

破冰号交易终端

API 手册

(更新日期: 2013-02-15)

内容目录

1. 欢迎使用破冰号交易终端(API).....	2
2. 安装和卸载.....	2
3. 下单命令详细解释.....	2
4. 程序开发指南.....	3
4.1. Visual C++ (以 VC2010 学习版为例)	3
4.2. C# (以 VC#2010 学习版为例)	7
4.3. Matlab (以 Matlab R2012a 为例)	9
4.4. R 语言 (以 R-2.15.1 为例)	10
4.5. 其他语言.....	11
5. 通过文件方式下单(1.2.7 版本及以上支持)	11
5.1. 首先建立一个文件夹 (比如 C:\ibt_order)	11
5.2. 在破冰号交易终端中设置好监控目录.....	12
5.3. 在行情软件中按规定的下单格式写入到 C:\ibt_order 文件夹的文件里。	12

1. 欢迎使用破冰号交易终端(API)

破冰号交易终端(API)是底层用 C 实现的自动化下单的 API，使用该 API，用户可以直接用 C/C++/C#/Matlab/R/等语言直接下单。

我们致力于为期货行业的投资者提供一个快捷的下单工具，但并不保证该软件能为所有的使用者带来盈利。

感谢您选择破冰号交易终端 API，希望您能够通过使用该系统找到乐趣，并能创造更多价值。

2. 安装和卸载

下载

用户可以登录破冰号交易终端下载页面，找到要下载的相关文件 ibt_client.rar，直接点击下载。

下载地址：<http://code.google.com/p/icebreaker-trader/downloads/list>

安装

直接解压 ibt_client.rar 即可（比如：d:\ibt_client）。

卸载

直接删除 ibt_client 目录即可。

ibt_client 文件夹中文件说明：

- 1) ibt_client.h <== 开发的头文件
- 2) ibt_log4cplus.properties <== log4cplus 日志配置文件
- 3) libibt_client_ex.dll <== dll 文件
- 4) libibt_client_ex.lib <== lib 文件
- 5) libibt_client_test.exe <== API 测试程序
- 6) msvcr100.dll <== 依赖的 VC 库文件
- 7) msvcp100.dll <== 依赖的 VC 库文件

3. 下单命令详细解释

```
int ibt_send_order(int Policy_ID, int BuyOrSell, int EntryOrExit, int Lot);
```

参数说明：

1. Policy_ID - 策略 ID，即破冰号交易终端的配置的策略 ID
2. BuyOrSell - 买入或卖出，IBT_Enum_Buy(0)-买入，IBT_Enum_Sell(1)-卖出
3. EntryOrExit - 开仓或平仓，IBT_Enum_Entry(0)-开仓，IBT_Enum_Exit(1)-平昨，IBT_Enum_ExitToday(2)-平今
4. Lot - 交易手数

返回值:

- 0 - 成功
- 1 - 失败

特别说明:

1. 对于平仓, Lots = 0, 则平掉该策略的所有仓位。
2. 对于平仓, 无论你选择平昨还是平今, 都会平掉相应的 Lot 手数, 选择平今只是优先平今。比如, 你有 5 手昨仓, 3 手今仓, 如果 Lot=8, 则会平掉所有的仓位。如果你选择了平今, Lot=5, 则会平今仓=3, 平昨仓=2。
3. 对于下单指令, 没有商品名和价格, 这两项都在破冰号交易终端中配置。

举例:

1. 开多仓

```
int i_ret = ibt_send_order( Policy_ID, IBT_Enum_Buy, IBT_Enum_Entry, Lot);
```

2. 开空仓

```
int i_ret = ibt_send_order( Policy_ID, IBT_Enum_Sell, IBT_Enum_Entry, Lot);
```

3. 平多仓

```
int i_ret = ibt_send_order( Policy_ID, IBT_Enum_Sell, IBT_Enum_ExitToday, Lot);
```

4. 平空仓

```
int i_ret = ibt_send_order( Policy_ID, IBT_Enum_Buy, IBT_Enum_ExitToday, Lot);
```

4. 程序开发指南

4.1. Visual C++ (以VC2010 学习版为例)

1. 新建一个“Win32 控制台应用程序”项目 test_ibt_client

新建项目

最近的模板

已安装的模板

Visual C++

CLR

Win32

常规

排序依据: 默认值

Win32 控制台应用程序

Win32 项目

Visual C++

Visual C++

搜索 已安装的模板

类型: Visual C++

用于创建 Win32 控制台应用程序的项目

名称(N): test_ibt_client

位置(L): C:\e\svn\

解决方案名称(M): test_ibt_client

浏览(B)...

☒ 为解决方案创建目录(D)

确定

取消

Win32 应用程序向导 - test_ibt_client



应用程序设置

概述

应用程序设置

应用程序类型:

☐ Windows 应用程序(W)

☒ 控制台应用程序(C)

☐ DLL(D)

☐ 静态库(S)

附加选项:

☐ 空项目(E)

☐ 导出符号(X)

☒ 预编译头(P)

添加公共头文件以用于:

☐ ATL(A)

☐ MFC(M)

< 上一步

下一步 >

完成

取消

2. 把 d:\ibt_client 目录加入到包含目录(include 路径)和库目录(lib 路径)中。



3. 复制下列代码到 `test_ibt_client.cpp` 文件

```
#include "stdafx.h"

#include <iostream>

#include "ibt_client.h"

#pragma comment(lib, "libibt_client_ex.lib")

using namespace std;

static int test_send_order();

int _tmain(int argc, _TCHAR* argv[])
{
    test_send_order();

    return 0;
}

int test_send_order()
{
    ibt_debug("test_send_order start...");

    int i_ret = 0;
    //发送开仓命令
```

```

int Policy_ID = 26;
// 买入开仓
/*int BuyOrSell = IBT_Enum_Buy;
int EntryOrExit = IBT_Enum_Entry;
int Lot = 1;*/

// 卖出平仓
int BuyOrSell = IBT_Enum_Sell;
int EntryOrExit = IBT_Enum_ExitToday;
int Lot = 0;

i_ret = ibt_send_order(Policy_ID,BuyOrSell,EntryOrExit,Lot);
if (0 != i_ret)
{
    ibt_error("破冰号客户端发送命令失败!");
}
else
{
    ibt_info("破冰号客户端发送命令成功!");
}

ibt_debug("test_send_order end.");

return i_ret;
}

```

4. 编译 test_ibt_client 项目

5. 运行

1) 把 **ibt_log4cplus.properties**, **libibt_client_ex.dll** 两个文件复制到 test_ibt_client.exe 所在的目录下。

2) 启动“破冰号交易终端”并登录成功

3) 运行 test_ibt_client.exe

2012-12-22 18:09:49.405 INFO [2624] [Trader] IBT_Client 发送命令. 策略 ID=26, BuyOrSell=1, EntryOrExit=2, Lot=0, SentTime=201

21222.180949, i_ret=0

2012-12-22 18:09:49.436 INFO [2624] [ibt_client] 破冰号客户端发送命令成功!

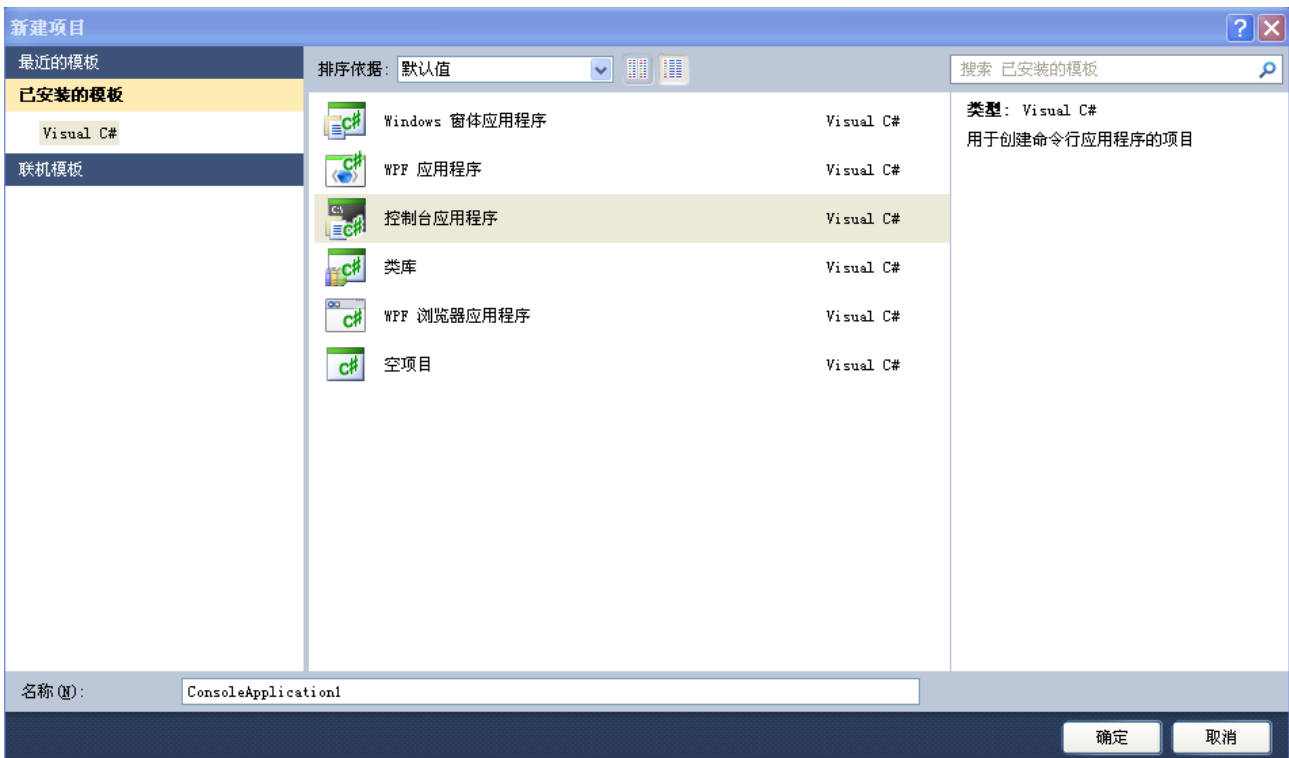
4) 在“破冰号交易终端”中交易日志中可以看到日志:

18:09:49.405 Task_SendOrder --->>> 收到 API 报单请求。 策略 ID=26, 类型=卖平仓, 手数=0, 下单时间=20121222.180949

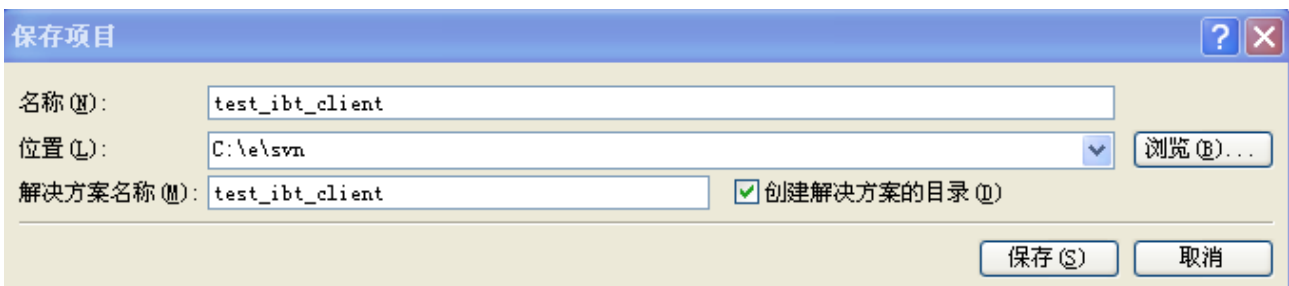
18:09:49.405 Task_SendOrder --->>> 没有找到对应的策略 ID(26), 因此该交易指令无效!

4.2. C# (以VC#2010 学习版为例)

1. 新建一个“控制台应用程序”项目 test_ibt_client



保存该项目



2. 把 ibt_log4cplus.properties, libibt_client_ex.dll 复制到 test_ibt_client\bin\Release 目录下
3. 复制下列代码到 Program.cs 文件

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Runtime.InteropServices; // 用 DllImport 需用此 命名空间
using System.Reflection; // 使用 Assembly 类需用此 命名空间
using System.Reflection.Emit; // 使用 ILGenerator 需用此 命名空间

namespace test_ibt_client
{
    class Program
    {
```

```

[DllImport("libibt_client_ex.dll")]
static extern int ibt_send_order(int Policy_ID, int BuyOrSell, int EntryOrExit, int Lot);
[DllImport("libibt_client_ex.dll")]
static extern void ibt_debug(string p_text);
[DllImport("libibt_client_ex.dll")]
static extern void ibt_info(string p_text);
[DllImport("libibt_client_ex.dll")]
static extern void ibt_error(string p_text);

static void Main(string[] args)
{
    ibt_debug("test_send_order start...");

    int i_ret = 0;
    //发送开仓命令
    int Policy_ID = 26;
    // 买入开仓
    /*int BuyOrSell = IBT_Enum_Buy;
    int EntryOrExit = IBT_Enum_Entry;
    int Lot = 1;*/

    // 卖出平仓
    const int IBT_Enum_Sell = 1;
    const int IBT_Enum_ExitToday = 2;

    int BuyOrSell = IBT_Enum_Sell;
    int EntryOrExit = IBT_Enum_ExitToday;
    int Lot = 0;

    i_ret = ibt_send_order(Policy_ID, BuyOrSell, EntryOrExit, Lot);
    if (0 != i_ret)
    {
        ibt_error("破冰号客户端发送命令失败!");
    }
    else
    {
        ibt_info("破冰号客户端发送命令成功!");
    }

    ibt_debug("test_send_order end.");
}
}

```

4. 编译

5. 运行

- 1) 启动“破冰号交易终端”并登录成功
- 2) 运行 test_ibt_client.exe

2012-12-22 18:09:49.405 INFO [2624] [Trader] IBT_Client 发送命令. 策略ID=26, BuyOrSell=1, EntryOrExit=2, Lot=0, SentTime=201

21222.180949, i_ret=0

2012-12-22 18:09:49.436 INFO [2624] [ibt_client] 破冰号客户端发送命令成功!

3) 在“破冰号交易终端”中交易日志中可以看到日志:

18:09:49.405 Task_SendOrder --->>> 收到API报单请求。 策略ID=26, 类型=卖平仓, 手数=0, 下单时间=20121222.180949

18:09:49.405 Task_SendOrder --->>> 没有找到对应的策略ID(26), 因此该交易指令无效

4.3. Matlab (以Matlab R2012a为例)

1. 解压ibt_client.rar文件到一个目录(假如把该文件解压到d:\ibt_client)

2. 把ibt_log4cplus.properties复制到MATLAB的bin目录下。

假如MATLAB安装在d:\MATLAB\R2012a, 则复制到: d:\MATLAB\R2012a\bin目录下

3. 设置编程环境

```
rehash toolboxcache
```

```
mex -setup
```

注: 此步骤只需要运行一次。

4. 加载动态库(假如把该文件解压到一个目录, 比如d:\ibt_client)

```
loadlibrary('d:\ibt_client\libibt_client_ex.dll','d:\ibt_client\ibt_client.h');
```

5. 调用下单函数

```
[a1] = calllib('libibt_client_ex','ibt_send_order',2,0,0,1);
```

6. 卸载动态库

```
unloadlibrary('libibt_client_ex');
```

7. 查看动态库中所有函数

```
libfunctions libibt_client_ex -full
```

8. 调用其他函数

1) 显示启动路径

```
[a1] = calllib('libibt_client_ex','ibt_current_dir');
```

2) 打印调试信息

加载log配置文件, 这一步可以不用做, 如果要做, 则必须在加载动态库后就做。

```
calllib('libibt_client_ex','ibt_init_ex',  
        'd:\ibt_client\ibt_log4cplus.properties');
```

打印DEBUG信息

```
calllib('libibt_client_ex','ibt_debug','hell world');
```

打印INFO信息

```
calllib('libibt_client_ex','ibt_info','hell world');
```

打印ERROR信息

```
calllib('libibt_client_ex','ibt_error','hell world');
```

3) 查看日志

所有的日志都保存在在 MATLAB 的 bin 目录下的 ibt_normal.log 文件里面

在 MATLAB 的 bin 目录下的 ibt_trader.log 是交易日志文件，如果你发了交易指令，则可以在 ibt_trader.log 中查看到。

4.4. R 语言 (以 R-2.15.1 为例)

特别说明，R 语言比较特殊，在调用 dll 时，必须传指针过去，在 libibt_client_ex.dll 中，**必须调用以“_r”为结尾的函数，否则会出错。**

1. 解压 ibt_client.rar 文件到一个目录（假如把该文件解压到 d:\ibt_client）
2. 启动 RGUI。
3. 切换当前目录到 d:\ibt_client

```
setwd("d:/ibt_client");
```

注：获得当前路径：getwd();

4. 加载动态库

```
dyn.load("libibt_client_ex.dll");
```

5. 调用下单函数

```
out=.C("ibt_send_order_r",Policy_ID=as.integer(2),BuyOrSell=as.integer(1),EntryOrExit=as.integer(2),Lot=as.integer(1),ret=as.integer(0));
```

注：传给 dll 的数字，必须用 as.integer() 做转换，否则会出错。

6. 卸载动态库

```
dyn.unload("libibt_client_ex.dll");
```

7. 调用其他函数

- 1) 显示启动路径

```
out=.C("ibt_current_dir_r",dir="NULL");
```

```
out$dir;
```

- 2) 打印调试信息

加载 log 配置文件，这一步可以不用做，如果要做，则必须在加载动态库后就做，否则会出错。

```
out=.C("ibt_init_ex_r",dir="d:/ibt_client/ibt_log4cplus.properties",ret=as.integer(0));
```

打印 DEBUG 信息

```
out=.C("ibt_debug_r","hello world");
```

打印 INFO 信息

```
out=.C("ibt_info_r","hello world");
```

打印 ERROR 信息

```
out=.C("ibt_error_r","hello world");
```

3) 查看日志

所有的日志都保存在 d:/ibt_client 目录下的 ibt_normal.log 文件里面

在 d:/ibt_client 目录下的 ibt_trader.log 是交易日志文件，如果你发了交易指令，则可以在 ibt_trader.log 中查看到。

附：R 语言调用 C 语言写的 DLL 的一些约定。

1、编写的 C 函数不能有显式返回值，即函数返回要声明为 void；

2、要得到函数的计算结果，只能通过传址实参的方式，在 C 参数中传递某个指针作为计算结果；

3、函数返回一个 list，其中每个元素对应同名字的参数。

4、**R 中传参时参数的类型为 vector，解释到 C 处对应就是指针。所以 C 函数参数必须声明为指针，这点要切记，否则会带来意想不到的结果。**

5、传递数值向量时，默认为 double 型，可通过 as.integer 函数转换为 integer 型。

6、例子中 R 调用的工作原理：参数被拷贝一份，传递到 C 函数，计算，各参数拷贝到 out 这个 list，返回。所以原工作区的变量不会被改变。

7、注意数据类型的显式声明，用 as.* 函数来显式注明要传递的参数类型，以免到 C 处发生不可预料的错误。

4.5. 其他语言

ibt_client 提供的是 C 语言 API 接口，理论上，只要能调用 C 语言 DLL 的语言都能使用 ibt_client API，如果你需要其他语言调用 ibt_client API，则请查看该语言的调用 DLL 规范。或者使用 SWIG 支持的语言的接口调用 ibt_client API，详情请见 (<http://www.swig.org/>)

5. 通过文件方式下单(1.2.7 版本及以上支持)

由于很多行情系统并不支持 DLL 方式，因此无法直接用 API 下单；但是这些行情系统支持通过写文件的方式输出日志。因此我们可以通过写文件的方式下单。

下面详细介绍“通过写文件的方式下单”步骤。

5.1. 首先建立一个文件夹（比如 C:\ibt_order）

特别说明：

1) **此文件夹需要用户自己创建**，破冰号交易终端不负责创建该文件夹。

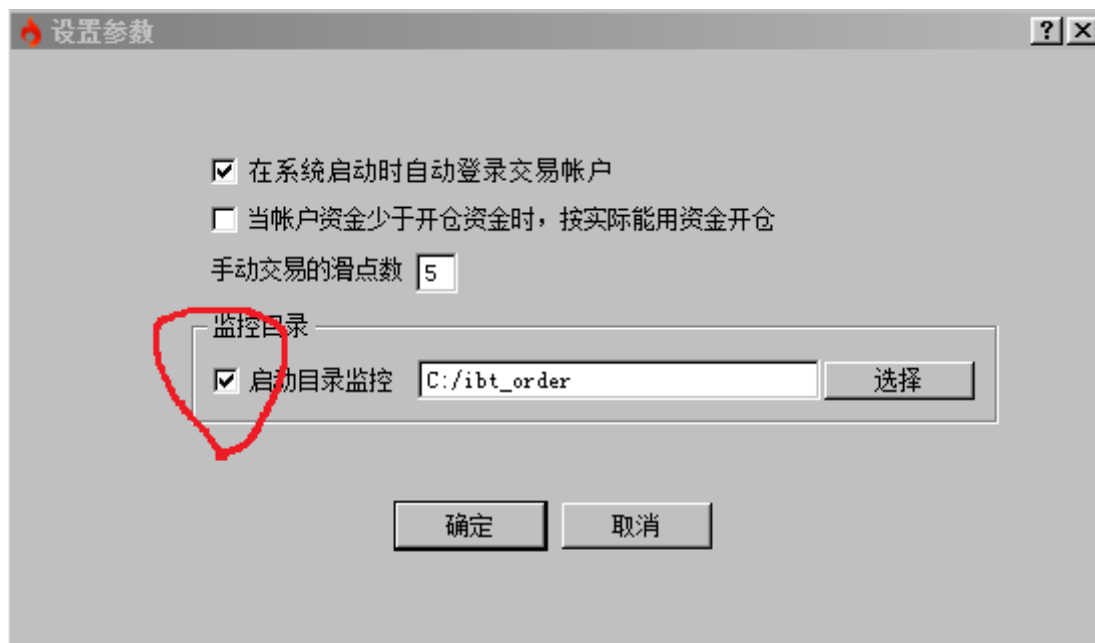
2) 破冰号交易终端只监控该文件夹的当前文件夹，并不监控该文件夹的子文件夹，即所有的下单文件必须放在该文件夹下。

3) 下单文件名可以是任何文件名，只要文件的内容符合下单格式即可。

4) 破冰号交易终端每次启动的时候，都会把该文件夹中的文件移动到当前目录下的 old 文件夹下。

5.2. 在破冰号交易终端中设置好监控目录

如图：



5.3. 在行情软件中按规定的下单格式写入到 C:\ibt_order 文件夹的文件里。

下单格式：

Policy_ID, BuyOrSell, EntryOrExit, Lot, Order_Time

文件格式举例：

买入平仓（平空）（策略 ID = 26，交易手数 = 1 手）

26, 0, 2, 1, 20130208.150900

买入开仓（开多）（策略 ID = 26，交易手数 = 1 手）

26, 0, 0, 1, 20130208.150900

卖出平仓（平多）（策略 ID = 26，交易手数 = 1 手）

26, 1, 2, 1, 20130208.151300

卖出开仓（开空）（策略 ID = 26，交易手数 = 1 手）

26, 1, 0, 1, 20130208.151300

特别说明：

- 1) 每个下单命令占 1 行，可以有多个命令同时出现在一个文件中（即可以有多行）。
- 2) 每次写入文件必须采用覆盖方式写入，而不是追加方式，因为破冰号交易终端每次都是读出该文件的所有内容，并执行。