TradeX Dev Team

[TradeX 交易接口开发手册 Python]

[版本 1.5.0]

[2018-06-29]

目 录

%%

重要提醒

- TradeX.dll 交易接口，是用于股票程序化交易的 DLL 接口，适合有编程基础的用户。如果您无编程基础且没有学习编程的心理准备，不建议您选择！
- 为节省您宝贵的时间，也让工程师、客服人员把时间和精力聚焦于更好的产品改进和用户体验，我们一次性提供有诚意、有竞争力的价格！谢绝议价！我们不对任何议价予以回复，感谢理解支持！
- 使用 TradeX.dll 接口之前，请仔细阅读“TradeX.dll 接口快速入门指南”！

%%

技术 QQ 群：318139137　QQ：3048747297
技术首页：https://tradexdll.com/
下载：Http://pan.baidu.com/s/1jIjYq1K

------------------------> 版本更新（2018/06/29）<------------------------

新发布的 TradeX2 交易接口有如下更新：

- 支持升级后的券商接入。
  如联讯证券、宏信证券、海通证券、西部证券、光大证券、万联证券、中泰证券、国融证券、广发证券、兴业证券、国海证券、华西证券、银河证券、大同证券、九州证券、浙商证券、联储证券、财富证券、红塔证券、华龙证券、山西证券等。
- Logon()参数格式调整
  Logon()函数中，参数“登录账户”的格式更新为“账户号+‘.标志符’”，具体参阅最新版开发手册。
  例如融资融券账户“888888888”，其参数的格式为“888888888.C”。
- 支持 Level 2 行情并发连接
  高级多账户版，允许使用多个 Level 2 行情账户建立并发行情连接（最多 32 个）

----

1. 简述

相关语言代码范例和演示程序（含 DLL 导出库、API 头文件、开发文档）可以从网盘链接/群文件夹中下载（链接：http://pan.baidu.com/s/1jIjYq1K）。

TradeX.dll 支持 32 位的 python27 和 python3.6.0，不支持 64 位的 python 环境。

TradeX.dll 标准版、精简版完全兼容 Trade.dll 和 TdxHqApi 里面的导出函数。

在最新发布的 v1.4.0 版本中，增加免配置的一键打新业务，多账户版增加了华福的委托/成交分页查询。

发布的 Python 文件包里面有 4 个文件夹，大家根据自己的情况选择。

TradeX-Py27
TradeX-Py3
TradeX-M-Py27
TradeX-M-Py3

2. 基础知识

2.1. 行情

TradeX.dll 的行情接口均要求客户端主动查询请求行情数据，而非全推送行情；交易所推送的分笔数据为 3s 切片，逐笔为每 3s 间隔内的所有成交。

当查询股票时，如果存在停牌、退市等无效的股票代码，行情服务器会使用代码 600839 自动填充。这种情况建议用户自行维护一个停牌/退市的股票列表，在批量获取行情时对列表中股票代码予以剔除。

API 接口的空闲超时缺省值为 2000ms，如果连接空闲超时，API 接口会主动释放到行情服务器的连接，并返回提示：“接收数据超时，连接断开，请重连服务器”。

2.1.1. 行情服务器

行情服务器一侧为保证系统资源的有效利用，会主动释放超时的连接；如果行情客户端要保持连接，应避免设置过长的空闲时间，可以每隔几秒请求一次行情数据，以保持保持心跳；否则超时无活动，服务器会主动断开连接。用户可以通过编写程序，每隔几秒调用一次任意一个 API 行情函数请求行情数据，以保持连接。

服务器侧同样有空闲超时机制，当查询连接超过阀值，服务器将主动释放连接，返回错误提示：“接收数据失

败，服务器主动断开连接！请检查请求数据并重连服务器”。

行情服务器对客户端频繁的连接，也会主动释放断开。

当行情服务器接收到客户端传送的错误参数（比如收到已停牌股票的代码、或不存在的股票代码），也会主动释放连接，返回错误提示：“接收数据失败，服务器主动断开连接！请检查请求数据并重连服务器”。

2.1.2. Level 2 行情

接口的 Level 2 行情服务需要用户自己购买通达信的 Level 2 金融终端账户。请到 http://vip.tdx.com.cn 购买专有 Level 2 行情账户后，开通 TradeX 授权。也可以通过客户号/资金账户号开通授权，待以后购买 Level 2 行情账户再另行绑定。

Level 2 服务器有自我保护机制（防止类 DDOS 服务攻击），针对过度频繁的访问可能视为异常连接予以并做短暂屏蔽后才重新开始接受访问。会返回错误提示：“行情链接失败！不能链接到服务器！无法验证用户”。遇到这种情况，用户应检查自己的程序是否过度频繁的重连，或者连续重连之间未设置空闲间隔或间隔太小；暂停一段，服务器恢复服务后，即可正常连接。

Level 2 行情服务器对连接的管理是后来者优先的逻辑；当用户 A 使用某行情账户建立连接，如果其他用户 B 使用相同行情账户登录，服务器检测到新的 Level 2 行情连接请求，会释放掉先前的连接并为用户 B 建立新的连接。**标准多账户版授权不支持并发多连接的 Level 2 连接行情业务;单个 Level 2 行情账户只能建立一个连接，不支持多点连接。**

新版高级多账户版允许建立最多 32 个 Level 2 的并发多连接。每个 Level 2 行情连接需要单独的行情账户，需要并发多连接 Level 2 行情的用户必须购买与连接数量对应的通达信行情账户和额外 Level 2 行情账户授权。

3. 交易 API（标准版）

3.1. API 使用流程

1）应用程序先调用 OpenTdx 做初始化；
2）调用 Logon 创建连接实例 clientID,再通过其他 API 函数向各个 ClientID 进行查询或下单；
3）应用程序退出时应调用 del clientID 删除对象引用，实例销毁自动调用 Logoff；
行情也是类似，del clientID 进行实例销毁时，自动调用 Disconnect。
4）最后调用 CloseTdx 关闭通达信实例.
5）OpenTdx 和 CloseTdx 在整个应用程序中只能被调用一次；交易 API 底层带有断线自动重连功能，应用程序只需根据 API 函数返回的出错信息进行适当错误处理即可。

注释：

- 参数分为传入、传出和双向（传入&传出）参数，需要注意区分；
- 交易接口执行后，如果失败，则字符串 ErrInfo 保存了出错信息中文说明；
- 如果成功，则字符串 Result 保存了结果数据，形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。
- Result 是\n，\t 分隔的中文字符串，比如查询股东代码时返回的结果字符串就是

  "股东代码\t 股东名称\t 帐号类别\t 保留信息\n
  0000064567\t\t0\t\nA000064567\t\t1\t\n
  2000064567\t\t2\t\nB000064567\t\t3\t"
  查得此数据之后，通过分割字符串， 可以恢复为几行几列的表格形式的数据

3.2. 函数

3.2.1. OpenTdx 初始化连接（TradeX2 插件版）

```
OpenTdx(ProcessId, nPort)
```

说明:
创建连接实例。

参数:
ProcessId - TdxW.exe 的进程 Id，（默认为 0，表示自动搜寻并使用第一个找到的 TdxW 进程 Id）；
nPort - 本地的监听端口，用户可以自行指定端口；

返回值:
无

示例:

```
error = TradeX2.OpenTdx(0, 12000)
error = TradeX2.OpenTdx(2056, 12000)
If error !=’’ :
    print (error)
```

3.2.2. OpenTdx 初始化连接

```
OpenTdx()
```

说明:
创建连接实例。

参数:
无

返回值:
无

示例:
TradeX.OpenTdx()

3.2.3. CloseTdx 关闭服务器连接实例

CloseTdx()

说明:
关闭服务器连接实例。

参数:
无

返回值:
无

示例:
TradeX.CloseTdx()

3.2.4. Logon 交易账户登录

TradeX2 的登录账户参数的格式已更新为: 原账户号+后缀类型标志符（区分大小写）。

Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)

说明:
交易账户鉴权登录。

参数:
sHost - 服务器 IP,可在券商通达信软件登录界面“通讯设置”按钮内查得;
nPort - 服务器端口，参考 etrade.xml;
sClientVersion - 券商客户端版本号，参考 tcoem.xml
nBranchID - 营业部代码，参考 etrade.xml
sClientAccount - 登录账户（Client Account），即用户在券商客户端登陆时需要输入的账户号，可以是客户号、资金账户号、股东代码等，因登录类型不同而异。
TradeX2 登录账户参数的格式已更新为: 原账户号+后缀类型标志符（区分大小写）。
“” - 普通账户
“.C” - 信用账户

例如，
普通账户“882838892”，则格式为“882838892”；
信用账户“882838892”，则格式为“882838892.C”。

sBrokerageAccount - 资金账户，用户在券商开户时资金账户号，登录券商客户端查询股东列表，从股东列表内的资金帐号一列获取。
后缀仅用于前述登录账户，资金账户无需后缀。
信用账户“882838892”，则格式为“882838892”。

sPassword - 交易密码
sCommPassword - 通信密码

返回值:
该函数返回一个行情客户端对象，发生错误的时候该函数会抛出异常。

示例:

```
sHost = "xxx.xxx.xxx.xxx"
nPort = 7708
sVersion = "6.40"
nBranchID = 9000
sClientAccount = "sssssssssssssss.C"
sBrokerageAccount = "sssssssssssssss"
sPassword = "xxxxxx"
sCommPassword = ""

try:
    client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
except TradeX.error, e:
    print "error: " + e.message
    sys.exit(-1)
```

3.2.5. IsConnectOK 判断交易连接是否正确

```
IsConnectOK()
```

说明:
判断交易连接是否正确。

参数:
无

返回值:

errinfo - 出错时函数抛出的异常信息;
1 - 成功
-1 - 参数错误 ERR_PARAM_CHECK
-2 - 内存错误 ERR_MEMORY
-3 - 逻辑错误 ERR_LOGIC

示例:

```
client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
errinfo = client.IsConnectOK()
if errinfo != 1:
    print errinfo
else:
    print 'OK'
```

3.2.6. QueryData 查询交易数据

```
QueryData(nCategory)
```

说明:

查询各种交易数据。

参数:

nCategory - 查询信息的种类,
0 资金
1 股份
2 当日委托
3 当日成交
4 可撤单
5 股东代码
6 融资余额
7 融券余额
8 可融证券
9
10
11
12 可申购新股查询
13 新股申购额度查询
14 配号查询
15 中签查询

返回值:

errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nCategory = 0

client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
errinfo, result = client.QueryData(nCategory)
if errinfo != "":
    print errinfo
else:
    print result
```

3.2.7. SendOrder 委托下单

```
SendOrder(nCategory, nOrderType, sGddm, sStockCode, sPrice, sVolume)
```

说明:

委托交易证券。

参数:

nCategory - 委托业务的种类
0 买入（包括两融担保品买入）
1 卖出（包括两融担保品卖出）
2 融资买入
3 融券卖出
4 买券还券
5 卖券还款
6 现券还券
nOrderType - 委托报价方式
0 限价委托； 上海限价委托 / 深圳限价委托
1 市价委托(深圳对方最优价格)
2 市价委托(深圳本方最优价格)
3 市价委托(深圳即时成交剩余撤销)
4 市价委托(上海五档即成剩撤 / 深圳五档即成剩撤)
5 市价委托(深圳全额成交或撤销)
6 市价委托(上海五档即成转限价)
sGddm - 股东代码
sStockCode - 证券代码
sPrice - 价格

sVolume - 委托证券的股数

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nCategory = 0
nOrderType = 4
sInvestorAccount = "p001001001005793"
sStockCode = "601988"
sPrice = 0
sVolume = 100

client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
errinfo, result = client.SendOrder(nCategory, nOrderType, sInvestorAccount, sStockCode, sPrice, sVolume)
if errinfo != "":
    print errinfo
else:
    print result
```

3.2.8. CancelOrder 委托撤单

```
CancelOrder(nMarket, Orderid)
```

说明:
获取证券的实时五档行情。

参数:
nMarket - 市场代码 0:深圳, 1:上海
Orderid - 可撤销的委托单号

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nMarket = 1
Orderid = "22"

client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount,
```

```
sPassword, sCommPassword)
    errinfo, result = client.CancelOrder(nMarket, Orderid)
    if errinfo != "":
        print errinfo
    else:
        print result
```

3.2.9. CancelOrderEx 委托撤单

```
    CancelOrderEx(nMarket, Orderid)
```

说明:

获取证券的实时五档行情。

参数:

```
 nMarket        - 市场代码    0:深圳, 1:上海
 Orderid        - 可撤销的委托单号
```

返回值:

```
    errinfo         - 出错时函数抛出的异常信息;
    result          - 查询到的数据。
```

示例:

```
    nMarket = 1
    Orderid = "22"

    client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount,
sPassword, sCommPassword)
    errinfo, result = client.CancelOrder(nMarket, Orderid)
    if errinfo != "":
        print errinfo
    else:
        print result
```

3.2.10. QueryDatas 单账户批量查询各类交易数据

```
    QueryDatas(nCategory)
```

说明:

支持单个账户同时批量查询多种类型的交易数据。

参数:

nCategory - 查询信息的种类的元组(a, b, c ...)

0 资金
1 股份
2 当日委托
3 当日成交
4 可撤单
5 股东代码
6 融资余额
7 融券余额
8 可融证券
9
10
11
12 可申购新股查询
13 新股申购额度查询
14 配号查询
15 中签查询

返回值:

errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nCategory = (0, 1, 3, 5)

client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
res = client.QueryDatas(Category)
for elem in res:
    errinfo, result = elem
    if errinfo != "":
        print errinfo
    else:
        print result
```

3.2.11. SendOrders 单账户批量委托交易证券

```
SendOrders(nCategory, nOrderType, sGddm, sStockCode, sPrice, sVolume)
```

说明:

单账户同时批量委托交易多支证券，参数为元组。

参数:

nCategory - 委托业务的种类
0 买入（包括两融担保品买入）
1 卖出（包括两融担保品卖出）
2 融资买入
3 融券卖出
4 买券还券
5 卖券还款
6 现券还券

nOrderType - 委托报价方式
0 限价委托； 上海限价委托 / 深圳限价委托
1 市价委托(深圳对方最优价格)
2 市价委托(深圳本方最优价格)
3 市价委托(深圳即时成交剩余撤销)
4 市价委托(上海五档即成剩撤 / 深圳五档即成剩撤)
5 市价委托(深圳全额成交或撤销)
6 市价委托(上海五档即成转限价)

sGddm - 股东代码
sStockCode - 证券代码
sPrice - 价格
sVolume - 委托证券的股数

返回值:

errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
    nCategory = 0
    nOrderType = 4
    sGddm = "p001001001005793"
    sStockCode = "601988"
    sPrice = 0
    sVolume = 100

    client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount,
sPassword, sCommPassword)
    res = client.SendOrders(((nCategory1, nOrderType1, sGddm1, sStockCode1, sPrice1, sVolume1),
                             (nCategory2, nOrderType2, sGddm2, sStockCode2, sPrice2, sVolume2)))
    for elem in res:
        errinfo, result = elem
        if errinfo != "":
            print errinfo
        else:
```

```
        print result
```

3.2.12. CancelOrders 单账户批量撤销委托

```
CancelOrders(nMarket, Orderid)
```

说明:

单账户批量撤销委托，参数为元组。

参数:

```
nMarket         - 市场代码     0:深圳, 1:上海
Orderid         - 可撤销委托的代码
```

返回值:

```
errinfo         - 出错时函数抛出的异常信息;
result          - 查询到的数据。
```

示例:

```
nMarket = 1
Orderid1 = 108
Orderid2 = 116

client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount,
sPassword, sCommPassword)
res = client.CancelOrders(((nMarket1, Orderid1), (nMarket2, Orderid2)))
for elem in res:
    errinfo, result = elem
    if errinfo != "":
        print errinfo
    else:
        print result
```

3.2.13. GetTradableQuantity 查询当前可交易的证券数量

```
GetTradableQuantity(nCategory, nOrderType, sAccount, sStockCode, sPrice)
```

说明:

查询当前可以交易的证券数量，比如当前可以买入或卖出的证券数量。

参数:

nCategory - 委托业务的种类
0 买入
1 卖出
2 融资买入
3 融券卖出
4 买券还券
5 卖券还款
6 现券还券
nOrderType - 委托报价方式
0 限价委托； 上海限价委托 / 深圳限价委托
1 市价委托(深圳对方最优价格)
2 市价委托(深圳本方最优价格)
3 市价委托(深圳即时成交剩余撤销)
4 市价委托(上海五档即成剩撤 / 深圳五档即成剩撤)
5 市价委托(深圳全额成交或撤销)
6 市价委托(上海五档即成转限价)
sInvestorID - 股东代码
sStockCode - 证券代码
sPrice - 价格
sVolume - 委托证券的股数

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nCategory = 0
nOrderType = 0
sInvestorID = "p001001001005793"
sStockCode = "000002"
sPrice = 3.11

client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
errinfo, result = client.GetTradableQuantity(nCategory, nOrderType, sInvestorID, sStockCode, sPrice)
if errinfo != "":
    print errinfo
else:
    print result
```

3.2.14. Repay 融资融券账户直接还款

```
Repay(sAmount)
```

说明:
融资融券账户直接还款。

参数:
sAmount - 还款金额

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
sAmount = '30000.00'

client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
errinfo, result = client.Repay(sAmount)
if errinfo != "":
    print errinfo
else:
    print result
```

3.2.15. QuickIPO 一键申购新股

```
QuickIPO()
```

说明:
一键申购新股

参数:
无

返回值:
status - 出错时函数抛出的异常信息;
>0 返回 IPO 委托申购的新股个数
=0 资金不足/无新股发行等
<0 错误异常信息

示例:

```
client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
errinfo, result = client.QuickIPO()
```

```
    if errinfo != "":
        print errinfo
    else:
        print result
```

3.2.16. QuickIPODetail 一键申购新股（明细）

```
    QuickIPODetail()
```

说明:

一键申购新股，与 QuickIPO()实现同样的业务，但返回详细的申购明细。

参数:

无

返回值:

```
    status          - 出错时函数抛出的异常信息;
                      >0  新股认购成功，返回 IPO 委托申购的新股个数
                      =0  没有申购资格，申购额度为 0；股东代码为空；没有合适的股票可以申购等
                      <0  错误异常信息
    result          - 如果新股认购成功，保存了委托编号数据；异常失败错误提示
```

示例:

```
    client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount,
sPassword, sCommPassword)
    status, content = client.QuickIPODetail()
    if status < 0:
        print content
    elif status == 0:
        print content
    else:
        for elem in content:
            errinfo, result = elem
            if errinfo != "":
                print errinfo
            else:
                print result
```

3.2.17. QueryHistoryData 查询历史交易数据

```
QueryHistoryData(nCategory, nStartDate, nEndDate)
```

说明:

查询历史交易日的委托、成交或交割单明细。

参数:

```
nCategory       - 查询信息的种类
                  0 历史委托
                  1 历史成交
                  2 资金流水
                  3 交割单
nStartDate      - 开始日期，格式为 yyyyMMdd,比如 2017 年 2 月 1 日为 20170201
nEndDate        - 结束日期，格式为 yyyyMMdd,比如 2017 年 2 月 1 日为 20170201
```

返回值:

```
errinfo         - 出错时函数抛出的异常信息;
result          - 查询到的历史交易数据列表。
```

示例:

```
nCategory = 1
nStartDate = “20170201”
nEndDate = “20170202”

client = TradeX.Logon(sHost, nPort, sClientVersion, nBranchID, sClientAccount, sBrokerageAccount,
sPassword, sCommPassword)
errinfo, result = client.QueryHistoryData(nCategory, nStartDate, nEndDate)
if errinfo != "":
    print errinfo
else:
    print result
```

4. 多连接/多账户交易 API

4.1. 使用流程

4.2. 函数

4.2.1. OpenTdx 初始化连接

```
OpenTdx(nClientType, sClientVersion, nCliType, nVipTermFlag)
```

说明:

创建连接实例。

参数:

```
nClientType      - 券商客户端类型，参考 tcoem.xml 中 “<Version ClientType=""
                   ClientVersion="7.06" CliType="" VipTermFlag="0"/>”；缺省值 14。
sClientVersion   - 客户端版本号,参考以上 ClientVersion 字段;
nCliType         - 指令接口类型，参考以上 CliType 字段；缺省值 12。
nVipTermFlag     - 终端标志代码，参考以上 VipTermFlag 字段，缺省值为 0。
```

返回值:

无

示例:

```
TradeX.OpenTdx(14, "6.40", 12, 0)
```

4.2.2. CloseTdx 关闭服务器连接实例

```
CloseTdx()
```

说明:

关闭服务器连接实例。

参数:

无

返回值:
无

示例:
```
TradeX.CloseTdx()
```

4.2.3. Logon 交易账户登录

TradeX2 登录账户参数的格式已更新为：原账户号+后缀类型标志符（区分大小写）。

```
Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
```

说明:
交易账户鉴权登录。

参数:

nBrokerID	- 券商代码，参考 tcoem.xml
sHost	- 服务器 IP,可在券商通达信软件登录界面“通讯设置”按钮内查得；
nPort	- 服务器端口，参考 etrade.xml；
sClientVersion	- 券商客户端版本号，参考 tcoem.xml
nBranchID	- 营业部代码，参考 etrade.xml
nAccountType	- 账户类型代码，参考 etrade.xml
sClientAccount	- 客户账户号（Client Account），即用户在券商客户端登陆时需要输入的账户号，可以是客户号、资金账户号、股东代码等，因登录类型不同而异。 TradeX2 登录账户参数的格式已更新为：原账户号+后缀类型标志符（区分大小写）。 “” - 普通账户 “.C” - 信用账户 例如， 普通账户“882838892”，则格式为“882838892”； 信用账户“882838892”，则格式为“882838892.C”。
sBrokerageAccount	- 资金账户，用户在券商开户时资金账户号，可以登录券商客户端以后查询获取。
sPassword	- 交易密码
sCommPassword	- 通信密码

返回值:
该函数返回一个行情客户端对象，发生错误的时候该函数会抛出异常。

示例:
```
nBrokerID = 117
sHost = "xxx.xxx.xxx.xxx"
nPort = 7708
sClientVersion = "6.40"
```

```
    nBranchID = 9000
    nAccountType = 50
    sClientAccount = "sssssssssssssss"
    sBrokerageAccount = "ssssssssssssssss"
    sPassword = "xxxxxx"
    sCommPassword = ""

try:
    client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType,
sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
except TradeX.error, e:
    print "error: " + str(e)
    sys.exit(-1)
```

4.2.4. IsConnectOK 判断交易连接是否正确

```
    IsConnectOK()
```

说明:

判断交易连接是否正确。

参数:

无

返回值:

```
    errinfo          - 出错时函数抛出的异常信息;
                       1 - 成功
                       0 - 失败参数错误 ERR_PARAM_CHECK
```

示例:

```
    client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType,
sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
    status = client.IsConnectOK()
    if status != 1:
        print errinfo
    else:
        print 'OK'
```

4.2.5. QueryData 查询各种交易数据

QueryData(nCategory)

说明:
查询各种交易数据。

参数:
nCategory - 查询信息的种类,
0 资金
1 股份
2 当日委托
3 当日成交
4 可撤单
5 股东代码
6 融资余额
7 融券余额
8 可融证券
9
10
11
12 可申购新股查询
13 新股申购额度查询
14 配号查询
15 中签查询

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nCategory = 0

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
status, content = client.QueryData(nCategory)
if status < 0:
    print "error: " + content
else:
    print content
```

4.2.6. QueryDataEx 分页查询交易数据

```
QueryDataEx(nCategory, nBatchNum, sPageMark)
```

说明:

分页查询各种交易数据。（支持华福等部分券商，其他券商暂不支持）

参数:

```
nCategory       - 查询信息的种类
                  0 资金
                  1 股份
                  2 当日委托
                  3 当日成交
                  4 可撤单
                  5 股东代码
                  6 融资余额
                  7 融券余额
                  8 可融证券
                  9
                  10
                  11
                  12 可申购新股查询
                  13 新股申购额度查询
                  14 配号查询
                  15 中签查询
nBatchNum       - 每分页的数据条目数量
sPageMark       - 当前分页的起始查询地址；第一页起始位为空字符串“”。
```

返回值:

```
errinfo         - 出错时函数抛出的异常信息；
result          - 查询到的数据。
sPageMark       - 下一页的起始查询地址；最后一页返回空字符串“”。
```

示例:

```
nCategory = 2
nBatchNum = 100
sPageMark = ""

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType,
sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
errinfo, result, sPosition = client.QueryDataEx(nCategory, nBatchNum, sPageMark)
if errinfo != "":
    print errinfo
```

```
    else:
        print result
```

完整查询示例:

```
    nCategory = 2
    nBatchNum = 100
    sPageMark = ""
    bMarkTag = False

    while(bMarkTag != True):
        print ("当前标签状态", bMarkTag)
        print (nCategory, nBatchNum, sPageMark, "\n")
        status, content, sPageMark = client.QueryHistoryDataEx(nCategory, nBatchNum, sPageMark)
        print (sPageMark,"\n")
        if status < 0:
            print ("Error: {0:s}".format(content))
        else:
            print ("{0:s}\n\n".format(content))
            if sPageMark != "":
                print ("\n\t 按任意键继续...\n")
                msvcrt.getch()
                print ("查询下一页\n")
                bMarkTag = False
            else:
                print ("最后一页\n")
                bMarkTag = True
```

4.2.7. SendOrder 委托下单

```
    SendOrder(nCategory, nOrderType, sGddm, sStockCode, sPrice, sVolume)
```

说明:

委托交易证券。

参数:

```
    nCategory       - 委托业务的种类
                      0 买入 (包括两融担保品买入)
                      1 卖出 (包括两融担保品卖出)
                      2 融资买入
                      3 融券卖出
                      4 买券还券
                      5 卖券还款
                      6 现券还券
```

nOrderType - 委托报价方式
0 限价委托； 上海限价委托 / 深圳限价委托
1 市价委托(深圳对方最优价格)
2 市价委托(深圳本方最优价格)
3 市价委托(深圳即时成交剩余撤销)
4 市价委托(上海五档即成剩撤 / 深圳五档即成剩撤)
5 市价委托(深圳全额成交或撤销)
6 市价委托(上海五档即成转限价)
sGddm - 股东代码
sStockCode - 证券代码
sPrice - 价格
sVolume - 委托证券的股数

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nCategory = 0
nOrderType = 4
sInvestorAccount = "p001001001005793"
sStockCode = "601988"
sPrice = 0
sVolume = 100

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
status, content = client.SendOrder(nCategory, nOrderType, sGddm, sStockCode, sPrice, sVolume)
if status < 0:
    print "error: " + content
else:
    print content
```

4.2.8. CancelOrder 委托撤单

```
CancelOrder(nMarket, Orderid)
```

说明:
获取证券的实时五档行情。

参数:
nMarket - 市场代码 0:深圳, 1:上海
Orderid - 可撤销委托的代码

返回值:

errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nMarket = 1
Orderid = "108"

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
status, content = client.CancelOrder(nMarket, Orderid)
if status < 0:
    print "error: " + content
else:
    print content
```

4.2.9. QueryDatas 单账户批量查询各类交易数据

```
QueryDatas(nCategory)
```

说明:

支持单个账户同时批量查询多种类型的交易数据。

参数:

nCategory - 查询信息的种类的元组(a, b, c ...)
0 资金
1 股份
2 当日委托
3 当日成交
4 可撤单
5 股东代码
6 融资余额
7 融券余额
8 可融证券
9
10
11
12 可申购新股查询
13 新股申购额度查询
14 配号查询
15 中签查询

返回值:
    errinfo - 出错时函数抛出的异常信息;
    result - 查询到的数据。

示例:
```
    nCategory = (0, 1, 3, 5)

    client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
    status, content = client.QueryDatas(nCategory)
    if status < 0:
        print content
    else:
        for elem in content:
            errinfo, result = elem
            if errinfo != "":
                print errinfo
            else:
                print result
```

4.2.10. SendOrders 单账户批量委托交易证券

```
    SendOrders(nCategory, nOrderType, sGddm, sStockCode, sPrice, sVolume)
```

说明:
    单账户同时批量委托交易多支证券，参数为元组。

参数:
    nCategory - 委托业务的种类
                0 买入 (包括两融担保品买入)
                1 卖出 (包括两融担保品卖出)
                2 融资买入
                3 融券卖出
                4 买券还券
                5 卖券还款
                6 现券还券
    nOrderType - 委托报价方式
                0 限价委托； 上海限价委托 / 深圳限价委托
                1 市价委托(深圳对方最优价格)
                2 市价委托(深圳本方最优价格)
                3 市价委托(深圳即时成交剩余撤销)
                4 市价委托(上海五档即成剩撤 / 深圳五档即成剩撤)

5 市价委托(深圳全额成交或撤销)
6 市价委托(上海五档即成转限价)

sGddm - 股东代码
sStockCode - 证券代码
sPrice - 价格
sVolume - 委托证券的股数

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nCategory = 0
nOrderType = 4
sInvestorID = "p001001001005793"
sStockCode = "601988"
sPrice = 0
sVolume = 100

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
status, content = client.SendOrders(((nCategory1, nOrderType1, sGddm1, sStockCode1, sPrice1, sVolume1),
                                     (nCategory2, nOrderType2, sGddm2, sStockCode2, sPrice2, sVolume2)))
if status < 0:
    print content
else:
    for elem in content:
        errinfo, result = elem
        if errinfo != "":
            print errinfo
        else:
            print result
```

4.2.11. CancelOrders 单账户批量撤销委托

```
CancelOrders(nMarket, Orderid)
```

说明:
单账户批量撤销委托，参数为元组。

参数:
nMarket - 市场代码 0:深圳，1:上海

Orderid - 可撤销委托的代码

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nMarket = 1
Orderid1 = 108
Orderid2 = 116

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
status, content = client.CancelOrders(((nMarket1, Orderid1), (nMarket2, Orderid2)))
if status < 0:
    print content
else:
    for elem in content:
        errinfo, result = elem
        if errinfo != "":
            print errinfo
        else:
            print result
```

4.2.12. GetTradableQuantity 查询当前可交易的证券数量

```
GetTradableQuantity(nCategory, nOrderType, sAccount, sStockCode, sPrice)
```

说明:
查询当前可以交易的证券数量，比如当前可以买入或卖出的证券数量。

参数:
nCategory - 委托业务的种类
0 买入
1 卖出
2 融资买入
3 融券卖出
4 买券还券
5 卖券还款
6 现券还券
nOrderType - 委托报价方式
0 限价委托； 上海限价委托 / 深圳限价委托
1 市价委托(深圳对方最优价格)

2 市价委托(深圳本方最优价格)
3 市价委托(深圳即时成交剩余撤销)
4 市价委托(上海五档即成剩撤 / 深圳五档即成剩撤)
5 市价委托(深圳全额成交或撤销)
6 市价委托(上海五档即成转限价)

sInvestorID - 股东代码
sStockCode - 证券代码
sPrice - 价格
sVolume - 委托证券的股数

返回值:

errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
nCategory = 0
nOrderType = 0
sInvestorID = "p001001001005793"
sStockCode = "000002"
sPrice = 3.11

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
status, content = client.GetTradableQuantity(nCategory, nOrderType, sInvestorID, sStockCode, sPrice)
if status < 0:
    print "error: " + content
else:
    print content
```

4.2.13. Repay 融资融券账户直接还款

```
Repay(sAmount)
```

说明:

融资融券账户直接还款。

参数:

sAmount - 还款金额

返回值:

errinfo - 出错时函数抛出的异常信息;
result - 查询到的数据。

示例:

```
    sAmount = '30000.00'

    client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType,
sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
    status, content = client.Repay(sAmount)
    if status < 0:
        print "error: " + content
    else:
        print content
```

4.2.14. QuickIPO 一键申购新股

```
    QuickIPO()
```

说明:

一键申购新股

参数:

无

返回值:

```
    status          - 出错时函数抛出的异常信息;
                      >0  返回 IPO 委托申购的新股个数
                      =0  资金不足/无新股发行等
                      <0  错误异常信息
```

示例:

```
    client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType,
sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
    status = client.QuickIPO()
    if status < 0:
        print "error: " + str(status)
    else:
        print "ok: " + str(status)
```

4.2.15. QuickIPODetail 一键申购新股 (明细)

```
    QuickIPODetail()
```

说明:

一键申购新股，与 QuickIPO()实现同样的业务，但返回详细的申购明细。

参数:

无

返回值:

status - 出错时函数抛出的异常信息;
>0 新股认购成功，返回 IPO 委托申购的新股个数
=0 没有申购资格，申购额度为 0；股东代码为空；没有合适的股票可以申购等
<0 错误异常信息

result - 如果新股认购成功，保存了委托编号数据；异常失败错误提示

示例:

```
client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
status, content = client.QuickIPODetail()
if status < 0:
    print "error: " + content
elif status == 0:
    print content
else:
    for elem in content:
        errinfo, result = elem
        if errinfo != "":
            print errinfo
        else:
            print result
```

4.2.16. QueryHistoryData 查询历史交易数据

```
QueryHistoryData(nCategory, nStartDate, nEndDate)
```

说明:

查询历史交易日的委托、成交或交割单明细。

参数:

nCategory - 查询信息的种类
0 历史委托
1 历史成交
2 资金流水
3 交割单

sStartDate - 开始日期，格式为 yyyyMMdd,比如 2017 年 2 月 1 日为 20170201
sEndDate - 结束日期，格式为 yyyyMMdd,比如 2017 年 2 月 1 日为 20170201

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的历史交易数据列表。

示例:

```
nCategory = 1
sStartDate = “20170201”
sEndDate = “20170202”

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
status, content = client.QueryHistoryData(nCategory, sStartDate, sEndDate)
if status < 0:
    print "error: " + content
else:
    print content
```

4.2.17. QueryHistoryDataEx 分页查询历史交易数据

```
QueryHistoryDataEx(nCategory, nBatchNum, sPageMark, nStartDate, nEndDate)
```

说明:
分页查询历史交易日的委托、成交或交割单明细。（支持华福等部分券商，其他券商暂不支持）

参数:
nCategory - 查询信息的种类
0 历史委托
1 历史成交
2 资金流水
3 交割单
nBatchNum - 每分页的数据条目数量；系统推荐值为 200;
sPageMark - 当前分页的起始查询地址；第一页起始位为空字符串“”。
sStartDate - 开始日期，格式为 yyyyMMdd,比如 2017 年 2 月 1 日为 20170201
sEndDate - 结束日期，格式为 yyyyMMdd,比如 2017 年 2 月 1 日为 20170201

返回值:
errinfo - 出错时函数抛出的异常信息;
result - 查询到的历史交易数据列表。

sPageMark - 下一页的起始查询地址；最后一页返回空字符串“”。

示例：

```
nCategory = 1
nBatchNum = 100
sPageMark = “”
sStartDate = “20170201”
sEndDate = “20170202”

client = TradeX.Logon(nBrokerID, sHost, nPort, sClientVersion, nBranchID, nAccountType, sClientAccount, sBrokerageAccount, sPassword, sCommPassword)
errinfo, result, sPosition = client. QueryHistoryDataEx (nCategory, nBatchNum, sPageMark, sStartDate, sEndDate)
if errinfo != "":
    print errinfo
else:
    print result
```

完整查询示例：

```
nCategory = 1
nBatchNum = 100
sPageMark = ""
bMarkTag = False
sStartDate = "20181112"
sEndDate = "20181112"

while(bMarkTag != True):
    print ("当前标签状态", bMarkTag)
    print (nCategory, nBatchNum, sPageMark, sStartDate, sEndDate, "\n")
    status, content, sPageMark = client.QueryHistoryDataEx(nCategory, nBatchNum, sPageMark, sStartDate, sEndDate)
    print (sPageMark,"\n")
    if status < 0:
        print ("Error: {0:s}".format(content))
    else:
        print ("{0:s}\n\n".format(content))
        if sPageMark != "":
            print ("\n\t 按任意键继续...\n")
            msvcrt.getch()
            print ("查询下一页\n")
            bMarkTag = False
        else:
            print ("最后一页\n")
            bMarkTag = True
```

5. 普通行情（五档）

- 所有行情函数均为客户端主动请求查询，而非服务器推送。

- 五档实时行情服务器侧允许单次请求股票的最大数量为 80；假如需要获取更多股票，需要分批循环查询。例如查询 3000 支股票，则需要 40*75。

- 实时行情数据每隔 3 秒刷新一次。
- 普通行情服务器可以从从配置文件 connect.cfg [HQHOST]取得。

5.1. 使用流程

1）应用程序先调用 TdxHq_Connect 连接通达信行情服务器；
2）然后才可以调用其他接口获取行情数据,应用程序应自行处理网络断线问题，接口是线程安全的。

5.2. 函数

5.2.1. TdxHq_Connect 连接行情服务器

```
TdxHq_Connect(sHost, nPort)
```

说明:
建立和行情服务器的连接，缺省连接超时 3000ms，另外在该函数设置了 tcp 的读写超时为 1000ms。

参数:
sHost -> 服务器 IP,可在券商通达信软件登录界面“通讯设置”按钮内查得；
nPort -> 服务器端口；

返回值:
该函数返回一个行情客户端对象，发生错误的时候该函数会抛出异常。

示例:

```
try:
    clientHq = TradeX.TdxHq_Connect(sHost, nPort)
except TradeX.TdxHq_error, e:
    print "error: " + e.message
    sys.exit(-1)
```

5.2.2. SetTimeout 设置行情连接超时

```
SetTimeout(nReadTimeout, nWriteTimeout)
```

说明:

设置 tcp 的读写超时，单位 ms，在连接行情服务器的时候缺省设置为 1000ms。

行情服务器超时断开后，接口会 close socket；用户需要先调用 Disconnect，然后再调用 Connect。

参数:

nReadTimeout -> 读(send)超时;

nWriteTimeout -> 写(recv)超时;

示例:

```
clientHq = TradeX.TdxHq_Connect(sHost, nPort)
clientHq.SetTimeout(500, 500)
```

5.2.3. GetSecurityCount 获取指定市场内的证券数量

```
GetSecurityCount(nMarket)
```

说明:

获取指定市场内的证券数量。

参数:

nMarket -> 市场代码 0:深圳，1:上海

返回值:

errinfo -> 出错时函数抛出的异常信息;

count -> 该函数返回指定市场内证券的总数量。

示例:

```
clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, count = clientHq.GetSecurityCount(0)
if errinfo != "":
    print errinfo
else:
    print count
```

5.2.4. GetSecurityList 获取市场内指定范围的证券代码

```
GetSecurityList(nMarket, nStart)
```

说明:

在指定市场内，抓取由指定位置起往后的 1000 支证券代码数据。

参数:

nMarket -> 市场代码 0:深圳，1:上海

nStart -> 指定的开始位置；开始位置取第一只股票（相对位置为 0），第二个相对位置为 1，依此类推；开始位置应参考 TdxHq_GetSecurityCount 返回的证券总数确定。

返回值:

errinfo -> 出错时函数抛出的异常信息

count -> 该函数返回指定市场内证券的总数量

result -> 证券代码数据；形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, count, result = clientHq.GetSecurityList(nMarket, nStart)
if errinfo != "":
    print errinfo
else:
    print count
    for line in result.split("\n"):
        print line
```

5.2.5. GetSecurityQuotes 获取多个证券的盘口五档报价数据

```
GetSecurityQuotes(((nMarket1, sStockCode1), (nMarket2, sStockCode2), ...))
```

说明:

获取多个证券的盘口五档价格数据。

参数:

(nMarket, sStockCode) -> 元组，最大允许 80 个元组，即单批最大可取证券数量不超过 80。

nMarket -> 市场代码 0:深圳，1:上海

sStockCode -> 证券代码

返回值:

errinfo -> 出错时函数抛出的异常信息

count -> 请求的证券数量，最大值为 80
result -> 证券的盘口五档价格数据；形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, count, result = clientHq.GetSecurityQuotes(((nMarket1, sStockCode1), (nMarket2, sStockCode2)))
if errinfo != "":
    print errinfo
else:
    print count
    for line in result.split("\n"):
        print line
```

5.2.6. GetSecurityBars 获取某个范围内的证券 K 线数据

```
GetSecurityBars(nCategory, nMarket, sStockCode, nStart, nCount)
```

说明:

获取市场内指定范围的证券 K 线，
指定开始位置和指定 K 线数量，指定数量最大值为 800。

参数:

nCategory -> K 线种类

0	5 分钟 K 线
1	15 分钟 K 线
2	30 分钟 K 线
3	1 小时 K 线
4	日 K 线
5	周 K 线
6	月 K 线
7	1 分钟
8	1 分钟 K 线
9	日 K 线
10	季 K 线
11	年 K 线

nMarket -> 市场代码 0:深圳，1:上海
sStockCode -> 证券代码;
nStart -> 指定的范围开始位置;
nCount -> 用户要请求的 K 线数目，最大值为 800。

返回值:

errinfo -> 出错时函数抛出的异常信息

count -> 实际获取 K 线的数量，最大值为 800；
result -> 证券的 K 线数据；形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, count, result = clientHq.GetSecurityBars(nCategory, nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

5.2.7. GetIndexBars 获取某个范围内的指数 K 线数据

```
GetIndexBars(nCategory, nMarket, sStockCode, nStart, nCount)
```

说明:

获取市场内指定范围的指数 K 线，
指定开始位置和指定 K 线数量，指定数量最大值为 800。

参数:

nCategory -> K 线种类

0	5 分钟 K 线
1	15 分钟 K 线
2	30 分钟 K 线
3	1 小时 K 线
4	日 K 线
5	周 K 线
6	月 K 线
7	1 分钟
8	1 分钟 K 线
9	日 K 线
10	季 K 线
11	年 K 线

nMarket -> 市场代码 0:深圳，1:上海
sStockCode -> 证券代码；
nStart -> 指定的范围开始位置；
nCount -> 用户要请求的 K 线数目。

返回值:

errinfo -> 出错时函数抛出的异常信息
count - 返回实际的 K 线数目，最大值为 800
result - 返回 K 线数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔；

示例:

```
clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, count, result = clientHq.GetIndexBars(nCategory, nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

5.2.8. GetMinuteTimeData 获取当日分时成交数据

```
GetMinuteTimeData(nMarket, sStockCode)
```

说明:

获取当日的分时成交数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码。

返回值:

errinfo -> 出错时函数抛出的异常信息;
result - 当日的分时成交数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"

clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, result = clientHq.GetMinuteTimeData(nMarket, sStockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

5.2.9. GetHistoryMinuteTimeData 获取历史分时成交数据

```
GetHistoryMinuteTimeData(nMarket, sStockCode, nDate)
```

说明:

获取指定的历史交易日的分时成交数据

参数:
nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码。
nDate -> 指定的历史交易日期, 如 2017 年 2 月 1 日, 则为整数 20170201

返回值:
errinfo -> 出错时函数抛出的异常信息;
result - 指定的历史交易日的分时成交数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:
```
nMarket = 0
sStockCode = "000001"
nDate = 20161209

clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, result = clientHq.GetHistoryMinuteTimeData(nMarket, sStockCode, nDate)
if errinfo != "":
    print errinfo
else:
    print result
```

5.2.10. GetTransactionData 获取当日指定范围的分笔成交数据

```
GetTransactionData(nMarket, sStockCode, nStart, nCount)
```

说明:

获取当日指定范围的分笔成交数据

参数:
nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;
nStart -> 指定的范围开始位置;
nCount -> 请求的分笔成交笔数, 最大值为 2000。

返回值:
errinfo -> 出错时函数抛出的异常信息;

count - 实际获取到的分笔成交笔数，最大值为 2000；
result - 指定范围内的分笔成交数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nStart = 0
nCount = 10

clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, count, result = clientHq.GetTransactionData(nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

5.2.11. GetHistoryTransactionData 获取历史交易日指定范围内的分笔成交数据

```
GetHistoryTransactionData(nMarket, sStockCode, nStart, nCount, nDate)
```

说明:

获取历史交易日指定范围内的分笔成交数据

参数:

nMarket -> 市场代码 0:深圳，1:上海
sStockCode -> 证券代码；
nStart -> 指定的范围开始位置；
nCount -> 请求的分笔成交笔数，最大值为 2000；
nDate -> 指定的历史交易日期，如 2017 年 2 月 1 日，则为整数 20170201。

返回值:

errinfo -> 出错时函数抛出的异常信息；
count - 实际获取到的分笔成交笔数，最大值为 2000；
result - 历史交易日指定范围内的分笔成交数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。

示例:

```
nMarket = 0
```

```
    sStockCode = "000001"
    nStart = 0
    nCount = 10
    nDate = 20170209

    clientHq = TradeX.TdxHq_Connect(sHost, nPort)

    errinfo, count, result = clientHq.GetHistoryTransactionData(nMarket, sStockCode, nStart, nCount,
nDate)
    if errinfo != "":
        print errinfo
    else:
        print count
        print result
```

5.2.12. GetCompanyInfoCategory 获取公司基本面信息（F10）类目

```
    GetCompanyInfoCategory(nMarket, sStockCode)
```

说明:

获取公司基本面信息（F10）类目

参数:

nMarket -> 市场代码 0:深圳，1:上海
sStockCode -> 证券代码

返回值:

errinfo -> 出错时函数抛出的异常信息；
result - 公司基本面信息类目列表数据。

示例:

```
    nMarket = 0
    sStockCode = "000001"

    clientHq = TradeX.TdxHq_Connect(sHost, nPort)
    errinfo, result = clientHq.GetCompanyInfoCategory(nMarket, sStockCode)
    if errinfo != "":
        print errinfo
    else:
        print result
```

5.2.13. GetCompanyInfoContent 获取公司基本面信息（F10）具体内容

```
GetCompanyInfoContent(nMarket, sStockCode, FileName, nStart, nLength)
```

说明:

获取某一类公司基本面信息（F10）具体内容

参数:

nMarket -> 市场代码 0:深圳，1:上海
sStockCode -> 证券代码;
FileName -> 该类目所在文件名，从 TdxHq_GetCompanyInfoCategory 返回信息中获取;
nStart -> 该类目的开始位置，从 TdxHq_GetCompanyInfoCategory 返回信息中获取;
nLength -> 该类目长度，从 TdxHq_GetCompanyInfoCategory 返回信息中获取。

返回值:

errinfo -> 出错时函数抛出的异常信息;
result - 指定类目的基本面信息具体内容；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
FileName = '000001.txt'
nStart = 221077
nLength = 39547

clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, result = clientHq.GetCompanyInfoContent(nMarket, sStockCode, FileName, nStart, nLength)
if errinfo != "":
    print errinfo
else:
    print result
```

5.2.14. GetXDXRInfo 获取除权除息数据

```
GetXDXRInfo(nMarket, sStockCode)
```

说明:

获取证券的除权除息数据

参数:
nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;

返回值:
errinfo -> 出错时函数抛出的异常信息;
result - 证券的除权除息数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"

clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, result = clientHq.GetXDXRInfo(nMarket, sStockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

5.2.15. GetFinanceInfo 获取财务数据

```
GetFinanceInfo(nMarket, sStockCode)
```

说明:

获取财务数据

参数:
nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;

返回值:
errinfo -> 出错时函数抛出的异常信息;
result -> 证券的财务数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"

clientHq = TradeX.TdxHq_Connect(sHost, nPort)
errinfo, result = clientHq.GetFinanceInfo(nMarket, sStockCode)
if errinfo != "":
```

```
    print errinfo
else:
    print result
```

6. Level 2 行情（十档/逐笔）

6.1. 使用流程

Level 2 行情服务器主机可以从从配置文件 zdcomet.cfg 取得。

6.2. 函数

6.2.1. TdxL2Hq_Connect 连接行情服务器

```
TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
```

说明:

建立和行情服务器的连接，缺省连接超时 3000ms，另外在该函数设置了 tcp 的读写超时为 1000ms。

参数:

sHost -> 服务器 IP,可在券商通达信软件登录界面“通讯设置”按钮内查得;
nPort -> 服务器端口;
L2User -> Level 2 行情账户名，从通达信购买
L2Password -> Level 2 行情账户密码

返回值:

该函数返回一个行情客户端对象，发生错误的时候该函数会抛出异常。

示例:

```
try:
```

```
    clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
except TradeX.TdxL2Hq_error, e:
    print "error: " + e.message
    sys.exit(-1)
```

6.2.2. SetTimeout 设置行情连接超时

```
SetTimeout(nReadTimeout, nWriteTimeout)
```

说明:

设置 tcp 的读写超时，单位 ms，在连接行情服务器的时候缺省设置为 1000ms。

行情服务器超时断开后，接口会 close socket；用户需要先调用 Disconnect，然后再调用 Connect。

参数:

nReadTimeout -> 读(send)超时;

nWriteTimeout -> 写(recv)超时;

示例:

```
clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
clientL2Hq.SetTimeout(500, 500)
```

6.2.3. GetSecurityCount 获取指定市场内的证券数量

```
GetSecurityCount(nMarket)
```

说明:

获取指定市场内的证券数量。

参数:

nMarket -> 市场代码 0:深圳，1:上海

返回值:

errinfo -> 出错时函数抛出的异常信息;

count -> 该函数返回指定市场内证券的总数量。

示例:

```
clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, count = clientL2Hq.GetSecurityCount(0)
if errinfo != "":
    print errinfo
```

```
    else:
        print count
```

6.2.4. GetSecurityList 获取市场内指定范围的证券代码

```
    GetSecurityList(nMarket, nStart)
```

说明:

在指定市场内，抓取由指定位置起往后的 1000 支证券代码数据。

参数:

nMarket -> 市场代码 0:深圳，1:上海

nStart -> 指定的开始位置；开始位置取第一只股票（相对位置为 0），第二个相对位置为 1，依此类推；开始位置应参考 TdxL2Hq_GetSecurityCount 返回的证券总数确定。

返回值:

errinfo -> 出错时函数抛出的异常信息

count -> 该函数返回指定市场内证券的总数量

result -> 证券代码数据；形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, count, result = clientL2Hq.GetSecurityList(nMarket, nStart)
if errinfo != "":
    print errinfo
else:
    print count
    for line in result.split("\n"):
        print line
```

6.2.5. GetSecurityQuotes 获取多个证券的盘口五档报价数据

```
    GetSecurityQuotes(((nMarket1, sStockCode1), (nMarket2, sStockCode2), ...))
```

说明:

获取多个证券的盘口五档价格数据。

参数:

(nMarket, sStockCode) -> 元组，最大允许 80 个元组，即单批最大可取证券数量不超过 80。

nMarket -> 市场代码 0:深圳，1:上海

sStockCode -> 证券代码

返回值:
errinfo -> 出错时函数抛出的异常信息
count -> 请求的证券数量，最大值为 80
result -> 证券的盘口五档价格数据；形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, count, result = clientL2Hq.GetSecurityQuotes(((nMarket1, sStockCode1), (nMarket2, sStockCode2)))
if errinfo != "":
    print errinfo
else:
    print count
    for line in result.split("\n"):
        print line
```

6.2.6. GetSecurityQuotes10 获取多个证券的盘口十档报价数据

```
GetSecurityQuotes10(((nMarket1, sStockCode1), (nMarket2, sStockCode2), ...))
```

说明:
获取多个证券的盘口十档价格数据。

参数:
(nMarket, sStockCode) -> 元组，最大允许 100 个元组，即单批最大可取证券数量不超过 100。
nMarket -> 市场代码 0:深圳，1:上海
sStockCode -> 证券代码

返回值:
errinfo -> 出错时函数抛出的异常信息
count -> 请求的证券数量，最大值为 100
result -> 证券的盘口十档价格数据；形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, count, result = clientL2Hq.GetSecurityQuotes10(((nMarket1, sStockCode1), (nMarket2, sStockCode2)))
if errinfo != "":
    print errinfo
else:
    print count
```

```
for line in result.split("\n"):
    print line
```

6.2.7. GetSecurityBars 获取某个范围内的证券 K 线数据

```
GetSecurityBars(nCategory, nMarket, sStockCode, nStart, nCount)
```

说明:

获取市场内指定范围的证券 K 线,
指定开始位置和指定 K 线数量，指定数量最大值为 800。

参数:

nCategory -> K 线种类

0	5 分钟 K 线
1	15 分钟 K 线
2	30 分钟 K 线
3	1 小时 K 线
4	日 K 线
5	周 K 线
6	月 K 线
7	1 分钟
8	1 分钟 K 线
9	日 K 线
10	季 K 线
11	年 K 线

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;
nStart -> 指定的范围开始位置;
nCount -> 用户要请求的 K 线数目，最大值为 800。

返回值:

errinfo -> 出错时函数抛出的异常信息
count -> 实际获取 K 线的数量，最大值为 800;
result -> 证券的 K 线数据; 形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, count, result = clientL2Hq.GetSecurityBars(nCategory, nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

6.2.8. GetIndexBars 获取某个范围内的指数 K 线数据

GetIndexBars(nCategory, nMarket, sStockCode, nStart, nCount)

说明:

获取市场内指定范围的指数 K 线,
指定开始位置和指定 K 线数量，指定数量最大值为 800。

参数:

nCategory -> K 线种类

0	5 分钟 K 线
1	15 分钟 K 线
2	30 分钟 K 线
3	1 小时 K 线
4	日 K 线
5	周 K 线
6	月 K 线
7	1 分钟
8	1 分钟 K 线
9	日 K 线
10	季 K 线
11	年 K 线

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;
nStart -> 指定的范围开始位置;
nCount -> 用户要请求的 K 线数目。

返回值:

errinfo -> 出错时函数抛出的异常信息
count - 返回实际的 K 线数目, 最大值为 800
result - 返回 K 线数据; 形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔;

示例:

```
clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, count, result = clientL2Hq.GetIndexBars(nCategory, nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

6.2.9. GetMinuteTimeData 获取当日分时成交数据

```
GetMinuteTimeData(nMarket, sStockCode)
```

说明:

获取当日的分时成交数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码。

返回值:

errinfo -> 出错时函数抛出的异常信息;
result - 当日的分时成交数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, result = clientL2Hq.GetMinuteTimeData(nMarket, sStockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

6.2.10. GetHistoryMinuteTimeData 获取历史分时成交数据

```
GetHistoryMinuteTimeData(nMarket, sStockCode, nDate)
```

说明:

获取指定的历史交易日的分时成交数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码。
nDate -> 指定的历史交易日期, 如 2017 年 2 月 1 日, 则为整数 20170201

返回值:

errinfo -> 出错时函数抛出的异常信息;
result - 指定的历史交易日的分时成交数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nDate = 20161209

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, result = clientL2Hq.GetHistoryMinuteTimeData(nMarket, sStockCode, nDate)
if errinfo != "":
    print errinfo
else:
    print result
```

6.2.11. GetTransactionData 获取当日指定范围的分笔成交数据

```
GetTransactionData(nMarket, sStockCode, nStart, nCount)
```

说明:

获取当日指定范围的分笔成交数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;
nStart -> 指定的范围开始位置;
nCount -> 请求的分笔成交笔数, 最大值为 2000。

返回值:

errinfo -> 出错时函数抛出的异常信息;
count - 实际获取到的分笔成交笔数, 最大值为 2000;
result - 指定范围内的分笔成交数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nStart = 0
nCount = 10

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
```

```
errinfo, count, result = clientL2Hq.GetTransactionData(nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

6.2.12. GetHistoryTransactionData 获取历史交易日指定范围内的分笔成交数据

```
GetHistoryTransactionData(nMarket, sStockCode, nStart, nCount, nDate)
```

说明:

获取历史交易日指定范围内的分笔成交数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;
nStart -> 指定的范围开始位置;
nCount -> 请求的分笔成交笔数, 最大值为 2000;
nDate -> 指定的历史交易日期, 如 2017 年 2 月 1 日, 则为整数 20170201。

返回值:

errinfo -> 出错时函数抛出的异常信息;
count - 实际获取到的分笔成交笔数, 最大值为 2000;
result - 历史交易日指定范围内的分笔成交数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nStart = 0
nCount = 10
nDate = 20170209

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)

errinfo, count, result = clientL2Hq.GetHistoryTransactionData(nMarket, sStockCode, nStart, nCount, nDate)
```

```
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

6.2.13. GetCompanyInfoCategory 获取公司基本面信息（F10）类目

```
GetCompanyInfoCategory(nMarket, sStockCode)
```

说明:

获取公司基本面信息（F10）类目

参数:
nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码

返回值:
errinfo -> 出错时函数抛出的异常信息;
result - 公司基本面信息类目列表数据。

示例:

```
nMarket = 0
sStockCode = "000001"

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, result = clientL2Hq.GetCompanyInfoCategory(nMarket, sStockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

6.2.14. GetCompanyInfoContent 获取公司基本面信息（F10）具体内容

```
GetCompanyInfoContent(nMarket, sStockCode, FileName, nStart, nLength)
```

说明:

获取某一类公司基本面信息（F10）具体内容

参数:
nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;
FileName -> 该类目所在文件名, 从 TdxL2Hq_GetCompanyInfoCategory 返回信息中获取;
nStart -> 该类目的开始位置, 从 TdxL2Hq_GetCompanyInfoCategory 返回信息中获取;
nLength -> 该类目长度, 从 TdxL2Hq_GetCompanyInfoCategory 返回信息中获取。

返回值:
errinfo -> 出错时函数抛出的异常信息;
result - 指定类目的基本面信息具体内容; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
FileName = '000001.txt'
nStart = 221077
nLength = 39547

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, result = clientL2Hq.GetCompanyInfoContent(nMarket, sStockCode, FileName, nStart, nLength)
if errinfo != "":
    print errinfo
else:
    print result
```

6.2.15. GetXDXRInfo 获取除权除息数据

```
GetXDXRInfo(nMarket, sStockCode)
```

说明:

获取证券的除权除息数据

参数:
nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;

返回值:
errinfo -> 出错时函数抛出的异常信息;
result - 证券的除权除息数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, result = clientL2Hq.GetXDXRInfo(nMarket, sStockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

6.2.16. GetFinanceInfo 获取财务数据

```
GetFinanceInfo(nMarket, sStockCode)
```

说明:

获取财务数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;

返回值:

errinfo -> 出错时函数抛出的异常信息;
result -> 证券的财务数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, result = clientL2Hq.GetFinanceInfo(nMarket, sStockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

6.2.17. GetDetailOrderData 获取指定范围内的逐笔委托数据

```
GetDetailOrderData(nMarket, sStockCode, nStart, nCount)
```

说明:

获取指定范围内的逐笔委托数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;
nStart -> 指定的范围开始位置;
nCount -> 请求的逐笔委托笔数, 最大值为 2000;

返回值:

errinfo -> 出错时函数抛出的异常信息;
count - 实际获取到的逐笔委托笔数, 最大值为 2000;
result -> 逐笔委托数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nStart = 0
nCount = 10

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, count, result = clientL2Hq.GetDetailOrderData(nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    for line in result.split("\n"):
        print line
```

6.2.18. GetDetailOrderDataEx 按时间顺序获取指定范围内的逐笔委托数据

```
GetDetailOrderDataEx(nMarket, sStockCode, nStart, nCount)
```

说明:

获取指定范围内的逐笔委托数据

参数:

```
    nMarket -> 市场代码     0:深圳，1:上海
    sStockCode -> 证券代码;
    nStart -> 指定的范围开始位置;
    nCount -> 请求的逐笔委托笔数，最大值为 2000;
```

返回值:

```
    errinfo -> 出错时函数抛出的异常信息;
    count - 实际获取到的逐笔委托笔数，最大值为 2000;
    result -> 逐笔委托数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。
```

示例:

```
    nMarket = 0
    sStockCode = "000001"
    nStart = 0
    nCount = 10

    clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
    errinfo, count, result = clientL2Hq.GetDetailOrderDataEx(nMarket, sStockCode, nStart, nCount)
    if errinfo != "":
        print errinfo
    else:
        print count
        for line in result.split("\n"):
            print line
```

6.2.19. GetDetailTransactionData 获取指定范围内的逐笔成交数据

```
    GetDetailTransactionData(nMarket, sStockCode, nStart, nCount)
```

说明:

```
    获取指定范围内的逐笔成交数据
```

参数:

```
    nMarket -> 市场代码     0:深圳，1:上海
    sStockCode -> 证券代码;
    nStart -> 指定的范围开始位置;
    nCount -> 请求的逐笔成交笔数，最大值为 2000;
```

返回值:

```
    errinfo -> 出错时函数抛出的异常信息;
    count - 实际获取到的逐笔成交笔数，最大值为 2000;
    result -> 逐笔成交数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。
```

示例:

```
nMarket = 0
sStockCode = "000001"
nStart = 0
nCount = 10

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, count, result = clientL2Hq.GetDetailTransactionData(nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    for line in result.split("\n"):
        print line
```

6.2.20. GetDetailTransactionDataEx 按时间顺序获取指定范围内的逐笔成交数据

```
GetDetailTransactionDataEx(nMarket, sStockCode, nStart, nCount)
```

说明:

获取指定范围内的逐笔成交数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;
nStart -> 指定的范围开始位置;
nCount -> 请求的逐笔成交笔数, 最大值为 2000;

返回值:

errinfo -> 出错时函数抛出的异常信息;
count - 实际获取到的逐笔成交笔数, 最大值为 2000;
result -> 逐笔成交数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nStart = 0
nCount = 10

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
```

```
errinfo, count, result = clientL2Hq.GetDetailTransactionDataEx(nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    for line in result.split("\n"):
        print line
```

6.2.21. GetBuySellQueue 获取买卖队列数据

```
GetBuySellQueue(nMarket, sStockCode)
```

说明:

获取买卖队列数据

参数:

nMarket -> 市场代码 0:深圳, 1:上海
sStockCode -> 证券代码;

返回值:

errinfo -> 出错时函数抛出的异常信息;
result -> 买卖队列数据; 形式为表格数据, 行数据之间通过\n 字符分割, 列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"

clientL2Hq = TradeX.TdxL2Hq_Connect(sHost, nPort, L2User, L2Password)
errinfo, result = clientL2Hq.GetBuySellQueue(nMarket, sStockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

7. 扩展行情（期货/外盘）

7.1. 使用流程

扩展行情主机信息从配置文件 connect.cfg [[DSHOST]取得。

7.2. 函数

7.2.1. TdxExHq_Connect 扩展行情模块连接扩展行情服务器

```
TdxExHq_Connect(sHost, nPort)
```

说明:

建立和行情服务器的连接，缺省连接超时 3000ms，另外在该函数设置了 tcp 的读写超时为 1000ms。

参数:

sHost -> 服务器 IP,可在券商通达信软件登录界面“通讯设置”按钮内查得;

nPort -> 服务器端口;

返回值:

该函数返回一个行情客户端对象，发生错误的时候该函数会抛出异常。

示例:

```
try:
    clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
except TradeX.TdxExHq_error, e:
    print "error: " + e.message
    sys.exit(-1)
```

7.2.2. SetTimeout 设置行情连接超时

```
SetTimeout(nReadTimeout, nWriteTimeout)
```

说明:

设置 tcp 的读写超时，单位 ms，在连接行情服务器的时候缺省设置为 1000ms。

行情服务器超时断开后，接口会 close socket；用户需要先调用 Disconnect，然后再调用 Connect。

参数:

nReadTimeout -> 读(send)超时;

nWriteTimeout -> 写(recv)超时;

示例:

```
clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
clientExHq.SetTimeout(500, 500)
```

7.2.3. GetMarkets 获取扩展行情所支持的各市场代码

```
GetMarkets()
```

说明:

获取扩展行情所支持的各市场代码。

参数:

无

返回值:

errinfo -> 出错时函数抛出的异常信息;

count -> 市场代码列表。

示例:

```
clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
errinfo, result = clientExHq.GetMarkets()
if errinfo != "":
    print errinfo
else:
    print result
```

7.2.4. GetInstrumentCount 获取市场内的商品总数量

```
GetInstrumentCount()
```

说明:

获取市场内的商品总数量。

参数:

无

返回值:

errinfo -> 出错时函数抛出的异常信息;

count -> 该函数返回获取市场内商品总数量。

示例:

```
clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
errinfo, count = clientExHq.GetInstrumentCount()
if errinfo != "":
    print errinfo
else:
    print count
```

7.2.5. GetInstrumentInfo 获取指定范围内商品的代码

GetInstrumentInfo(nStart)

说明:

获取指定范围内商品的代码。

参数:

nStart -> 指定的开始位置; 开始位置取第一只商品 (相对位置为 0), 第二个相对位置为 1, 依此类推; 开始位置应参考 TdxExHq_GetInstrumentCount 返回的商品总数确定。

返回值:

errinfo -> 出错时函数抛出的异常信息

count -> 该函数返回市场内商品的数量

result -> 商品代码数据; 形式为表格数据, 行数据以‘\n’分割, 列数据以‘\t’分隔。

示例:

```
nStart = 0

clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
errinfo, count, result = clientExHq.GetInstrumentInfo(nStart)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

7.2.6. GetInstrumentQuote 获取指定商品的盘口报价

```
GetInstrumentQuote(nMarket, StockCode)
```

说明:

获取指定商品的盘口价格数据。

参数:

nMarket -> 市场代码
sStockCode -> 商品代码

返回值:

errinfo -> 出错时函数抛出的异常信息
result -> 商品的盘口价格数据；形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
nMarket = 47
StockCode = "IF1706"

clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
errinfo, result = clientExHq.GetInstrumentQuote(nMarket, StockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

7.2.7. GetInstrumentBars 获取某个范围内的商品 K 线数据

```
GetInstrumentBars(nCategory, nMarket, StockCode, nStart, nCount)
```

说明:

获取市场内指定范围的商品 K 线数据;
指定开始位置和指定 K 线数量，指定数量最大值为 800。

参数:

nCategory -> K 线种类
0 5 分钟 K 线
1 15 分钟 K 线
2 30 分钟 K 线
3 1 小时 K 线
4 日 K 线

5 周 K 线
6 月 K 线
7 1 分钟
8 1 分钟 K 线
9 日 K 线
10 季 K 线
11 年 K 线

nMarket -> 市场代码;
sStockCode -> 商品代码;
nStart -> 指定的范围开始位置;
nCount -> 用户要请求的 K 线数目，最大值为 800。

返回值:
errinfo -> 出错时函数抛出的异常信息
count -> 实际获取 K 线的数量，最大值为 800;
result -> 商品的 K 线数据；形式为表格数据，行数据以‘\n’分割，列数据以‘\t’分隔。

示例:

```
clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
errinfo, count, result = clientExHq.GetSecurityBars(nCategory, nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

7.2.8. GetMinuteTimeData 获取当日分时成交数据

```
GetMinuteTimeData(nMarket, sStockCode)
```

说明:

获取当日的分时成交数据

参数:
nMarket -> 市场代码
sStockCode -> 商品代码。

返回值:
errinfo -> 出错时函数抛出的异常信息;
result - 当日的分时成交数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"

clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
errinfo, result = clientExHq.GetMinuteTimeData(nMarket, sStockCode)
if errinfo != "":
    print errinfo
else:
    print result
```

7.2.9. GetHistoryMinuteTimeData 获取历史分时成交数据

```
GetHistoryMinuteTimeData(nMarket, sStockCode, nDate)
```

说明:

获取指定的历史交易日的分时成交数据

参数:

nMarket -> 市场代码
sStockCode -> 商品代码。
nDate -> 指定的历史交易日期，如 2017 年 2 月 1 日，则为整数 20170201

返回值:

errinfo -> 出错时函数抛出的异常信息；
result - 指定的历史交易日的分时成交数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nDate = 20161209

clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
errinfo, result = clientExHq.GetHistoryMinuteTimeData(nMarket, sStockCode, nDate)
if errinfo != "":
    print errinfo
else:
    print result
```

7.2.10. GetTransactionData 获取当日指定范围的分笔成交数据

```
GetTransactionData(nMarket, sStockCode, nStart, nCount)
```

说明:

获取当日指定范围的分笔成交数据

参数:

nMarket -> 市场代码
sStockCode -> 商品代码;
nStart -> 指定的范围开始位置;
nCount -> 请求的分笔成交笔数，最大值为 2000。

返回值:

errinfo -> 出错时函数抛出的异常信息;
count - 实际获取到的分笔成交笔数，最大值为 2000;
result - 指定范围内的分笔成交数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nStart = 0
nCount = 10

clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)
errinfo, count, result = clientExHq.GetTransactionData(nMarket, sStockCode, nStart, nCount)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```

7.2.11. GetHistoryTransactionData 获取历史交易日指定范围内的分笔成交数据

```
GetHistoryTransactionData(nMarket, sStockCode, nStart, nCount, nDate)
```

说明:

获取历史交易日指定范围内的分笔成交数据

参数:
nMarket -> 市场代码
sStockCode -> 商品代码;
nStart -> 指定的范围开始位置;
nCount -> 请求的分笔成交笔数，最大值为 2000;
nDate -> 指定的历史交易日期，如 2017 年 2 月 1 日，则为整数 20170201。

返回值:
errinfo -> 出错时函数抛出的异常信息;
count - 实际获取到的分笔成交笔数，最大值为 2000;
result - 历史交易日指定范围内的分笔成交数据；形式为表格数据，行数据之间通过\n 字符分割，列数据之间通过\t 分隔。

示例:

```
nMarket = 0
sStockCode = "000001"
nStart = 0
nCount = 10
nDate = 20170209

clientExHq = TradeX.TdxExHq_Connect(sHost, nPort)

errinfo, count, result = clientExHq.GetHistoryTransactionData(nMarket, sStockCode, nStart, nCount, nDate)
if errinfo != "":
    print errinfo
else:
    print count
    print result
```