BackTestApi股票Lev2行情回测工具使用指南

目录

[BackTestApi股票Lev2行情回测工具使用指南 1](#_Toc50825915)

[1 BackTestApi股票Lev2行情回测工具介绍 2](#_Toc50825916)

[1.1 概述 2](#_Toc50825917)

[1.2 Lev2-BackTestApi接口文件 2](#_Toc50825918)

[1.3 Lev2-BackTestApi接口类 4](#_Toc50825919)

[2 Lev2回测前必须知道的信息 5](#_Toc50825920)

[2.1 Lev2回测服务地址 5](#_Toc50825921)

[2.2 Lev2回测的用户名和密码 5](#_Toc50825922)

[2.3 Lev2回测的历史行情 5](#_Toc50825923)

[2.4 Lev2回测日期跨度 6](#_Toc50825924)

[2.5 Lev2回测报单类型 6](#_Toc50825925)

[2.6 Lev2回测交易成交规则 6](#_Toc50825926)

[2.7 Lev2回测缓存机制 7](#_Toc50825927)

[3 如何快速开始Lev2回测（C++） 7](#_Toc50825928)

[3.1 开发环境准备 8](#_Toc50825929)

[3.2 设置回测初始参数 8](#_Toc50825930)

[3.3 如何编写回测程序 9](#_Toc50825931)

[3.3.1 初始化TraderAPI 9](#_Toc50825932)

[3.3.2 初始化Lev2MdAPI 10](#_Toc50825933)

[3.3.3 回测单个交易日 11](#_Toc50825934)

[3.3.4 登录Login 12](#_Toc50825935)

[3.3.5 报单申报 16](#_Toc50825936)

[3.3.6 如何运行Lev2回测实例 17](#_Toc50825937)

[3.3.7 查询回测结果 17](#_Toc50825938)

[4 如何快速开始Lev2回测（Python） 17](#_Toc50825939)

[4.1 开发环境准备 17](#_Toc50825940)

[4.2 如何运行回测程序 18](#_Toc50825941)

[5 Lev2MdApi如何使用详解 18](#_Toc50825942)

[5.1 登录&订阅Lev2行情 18](#_Toc50825943)

[5.2 Lev2行情推送 20](#_Toc50825944)

[5.3 回测lev2mdapi与生产lev2mdapi区别 20](#_Toc50825945)

[5.4 回测lev2mdapi支持生产lev2mdapi的哪些接口 21](#_Toc50825946)

# BackTestApi股票Lev2行情回测工具介绍

## 概述

感谢您使用BackTestApi股票Lev2行情回测工具，本文主要介绍此工具的使用方法。在本文中，为区别原有BackTestApi股票普通行情回测工具（简称BackTestApi），BackTestApi股票Lev2行情回测工具会简称为Lev2-BackTestApi。

奇点快速股票交易系统（简称奇点交易系统）是一款快速证券交易柜台系统，并提供交易API和行情Lev2 API（简称奇点Api）接口来编写策略程序。

Lev2-BackTestApi支持基于Lev2行情对“基于奇点交易系统API编写的策略”进行回测。

Lev2-BackTestApi采用与“奇点交易系统”的完全一致的API接口，无缝对接实盘。因此，用Lev2-BackTestApi接口编写的策略程序经过回测后，可以直接用“奇点交易系统API”替换掉Lev2-BackTestApi就可以进行实盘交易。

需要注意的是：Lev2-BackTestApi只能用来做回测，不能用来连接真实的奇点交易系统。

## Lev2-BackTestApi接口文件

Lev2-BackTestApi采用与“奇点交易系统”的完全一致的API接口，并且接口文件的名称也完全相同。

目前，Lev2-BackTestAPI支持平台和编程语言接口：

|  |  |  |
| --- | --- | --- |
| 操作系统 | C++ | Python（**只支持*3.6*版本**） |
| 64位Linux | √ | √ |
| 32位Windows | √ | √ |
| 64位Windows | √ | √ |

C++接口文件列表如下：

|  |  |
| --- | --- |
| 文件名 | 详情 |
| TORATstpTraderApi.h | 包含交易相关的请求和回调接口 |
| TORATstpLev2MdApi.h | 包含获取Lev2行情相关的请求和回调接口 |
| TORATstpUserApiStruct.h | 包含了所有用到的数据结构。 |
| TORATstpUserApiDataType.h | 包含了所有用到的数据类型。 |
| traderapi.dll, traderapi.lib (windows)  libtraderapi.so (linux) | 交易部分的动态链接库和静态链接库 |
| Lev2mdapi.dll, lev2mdapi.lib (windows)  liblev2mdapi.so (linux) | Lev2行情部分的动态链接库和静态链接库 |

Python接口文件列表如下：

|  |  |
| --- | --- |
| 文件名 | 详情 |
| lev2mdapi.py | Lev2行情相关Python接口 |
| \_lev2mdapi.pyd(windows)  \_lev2mdapi.so(linux) | Lev2行情Python转换C++接口组件 |
| traderapi.py | 交易相关Python接口 |
| \_traderapi.pyd(windows)  \_traderapi.so(linux) | 交易Python转换C++接口组件 |
| traderapi.dll(windows)  libtraderapi.so (linux) | 交易部分的动态链接库 |
| lev2mdapi.dll(windows)  liblev2mdapi.so (linux) | Lev2行情部分的动态链接库 |

## Lev2-BackTestApi接口类

CTORATstpTraderApi是交易API，负责主动发起请求查询和订阅交易回报信息的接口函数；CTORATstpTraderSpi含有对应所有的响应和回报函数，用于接收证券柜台交易系统发送的请求响应和回报信息，开发者需要继承该接口类，并实现其中相应的虚函数。

CTORATstpLev2MdApi是Lev2行情API，负责主动发起订阅Lev2行情信息的接口函数；CTORATstpLev2MdSpi含有对应所有的响应和回报函数，用于接收证券柜台交易系统发送的行情信息和查询结果，开发者需要继承该接口类，并实现其中相应的虚函数。

接口函数主要有请求，响应，回报，错误响应这四种。除了请求类型的接口函数属于请求CTORATstpTraderApi或者CTORATstpLev2MdApi类，其余都属于CTORATstpTraderSpi或者CTORATstpLev2MdSpi。

|  |  |  |  |
| --- | --- | --- | --- |
| 消息 | 格式 | 示例 | 所属类 |
| 请求 | Req\*\*\* | ReqUserLogin  ReqQrySecurity | CTORATstpTraderApi或者CTORATstpLev2MdApi |
| 响应 | OnRsp\*\*\* | OnRspUserLogin  OnRspQrySecurity | CTORATstpTraderSpi或者CTORATstpLev2MdSpi |
| 回报 | OnRtn\*\*\* | OnRtnOrder | CTORATstpTraderSpi或者CTORATstpLev2MdSpi |
| 错误回报 | OnErrRtn\*\*\* | OnErrRtnOrderInsert | CTORATstpTraderSpi或者CTORATstpLev2MdSpi |

# Lev2回测前必须知道的信息

## Lev2回测服务地址

Lev2回测服务器的地址与BackTestApi股票普通行情回测相同，即tcp://210.14.72.18:26666；用户必须使用Lev2-BackTestApi中的traderapi的RegisterFront函数来注册此地址去连接上Lev2回测服务器。

## Lev2回测的用户名和密码

Lev2回测的用户名和密码，与BackTestApi股票普通行情回测的用户名密码相同。

## Lev2回测的历史行情

生产Lev2行情类型共有六种：秒级普通行情（不复权），指数行情，逐笔委托，逐笔成交，秒级盘后定价行情（不复权），盘后定价逐笔成交。

在Lev2-BackTestApi回测中，不包括盘后定价行情和盘后定价逐笔成交。而且，生产的上交所Lev2行情目前没有提供逐笔委托。具体情况，请看以下表格。

|  |  |  |
| --- | --- | --- |
| Lev2行情类型名 | 上交所 | 深交所 |
| 秒级普通行情（不复权） | **包括** | **包括** |
| 指数行情 | **包括** | **包括** |
| 逐笔委托 | **不包括** | **包括** |
| 逐笔成交 | **包括** | **包括** |
| 秒级盘后定价行情 | **不包括** | **不包括** |
| 盘后定价逐笔成交 | **不包括** | **不包括** |

## Lev2回测日期跨度

Lev2-BackTestApi回测目前只支持单日回测，不支持多个交易日回测。

## Lev2回测报单类型

目前，Lev2-BackTestApi回测工具支持上证深证的A股的竞价交易业务，支持上证和深证所有的竞价交易报单类型如下：

|  |
| --- |
| 限价单 |
| 五档即时成交剩余撤销 |
| 五档即时成交剩余转限价单(上证) |
| 本方最优(深证) |
| 对手方最优剩余转限价(深证) |
| 市价立即成交剩余撤销(深证) |
| 市价全额成交或撤销(深证) |

## Lev2回测交易成交规则

在Lev2回测中，上交所和深交所报单撮合成交规则，并不相同。

* 上交所的报单撮合成交是由普通十档行情快照和逐笔成交来联合驱动。

1. 首先，根据行情对手方的十档价格来对价成交，按照一档到十档价格的顺序来依次成交，并且每次成交的数量都会消耗掉该行情档位价格的挂单剩余量；
2. 其次，挂单的委托会与后继的逐笔成交来进行对价撮合，如果挂单的委托价格优于逐笔成交的成交价，那么此委托会以逐笔成交的成交价来成交，并且成交数量是取（挂单委托的未成交数量）和（逐笔委托成交数量）的最小值。

* 深交所的报单撮合成交是由逐笔委托来驱动。所有的逐笔委托组成一个报单队列，回测申报的委托也将进入到该报单队列。在报单队列中，回测申报的委托将按照价格优先，时间优先的原则来进行撮合成交。

## Lev2回测缓存机制

缓存Lev2回测历史行情数据，使得反复Lev2回测时效率提升。

# 如何快速开始Lev2回测（C++）

为了快速开始上手回测，本节将会介绍一个windows平台上的简单示例（BackTestAPI\_demo\_c.zip），来演示如何使用BackTestApi去编写和执行一个C++语言编写的回测程序。

示例下载，可以通过登录N视界网站<http://www.n-sight.com.cn>，并访问百宝箱页面<http://www.n-sight.com.cn/u/w/download/download.html>，然后点击BackTestApi的C++ Lev2 Demo按钮去下载。

## 开发环境准备

* 需要安装visual studio 2013以上版本
* Lev2-BackTestApi C++组件traderapi.dll，lev2mdapi.dll，traderapi.lib，lev2mdapi.lib，以及配置文件traderapi.ini都需要放置在执行程序目录下面。

## 设置回测初始参数

首先需要在traderapi.ini配置文件中设置回测的初始化参数。

在本例子的traderapi.ini文件，由于Lev2回测只支持单日回测，因此设置回测时间段为20200203到20200203，初始资金设置为10000000，使用缓存行情。使用缓存行情机制可以避免重复下载同一个股票的相同Lev2行情。

|  |
| --- |
| #订阅历史行情的起始日期  SubMD\_Begin\_date=20200203  #订阅历史行情的结束日期  SubMD\_End\_date=20200203  #用户设置初始资金。如果不设置则使用默认金额10000000  InitMoney=10000000  #是否使用缓存行情，如果不设置则默认使用  UseMDCache=1  #股票佣金费率，如果不设置默认0.0003（万三）  StockFeeRatio=0.0003  #股票最低佣金金额，如果不设置默认0  StockFeeMin=0  #是否上传回测结果到回测服务器，如果不设置则默认上传  IsToUploadResult=1  #初始持仓需要在InitPosition配置项里设置：设置格式 (交易所代码,股票代码,数量),如果#设置多个持仓请用分号分隔，交易所代码1是上海2是深圳。  #例如设置初始持仓浦发5000股，万科6000股。  InitPosition=(1,600000,5000);(2,000002,6000)  #Lev2回测开关，要进行Lev2回测，则必须设置为1  UseLev2=1 |

在traderapi.ini配置文件中，必须要配置行情起始日期“SubMD\_Begin\_date”和结束日期“SubMD\_End\_date”，这两项配置决定了Lev2回测的起止时间段。目前Lev2回测服务只允许单个交易日回测，因此起始日期和结束日期必须为同一个交易日。

## 如何编写回测程序

在本节，将会以示例程序为例（BackTestAPI\_lev2demo\_c.zip），演示一下如何编写一个简单回测程序。此程序由lev2simspi.h，lev2simspi.cpp，lev2demo.cpp三个源代码文件组成。

### 初始化TraderAPI

建立**SimpleTraderSpi**类继承交易响应回调基类**CTORATstpTraderSpi**，并且实现其中所需的回调函数，具体实现请查看**lev2simspi.h**文件。

|  |  |
| --- | --- |
| OnFrontConnected | 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。 |
| OnFrontDisconnected | 当客户端与交易后台通信连接断开时，该方法被调用。 |
| OnRspUserLogin | 登录应答 |
| OnRspOrderInsert | 报单录入应答 |
| OnRspOrderAction | 撤单申报应答 |
| OnRtnOrder | 报单回报通知推送应答，每次投资者报单状态变化都会推送报单回报。 |
| OnRtnTrade | 成交回报通知推送应答，每次投资者申报报单有成交就会推送成交回报。 |
| OnRspQryShareholderAccount | 查询股东账户结果应答 |
| OnRspQryTradingAccount | 查询资金账户结果应答 |
| OnRspQryPosition | 查询持仓结果应答 |
| OnRspQryOrder | 查询报单结果应答 |
| OnRspQryOrderAction | 查询撤单记录结果应答 |
| OnRspQryTrade | 查询成交结果应答 |
| OnRspQryTradingFee | 查询交易所基础税费结果应答 |
| OnRspQryInvestorTradingFee | 查询投资者佣金费率结果应答 |

初始化TraderApi实例，创建继承自**SimpleTraderSpi**类的自定义交易回调类DemoTraderSpi实例，设置私有流和公有流的订阅方式，并且注册Lev2回测服务器的地址。

|  |
| --- |
| CTORATstpTraderApi\* tradeapi = CTORATstpTraderApi::CreateTstpTraderApi();  DemoTraderSpi\* traderspi = new DemoTraderSpi(tradeapi);  tradeapi->SubscribePrivateTopic(TORA\_TERT\_RESUME);  tradeapi->SubscribePublicTopic(TORA\_TERT\_RESUME);  tradeapi->RegisterSpi(traderspi);  tradeapi->RegisterFront(g\_address); |

### 初始化Lev2MdAPI

建立**SimpleLev2MdSpi**类继承Lev2行情响应回调基类**CTORATstpLev2MdSpi**，并且实现其中所需的回调函数，具体实现请查看**lev2simspi.h**文件。

|  |  |
| --- | --- |
| OnFrontConnected | 当客户端与Lev2行情后台建立起通信连接时（还未登录前），该方法被调用。 |
| OnFrontDisconnected | 当客户端与Lev2行情后台通信连接断开时，该方法被调用。 |
| OnRspUserLogin | 登录应答 |
| OnRspSubMarketData | 普通行情订阅应答 |
| OnRspUnSubMarketData | 普通行情取消订阅应答。 |
| OnRspSubIndex | 指数行情订阅应答 |
| OnRspUnSubIndex | 指数行情取消订阅应答 |
| OnRspSubTransaction | 逐笔成交订阅应答 |
| OnRspUnSubTransaction | 逐笔成交取消订阅应答 |
| OnRspSubOrderDetail | 逐笔委托订阅应答 |
| OnRspUnSubOrderDetail | 逐笔委托取消订阅应答 |
| OnRtnMarketData | 普通行情推送通知 |
| OnRtnIndex | 指数行情推送通知 |
| OnRtnTransaction | 逐笔成交推送通知 |
| OnRtnOrderDetail | 逐笔委托推送通知 |

初始化Lev2MdApi实例，创建继承自**SimpleLev2MdSpi**类的自定义Lev2行情回调类DemoLev2MdSpi实例，并且注册回测Lev2服务器的地址。

|  |
| --- |
| CTORATstpMdApi\* mdapi = CTORATstpMdApi::CreateTstpMdApi();  DemoLev2MdSpi\* mdspi = new DemoLev2MdSpi (mdapi, traderspi, submd\_sse\_securities, sse\_submd\_num, submd\_szse\_securities, szse\_submd\_num);  mdapi->RegisterSpi(mdspi);  mdapi->RegisterFront(g\_address); |

### 回测单个交易日

在Lev2回测开始时，必需调用Lev2MdApi和TraderApi的init函数，去连接上回测引擎，并会产生OnFrontConnected回调函数；而当交易日结束时，回测引擎会断开所有的Lev2MdApi和TraderApi，会产生相应OnFrontDisconnected回调函数，而调用的Lev2MdApi和TraderApi的join函数会立即退出并且返回整数值0。

|  |
| --- |
| *time\_t* begin\_time = time(NULL);  while (true)  {  /////////////////////////////////////////////////////////////////////////////////////////////////////////////////  //回测开始时，首先需调用lev2mdapi和traderapi的init函数，去连接上回测引擎，  //并会产生OnFrontConnected回调函数；接着再进行登录报单等操作  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////  tradeapi->Init();  lev2mdapi->Init();  /////////////////////////////////////////////////////////////////////////////////////////////////////////////////  //当回测结束时，回测引擎会断开所有的lev2mdapi和traderapi，会产生相应OnFrontDisconnected回调函数；  //同时如果调用了lev2mdapi和traderapi的join函数，则join函数会立即退出并返回整数值0: /////////////////////////////////////////////////////////////////////////////////////////////////////////////////  int ret1 = lev2mdapi->Join();  int ret2 = tradeapi->Join();  if (0 == ret1 && 0 == ret2)  {  *printf*("Join exit because the whole test is finished timecost=%u \n", (time(NULL) - begin\_time));  *printf*("please press any key to exit\n");  *getchar*();  }  else  {  *printf*("join return value is unexpected\n");  }  break;  } |

### 登录Login

Lev2回测程序必须同时使用Lev2-BackTestApi的Lev2MdApi，TraderApi去完成登录交易引擎和Lev2行情引擎，两者都登录成功后，才能开始进行回测。

在回测TraderApi的OnFrontConnected回调函数，需调用ReqUserLogin函数发送登录请求，并且当调用TraderApi的ReqUserLogin函数去登录时，可以在登录请求报文中的TerminalInfo字段里填写本次回测的名称，还可以用登录请求报文中的UserProductInfo字段来设置回测的历史行情起始日期（traderapi.ini中的SubMD\_Begin\_date字段会失效）。在登录成功后，需要获取股东账号和资金账号等必需信息，在OnRspUserLogin回调函数中，发送查询股东账户请求，查询股东账户完成后，接着去查询资金账户请求，最后再查询持仓。查询过程必须是一次查询结束，再进行下一次查询，不能同时发送多次查询。

|  |
| --- |
| virtual void OnFrontConnected()  {  LOG("trade OnFrontConnected\n");  ReqUserLogin();  }  virtual void OnFrontDisconnected(int nReason)  {  LOG("trade OnFrontDisconnected\n");  }  ////发送登录请求  void ReqUserLogin()  {  /////连接成功后直接去登录  /////调用traderapi的ReqUserLogin函数去登录时，可以在登录请求报文中的TerminalInfo字段里填写本次回测的名称。  CTORATstpReqUserLoginField ReqUserLoginField;  *memset*(&ReqUserLoginField, 0, sizeof(ReqUserLoginField));  *strcpy*(ReqUserLoginField.LogInAccount, g\_userid);  ReqUserLoginField.LogInAccountType = TORA\_TSTP\_LACT\_UserID;  *strcpy*(ReqUserLoginField.Password, g\_passwd);  *strcpy*(ReqUserLoginField.TerminalInfo, g\_testname);  m\_api->ReqUserLogin(&ReqUserLoginField, g\_requestid++);  }  //登录请求响应  virtual void OnRspUserLogin(CTORATstpRspUserLoginField \*pRspUserLoginField, CTORATstpRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)  {  LOG("OnRspUserLogin trader TradingDay=%s LogInAccount=%sErrorID=%d ErrorMsg=%s\n",  pRspUserLoginField->TradingDay, pRspUserLoginField->LogInAccount, pRspInfo->ErrorID, pRspInfo->ErrorMsg);  if (0==pRspInfo->ErrorID)  {  //登录成功后，查询股东账号  ReqQryShareholderAccount();  }  }  ///发送查询股东账户请求  void ReqQryShareholderAccount()  {  CTORATstpQryShareholderAccountField QryShareholderAccountField;  *memset*(&QryShareholderAccountField, 0, sizeof(QryShareholderAccountField));  m\_api->ReqQryShareholderAccount(&QryShareholderAccountField, g\_requestid++);  }  ///股东账号查询  virtual void OnRspQryShareholderAccount(CTORATstpShareholderAccountField \*pShareholderAccount, CTORATstpRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)  {  if (pShareholderAccount)  {  if (TORA\_TSTP\_EXD\_SSE == pShareholderAccount->ExchangeID)  {  *strcpy*(g\_sseshareholderID, pShareholderAccount->ShareholderID);  }  else if (TORA\_TSTP\_EXD\_SZSE == pShareholderAccount->ExchangeID)  {  *strcpy*(g\_szseshareholderID, pShareholderAccount->ShareholderID);  }  }  if (bIsLast)  {  ////完成股东账户查询后，再查询资金账号  ReqQryTradingAccount();  }  }  ////发送查询资金账户请求  void ReqQryTradingAccount()  {  CTORATstpQryTradingAccountField QryTradingAccountField;  *memset*(&QryTradingAccountField, 0, sizeof(QryTradingAccountField));  m\_api->ReqQryTradingAccount(&QryTradingAccountField, g\_requestid++);  }  //资金账户查询  virtual void OnRspQryTradingAccount(CTORATstpTradingAccountField \*pTradingAccount, CTORATstpRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)  {  if (pTradingAccount)  {  *strcpy*(g\_accountid, pTradingAccount->AccountID);  *strcpy*(g\_investorid, pTradingAccount->AccountOwner);  g\_have\_got\_tradingaccount\_shareholderID = true;  }  if (bIsLast)  {  ReqQryPosition();  }  }  ////发送查询持仓请求  void ReqQryPosition()  {  CTORATstpQryPositionField QryPositionField;  *memset*(&QryPositionField, 0, sizeof(QryPositionField));  m\_api->ReqQryPosition(&QryPositionField, g\_requestid++);  }  //请求查询投资者持仓响应  virtual void OnRspQryPosition(CTORATstpPositionField \*pPosition, CTORATstpRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)  {  if (pPosition)  {  LOG("OnRspQryPosition InvestorID=%s TradingDay=%s ExchangeID=%c SecurityID=%s AvailablePosition=%d CurrentPosition=%d PrePosition=%d HistoryPos=%d HistoryPosFrozen=%d TodayBSPos=%d TodayBSFrozen=%d LastPrice=%f TotalPosCos=%f \n",  pPosition->InvestorID, pPosition->TradingDay, pPosition->ExchangeID, pPosition->SecurityID,  pPosition->AvailablePosition, pPosition->CurrentPosition, pPosition->PrePosition,  pPosition->HistoryPos, pPosition->HistoryPosFrozen, pPosition->TodayBSPos, pPosition->TodayBSFrozen,  pPosition->LastPrice, pPosition->TotalPosCost);  }  } |

由于Lev2回测交易是由Lev2行情来驱动的，因此Lev2MdApi在登录成功的OnRspUserLogin回调必须立刻订阅行情。

|  |
| --- |
| ///登录请求响应  virtual void OnRspUserLogin(CTORATstpRspUserLoginField \*pRspUserLogin, CTORATstpRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)  {  LOG("OnRspUserLogin MD LogInAccount=%s ErrorID=%d ErrorMsg=%s\n",  pRspUserLogin->LogInAccount, pRspInfo->ErrorID, pRspInfo->ErrorMsg);  if (pRspInfo->ErrorID == 0)  {  ////登录成功后直接订阅行情，指数行情，逐笔成交，逐步委托。  SubscribeLev2MarketData(m\_sublev2md\_sse\_securities, m\_sse\_sublev2md\_num, m\_sublev2md\_szse\_securities, m\_szse\_sublev2md\_num);  SubscribeLev2Index(m\_sublev2index\_sse\_securities, m\_sse\_sublev2index\_num, m\_sublev2index\_szse\_securities, m\_szse\_sublev2index\_num);  SubscribeLev2Transaction(m\_sublev2transaction\_sse\_securities, m\_sse\_sublev2transaction\_num, m\_sublev2transaction\_szse\_securities, m\_szse\_sublev2transaction\_num);  SubscribeLev2OrderDetail(m\_sublev2order\_sse\_securities, m\_sse\_sublev2order\_num, m\_sublev2order\_szse\_securities, m\_szse\_sublev2order\_num);  }  } |

### 报单申报

本例程序，以9:30:00开始收到的第一笔lev2逐笔成交,用（该笔成交的成交价格+0.01元）来申报买入委托。

|  |
| --- |
| //逐笔成交通知  virtual void OnRtnTransaction(CTORATstpLev2TransactionField \*pTransaction)  {  SimpleLev2MdSpi::OnRtnTransaction(pTransaction);  /////////////////////////////////////////////////////////////////////////////////////////////////////  //以9:30:00开始收到的第一笔lev2成交,用（该笔成交的成交价格+0.01元）来申报买入委托报单。  //上证成交TradeTime是HHMMSSss格式,HH是小时,MM是分钟，SS是秒,ss是指10毫秒倍数(例如36是指360毫秒),上证成交ExecType数值始终为'\0'。  //深证成交TradeTime是HHMMSSsss格式，HH是小时，MM是分钟，SS是秒，sss是毫秒(范围为0至999毫秒)  ////////////////////////////////////////////////////////////////////////////////////////////////////  if ( (!IsHaveInsertOrder(g\_curr\_tradingday, pTransaction->SecurityID)) &&  ((TORA\_TSTP\_EXD\_SSE == pTransaction->ExchangeID && pTransaction->TradeTime >= 9300000) ||  (TORA\_TSTP\_EXD\_SZSE == pTransaction->ExchangeID && pTransaction->TradeTime >= 93000000 && TORA\_TSTP\_ECT\_Fill == pTransaction->ExecType)))  {  TTORATstpLongVolumeType vol = (pTransaction->TradeVolume < 100LL ? 100LL : (100LL \* (pTransaction->TradeVolume / 100LL)));  m\_traderspi->ReqOrderInsert\_Limit(pTransaction->SecurityID, pTransaction->ExchangeID,  (pTransaction->TradePrice + 0.01), vol, TORA\_TSTP\_D\_Buy);  ////在本例中，每个交易日，单个股票就进行一次买入操作  ////设置此股票在当前交易日已经买入过  SetHaveInsertOrder(g\_curr\_tradingday, pTransaction->SecurityID);  }  } |

### 如何运行Lev2回测实例

使用回测时，lev2mdapi.dll，traderapi.dll和traderapi.ini配置文件，都必须和执行程序lev2demo.exe放在同一目录下。

启动命令如下:

lev2demo.exe tcp://210.14.72.18:26666 user passwd

### 查询回测结果

运行完lev2demo回测程序，可以通过登录N视界网站<http://www.n-sight.com.cn>，来查询本次回测结果。查询操作与BackTestApi股票普通行情回测相同。

# 如何快速开始Lev2回测（Python）

本节内容将演示用Lev2-BackTestApi的python接口来重新编写上一节的C++ 示例。

示例下载（BackTestAPI\_lev2demo\_python.zip），可以通过登录N视界网站<http://www.n-sight.com.cn>，并访问百宝箱页面<http://www.n-sight.com.cn/u/w/download/download.html>，然后点击BackTestApi的Python lev2Demo按钮去下载。

## 开发环境准备

* 需要安装Python3.6版本
* 安装Python开发IDE，推荐PyCharm

## 如何运行回测程序

traderapi.ini配置文件与C++示例的配置文件相同，查看回测结果方式也与C++示例相同。

使用回测时，traderapi.dll，lev2mdapi.dll，traderapi.py，lev2mdapi.py，\_traderapi.pyd，\_lev2mdapi.pyd和traderapi.ini配置文件，都必须和执行程序lev2demo.py放在同一目录下。Lev2demo.py是回测程序的源代码。

启动命令如下:

python lev2demo.py tcp://210.14.72.18:26666 user passwd

# Lev2MdApi如何使用详解

Lev2-BackTestApi回测工具中的TraderApi使用方法，是与BackTestApi股票普通行情回测工具完全一致。因此本章节，将重点详细讲解Lev2MdApi的使用。

在Lev2MdApi中，CTORATstpLev2MdApi是Lev2行情API，负责主动发起订阅Lev2行情信息的接口函数；CTORATstpLev2MdSpi含有对应所有的响应和回报函数，用于接收证券柜台交易系统发送的Lev2行情信息，开发者需要继承该接口类，并实现其中相应的虚函数。

## 登录&订阅Lev2行情

调用lev2mdapi的init函数之后，交易连接立即建立后对应lev2行情的回调函数OnFrontConnected会被调用；而在每个交易日回测结束时，Lev2行情连接会断开，则对应交易的回调函数OnFrontDisconnected会被调用。

当Lev行情连接建立时，在对应回调函数OnFrontConnected中，即刻使用函数 ReqUserLogin 请求登录Lev2行情系统。发送登录请求之后，lev2mdapi会接收到登录请求响应，对应登录回调函数OnRspUserLogin会被调用，OnRspUserLogin的第二个参数pRspInfo应答信息的ErrorID值若为0，则表示登录成功，否则登录失败。在行情登录成功之后，在OnRspUserLogin回调函数中需即刻使用Lev2行情订阅函数去订阅所关注证券的Lev2行情。

在Lev2-BackTestApi里面，lev2mdapi支持订阅普通行情(SubscribeMarketData)，订阅指数行情(SubscribeIndex)，订阅逐笔成交(SubscribeOrderDetail)，订阅逐笔委托(SubscribeMarketData)，并且支持上述Lev2行情的取消订阅操作。

目前lev2mdapi在回测中，不支持订阅盘后定价行情 (SubscribePHMarketData)，订阅盘后定价逐笔成交 (SubscribePHTransaction)，订阅逐笔重传成交(SubscribeResendTransaction)，订阅逐笔重传委托(SubscribeResendOrderDetail)。

如果需要回测某个指定股票，那必须在OnRspUserLogin回调函数中即刻去订阅该支股票的普通行情，逐笔成交，逐笔委托（深圳必须订阅，上证不需要订阅），三者缺一不可。

|  |
| --- |
| virtual void OnFrontConnected()  {  LOG("Lev2 MD OnFrontConnected \n");  ReqUserLogin();  }  virtual void OnFrontDisconnected(int nReason)  {  LOG("Lev2 MD OnFrontDisconnected \n");  }  ////发送登录请求  void ReqUserLogin()  {  CTORATstpReqUserLoginField ReqUserLoginField;  *memset*(&ReqUserLoginField, 0, sizeof(ReqUserLoginField));  *strcpy*(ReqUserLoginField.LogInAccount, g\_userid);  ReqUserLoginField.LogInAccountType = TORA\_TSTP\_LACT\_UserID;  *strcpy*(ReqUserLoginField.Password, g\_passwd);  m\_lev2md\_api->ReqUserLogin(&ReqUserLoginField, g\_requestid++);  }  ///登录请求响应  virtual void OnRspUserLogin(CTORATstpRspUserLoginField \*pRspUserLogin, CTORATstpRspInfoField \*pRspInfo, int nRequestID, bool bIsLast)  {  LOG("OnRspUserLogin MD LogInAccount=%s ErrorID=%d ErrorMsg=%s\n",  pRspUserLogin->LogInAccount, pRspInfo->ErrorID, pRspInfo->ErrorMsg);  if (pRspInfo->ErrorID == 0)  {  ////登录成功后直接订阅行情，指数行情，逐笔成交，逐步委托。  SubscribeLev2MarketData(m\_sublev2md\_sse\_securities, m\_sse\_sublev2md\_num, m\_sublev2md\_szse\_securities, m\_szse\_sublev2md\_num);  SubscribeLev2Index(m\_sublev2index\_sse\_securities, m\_sse\_sublev2index\_num, m\_sublev2index\_szse\_securities, m\_szse\_sublev2index\_num);  SubscribeLev2Transaction(m\_sublev2transaction\_sse\_securities, m\_sse\_sublev2transaction\_num, m\_sublev2transaction\_szse\_securities, m\_szse\_sublev2transaction\_num);  SubscribeLev2OrderDetail(m\_sublev2order\_sse\_securities, m\_sse\_sublev2order\_num, m\_sublev2order\_szse\_securities, m\_szse\_sublev2order\_num);  }  } |

## Lev2行情推送

目前，Lev2回测支持普通行情推送，指数行情推送，逐笔成交推送，逐笔委托推送；但不支持盘后定价行情推送，盘后定价逐笔成交推送，订阅逐笔重传成交推送，逐笔重传委托推送。

## 回测lev2mdapi与生产lev2mdapi区别

1. 虽然两者接口相同，但是回测lev2mdapi只支持部分生产lev2mdapi接口的功能，其他未支持的接口内部实现为空。
2. 前者只能用来做回测，不能代替生产lev2mdapi去连接生产Lev2行情系统。
3. Lev2回测必须同时使用Lev2-BackTestApi的lev2mdapi，traderapi去完成登录交易引擎和Lev2行情引擎。而奇点生产API则可以单独使用其中任意一个API。
4. Lev2回测要进行买卖交易的证券品种，必须事先用lev2mdapi的Lev2行情订阅函数来订阅该证券的普通行情，逐笔成交，逐笔委托，否则无法进行回测交易。
5. 奇点生产API的traderapi和lev2mdapi的join函数一旦调用就永远不会退出，除非进程退出。而Lev2-BackTestApi回测过程中，当每个交易日结束时，调用的lev2mdapi和traderapi的join函数会立即退出，且如果所有交易日都回测完之后则join函数返回整数值0，否则返回1。

## 回测lev2mdapi支持生产lev2mdapi的哪些接口

CTORATstpLev2MdApi类支持如下请求接口，其他未支持的接口内部实现为空。

|  |  |  |
| --- | --- | --- |
| **接口名称** | **回测是否支持** | **接口功能介绍** |
| CreateTstpLev2MdApi | 是 | 创建lev2mdapi实例 |
| GetApiVersion | 是 | 获取API版本号 |
| Release | 是 | 删除接口对象实例 |
| Init | 是 | 初始化运行环境,只有调用后,接口才开始工作 |
| Join | 是 | 等待接口线程结束运行 |
| RegisterFront | 是 | 注册lev2行情前置机网络地址 |
| RegisterNameServer | 否 | 注册名字服务器网络地址 |
| RegisterMulticast | 否 | 注册组播地址 |
| RegisterSpi | 是 | 注册回调接口 |
| ReqUserLogin | 是 | 用户登录请求 |
| ReqUserLogout | 是 | 用户登出请求 |
| SubscribeMarketData | 是 | 订阅普通行情 |
| UnSubscribeMarketData | 是 | 退订普通行情 |
| SubscribeIndex | 是 | 订阅指数行情 |
| UnSubscribeIndex | 是 | 退订指数行情 |
| SubscribeTransaction | 是 | 订阅逐笔成交 |
| UnSubscribeTransaction | 是 | 退订逐笔成交 |
| SubscribeOrderDetail | 是 | 订阅逐笔委托 |
| UnSubscribeOrderDetail | 是 | 退订逐笔委托 |
| SubscribePHMarketData | 否 | 订阅盘后定价行情 |
| UnSubscribePHMarketData | 否 | 退订盘后定价行情 |
| SubscribePHTransaction | 否 | 订阅盘后定价逐笔成交 |
| UnSubscribePHTransaction | 否 | 退订盘后定价逐笔成交 |
| SubscribeResendTransaction | 否 | 订阅逐笔重传成交 |
| UnSubscribeResendTransaction | 否 | 退订逐笔重传成交 |
| SubscribeResendOrderDetail | 否 | 订阅逐笔重传委托 |
| UnSubscribeResendOrderDetail | 否 | 退订逐笔重传委托 |