BUBASE框架使用手册

# 框架的整体及目录结构

 图1 整体结构

框架DREB、BPC、BPU、CGATE及工具程序五部分组成，支持windows/aix/hp\_ux/linux。

框架是基于总线的，支持面向服务的架构(SOA)，支持服务、交易的注册。

## WINDOWS

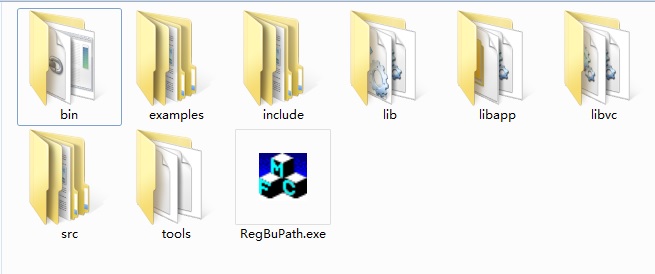


图2 目录结构

下面针对目录进行说明，其中ReqBuPath.exe为windows xp下面的框架路径注册程序，运行此程序时会自动在环境变量PATH里增加bin、lib、libapp、libvc的目录。可以手工添加，因为程序运行时会查找动态库，若不在环境变量PATH里增加目录，则运行报错。（手工添加或运行程序添架时需重启才生效）。

### Bin

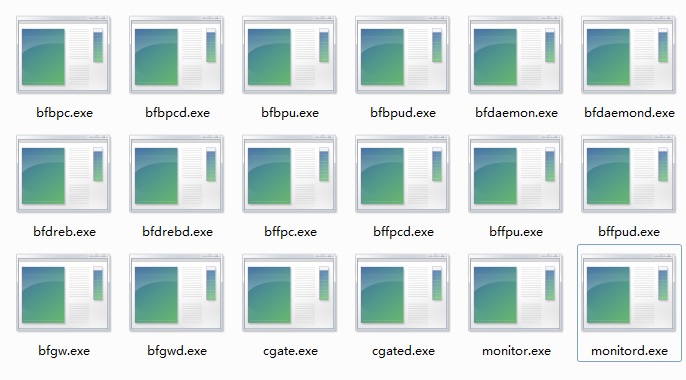


图3 可执行程序目录

**如上图所示，可执行程序后面多了一个d的是DEBUG版本，在程序调试时可使用DEBUG版本进行调试。**

|  |  |
| --- | --- |
| 程序名 | 说明 |
| bfbpc.exe/ bfbpcd.exe | BPC主程序 |
| bfbpu.exe/bfbpud.exe | C/C++的BPU主程序 |
| bfdaemon.exe/bfdaemond.exe | 进程守护主程序 |
| bfdreb.exe/bfdrebd.exe | 通讯总线主程序 |
| bffpc.exe/bffpcd.exe | 前置接入主程序 |
| bffpu.exe/bffpud.exe | 前置处理主程序 |
| bfgw.exe/bfgwd.exe | 堡垒机网关主程序 |
| cgate.exe/cgated.exe | 接入网关主程序 |
| monitor.exe/monitord.exe | 黄金二级系统主机资源主程序 |

### Examples

两个C/C++的BPU例子，VC6工程文件，生成dll动态库。

### Include

框架的头文件目录

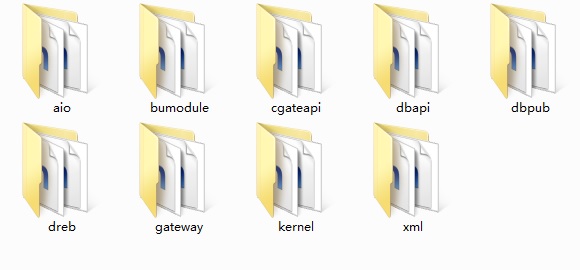


图4 头文件目录

|  |  |
| --- | --- |
| 目录 | 说明 |
| aio | 异步IO框架头文件目录 |
| bumodule | BPU头文件目录 |
| cgateapi | 接入网关的api头文件目录 |
| dbapi | 数据库操作头文件目录 |
| dbpub | Dreb、bpc字节序、报文日志公共头文件目录 |
| dreb | 总线头文件目录 |
| gateway | 接入网关公共头文件目录 |
| kernel | 框架公共类库头文件目录 |
| xml | 老的xml解析器头文件目录 |

### Lib

动态库目录文件列表，多了一个d的文件为DEBUG版本

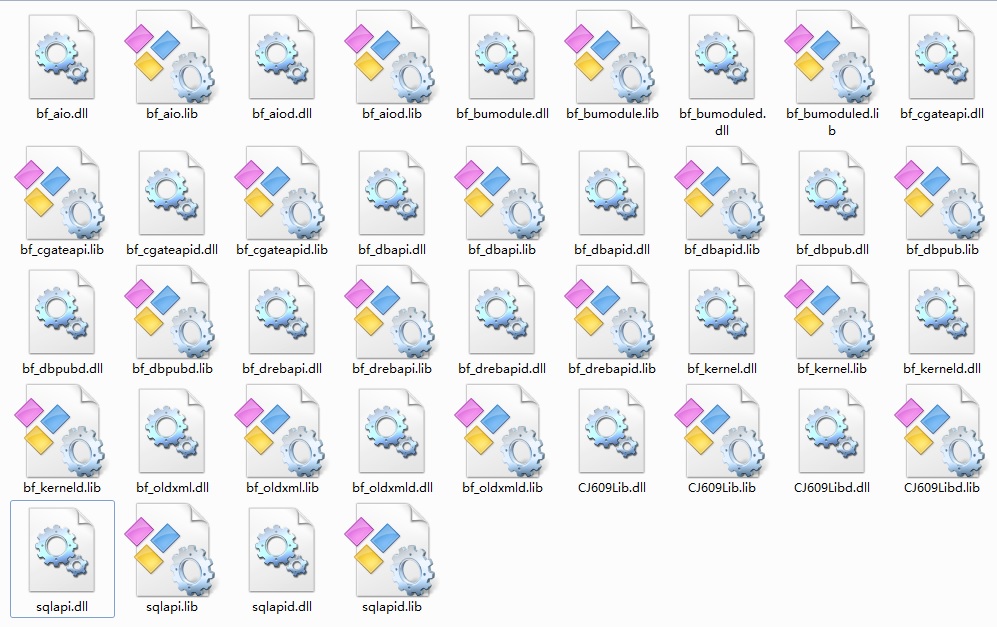


图5 动态库

|  |  |
| --- | --- |
| 动态库 | 说明 |
| bf\_aio.dll/bf\_aio.lib/bf\_aiod.dll/bf\_aiod.lib | AIO框架动态库 |
| bf\_bumodule.dll/bf\_bumodule.lib/ bf\_bumoduled.dll/ bf\_bumoduled.lib | BPU框架动态库 |
| bf\_cgateapi.dll/bf\_cgateapi.lib  bf\_cgateapid.dll/bf\_cgateapid.lib | 接入网关api动态库 |
| bf\_dbapi.dll/ bf\_dbapi.lib  bf\_dbapid.dll/bf\_dbapid.lib | 数据库操作动态库 |
| bf\_dbpub.dll/ bf\_dbpub.lib  bf\_dbpubd.dll/ bf\_dbpubd.lib | Dreb/bpc字符序报文日志公共动态库 |
| bf\_drebapi.dll/ bf\_drebapi.lib  bf\_drebapid.dll/ bf\_drebapid.lib | 数据总线api动态库 |
| bf\_kernel.dll/ bf\_kernel.lib  bf\_kerneld.dll/ bf\_kerneld.lib | 框架类库动态库 |
| bf\_oldxml.dll/ bf\_oldxml.lib  bf\_oldxmld.dll/ bf\_oldxmld.lib | 旧版xml解析动态库 |
| CJ609LIB.dll/CJ609LIB.lib | 某些windows工具程序用到的动态库 |
| sqlapi.dll/sqlapi.lib  sqlapid.dll/sqlapid.lib | sqlapi数据库操作动态库 |

### Libapp

应用动态库目录

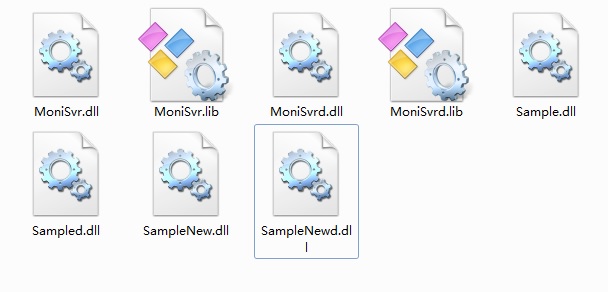


图6 应用动态库

|  |  |
| --- | --- |
| 名称 | 说明 |
| MoniSvr.dll/MoniSvrd.dll | 监控BPU动态库 |
| Sample.dll/Sampled.dll | Bpu例子动态库 |
| SampleNew.dll/SampleNewd.dll | Bpu例子动态库 |

### Livbc

VC6的动态库。

### Src

源程序目录

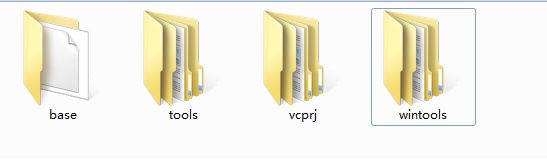


图7 源码目录

#### base

框架基础类库及主程序源码目录

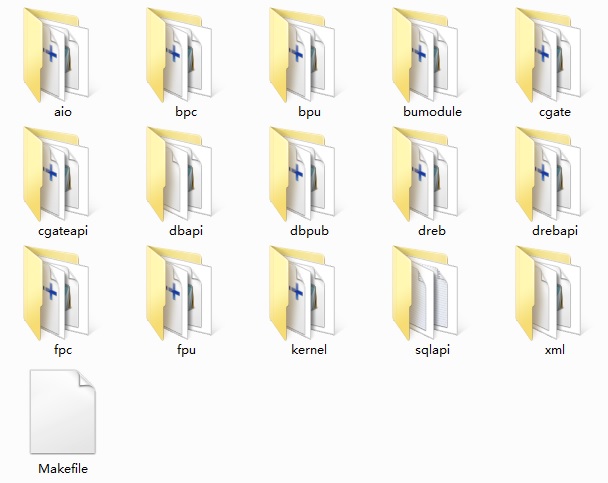


图8 框架基础类库及主程序源码目录

|  |  |
| --- | --- |
| 目录 | 说明 |
| aio | Aio框架动态库源码目录 |
| bpc | BPC主程序源码目录 |
| bpu | BPU主程序源码目录 |
| bumodule | BPU框架动态库源码目录 |
| cgate | 接入网关主程序源码目录 |
| cgateapi | 接入网关api动态库源码目录 |
| dbapi | 数据操作动态库源码目录 |
| dbpub | Dreb/bpc字节序报文日志公共动态库源码目录 |
| drebapi | 总线api动态库源码目录 |
| fpc | 前置接入主程序源码目录 |
| fpu | 前置处理主程序源码目录 |
| kernel | 框架类库动态库源码目录 |
| sqlapi | sqlapi数据操作动态库源码目录 |
| xml | 旧版xml解析器动态库源码目录 |

#### tools

工具程序源码目录

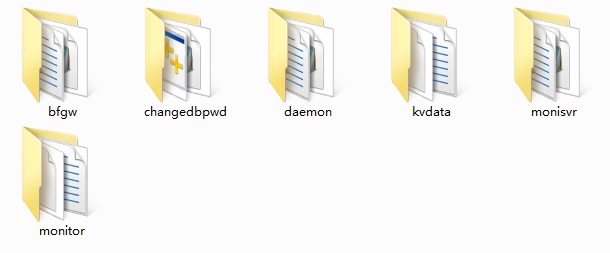


图9 工具程序源码目录

|  |  |
| --- | --- |
| 目录 | 说明 |
| bfgw | 堡垒机网关主程序源码目录 |
| changedbpwd | 修改配置文件数据库配置主程序源码目录 |
| daemon | 进程守护主程序源码目录 |
| kvdata | 黄金二级系统kv报文解析器源码目录 |
| monisvr | 监控服务的bpu动态库源码目录 |
| monitor | 黄金二级系统主机资源主程序源码目录 |

#### vcprj

VC的工程文件目录

#### wintools

windows下的工具程序源码目录

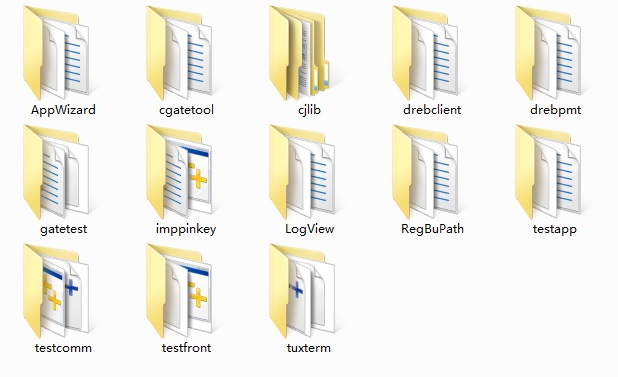


图10 windows下的工具程序源码目录

|  |  |
| --- | --- |
| 目录 | 说明 |
| AppWizard | c/c++ Bpu动态库向导主程序源码目录 |
| cgatetool | 接入网关工具源码目录 |
| cjlib | 针对mfc的一些界面动态库源码目录 |
| drebclient | 总线客户端测试主程序源码目录(单元测试工具) |
| drebpmt | 总线压力测试工具源码目录 |
| gatetest | 网关api的测试程序源码目录 |
| imppinkey | 针对邮储导出交易端使用的加密密钥源码目录 |
| logview | Dreb/bpc/cgate的报文日志查看工具源码目录 |
| reqbupath | 框架路径注册程序源码目录 |
| testcomm | 前置测试工具源码目录(单元测试工具) |
| testfront | 前置压力测试工具源码目录 |
| tuxterm | 针对tuxedo的fml32报文测试工具源码目录(单元测试工具) |

### Tools

工具程序目录

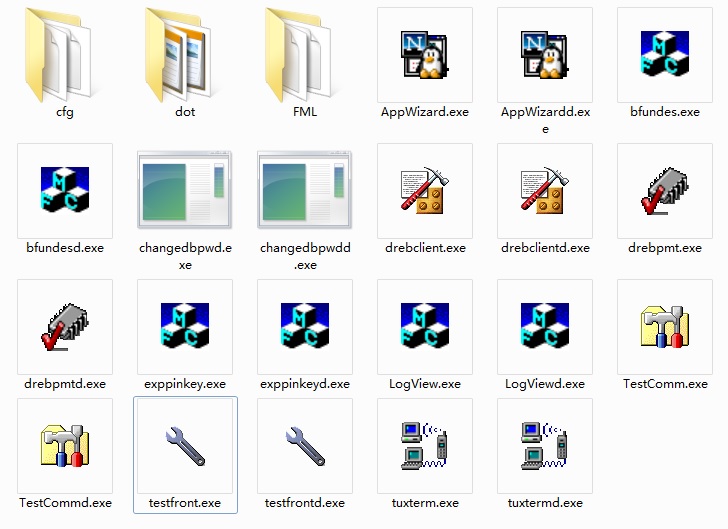


图11工具程序目录

|  |  |
| --- | --- |
| 目录/程序 | 说明 |
| cfg | Testcomm前置测试工具配置文件目录 |
| dot | Bpu业务动态库向导AppWizard的模板目录 |
| fml | Tuxedo的fml32报文测试工具tuxterm的报文配置目录 |
| drebclient.exe/drebclientd.exe | 总线客户端测试主程序(单元测试工具) |
| drebpmt.exe/drebpmtd.exe | 总线压力测试工具 |
| exppinkey.exe/exppinkeyd.exe | 针对邮储导出交易端使用的加密密钥程序 |
| logview.exe/logviewd.exe | Dreb/bpc/cgate的报文日志查看工具 |
| testcomm.exe/testcommd.exe | 前置测试工具(单元测试工具) |
| testfront.exe/testfrontd.exe | 前置压力测试工具源码目录 |
| tuxterm.exe/tuxtermd.exe | 针对tuxedo的fml32报文测试工具 (单元测试工具) |

## UNIX

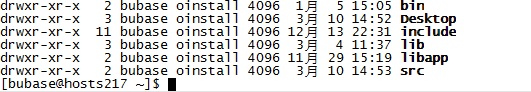


图12 unix下的目录结构

### Bin

可执行程序目录

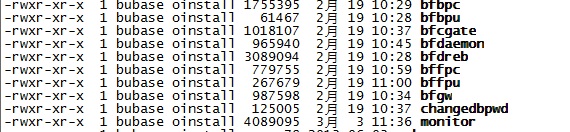


图13 unix下的bin目录内容

|  |  |
| --- | --- |
| 程序名 | 说明 |
| bfbpc | BPC主程序 |
| bfbpu | C/C++的BPU主程序 |
| Bfdaemon | 进程守护主程序 |
| bfdreb | 通讯总线主程序 |
| bffpc | 前置接入主程序 |
| bffpu | 前置处理主程序 |
| bfgw | 堡垒机网关主程序 |
| bfcgate | 接入网关主程序 |
| monitor | 黄金二级系统主机资源主程序 |
| changedbpwd | 修改配置文件里的数据库参数工具主程序 |

### Include

头文件目录

u3.jpg

图14 头文件目录

|  |  |
| --- | --- |
| 目录 | 说明 |
| aio | 异步IO框架头文件目录 |
| bumodule | BPU头文件目录 |
| cgateapi | 接入网关的api头文件目录 |
| dbapi | 数据库操作头文件目录 |
| dbpub | Dreb、bpc字节序、报文日志公共头文件目录 |
| dreb | 总线头文件目录 |
| gateway | 接入网关公共头文件目录 |
| kernel | 框架公共类库头文件目录 |
| xml | 老的xml解析器头文件目录 |

### Lib

动态库目录

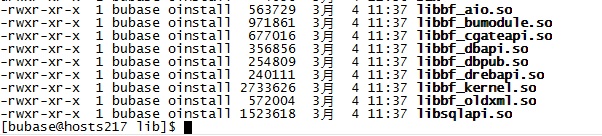


图15 lib目录内容

|  |  |
| --- | --- |
| 动态库 | 说明 |
| libbf\_aio.so | AIO框架动态库 |
| libbf\_bumodule.so | BPU框架动态库 |
| libbf\_cgateapi.so | 接入网关api动态库 |
| libbf\_dbapi.so | 数据库操作动态库 |
| libbf\_dbpub.so | Dreb/bpc字符序报文日志公共动态库 |
| libbf\_drebapi.so | 数据总线api动态库 |
| libbf\_kernel.so | 框架类库动态库 |
| libbf\_oldxml.so | 旧版xml解析动态库 |
| libsqlapi.so | sqlapi数据库操作动态库 |

### Libapp

应用动态库目录

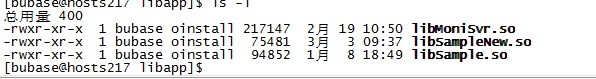


图15 libapp目录内容

|  |  |
| --- | --- |
| 名称 | 说明 |
| libMoniSvr.so | 监控BPU动态库 |
| libSampleNew.so | Bpu例子动态库 |
| libSample.so | Bpu例子动态库 |

### Src

源码目录

u6.jpg

图16 src目录

#### Base

框架基础类库及主程序源码目录

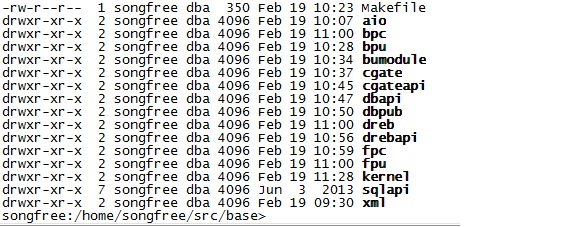


图17 框架基础类库及主程序源码目录

|  |  |
| --- | --- |
| 目录 | 说明 |
| aio | Aio框架动态库源码目录 |
| bpc | BPC主程序源码目录 |
| bpu | BPU主程序源码目录 |
| bumodule | BPU框架动态库源码目录 |
| cgate | 接入网关主程序源码目录 |
| cgateapi | 接入网关api动态库源码目录 |
| dbapi | 数据操作动态库源码目录 |
| dbpub | Dreb/bpc字节序报文日志公共动态库源码目录 |
| drebapi | 总线api动态库源码目录 |
| fpc | 前置接入主程序源码目录 |
| fpu | 前置处理主程序源码目录 |
| kernel | 框架类库动态库源码目录 |
| sqlapi | sqlapi数据操作动态库源码目录 |
| xml | 旧版xml解析器动态库源码目录 |

#### Tools

工具源码目录

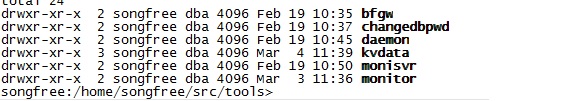


图18 *工具源码目录*

|  |  |
| --- | --- |
| 目录 | 说明 |
| bfgw | 堡垒机网关主程序源码目录 |
| changedbpwd | 修改配置文件数据库配置主程序源码目录 |
| daemon | 进程守护主程序源码目录 |
| kvdata | 黄金二级系统kv报文解析器源码目录 |
| monisvr | 监控服务的bpu动态库源码目录 |
| monitor | 黄金二级系统主机资源主程序源码目录 |

# 框架介绍

## 层次结构



**图2.1 层次结构**

如上图所示，框架分为表示层、接入层、传输层、服务层、功能层、数据层。其中服务层和功能层对应系统逻辑结构中的服务层：

* 数据层

数据层指数据库系统和其他数据文件，数据库系统采用通用的大型关系数据库，存放并处理客户资料及其交易数据，数据文件主要用来保存各种配置参数等，数据层是系统中集中存储数据的地方，根据数据生命周期特点，可分为历史数据和生产数据。在生产数据中，根据数据的性质又可分为业务数据、框架数据等。根据数据的存储形式，又可分为数据库内的数据和文件等。

* 功能层

即为业务逻辑层，为各系统实现业务功能的地方，在框架开发类库的支持下，可以方便快速地实现业务开发。功能层向服务层注册交易码，按交易码对外提供服务。

* 服务层

提供服务支持的层，在此层实现服务的负载均衡，可将相同的交易或相同的业务组装在一起对外提供服务(服务号和私有服务号)。服务要在传输层进行注册，这样消息才能在传输层正确路由。

* 传输层

传输层主要是由各通讯模块组成。通讯模块作为黄金买卖业务系统中的数据总线，主要负责连接各个业务模块及客户端，进行数据的通讯传输，所以这一层的可靠性对整个交易系统的可靠性起着决定性的作用。传输层的引入能够使交易所信息系统的部署更加灵活，同时便于周边系统的接入以及与其他应用系统的集成。传输层可提供可靠传输、加密压缩、应用路由、连接复用、负载均衡、故障隔离/恢复等功能，以保证系统的可靠性、安全性和扩展性，采用的通讯协议是可以灵活扩展的数据路由交换总线，消息在此层进行路由转发，可根据指定总线节点进行路由、根据服务进行路由和根据交易码进行路由。

* 接入层

接入层由网关服务或通讯接入模块组成，接入层是实现和其他系统进行交互的桥梁，负责大量并发的客户接入，采用异步IO处理大量并发的连接或将外部的报文协议转为框架内流转的报文协议进行接入。

* 表示层

表示层是本系统的用户接口部分，它担负着用户与应用间的对话功能。表示层检查用户从键盘或其它终端设备上输入的数据，显示应用服务输出的数据。在这一层的设计中，对用户输入数据的检查只限于数据的形式和值的范围，不包括有关业务本身的处理逻辑，因此在用户接口变更时，只需改写操作显示控制和数据检查代码，而不影响其它层。用户输入的数据通过通讯平台传入功能层。表示层与应用逻辑无关，能保证风格统一、操作方便。

## 系统总线DREB

数据总线(DREB : Data Route Exchange Bus 数据总线)通过节点（数据总线程序的一个运行实例）的互联，构造了一个可动态调整的网络应用系统数据交换环境，为网络应用系统提供了统一的数据交换方式和环境

从调用的方式上，可分为同步和异步两种方式，在对通讯数据的使用上，应用系统可以在数据的提供者，使用者，分派者三种角色中使用一种或者多种。总之，数据总线为网络应用系统提供了强有力的开发和部署的支持。

每个DREB节点都有一个公共节点号及私有节点号。私有节点号是在公共节点号相同的情况下，进行节点的区别。服务要在总线上进行注册。

DREB 通过将系统间的耦合统一为消息的耦合（一种松散的耦合），降低了系统间的耦合度，建立起了动态的系统间的应用消息总线（即插即用）

 **图2.2-1 DREB消息耦合**

数据总线由即时通讯组件、API接口、监控三部分组成

 **图2.2-2 DREB组成**

### DREB的数据交换模式

采用存储转发机制，未来可扩展为直通链路、持久化转发等

1、上下文无关的模式：交易可由多个不同的服务分别来处理，例如贵金属业务加办，由公共模块和代理金模块或实物金共同完成。

2、上下文相关的模式：一个交易必须由一个服务来处理，例如取后续包，服务生成后续包缓存在内存或文件中，下次取后续包必须发送到此服务才能获取后续包。

 **图2.2.1 DREB的消息交换模式**

### 调用模式

总线节点提供的消息调用模式，根据需要来使用不同的模式。

共有7种模式，分别为DPCALL、DPACALL、DPPUSH、DPPOST、DPBC、DPABC、DPTRANS。

#### 同步调用DPCALL

同步请求应答模式 DPCall，类似TUXEDO的TPCALL，此模式由服务处理完成后发送响应给请求方。请求方一直等待服务应答，直到收到服务应答或超时。例如管理端登录，操作员输入用户名密码后发送登录请求至公共模块，一直等待公共模块的应答，若超时重新登录。

 **图2.2.2.1 同步调用**

#### 异步模式 DPACall

此模式由接收到的通讯节点发送确认给请求方，并将请求发给服务进行处理，服务处理完成后再应答。

 **图2.2.2.2 异步调用**

#### 消息推送 DPPush

此模式一般由服务主动发起给客户端或服务，无需应答的交易，例如委托回报、成交回报。

 **图2.2.2.3 消息推送**

#### 消息投递 DPPost

此模式和消息推送相似，差别是此消息需要应答。

 **图2.2.2.4 消息投递**

#### 消息广播 DPBC

发送消息给DREB上的所有服务，若不指定DREB，则发送给所有的DREB,即全域广播，要求第一个收到的DREB发送确认。

 **图2.2.2.5 消息广播DPBC**

#### 消息广播 DPABC

类似消息广播DPBC，差别是此消息不需要确认，例如行情就使用此种模式。

 **图2.2.2.6 消息广播DPABC**

#### 消息传递DPTrans

服务接收消息后将消息转发给另外的服务，由另外的服务处理并应答。

 **图2.2.2.7 消息传递**

## 业务处理中心BPC

**BPC全称为Business Process Center 又名SAP 全称为Service Access Point或者Service Adapter。**

BPC业务处理中心，负责在DREB上注册服务，接收BPU的连接并注册交易码至DREB。以下为BPC的工作模式

 **图2.3 BPC的工作模式**

BPC通过TCP连到DREB,并在DREB上注册服务号，分公共服务号和私有服务号，私有服务号是用来区别相同的公共服务号的服务。多个相同的公共服务号进行服务的负载均衡，后续包等通过私有服务号来获取。

BPC接收DREB分派过来的数据进行服务分派。

BPC和BPU部署在同一台机器上，由BPC来管理BPU，BPU通过TCP连接BPC。

BPC上可连接多个BPU组,但不同的BPU组里的交易码不可重复,每个BPU组里有多个BPU实例具有相同的服务功能，由BPC将请求分派给BPU进行处理。

BPC没有业务逻辑，只是根据BPU注册的交易码进行交易分派。

## 业务处理单元BPU

**BPU全称为Business Process Unit 又名SPU 全称为Service Process Unit** 。

BPU有两个版本，一个是C/C++；另一个是JAVA的。

这里以C/C++版本作介绍

* BPU是业务的具体实现
* BPU是单独一个进程
* BPU由主程序+业务动态库组成
* BPU通过TCP连接BPC
* BPU的主要业务是由业务动态库里的原子交易构成
* BPU内可以调用本BPU内的所有原子
* BPU可以通过连接的BPC调用连接的BPC上的所有功能
* BPU可以调用DREB上注册的所有服务
* BPU可以边处理边响应客户端请求
* BPU支持XML及其它多种数据格式

BPU是构成服务的基本单元

BPU由BPU主程序+业务动态库组成

BPU内部由业务流程控制和业务原子功能及底层类库组成

  **图2.4 BPU组成**

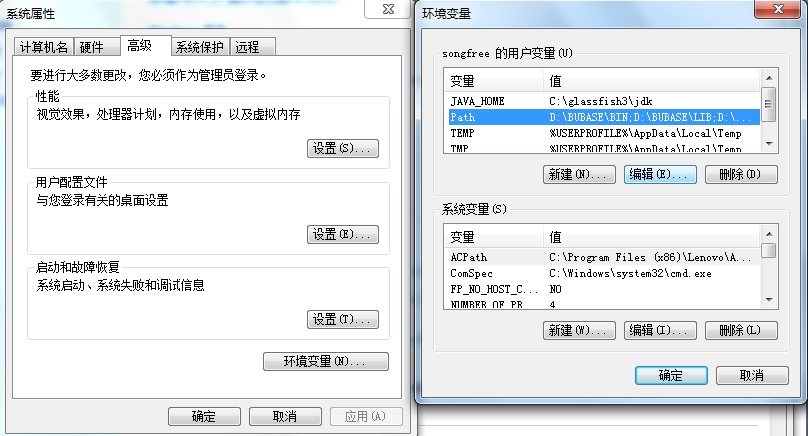
# 框架的安装运行

## 安装

### WINDOWS

1. 获取安装包bubase.rar
2. 解开bubase.rar到本地硬盘
3. 手动修改环境变量PATH，增加解开目录下的bin lib libapp libvc目录路径至PATH中

（WINDOWS XP可运行RegBuPath.exe自动修改）



1. 重启系统以使变量生效

### UNIX

1. 获取安装包bubase.tar.gz
2. 在用户的$HOME下解开安装包bubase.tar.gz

tar zxvf bubase.tar.gz

1. 修改.bash\_profile（linux） .profile (AIX,HP\_UX)

在PATH变量里增加如下一行：

export PATH=$(HOME)/bin:$PATH

在LD\_LIBRARY\_PATH变量增加如下：

export LD\_LIBRARY\_PATH= $HOME/lib:$HOME/libapp:$LD\_LIBRARY\_PATH

在LIBPATH变量增加如下：

export LIBPATH= $HOME/lib:$HOME/libapp:$LIBPATH

1. 重新登录以使.bash\_profile生效。

## 运行

### 总线dreb

#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfdreb | 自动查找当前目录下的bfdreb.xml并启动，在后台运行。若当前目录下dreb在共享内存的标志为已运行，则提示已运行，先停止再启动 |
| bfdreb xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在后台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfdreb stop | 程序使用共享内存来记录启动的标识 |
| bfdreb STOP | 退出当前目录下运行的dreb |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <package>  <head>  <日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">6</日志级别>  <DATA日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</DATA日志级别>  <日志文件 remark="记录日志信息的文件名,前加日期">dreb.log</日志文件>  <日志目录 remark=”记录日志的目录”></日志目录>  <ListenPort remark="数据总线对外提供客户端服务的侦听端口">9011</ListenPort>  <公共节点号 remark="数据总线公共编号">100</公共节点号>  <私有节点号 remark="数据总线私有编号">1</私有节点号>  <是否压缩 remark="若数据未压缩，数据总线是否自动压缩0否1是">0</是否压缩>  <最大连接数 remark="对外提供服务的最大连接，大于最大服务数但不要超过640">100</最大连接数>  <节点路由时间 remark="节点路由的间隔">10</节点路由时间>  <服务功能路由时间 remark="服务功能路由的间隔">30</服务功能路由时间>  <服务队列过期时间 remark="服务队列过期时间">300</服务队列过期时间>  <服务功能过期时间 remark="服务功能过期时间">300</服务功能过期时间>  <连接心跳检测 remark="针对连接的客户端或服务端是否发送心跳0否1是">1</连接心跳检测>  <断开心跳个数 remark="发送心跳个数后仍未收到心跳应答，断开此连接">5</断开心跳个数>  <发送缓冲个数 remark="连接的发送数据个数限制0不限制，最小100">0</发送缓冲个数>  <单独处理线程 remark="接收和处理是否两个线程">0</单独处理线程>  <监控配置 use="0" host="0" remark="use为主动报告选项，0不报告 host为主机资源报告选择0不报告">  <监控公共节点号>0</监控公共节点号>  <监控私有节点号>0</监控私有节点号>  <监控公共服务号>0</监控公共服务号>  <监控私有服务号>0</监控私有服务号>  <监控报告交易码>99001</监控报告交易码>  </监控配置>  </head>  <pack 级连个数="2" remark="连接的数据总线的ip端口、节点编号及带宽,单位为K">  <级连数据总线a ip="20.13.0.217" port="13913" bandwidth="102400" remark="连接的数据总线的ip端口、节点编号及带宽,单位为K" />  <级连数据总线a ip="20.13.0.217" port="13914" bandwidth="102400" remark="连接的数据总线的ip端口、节点编号及带宽,单位为K" />  </pack>  </package> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**DATA日志级别：**即接收发送报文日志的级别，当大于等于5-DEBUG时将报文记录到文件中，文件可用logview工具来查看。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmdd+x，当为报文日志时，文件名为yyyymmdd+x.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录。

**ListenPort：**本总线的侦听端口，所有的连接都是通过此端口。

**公共节点号：**总线的公共编号，多个负载均衡的节点相同。

**私有节点号：**总线的私有编号，公共+私有编号才是全网唯一的编号，多个负载均衡的节点私有编号一定不能相同。

**是否压缩：**此参数已取消不用，当为调用时，判断数据长度若大于200字节且cZip标志为明文时，则自动进行数据压缩。

**最大连接数：**允许连接的最大数，为客户端+服务端+总线级连的最大数。

**节点路由时间：**总线节点之间的路由时间，即每隔此时间就发送一次总线节点路由信息，一般设置为5秒或10秒左右

**服务功能路由时间：**总线里注册的交易的路由时间，即每隔此时间就发送一次交易路由信息，这样所有的节点上就有交易信息，不管在哪个节点上，都可以通过交易来调用，无需填写目标节点。此时间一般以大于30秒为宜。

**服务队列过期时间：**交易请求在服务队列里过期的时间，当服务来不及处理时，请求就阻塞在队列中，当超过此时间后，请求被丢弃，一般设置为300秒

**服务功能过期时间：**交易在总线节点的过期时间，当交易在此时间未更新时，自动在总线里删除。此时间一定要大于服务功能路由时间的5倍，一般设置为300秒。

**连接心跳检测：** 当为1时针对服务端和总线级连连接发送心跳。客户端不主动发送心跳。若无数据则每隔5秒发送一次心跳

**断开心跳个数：** 当为服务端和总线级连连接时，如果在此个数\*5秒内没有收到数据，则断开此连接。

**发送缓冲个数：**此参数已取消不用。

**单独处理线程：**此参数已取消不用。

**监控配置：**属性use设置为1时，主动发送监控报告，属性host设置为1时，监控报告里包含主机信息(cpu，内存，磁盘)。当为use为1时，下面的几个节点为监控服务所在的信息，若通过交易来调用，总线节点和服务全填0就可以了,否则必须填写。

**级连数据总线：**配置级连的总线信息，注意带宽若不相同，交易按带宽大的路由，若相同则随机路由。

#### 运行说明

总线没有业务逻辑，只是对数据包进行转发，所以，发给总线或接收总线的数据结构如下：

 如上图所示，在业务数据前要加上68字节的报文头。服务端通过BPC模块(SAP或ada)接入，客户端通过总线api接入。

总线的连接分为总线间级连，客户端连接，服务端连接。

如上图所示，DREB报文头由12组数据组成，共28个字段。

##### FIELD01 -加密标志

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 加密标志  加密压缩标志 | 1  Char  取值为asc值 | 0 | 明文，不加密不压缩 |
| 1 | DES加密，密钥由DREB固定，数据为可见字符，长度增大一倍 |
| 2 | DREB压缩 |
| 3 | 3DES加密，密钥由DREB固定 |
| 4 | DES加密并压缩, DES密钥由DREB固定，相当于1+2 |
| 其它 | 业务自定义，DREB及BPC不做处理，由业务处理 |

##### FIELD02 -总线命令

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线命令 | 1  Char  取值为asc值 | 1 | 数据总线节点发送路由通知的命令字 CMD\_ROUTER |
| 2 | 数据总线节点之间取消数据总线节点的命令字,只发送至主动连接的数据总线节点 CMD\_DEL\_NODE |
| 3 | 服务端取消注册服务的命令字 CMD\_DEL\_SVR |
| 4 | 心跳请求的命令字 CMD\_PING |
| 5 | 连接注册CMD\_CONNECT |
| 6 | 服务路由CMD\_SERVICE |
| 7 | 注册服务 CMD\_REGSERVICE |
| 8 | 数据总线节点同步调用 要求最终处理完成后应答 CMD\_DPCALL |
| 9 | 数据总线节点异步调用 要求接收到的数据总线节点发送至服务后确认应答 CMD\_DPACALL |
| 10 | 数据总线节点广播，即将信息发给指定数据总线节点上所有注册的服务,要求数据总线节点应答，若不指定节点则全网所有节点都发送 CMD\_DPBC |
| 11 | 数据总线节点广播，即将信息发给指定数据总线节点上所有注册的服务,不要求数据总线节点应答，若不指定节点则全网所有节点都发送CMD\_DPABC |
| 12 | 数据总线节点推送，无须应答 CMD\_DPPUSH |
| 13 | 数据总线节点投递，要求接收到的数据总线节点应答 CMD\_DPPOST |
| 14 | 非DREB命令，备用 |
| 15 | 数据总线节点监控 CMD\_MONITOR\_DREB |
| 16 | 业务处理中心BPC内部交易 CMD\_MONITOR\_BPC |

##### FIELD03-CRC

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| CRC校验位 | 4  int |  | DREB头的校验位，防止非法篡改消息，由DREB生成校验位和检查。 |

##### FIELD04-RA标志

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| RA标志 | 1  Char  asc | 0 | 请求应答标志 0请求 |
| 1 | 请求应答标志 1应答 |

##### FIELD05-后续标志

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 后续标志 | 1  Char  asc | 0 | 请求时，表示无后续包请求；  应答时，表示无后续包，是尾包 |
| 1 | 应答时，表示有后续包在发；请求时此字值无意义 |
| 2 | 请求时，表示此请求是去BPC取后续包;  应答时，当为DREB通讯数据时表示是BPC处理后续包的应答，当为BPU-BPC通讯时，为BPU应答给BPC，通知BPC这是一个要在BPC生成多包数据缓冲的应答，前端调用取后续包的请求主动来取后续包。 |
| 3 | 请求时，表示此请求是去BPU取后续包，其中BPC信息里存放BPU信息(时间戳和BPU连接索引)  应答时，表示此为BPU应答出来的数据， |
| 10 | 对应应答1，表示1类的应答无后续包了 |
| 20 | 对应去BPC取后续包的的应答，20表示无后续包 |
| 30 | 对应去BPU取后续包的的应答，30表示无后续包 |

##### FIELD06-确认标志

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 确认标志 | 1  Char  asc | 0 | DREB确认的标志 0 由应用确认 |
| 1 | DREB确认的标志 1 由DREB确认，如CMD\_DPACALL |

##### DREB源节点信息

初始请求发出的DREB节点信息，标识是哪个节点发出的，最终的确认和应答要根据此节点信息返回给请求方。客户端在注册时，必须保存DREB返回的此节点信息，以DREB启动校验时，请求数据发到连接的DREB时，DREB会校验此段信息是否匹配。包括9部分内容

###### FIELD07-总线节点ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线ID | 2  short | 0-65535 | 标识DREB总线节点，多个负载均衡的DREB节点此ID相同，但私有节点一定不相同。也就是说总线节点ID+总线私有节点ID是唯一的  客户端登录要保存此字段，以便回报推送 |

###### FIELD08-总线私有ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线私有ID | 1  Char  asc | 0-255 | 标识DREB总线节点，多个负载均衡的DREB节点此ID相同，但私有节点一定不相同。也就是说总线节点ID+总线私有节点ID是唯一的，标识总线私有ID是从1开始，当请求不确定是具体哪个节点来处理时，此值填0  客户端登录要保存此字段，以便回报推送 |

###### FIELD09-服务ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 服务ID | 4  int | >1000 | 标识服务，多个负载均衡的服务此ID相同，但服务私有ID一定不相同。也就是说服务ID+服务私有ID是唯一的。在系统设计时，要全局考虑服务号的分配。  客户端登录要保存此字段，以便回报推送 |

###### FIELD10-服务私有ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 服务私有ID | 1  char |  | 标识服务，多个负载均衡的服务此ID相同，但服务私有ID一定不相同。也就是说服务ID+服务私有ID是唯一的。在系统设计时，要全局考虑服务号的分配。标识服务私有ID是从1开始，当请求不确定是具体哪个服务来处理时，此值填0  客户端登录要保存此字段，以便回报推送 |

###### FIELD11-请求标识

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 请求标识 | 4  int |  | 由请求方填写，不作处理，原样返回。若是BPU外调，BPU可通过此字段来标识请求是哪个BPU调用的。 S\_HOOK。CGATE网关用来存放连接的时间戳，客户端登录时保存此时间戳，当回报的时候根据此时间戳来判断是否原来的客户端。 |

###### FIELD12-请求流水

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 请求流水 | 4  int |  | 由请求方填写，不作处理，原样返回。若是BPU外调，BPU可通过此字段来标识请求是哪个BPU调用的。  S\_SERIAL。CGATE网关用来存放连接索引和客户端的请求流水(前2字节为索引，后2字节为请求流水)，当回报的时候根据此字段前2字节发给对应的客户端(配合字段11)。 |

###### FIELD13-总线流水

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线流水 | 4  int |  | 由DREB填写，主要是用作日志跟踪，同时根据此字段+总线ID+总线私有ID可知应答多包的数据是哪个请求的应答。 |

###### FIELD14-总线连接

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线连接 | 2  short |  | 由DREB填写，标识客户端或服务端的连接在DREB的序号，最终通过此连接将应答返回给请求方。 |

###### FIELD15-时间戳

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 时间戳 | 4  int |  | 和DREB建立连接时的连接时间戳，当客户端连接注册时返回，请求包需原样发送，当启动连接认证时，DREB会验证数据包里的此时间戳和连接的时间戳是否相同，若不相同则说明连接已变更，请求或应答无效 |

##### DREB目的节点信息

数据要送达的DREB节点信息。

###### FIELD16-总线节点ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线ID | 2  short | 0-65535 | 标识DREB总线节点，多个负载均衡的DREB节点此ID相同，但私有节点一定不相同。也就是说总线节点ID+总线私有节点ID是唯一的 |

###### FIELD17-总线私有ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线私有ID | 1  char | 0-255 | 标识DREB总线节点，多个负载均衡的DREB节点此ID相同，但私有节点一定不相同。也就是说总线节点ID+总线私有节点ID是唯一的，标识总线私有ID是从1开始，当请求不确定是具体哪个节点来处理时，此值填0 |

###### FIELD18-服务ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 服务ID | 4  int | >1000 | 标识服务，多个负载均衡的服务此ID相同，但服务私有ID一定不相同。也就是说服务ID+服务私有ID是唯一的。在系统设计时，要全局考虑服务号的分配。 |

###### FIELD19-服务私有ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 服务私有ID | 1  char | >1000 | 标识服务，多个负载均衡的服务此ID相同，但服务私有ID一定不相同。也就是说服务ID+服务私有ID是唯一的。在系统设计时，要全局考虑服务号的分配。标识服务私有ID是从1开始，当请求不确定是具体哪个服务来处理时，此值填0 |

###### FIELD20-交易码

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 交易码 | 4  int | >1000 | 标识服务，多个负载均衡的服务此ID相同，但服务私有ID一定不相同。也就是说服务ID+服务私有ID是唯一的。在系统设计时，要全局考虑服务号的分配。标识服务私有ID是从1开始，当请求不确定是具体哪个服务来处理时，此值填0 |

##### DREB确认

当数据需要DREB确认时，确认收到数据的DREB填入此信息。

###### FIELD21-总线节点ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线ID | 2  short | 0-65535 | 标识DREB总线节点，多个负载均衡的DREB节点此ID相同，但私有节点一定不相同。也就是说总线节点ID+总线私有节点ID是唯一的 |

###### FIELD22-总线私有ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 总线私有ID | 1  char | 0-255 | 标识DREB总线节点，多个负载均衡的DREB节点此ID相同，但私有节点一定不相同。也就是说总线节点ID+总线私有节点ID是唯一的，标识总线私有ID是从1开始，当请求不确定是具体哪个节点来处理时，此值填0 |

###### FIELD23-返回码

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 返回码 | 2  short |  | DREB确认的返回码，0成功  #define SUCCESS 0 //成功  #define ERR\_SVRID 1 //服务号不符  #define ERR\_PRIVATEID 2 //私有服务号不符  #define ERR\_REGISTER 3 //服务已注册  #define ERR\_SVRNOTREGISTER 4 //服务未注册  #define ERR\_DREBID 5 //通讯节点ID不符  #define ERR\_DREBPRIVATEID 6 //通讯私有节点ID不符  #define ERR\_DREBREGISTER 7 //通讯节点已注册  #define ERR\_DREBNODE 8 //通讯节点重复  #define ERR\_QUEUE 9 //入队列失败  #define ERR\_DREBROUTE 10 //通讯节点路由不存在  #define ERR\_VERSION 11 //连接版本不符  #define ERR\_SERVICENO 12 //注册功能号不符  #define ERR\_FUNCNOTFUND 13 //功能号不存在  #define ERR\_NEXTFILENOT 14 //后续包文件不存在  #define ERR\_BPC\_REGTX 21 //注册交易错误  #define ERR\_BPC\_NOGROUP 22 //注册交易错误,无BPU组  #define ERR\_BPC\_NOREGISTER 23 //注册交易错误,无需注册  #define ERR\_BPC\_BPULIMIT 24 //注册交易错误,BPU连接限制,已满  #define ERR\_BPC\_TXFIND 25 //注册交易错误,交易已注册  #define ERR\_START\_DREBTIMER 26 //启动dreb监控定时器失败  #define ERR\_STOP\_DREBTIMER 27 //停止dreb监控定时器失败 |

##### 后续包信息

当应答数据包超过框架定义的缓冲区长度时，要使用此字段通知BPC或前端有后续包要进行处理。

###### FIELD24-NEXT序号

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| NEXT序号 | 4  Unsigned int |  | 当RA标志为0请求时，且后续标志为2或3时，为取后续包请求，此字段保存为后续包的唯一序号；  当RA标志为1时，后续标志为1时，表示数据为应答数据，后台主动发送后续包，无须前台来取后续包，但因为网络路由原因，可能前端收到的包顺序不一样，所以此字段保存总共数据包个数；  当RA标志为1时，后续标志为2时，若为BPC-BPU通讯，则此字段值为总共数据包个数，若为DREB通讯数据，则为后续包的唯一序号；  当RA标志为1时，后续标志为3时，此字段为后续包的唯一序号 |

###### FIELD25-NEXT位移

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| NEXT位移 | 4  Unsigned int |  | 当RA标志为0请求时，且后续标志为2或3时，为取后续包请求，此字段保存为要取的后续包在数据缓冲的位置；  当RA标志为1时，后续标志为1时，表示数据为应答数据，后台主动发送后续包，无须前台来取后续包，但因为网络路由原因，可能前端收到的包顺序不一样，所以此字段值为第几个数据包(1开始)；  当RA标志为1时，后续标志为2时，若为BPC-BPU通讯，则此字段值为第几个数据包，若为DREB通讯数据，则为后续包的偏移位置；  当RA标志为1时，后续标志为3时，此字段为后续包的偏移位置。 |

##### BPC信息

BPC调用DREB使用的数据，当BPU进行外调或内调时，通过此字段保存BPU信息，以便能返回给调用者。

###### FIELD26-BPC时间戳

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| BPC时间戳 | 4  int |  | BPC外调或内调使用的数据，存放请求连接的时间戳 |

###### FIELD27-BPC索引

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| BPC索引 | 2  Unsigned short |  | BPC外调或内调使用的数据，存放请求连接索引，根据此索引返回给调用的BPU |

##### FIELD28-数据长度

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 数据长度 | 2  short | 0-65534 | 不包含本报文头的长度，为业务数据的长度。 |

### 业务处理中心bpc(服务访问点SAP)

#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfbpc | 自动查找当前目录下的bfbpc.xml并启动, 在后台运行，若共享内存标志bpc为已运行则提已运行，需先停止 |
| bfbpc xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在后台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfbpc stop | 程序使用共享内存来记录启动的标识, 退出当前目录下运行的bpc |
| bfbpc STOP | 退出当前目录下运行的bpc |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <package>  <head>  <公共配置>  <日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">6</日志级别>  <DATA日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</DATA日志级别>  <日志文件 remark="记录日志信息的文件名,前加日期">bfbpc.log</日志文件>  <日志目录 remark="日志存放的目录"></日志目录>  </公共配置>  <DREB配置>  <DREB1 ip="127.0.0.1" port="9011" bandwidth="100" remark="配置数据总线的参数" />  </DREB配置>  <监控配置 use="1" host="1" remark="use为主动报告选项，0不报告 host为主机资源报告选择0不报告">  <监控公共节点号>0</监控公共节点号>  <监控私有节点号>0</监控私有节点号>  <监控公共服务号>0</监控公共服务号>  <监控私有服务号>0</监控私有服务号>  <监控报告交易码>99001</监控报告交易码>  </监控配置>  <BPC配置>  <公共服务号 remark="在dreb上注册的服务号">100</公共服务号>  <私有服务号 remark="在dreb上注册的私有服务号">1</私有服务号>  <最大流水计数 remark="Bpu共享的流水最大值，超过则从1重新开始">99999999</最大流水计数>  <本次流水开始值 remark="Bpu共享的流水起始值">1</本次流水开始值>  <请求分配超时时间 remark="请求分配给BU的超时时间，秒">10</请求分配超时时间>  <连接过期时间 remark="当连接长时间未有数据时断开,秒">600</连接过期时间>  <DREB心跳时间 remark="和DREB的心条时间,秒">60</DREB心跳时间>  <交易注册模式 remark="0表示动态注册，1表示静态注册 2表示不注册">0</交易注册模式>  <交易注册文件 remark="静态注册时的文件名">txlist.xml</交易注册文件>  </BPC配置>  <BPU配置>  <BPU侦听端口 remark="接受BPU连接上来的端口">29001</BPU侦听端口>  <BPU组>  <monibpu>  <发送明文 remark="由BPU解压缩解密0,明文发给BPU1">0</发送明文>  <心跳应答 remark="当为c/c++多bpu时，不用应答心跳消息">0</心跳应答>  <分配模式 remark="0表示空闲分配，1表示BPU数只能为1，一直分配">0</分配模式>  <通讯字节序 remark="0主机序，1表示网络序">0</通讯字节序>  <可启动的BPU数 remark="即可以处理业务的单元数目，当请求分配模式为1时只能为1">1</可启动的BPU数>  <AUTOLOADBPU use="1" remark="自动将BPU程序启动1，否则为0">  <BPU程序名 remark="BU程序的名称，不带路径">bfBPUd.exe</BPU程序名>  <BPU程序路径 remark="绝对或相对路径">D:\BUBASE\bin</BPU程序路径>  <BPU运行目录 remark="绝对或相对目录">D:\BUBASE\bin</BPU运行目录>  <BPU启动参数 remark="启动BPU所需的参数">D:\BUBASE\bin\bfBPU.xml</BPU启动参数>  </AUTOLOADBPU>  </monibpu>  <bpu2>  <发送明文 remark="由BPU解压缩解密0,明文发给BPU1">1</发送明文>  <心跳应答 remark="当为c/c++多bpu时，不用应答心跳消息">0</心跳应答>  <分配模式 remark="0表示空闲分配，1表示BPU数只能为1，一直分配">0</分配模式>  <可启动的BPU数 remark="即可以处理业务的单元数目，当请求分配模式为1时只能为1">2</可启动的BPU数>  <AUTOLOADBPU use="0" remark="自动将BPU程序启动1，否则为0">  <BPU程序名 remark="BU程序的名称，不带路径">bfBPU.exe</BPU程序名>  <BPU程序路径 remark="绝对或相对路径">./</BPU程序路径>  <BPU运行目录 remark="绝对或相对目录">./</BPU运行目录>  <BPU启动参数 remark="启动BPU所需的参数"></BPU启动参数>  </AUTOLOADBPU>  </bpu2>  </BPU组>  </BPU配置>  </head>  <pack />  </package> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**DATA日志级别：**即接收发送报文日志的级别，当大于等于4时将报文记录到文件中，文件可用logview工具来查看。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmddx，当为报文日志时，文件名为yyyymmddx.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录

**DREB配置：**配置连接总线节点，可以连接多个负载均衡的总线节点。

**监控配置：**同总线的监控配置, 属性use设置为1时，主动发送监控报告，属性host设置为1时，监控报告里包含主机信息(cpu，内存，磁盘)。当为use为1时，下面的几个节点为监控服务所在的信息，若通过交易来调用，总线节点和服务全填0就可以了,否则必须填写。

**BPC配置：**配置BPC的信息，

**公共服务号：**本服务的公共编号

**私有服务号：**本服务的私有编号，公共+私有编号是全网唯一的。

**最大流水计数：**多个BPU可以通过接口共享同一个交易流水，相当于通过BPC获得一个唯一的序号，此参数是流水的最大值，当大于此最大值时从1重新开始，主要是给C++版的BPU使用。

**本次流水开始值：**BPC启动时的唯一流水的开始值，每次启动都从此值开始。

**请求分配超时时间：**BPC从总线接收到数据后放到队列中，由分配线程从队列取请求分配给BPU进行处理，如果BPU一直很忙或没有连接BPU时，超过此时间就将此请求丢弃，返回错误信息给请求方。

**连接过期时间：**当总线连接和BPU连接在此时间内没有发送数据给BPC时，BPC将主动断开连接。

**DREB心跳时间：**和总线的连接，若在此时间内没有收到数据，则主动发送心跳给总线。

**交易注册模式：**在总线上注册交易的模式，有三种，0表示动态注册，BPU向BPC注册，BPC再向总线注册，1表示静态注册，读取交易注册文件，将交易注册到总线，不再接收BPU注册， 2表示不向总线注册，但接收BPU注册，此种情况适合分布式系统。

**交易注册文件：**在交易注册模式为1时起作用。此文件配置了要向总线注册的交易。若BPU无此交易，由BPU返回出错信息。

**BPU配置：**配置针对BPU连接的侦听端口及BPU组。注意BPU组名不能超过20个字符。

**BPU侦听端口：**接收BPU连接上来的端口。

**BPU组配置：**配置哪些BPU可以连接上来，可以配置多组BPU，即一个BPC可以接受多种BPU的连接。子节点名称为BPU的名称，BPU连接时要指定此名称来连接。如上表配置中的monibpu和bpu2，在bpu的配置中，要指定此BPU是用哪个名称来连接BPC的。

**发送明文：**0为BPC不处理，直接发给BPU，C++的BPU使用，为1则由BPC解成明文发给BPU，JAVA的BPU使用1

**心跳应答：**0不用BPC给BPU心跳应答，C++的BPU使用，1需要BPC给BPU心跳应答，JAVA的BPU使用。

**通讯字节序：**0表示发送给BPU的字节序为LITTLE\_ENDIAN，1表示发送给BPU的字节序为BIG\_ENDIAN 。如果没有配置，默认为0。WINDOWS/LINUX字节序为LITTLE\_ENDIAN，AIX/HP\_UX为BIG\_ENDIAN，同时JAVA内部也是使用BIG\_ENDIAN。

**分配模式：**0表示只有BPU空闲时才分配请求给该BPU，C++的BPU使用，1表示BPU数只能为1，请求来了就发给该BPU，不管是否空闲，JAVA的BPU使用。

**可启动的BPU数：**此数目表示最多有多少个BPU可以连接BPC。超过此数目的BPU连接将被拒绝。

**AUTOLOADBPU：**此配置配置是否由BPC主动启动管理BPU，只有在一台机器上才能使用此配置。相当于BPC作为进程守护来管理启动BPU。属性use为1时由BPC管理启动BPU，启动BPU的参数见如下配置

**BPU程序名：**BPU的主程序名称

**BPU程序路径：**BPU主程序所在的目录，若是java的bpu，为java所在的目录，但注意目录里的java一定要是实际的程序，不能是link。

**BPU运行目录：**指定BPU在哪个目录下运行

**BPU启动参数：**BPU启动时的参数，参数不能大于2k。

#### 运行说明

BPC连接总线，在总线上注册，同时BPC接受BPU的连接。BPC在总线和业务处理之间起到一个桥梁的作用。

BPC相当于服务访问点(SAP)或服务连接器(AP adapter)，在具体的业务和总线间搭建一座桥梁。

BPC和总线间通讯业务报文前会加DREB报文头，和BPU间通讯再加上BPC报文头

 BPC报文头由7部分组成，共17字节。注：在DREB节点间是没有BPC头数据的，BPC头数据只是在BPC和BPU之间通讯使用。

##### BFIELD1-消息类型

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 消息类型 | 1  Char  取值为asc值 | 0 | BPC发给BPU的业务请求 MSG\_REQ |
| 1 | BPU通知BPC我就绪，可处理业务请求 MSG\_FREE |
| 4 | BPU取共享唯一流水 多个BPU共享此流水MSG\_GETSERIAL |
| 5 | BPU发送SETGPARA1,设置全局参数1 MSG\_SETGPARA1 |
| 6 | BPU发送SETGPARA2,设置全局参数2 MSG\_SETGPARA2 |
| 7 | BPU发送SETGPARA3,设置全局参数3 MSG\_SETGPARA3 |
| 9 | BPU 发送消息传递 MSG\_TRANS |
| 10 | BPU发送外调请求 MSG\_EXTCALL |
| 11 | BPU发送BPU组调用请求，即调用本BPC内的交易 MSG\_BPCCALL |
| 12 | BPU发送注册请求 MSG\_BPUREG |
| 13 | BPU发送是否注册请求，相同的BPU无需重复注册 MSG\_BPUISREG |
| 15 | DREB发过来的给BPU的取后续包的消息 MSG\_GETNEXT |

注：所有的上述消息在应答返回时，不更改消息类型，原样返回，只不过要置DREB头的FIELD4-RA标志为1。

##### BFIELD2-消息标志

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 消息标志 | Short  2 | 0x0128 | 使用LITTLE-ENDIAN字节序的程序填写的值 |
| 0x2801 | 使用BIG-ENDIAN字节序的程序填写的值 |

##### BFIELD3-源连接序号

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 源连接序号 | 2  short | 0-65534 | 接收dreb的数据时的连接的序号，标识此数据从哪个dreb连接上过来的 |

##### BFIELD4-BPU连接序号

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| SPU连接序号 | 2  short | 0-65534 | 接收BPU请求时BPU连接的序号 |

##### BFIELD5-连接时间戳

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 连接时间戳 | 4  int |  | BPU连接BPC时的连接时间戳，由BPC填写，发送应答时和原连接的时间戳进行比较，若不同表示连接已断并重用，取消发送。 |

##### BFIELD6-BPC内部用/BPU进程ID

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| SAP内部用/SPU进程ID | 4  Int | 当前时间秒/BPU进程ID | BPC内部队列使用，表示请求时间。  当为BPU注册时，填写为BPU的进程ID，BPC将此ID保存，若为本机则可进行管理。 |

##### BFIELD7-后续消息长度

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 长度 | 取值 | 说明 |
| 后续消息长度 | 2  short | 0-65534 | 不包含本报文头的长度，为DREB报文头加上业务数据的长度。根据此长度可知数据长度 |

### 业务处理单元bpu(服务处理单元SPU)

**下面以C++的BPU作说明**。

#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfbpu | 自动查找当前目录下的bfbpu.xml并启动, 在前台运行 |
| bfbpu xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在前台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfbpu stop | 退出当前目录下运行的所有BPU |
| bfbpu STOP | 退出当前目录下运行的所有BPU |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <package>  <head>  <公共配置>  <日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</日志级别>  <日志文件 remark="记录日志信息的文件名,前加日期">bpu.log</日志文件>  <日志目录 remark="日志文件写入的目录"></日志目录>  </公共配置>  <*PU配置>*  *<BPC*PORT remark="BPC侦听端口">29001</BPCPORT>  <BPCIP remark="BPC侦听端口">127.0.0.1</BPCIP>  <PU组名称 remark="为bfbpc.xml的package/head/BPU配置/BPU组/下的节点名称">monibpu</PU组名称>  <加密压缩 处理数据长度="200" 加密压缩标志="1" remark="0明文 1des加密密钥长度16，加密后长度增大 2压缩 3DES加密密钥长度8，加密后长度不变 4DES加密并压缩" />  </PU配置>  <业务配置>  <测试配置>  <AUTOLOADDLL remark="是否自动装载动态库1自动,测试时使用，生产禁止">0</AUTOLOADDLL>  <测试回路 testfilepath="/home/songfree/test" remark="为1从testfilepath目录下读取test+交易码.xml的文件数据内容返回">0</测试回路>  <时间测试 remark="为1将每一步的时间记录至日志文件中，程序优化使用，生产禁用">0</时间测试>  <原子交易转换 remark="为1将原子转换为对外提供服务的交易">1</原子交易转换>  </测试配置>  <业务动态库 dllpath="c:\libapp" remark="dllpath为业务动态库的路径，若不填或错误的路径将按PATH或LIBPATH来打开，在下面增加业务动态库">  <Sampled.dll method="getInstance" remark="业务原子功能动态库,method为实例化方法名" />  </业务动态库>  <流程控制配置 use="0" dllpath="/home/songfree/lib" dllname="libbf\_flowcontrol.so" method="getFlowInstance" remark="流程控制,use为1起作用" />  <数据库配置 use="0" commit="0" remark="此处数据库的连接配置，可通过工具修改,use是否连接数据库0不连1连接commit交易(非原子流程)是否自动提交事务1提交">  <databaseserver remark="数据库名字">GOLDDB218</databaseserver>  <databaseuser remark="数据库用户名">pmisppsbc</databaseuser>  <userpasswd remark="数据库密码">pmisppsbc</userpasswd>  <passwdtype remark="数据库密码加密标志1加密0不加密">0</passwdtype>  <databasetype remark="数据库类型0-oracl 1-sybase 2-sqlserver 3-infomi 4-db2 5-odbc 6-mysql">0</databasetype>  </数据库配置>  </业务配置>  </head>  <pack>  </pack>  </package> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmddx，当为报文日志时，文件名为yyyymmddx.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录

**PU配置：**配置连接BPC的一些参数，如IP、端口、BPU组名称等。

**BPCPORT：**对应BPC配置文件里的BPU侦听端口，

**BPCIP：** BPC所在机器的IP地址

**PU组名称：**对应BPC配置文件里的BPU组名称，标识此BPU是何BPU。

**加密压缩：**定义发送给BPC的报文是否加密压缩，属性”处理数据长度”表示数据大于等于此长度就进行加密压缩处理。在DREB里也可以进行压缩，但若在BPU时处理才是效率较高的一种方式。

**业务配置：**配置BPU要加载的动态库，连接数据库参数，是否测试回路档板等。

**测试配置：**配置每次重新加载动态库、测试回路档板、交易时间等

**AUTOLOADDLL：**为1表示每次交易前重新加载业务动态库

**测试回路：**为1时表示设置成档板，按交易码读取指定目录下的文件返回。

**时间测试：**为1时将交易处理时间写入日志，可通过查看日志得知交易处理的时间。

**原子交易转换：**为1时将内部的原子交易对外进行注册，使内部的原子交易可通过工具进行单元测试。主要是配合业务流程控制(工作流)使用。

**业务动态库：**配置要加载的业务动态库，可多个。业务动态库是继承框架的业务基类来实现的。

**流程控制配置：**业务流程控制(工作流)的配置，指定工作流工作的动态库名称等，同样是继承框架的工作流基类实现。

**数据库配置：**配置连接数据的参数，可通过changedbpwd工具来修改。

#### 运行说明

BPU连接BPC，并在BPC上注册交易**。**C/C++的BPU采用多进程来实现并发，所以BPU数一般大于1，使用多进程的好处是简化BPU开发要求，不需要开发人员掌握多线程开发技术，而把重点关注到业务开发。同时运行时一个BPU死掉后，还有多个BPU在提供服务，服务不会中断。

目前总线BPC框架的数据缓冲大小是64K，但有的网络路由配置时，是以8k大小进行数据路由的，如果BPU的单个数据包大小超时8k，可能导致数据发送接收出现问题，因此在JAVA版的BPU中，有一个参数来配置多记录应答时单个数据包的最大大小。C/C++的版的BPU要在AnsData方法中指定数据包大小不能超过8k。

两个版本的BPU的多包应答方式不同，JAVA版是把所有的多记录数据生成后再应答给BPC，如果大小超过配置参数，则自动按数据包大小分成多数据包，每个数据包之间有次序之分，前端收到应答包后，要按次序重新生成数据包再进行解析展示; C/C++版的BPU是提供了一个AnsData方法，可以一边生成记录(比如从数据库取一条记录)一边应答，每个应答数据包是一条完整的记录，前端接收一条展示一条然后再接收后续记录直到结束（这种方式可用来针对长时间处理的交易，报告给前端当前的进度）。同时C/C++的应答也支持JAVA版的应答方式。

### 接入网关cgate

网关cgate是接受管理端和交易端的连接(大量的客户连接，可通过数字证书认证)，并使用服务方式连接至总线DREB。网关将客户端发过来的请求发往总线，同时将总线发来的的应答、推送及广播发给指定的客户端。



#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfcgate | 自动查找当前目录下的cgate.xml并启动, 在后台运行 |
| bfcgate xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在后台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfcgate stop | 退出当前目录下运行的bfcgate |
| bfcgate STOP | 退出当前目录下运行的bfcgate |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <网关>  <公共参数>  <日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</日志级别>  <日志文件名 remark="记录日志信息的文件名,前加日期">cgate.log</日志文件名>  <日志目录 remark="存放日志的目录"></日志目录>  <心跳时间 remark="连接未收到数据并送心跳的时间 秒">20</心跳时间>  <未使用断开时间 remark="连接在此时间内未收到数据，则断开 秒">40</未使用断开时间>  <消息分配时间 remark="消息在队列里的超时时间 秒">30</消息分配时间>  <加密压缩标志 remark="应答给客户端的数据是否用会话密钥加密,明文0,1DES加密">1</加密压缩标志>  <解压缩 remark="收到客户端的数据是否解压缩">0</解压缩>  <行情过期时间 remark="队列中的行情超过此时间将丢弃 秒">5</行情过期时间>  <行情保留数目 remark="行情队列里保留最新的行情数目">40</行情保留数目>  <客户发送线程数 remark="针对客户端发送的线程数目">10</客户发送线程数>  <单次发送窗口 remark="一个客户连接网络连接由不正常转为正常时发次的数据包数">16</单次发送窗口>  </公共参数>  <IO模型>  <运行模式 remark="模式0-select模型1-epoll模型(LINUX）2-iocp模型(WINDOWS">0</运行模式>  <侦听端口 remark="网关的侦听端口">8001</侦听端口>  <最大连接数 remark="接收客户端的最大连接数">5000</最大连接数>  <IO线程数 remark="处理接收客户端的线程数，EPOLL为1,iocp为处理器个数">4</IO线程数>  <IOCPACCEPT数 remark="IOCP时发送AcceptEx的个数">5</IOCPACCEPT数>  </IO模型>  <通讯总线>  <公共服务号 remark="注册在总线上的公共服务号">21712</公共服务号>  <私有服务号 remark="私有服务号">4</私有服务号>  <总线节点 remark="连接DREB的信息">  <dreb1 ip="20.13.0.215" port="13921" 带宽="1000" remark="dreb的IP及端口" />  <dreb2 ip="20.13.0.215" port="13922" 带宽="1000" remark="dreb的IP及端口" />  </总线节点>  </通讯总线>  <监控配置 use="1" host="1" remark="use为主动报告选项，0不报告 host为主机资源报告选择0不报告">  <监控公共节点号>0</监控公共节点号>  <监控私有节点号>0</监控私有节点号>  <监控公共服务号>0</监控公共服务号>  <监控私有服务号>0</监控私有服务号>  <监控报告交易码>99001</监控报告交易码>  </监控配置>  <PKI认证 use="1" dllpath="" dllname="cfcaapid.dll" method="getInstance" certpath="./cfca" certpwd="11111111" certpwdflag="0" remark="是否使用证书及证书动态库目录,名称、实例化方法 " />  </网关> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmddx，当为报文日志时，文件名为yyyymmddx.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录。

**心跳时间：**网关和连接的客户端之间的心跳时间，当超时此时间未收到数据，则网关主动发送心跳，客户端对心跳进行应答，维持长连接，否则网关将主动断开连接。

**未使用断开时间：**当连接超过此时间未收到数据，网关主动断开连接，故此参数一定要大于心跳时间。

**消息分配时间：**网关将客户端的请求接收到后，放入到消息队列，由分配线程将消息发给总线，当消息在队列里的时间超过此时间后，将消息丢弃，不再发往后台处理。

**加密压缩标志：** 网关应答给客户端的数据是否进行会话密钥加密，一般此标志配置为1。

**解压缩：**网关收到客户端的数据后，先用会话密钥解密，解密后的数据为压缩的，此参数配置为是否要解压缩。一般此标志配置为0，不需要解压缩，由BPC或BPU进行解压缩即可，否则总线数据路由时还要进行压缩。

**行情过期时间：**当客户的网络不够快时，行情数据较多，会缓存在客户连接的队列中，当连接可写时再发，如果行情在队列中的时间超过此设置值，行情将丢弃，不再发送。

**行情保留数目：**当客户的网络不够快时，行情数据较多，会缓存在客户连接的队列中，但队列中行情有个上限，当超过此上限时，将最前面未发的完整行情丢弃。

**客户发送线程数：**由多个线程从队列里取数据，发给客户，这里配置有多少个线程来发送，当线程数不够多时，可能导致数据拥堵。

**单次发送窗口：**数据放在客户连接的队列时，是由IO线程来处理发送，如果某一队列数据较多，如果一直发送完，会导致其它客户响应慢，此参数表示当发送完此定义的数据后就转为处理其它数据，当一个偱环结束后再来处理发送。

**监控配置：**同总线的监控配置, 属性use设置为1时，主动发送监控报告，属性host设置为1时，监控报告里包含主机信息(cpu，内存，磁盘)。当为use为1时，下面的几个节点为监控服务所在的信息，若通过交易来调用，总线节点和服务全填0就可以了,否则必须填写。

**IO模型：**配置网关使用的侦听端口、IO使用的模型等一些参数。

**侦听端口：**接受客户端连接的侦听端口。

**运行模式：**IO模型，有select，IOCP及EPOLL三种，select针对所有的操作系统，IOCP仅windows使用，EPOLL为LINUX使用。根据使用的操作系统环境配置运行模式，目前IOCP未进行测试，可能有些问题。

**最大连接数：**网关接收客户端连接的最大数，超过此数将被拒绝。

**IO线程数：**此参数仅当模型为IOCP时才起作用

**IOCPACCEPT数：**此参数仅当模型为IOCP时才起作用

**通讯总线：**配置连接总线的一些参数，如在总线上注册的服务号，连接的多个总线IP及端口，本网关可同时连接多个总线，采用轮偱的方式发送数据。

**公共服务号：** 在总线上注册的公共服务号。

**私有服务号：** 在总线上注册的私有服务号，多个网关公共服务号相同，要注意私有服务号一定不能相同。

**总线节点：** 连接的多个总线信息。

**PKI认证：**配置客户端和网关之间通过数据证书认证的一些参数。不管采用何种证书及证书api，只要继承网关的接口基类并实现，在这里配置实现的参数即可。

**Use：**是否使用PKI认证，1表示需要，0表示不通过证书认证。

**Dllpath：**证书操作动态库的路径，证书操作动态库是继承接口基类CICertApiBase实现的。

**dllname：**动态库名称，证书操作动态库名字。

**Method：**实例化动态库操作类的C方法，网关通过调用此方法获取类的实例。

**Certpath：**数字证书所在的路径，证书操作所用。

**Certpwd：**数字证书的密码。

**Certpwdflag：**数字证书的密码标志，0为明文，1为加密，加密方式和BPU数据库用户密码相同。

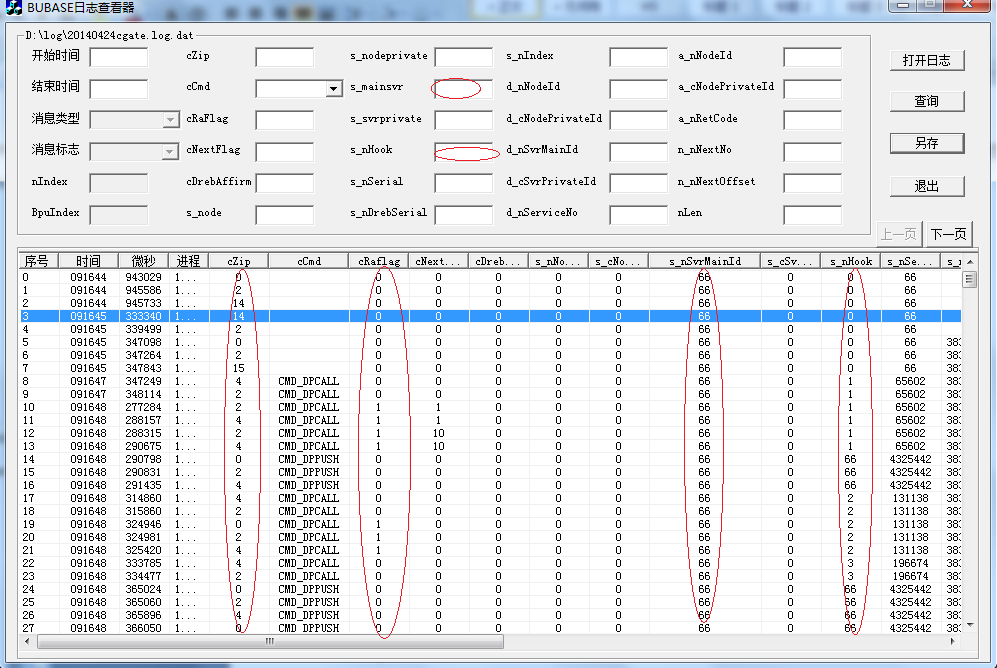
#### 运行说明

客户端和网关之前的连接使用长连接，网关为每个连接分配一个唯一的序号并将序号返回给客户端，要求客户端的每个请求都保证一个唯一的序号，这样就可以通过连接序号和请求序号在网关的日志中找到对应的报文等跟踪信息，从而在后台的BPU里也能找到。

目前框架提供了网关的API供客户端使用，网关和客户端之间的通讯报文协议暂未公开。网关api在日志级别大于等于5时，会将报文记录到dat文件中，可用日志查看工具logview来查看。

当网关的日志级别大于等于5时，会将客户端及总线发过来的报文记录到dat中，供logview查看，分成两个dat文件，多dreb字样的为总线报文日志文件。

在用logview查看客户端报文日志时，连接序号和请求序号放在s\_nSvrMainId和s\_nHook字段中，cRaFlag取值为两个，1为应答，0为请求。cZip标志0为明文，2为压缩，4为会话密钥加密并压缩，14为固定DES加密并压缩，15为证书加密并压缩。



### 前置处理中心fpc

前置处理中心FPC和前置处理单元FPU是为了快速开发（简单的应用程序和前置程序）而设计的，FPC进行端口侦听，接受外部连接和FPU的连接并接收数据，将外部发来的数据发给空闲的FPU进行处理（请求处理），将FPU发来的数据发给外部连接(处理应答)。



FPC接收的报文结构为：报文头+报文体，报文头为一段数字，表示后续的报文体的长度，例如定议报文头长度为5，报文体长度为4096，则报文头的内容为04096，注意不足报文头长度，前面补0。

#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bffpc | 自动查找当前目录下的bffpc.xml并启动, 在后台运行 |
| bffpc xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在后台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bffpc stop | 退出当前目录下运行的bffpc |
| bffpc STOP | 退出当前目录下运行的bffpc |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <package>  <head>  <公共配置>  <日志级别 name="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</日志级别>  <日志文件 name="记录日志信息的文件名,前加日期">bffpc.log</日志文件>  <日志目录 remark="存放日志的目录"></日志目录>  </公共配置>  <前置配置>  <LISTENPORT name="对外提供服务的侦听端口">9001</LISTENPORT>  <通讯报文头长度 name="通讯报文头长度,最小为4">5</通讯报文头长度>  <最大连接数 name="可以同时接受socket连接的最大数，最大640">500</最大连接数>  <未使用断开时间 name="如果在连接池时间超过此设置则断开，单位秒，最小60">600</未使用断开时间>  <心跳时间 name="超时此时间若连接未使用，主动发送心跳秒">600</心跳时间>  <最大流水计数 name="线程共享的流水最大值，超过则从1重新开始">99999999</最大流水计数>  <本次流水开始值 name="线程共享的流水起始值">1</本次流水开始值>  <请求分配超时时间 name="请求分配给BU的超时时间，秒">10</请求分配超时时间>    </前置配置>  <BU配置>  <可启动的BU数 name="即可以处理业务的单元数目">10</可启动的BU数>  <BU侦听端口 name="接受BU连接上来的端口">9002</BU侦听端口>  <AUTOLOADBU use="0" name="自动将BU程序启动1，否则为0">  <BU程序名 name="BU程序的名称，不带路径">bffpud.exe</BU程序名>  <BU程序路径 name="绝对或相对路径">D:\BUBASE\bin</BU程序路径>  <BU运行目录 name="绝对或相对目录">D:\BUBASE\bin</BU运行目录>  <BU启动参数 name="启动BU所需的参数">D:\BUBASE\bin\bffpu.xml</BU启动参数>  </AUTOLOADBU>  </BU配置>  </head>  <pack />  </package> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmddx，当为报文日志时，文件名为yyyymmddx.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录。

**LISTENPORT**：接收外部连接的侦听端口。

**通讯报文头长度**：定义报文头的长度，为大于等于0的值。例如定义报文头长度为5，报文体长度为4096，则报文头的内容为04096，注意不足报文头长度，前面补0。

**最大连接数**：FPC接受连接的最大数目，超时此数目的连接被拒绝，最小值为100。

**未使用断开时间**：当连接超过此时间仍未收到数据后，FPC主动断开连接。最小值为6秒

**心跳时间**：当连接超过此时间未收到数据时，主动发送心跳，单位为秒，若未配置默认为5秒。FPC主动发送的心跳报文体为”xt”两个字符，客户端主动发送给FPC的心跳报文为只有报文头无报文体。

**最大流水计数**：多个FPU在FPC共享的流水计数(可作为交易流水序号)的最大值，超时此值则从1重新开始。

**本次流水开始值**：当FPC启动时，读入此配置值作为流水计数的开始值，当FPC重启后，重新从此值开始。

**请求分配超时时间**：配置FPC将请求分配给FPU的超时时间，单位为秒。若不配置默认为10秒。超时后将此请求丢弃，无应答。

**BU配置：**配置针对FPU连接的侦听端口及FPU启动的参数。

**BU侦听端口：**接收FPU连接上来的端口。

**可启动的BU数：**此数目表示最多有多少个FPU可以连接FPC。超过此数目的FPU连接将被拒绝。

**AUTOLOADBU：**此配置是否由FPC主动启动管理FPU，只有在一台机器上才能使用此配置。相当于FPC作为进程守护来管理启动FPU。属性use为1时由FPC管理启动FPU，启动FPU的参数见如下配置

**BU程序名：**FPU的主程序名称

**BU程序路径：F**PU主程序所在的目录。

**BU运行目录：**指定FPU在哪个目录下运行

**BU启动参数：**FPU启动时的参数，参数不能大于2k。

#### 运行说明

FPC在运行时，对接收到的数据不做转换，直接将报文体数据发给FPU进行处理，FPU处理后将应答数据发给FPC，由FPC再应答给请求方。FPC是多线程程序，FPU是单进程程序，启动多个FPU来实现并发。这样做的好处是一个FPU的交易出问题后，不影响其它的FPU继续服务，相当于降低了开发人员要求，同时性能也不会降低。

### 前置处理单元fpu

配合FPC一起使用，FPU同BPU类似，开发也是使用动态库的方式，继承的基类也相同。

#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bffpu | 自动查找当前目录下的bffpu.xml并启动, 在前台运行 |
| bffpu xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在前台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bffpu stop | 退出当前目录下运行的所有FPU |
| bffpu STOP | 退出当前目录下运行的所有FPU |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <package>  <head>  <公共配置>  <日志级别 name="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</日志级别>  <日志文件 name="记录日志信息的文件名,前加日期">fpu.log</日志文件>  <日志目录 remark="日志文件写入的目录"></日志目录>  </公共配置>  <BU配置>  <BCCPORT name="bcc侦听端口">9001</BCCPORT>  <BCCIP name="bcc侦听端口">127.0.0.1</BCCIP>  <是否加密压缩 name="0数据不加密1数据包加密2数据压缩3DES加密4DES加密并压缩">0</是否加密压缩>  <交易统一入口 use=”0” 统一入口功能号="10000" remark=”所有的请求都交给一个交易码里进行处理” />  </BU配置>  <业务配置>  <测试配置>  <AUTOLOADDLL remark="是否自动装载动态库1自动,测试时使用，生产禁止">0</AUTOLOADDLL>  <测试回路 testfilepath="/home/songfree/test" remark="为1从testfilepath目录下读取test+交易码.xml的文件数据内容返回">0</测试回路>  <时间测试 remark="为1将每一步的时间记录至日志文件中，程序优化使用，生产禁用">0</时间测试>  <原子交易转换 remark="为1将原子转换为对外提供服务的交易">1</原子交易转换>  </测试配置>  <业务动态库 dllpath="D:\BUBASE\libapp" remark="dllpath为业务动态库的路径，若不填或错误的路径将按PATH或LIBPATH来打开，在下面增加业务动态库">  <SampleNewd.dll method="getInstance" remark="业务原子功能动态库,method为实例化方法名" />  </业务动态库>  <流程控制配置 use="0" dllpath="/home/songfree/lib" dllname="libbf\_flowcontrol.so" method="getFlowInstance" remark="流程控制,use为1起作用" />  <数据库配置 use="0" commit="0" remark="此处数据库的连接配置，可通过工具修改,use是否连接数据库0不连1连接commit交易(非原子流程)是否自动提交事务1提交">  <databaseserver remark="数据库名字">ORCL</databaseserver>  <databaseuser remark="数据库用户名">ylivsd</databaseuser>  <userpasswd remark="数据库密码">ylivsd</userpasswd>  <passwdtype remark="数据库密码加密标志1加密0不加密">0</passwdtype>  <databasetype remark="数据库类型0-oracl 1-sybase 2-sqlserver 3-infomi 4-db2 5-odbc 6-mysql">0</databasetype>  </数据库配置>  </业务配置>  </head>  <pack>  </pack>  </package> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmddx，当为报文日志时，文件名为yyyymmddx.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录

**BU配置：**配置连接FPC的一些参数，如IP、端口等。

**BPCPORT：**对应FPC配置文件里的FPU侦听端口，

**BPCIP：** FPC所在机器的IP地址

**是否加密压缩：**定义发送给FPC的报文是否加密压缩。0数据不加密1数据包加密2数据压缩3DES加密4DES加密并压缩。

**交易统一入口**：当use为1时，表示非xml的报文交给固定的交易(统一入口功能号定义的交易码)进行处理。框架支持原XMLPACK的报文，若是XMLPACK报文则取报文里的交易码，交给业务动态库里的XMLPACK参数的方法进行处理。否则转换为XMLPACK的BINDATA方式交给业务动态库里的XMLPACK参数的方法(对应的统一入口功能号)进行处理。

**业务配置：**配置BPU要加载的动态库，连接数据库参数，是否测试回路档板等。

**测试配置：**配置每次重新加载动态库、测试回路档板、交易时间等

**AUTOLOADDLL：**为1表示每次交易前重新加载业务动态库

**测试回路：**为1时表示设置成档板，按交易码读取指定目录下的文件返回。

**时间测试：**为1时将交易处理时间写入日志，可通过查看日志得知交易处理的时间。

**原子交易转换：**为1时将内部的原子交易对外进行注册，使内部的原子交易可通过工具进行单元测试。主要是配合业务流程控制(工作流)使用。

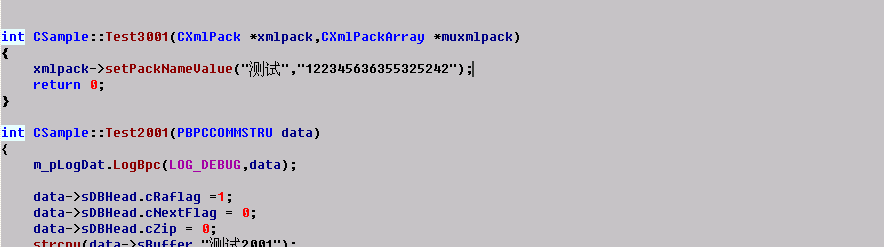
**业务动态库：**配置要加载的业务动态库，可多个。业务动态库是继承框架的业务基类来实现的。

**流程控制配置：**业务流程控制(工作流)的配置，指定工作流工作的动态库名称等，同样是继承框架的工作流基类实现。

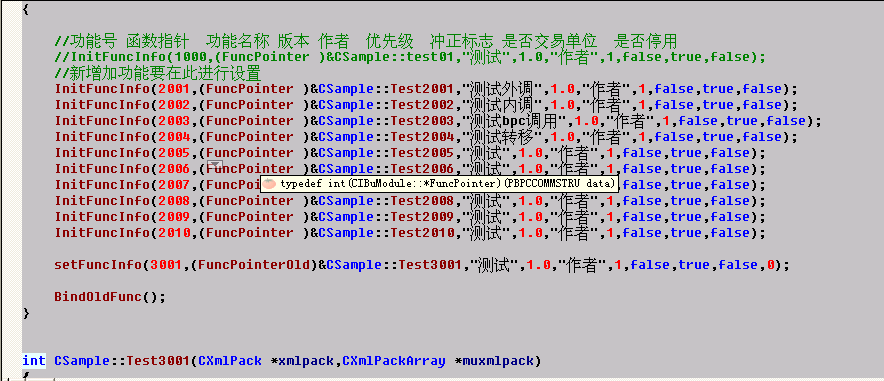
**数据库配置：**配置连接数据库的参数，可通过changedbpwd工具来修改。

#### 运行说明

在开发FPU时，动态库的工程可由应用程序向导生成，但在编写代码时，要注意交易有两种参数形式，如下：



**上图的Test3001为兼容旧版框架的XMLPACK报文数据; Test2001为新版报文格式，没有固定报文。本FPU要使用Test3001的格式，在构造方法里有区别，如下图**

****

**setFuncinfo是将旧版的交易注册到BPU中，并使用BindOldFunc方法将旧版的交易转为新版的交易方式，这样BPU和FPU的业务动态库可以互相兼容。**

### 进程守护daemon

进程守护daemon是一个守护进程程序，由它来管理进程的停止、启动来状态的报告，通过连接总线，将进程的状态及主机资源信息报告给监控服务。

#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfdaemon | 自动查找当前目录下的bfdaemon.xml并启动, 在后台运行 |
| bfdaemon xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在后台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfdaemon stop | 退出当前目录下运行的进程守护 |
| bfdaemon STOP | 退出当前目录下运行的进程守护 |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <package>  <head>  <公共配置>  <日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</日志级别>  <日志文件 remark="记录日志信息的文件名,前加日期">daemon.log</日志文件>  <日志目录 remark="日志存放的目录"></日志目录>  <公共服务号 remark="在dreb上注册的服务号">1010</公共服务号>  <私有服务号 remark="在dreb上注册的私有服务号">2</私有服务号>  <DREB心跳时间 remark="和DREB的心条时间,秒">5</DREB心跳时间>  <DREB连接过期时间 remark="当连接长时间未有数据时断开,秒">60</DREB连接过期时间>  <连接过期时间 remark="当连接长时间未有数据时断开,秒">600</连接过期时间>  <心跳侦听端口 remark="接收管理的进程的心跳端口">29000</心跳侦听端口>  </公共配置>  <监控配置 use="1" host="1" remark="use为主动报告选项，0不报告 host为主机资源报告选择0不报告">  <监控公共节点号>100</监控公共节点号>  <监控私有节点号>0</监控私有节点号>  <监控公共服务号>100</监控公共服务号>  <监控私有服务号>0</监控私有服务号>  <监控报告交易码>99001</监控报告交易码>  </监控配置>  <DREB配置>  <DREB1 ip="20.13.0.217" port="13911" bandwidth="100" remark="配置数据总线的参数" />  <DREB2 ip="20.13.0.217" port="13912" bandwidth="100" remark="配置数据总线的参数" />  </DREB配置>  <进程守护>  <1001>  <进程名称>通讯接口机服务器</进程名称>  <程序名 remark="程序的名称，不带路径">testdaemond.exe</程序名>  <程序路径 remark="绝对或相对路径">\\tsclient\d\BUBASE\bin</程序路径>  <运行目录 remark="绝对或相对目录">\\tsclient\d\BUBASE\bin</运行目录>  <启动参数 remark="启动进程所需的参数">\\tsclient\d\BUBASE\bin\testdaemon.xml</启动参数>  <停止脚本 remark="停止进程的脚本，为空直接kill掉"></停止脚本>  <无心跳是否重启 remark="为1时表示两个心跳时间没有收到心跳则重启进程">1</无心跳是否重启>  <是否接受控制 remark="为1时表示可由此程序进行停止">1</是否接受控制>  </1001>  <2001>  <进程名称>通讯接口机服务器</进程名称>  <程序名 remark="程序的名称，不带路径">bfB.exe</程序名>  <程序路径 remark="绝对或相对路径">D:\BUBASE\bin</程序路径>  <运行目录 remark="绝对或相对目录">D:\BUBASE\bin</运行目录>  <启动参数 remark="启动进程所需的参数">D:\BUBASE\bin\bfBPU.xml</启动参数>  <停止脚本 remark="停止进程的脚本，为空直接kill掉">D:\BUBASE\bin\bfBPU.xml</停止脚本>  <无心跳是否重启 remark="为1时表示3个心跳时间没有收到心跳则重启进程">0</无心跳是否重启>  <是否接受控制 remark="为1时表示可由此程序进行停止">0</是否接受控制>  </2001>  </进程守护>  </head>  <pack>  </pack>  </package> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmddx，当为报文日志时，文件名为yyyymmddx.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录。

**公共服务号：**进程守护是作为服务注册到总线上的，此处配置服务的公共编号

**私有服务号：**本服务的私有编号，公共+私有编号是全网唯一的。

**DREB心跳时间**：和总线的心跳时间，单位秒，当超时此时间未收到总线数据时，主动发送心跳包给总线，总线会心跳应答。

**DREB连接过期时间**：单位秒，当和总线的连接超时此时间仍未收到数据时，主动断开连接。

**连接过期时间**：心跳连接的过期时间，当所守护的进程的心跳连接超时此时间未收到数据时，主动断开连接。是否重启进程由进程配置的无心跳是否重启参数来控制。

**心跳侦听端口**：接受所守护的进程的心跳连接。心跳报文为报文头+报文体组成，报文头固定为8个字节，表示报文体的长度，不足8位前补0。报文体为KV格式报文，无固定报文头，其中必须有node\_id字段，此字段和进程编号相同，表示是哪个进程所发的心跳包。

**监控配置：**同总线的监控配置, 属性use设置为1时，主动发送监控报告，属性host设置为1时，监控报告里包含主机信息(cpu，内存，磁盘)。当为use为1时，下面的几个节点为监控服务所在的信息，若通过交易来调用，总线节点和服务全填0就可以了,否则必须填写。

**DREB配置：**配置连接总线节点，可以连接多个负载均衡的总线节点。

**进程守护**：配置守护的进程信息，子节点名为进程在整个总线范围内的进程编号

【**进程编号**】：唯一编号，进程发送心跳时必须在KV报文体的node\_id节点填上此值。

**进程名称**：进程的中文名称

**程序名**：运行的程序名

**程序路径**：运行程序所在的目录

**运行目录**：进程的运行目录

**启动参数**：进程启动的参数

**停止脚本**：停止进程的脚本，若无则使用kill来停止进程

**无心跳是否重启**：当进程没有心跳时，是否由进程守护重启该进程(kill后重启)

**是否接受控制**：当接收控制时，可以进程守护来重启、停止、KILL该进程，这样可以在总线的任意一点来控制该进程的状态。控制报文参见监控设计。

#### 运行说明

进程守护可以连接总线，将所守护的进程及主机资源信息发给监控服务，也可以不连总线单机运行。

### 透明网关bfgw

#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfgw | 自动查找当前目录下的bfgw.xml并启动, 在后台运行 |
| bfgw xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在后台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| bfgw stop | 退出当前目录下运行的透明网关 |
| bfgw STOP | 退出当前目录下运行的透明网关 |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <package>  <head>  <公共配置>  <日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</日志级别>  <日志文件 remark="记录日志信息的文件名,前加日期">bfgw.log</日志文件>  <日志目录 remark="日志存放的目录"></日志目录>  </公共配置>  <网关配置>  <公共服务号 remark="在dreb上注册的服务号">1010</公共服务号>  <私有服务号 remark="在dreb上注册的私有服务号">1</私有服务号>  <DREB心跳时间 remark="和DREB的心条时间,秒">5</DREB心跳时间>  <DREB连接过期时间 remark="当连接长时间未有数据时断开,秒">60</DREB连接过期时间>  <转发连接过期时间 remark="当连接长时间未有数据时断开,秒">60000</转发连接过期时间>  </网关配置>  <监控配置 use="1" host="1" remark="use为主动报告选项，0不报告 host为主机资源报告选择0不报告">  <监控公共节点号>100</监控公共节点号>  <监控私有节点号>0</监控私有节点号>  <监控公共服务号>100</监控公共服务号>  <监控私有服务号>0</监控私有服务号>  <监控报告交易码>99001</监控报告交易码>  </监控配置>  <DREB配置>  <DREB1 ip="20.13.0.217" port="13911" bandwidth="100" remark="配置数据总线的参数" />  <DREB2 ip="20.13.0.217" port="13912" bandwidth="100" remark="配置数据总线的参数" />  </DREB配置>  <转发配置>  <转发1 INPUTPORT="3000" OUTPUTIP="20.13.0.217" OUTPUTPORT="23" />  <转发2 INPUTPORT="3001" OUTPUTIP="20.13.0.217" OUTPUTPORT="13913" />  <转发3 INPUTPORT="3002" OUTPUTIP="20.13.0.217" OUTPUTPORT="13914" />  </转发配置>  <转发黑名单>  <黑名单 IP="20.13.0.217" />  <黑名单 IP="20.13.0.218" />  <黑名单 IP="20.13.0.216" />  </转发黑名单>  </head>  <pack>  </pack>  </package> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmddx，当为报文日志时，文件名为yyyymmddx.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录。

**公共服务号：**进程守护是作为服务注册到总线上的，此处配置服务的公共编号

**私有服务号：**本服务的私有编号，公共+私有编号是全网唯一的。

**DREB心跳时间**：和总线的心跳时间，单位秒，当超时此时间未收到总线数据时，主动发送心跳包给总线，总线会心跳应答。

**DREB连接过期时间**：单位秒，当和总线的连接超时此时间仍未收到数据时，主动断开连接。

**转发连接过期时间**：当连接长时间未收到数据时，主动断开的时间，单位秒

**监控配置：**同总线的监控配置, 属性use设置为1时，主动发送监控报告，属性host设置为1时，监控报告里包含主机信息(cpu，内存，磁盘)。当为use为1时，下面的几个节点为监控服务所在的信息，若通过交易来调用，总线节点和服务全填0就可以了,否则必须填写。

**DREB配置：**配置连接总线节点，可以连接多个负载均衡的总线节点。

**转发配置**：配置网关的转入及转出端口，即转入端口对外，当转入端口有连接时，主动连接转出的IP及端口，在这两个连接间建立直通关系。

**转发黑名单**：配置禁止连接过来的客户端IP。比如F5不停的会连连接探测端口，可以在这里把F5的IP配置上，就会减少不必要的连接。

#### 运行说明

透明网关可以连接总线，将主机资源信息发给监控服务，也可以不连总线单机运行。

### 主机资源monitor

本主机资源是原来的主机资源管理器，仅限不使用总线的二级系统使用。

#### 启动

|  |  |
| --- | --- |
| 命令行 | 说明 |
| monitor | 自动查找当前目录下的monitor.xml并启动, 在后台运行 |
| monitor xx.xml | 查找xx.xml并启动，若xx.xml不带路径则在当前目录下查找, 在后台运行 |

#### 停止

|  |  |
| --- | --- |
| 命令行 | 说明 |
| monitor stop | 退出当前目录下运行的主机资源 |
| monitor STOP | 退出当前目录下运行的主机资源 |

#### 配置文件

xml配置文件内容

|  |
| --- |
| <?xml version="1.0" encoding="GB2312"?>  <package>  <head>  <公共配置>  <日志级别 remark="日志跟踪级别0严重错误1重要错误2一般错误3警告4提示警告5调试信息">5</日志级别>  <日志文件 remark="记录日志信息的文件名,前加日期">monitor.log</日志文件>  <日志目录 remark="日志存放的目录"></日志目录>  <主机ID remark="主机ID">1000</主机ID>  <主机名称 remark="主机名称">风控服务器</主机名称>  <节点ID remark="节点ID">1010</节点ID>  <心跳时间 remark="和监控服务器的心跳时间,秒">5</心跳时间>  <监控服务连接过期时间 remark="当连接长时间未有数据时断开,秒">600000</监控服务连接过期时间>  <连接过期时间 remark="管理的进程心跳连接过期时间,秒">600</连接过期时间>  <心跳侦听端口 remark="接收管理的进程的心跳端口">29000</心跳侦听端口>  </公共配置>  <监控服务端口配置>  <H1 ip="20.13.0.217" port="13911" remark="监控服务器H1端口" />  <H2 ip="20.13.0.217" port="13912" remark="监控服务器H2端口" />  </监控服务端口配置>  <进程守护>  <1001>  <进程名称>通讯接口机服务器</进程名称>  <程序名 remark="程序的名称，不带路径">testdaemond.exe</程序名>  <程序路径 remark="绝对或相对路径">\\tsclient\d\BUBASE\bin</程序路径>  <运行目录 remark="绝对或相对目录">\\tsclient\d\BUBASE\bin</运行目录>  <启动参数 remark="启动进程所需的参数">\\tsclient\d\BUBASE\bin\testdaemon.xml</启动参数>  <停止脚本 remark="停止进程的脚本，为空直接kill掉"></停止脚本>  <无心跳是否重启 remark="为1时表示两个心跳时间没有收到心跳则重启进程">1</无心跳是否重启>  <是否接受控制 remark="为1时表示可由此程序进行停止">1</是否接受控制>  </1001>  <2001>  <进程名称>通讯接口机服务器</进程名称>  <程序名 remark="程序的名称，不带路径">bfB.exe</程序名>  <程序路径 remark="绝对或相对路径">D:\BUBASE\bin</程序路径>  <运行目录 remark="绝对或相对目录">D:\BUBASE\bin</运行目录>  <启动参数 remark="启动进程所需的参数">D:\BUBASE\bin\bfBPU.xml</启动参数>  <停止脚本 remark="停止进程的脚本，为空直接kill掉">D:\BUBASE\bin\bfBPU.xml</停止脚本>  <无心跳是否重启 remark="为1时表示3个心跳时间没有收到心跳则重启进程">0</无心跳是否重启>  <是否接受控制 remark="为1时表示可由此程序进行停止">0</是否接受控制>  </2001>  </进程守护>  </head>  <pack>  </pack>  </package> |

**日志级别：**为5时将DEBUG信息写入文本日志，大于5将一些更多的信息写入文本日志，在正常运行一段时间后无问题建议将日志级别设为4，压力测试时设为3或2。

**日志文件：**文件名不带路径，假设配置的是x，当为文件日志时，文件名为yyyymmddx，当为报文日志时，文件名为yyyymmddx.dat。

**日志目录：**不设置时则为当前目录下的log目录，此log目录会自动建立，当设置目录后，要确定目录确实存在，程序不会自动建立目录。

**主机ID：机器的编号**

**主机名称：主机中文名称**

**节点ID**：本应用程序的节点编号

**心跳时间**：和监控服务器的心跳时间，单位秒，注意JAVA版的监控服务器不主动发送心跳，也无心跳应答。

**监控服务连接过期时间**： 当连接超时此时间未收到数据主动断开连接，注意监控服务器无心跳应答，所以这个值要设置的很大，否则和监控服务器的连接会断开。

**连接过期时间**：心跳连接的过期时间，当此连接建立但超时此时间未收到数据时，关闭此连接。

**心跳侦听端口**：接受所守护的进程的心跳连接。心跳报文为报文头+报文体组成，报文头固定为8个字节，表示报文体的长度，不足8位前补0。报文体为KV格式报文，无固定报文头，其中必须有node\_id字段，此字段和进程编号相同，表示是哪个进程所发的心跳包。

**监控服务端口配置：**配置和监控服务器的两个连接，第一个为H1，第二个为H2。

**进程守护**：配置守护的进程信息，子节点名为进程在整个总线范围内的进程编号

【**进程编号**】：唯一编号，进程发送心跳时必须在KV报文体的node\_id节点填上此值。

**进程名称**：进程的中文名称

**程序名**：运行的程序名

**程序路径**：运行程序所在的目录

**运行目录**：进程的运行目录

**启动参数**：进程启动的参数

**停止脚本**：停止进程的脚本，若无则使用kill来停止进程

**无心跳是否重启**：当进程没有心跳时，是否由进程守护重启该进程(kill后重启)

**是否接受控制**：当接收控制时，可以进程守护来重启、停止、KILL该进程，这样可以在总线的任意一点来控制该进程的状态。控制报文参见监控设计。

#### 运行说明

### 前置测试工具testcomm

安装完BUBASE的WINDOWS开发包后，会在程序组下成工具->功能测试工具 这个程序项，通过它或点击安装目录的tools目录处的TestComm.exe即可进入界面。TestComm工具即可用来测试原子功能或交易，也可以用来测试定长或分隔符格式的数据包的前置功能。测试原子功能时需将应用服务的配置文件中业务配置段中的参数”**原子交易转换**”设置为1。

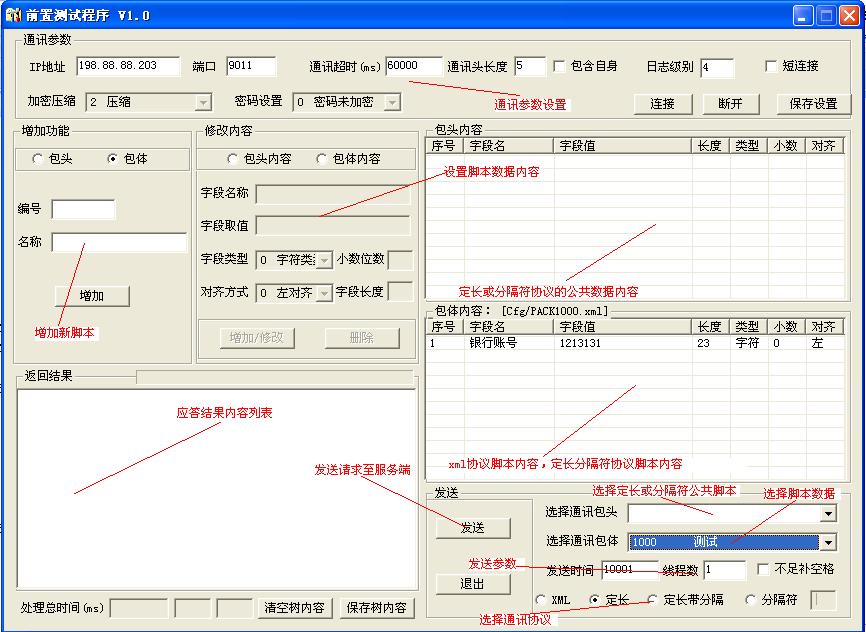


图3-1 TestComm界面

#### 通讯参数设置

如图3-1所示，通讯参数有IP地址、端口、接收socket超时、通讯头长度、及是否包含自身，日志级别，连接类型，前三项不用多说，中间两项要和服务配置文件中前置配置段的参数” 通讯报文头长度”及” 通讯报文头是否包含自身”相一致，包含自身选中值为1，日志级别指在发送数据时，可以将日志设置调试状态，将发送及收到的应答记录至日志文件Testcomm.log中，连接类型主要有短连接和长连接，若是短连接的情况，每个请求都先建立连接，用完后再关闭连接，长连接只在第一次请求时建立连接，只有在出错时才关闭重连。若在右下角选中的通讯数据协议为xml时，加密压缩及密码设置将为可选，加密压缩参数要和服务的配置文件中的参数”是否加密压缩”一致，否则会导致接收不了应答结果。密码设置主要和请求的数据包相关，**若选择”0 密码未加密”，则测试程序将会对密码字段进行加密，然后再发到后台**。若选择”1 密码已加密”则测试程序不对配置脚本里的密码字段进行处理，直接发往后台。注：密码字段名称有: 密码、验证码、旧验证码、新验证码、预留密码、账户密码、旧密码、新密码。

通讯参数设置完成后，按保存设置则将设置的参数保存到config.xml中。下次启动时自动从config.xml读取参数。

点击连接尝试去连接服务，测试IP及端口是否正确。

#### 增加测试脚本

通过图3-1的左边增加功能框里增加可以增加测试脚本，测试脚本分为通讯包头和通讯包体，通讯包头主要是定长及分隔符包格式使用。通讯包体所有的协议包都使用，通讯包头的脚本文件为HEAD+名称.xml，通讯包体的脚本文件为PACK+交易编码.xml文件，两种脚本文件皆放在cfg目录下。

##### 增加通讯包头脚本

此脚本只有定长或分隔符包才使用。XML通讯不需要此脚本。

如图3-1，点击包头单选按钮选择增加通讯包头脚本，然后在下面的编号及名称里输入通讯包头脚本编号及脚本说明，按增加即可增加一个脚本，脚本放在cfg目录下，文件名称为HEAD+编号.XML，增加成功后同时显示在右下角的选择通讯包头下拉列表框里。



图3-2 选择通讯包头脚本

##### 增加通讯包体脚本

通讯包体脚本不管是何种通讯格式，是一定要用的。如图3-1，在增加功能框里点击包体单选按钮，选择增加通讯包体脚本，然后在下面的编号及名称里输入交易的编码及交易名称，按增加即在cfg目录下生成一个脚本文件,文件格式为PACK+交易编码.xml文件，增加成功同时会在右下角的选择通讯包体的列表框里显示出来。



图3-3 选择通讯包体脚本

#### 增改删脚本内容

在增加成功脚本后，脚本还是一个空的脚本。可以直接编辑脚本文件或通过本工具来修改增加脚本内容。前面说过，脚本有两种类型，通讯包头和通讯包体，要修改，首先如图3-2或图3-3上选择通讯包头脚本或选择通讯包体脚本，在选择后，会在界面右边的包头内容框里和在界面右边的包体内容框里显示选择包头脚本和包体脚本。

此时在界面的中间选修改包头内容还是包体内容，然后可以对具体的内容进行增加修改删除，选中包头内容再在包头内容列表框里单击列数据，会将数据显示在中间供修改。选中包体内容再在包体内容列表框里单击列数据，会将数据显示在中间供修改。



图3-4 增改删脚本界面

##### 增改删通讯包头脚本内容

下面对脚本内容的字段作一些说明：

##### 字段名称

若是XML数据包时，此字段即为xml字段的名称，若是定长分隔符数据包时，则是字段的说明。

##### 字段取值

字段值是指发送请求时的数据内容。当为xml数据包时，是字段内容。当为非xml包时，如果是以下几种情况，则测试程序自动进行替换。

**##：为此内容时，发送数据时取选择的通讯包体脚本的交易码替换。**

**@DATE：为此内容时，发送数据时取系统日期(YYYYMMDD格式)替换。**

**@DATE18：为此内容时，发送数据时取系统日期时间(YYYY-MM-DD HH:MM:SS格式)替换。**

**@TIME：为此内容时，发送数据时取系统时间(HH:MM:SS格式)替换。**

**@TIME6：为此内容时，发送数据时取系统时间(HHMMSS格式)替换。**

**@SEC：为此内容时，发送数据时取tickcount时间(累积秒数)替换。**

**~n：为此内容时，发送数据时取n值+1(n为整型值)替换，每发送一次值加1。**

##### 字段类型

字段类型有三种，字符、浮点和整型，当数据包为xml时，此字段不起作用。

##### 小数位数

当数据包不为xml且当字段类型为浮点型时此字段才起作用。定长分隔时自动对字段值进行格式化输入。

##### 对齐方式

当数据包为定长或定长分隔时此字段才起作用。决定是作补齐长度还是右补齐长度。

##### 字段长度

当数据包为定长或定长分隔时此字段才起作用，根据此长度来组数据包。

##### 增改删通讯包体脚本内容

增改删同通讯包头操作。

#### 选择协议类型

选择数据包类型，有xml、定长、定长分隔、分隔四种类型，若选择了定长分隔或分隔，还需设置分隔符。选择了xml后，还需确定通讯参数中的加密压缩及密码设置是否正确。

#### 设置发送参数

发送参数有3个，有一个线程发送请求的个数、发送线程数、定长包时空值是否用空格代替。

线程数指启动的发送线程个数，通过多线程来模拟并发的情况。

**若发送次数小于10000则为单线程最大要发送请求的个数**，总共发送的请求数为 线程数 X 发送次数，当发送的请求数超过1000个时，将不会在结果栏里把结果显示出来。

**若发送次数大于等于10000时，则为发送时间**，指线程在指定的时间内不停的发送请求，每个线程在尝试连续三次连接不成功时停止线程发送。

#### 发送请求

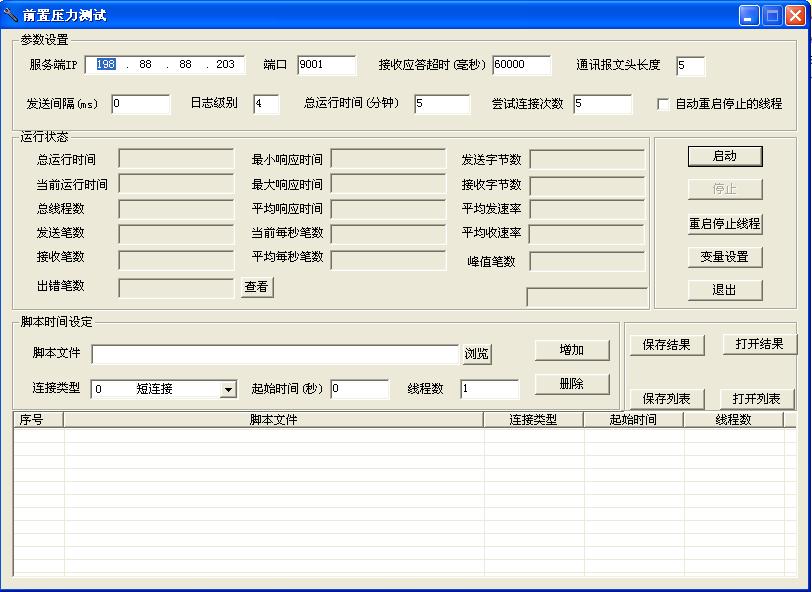
点击发送按钮后，测试程序根据选择的通讯包类型，根据选择通讯包头脚本和通讯包体脚本，组成发送的请求包，发至服务进行处理，若发送的总请求数小于1000，则将返回的结果在左边的列表框显示出来，可以查看返回的结果是否符合期望。当为xml数据包类型时，只取通讯包体脚本的内容进行组包。

#### 查看返回结果



### 前置压力测试工具testfront

此工具为压力测试工具，使用多线程将指定的数据包发给服务端并接收应答来测试服务端的性能。此工具是参考了loadrunner工具来做的，变量设置和其一样。



#### 参数设置

##### 服务端ip

要测试的服务端的IP地址

##### 端口

服务端提供的服务端口

##### 接收应答超时

接收应答的超时时间，若超过这个时间未应答，认为交易失败。在界面的运行状态的出错笔数将增加一笔，同样接收笔数和发送笔数就不相等。

##### 通讯报文头长度

当通讯报文头长度大于0时，自动在脚本前面加上脚本里的数据长度，如通讯报文头长度为4，脚本里的报文数据长度为991则实际发送的报文为0991xxx..，xxx..为脚本数据。当等于0时直接发送脚本里的报文数据。

##### 发送间隔

此参数是指在一个线程中，在发送给服务端并接收到应答后再次发送的间隔，可参数控制发送的频率。

##### 日志级别

日志级别是指本工具写入日志的级别，调试为5，一般情况下为4.

##### 总运行时间

指工具运行的时间

##### 尝试连接次数

当某一脚本定为短连接时，尝试连接服务端的次数。当次数超过此参数时，线程自动终止，不再发送。

##### 自动重启停止的线程

当线程终止后，如果不手动启动的话，将不会再运行，若需要线程停止的时候主动重启的话，请复选此按钮。

#### 脚本时间设定

##### 脚本文件

选择一个脚本文件，此文件为交易的报文数据，由测试者生成，脚本文件里可包含变量，由测试工具在测试时根据变量的定义计算填值。前提要在变量设置里增加此变量。变量由<xx>组成，其中xx为变量名称，要在变量设置里定义。

##### 连接类型

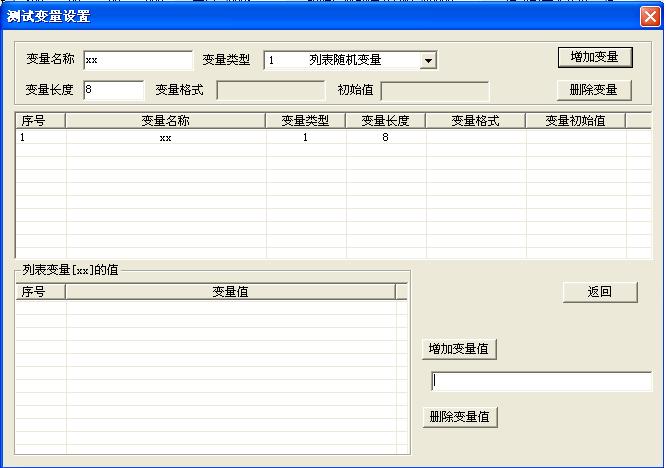
连接服务端的类型，有长连接和短连接，长连接即只连接一次，后续的请求一直通过此连接发送并接收应答，直至结束。短连接为每次发送前进行连接，接收应答后断开。

##### 起始时间

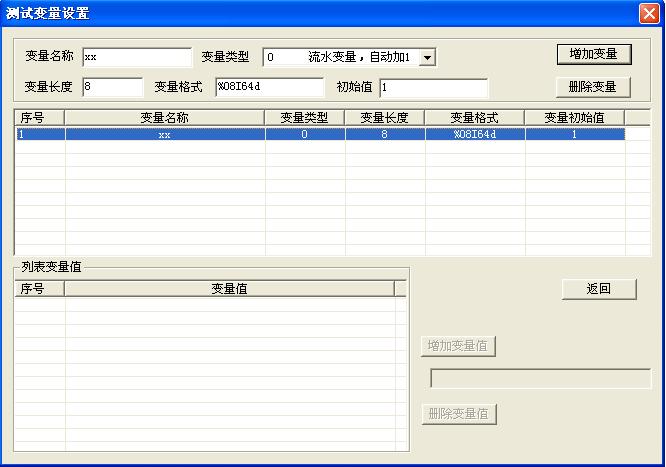
起始时间为此脚本开始运行的时间。是相对于开始运行的时间来定的，可通过此时间来设定到了某一个时间启动某些脚本的运行，逐步增加测试的压力。

#### 变量设置

可以设置工具运行的变量，当检测到脚本中含有此变量后，自动使用变量的值替换掉脚本里的变量组成报文。脚本里的变量格式为<xx>，其中xx为变量名，注意<xx>的长度不能超过脚本里定义的字段长度，否则可能导致报文数据不符。



##### 流水变量



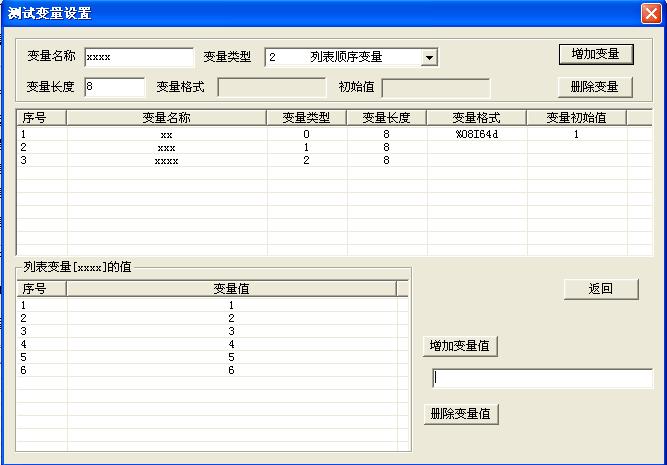
如上图，流水变量即为自动增加的变量，每运行一次自动加1。流水变量无需增加变量值。

##### 列表随机变量



如上图，列表随机变量需要增加变量值。如上所示增加了11111、333333、444444三个值，在运行时，随机在这三个值中取出来替换脚本里的变量。

##### 列表顺序变量



如上图所示，列表顺序变量和列表随机变量类似，不同的是取值时是按顺序来取的。

#### 保存打开

##### 保存结果

保存运行状态里的测试结果。

##### 打开结果

打开保存的运行结果并显示到运行状态。

##### 保存列表

保存增加的脚本列表及参数，不用每次都新增，当脚本数很多的情况下比较有用。

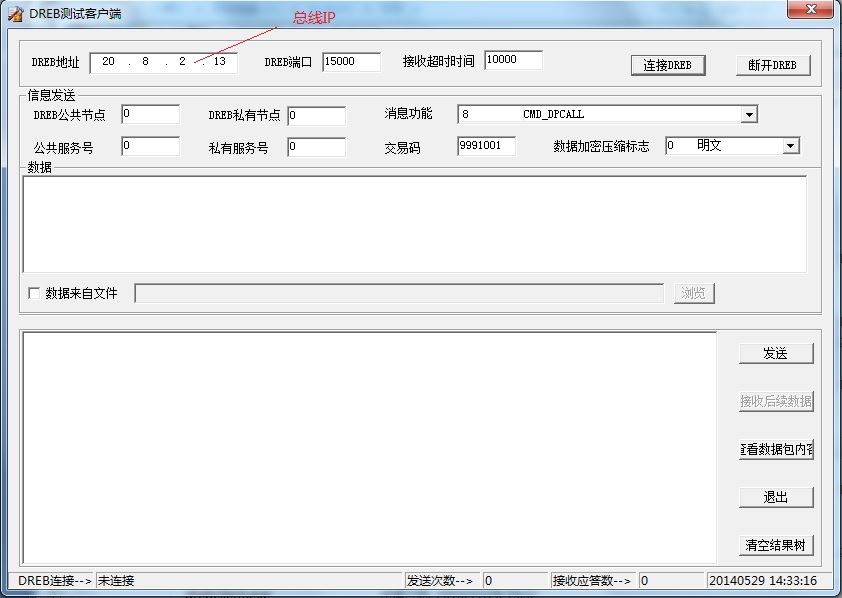
##### 打开列表

打开保存的脚本列表至列表框，然后就可以点击运行测试了。

### 总线客户端drebclient

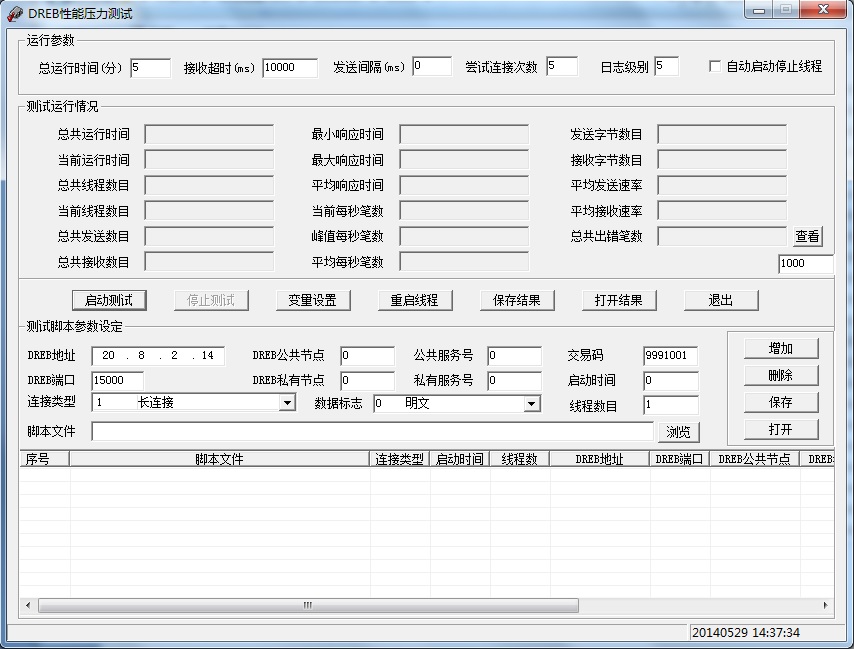
此工具是使用客户端连接至总线，发送报文给总线、BPC或后台业务。

单元测试可用此工具来做测试。



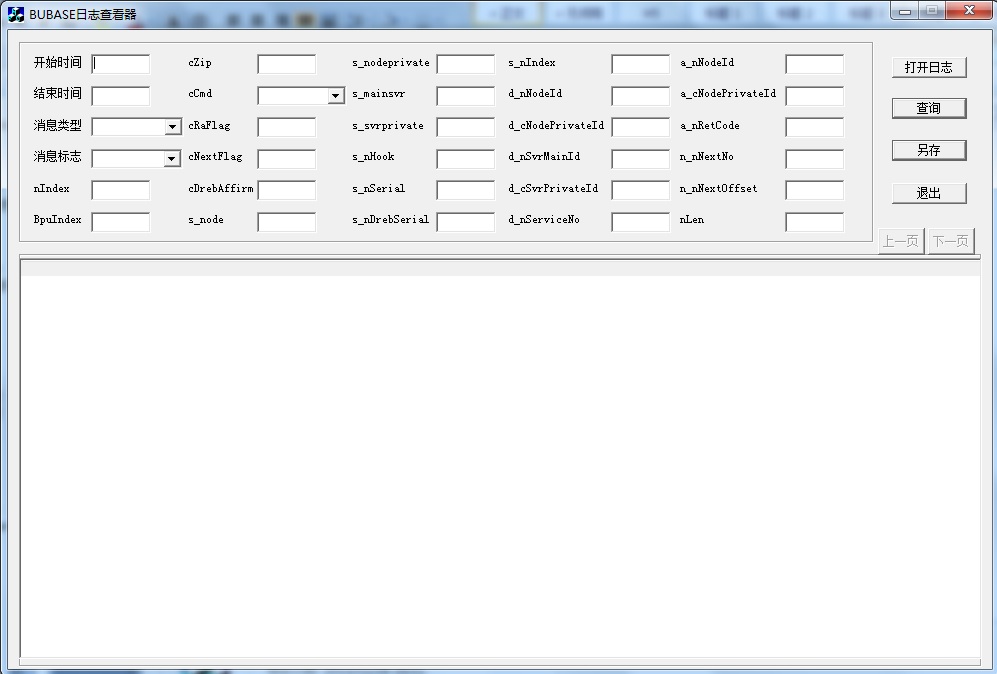
### 总线压力测试工具drebpmt

此工具主要用来作压力测试，同样是使用客户端方式连接到总线，使用方法和前置压力测试工具testfront类似。



### 日志查看工具logview

此工具用来查看cgate，cgateapi，BPC，dreb的报文日志。



# C/C++版的BPU开发

# JAVA版的BPU开发