：

void SubscribePrivateTopic(TE_RESUME_TYPE nResumeType);

参数：

nResumeType： 私有流重传方式

TERT_RESTART:从本交易日开始重传

TERT_RESUME:从上次收到的续传

TERT_QUICK:只传送登录后私有流的内容

4.3.8. SubscribePublicTopic 方法

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

函数原形：

void SubscribePublicTopic(TE_RESUME_TYPE nResumeType);

参数：

nResumeType： 公共流重传方式

TERT_RESTART:从本交易日开始重传

TERT_RESUME:从上次收到的续传

TERT_QUICK:只传送登录后公共流的内容

4.3.9. ReqUserLogin 方法

用户发出登陆请求。

函数原形：

int ReqUserLogin(

CThostFtdcReqUserLoginField *pReqUserLoginField,

int nRequestID)；

参数：

pReqUserLoginField：指向用户登录请求结构的地址。

用户登录请求结构：

```
struct CThostFtdcReqUserLoginField
{
  ///交易日
  TThostFtdcDateType    TradingDay;
  ///经纪公司代码
  TThostFtdcBrokerIDType    BrokerID;
  ///用户代码
  TThostFtdcUserIDType UserID;
  ///密码
  TThostFtdcPasswordType    Password;
  ///用户端产品信息
  TThostFtdcProductInfoType    UserProductInfo;
  ///接口端产品信息
  TThostFtdcProductInfoType    InterfaceProductInfo;
  ///协议信息
  TThostFtdcProtocolInfoType ProtocolInfo;
};
```

nRequestID：用户登录请求的 ID，该 ID 由用户指定，管理。

用户需要填写 UserProductInfo 字段，即客户端的产品信息，如软件开发商、版本号等，例如：SFITTraderV100。

InterfaceProductInfo 和 ProtocolInfo 只须占位，不必有效赋值。

返回值：
0，代表成功。
-1，表示网络连接失败；
-2，表示未处理请求超过许可数；
-3，表示每秒发送请求数超过许可数。

4.3.10. ReqUserLogout 方法

用户发出登出请求。
函数原形：
int ReqUserLogout(
CThostFtdcUserLogoutField *pUserLogout,
int nRequestID)；
参数：
pReqUserLogout：指向用户登出请求结构的地址。
用户登出请求结构：
struct CThostFtdcUserLogoutField
{
///经纪公司代码
TThostFtdcBrokerIDType BrokerID;
///用户代码
TThostFtdcUserIDType UserID;
};
nRequestID：用户登出请求的 ID，该 ID 由用户指定，管理。
返回值：
0,代表成功。
-1，表示网络连接失败；
-2，表示未处理请求超过许可数；
-3，表示每秒发送请求数超过许可数。

4.3.11. ReqUserPasswordUpdate 方法

用户密码修改请求。
函数原形：
int ReqUserPasswordUpdate(
CThostFtdcUserPasswordUpdateField *pUserPasswordUpdate,
int nRequestID)；
参数：
pUserPasswordUpdate：指向用户口令修改结构的地址。
用户口令修改结构：
struct CThostFtdcUserPasswordUpdateField
{

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

nRequestID：用户操作请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

4.3.12. ReqOrderInsert 方法

客户端发出报单录入请求。

函数原形：

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

参数：

pInputOrder：指向输入报单结构的地址。

输入报单结构：

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

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

nRequestID：用户报单请求的 ID，该 ID 由用户指定，管理。在一次会话中，该 ID 不能重复。

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

因为交易后台按照字符串比较 OrdeRef 的大小，所以在设置 OrderRef 时要填满 TThostFtdcOrderRefType 的全部空间。

返回值：

0，代表成功；

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

4.3.13. ReqOrderAction 方法

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

函数原形：

int ReqOrderAction(
CThostFtdcOrderActionField *pOrderAction,
int nRequestID)；

参数：

pOrderAction：指向报单操作结构的地址。

报单操作结构：

```
struct CThostFtdcOrderActionField
{
  ///经纪公司代码
  TThostFtdcBrokerIDType  BrokerID;
  ///投资者代码
  TThostFtdcInvestorIDType  InvestorID;
  ///报单操作引用
  TThostFtdcOrderActionRefType  OrderActionRef;
  ///报单引用
  TThostFtdcOrderRefType  OrderRef;
  ///请求编号
  TThostFtdcRequestIDType  RequestID;
  ///前置编号
  TThostFtdcFrontIDType FrontID;
  ///会话编号
  TThostFtdcSessionIDType  SessionID;
  ///交易所代码
  TThostFtdcExchangeIDType ExchangeID;
  ///报单编号
  TThostFtdcOrderSysIDType  OrderSysID;
  ///操作标志
  TThostFtdcActionFlagType  ActionFlag;
  ///价格
  TThostFtdcPriceType  LimitPrice;
  ///数量变化
  TThostFtdcVolumeType VolumeChange;
  ///操作日期
  TThostFtdcDateType  ActionDate;
  ///操作时间
  TThostFtdcTimeType  ActionTime;
  ///交易所交易员代码
  TThostFtdcTraderIDType  TraderID;
  ///安装编号
```

```
    TThostFtdcInstallIDType    InstallID;
    ///本地报单编号
    TThostFtdcOrderLocalIDType    OrderLocalID;
    ///操作本地编号
    TThostFtdcOrderLocalIDType    ActionLocalID;
    ///会员代码
    TThostFtdcParticipantIDType    ParticipantID;
    ///客户代码
    TThostFtdcClientIDType    ClientID;
    ///业务单元
    TThostFtdcBusinessUnitType    BusinessUnit;
    ///报单操作状态
    TThostFtdcOrderActionStatusType    OrderActionStatus;
    ///用户代码
    TThostFtdcUserIDType UserID;
    ///状态信息
    TThostFtdcErrorMsgType    StatusMsg;
};
```

nRequestID：用户报单操作请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

4.3.14. ReqSettlementInfoConfirm 方法

投资者结算结果确认。

函数原形：

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

参数：

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

投资者结算结果确认结构：

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

```
    ///确认时间
    TThostFtdcTimeType    ConfirmTime;
};
```

nRequestID：用户报单操作请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

4.3.15. ReqQryOrder 方法

报单查询请求。

函数原形：

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

参数：

pQryOrder：指向报单查询结构的地址。

报单查询结构：

```
struct CThostFtdcQryOrderField
{
    ///经纪公司代码
    TThostFtdcBrokerIDType    BrokerID;
    ///投资者代码
    TThostFtdcInvestorIDType    InvestorID;
    ///合约代码
    TThostFtdcInstrumentIDType    InstrumentID;
    ///交易所代码
    TThostFtdcExchangeIDType ExchangeID;
    ///报单编号
    TThostFtdcOrderSysIDType    OrderSysID;
};
```

nRequestID：用户报单查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

注：不写 BrokerID 可以收全所有报单。

4.3.16. ReqQryTrade 方法

成交单查询请求。

函数原形：

int ReqQryTrade(
CThostFtdcQryTradeField *pQryTrade,
int nRequestID)；

参数：

pQryTrade：指向成交查询结构的地址。

成交查询结构：

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

nRequestID：用户成交单查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。
-1，表示网络连接失败；
-2，表示未处理请求超过许可数；
-3，表示每秒发送请求数超过许可数。

4.3.17. ReqQry Investor 方法

会员客户查询请求。

函数原形：

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

参数：

pQry Investor：指向客户查询结构的地址。

客户查询结构：

```
struct CThostFtdcQryInvestorField
{
  ///经纪公司代码
```

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

nRequestID：用户客户查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

4.3.18. ReqQryTradingAccount 方法

请求查询资金账户。

函数原形：

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

参数：

pQryTradingAccount：指向查询资金账户结构的地址。

查询资金账户结构：

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

nRequestID：会员持仓查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

4.3.19. ReqQryInstrument 方法

请求查询合约。

函数原形：

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

参数：

pQryInstrument：指向查询查询合约结构的地址。

查询合约结构：

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

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

4.3.20. ReqQrySettlementInfo 方法

请求查询投资者结算结果。

函数原形：

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

参数：

pQrySettlementInfo：指向查询投资者结算结果的地址。

查询投资者结算结果结构：

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

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；
-2，表示未处理请求超过许可数；
-3，表示每秒发送请求数超过许可数。

4.3.21. ReqQryInvestorPositionDetail 方法

请求查询投资者持仓明细。

函数原形：

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

参数：

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

查询投资者持仓明细结构：

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

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。
-1，表示网络连接失败；
-2，表示未处理请求超过许可数；
-3，表示每秒发送请求数超过许可数。

4.3.22. ReqQryNotice 方法

请求查询客户通知。

函数原形：

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

参数：

pQryNotice：指向查询客户通知结构的地址。

查询客户通知结构：

```
struct CThostFtdcQryNoticeField
{
```

```
  ///经纪公司代码
  TThostFtdcBrokerIDType  BrokerID;
};
```

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。

4.3.23. ReqQrySettlementInfoConfirm 方法

函数原形：

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

参数：

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

查询结算信息确认：

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

nRequestID：合约查询请求的 ID，该 ID 由用户指定，管理。

返回值：

0，代表成功。

-1，表示网络连接失败；

-2，表示未处理请求超过许可数；

-3，表示每秒发送请求数超过许可数。