6.0后台框架(bg\_public)设计文档及开发手册

v2.1.1

更 改 记 录

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 更改  序号 | 更 改 内 容 | 更改  页码 | 更改前版本号 | 更改后版本号 | 更改人 | 生效日期 | 备 注 |
| 1 | 首版本 |  | 1.0 | 1.0 | 王智 | 15/12/01 |  |
| 2 | 云化版本，数据库访问层使用了ibear，重新调整了整体架构，数据分发采用数据池的方式 |  | 1.0 | 2.0 | 王智 | 16/10/01 |  |
| 3 | 降低框架直接模块的耦合度：  将日志管理、配置文件管理、文件管理模块拆分为动态库；  重新调整了框架结构 |  | 2.0 | 2.1 | 王智 | 17/06/21 |  |
| 4 | 修改sdfs处理bug；  修改代码结构 |  | 2.1 | 2.1.1 | 王智 | 18/08/01 |  |
| 5 | 新增当配置文件中db\_num为空时，直接使用DSCI/cfg/db.cfg中的标签编号作为数据库编号 | 3.1.1 | 2.1.1 | 2.1.2 | 王智 | 18/09/13 |  |
| 6 | 新增对ibear自定义路由的支持;  新增对ilog日志级别失效字段的支持； | 3.4.6  3.4.7  3.1.1 | 2.1.2 | 2.1.3 | 王智 | 18/09/18 |  |

目录

[1. 简介 5](#_Toc525047674)

[1.1. 驱动力 5](#_Toc525047675)

[1.2. 软件介绍 5](#_Toc525047676)

[1.3. 软件优势 5](#_Toc525047677)

[1.4. 第三方工具说明 5](#_Toc525047678)

[1.5. 目前使用情况 5](#_Toc525047679)

[2. 设计图 6](#_Toc525047680)

[2.1. 程序结构设计 6](#_Toc525047681)

[2.2. 类图设计 7](#_Toc525047682)

[2.3. 流程图设计 8](#_Toc525047683)

[3. 工具类管理 10](#_Toc525047684)

[3.1. 配置文件 10](#_Toc525047685)

[3.1.1. server.conf 10](#_Toc525047686)

[3.1.2. sqlMapConfig.xml 11](#_Toc525047687)

[3.1.3. db.xml 11](#_Toc525047688)

[3.2. 开发规则 12](#_Toc525047689)

[3.4.1. 实时接口开发 12](#_Toc525047690)

[3.4.2. 定时接口开发 12](#_Toc525047691)

[3.4.3. 自定义接口开发 13](#_Toc525047692)

[3.3. 函数 14](#_Toc525047693)

[3.3.1. CProServerManager::Init 14](#_Toc525047694)

[3.3.2. CProServerManager::Start 14](#_Toc525047695)

[3.3.3. CProServerManager::GetServerSize 14](#_Toc525047696)

[3.3.4. CProServerManager::Close 14](#_Toc525047697)

[3.3.5. Mgr::init 15](#_Toc525047698)

[3.3.6. Mgr::process 15](#_Toc525047699)

[3.3.7. Mgr::deal\_process 15](#_Toc525047700)

[3.3.8. Mgr::ProInterface 16](#_Toc525047701)

[3.3.9. Mgr::UpdPoorStatus 16](#_Toc525047702)

[3.3.10. Mgr::ProTimeList 16](#_Toc525047703)

[3.3.11. Mgr::setTask 16](#_Toc525047704)

[3.3.12. Data::init 17](#_Toc525047705)

[3.3.13. Data::process 17](#_Toc525047706)

[3.3.14. Data::fprocess 17](#_Toc525047707)

[3.4. 代码 19](#_Toc525047708)

[3.4.1. XXX.cpp 19](#_Toc525047709)

[3.4.2. XXXMgr.h： 19](#_Toc525047710)

[3.4.3. XXXMgr.cpp： 20](#_Toc525047711)

[3.4.4. XXXData.h 20](#_Toc525047712)

[3.4.5. XXXData.cpp 20](#_Toc525047713)

[3.4.6. myibear.h 21](#_Toc525047714)

[3.4.7. myibear.cpp 21](#_Toc525047715)

[4. bg\_public环境搭建及DEMO 21](#_Toc525047716)

[6.1 环境搭建 21](#_Toc525047717)

[6.2 demo使用： 22](#_Toc525047718)

# 简介

## 驱动力

电信行业业务较为庞大，后台批量处理的程序较多，而c++中却缺少一个能对后台任务进行统一管理的开发框架。正是基于此考虑，设计并开发了这款支持c++后台开发的框架，目前支持hp-ux和linux版本；

## 软件介绍

该框架融合了数据库访问工具（ibear）、日志工具（ilog）、文件处理工具（ifile）、配置文件解析工具（icfg），针对c/c++后台批量处理程序进行设计，支持zk（需要framework），数据分发、多进程、多线程、云化模式等；

## 软件优势

适用于后台批量处理程序的开发（定时、实时），部署简单，开发方便，无需关注系统级的开发工作，只需要关注要实现的业务即可；

## 第三方工具说明

ibear：在数据库访问层，使用该工具完成；

ifile：在文件读写方面，使用了该工具完成；

icfg：在配置文件解析方面，使用了该工具完成；

ilog：在日志管理方面，使用了该工具完成。

framework：zk注册，使用工具；

## 目前使用情况

安徽账务云化；

黑龙江后台应用；

山西事件中心；

# 设计图

## 程序结构设计



## 类图设计



## 流程图设计



*实时流程*

**

*定时流程*

**

*自定义*

# 工具类管理

## 配置文件

### server.conf

*灰色*字体为可选内容

其中USER\_DEFINED为自定义配置标签，可根据需求增加或修改，其他标签均为系统级别标签；

[SYSTEM]

#程序名称

program\_name=CHKBDS

#连接的数据库个数,为空时自动导入${DSCI\_PATH}/cfg/db.cfg中的标签号码

db\_num=3

db\_id\_0=1 #物理A库billing\_data

db\_id\_1=2 #稽核库billing\_data

db\_id\_2=3 #内存库

#采用DSCI连接数据库,请配置${DSCI\_PATH}/cfg/db.cfg

ibear\_name=./chkbds\_sqlMapConfig.xml

[PROC\_SERVER]

process\_num=1

thread\_num=3

#执行方式：1 实时接口，2 定时接口，3 自定义接口

deal\_type=1

#是否需要数据分发： 0 不做分发，1 数据池方式，2 普通分发；

is\_give=1

#接口类型：1 文件接口，2 接口表，3 fedx接口，4 api接口

interface\_type=2

#若配置文件接口类型，需要配置以下路径

indir=./indir

bakdir=./bakdir

errdir=./erridr

breakdir=./breakdir

#对应日志服务器名称

server\_name=CHKBDS

#接口表模式数据备份方式

#备份模式（1：文件方式，2：表方式）

tab\_type=0

[LOG]

#日志服务器路径

log\_path=/bossbgapp/ibf/bg\_public/demo

log\_name=CHKBDS

#log\_thread\_num=0时,各线程共享一个日志

log\_thread\_num=3

#日志服务器包含日志文件数目

log\_file\_num=3

#日志服务器类型，1000,1001,1002,1003,1004（日志类型以","进行分割)

#1000:通用日志，1001:运行日志，1002:告警日志，1003:错误日志，1004:统计日志

log\_type0=1003

#日志是否失效（true：失效；false：未失效）为空默认false

log\_invalid0=false

#日志处理模式（1：同步方式，2：异步方式）

log\_procmode0=2

#日志备份类型（1：日期，2：文件大小，3：时间间隔）

log\_backup\_type0=1

log\_type1=1001

log\_invalid1=true

log\_procmode1=2

log\_backup\_type1=1

log\_type2=1000

log\_procmode2=2

log\_backup\_type2=1

***#自定义配置,请将需要新增的配置信息放入以下标签中***

[USER\_DEFINED]

***#1.计费；2.账务；3.透传***

***business\_type=1***

### sqlMapConfig.xml

ibear配置文件，详见ibear使用手册；

### db.xml

ibear配置文件，详见ibear使用手册；

## 开发规则

### 实时接口开发

**#执行方式为：1 实时接口**

**#接口类型：1 文件接口，2 接口表，3 fedx接口，4 api接口**

* interface\_type=1，程序循环处理indir下的文件；
* interface\_type=2，程序循环调用：Mgr::**ProInterface**(vector<stDealTask\*> \*values)以获取数据；
* interface\_type=3，程序循环处理indir下的文件；
* interface\_type=4，暂未开发；

**#是否需要数据分发： 0 不做分发，1 数据池方式，2 普通分发；**

* is\_give=0，程序执行Mgr::**deal\_process**(task)函数；XXXData类不处理业务；
* is\_give=1，**UpdPoorStatus**函数将入池数据打标示；业务线程XXXData中负责处理业务数据，表接口**process**、文件接口**fprocess**
* is\_give=2，业务线程XXXData中负责处理业务数据，表接口**process**、文件接口**fprocess**；

### 定时接口开发

**#执行方式：1 实时接口，2 定时接口，3 自定义接口**

**#接口类型：1 文件接口，2 接口表，3 fedx接口，4 api接口**

* interface\_type=1，定时无文件接口；
* interface\_type=2，执行XXXMgr::**ProTimeList**(char \*sql\_key)函数获取数据；**setTask**(vector<string> v\_row,stDealTask \*pTask)将查询数据v\_row赋值到pTask中；
* interface\_type=3，定时无文件接口；
* interface\_type=4暂未开发；

**#是否需要数据分发： 0 不做分发，1 数据池方式，2 普通分发；**

* is\_give=0，程序会执行**deal\_process**(task)函数；XXXData.cpp不处理；
* is\_give=1，业务线程Data中**process**负责处理数据；
* is\_give=2，业务线程Data中**process**负责处理数据；

### 自定义接口开发

#执行方式：1 实时接口，2 定时接口，3 自定义接口

deal\_type=3

#是否需要数据分发：该配置失效；

#接口类型：该配置失效；

程序会直接执行XXXXMgr::**process**()函数，执行完毕后程序将退出；

## 函数

### CProServerManager::Init

|  |  |
| --- | --- |
| **函数原型** | **bool CProServerManager::Init (char \*fileName,string input[])** |
| **功能说明** | **系统初始化** |
| **参数说明** | **char \*fileName：程序名称；**  **string input[]：入参列表；** |
| **返回值** | **true：成功；**  **false：失败；** |
| **备注** | **CProServerManager类** |

### CProServerManager::Start

|  |  |
| --- | --- |
| **函数原型** | **bool CProServerManager::Start(map<string,string> classMap)** |
| **功能说明** | **系统启动函数** |
| **参数说明** | **map<string,string> classMap：系统注册集合；** |
| **返回值** | **true：成功；**  **false：失败；** |
| **备注** | **CProServerManager类** |

### CProServerManager::GetServerSize

|  |  |
| --- | --- |
| **函数原型** | **int CProServerManager::GetServerSize ()** |
| **功能说明** | **获取业务主线程数** |
| **参数说明** | **无** |
| **返回值** | **业务主线程数** |
| **备注** | **CProServerManager类** |

### CProServerManager::Close

|  |  |
| --- | --- |
| **函数原型** | **CProServerManager::Close ()** |
| **功能说明** | **系统关闭函数** |
| **参数说明** | **无** |
| **返回值** | **无** |
| **备注** | **CProServerManager类** |

### Mgr::init

|  |  |
| --- | --- |
| **函数原型** | **int init(ibear \*br, char \*logName, int thread\_num,map<string,icfg\_Map\*> user\_defined);** |
| **功能说明** | **主线程Mgr类初始化函数** |
| **参数说明** | **ibear \*br：主线程ibear**  **char \*logName：主线程日志名称**  **int thread\_num：日志线程号**  **map<string,icfg\_Map\*> user\_defined：自定义配置文件** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **Mgr类** |
| **实例** | **m\_LogName = string(logName);**  **m\_ThreadNum = thread\_num;**  **m\_Ibear = br;**  **map<string,icfg\_Map\*>::iterator it = user\_defined.find(string("business\_type"));**  **if (it!=user\_defined.end()){ strcpy(m\_BusinessType,it->second->chValue);**  **}**  **WriteLog(m\_LogName,LOG\_SYS\_ERROR,m\_ThreadNum,"init\n");** |

### Mgr::process

|  |  |
| --- | --- |
| **函数原型** | **int process()** |
| **功能说明** | **主线程Mgr类，其他接口类型时调用该函数** |
| **参数说明** | **无** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **任务类型：其他** |

### Mgr::deal\_process

|  |  |
| --- | --- |
| **函数原型** | **int deal\_process(stDealTask \*pTask)** |
| **功能说明** | **主线程Mgr类，不做分发时，调用该函数进行业务处理** |
| **参数说明** | **stDealTask \*pTask：单条数据** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **任务类型：定时、实时；**  **分发类型：不做分发；** |

### Mgr::ProInterface

|  |  |
| --- | --- |
| **函数原型** | int ProInterface(vector<stDealTask\*> \*values) |
| **功能说明** | **主线程Mgr类，表接口时，调用该函数取接口表数据** |
| **参数说明** | vector<stDealTask\*> \*values：数据集 |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **任务类型：实时；**  **接口类型：表接口** |

### Mgr::UpdPoorStatus

|  |  |
| --- | --- |
| **函数原型** | **int UpdPoorStatus(stDealTask \*pTask)** |
| **功能说明** | **主线程Mgr类，接口表类型，使用数据池进行分发时，调用该函数更新池数据标示** |
| **参数说明** | **vector<stDealTask\*> \*values：数据集** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **任务类型：实时；**  **接口类型：表接口；**  **分发类型：数据池；** |

### Mgr::ProTimeList

|  |  |
| --- | --- |
| **函数原型** | int ProTimeList(char \*sql\_key) |
| **功能说明** | **主线程Mgr类，定时任务，接口表类型时，使用该函数获取源数据** |
| **参数说明** | char \*sql\_key：ibear xml中的sqlid |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **任务类型：定时；**  **接口类型：表接口；** |

### Mgr::setTask

|  |  |
| --- | --- |
| **函数原型** | **int setTask(vector<string> v\_row,stDealTask \*pTask)** |
| **功能说明** | **主线程Mgr类，定时任务，接口表类型时，根据游标结果为数据结构体赋值** |
| **参数说明** | **vector<string> v\_row：**  **stDealTask \*pTask：** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **任务类型：定时；**  **接口类型：表接口；** |

### Data::init

|  |  |
| --- | --- |
| **函数原型** | **int init(ibear \*ib,char \*logName,int cur\_thread\_num, stMessage \*stmsg,map<string,icfg\_Map\*> user\_defined)** |
| **功能说明** | **业务线程Data类，初始化** |
| **参数说明** | **ibear \*ib：主线程ibear；**  **char \*logName：主线程日志名称；**  **int cur\_thread\_num：日志线程号；**  **stMessage \*stmsg：错误信息结构体；**  **map<string,icfg\_Map\*> user\_defined：自定义配置文件；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **任务类型：实时、定时；** |

### Data::process

|  |  |
| --- | --- |
| **函数原型** | **int process(stDealTask \*pTask)** |
| **功能说明** | **业务线程Data类，数据分发由业务线程处理函数** |
| **参数说明** | **stDealTask \*pTask：待处理数据；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **表接口** |

### Data::fprocess

|  |  |
| --- | --- |
| **函数原型** | **int fprocess(stDealTask \*pTask)** |
| **功能说明** | **业务线程Data类，数据分发由业务线程处理函数** |
| **参数说明** | **stDealTask \*pTask：待处理数据；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **文件接口** |

## 代码

需要有Mgr、Data类；

自定义XXXMgr类需要继承CPublicMgr、Object类；

并需要实现Mgr类。

实例如下：

### XXX.cpp

*//加载头文件*

#include "CProServerManager.h"

*//注册实现类*

*classMap.insert(pair<string,string>(string("Mgr"),string("XXXMgr")));*

*classMap.insert(pair<string,string>(string("Data"),string("XXXData")));*

*//系统初始化*

*!App\_ProServerManager::Instance()->Init (fileName,inputs));*

*//系统启动*

*App\_ProServerManager::Instance()->Start(classMap));*

*//系统关闭*

*App\_ProServerManager::Close();*

### XXXMgr.h：

*#include "CPublicMgr.h"*

*#include "DynBase.h"*

*class XXXMgr:public CPublicMgr,public Object*

*{*

*DECLARE\_CLASS(XXXMgr)*

*public:*

*XXXMgr();*

*~ XXXMgr();*

*int init(ibear \*br,char \*logName,int thread\_num, map<string,icfg\_Map\*> user\_defined);*

*int process();*

*int deal\_process(stDealTask \*pTask);*

*int ProInterface(vector<stDealTask\*> \*values);*

*int ProTimeList(char \*sql\_key);*

*int setTask(vector<string> v\_row,stDealTask \*pTask);*

*int UpdPoorStatus(stDealTask \*pTask);*

*}*

### XXXMgr.cpp：

#include "XXXMgr.h"

#include "CDealData.h"

*IMPLEMENT\_CLASS(XXXMgr)*

### XXXData.h

#include "CPublicData.h"

#include "DynBase.h"

using namespace std;

class Check\_BDSData:public CPublicData,public Object

{

DECLARE\_CLASS(Check\_BDSData)

public:

Check\_BDSData();

~Check\_BDSData();

int init(ibear \*br,char \*logName,int cur\_thread\_num, stMessage \*stmsg,map<string,icfg\_Map\*> user\_defined);

int process(stDealTask \*pTask);

int fprocess(stDealTask \*pTask);

public:

ibear \*m\_Ibear;

stMessage \*m\_stMsg;

string m\_LogName;

int m\_CurThreadNum;

char m\_BusinessType[2+1];

};

### XXXData.cpp

#include "XXXData.h"

#include "DynBase.h"

IMPLEMENT\_CLASS(XXXData)

### myibear.h

#include "DynBase.h"

#include "ibear.h"

class myibear : public ibear,public Object

{

DECLARE\_CLASS(myibear)

public:

myibear();

~myibear();

int getRouteDef(void \*rules,int \*db\_no);

};

### myibear.cpp

#include "myibear.h"

IMPLEMENT\_CLASS(myibear)

myibear::myibear(){}

myibear::~myibear(){}

int myibear::getRouteDef(void \*rules,int \*db\_no)

{

\*db\_no = 1;

printf("子类\n");

return 0;

}

# bg\_public环境搭建及DEMO

### 环境搭建

搭建好ibear、ifile、ilog、icfg后；

进入bg\_public/src目录下执行sh make\_all即可；

若需要使用zk，请添加“\_ZK\_CLIENT\_”宏；

若需要sdfs的配置文件，请添加“\_SDFS\_CFG\_”宏

## demo使用：

进入demo目录执行make编译；

如需要sdfs，请修改makefile；