**第三方接口需求**

1. **业务需求**
2. 控制与控制应答
3. 登录：第三方程序向FastTrader发送登录申请，FastTrader接收到后，返回消息标识登录成功与否。当第三方程序收到返回信息后，根据相关数据进行初始化操作。（如有需要，可以进行定制或过滤一些命令和信息，比如常见的行情推送过滤、下单相关的应答推送）
4. 握手：第三方程序收到登录成功信息后，继续向FastTrader发送尝试握手申请，FastTrader收到申请后，回复同意握手的命令。第三方程序接收到此回复，代表通道正常，可以使用此通道进行业务命令发送。
5. 心跳：第三方程序通过发送心跳指令（间隔不能小于10秒，不能大于180秒）保持与FastTrader的联系。FastTrader收到后返回心跳应答。 FastTrader如果长时间（例如超过180秒）未收到心跳命令，则判断为超时，注销此心跳所代表程序的登录状态。第三方程序需要重新进行登录操作。
6. 注销：第三方程序发送注销指令，FastTrader收到后返回注销成功。
7. 查询相关与查询相关应答
8. 查询资金信息：第三方程序发送查询资金信息命令，FastTrader收到后，一次性返回查询状态及具体数据。
9. 查询全部合约信息：同上
10. 查询全部委托单信息： 同上
11. 查询全部持仓信息： 同上
12. 查询交易所当前日期时间：同上
13. 查询某单个合约的深度行情（主动查询）：同上
14. 查询某单个合约的合约信息（主动查询）：同上
15. 查询某单个合约的持仓信息（主动查询）：同上
16. 查询持仓模块光标选中行的合约、方向、开平、账号等数据：同上
17. 查询报价表模块选中合约的到期时间、交易时间段、是否模拟行情：同上
18. 行情相关与行情相关应答
19. 订阅行情合约：第三方程序发送订阅合约行情命令，FastTrader收到后，会先判断第三方程序已订阅的合约数量，如果订阅数量达到限制（底层逻辑判断），会返回错误。如成功返回订阅成功。
20. 推送行情合约：FastTrader每收到一条行情时，根据订阅情况，向第三方程序进行推送。无返回。
21. 退订行情合约：第三方程序发送退订合约行情命令，FastTrader收到后，返回退订成功信息。
22. 订单相关与订单相关应答
23. 下单：第三方程序发送下单命令，FastTrader收到后，进行下单操作， API直接返回信息后判断成功与否，并返回给第三方程序。如果是失败，需要附加错误码和失败原因。
24. 委托单错误状态应答：如果下单在经纪公司或交易所失败了，则推送消息给第三方程序，通知具体的错误码和失败原因
25. 委托单状态应答：当委托单状态改变时，会推送这个信息给第三方程序，附加改变后的委托单数据。
26. 成交应答：当有成交回报时，会推送这个消息给第三方程序，附加成交数据。
27. 撤单：第三方程序发送撤单命令，FastTrader收到后，进行撤单操作，API直接返回信息后判断成功与否，并返回给第三方程序。如果是失败，需要附加错误码和失败原因。
28. 撤单错误状态应答：如果撤单在经纪公司或交易所失败了，则推送消息给第三方程序，通知失具体的错误码和失败原因
29. **FastTrader第三方插件规范**
30. 第三方插件为独立的exe程序，其加载步骤如下：
31. 第三方插件启动时，获取启动参数，并转化为窗口句柄，备后面与FastTrader进程通讯使用
32. 第三方插件发送登录消息给FastTrader进程，进行有效性验证，登录消息内容包括：消息类型、插件标识符、License、插件顶层窗口句柄、插件期望显示的位置。
33. 第三方插件接收登录反馈消息，若为登录成功，则执行步骤4，否则，退出第三方插件进程。
34. 设置第三方插件窗体属性，去除标题栏，禁止插件顶层窗体拖动。
35. 与FastTrader进程进行业务通讯。
36. 结束进程。
37. FastTrader应用第三方插件流程图如下图：
38. FastTrader与第三方插件通讯参数定义如下：
39. 启动参数

只有一个参数，long型，代表FastTrader通讯窗口句柄，供插件与FastTrader进程通讯使用。

1. 消息类型定义

0x00000001 // 登录消息

0x00000002 // 登录反馈消息

0x00000003 // 隐藏插件窗口消息

0x00000004 // 显示插件窗口消息

1. 显示位置

enum ShowPos

{

POS\_NORMAL, // 对位置无要求

POS\_LEFT, // 左边显示

POS\_RIGHT, // 右边显示

POS\_TOP, // 顶端显示

POS\_BOTTOM, // 底端显示

POS\_QUOTE, // 报价表位置显示

};

1. 登录消息结构

struct LogonStu

{

char szAppID[100]; // 插件的唯一描述符，标识插件身份

char szLicense[100]; // 插件License

HWND hWnd; // 插件顶层窗口句柄

ShowPos ePos; // 插件期望显示的位置

};

1. 登录反馈消息

BOOL类型，TRUE表示登录成功，FALSE表示登录失败。

1. 使用CopyData发送消息

COPYDATASTRUCT结构定义为：

typedef struct tagCOPYDATASTRUCT {

DWORD dwData;

DWORD cbData;

PVOID lpData;

} COPYDATASTRUCT;

其中，dwData存储消息类型，cbData存储消息程度，lpData存储消息指针。

附件：第三方接口协议。仅供参考

FastTrade 第三方接口协议

杨桂明/2011.8.11

本文概述的是FastTrade提供给第三方的接口协议，第三方程序员可以通过编写程序的方式实现接口协议，达到与FastTrade进行通信的目的。

通信技术：

使用Windows命名管道的字节模式。

协议需要用两条管道（上行管道、下行管道）来完成，同时，在插件管理器加载第三方插件时会传入一个FastTrader主窗体句柄，通过这个句柄获得FastTrader的PID。

上行管道，也称为Master pipe，是FastTrader用于接收第三方客户端消息的管道，第三方所有的上行消息都发送到这个管道中，由FastTrader统一处理。管道名称为"\\[x]\pipe\FastTrader\v0\_1\Master\[PID]"，"[x]"表示IP地址或主机名，如果是本机地址，也可以用"."来表示。

例如，如果FastTrader的PID为1f6d，这实际使用的MasterPipe的名为"\\.\pipe\FastTrader\v0\_1\Master\1f6d"，大三方使用这个字符串来进行管道通信。

下行管道，也称为App pipe，由第三方程序创建。下行管道的名称从Master pipe中获取，当第三方程序正确的登录到FastTrade后，FastTrader会通过Master pipe返回一个下行管道的名称给第三方程序，第三方程序根据这个名称创建一个命名管道。而FastTrade会向这个管道推送用来接收FastTrade提供的所有业务数据。

技术限制：

本协议使用Windows模式的命名管道，故一般情况下只支持windows下的程序开发。或者可以与windows命名管道正常通信的平台系统。

关于开发语言，所有可以支持windows命名管道开发技术的语言都可以使用，例如C++、C#、VB、VB.NET、Delphi等，java则可以使用如JNI的方式实现与windows命名管道通信。

字符集：GB2312

本协议最大支持20个第三方客户端。

1. 协议头

typedef struct

{

DWORD SyncID; //0xac34cd12

ISessionIDType SessionID; //sessionID

DWORD MsgType; //消息类型

DWORD CMDID; //命令ID

DWORD dwRspMsgNum; //返回的消息编号

DWORD Length; //包长度，包含的协议头和协议尾，即sizeof(PackageHead + Data + PackageTail)

DWORD CRC; //校验，=0

} IPackageHeadStru;

无论是FastTrader给第三方程序，还是第三方程序发给FastTrader消息，必须以次作为开头。

说明：

参数：SyncID

固定取值：0xac34cd12

包头标志。失去同步时，通过这个标志、包尾标志来重新同步。现在你可以不管失步的问题，只用它来判断一个包是否合法。

参数：SessionID

FastTrader给第三方的Session编号，如果不知道Session编号，则填一个"0"，这是个字符串，当Session编号为0时，能被FastTrader接收并处理的只有登录命令。

参数：MsgType，消息类型，不分大小写，有如下几类取值。

A，控制消息，如登录等。

B，控制应答消息，如同意握手等。

C，业务查询消息，包括资金查询、合约查询、持仓查询、委托单查询、成交记录查询等。

D，业务查询应答消息，包括交易所状态、合约信息、资金信息、持仓信息、委托单信息、成交记录信息等的返回。

E，行情相关的消息，包括订阅、退订消息等。

F，行情相关的应答消息，包括行情推送等。

G，订单相关消息。

H，订单相关应答消息。

FastTrader对消息类型的应答模式

A，这类消息由第三方向FastTrader发送(Master Pipe)，FastTrader使用同一个Pipe立即进行应答。

C、E、G，这几类消息由第三方向FastTrader发送(Master Pipe)，FastTrader不进行立即应答，会通过相关的命令字进行应答，并把消息发送到客户端的App Pipe中。

B、D、F、H，这几类应答消息有FastTrader向第三方发送(App Pipe)，第三方接收后进行处理，不需要进行应答。

参数：dwRspMsgNum，应答消息号，B、D、F、H类命令中有意义。0表示成功或无错误，非0则出现各种异常情况，具体看不同的命令字，不同的命令字有不同的定义。

对于A、C、E、G类命令，填为0。

参数：CMDID

命令的编号，用于标识当前传递的是什么命令。

FastTrader的命令总体上分为控制类型和业务类型两种；命令按发送方的不同又分为客户端命令和服务器命令。客户端命令，按命令的处理模式又立即分回写命令和异步命令两种。

立即回写命令

由第三方程序向FastTrader发送(Master Pipe)，FastTrader使用同一个Pipe(Master Pipe)立即进行应答或回复。

异步命令

由第三方程序向FastTrader发送(Master Pipe)，FastTrader不通过同一个Pipe(Master Pipe)立即进行应答，会通过相关的命令，把应答或回复发送到客户端的App Pipe中。

1. 协议尾

Typedef DWORD PackageTailType;

固定取值为：0x12342321

1. 报文数据结构

报文数据结构：PackageHead + Data + PackageTail

命令字有很多，下面对命令字一一介绍。

1. **控制命令与控制应答命令**
2. 登录

SessionID=0

MsgType=1

CMDID=1

dwRspMsgNum=0

附加数据内容与格式

Struct {

Char[64] strAppID;

Char[64] strLicense;

Char[64] strHostName; // 客户端计算机名

Char[32] strUserName; // 用户名

Char[32] strPassword; // 密码

Char[32] strTraderAccount; // 交易帐号

Char cUseQuot; // 使用行情，'1'表示使用，'0'表示不用

Char cUseOrder; // 使用下单交易，'1'表示使用，'0'表示不用

}

FastTrader立即返回一调消息，标识登录成功与否。

SessionID=[cltSession]

MsgType=1

CMDID=2

dwRspMsgNum=0、1、2、999

dwRspMsgNum详解

0，表示登录成功，并再附加一个数据结构1×pipeLoginReturnStruct

1，登录系统繁忙，等待3秒后再发起登录，无附加数据

2，在线用户已达到上限，无法登录，无附加数据

999，未知错误，无附加数据

返回的数据结构

Typedef Struct {

Char[256] strMessage; // 返回消息字符串

Char[256] strAppPipeName; // AppPipe，一个names pipe

} pipeLoginReturnStruct;

注：当登录成功后，即[MsgNum]==0时，第三方程序需要创建一个命名管道，管道的名字为strAppPipeName的内容，并建立对这个命名管道的侦听，FastTrader会通过这个命名管道发送业务信息给第三方程序。

1. 注销

SessionID=[cltSession]

MsgType=1

CMDID=3

dwRspMsgNum=0

无附加数据

1. 尝试握手

SessionID=[cltSession]

MsgType=1

CMDID=4

dwRspMsgNum=0

无附加数据

FastTrader会尝试向对应的App pipe发送异步命令，说明同意进行握手。当收到愿意握手的命令时，表示与FastTrader可以正常的通信了。否则定时尝试握手，直到收到FastTrader发来的愿意握手命令为止。

1. 同意握手

当FastTrader收到尝试握手命令时，会通过App Pipe应答一个愿意握手的命令给第三方程序，收到这个命令，表示所有的通信通道已正常。这时，第三方程序可以进行正常的业务命令了。

SessionID=[cltSession]

MsgType=2

CMDID=5

dwRspMsgNum=0

无附加数据

1. 心跳

第三方程序的心跳间隔不能小于10秒，不能大于180秒

SessionID=[cltSession]

MsgType=1

CMDID=6

dwRspMsgNum=0

无附加数据

1. 心跳应答

SessionID=[cltSession]

MsgType=2

CMDID=7

dwRspMsgNum=0

0，表示成功，并返回结构2×DWORD的数据，第一个DWORD为Master向AppPipe发送数据成功的次数，第二个DWORD为Master向AppPipe发送数据失败的次数。

1. **查询相关命令与查询相关的应答命令**
2. 查询资金信息

SessionID=[cltSession]

MsgType=3

CMDID=100

dwRspMsgNum=0

无附加数据

1. 返回资金账户信息

SessionID=[cltSession]

MsgType=4

CMDID=101

dwRspMsgNum=0、1、999

dwRspMsgNum详解

0，表示查询成功，并返回结构1×PlatformStru\_TradingAccountInfo的数据

1，查询系统繁忙，等待3秒后再发起查询，无其他附加数据

999，未知错误，无其他附加数据

1. 查询合约信息

SessionID=[cltSession]

MsgType=4

CMDID=102

dwRspMsgNum=0

无附加数据

1. 返回合约信息

SessionID=[cltSession]

MsgType=4

CMDID=103

dwRspMsgNum=0、1、999

dwRspMsgNum详解

0，表示查询正常结束，返回结构n×PlatformStru\_InstrumentInfo的数据，其中

ｎ＝(PackageHead.Length-sizeof(PackageHead)-sizeof(PackageTail)-

sizeof([MsgNum])/PlatformStru\_InstrumentInfo

1，查询系统繁忙，等待3秒后再发起查询，无其他附加数据

999，未知错误，无其他附加数据

1. 查询委托单信息

SessionID=[cltSession]

MsgType=3

CMDID=104

dwRspMsgNum=0

无附加数据

1. 返回委托单信息

SessionID=[cltSession]

MsgType=4

CMDID=105

dwRspMsgNum=0、1、999

dwRspMsgNum详解

0，表示查询正常结束，返回结构n×PlatformStru\_OrderInfo的数据，其中

ｎ＝(PackageHead.Length-sizeof(PackageHead)-sizeof(PackageTail)-

sizeof([MsgNum])/PlatformStru\_OrderInfo

1，查询系统繁忙，等待3秒后再发起查询，无其他附加数据

999，未知错误，无其他附加数据

1. 查询持仓信息

SessionID=[cltSession]

MsgType=3

CMDID=106

dwRspMsgNum=0

无附加数据

1. 返回持仓信息

SessionID=[cltSession]

MsgType=4

CMDID=107

dwRspMsgNum=0、1、999

dwRspMsgNum详解

0，表示查询正常结束，返回结构n×PositionStru的数据，其中

ｎ＝(PackageHead.Length-sizeof(PackageHead)-sizeof(PackageTail)-

sizeof([MsgNum])/PositionStru

1，查询系统繁忙，等待3秒后再发起查询，无其他附加数据

999，未知错误，无其他附加数据

1. 查询交易所当前日期时间

SessionID=[cltSession]

MsgType=3

CMDID=108

dwRspMsgNum=0

附加结构1×char[32]的数据，用来存储交易所ID

1. 查询交易所当前日期时间返回

当第三方程序请求查询交易所日期时间时，FastTrader通过这个命名返回交易所的日期时间。

SessionID=[cltSession]

MsgType=4

CMDID=109

dwRspMsgNum=0、1、999

dwRspMsgNum详解

0，查询成功，附加结构1×char[32]的数据，存储日期时间，日期时间为"'交易所日期' '交易所当前时间' 'TickCount'"

1，行情系统繁忙，等待3秒后再发起订阅，无其他附加数据

999，未知错误，无其他附加数据

1. **行情相关的命令与行情相关的应答命令**
2. 订阅合约行情

SessionID=[cltSession]

MsgType=5

CMDID=200

dwRspMsgNum=0

附加结构1×char[64]的数据，用来存储合约ID

1. 订阅返回

当第三方程序订阅一个合约的行情时，FastTrader会检查请求的第三方程序已订阅的合约数量，如果订阅数达到限制，会返回[MsgNum]=2的消息给第三方程序。

SessionID=[cltSession]

MsgType=6

CMDID=201

dwRspMsgNum=0、1、2、999

dwRspMsgNum详解

0，订阅成功，无附加数据

1，行情系统繁忙，等待3秒后再发起订阅，无其他附加数据

2，订阅数达到极限，不能再订阅行情，无其他附加数据

999，未知错误，无其他附加数据

1. 推送合约行情

当收到一条行情时，会向第三方发送。

SessionID=[cltSession]

MsgType=6

CMDID=202

dwRspMsgNum=0

dwRspMsgNum详解

0，表示推送一条行情数据，附加结构1×DataRtnDepthMarketData的数据

1. **订单相关命令与订单相关应答命令**
2. 进行一次下单

SessionID=[cltSession]

MsgType=7

CMDID=300

dwRspMsgNum=0

附加结构1×PlatformStru\_InputOrder的数据

1. 下单返回

下单被正常接受，返回0；

如果下单直接失败了，则通过这个命令字返回消息给第三方程序，通知失败原因，具体的错误码和失败原因在结构CThostFtdcRspInfoField的数据中

SessionID=[cltSession]

MsgType=8

CMDID=301

dwRspMsgNum=0、999

dwRspMsgNum详解

0，表示下单被正常接受，附加结构1×PlatformStru\_InputOrder的数据

999，未知错误，附加结构１×CThostFtdcRspInfoField的数据

1. 委托单错误状态应答

如果下单在经纪公司或交易所失败了，则通过这个命令字返回消息给第三方程序，通知失败原因，具体的错误码和失败原因在结构DataRspOrderInsert的数据中

SessionID=[cltSession]

MsgType=8

CMDID=302

dwRspMsgNum=0

dwRspMsgNum详解

0，表示一条正常的委托单错误状态应答，附加结构1×DataRspOrderInsert的数据

1. 进行一次撤单

SessionID=[cltSession]

MsgType=7

CMDID=303

dwRspMsgNum=0

附加结构1×PlatformStru\_InputOrderAction的数据

1. 撤单返回

撤单被正常接受，返回0；

如果撤单直接失败了，则通过这个命令字返回消息给第三方程序，通知失败原因，具体的错误码和失败原因在结构CThostFtdcRspInfoField的数据中

SessionID=[cltSession]

MsgType=8

CMDID=304

dwRspMsgNum=0、999

dwRspMsgNum详解

0，表示撤单被正常接受，附加结构1×PlatformStru\_InputOrderAction的数据

999，未知错误，附加结构1×CThostFtdcRspInfoField的数据

1. 撤单错误状态应答

如果撤单在经纪公司或交易所失败了，则通过这个命令字返回消息给第三方程序，通知失败原因，具体的错误码和失败原因在结构DataRspOrderAction1的数据中

SessionID=[cltSession]

MsgType=8

CMDID=305

dwRspMsgNum=0

[MsgNum]=0

[MsgNum]详解

0，表示一条正常的委托单错误状态应答，附加结构1×DataRspOrderAction1的数据

1. 委托单状态应答

当委托单状态改变时，回推送这个信息给第三方程序

SessionID=[cltSession]

MsgType=8

CMDID=306

dwRspMsgNum=0

[MsgNum]=0

[MsgNum]详解

0，表示一条正常的委托单应答，附加结构1×DataRtnOrder的数据

1. 成交应答

SessionID=[cltSession]

MsgType=8

CMDID=307

dwRspMsgNum=0

[MsgNum]=0

[MsgNum]详解

0，表示一条正常的成交应答，附加结构1×DataRtnTrade的数据

**附: 如何使用管道的代码**

1. // 连接到FastTrader的代码

CString strMasterPipe(MASTERNAMEDPIPESTRING);

strMasterPipe.Replace(".", m\_strMasterHost);

// 等待MasterPipe可用

if(!WaitNamedPipe(strMasterPipe, NMPWAIT\_NOWAIT)) {

// 管道忙，处理无法连接的问题

return;

}

// 连接到MasterPipe

HANDLE hPipe = CreateFile(strMasterPipe.GetBuffer(0), GENERIC\_READ | GENERIC\_WRITE,

0, NULL, OPEN\_EXISTING, 0, NULL);

if(hPipe == INVALID\_HANDLE\_VALUE) {

// 处理创建连接到MasterPipe失败的问题

//if(GetLastError() != ERROR\_PIPE\_BUSY)

return;

}

// 设置读写模式为BYTE

dwMode = PIPE\_READMODE\_BYTE;

SetNamedPipeHandleState(hPipe, &dwMode, NULL, NULL);

// 向MasterPipe写数据

fSuccess = WriteFile(hPipe, cbBuff, packHead.Length, &cbWritten, NULL);

// 一般只有登录命令有直接返回数据，才使用一下代码

// 检查MasterPipe返回的数据长度，

PeekNamedPipe(hPipe, NULL, 0, NULL, &dwTotalBytesAvail, NULL);

// 如果数据长度合适，则接收数据

if(dwTotalBytesAvail > sizeof(IPackageHeadStru)) {

// 从MasterPipe中读数据

ReadFile(hPipe, pBuff, dwTotalBytesAvail, &cbBytesRead, NULL);

}

// 登录读取直接返回的数据代码到处为止

// 关闭与MasterPipe的连接

CloseHandle(hPipe);

2. // 创建AppPipe的代码

if((g\_hAppPipe = CreateNamedPipe(pRspLogin->strAppPipeName,

PIPE\_ACCESS\_DUPLEX, PIPE\_TYPE\_BYTE | PIPE\_READMODE\_BYTE | PIPE\_WAIT,

PIPE\_UNLIMITED\_INSTANCES, IBUFSIZE, IBUFSIZE, 3000, NULL))

!=INVALID\_HANDLE\_VALUE) {

DWORD dwAppPipeThreadId = 0;

// 创建一个AppPipe的服务线程

g\_hAppPipeThread = CreateThread(NULL, 0, proc\_AppPipe,

NULL, 0, &dwAppPipeThreadId);

}

3. // AppPipe服务代码

// 等待客户端连接

if(!ConnectNamedPipe(g\_hAppPipe, NULL)) {

if(GetLastError() == ERROR\_PIPE\_CONNECTED)

break;

Sleep(1);

}

// 检查客户端写入的数据长度

PeekNamedPipe(g\_hAppPipe, NULL, 0, NULL, &dwTotalBytesAvail, NULL);

if(dwTotalBytesAvail > sizeof(IPackageHeadStru)) {

// 读取数据

ReadFile(g\_hAppPipe, pReqBuff, dwTotalBytesAvail, &cbBytesRead, NULL);

if(cbBytesRead != dwTotalBytesAvail) {

DisconnectNamedPipe(g\_hAppPipe);

continue;

}

// 这里添加协议解析代码

// ...

}

// 断开连接，为新的接入做准备

DisconnectNamedPipe(g\_hAppPipe);