中证期货技术创新部系列培训之二

——CTP 接口封装及交易终端开发

- 开发中常遇到的问题总结
  - 接口无任何响应
    - CTP 平台关闭
    - 网络不正常(系统设置/防火墙)
  - 无法连接
    - 前置地址错误，行情与交易前置搞混。
    - 前置地址前没有加 tcp://
  - 登录失败
    - Brokerid 不正确
    - 帐号密码错误
    - 未开通交易权限
  - 查询无响应
    - 查询流控 1 笔/秒
  - 交易无响应
    - 处理错误响应
      1. OnRspOrderInsert
      2. OnErrRtnOrderInsert

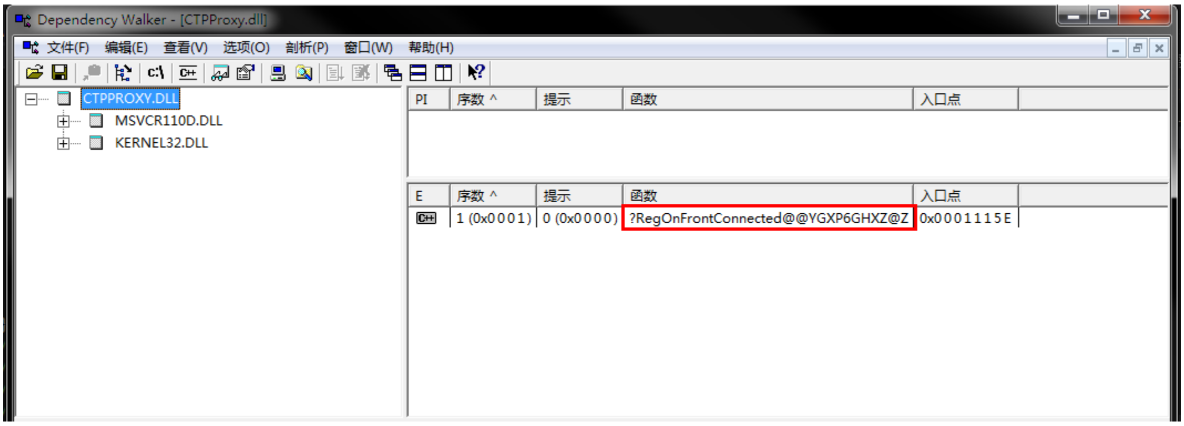
- 接口封装
  - 目的
    - 将 CTP 官方接口中的函数导出
    - 注册响应函数并被调用
  - 封装方式
    - 将指令和回调 Export
    - C++/CLI(略)
  - 实现
    - 创建 dll 项目

✧ Spi 封装(以行情为例)

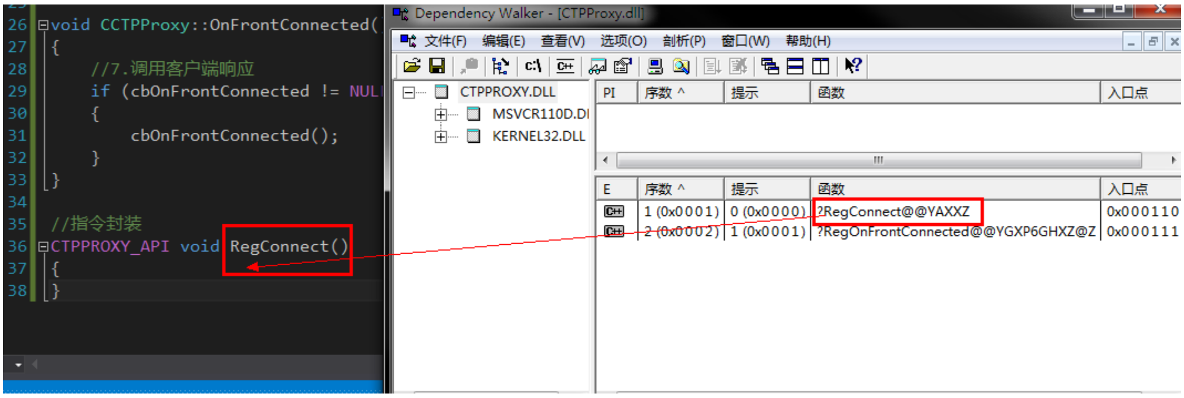
```
//1.引入.h
#include "../_CTPapi_20120530/20120530tradeapi_windows/ThostFtdcMdApi.h"

// 此类是从 CTPProxy.dll 导出的  //2.继承Spi
class CTPPROXY_API CCTPProxy : CThostFtdcMdSpi {
public:
    CCTPProxy(void);
    // TODO: 在此添加您的方法。

    //3.实现Spi函数(拷贝过来之后,替换 "{};" 为 ";"
    ///当客户端与交易后台建立起通信连接时（还未登录前），该方法被调用。
    virtual void OnFrontConnected();
```

```
//4.定义响应类型
typedef int (WINAPI *CBOnFrontConnected)(void);

//5.定义响应变量
CBOnFrontConnected cbOnFrontConnected = 0;

// 这是已导出类的构造函数。
// 有关类定义的信息，请参阅 CTPProxy.h
CCTPProxy::CCTPProxy()
{
    return;
}

//6.注册客户端响应
CTPPROXY_API void WINAPI RegOnFrontConnected(CBOnFrontConnected cb)
{
    cbOnFrontConnected = cb;
}

void CCTPProxy::OnFrontConnected()
{
    //7.调用客户端响应
    if (cbOnFrontConnected != NULL)
    {
        cbOnFrontConnected();
    }
}
```

✧ 生成的 dll 中对应的函数

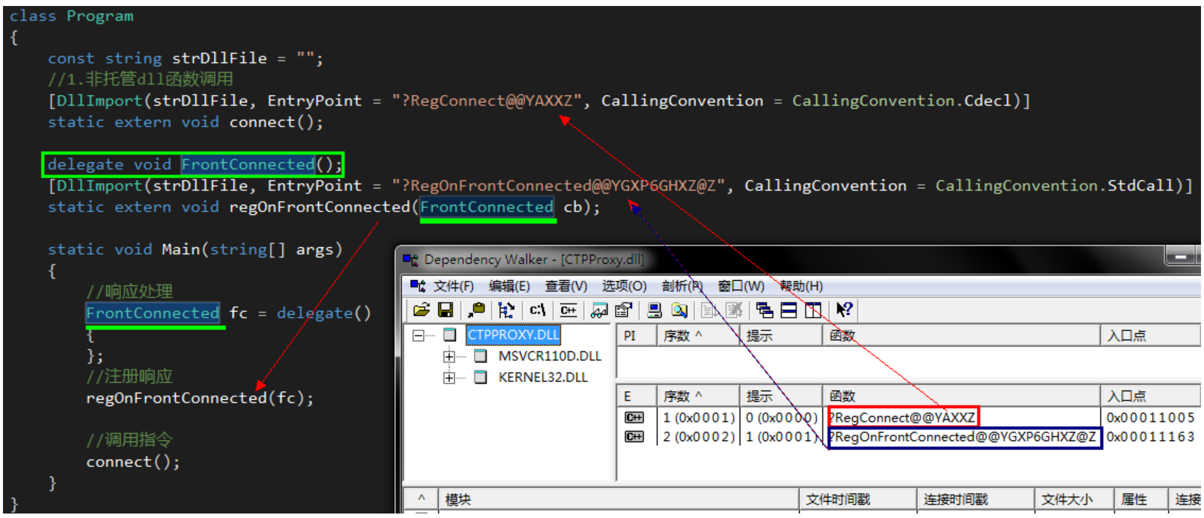


✧ 接口指令封装



✧ C#中调用

1. DllImport
2. 声明 delegate 类型(与 C++中的 typedef 对应)
3. 创建 delegate 变量,并注册
4. 指令调用



✧ struct 封装

1. char->char
2. char[]->string
3. int ->int
4. double->double
5. Offset & hedge 只使用 char[0]

```
/////////////////////////////////////////////////////////////////////////
///TFtdcCombOffsetFlagType是一个组合开平标志类型
/////////////////////////////////////////////////////////////////////////
typedef char TThostFtdcCombOffsetFlagType[5];

/////////////////////////////////////////////////////////////////////////
///TFtdcCombHedgeFlagType是一个组合投机套保标志类型
/////////////////////////////////////////////////////////////////////////
typedef char TThostFtdcCombHedgeFlagType[5];
```

6. 对应示例

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

```
/// <summary>
/// 用户登录应答
/// </summary>
[StructLayout(LayoutKind.Sequential)]
public struct CThostFtdcRspUserLoginField
{
    /// <summary>
    /// 交易日
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 9)]
    public string TradingDay;
    /// <summary>
    /// 登录成功时间
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 9)]
    public string LoginTime;
    /// <summary>
    /// 经纪公司代码
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 11)]
    public string BrokerID;
    /// <summary>
    /// 用户代码
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 16)]
    public string UserID;
    /// <summary>
    /// 交易系统名称
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 41)]
    public string SystemName;
    /// <summary>
    /// 前置编号
    /// </summary>
    public int FrontID;
    /// <summary>
    /// 会话编号
    /// </summary>
    public int SessionID;
    /// <summary>
    /// 最大报单引用
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 13)]
    public string MaxOrderRef;
    /// <summary>
    /// 上期所时间
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 9)]
    public string SHFETime;
    /// <summary>
    /// 大商所时间
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 9)]
    public string DCETime;
    /// <summary>
    /// 郑商所时间
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 9)]
    public string CZCETime;
    /// <summary>
    /// 中金所时间
    /// </summary>
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 9)]
    public string FFEXTime;
}
```

- 程序示例(连接/登录)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;   //Dllimport

namespace CTPProxyTest
{
    class Program
    {
        const string strDllFile = @"../../../CTPProxy/Debug/CTPProxy.dll";
        //1.非托管dll函数调用
        [DllImport(strDllFile, EntryPoint = "?ReqConnect@@YAXXZ", CallingConvention = CallingConvention.Cdecl)]
        static extern void ReqConnect();
        [DllImport(strDllFile, EntryPoint = "?ReqUserLogin@@YAXPAUCThostFtdcReqUserLoginField@@@Z", CallingConvention = CallingConvention.C
        static extern void ReqUserLogin(ref CThostFtdcReqUserLoginField f);

        delegate void FrontConnected();
        [DllImport(strDllFile, EntryPoint = "?RegOnFrontConnected@@YGXP6GHXZ@Z", CallingConvention = CallingConvention.StdCall)]
        static extern void regOnFrontConnected(FrontConnected cb);

        delegate void RspUserLogin(ref CThostFtdcRspUserLoginField pRspUserLogin, ref CThostFtdcRspInfoField pRspInfo, int nRequestID, bool
        [DllImport(strDllFile, EntryPoint = "?RegOnRspUserLogin@@YGXP6GHPAUCThostFtdcRspUserLoginField@@PAUCThostFtdcRspInfoField@@H_N@Z@Z"
        static extern void regOnRspUserLogin(RspUserLogin cb);
      static void Main(string[] args)
      {
          //响应处理
          FrontConnected fc = delegate()
          {
              Console.WriteLine("FrontConnected.");

              CThostFtdcReqUserLoginField f = new CThostFtdcReqUserLoginField();
              f.BrokerID = "2030";
              f.UserID = "879487";
              f.Password = "879487";
              ReqUserLogin(ref f);
          };
          RspUserLogin ul = new RspUserLogin(OnRspUserLogin);

          //注册响应
          regOnFrontConnected(fc);
          regOnRspUserLogin(ul);

          //调用指令
          ReqConnect();

          Console.Read();
      }

    static void OnRspUserLogin(ref CThostFtdcRspUserLoginField pRspUserLogin, ref CThostFtdcRspInfoField pRspInfo, int nRequestID, bool
    {
        Console.WriteLine(pRspInfo.ErrorMsg);
    }
}
```

- 交易终端

  - 新的封装:共 8 个函数

E	序数	提示 ^	函数	入口点
C++	7 (0x0007)	0 (0x0000)	?CreateFtdcTraderApi@@YGPAX_N@Z	0x000112E9
C	3 (0x0003)	1 (0x0001)	CreateFtdcQuoteApi	0x0001123A
C	6 (0x0006)	2 (0x0002)	CreateFtdcQuoteSpi	0x00011514
C	4 (0x0004)	3 (0x0003)	CreateFtdcTraderSpi	0x0001131B
C	5 (0x0005)	4 (0x0004)	RegCallBack	0x000113D9
C	8 (0x0008)	5 (0x0005)	RegCallBackQuote	0x000114DD
C	2 (0x0002)	6 (0x0006)	ReqCmd	0x000111D6
C	9 (0x0009)	7 (0x0007)	ReqCmdQuote	0x000110A0

  - 登录
    - Brokerid 错误
    - 帐号密码不对
    - 未初始化

- 登录:交易所时间

```
#region 取各交易所时间差
//初始化后,SHFETime字段可能为--:--:--;可考虑按登录失败处理********************************
this.tsLocal_Shfe = DateTime.Now.TimeOfDay - TimeSpan.Parse(pRspUserLogin.SHFETime);

if (TimeSpan.TryParse(pRspUserLogin.CZCETime, out tsLocal_Czce))
{
    this.tsLocal_Czce = DateTime.Now.TimeOfDay - this.tsLocal_Czce;
}
else
{
    this.tsLocal_Czce = this.tsLocal_Shfe;
}

if (TimeSpan.TryParse(pRspUserLogin.DCETime, out this.tsLocal_Dce))
{
    this.tsLocal_Dce = DateTime.Now.TimeOfDay - this.tsLocal_Dce;
}
else
{
    this.tsLocal_Dce = this.tsLocal_Shfe;
}

if (TimeSpan.TryParse(pRspUserLogin.FFEXTime, out this.tsLocal_Cffex))
{
    this.tsLocal_Cffex = DateTime.Now.TimeOfDay - this.tsLocal_Cffex;
}
else
{
    this.tsLocal_Cffex = this.tsLocal_Shfe;
}
#endregion
```

```
this.labelCFFEX.Text = (DateTime.Now.TimeOfDay - proxy.tsLocal_Cffex).ToString();
this.labelCZCE.Text = (DateTime.Now.TimeOfDay - proxy.tsLocal_Czce).ToString();
this.labelDCE.Text = (DateTime.Now.TimeOfDay - proxy.tsLocal_Dce).ToString();
this.labelSHFE.Text = (DateTime.Now.TimeOfDay - proxy.tsLocal_Shfe).ToString();
```

- 查询流控处理
  - 声明查询序列

```
ConcurrentStack<Tuple<EnumReqCmdType, object>> StackQuery = new ConcurrentStack<Tuple<EnumReqCmdType, object>>();
```

  - 将查询内容放入查询序列中

```
this.StackQuery.Push(new Tuple<EnumReqCmdType, object>(EnumReqCmdType.ReqQrySettlementInfoConfirm, f));
```

  - 线程中处理查询队列

```
//处理查询队列
void execQueryStack()
{
    while (isRunning)
    {
    Repeat:
        Tuple<EnumReqCmdType, object> query;
        if (this.StackQuery.TryPop(out query))
        {
            switch (query.Item1)
            {
                手续费 & 保证金
                default:
                    this.ReqCmd((EnumReqCmdType)query.Item1, query.Item2);
                    break;
            }
        }
        Thread.Sleep(1000);
    }
}
```

- 实时计算权益
  - 查询持仓明细响应中的处理
    1. 持仓明细数据存放队列
    2. 将所有得到的明细数据(格式自定义)合成为持仓数据

```csharp
void RspQryInvestorPositionDetail(ref CThostFtdcInvestorPositionDetailField pInvestorPositionDetail, ref  CThostFtdcRspInfoField p
{
    CThostFtdcInvestorPositionDetailField field = pInvestorPositionDetail;
    if (!string.IsNullOrEmpty(field.TradeID))
    {
        this.ListInvesterPositionDetail.Add(field.OpenDate + field.TradeID.Replace(' ', '0'), field);
    }
    if (bIsLast)
    {
        //将明细合成持仓
        DicInvesterPosition.Clear();
        foreach (var detail in ListInvesterPositionDetail.Values)
        {
            string key = detail.InstrumentID + "_" + (detail.Direction == Constants.THOST_FTDC_D_Buy ? "Buy" : "Sell");
            PositionField f = DicInvesterPosition.GetOrAdd(key, new PositionField());
            f.InstrumentID = detail.InstrumentID;
            f.Direction = detail.Direction;

            int multy = DicInstrumentField[f.InstrumentID].VolumeMultiple;
            f.PreSettlementPrice = detail.LastSettlementPrice;
            f.Position += detail.Volume;
            if (detail.OpenDate == this.tradingDay)  //今仓
            {
                f.TdPosition += detail.Volume;
                f.TdPositionCost += detail.Volume * detail.OpenPrice;
                f.TdUseMargin += detail.Margin;
            }
            else
            {
                f.YdPosition += detail.Volume;
                //持仓成本
                f.YdPositionCost += detail.Volume * detail.LastSettlementPrice;
                f.YdUseMargin += detail.Margin;
            }

            f.HedgeFlag = detail.HedgeFlag;
            //平仓盈亏
            f.CloseProfitByDate += detail.CloseProfitByDate;
            //持仓盈亏
            f.PositionProfit += detail.PositionProfitByDate;
            DicInvesterPosition[key] = f;
        }
        show(EnumCallBackType.OnRspQryInvestorPositionDetail);
    }
}
```

  - 行情中刷新持仓与资金权益

```csharp
void execTick(object pTick)
{
    CThostFtdcDepthMarketDataField f = (CThostFtdcDepthMarketDataField)pTick;
    数据过滤
    #region 刷新权益
    bool isNeedFresh = false;
    for (int i = 0; i < DicInvesterPosition.Count; ++i)
    {
        var v = DicInvesterPosition.ElementAt(i);
        if (v.Value.InstrumentID == f.InstrumentID)
        {
            PositionField pf = v.Value;
            if (v.Value.Position == 0)
                pf.PositionProfit = 0;
            else
                pf.PositionProfit = (v.Value.Direction == Constants.THOST_FTDC_D_Buy ? 1 : -1)
                    * (f.LastPrice - (pf.TdPositionCost + pf.YdPositionCost) / v.Value.Position)
                    * pf.Position * DicInstrumentField[v.Value.InstrumentID].VolumeMultiple;
            DicInvesterPosition[v.Key] = pf;
            isNeedFresh = true;
        }
    }
    if (isNeedFresh)
    {
        //刷新持仓盈亏
        TradingAccount.PositionProfit = DicInvesterPosition.Sum(n => n.Value.PositionProfit);
        //刷新可用资金
        TradingAccount.Available = TradingAccount.PreBalance - TradingAccount.Withdraw + TradingAccount.Deposit - TradingAccount.Commission
            - TradingAccount.CurrMargin - TradingAccount.FrozenCommission - TradingAccount.FrozenMargin
            - TradingAccount.DeliveryMargin + TradingAccount.Credit
            + (TradingParams.AvailIncludeCloseProfit == Constants.THOST_FTDC_ICP_NotInclude ? 0 : TradingAccount.CloseProfit)
            + (TradingParams.Algorithm == Constants.THOST_FTDC_AG_None ? 0 : TradingParams.Algorithm == Constants.THOST_FTDC_AG_All ? TradingAccount.P
            : TradingParams.Algorithm == Constants.THOST_FTDC_AG_OnlyLost ? Math.Min(0, TradingAccount.PositionProfit) : Math.Max(0, TradingAccount.Po
    }
    #endregion
```

- 成交中更新

```
void RtnTrade(ref CThostFtdcTradeField pTrade)
{
    this.DicTradeID_TradeField.TryAdd(pTrade.TradeID, pTrade);

    TradeUpdate tu = DicOrderSysID_TradeUpdate.GetOrAdd(pTrade.OrderSysID, new TradeUpdate());
    tu.OrderSysID = pTrade.OrderSysID;
    tu.TradeTime = pTrade.TradeTime;
    tu.AvgPrice = (tu.AvgPrice * tu.VolumeTraded + pTrade.Price * pTrade.Volume) / (tu.VolumeTraded + pTrade.Volume);
    tu.VolumeTraded += pTrade.Volume;
    this.DicOrderSysID_TradeUpdate[pTrade.OrderSysID] = tu;

    // 更新OrderField
    int reqID = -1;
    OrderField of = new OrderField();
    if (DicOrderSysID_RequestID.TryGetValue(pTrade.OrderSysID, out reqID))
    {
        of = this.DicOrder[reqID];
        of.AvgPrice = tu.AvgPrice;
        of.VolumeTraded = tu.VolumeTraded;
        of.VolumeLeft -= pTrade.Volume;
        of.TradeTime = tu.TradeTime;
        of.FrozenCommission = of.FrozenCommission / (of.VolumeLeft + pTrade.Volume) * of.VolumeLeft;
        of.FrozenMargin = of.FrozenMargin / (of.VolumeLeft + pTrade.Volume) * of.VolumeLeft;
        if (isLoginFinished)
            updateOrderFieldFrozenCommAndMargin(of);
        else
            DicOrder[of.RequestID] = of;
    }
```

- 根据保证金&手续费更新权益(成交中调用)

1. 冻结手续费->手续费
2. 冻结保证金-.保证金
3. 更新帐户中的冻结资金

```
//根据保证金和手续费更新权益
void updateOrderFieldFrozenCommAndMargin(OrderField f)
{
    //冻结保证金
    if (f.Status == Constants.THOST_FTDC_OST_AllTraded || f.Status == Constants.THOST_FTDC_OST_Canceled)
    {
        f.FrozenMargin = 0;
        f.FrozenCommission = 0;
    }
    else
    {
        //冻结手续费
        CThostFtdcInstrumentCommissionRateField cf;
        if (this.DicProduct_CommissionRate.TryGetValue(this.DicInstrumentField[f.InstrumentID].ProductID, out cf))
        {
            if (cf.OpenRatioByMoney == 0)
            {
                f.FrozenCommission = f.VolumeLeft * cf.OpenRatioByVolume;
            }
            else
            {
                f.FrozenCommission = f.VolumeLeft * f.LimitPrice * cf.OpenRatioByMoney;
            }
        }
        //开仓-冻结保证金
        if (f.Offset == Constants.THOST_FTDC_OF_Open)
        {
            CThostFtdcInstrumentMarginRateField mf;
            if (DicInstrumentHedge_MarginRate.TryGetValue(new Tuple<string, char>(f.InstrumentID, f.Hedge), out mf))
            {
                if (f.Director == Constants.THOST_FTDC_D_Buy)
                {
                    if (mf.LongMarginRatioByVolume == 0)
                        f.FrozenMargin = f.VolumeLeft * f.LimitPrice * this.DicInstrumentField[f.InstrumentID].VolumeMultiple * mf
                    else
                        f.FrozenMargin = f.VolumeLeft * mf.LongMarginRatioByVolume;
                }
                else //Constants.THOST_FTDC_D_Sell
                {
                    if (mf.ShortMarginRatioByVolume == 0)
                        f.FrozenMargin = f.VolumeLeft * f.LimitPrice * this.DicInstrumentField[f.InstrumentID].VolumeMultiple * mf
                    else
                        f.FrozenMargin = f.VolumeLeft * mf.ShortMarginRatioByVolume;
                }
            }
        }
    }
    this.DicOrder[f.RequestID] = f;
    //更新帐户资金
    this.TradingAccount.FrozenCommission = this.DicOrder.Sum(n => n.Value.FrozenCommission);
    this.TradingAccount.FrozenMargin = this.DicOrder.Sum(n => n.Value.FrozenMargin);
}
```

中国深圳市福田区中心三路 8 号卓越时代广场二期 14 层　518048　　电话：400 6789 819　　传真：0755-8321 7421
Excellence Times Plaza II Building,No.8 Zhong Xin 3rd Road, Futian District, Shenzhen,518048,China　　Tel :86-400 6789 819　　Fax:86-755-8321 7421

✧ 手续费查询响应中更新权益

1. 保存查到的数据
2. 更新委托单中的冻结手续费(原手续费为默认值)
3. 更新持仓中占用的手续费(原手续费为默认值)
4. 更新资金帐户中的占用手续费和手续费

```
void RspQryInstrumentCommissionRate(ref CThostFtdcInstrumentCommissionRateField pInstrumentCommissionRate, ref  CTho
{
    if (!string.IsNullOrEmpty(pInstrumentCommissionRate.InstrumentID))
    {
        CThostFtdcInstrumentCommissionRateField f = pInstrumentCommissionRate;
        this.DicProduct_CommissionRate.AddOrUpdate(f.InstrumentID, f, (key, oldvalue) => oldvalue = f);

        bool isExist = false;
        foreach (XmlNode xn in xmlCommision.DocumentElement)
        {
            if (xn.Attributes["ProductID"].Value == pInstrumentCommissionRate.InstrumentID)
            {
                xn.Attributes["UpdateDate"].InnerText = this.TradingDay;
                xn["CloseRatioByMoney"].InnerText = f.CloseRatioByMoney.ToString();
                xn["CloseRatioByVolume"].InnerText = f.CloseRatioByVolume.ToString();
                xn["CloseTodayRatioByMoney"].InnerText = f.CloseTodayRatioByMoney.ToString();
                xn["CloseTodayRatioByVolume"].InnerText = f.CloseTodayRatioByVolume.ToString();
                xn["OpenRatioByMoney"].InnerText = f.OpenRatioByMoney.ToString();
                xn["OpenRatioByVolume"].InnerText = f.OpenRatioByVolume.ToString();

                xmlCommision.Save(commissionFile);
                isExist = true;
                break;
            }
        }
        if (!isExist)
        {
            XmlNode commission = xmlCommision.CreateElement("commission");
            XmlAttribute xa1 = xmlCommision.CreateAttribute("ProductID"); xa1.InnerText = f.InstrumentID;
            XmlAttribute xa2 = xmlCommision.CreateAttribute("UpdateDate"); xa2.InnerText = this.TradingDay;
            commission.Attributes.Append(xa1);
            commission.Attributes.Append(xa2);
            XmlNode xn1 = xmlCommision.CreateElement("CloseRatioByMoney"); xn1.InnerText = f.CloseRatioByMoney.ToString();
            XmlNode xn2 = xmlCommision.CreateElement("CloseRatioByVolume"); xn2.InnerText = f.CloseRatioByVolume.ToString();
            XmlNode xn3 = xmlCommision.CreateElement("CloseTodayRatioByMoney"); xn3.InnerText = f.CloseTodayRatioByMoney.ToString()
            XmlNode xn4 = xmlCommision.CreateElement("CloseTodayRatioByVolume"); xn4.InnerText = f.CloseTodayRatioByVolume.ToString
            XmlNode xn5 = xmlCommision.CreateElement("OpenRatioByMoney"); xn5.InnerText = f.OpenRatioByMoney.ToString();
            XmlNode xn6 = xmlCommision.CreateElement("OpenRatioByVolume"); xn6.InnerText = f.OpenRatioByVolume.ToString();
            commission.AppendChild(xn1);
            commission.AppendChild(xn2);
            commission.AppendChild(xn3);
            commission.AppendChild(xn4);
            commission.AppendChild(xn5);
            commission.AppendChild(xn6);
            xmlCommision.DocumentElement.AppendChild(commission);
            xmlCommision.Save(commissionFile);
        }
```

```
        //更新委托单的冻结手续费
        for (int i = 0; i < this.DicOrder.Count; ++i)
        {
            var v = this.DicOrder.ElementAt(i);
            OrderField of = v.Value;
            if (of.InstrumentID == f.InstrumentID)
            {
                if (of.Status == Constants.THOST_FTDC_OST_Canceled || of.Status == Constants.THOST_FTDC_OST_AllTraded)
                {
                    of.FrozenMargin = 0;
                }
                else
                {
                    if (f.OpenRatioByMoney == 0)
                    {
                        of.FrozenCommission = of.VolumeLeft * f.OpenRatioByVolume;
                    }
                    else
                    {
                        of.FrozenCommission = of.VolumeLeft * of.LimitPrice * f.OpenRatioByMoney;
                    }
                }
                this.DicOrder[v.Key] = of;
            }
        }
        //更新持仓占用手续费
        for (int i = 0; i < this.DicInvesterPosition.Count; ++i)
        {
            var v = this.DicInvesterPosition.ElementAt(i);
            if (v.Value.InstrumentID == f.InstrumentID)
            {
                PositionField pf = v.Value;
                if (f.OpenRatioByVolume == 0)
                    pf.Commission = (pf.TdPositionCost + pf.YdPositionCost) * this.DicInstrumentField[pf.InstrumentID].VolumeMultip
                else
                    pf.Commission = pf.Position * f.OpenRatioByVolume;
                this.DicInvesterPosition[v.Key] = pf;
            }
        }
        //更新帐户资金
        this.TradingAccount.FrozenCommission = this.DicOrder.Sum(n => n.Value.FrozenCommission);
        this.TradingAccount.Commission = this.DicInvesterPosition.Sum(n => n.Value.Commission);
    }

    if (bIsLast)
    {
        show(EnumCallBackType.OnRspQryInstrumentCommissionRate, nRequestID);
    }
}
```

✧ 保证金查询响应中更新权益

1. 保存查到的数据
2. 更新委托单中的冻结保证金(原手续费为默认值)
3. 更新持仓中占用的保证金(原手续费为默认值)
4. 更新资金帐户中的占用保证金和保证金

```
void RspQryInstrumentMarginRate(ref CThostFtdcInstrumentMarginRateField pInstrumentMarginRate, ref  CThostFtdcRspInfoField pRspInf
{
    if (!string.IsNullOrEmpty(pInstrumentMarginRate.InstrumentID))
    {
        CThostFtdcInstrumentMarginRateField f = pInstrumentMarginRate;
        this.DicInstrumentHedge_MarginRate.AddOrUpdate(new Tuple<string, char>(f.InstrumentID, f.HedgeFlag), f, (key, oldvalue) =>
        //更新写入文件
        bool isExist = false;
        foreach (XmlNode xn in xmlMargin.DocumentElement)
        {
            if (xn.Attributes["InstrumentID"].Value == f.InstrumentID && xn.Attributes["HedgeFlag"].Value == f.HedgeFlag.ToString(
            {
                xn["LongMarginRatioByMoney"].InnerText = f.LongMarginRatioByMoney.ToString();
                xn["LongMarginRatioByVolume"].InnerText = f.LongMarginRatioByVolume.ToString();
                xn["ShortMarginRatioByMoney"].InnerText = f.ShortMarginRatioByMoney.ToString();
                xn["ShortMarginRatioByVolume"].InnerText = f.ShortMarginRatioByVolume.ToString();
                xmlMargin.Save(marginFile);
                isExist = true;
                break;
            }
        }
```

```
        //添加保证金节点
        if (!isExist)
        {
            XmlNode xnMargin = xmlMargin.CreateElement("margin");
            XmlAttribute xa1 = xmlMargin.CreateAttribute("InstrumentID"); xa1.InnerText = f.InstrumentID;
            XmlAttribute xa2 = xmlMargin.CreateAttribute("HedgeFlag"); xa2.InnerText = f.HedgeFlag.ToString();
            XmlAttribute xa3 = xmlMargin.CreateAttribute("UpdateDate"); xa3.InnerText = this.TradingDay;
            xnMargin.Attributes.Append(xa1);
            xnMargin.Attributes.Append(xa2);
            xnMargin.Attributes.Append(xa3);
            XmlNode xn1 = xmlMargin.CreateElement("LongMarginRatioByMoney"); xn1.InnerText = f.LongMarginRatioByMoney.ToString();
            XmlNode xn2 = xmlMargin.CreateElement("LongMarginRatioByVolume"); xn2.InnerText = f.LongMarginRatioByVolume.ToString();
            XmlNode xn3 = xmlMargin.CreateElement("ShortMarginRatioByMoney"); xn3.InnerText = f.ShortMarginRatioByMoney.ToString();
            XmlNode xn4 = xmlMargin.CreateElement("ShortMarginRatioByVolume"); xn4.InnerText = f.ShortMarginRatioByVolume.ToString(
            xnMargin.AppendChild(xn1);
            xnMargin.AppendChild(xn2);
            xnMargin.AppendChild(xn3);
            xnMargin.AppendChild(xn4);
            xmlMargin.DocumentElement.AppendChild(xnMargin);
            xmlMargin.Save(marginFile);
        }
        //更新委托单的冻结保证金
        for (int i = 0; i < this.DicOrder.Count; ++i)
        {
            var v = this.DicOrder.ElementAt(i);
            OrderField of = v.Value;
            if (of.InstrumentID == f.InstrumentID)
            {
                if (of.Offset != Constants.THOST_FTDC_OF_Open || of.Status == Constants.THOST_FTDC_OST_Canceled || of.Status == Co
                {
                    of.FrozenMargin = 0;
                }
                else
                {
                    if (of.Director == Constants.THOST_FTDC_D_Buy)
                    {
                        if (f.LongMarginRatioByVolume == 0)
                            of.FrozenMargin = of.VolumeLeft * of.LimitPrice * this.DicInstrumentField[f.InstrumentID].VolumeMultip
                        else
                            of.FrozenMargin = of.VolumeLeft * f.LongMarginRatioByVolume;
                    }
                    else //Constants.THOST_FTDC_D_Sell
                    {
                        if (f.ShortMarginRatioByVolume == 0)
                            of.FrozenMargin = of.VolumeLeft * of.LimitPrice * this.DicInstrumentField[f.InstrumentID].VolumeMultip
                        else
                            of.FrozenMargin = of.VolumeLeft * f.ShortMarginRatioByVolume;
                    }
                }
                this.DicOrder[v.Key] = of;
            }
        }
        //更新持仓占用保证金
        for (int i = 0; i < this.DicInvesterPosition.Count; ++i)
        {
            var v = this.DicInvesterPosition.ElementAt(i);
            if (v.Value.InstrumentID == f.InstrumentID)
            {
                PositionField pf = v.Value;
                if (pf.Direction == Constants.THOST_FTDC_D_Buy)
                {
                    if (f.LongMarginRatioByVolume == 0)
                    {
                        double tmp = this.DicInstrumentField[pf.InstrumentID].VolumeMultiple * f.LongMarginRatioByMoney;
                        pf.TdUseMargin = pf.TdPositionCost * tmp;
                        pf.YdUseMargin = pf.YdPositionCost * tmp;
                    }
                    else
                    {
                        pf.TdUseMargin = pf.TdPosition * f.LongMarginRatioByVolume;
                        pf.YdUseMargin = pf.YdPosition * f.LongMarginRatioByVolume;
                    }
                }
                else
                {
                    if (f.ShortMarginRatioByVolume == 0)
                    {
                        double tmp = this.DicInstrumentField[pf.InstrumentID].VolumeMultiple * f.ShortMarginRatioByMoney;
                        pf.TdUseMargin = pf.TdPositionCost * tmp;
                        pf.YdUseMargin = pf.YdPositionCost * tmp;
                    }
                    else
                    {
                        pf.TdUseMargin = pf.TdPosition * f.ShortMarginRatioByVolume;
                        pf.YdUseMargin = pf.YdPosition * f.ShortMarginRatioByVolume;
                    }
                }
                this.DicInvesterPosition[v.Key] = pf;
            }
        }
        //更新帐户资金
        this.TradingAccount.FrozenMargin = this.DicOrder.Sum(n => n.Value.FrozenMargin);
        this.TradingAccount.CurrMargin = this.DicInvesterPosition.Sum(n => n.Value.TdUseMargin + n.Value.YdUseMargin);
    }
```

◆ 复盘小工具

➢ 环境需求

.Net4.0 可从微软官方或 http://yunpan.cn/lk/sV6u7gsrtgFgQ 下载

➢ 下载安装

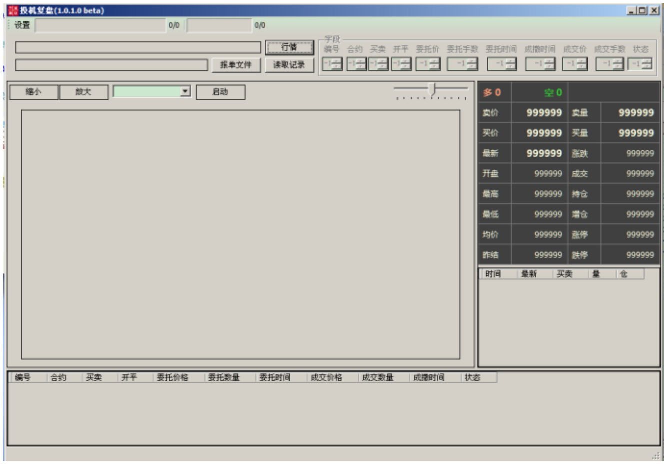
✧ 首先下载.Net4 并安装

✧ 下载压缩包后，解压至任一文件夹便可运行。

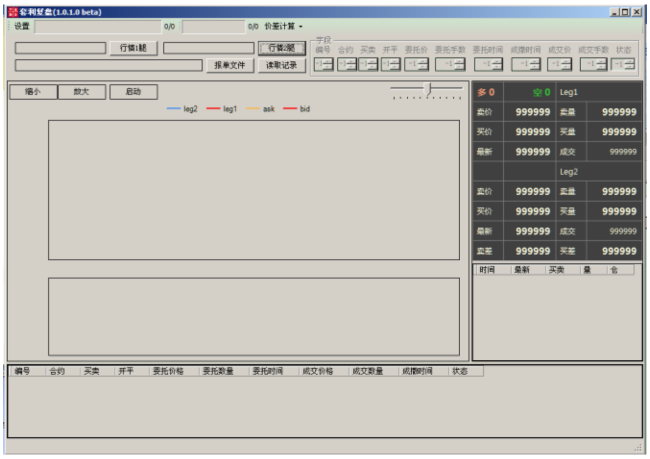
◆ 界面

➢ 选择投机（单合约）或套利（双合约）显示

➢ 投机



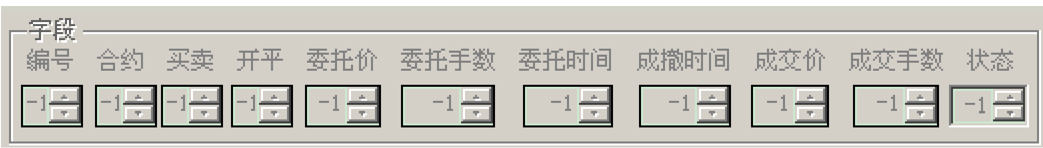
➢ 套利



◆ 文档格式

➢ 行情文档请从 http://pan.baidu.com/share/link?shareid=8031&uk=3959766851 下载

➢ 交易报单文档请在首行包含必要的字段



◆ 使用过程

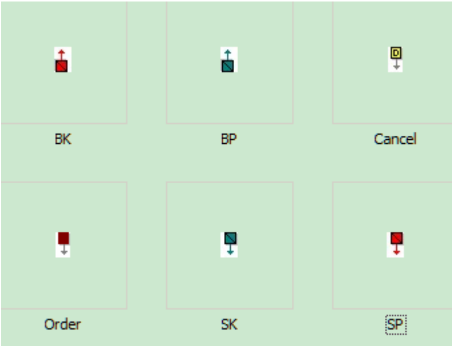
➢ 点击“行情”选择合约数据，上面会显示出行情数量。

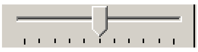
➢ 点击“报单文件”选择报单记录
此时在右侧“字段”中会显示各列对应的编号，如编号为-1 或不正确可进行调整。

➢ 点击“读取记录”，获取将报单文件中的记录。此时在上面将会显示出具体的报单数量。

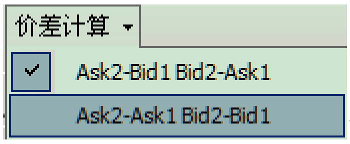
➢ 点击“启动”显示行情和报单

➢ 标志



➢ 速度控制
行情显示右上角  可控制显示速度

➢ 显示缩放
行情显示左上角“缩小”“放大”按钮可控制显示范围

➢ 价差计算
上面的 

可选择价差计算方式