CTP 接口开发

一、 CTP 介绍

- 背景
  - 国内程序化发展
    - 统计数据显示，目前国内期货市场使用全自动程序化交易的投资者数量占投资者总数的 1%左右
    - 借助程序化交易系统提示交易信号进行手动交易或进行半自动交易的投资者数量占投资者总量的 5%—10%
    - 而通过程序化交易完成的交易量占市场总交易量的 20%—30%。
  - 交易平台
    - 文华
    - 交易开拓者(TB)
    - MultiCharts (MC)
    - 金字塔
    - YesTrader
- 上期技术成长脚印
  - 2004 年
    - 公司成立,核心业务定位为:一、开发期货交易所 NGES 新一代交易系统;二、面向期货行业会员市场提供 IT 服务。
  - 2005 年
    - 面向会员端市场提供第一个产品——“共享灾备”服务。在上期所的支持下,上期技术为期货行业提供了低成本、经济型共享灾备系统,有效地弥补与解决了期货行业因技术保障滞后可能引发的交易安全与稳定性的问题。下半年即迎来第一个客户。
  - 2006 年
    - 开创国内交易托管业务的先河,全面快速提升全行业期货公司 IT 水平。1 月,上期技术推出新产品——主机托管服务,开启中国期货行业托管服务的先河。通过提供交易所等级的统一、标准机房、统一的基础运维服务,灵活多样的托管选项(如机柜租赁、机房租赁等),使期货公司 IT 建设标准上了一个台阶,有效保障了期货交易安全和投资者的利益。10 月,新一代交易系统 NGES 成功上线,自此开始,上期技术技术服务客户逐渐覆盖上期所、中金所、上证所、郑商所、行业监管机构等。
  - 2007 年
    - 推出创新产品——综合交易平台(CTP)应用托管解决方案。综合交易平台创新的“接口开放”模式,打破了行业接口封闭的发展壁垒,促成了期货行业基于 CTP 平台之上的终端厂商——上期技术与期货公司和投资者强强连手、利益趋同、合作共赢的发展新格局。12 月 28 日,CTP 第一个正式交易产生。
  - 2008 年
    - 主机托管临危受命,服务全行业应对百年雪灾:2008 年初,一场百年一遇的暴雪席卷华夏大地,为防止期货公司因雪灾造成交易系统的瘫痪,上期技术 24 小时待命,准备机房空间和应急设备,随时应期货公司要求,免费向受灾会员提供所需托管服务:24 小之内完成系统搭建,帮助期货公司数据库的倒录、转存,坚决扞卫行业交易的持续性、稳定性,安全性,免受雪灾的影响。
    - 8 月,综合交易平台第一家会员上海金鹏完成所有客户上线切换。当年底 CTP 完成 4 家会员上线。11 月,上期技术托管中心张江机房正式启用,第一家非期货行业用户深发

展入驻张江托管中心。

- 2009 年
  - 4 月,CTP 打开国内程序化交易最合适的 IT 工具大门,成为程序化交易投资者的利器。11 月,综合交易平台第一个独立部署主席会员——海通期货成功上线,由此综合交易平台由小众、小会员走向大众、大会员的市场。2009 年底即完成 10 家主席会员的上线,二席会员达到 20 家。托管中心使用 1 年时间完成所有会员到张江托管中心的搬迁。主机托管会员在享有较低费用支出的基础上,享有更优质的托管服务。10 月,托管中心正式推出针对 VIP 会员的机房托管模式。国金期货和东证期货作为首批机房托管用户正式入驻。
- 2010 年
  - 继往开来,合作共赢,共创辉煌:2010 年 6 月底,CTP 已完成 21 家应用托管客户的签约待上线。张江交易管托中心,托管服务已细分为机房托管、主机托管、零星托管、应用托管等诸多类型,会员市场从最初的小会员公司,发展到现在占有全行业 70%的托管市场,机房面积从 100 多平米,发展到目前的 2000 多平米。
- 2012 年
  - 7 月 30 日，中证期货 CTP 主席系统上线。
  - 中证 CTP 主席平台部署在上期技术张江机房，通过千兆局域网接入中金所和上期所交易系统。

■ 综合交易平台

- 做为一个开放、快速、稳定、安全的期货交易、结算系统解决方案，在期货界也获得了越来越普遍的认同。综合交易平台开放的接口、优异的性能、集中部署的创新模式以及经验丰富的技术背景都为程序化交易在国内的快速发展提供了最为优异的平台。综合交易平台现有的程序化交易客户对综合交易平台的解决方案给了很高的评价，其交易量也不断攀升。
- 综合交易平台 8000 笔/秒处理速度的交易引擎，整套系统在 0.5 毫秒以内处理完成报单、成交全过程的资金持仓计算的能力，无单点故障并实现负载均衡的交易系统体系架构构成了综合交易平台的高性能体验。
- 综合交易平台目前的系统配置拥有 2 万个客户同时在线的处理能力，还可以通过扩展前置机群进一步提升系统对更多客户在线的处理能力。
- 综合交易平台通过千兆局域网接入中金所和上期所交易系统，通过三所联网主干接入大商所和郑商所。 投资者在综合交易平台的报单直接进入综合交易平台的前置机，经过交易后台高速的资金持仓计算后再经局域网报到中金所和上期所，通过三所联网主干报到大商所和郑商所。行情服务器直连交易所并在同一个进程实现分发到行情前置，接收和分发完全在内存中完成，网络迟延也被压缩到了极点。 托管于上期技术的程序化交易终端，因为通过局域网接入综合交易平台，其报单和行情速度处于目前业内最快水平。

■ 期货交易数据交换协议(FTD)Futures trading data exchange protocol

- 2005 年证监会发布

二、 规则

■ 主席与二席

- 功能:二席没有银期转帐
- 性能:二席因没有日志所以响应要快些
- 收费:二席加收费用

■ 命名规则

- 请求指令:**Req*** ** 如 ReqUserLogin
  请求响应:On**Rsp*** ** 如 OnRspUserLogin

- 查询指令:**Req**Qry*** 如 ReqQryInstrument
  查询响应:On**Rsp**Qry*** 如 OnRspQryInstrument
- 回报响应:On**Rtn***** 如 OnRtnOrder
- 错误响应:On**ErrRtn***** 如 OnErrRtnOrderInsert

■ 通讯模式：
- 对话通讯模式：由客户端主动发起请求。Thost 收到请求、处理请求后，返回 1 条或者多条响应纪录。例如登入、各项查询、报单、撤单等操作。
- 私有通讯模式：由 Thost 主动向客户端发出的相关信息。例如委托回报、成交回报、错单回报等
- 广播通讯模式：由 Thost 主动向所有客户端发出的公共信息，例如行情等

■ 数据流重传方式
- 通常使用 Restart 模式较为方便
- 本地数据落地可用 Resume 模式
  - 程序使用 TradeApi 和 MdApi，并且把这 2 个 dll 放在同一个目录下。程序再次启动后，如果某个 api 采用 Resume 模式订阅公有流/私有流，就会去参考相关的本地流文件。可能会导致数据异常？
  - 答：相同目录下的 2 个 dll 会把数据写入相同本地流文件，导致 2 个 dll 不断的覆盖对方写下的流文件。程序再次启动时，TradeApi 可能去参考 MdApi 写下的流文件，所以导致数据流不连续。
  - 解决方法：如果一定要把 2 个 dll 放在相同的目录下，可以在创建 api 时指定流文件的路径。使得不同的 dll 写入不同流文件

```
//数据流重传方式
enum THOST_TE_RESUME_TYPE
{
    // 从当天的第一条记录开始接收数据流
    THOST_TERT_RESTART = 0,
    // 接收上次断线以后的数据流
    THOST_TERT_RESUME,
    // 接收本次登入以后的数据流
    THOST_TERT_QUICK
};
```

■ 交易接口流控
- 查询 1 笔/秒
- 指令(报单/撤单/查询):每客户每连接 6 笔/秒,超过部分将排队
- 同一帐户连接最大前置数:6 个

登录错误(60):综合交易平台：用户在线会话超出上限

■ Spi 与 Api
- Spi:响应函数,需继承并实现
- Api:指令函数

■ nRequestID
- 发送请求时需要设定 RequestID，TraderApi 返回响应时返回相关请求的 RequestID。因为 TraderApi 是异步实现的，终端程序可能连续发出多个请求和查询指令。RequestID 可以把请求/查询指令和相关的回报关联起来

■ IsLast
- 无论是否有查询响应数据，只要查询响应结束，IsLast 为 true

- 响应信息 RspInfo
  - 如果 RspInfo 为空，或者 RspInfo 的错误代码为 0，说明查询成功。
  - 否则 RspInfo 中会保存错误编码和错误信息。

```
bool IsErrorRspInfo(CThostFtdcRspInfoField* pRspInfo)
{
    // 如果ErrorID != 0, 说明收到了错误的响应
    bool bResult = ((pRspInfo) && (pRspInfo->ErrorID != 0));
    if (bResult)
        cerr << "--->>> ErrorID=" << pRspInfo->ErrorID
        << ", ErrorMsg=" << pRspInfo->ErrorMsg << endl;
    return bResult;
}
```

- 查询响应数据
  - 查询响应方法每次返回 1 条记录。如果没有查询结果，就返回**空指针**

三、 开发准备

- C++开发环境
  - 跨平台的 CodeBlocks
  - wxWidgets
  - Boost
- 继承 Spi
- 实现虚方法

四、 连接

- CreatApi
- RegisterSpi
- RegisterFront
  - 多前置注册
- (Trade)SubscribePrivateTopic
- (Trade)SubscribePublicTopic
- Init
- //Join
  - OnFrontConnected

```
void ReqConnect(char* f, const char* b)
{
    pUserApi = CThostFtdcTraderApi::CreateFtdcTraderApi("logTrade", false);
    pUserApi->RegisterSpi((CThostFtdcTraderSpi*)this);
    pUserApi->SubscribePublicTopic(THOST_TERT_RESTART);
    pUserApi->SubscribePrivateTopic(THOST_TERT_RESTART);

    strcpy(this->broker, b);

    pUserApi->RegisterFront(f);
    pUserApi->Init();
}
```

五、 断开

- Release
- OnFrontDisconnected

  0x1001 网络读失败 => 4097
  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

六、 登录

- ReqUserLogin
  - OnRspUserLogin
    - 非法登录
    - 未初始化
      - 隔夜自动重连

```
void Trade::OnRspUserLogin(CThostFtdcRspUserLoginField* pRspUserLogin, CThostFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
{
    if(IsErrorRspInfo(pRspInfo))
    {
        string msg = str(boost::format("登录错误(%1%):%2%") %pRspInfo->ErrorID % pRspInfo->ErrorMsg);
        show(msg);
    }
    else
    {
        show("确认结算...");
        CThostFtdcSettlementInfoConfirmField req;
        memset(&req, 0, sizeof(req));
        strcpy(req.BrokerID, broker);
        strcpy(req.InvestorID, user);
        pUserApi->ReqSettlementInfoConfirm(&req, ++iReqID);

        strcpy(tradingDay, pUserApi->GetTradingDay());
    }
}
```

七、 确认结算

- ReqSettlementInfoConfirm
  - OnRspSettlementInfoConfirm

```
void Trade::OnRspSettlementInfoConfirm(CThostFtdcSettlementInfoConfirmField* pSettlementInfoConfirm,
                                       CThostFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
{
    //cout << pSettlementInfoConfirm->ConfirmDate << endl;
    show("查合约...");
    CThostFtdcQryInstrumentField req;
    memset(&req, 0, sizeof(req));
    pUserApi->ReqQryInstrument(&req, ++iReqID);
}
```

- 相关(当日首次登录必须确认结算)
  - ReqQrySettlementInfoConfirm
    - OnRspQrySettlementInfoConfirm(是否当日结算)
  - ReqQrySettlementInfo(查结算信息)
    - OnRspQrySettlementInfo
  - ReqCFMMCTradingAccountKey(查保证金监控中心密钥)
    - “https://investorservice.cfmmc.com/loginByKey.do?companyID= “ + f.ParticipantID + "&userid=" + f.AccountID + "&keyid=" + f.KeyID + "&passwd=" + f.CurrentKey

八、 查合约

- ReqQryInstrument
  - OnRspQryInstrument
    - 套利合约: SPxxx&xx(大商所跨期)/SPC(大商所跨品种)/SPD(郑商所跨期)
- 此处可登录行情接口

```
void Trade::OnRspQryInstrument(CThostFtdcInstrumentField* pInstrument,
                               CThostFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
{
    if(!IsErrorRspInfo(pRspInfo))
    {
        //cout << pInstrument->InstrumentID << endl;
        int len = strlen(pInstrument->InstrumentID) + 1;
        char* tmp = new char[len];
        strcpy(tmp, pInstrument->InstrumentID);
        dicInstrument[string(tmp)] = *pInstrument;  //合约名称标识
    }
    if(bIsLast)
    {
        show("行情登录中...");
        if(this->addrID == -1)  //连接
        {
            char* tmp = new char[256];
            strcpy(tmp, string(this->frontAddr).substr(0, strlen(this->frontAddr)-2).append("13").c_str());
            quote->ReqConnect(tmp, this->broker);
        }
        else
            quote->ReqConnect(addrID);
    }
}
```

九、 登出

- ReqUserLogout
  - OnRspUserLogout

十、 行情

- SubscribeMarketData
  - 合约可重复订阅,对响应无影响
  - OnRspSubMarketData
    - 注册错误
  - OnRtnDepthMarketData
    - Tick 推送
    - 行情响应中处理延时,可能导致接口断开.
    - 日期字段为 NULL
    - 交易所字段为 NULL
    - 非法数据
      - LastPrice 为最大值
      - 单边挂单(封板)
    - 成交额与均价(CZCE)

```
void Quote::OnRspUserLogin(CThostFtdcRspUserLoginField* pRspUserLogin,
                           CThostFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
{
    show("行情登录成功, 订阅行情! ");

    char* tmp[1] = {new char[256]};
    BOOST_FOREACH(mapInstrument::value_type i, dicInstrument)
    {
        strcpy(tmp[0], i.first.c_str());
        pUserApi->SubscribeMarketData(tmp, 1);
    }
}
```

```
void Quote::OnRtnDepthMarketData(CThostFtdcDepthMarketDataField* pDepthMarketData)
{
    CThostFtdcDepthMarketDataField field;
    memset(&field, 0, sizeof(field));
    field = *pDepthMarketData;

    boost::thread tExecTick(&tickToFile, field);     //启线程
}
```

```
void tickToFile(CThostFtdcDepthMarketDataField f)
{
    try
    {
        //合法合约//合法数据
        if(dicInstrument.find(f.InstrumentID) != dicInstrument.end() && f.LastPrice < f.UpperLimitPrice)
        {
            CThostFtdcInstrumentField instField;
            memset(&instField, 0, sizeof(instField));
            instField = dicInstrument[f.InstrumentID];  //合约信息

            strcpy(f.TradingDay, tradingDay);           //日期
            strcpy(f.ExchangeID, instField.ExchangeID); //交易所

            if(f.AskPrice1 > f.UpperLimitPrice) //单边有挂边的情况
                f.AskPrice1 = f.LastPrice;
            if(f.BidPrice1 > f.UpperLimitPrice)
                f.BidPrice1 = f.LastPrice;

            if(instField.ExchangeID == "CZCE") //成交额与均价
                f.Turnover *= instField.VolumeMultiple;
            else
                f.AveragePrice /= instField.VolumeMultiple;

            dicTick[f.InstrumentID] = f;     //更新Tick
```

十一、 报单

- 标识
  - FrontID + SessionID + OrderRef
    - OrderRef(int atoi 注意长度)
  - BrokerID + BrokerOrderSeq
  - ExchangeID + OrderSysID
- ReqOrderInsert
  - OnRspOrderInsert
    - Thost 收到报单指令，如果没有通过参数校验，拒绝接受报单指令
  - OnErrRtnOrderInsert
    - 交易所收到报单后认为报单错误
  - OnRtnOrder
    - 报单委托状态

      ///TFtdcOrderStatusType 是一个报单状态类型

      ///全部成交

      #define THOST_FTDC_OST_AllTraded '0'

      ///部分成交还在队列中

      #define THOST_FTDC_OST_PartTradedQueueing '1'

      ///部分成交不在队列中

      #define THOST_FTDC_OST_PartTradedNotQueueing '2'

      ///未成交还在队列中

      #define THOST_FTDC_OST_NoTradeQueueing '3'

      ///未成交不在队列中

      #define THOST_FTDC_OST_NoTradeNotQueueing '4'

      ///撤单

      #define THOST_FTDC_OST_Canceled '5'

      ///未知

      #define THOST_FTDC_OST_Unknown 'a'

      ///尚未触发

      #define THOST_FTDC_OST_NotTouched 'b'

- 多次响应对应
  - FrontID + SessionID + OrderRef
  - BrokerID + BrokerOrderSeq
  - ExchangeID + OrderSysID
- OnRtnTrade
  - 与 OnRtnOrder 对应
    - BrokerID + BrokerOrderSeq
    - ExchangeID + OrderSysID
  - 成交价格: Price
  - 成交时间:TradeTime

■ 限价单

```
//报单-限价
void OrderInsert(const char* instrument, double price, int director, int offset, int volume)
{
    CThostFtdcInputOrderField f;
    memset(&f, 0, sizeof(f));
    f.CombHedgeFlag[0] = THOST_FTDC_HF_Speculation; //1投机
    f.ContingentCondition = THOST_FTDC_CC_Immediately;//立即触发
    f.ForceCloseReason = THOST_FTDC_FCC_NotForceClose;
    f.IsAutoSuspend = 0;
    f.OrderPriceType = THOST_FTDC_OPT_LimitPrice;       //***任意价1  限价2***
    f.TimeCondition =  THOST_FTDC_TC_GFD;       //***立即完成1  当日有效3***
    f.VolumeCondition = THOST_FTDC_VC_AV;   //任意数量1  最小数量2  全部成交3
    f.MinVolume = 1;

    strcpy(f.InvestorID, trade->investor);
    strcpy(f.UserID, trade->investor);
    strcpy(f.BrokerID, trade->broker);

    strcpy(f.InstrumentID, instrument); //合约

    f.LimitPrice = price;               //***价格***

    if(director == 0)
        f.Direction = THOST_FTDC_D_Buy;         //买
    else
        f.Direction = THOST_FTDC_D_Sell;            //卖

    if(offset == 0)
        f.CombOffsetFlag[0] = THOST_FTDC_OF_Open;//开仓
    else if(offset == 1)
        f.CombOffsetFlag[0] = THOST_FTDC_OF_Close;  //平仓
    else
        f.CombOffsetFlag[0] = THOST_FTDC_OF_CloseToday; //平今

    f.VolumeTotalOriginal = volume;//数量

    sprintf(f.OrderRef, "%d", ++orderRef);//OrderRef

    ptime = boost::posix_time::microsec_clock::local_time();    //计时
    show(str(boost::format("报单(编号%1%).....") %f.OrderRef), 1);

    trade->pUserApi->ReqOrderInsert(&f, trade->iReqID++);   //报单
}
```

■ 市价单

- 任意价格
- 价格为 0

```
//报单-市价
void ReqOrderInsert(const char* instrument, int director, int offset, int volume)
{
    CThostFtdcInputOrderField f;
    memset(&f, 0, sizeof(f));
    f.CombHedgeFlag[0] = THOST_FTDC_HF_Speculation; //1投机
    f.ContingentCondition = THOST_FTDC_CC_Immediately;//立即触发
    f.ForceCloseReason = THOST_FTDC_FCC_NotForceClose;
    f.IsAutoSuspend = 0;
    f.OrderPriceType = THOST_FTDC_OPT_AnyPrice;     //***任意价1  限价2***
    f.TimeCondition =  THOST_FTDC_TC_IOC;        //***立即完成1  当日有效3***
    f.VolumeCondition = THOST_FTDC_VC_AV;    //任意数量1  最小数量2  全部成交3
    f.MinVolume = 1;

    strcpy(f.InvestorID, trade->investor);
    strcpy(f.UserID, trade->investor);
    strcpy(f.BrokerID, trade->broker);

    strcpy(f.InstrumentID, instrument); //合约

    f.LimitPrice = 0;                   //***价格***

    if(director == 0)
        f.Direction = THOST_FTDC_D_Buy;         //买
    else
        f.Direction = THOST_FTDC_D_Sell;            //卖
```

- 触发单

  ///触发条件：用户设定

  ContingentCondition = ……;

  ///止损价：用户设定

  StopPrice = ……;

  /// 报单价格条件类型：限价

  OrderPriceType = THOST_FTDC_OPT_LimitPrice;

  /// 价格：用户设定

  LimitPrice = ……;

  /// 有效期类型类型：当日有效

  TimeCondition = THOST_FTDC_TC_GFD;

- 关于平仓
  - 上期所区分昨仓和今仓。
    - 平昨仓时，开平标志类型设置为平仓 THOST_FTDC_OF_Close
    - 平今仓时，开平标志类型设置为平今仓 THOST_FTDC_OF_CloseToday
  - 其他交易所不区分昨仓和今仓。
    - 开平标志类型统一设置为平仓 THOST_FTDC_OF_Close

十二、 撤单

- ReqOrderAction
  - OnRspOrderAction
    - Thost 收到撤单指令，如果没有通过参数校验，拒绝接受撤单指令
  - OnErrRtnOrderAction
    - 如果交易所认为报单错误
  - OnRtnOrder
    - 报单状态 THOST_FTDC_OST_Canceled
- 标识
  - FrontID + SessionID + OrderRef
  - **ExchangeID + OrderSysID(推荐)**

```
//撤单
void ReqOrderAction(const char* instrument, int session, int frontid, const char* orderref)
{
    CThostFtdcInputOrderActionField f;
    memset(&f, 0, sizeof(f));
    f.ActionFlag = THOST_FTDC_AF_Delete;
    strcpy(f.BrokerID, trade->broker);
    strcpy(f.InvestorID, trade->investor);

    strcpy(f.InstrumentID, instrument);
    f.SessionID = session;
    f.FrontID = frontid;
    strcpy(f.OrderRef, orderref);
    trade->pUserApi->ReqOrderAction(&f, ++trade->iReqID);
}
```

十三、 查持仓

- ReqQryInvestorPosition
  - OnRspQryInvestorPosition
    - 合约+多空+今昨

- 持仓为 0
  - 报单未成交
  - 持仓全部平掉

```
void Trade::OnRspQryInvestorPosition(CThostFtdcInvestorPositionField* pInvestorPosition,
                                     CThostFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
{
    string positionID = str(boost::format("%1%|%2%|%3%") %pInvestorPosition->InstrumentID
                            %pInvestorPosition->PosiDirection %pInvestorPosition->PositionDate);
    dicPosition[positionID] = *pInvestorPosition;

    if(bIsLast)
    {
        string msg = str(boost::format("\n%|1$+12|%|2$+12|%|3$+12|%|4$+12|%|5$+12|%|6$+12|\n")
                         %"合约" %"买卖" %"今昨" %"持仓" %"成本" %"持仓盈亏");
        BOOST_FOREACH(mapPosition::value_type i, dicPosition)
        {
            CThostFtdcInvestorPositionField f = i.second;
            msg += str(boost::format("%|1$+11|%|2$+11|%|3$+11|%|4$+11|%|5$+11|%|6$+11|\n")
                       %f.InstrumentID %(f.PosiDirection == THOST_FTDC_PD_Long?"多":"空")
                       %(f.PositionDate == THOST_FTDC_PSD_History?"昨仓":"今仓") %f.Position
                       %(f.PositionCost / dicInstrument[string(pInvestorPosition->InstrumentID)].VolumeMultiple)
                       %f.PositionProfit);
        }
        show(msg, 1);
    }
}
```

- ReqQryInvestorPositionDetail (推荐)
  - OnRspQryInvestorPositionDetail
  - 合成为 InvestorPositionField

```
            CThostFtdcInvestorPositionField pf;
            if(iter == dicPosition.end())
            {
                memset(&pf, 0, sizeof(pf
                strcpy(pf.InstrumentID, f.InstrumentID);
                pf.PosiDirection = (f.Direction == THOST_FTDC_D_Buy ? THOST_FTDC_PD_Long: THOST_FTDC_PD_Short);
                pf.PositionDate = (strcmp(f.OpenDate, tradingDay) == 0? THOST_FTDC_PSD_Today : THOST_FTDC_PSD_History);
                pf.Position = 0;//f.Volume;
                pf.PositionCost = 0;//f.OpenPrice;
                pf.PositionProfit = 0;//f.PositionProfitByDate
                pf.TodayPosition = 0
                pf.YdPosition = 0;

            }
            else
            {
                pf = dicPosition[positionID];
            }

            pf.Position  += f.Volume;
            if(pf.PositionDate == THOST_FTDC_PSD_History)
            {
                pf.YdPosition += f.Volume;
                pf.PositionCost += f.Volume * f.LastSettlementPrice * multi;
                //持仓盈亏需要自己运算detail始终为0
                pf.PositionProfit += (f.Direction == THOST_FTDC_PD_Long? 1 : -1) *
                                     (dicTick[f.InstrumentID].LastPrice - f.LastSettlementPrice) * f.Volume * multi;
            }
```

```
            else
            {
                pf.TodayPosition += f.Volume;
                pf.PositionCost += f.Volume * f.OpenPrice * multi;
                //持仓盈亏需要自己运算detail始终为0
                pf.PositionProfit += (f.Direction == THOST_FTDC_PD_Long? 1 : -1) *
                                     (dicTick[f.InstrumentID].LastPrice - f.OpenPrice) * f.Volume * multi;
            }

            dicPosition[positionID] = pf;
        }
        show(msg, 1);

        //显示持仓汇总
        msg = str(boost::format("\n%|1$+12|%|2$+12|%|3$+12|%|4$+12|%|5$+12|%|6$+12|\n")
                  %"合约" %"买卖" %"今昨" %"持仓" %"成本" %"持仓盈亏");
        BOOST_FOREACH(mapPosition::value_type i, dicPosition)
        {
            CThostFtdcInvestorPositionField f = i.second;
            msg += str(boost::format("%|1$+11|%|2$+11|%|3$+11|%|4$+11|%|5$+11|%|6$+11|\n")
                       %f.InstrumentID %(f.PosiDirection == THOST_FTDC_PD_Long?"多":"空")
                       %(f.PositionDate == THOST_FTDC_PSD_History?"昨仓":"今仓") %f.Position
                       %(f.PositionCost / f.Position / dicInstrument[string(f.InstrumentID)].VolumeMultiple) %f.PositionProfit);
        }
        show(msg, 1);
    }
}
```

十四、 查资金

- ReqQryTradingAccount
  - OnRspQryTradingAccount
    - 静态权益=上日结算-出金金额+入金金额
      account.PreBalance - account.Withdraw + account.Deposit
    - 动态权益=静态权益+ 平仓盈亏+ 持仓盈亏- 手续费
      静态权益+ account.CloseProfit + account.PositionProfit - account.Commission
    - 可用资金(动态权益-占用保证金- 冻结保证金- 冻结手续费- 交割保证金)
      account.Available

```
//查资金
void Trade::OnRspQryTradingAccount(CThostFtdcTradingAccountField* pTradingAccount,
                                   CThostFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
{
    if(bIsLast)
    {
        //静态权益=上日结算-出金金额+入金金额
        double preBalance = pTradingAccount->PreBalance - pTradingAccount->Withdraw + pTradingAccount->Deposit;
        //动态权益=静态权益+ 平仓盈亏+ 持仓盈亏- 手续费
        double current = preBalance
                         + pTradingAccount->CloseProfit + pTradingAccount->PositionProfit - pTradingAccount->Commission;
        string msg = str(boost::format("\n%|1$15|%|2$15|%|3$15|%|4$15|%|5$15|%|6$15|%|7$15|%|8$15|\n")
                         %"可用资金" %"平仓盈亏" %"持仓盈亏" %"静态权益" %"动态权益" %"占用保证金" %"冻结资金" %"风险度");
        msg += str(boost::format("%|1$14.2f|%|2$14.2f|%|3$14.2f|%|4$14.2f|%|5$14.2f|%|6$14.2f|%|7$14.2f|%|8$14.2f|%%")
                   %pTradingAccount->Available %pTradingAccount->CloseProfit %pTradingAccount->PositionProfit
                   %preBalance %current %pTradingAccount->CurrMargin
                   %(pTradingAccount->FrozenMargin + pTradingAccount->FrozenCommission)
                   %(pTradingAccount->CurrMargin / current * 100));
        show(msg, 1);
    }
}
```

十五、 查手续费

- ReqQryInstrumentCommissionRate
  - OnRspQryInstrumentCommissionRate
    - 按合约查询,以品种响应

十六、 查保证金

- ReqQryInstrumentMarginRate
  - OnRspQryInstrumentMarginRate
    - 只能按合约查询

十七、 银期转帐

- ReqQryAccountregister(查签约银行)
  - OnRspQryAccountregister
    - 银行卡号

```
//签约银行
void Trade::OnRspQryAccountregister(CThostFtdcAccountregisterField* pAccountregister,
                                    CThostFtdcRspInfoField* pRspInfo, int nRequestID, bool bIsLast)
{
    if(IsErrorRspInfo(pRspInfo))
        show(str(boost::format("查签约银行错误:%1%") %pRspInfo->ErrorMsg));
    else
    {
        string bank = "";
        if(pAccountregister->BankID[0] == THOST_FTDC_BF_ABC)
            bank = "农业银行";
        else if(pAccountregister->BankID[0] == THOST_FTDC_BF_BC)
            bank = "中国银行";
        else if(pAccountregister->BankID[0] == THOST_FTDC_BF_BOC)
            bank = "交通银行";
        else if(pAccountregister->BankID[0] == THOST_FTDC_BF_CBC)
            bank = "建设银行";
        else if(pAccountregister->BankID[0] == THOST_FTDC_BF_ICBC)
            bank = "工商银行";
        else if(pAccountregister->BankID[0] == THOST_FTDC_BF_Other)
            bank = "其他银行";

        string bankID = string(pAccountregister->BankAccount);
        bankID = bankID.substr(strlen(pAccountregister->BankAccount)-4, 4);
        show(str(boost::format("%1%.%2%(****%3%)") %pAccountregister->BankID[0] % bank
                 %bankID), 3));
    }
}
```

- ReqFromBankToFutureByFuture(银转期)
  - OnRspFromBankToFutureByFuture
    - 错误响应
  - OnRtnFromBankToFutureByFuture
    - 成功
- ReqFromFutureToBankByFuture(期转银)
  - OnRspFromFutureToBankByFuture
    - 错误响应
  - OnRtnFromFutureToBankByFuture
    - 成功
- 要点
  - 涉及农行/中行的指令需要输入银行密码
  - BankBranchID=”0000”

```
//银转功能
void ReqTransferByFuture(const char* bankID, const char* bankPWD, const char* accountPWD, double amount, bool f2B)
{
    CThostFtdcReqTransferField f;
    memset(&f, 0, sizeof(f));
                    class CThostFtdcReqTransferField {...}
    strcpy(f.BankID, bankID);
    strcpy(f.BankBranchID, "0000");      //必须有
    strcpy(f.BankPassWord, bankPWD);

    strcpy(f.BrokerID, trade->broker);
    strcpy(f.AccountID, trade->investor);
    strcpy(f.Password, accountPWD);
    f.SecuPwdFlag = THOST_FTDC_BPWDF_BlankCheck;     //明文核对
    strcpy(f.CurrencyID, "RMB");                     //币种: RMB-人民币 USD-美圆 HKD-港元

    f.TradeAmount = amount;
    if(f2B) //期转银
        trade->pUserApi->ReqFromFutureToBankByFuture(&f, ++trade->iReqID);
    else    //银转期
        trade->pUserApi->ReqFromBankToFutureByFuture(&f, ++trade->iReqID);
}
```

十八、 QA

Q:错误提示

```
./src/t1.0: In function `main':
ti.cpp:(.text+0x8b): undefined reference to `CThostFtdcMdApi::CreateFtdcMdApi(char const*, bool)'
collect2: ld 返回 1
```

A:在 windows 环境中,要使用 VC++编译器,否则会出现上面的提示.

Q:生产环境中遇到报单响应 OnRtnorder 在 OnRtnTrade 之前响应的情况.

A:

Q:OrderRef 报单失败

A:大商所与郑商所在报单未到达交易所时,用 OrderRef 撤单会失败.

Q:新版接口(20120530)中 CThostFtdcInputOrderField 中的 IsSwapOrder 是什么?

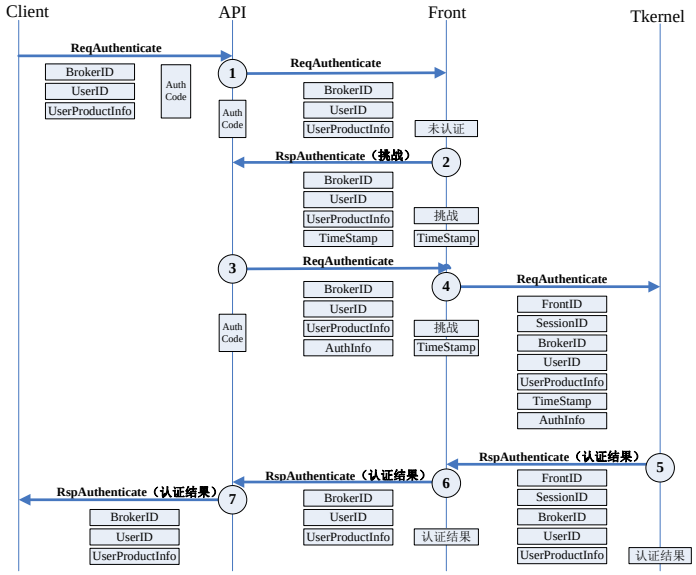
A:

Q:中金所套利编码

A: OrderInsert 中的 CombHedgeFlag 字段设置为:THOST_FTDC_HF_Arbitrage

Q:客户端认证

A:认证过程



十九、 附录

- 官方接口及文档 http://pan.baidu.com/share/link?shareid=31291&uk=3959766851
- Tick 数据 http://pan.baidu.com/share/link?shareid=8031&uk=3959766851
- 开发分享 http://pan.baidu.com/share/link?shareid=45562&uk=3959766851
  - C#封装源码
  - CTP 示例(C++)

二十、 预告

交易终端编程、K 线生成与入库、策略平台架构