



## Teaf(isgw) 框架设计和使用说明书

拟制:	<u>awayfang</u>	日期:	<u>2008-6-4</u>
审核:	<u></u>	日期:	<u></u>
项目名称:	<u>Teaf(isgw) 框架</u>	项目经理	<u></u>

深圳腾讯计算机系统有限公司

版权所有 不得复制

## 修订记录

版本号	发布日期	编制人	审核人/批准人	修改章节号
V1. 0D100		Awayfang		
...		sandypan		
...		kylemao		
V1. 4D100	2008-7-21	awayfang		增加epoll模式支持
V1. 5D100	2008-8-15	awayfang		增加svr框架图说明
V1. 5D103	2008-8-18	awayfang		优化框架和业务的接口
V1. 5D104	2008-8-26	Awayfang		增加特殊协议支持
V1. 5D105	2008-9-17	awayfang		增加网络操作(同步)接口
V1. 6D100	2009-3-31	awayfang		新增oracle, sqlsvr等数据库连接支持, 开放ip权限控制
	2009-4-8	awayfang		增加FAQ章节及相关个说明
	2009-5-21	awayfang		调整ace编译说明和配置文件说明
V2. 0	2010-1-29	awayfang		解决了ack中的ACE的notify机制不可靠的问题
V3. 2	2010-9-6	Awayfang jinglinma		1 增加stat模块及相关统计项分配 2 扩展连接管理类 3 增加异步连接管理类 4 增加ibc批量处理模块 5 支持透传特定字段 6 支持自动获取本机ip进行监听
v3. 2d200	2010-12	ianyuan		1 增加流量控制 2 和外部的接口优化为IsgwOperBase和IsgwOperBase* factory_method() {...}
v3. 2d201	2011-02	Awayfang、 ianyuan		1 支持21亿以上号码 2 优化连接管理

				3 动态加载配置抽象接口 4 ACK 定时器时间可配置
v3. 2d202	2011-09			1 支持消息单纯的路由转发功能(可配置) 2 解决部分bug, 去掉多余的工作线程锁等
v3. 2d203	2012-05	Away Ian taochen		1 支持tlinux下的编译 2 数据库连接管理支持存储过程 3 安全退出功能支持
v3. 2d204	2012-09	Away Ian taochen		1 ibc 模块支持int IBCOperBase::end(IBCValue&rvalue) 回调函数 2 SysConf 支持加载多个配置文件 3 统计模块初始化提前到工作线程池(ISGWMgrSvc)初始化之前 防止出现在ISGWMgrSvc初始化中使用到连接管理类(PlatConnMgrEx), 然后该类默认初始化统计文件为".stat"的情况.
v3. 2d205	2012-09			1 去掉ace5.4及更早版本的makefile支持 只支持5.6之后的ace版本 2 增加 comm/isgw_task_base.h 的IsgwTaskBase 类 支持独立线程组
v3. 2d206	2012-10			1 优化udp协议支持, 解决在大并发量情况下可能出现的句柄问题 2 优化代码, 去掉部分snprintf前的多余memset操作

1	框架介绍 .....	7
1.1	简介 .....	7
1.2	逻辑结构图 .....	7
2	接口说明 .....	8
2.1	功能简述 .....	8
2.2	特性优化历史 .....	8
2.3	轻量级svr框架包 .....	8
2.3.1	ACE基础库 .....	9
2.3.2	Easyace基础库 .....	10
2.3.3	轻量级svr源码包 .....	11
2.4	协议及特性说明 .....	12
2.4.1	字符串协议 .....	12
2.4.2	带长度的字符串协议 .....	14
2.4.3	框架最简化 .....	14
2.4.4	通用返回值说明 .....	14
2.4.5	内部默认值说明 .....	15
3	轻量级svr开发指导 .....	15
3.1	头文件 .....	15
3.1.1	isgw_comm.h .....	15
3.1.2	isgw_config.h .....	16
3.1.3	comm .....	16
3.2	类文件及函数说明 .....	16
3.2.1	工作线程类 .....	16
3.2.2	业务逻辑类 .....	17
3.2.3	通用库类 .....	17
3.3	设计编码指导 .....	19
3.3.1	设计原则 .....	19
3.3.2	业务流程 .....	19
3.3.3	编码指导 .....	19
3.4	样例程序 .....	19
3.5	编译链接 .....	21
3.5.1	ACE5.6之前编译连接 .....	21
3.5.2	ACE5.6之后编译连接 .....	21
3.6	命令字分配 .....	22

---

4	客户端开发指导 .....	24
4.1	接口协议 .....	24
4.2	开发编码 .....	24
5	联调测试指导 .....	24
5.1	网络访问权限申请 .....	24
5.2	轻量级svr环境搭建 .....	24
5.2.1	安装部署环境 .....	24
5.2.2	修改系统配置isgw_svr.ini .....	25
5.2.3	启动轻量级svr .....	27
5.3	客户端测试 .....	28
6	应用举例 .....	29
6.1	DNF数据接口svr .....	29
7	FAQ .....	30
7.1	框架的性能怎样？同时的并发数为多少？ .....	30
7.2	怎么查看系统异常情况？ .....	30
7.3	ACESvc enqueue msg failed? .....	31
7.4	怎么打开/关闭调式信息？ .....	31
7.5	日志说明 .....	31
7.6	常见core的问题 .....	32

## 摘要

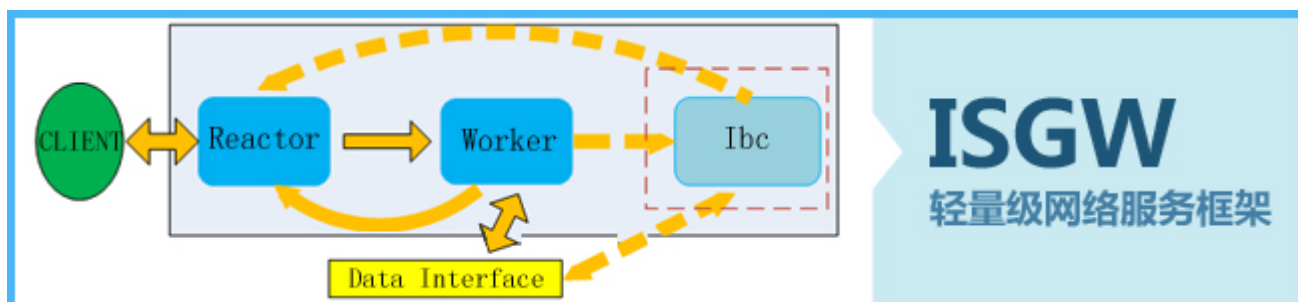
本文档主要介绍怎么使用轻量级**svr**框架并加入相应的业务逻辑完成搭建一套业务自己的轻量级**svr**。

## 关键词

轻量级**svr**框架：通用**svr**的主体程序，包括网络通信数据收发，数据库访问封装，消息流程控制主体。

## 1 框架介绍

### 1.1 简介

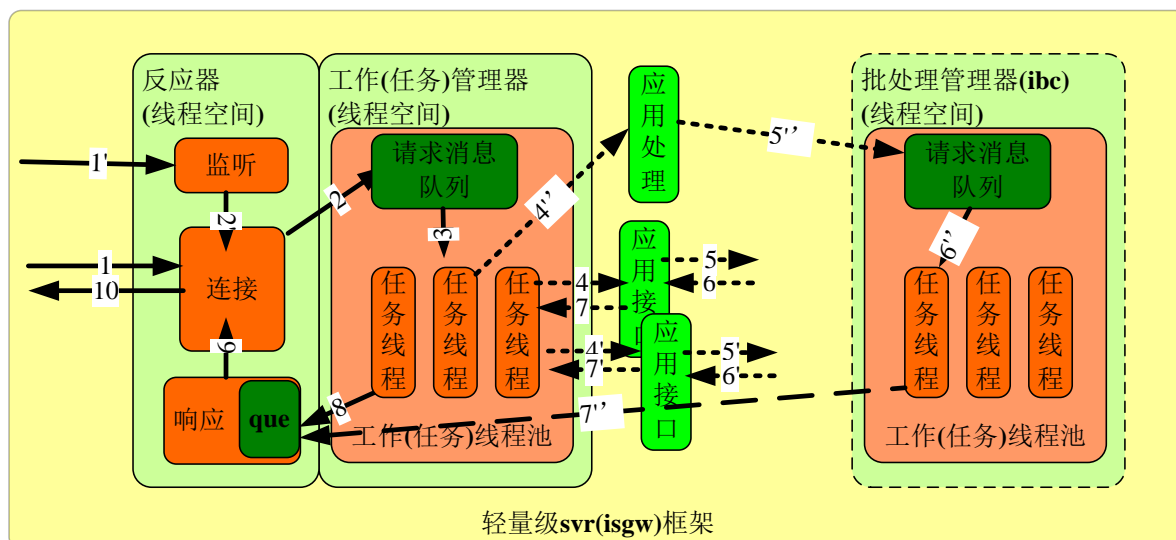


本框架主要基于ACE库构建，目的在用为互娱运营部对游戏数据操作提供一套统一的网络通信svr框架，简单的接口形式和协议，业务在此基础上能够快速通过实现业务逻辑，完成游戏数据操作服务的搭建，此框架暂命名为轻量级svr框架。

单进程多线程，支持select/epoll等模型，同时支持tcp和udp协议，支持二进制和文本格式，开源，相对多进程模型的框架来说更易维护，更轻量。业务侧只需要开发自己的逻辑处理即可实现高效的业务服务器。已经在互娱(IEG)大部分平台类产品中成熟应用，比如idip，游戏人生，心悦，帮帮，新终端游戏中心aj，cross等，现在公司很多BG都有产品在逐渐使用。

### 1.2 逻辑结构图

内部结构图如下：



反应器：主要用户接受外部连接，连接建立之后接收和发送相关的消息。

工作（任务）管理器：主要负责具体的业务逻辑处理，调用外部接口或者操作相应的数据等。

应用接口：这里应用接口包括一般的应用api或者数据库db访问接口等。

批处理管理器：专门用于并行批量查询或者操作一批用户的情况，比如查好友列表的荣誉信息。

## 2 接口说明

### 2.1 功能简述

- 1 单进程多线程模型，易于部署和扩展；
- 2 使用文本的协议，易于理解，开发成本低；新增二进制（比如pb）等协议格式的支持；
- 3 流量控制、请求监控等特性；
- 4 提供多种数据库访问封装；
- 5 提供统一的访问量（统计）数据采集；
- 6 可以支持消息路由转发；
- 7 提供批量处理特性(常用于批量的好友信息查询)；
- 8 支持业务控制是否返回消息；
- 9 支持后端同步和异步两种连接模式；

[具体详情参考发布包的readme.txt文件说明](#)

### 2.2 特性优化历史

- 1 优化了内存操作，去掉请求和响应消息的new操作，改用对象池。
- 2 字符串编码的prot\_encode函数可能存在和内部函数冲突的问题，增加名空间保护隔离。
- 3 增加支持epoll模式(支持大并发量接入)

#### V1.6D100

- 1 内部消息队列放到10M，支持更大量的并发请求。
- 2 修复了部分bug。
- 3 调整了返回消息的内容，增加了cmd信息。

#### V1.6D101

- 1 开放数据库连接类PlatDbAccess的端口配置，可以在isgw\_svr.ini文件中进行配置
- 2 修改了网络连接类PlatConnMgr的bug，对外主动断开时关闭连接

[具体详情参考发布包的readme.txt文件说明](#)

### 2.3 轻量级 svr 框架包

轻量级svr框架基于ACE版本构建，如果已经安装ACE则可以不再安装



## 2.3.1 ACE 基础库

### 2.3.1.1 说明

如果已经安装ACE则不需要再安装。

### 2.3.1.2 ACE6.0 之前安装

1 tar -jxvf ACE-5.6.tar.bz2 (如果是gz包, 参数j换为z)

2 进入ACE\_wrappers目录

3 创建一个单独的目录用于构建ace的库, 比如mkdir build, 进入build目录

4 进入build目录后, 执行../configure --enable-static --enable-threads --disable-ssl --disable-acexml --disable-ace-examples --disable-ace-tests --prefix=/data/app/ 进行配置

注意:

4.1 如果需要在ace/config.h中加入 #define ACE\_HAS\_EVENT\_POLL 1

4.2 如果使用同步的超时机制, 默认是使用select的实现, 会有句柄数的限制, 所以需要加这个编译选项改成poll的超时机制

#define ACE\_HAS\_POLL 1 #define ACE\_HAS\_LIMITED\_SELECT 1

4.3 因为我们修改了ACE的日志风格, 所以需要替换文件ace.cpp修改了这个函数的实现。

```
ACE_TCHAR * ACE::timestamp (ACE_TCHAR date_and_time[],
                             int date_and_timelen,
                             int return_pointer_to_first_digit)
```

和定义这个宏 #define ACE\_USE\_MY\_TIMESTAMP 1

4.4 下面3个定义需要关注:

```
//ACE的NOTIFY队列使用内存QUEUE机制, 而不使用传统的PIPE
#define ACE_HAS_REACTOR_NOTIFICATION_QUEUE 1
//CDR的字节对齐问题
#define ACE_CDR_IGNORE_ALIGNMENT 1
//CDR的字节对齐问题
#define ACE_LACKS_CDR_ALIGNMENT 1
```

4.5 有时候可能还需要定义这个宏 #define ACE\_HAS\_MEMCPY\_LOOP\_UNROLL 1 解决fast\_memcpy的问题

4.6 为了提高定时器的精度 可以把ACE默认的10ms 调整为 1ms #define ACE\_TIMER\_SKEW (1000 \* 1)

4.7 如果存在大量短连接接入你可能需要修改 listen 函数的 backlog 参数的值为更大的 性能应该会更好些, 宏定义为 ACE\_DEFAULT\_BACKLOG 在 Default\_Constants.h 文件中



参考样例文件: Ace.cpp Config.h

## 5 配置完成执行make命令进行编译( buildbits=32 可选 )

```
make static_libs=1 versioned_so=0 optimize=1 buildbits=32
```

## 6 编译完成执行make install进行安装，可以用 --prefix= 选项指定安装到的目录

注意：默认ace的库和头文件安装在/usr/local/lib/和/usr/local/include/ace/目录

### 2.3.1.3 ACE6.0 之后安装

解包之后的编译步骤参考如下：

1. ACE\_ROOT=/Users/away/Documents/software/ACE\_wrappers; export ACE\_ROOT

2. 修改 include/makeinclude/platform\_macros.GNU

```
include $(ACE_ROOT)/include/makeinclude/platform_macosx.GNU
```

```
INSTALL_PREFIX = /usr/local
```

3. 修改ace/config.h

```
#define ACE_HAS_EVENT_POLL
```

```
#define ACE_DEFAULT_BACKLOG 1024
```

```
#define ACE_HAS_REACTOR_NOTIFICATION_QUEUE
```

```
#define ACE_USE_MY_TIMESTAMP 1
```

```
#define ACE_TIMER_SKEW (1000 * 1)
```

4. 直接在ace目录下执行

```
make static_libs_only=1
```

5. 如果要编译其他目录 也可以在其他目录下执行 make

## 2.3.2 Easyace 基础库

### 2.3.2.1 说明

easyace\_v1.4\_release.tgz

如果已经安装则可以不安装，相关最新版本更新会由平台开发组相关同事及时通知。

### 2.3.2.2 安装

1 tar -zxvf easyace\_v1.4\_release.tgz (如果是bz2包，参数z替换为j)

2 进入 easyace 目录

3 执行 make 命令(如果使用ace5.6版本请使用makefile.ace5.6)编译好需要的库

注1: 从isgw\_v3.2d203之后的版本自带easyace的库, 不需要再单独获取, 最新版本请联系 awayfang ianyuan taochen

注2: 从isgw\_v3.3d202之后的版本去除了easyace库, 分解为comm库, ace相关的类合并到 isgw 目录下

### 2.3.3 轻量级 svr 源码包

#### 2.3.3.1 目录结构说明

库名如 isgw-v1.5d103-release.tgz (二进制发布版, 不包括源码) 或者 isgw-v1.5d103.tgz (源码包)

注: 最新版本请联系awayfang ianyuan taochen获取

解包后产生 comm frm svr sql client 五个源码目录

./comm 存放轻量级svr编译需要的公共源码, 供商城框架和业务共同使用, 请业务使用人员不要修改

./frm 存放轻量级svr的框架相关的程序源码, 请业务使用人员不要修改

./svr 存放轻量级svr的业务逻辑相关的程序源码, 业务使用开发的源码都可以放这个目录

./sql 存放轻量级svr测试用的数据库创建脚本

./client 存放客户端的测试相关程序

注: 最新的版本目录结构调整如下:

```
isgw
|-- bin
|-- cfg
|-- client
|-- comm
|-- easyace
|-- frm
|-- log
`-- svr
```

comm 独立的公共库 可以独立使用 本库会使用到其中的部分代码

三个运行相关目录

./bin 存放可执行程序和本脚本

./cfg 存放配置文件, 包括系统配置文件和业务配置文件

./log 存放日志目录, 一般建议指向/data目录下的相应目录

### 2.3.3.2 简易安装

ACE5.4, 执行如下:

```
cd isgw/svr
make
```

ACE5.6, 执行如下:

```
cd isgw/svr
make -f makefile.ace5.6 conf
make -f makefile.ace5.6
```

注: 如果make有问题, 请先执行make -f makefile.ace5.6 cleanconf  
和 make -f makefile.ace5.6 clean 清理之后再编译

## 2.4 协议及特性说明

### 2.4.1 字符串协议

本框架与客户端程序采用文本格式的通信协议(字符串),请求协议包采用cgi的参数格式, 每个字段用'&'分割, 以"\n"作为结束符, 字段的前后顺序不固定, 字段名和值用'='分割。

请求消息如: "cmd=001001&uin=88883960&area=0\n"

由于&号作为各个字段的分隔符, 不能作为内容使用, 字符串内容中不允许有多余的'&'\n'符号出现。

响应协议包也建议是文本协议, 并且格式上没有特别要求, 建议跟返回协议保持一致。

响应消息如: "cmd=001001&result=0&info=...\n"

具体注意点如下:

- (1) 协议格式: 文本字符串, 必须以 "\n" (可配置为"\r\n"等) 结尾, 以'&'作为一级分隔符, '='作为二级分隔符, 参数串长度不超过 8192 个字节(可配置)。
- (2) 字符串采用类似于 cgi 的参数形式, 如 para1=value1&para2=value2 ...; 字段名长度不超过 24 个字符, 取值长度不超过 8192 个字符;字段的先后顺序不固定, 请不要按照字段的位置来解析。
- (3) 如参数值中可能含有特殊字符(协议的分割符, 比如'&=%'等), 则用%XX (XX 为该字符的 16 进制值) 表示字符串, 和浏览器下调用 cgi 输入的参数类似。例如: sQQPasswd=&%改为用 sQQPasswd=%26 %25 表示。字符转换规则如下:
  - 1) 字符串转换规则为: %+字符的 ASCII 码的 16 进制数, 且字母必须大写, 例如: '\n' 的正确

转换值为 %0D,其他的都是错误的,比如: %13, %0d。对于中文字符占大多数的系统建议统一采用 base64 编码,对所有字符转换成 16 进制的两个字母的形式。以便节省编码后的空间。

- 2) 需要转换的字符: 与协议有关的字符,包括: ' & ', ' = ', ' | ', ' ' , ' \n ', ' \r ', ' : ' 以及所有不可见字符
- 3) 除 2) 规定的字符外,所有字符均不要转换。特别强调,下列字符无需转换: 汉字字符
- (4) 内部定义的一些协议字段有: `cmd result _sockfd _sock_seq _msg_seq` 等,应用层定协议的时候请避免使用这些字段名称
- (5) 协议制定的时候参数是列表的形式建议参数名称为 "...list", 取值的协议格式为 ' ' 作为一级分隔符(一般也用于不用用户之间数据的分割), 空格或者逗号作为二级分隔符的列表(如果一个字段不存在两级分割的问题建议尽量用空格做分割,这样将来批量查询的时候要一个字段传递多个用户的信息可以用|做分割), ~\_等可以作为三级分隔符(设计时尽量少用到三层分割以上的情况)。比如: `uin1 time1|uin2 time2|...|`

注意：协议内容的转换需要在用户拼装协议时用户自己调用相关的接口函数进行，拼装后框架无法进行转换。转换的函数有`prot_encode`(转换)和`prot_strim`(去除)，都存放在`isgw_comm.h`中。

为了保证信息解析的正常，通信协议尽量采用原始的ASCII编码方式传递即可，如果业务逻辑需要做转换，请在轻量级svr后台进行。

#### 2.4.2 带长度的字符串协议

协议体本身和2.4.1章节描述的一样，为字符串协议，但是协议包头部包含消息体的长度字节信息。具体差别如下：

协议头的前4个字节是后面内容的长度字段（不包括本身的长度），网络字节序；

发送的数据可能有多行，如 `htonl(16)a=b&c=d\ne=f&b=3\n`

#### 2.4.3 框架最简化

用户只需关心上层的业务流程，而无需关心底层的实现，使得用户使用起来最简单。

业务流程上用户可以自己选择是由api内部自动确认还是由用户确认，可以通过相关环境变量配置来实现。

#### 2.4.4 通用返回值说明

通用的接口返回值定义如下：

```
/******  
// 常用错误定义  
// 0 表示正常/成功  
// 负数(<0) 表示系统级别异常  
//      比如配置问题 网络问题 需要人工干预修复的  
//      这种异常通常是不应该存在的 需要高度关注的  
//      不然会导致系统严重不可用 或者不处理会有很大的潜在风险  
// 正数(>0) 表示业务逻辑级别异常  
//      比如记录不存在 记录重复等  
//      这种情况是可以存在的 但是应该尽量减少  
*****/  
typedef enum _PP_ERROR  
{  
    ERROR_NO_FIELD = -123456789, //协议字段不存在 业务定义的返回尽量避免和这个定义冲突  
    ERROR_TIMEOUT_SER = -10, //业务接口发现消息超时  
    ERROR_NO_PERMIT = -5, //无权限访问  
    ERROR_PARA_ILL = -4, //参数非法  
    ERROR_TIMEOUT_FRM = -3, //框架内部发现消息超时  
    ERROR_CONFIG = -2, //配置异常
```

```
ERROR_NET = -1, //网络异常
ERROR_OK = 0, //正常情况
ERROR_NOT_FOUND = 1, //业务接口返回记录不存在 或者数据库中无记录
ERROR_DUPLICATE = 2, //业务接口返回记录重复 或者数据库中记录重复

}PP_ERROR;
```

注：日志里面记录的错误码请尽量保持和此返回值一致。

### 2.4.5 内部默认值说明

为了提高整个系统的响应性能，快速恢复系统的状态，加强过载保护等，整个系统的默认配置项取值如下：

```
-----框架超时相关配置-----
# 0 超时不丢弃 1 丢弃(默认值)
discard_flag = 1
# 默认的超时时间 2 S
discard_time = 2

-----连接管理器相关配置-----
# 是否使用连接策略，不配置默认使用
use_strategy = 1
# 最大允许的连续失败次数，超过则发到这个连接上的消息自动告知发送失败，不会尝试发送，直到超过重连
间隔
max_fail_times = 3
# 重连间隔，单位 s
recon_interval = 10
# 默认连接数
conn_nums = 10
```

## 3 轻量级 svr 开发指导

### 3.1 头文件

#### 3.1.1 isgw\_comm.h

comm./isgw\_comm.h 商城框架使用的通用文件，里面包括了数据库访问，基础库使用，协议操作指令定义头文件；轻量级svr的业务开发人员的业务逻辑代码中只需要包含isgw\_comm.h头文件即可进行轻量级svr的开发

### 3.1.2 isgw\_config.h

svr/isgw\_config.h 修改此文件可以重新定义框架里面的某些宏或者变量值。可定义的变量如下

成员/宏定义	说明
MAX_INNER_MSG_LEN	定义接收缓冲区的大小，内部默认为4096字节
MSG_SEPARATOR	消息协议分割符，内部默认为“\n”
MSG_LEN_SIZE	消息头长度字节的大小，取值可以为2/4，内部没定义（即默认为不需要协议头表示消息包长度）
FIELD_NAME_CMD	协议内容的命令字字段名，默认为“cmd”

### 3.1.3 comm

comm目录下存放这连接管理（包括同步和异步）类，框架接口的基类，协议定义等头文件，开发时可以多关注。

## 3.2 类文件及函数说明

### 3.2.1 工作线程类

工作线程类为 ISGWMgrSvc （文件isgw\_mgr\_svc.h isgw\_mgr\_svc.cpp），主要控制商城客户端请求消息的流向控制，调用业务接口进行相关操作后把结果信息回送给客户端。此类为主动对象类，用户可以不关注实现原理。

开发时只需要把业务逻辑的头文件加入 isgw\_mgr\_svc.cpp 文件头部，然后再在下面函数中加入相应的代码即可。

开发需要关注的函数

方法	说明
OUT PriProAck* ISGWMgrSvc::process(IN PriProReq*& pp_req)	此回调函数根据请求的数据包PriProReq *& pp_req获取到用户请求的信息内容，传递给业务处理逻辑函数，业务处理逻辑函数处理完成之后返回消息存放到PriProAck的数据结构中



### 3.2.2 业务逻辑类

业务逻辑类的代码由业务自己进行开发，下面是一下业务需要关注的函数说明

方法	说明
int process(QModeMsg& req, char* ack, int& ack_len);	此函数形式是建议业务使用的通用形式，在此函数中。 <pre> /** @name process  * @param req 入参，请求消息  * @param ack 出参，存放返回的响应消息  * @param ack_len 入参兼出参，传入为ack指向的最大空间，  *                                     传出为ack实际存放的消息长度  * @retval 0 成功  * @retval 其他值表示不 成功  */ </pre>
IsgwOperBase	业务需要继承此类，具体可以参考svr/ pdb_oper.h文件
IsgwOperBase* factory_method()	业务需要实现此类，具体可以参考svr/ pdb_oper.h文件

### 3.2.3 通用库类

通用库主要包括数据库访问接口类，网络通信操作接口，配置文件操作类（具体请看easyace的说明文档），协议编解码操作接口等。

#### 3.2.3.1 配置文件类/函数

方法	说明
int SysConf::get_conf_str(const char* section, const char* name, char* value, int buf_len)	从配置文件的段section和键name获取字符串类型的值存放到value中
int SysConf::get_conf_int(const char* section, const char* name, int* value)	从配置文件的段section和键name获取整型的值存放到value中

#### 3.2.3.2 协议处理类/函数

方法	说明
char *EASY_UTIL::prot_encode(char *dest, char *src)	编码特殊字符，特殊字符包括"&= % :"
char *EASY_UTIL::prot_strim(char *dest, char *src)	过滤特殊字符，特殊字符包括"&= % :"

### 3.2.3.3 数据库操作类/函数

通用库主要包括数据库访问接口类，配置文件操作类（具体请看easyace的说明文档）

**Mysql 数据库访问类 PlatDbAccess**

方法	说明
int init(char *host, char *user, char *passwd)	通过host, user, passwd初始化数据库连接
int exec_query(const char* sql, MYSQL_RES*& result_set)	执行返回结果集的sql语句函数，比如select等
int exec_update(const char* sql, int& last_insert_id, int& affected_rows)	执行不返回结果集的sql语句函数，比如update,insert等
char *get_err_msg(void)	返回数据库访问的错误信息

### 3.2.3.4 网络操作类/函数

主要用于业务应用模块操作其他的网络服务器，要和其他服务器进行网络通信，提供了PlatConnMgr类(plat\_conn\_mgr.h文件)和PlatConnMgrAsy 类(plat\_conn\_mgr\_asy.h文件)

方法	说明
int init(const char *section)	通过配置文件配置的ip,端口进行连接初始化
int init_conn(int index, const char *ip, int port)	通过ip port初始化网络连接，index为连接的索引号
int send(const void * buf, int len)	发送指定长度的数据，失败返回-1，成功返回发送成功的长度，默认超时时间为2秒

<code>int recv(void * buf, int len)</code>	接收指定长度的数据，失败返回-1，成功返回接收成功的长度，默认超时时间为2秒
<code>int send_recv_ex (const void *send_buf, int send_len, void *recv_buf                   , int recv_len, const char* separator=NULL, const unsigned int uin=0);</code>	发送数据，并等待响应信息，默认超时为框架统一设置

### 3.3 设计编码指导

#### 3.3.1 设计原则

#### 3.3.2 业务流程

#### 3.3.3 编码指导

### 3.4 样例程序

```
//=====
==
/**
 * @file    pdb_oper.h
 *
 * 此文件为pdb内部保留的业务应用接口定义文件，可以作为开发样例参考
 *
 * @author awayfang
 */
//=====
==
#ifndef _PDB_OPER_
```

```
#define _PDB_OPER_
#include "isgw_comm.h"
#include "pdb_prot.h"
#include "isgw_oper_base.h"

#include "temp_proxy.h"
#include "vip_db_oper.h"

class PdbOper : public IsgwOperBase
{
public:
    PdbOper();
    ~PdbOper();

    /** @name process
    *
    *
    * @param req 入参，请求消息
    * @param ack 出参，存放返回的响应消息
    * @param ack_len 入参兼出参，传入为ack指向的最大空间，
    *                                     传出为ack实际存放的消息长度
    *
    * @retval 0 成功
    * @retval 其他值表示不 成功
    */
    int process(QModeMsg& req, char* ack, int& ack_len);

private:

private:

};

IsgwOperBase* factory_method()
{
    TempProxy::init();
```

```

    IsgwOperBase* obj = new PdbOper();
    return obj;
}

```

```
#endif //_PDB_OPER_
```

业务逻辑程序具体请参考[isgw\svr\pdb\\_oper.h](#)和[pdb\\_oper.cpp](#)

### 3.5 编译链接

#### 3.5.1 ACE5.6 之前编译连接

1 修改svr目录下的makefile的FILES部分，增加业务新增的代码文件名（只输入.cpp文件的前缀名称即可）

```

FILES    = isgw_svr \
            isgw_app \
            isgw_intf \
            ace_sock_hdr_base \
            isgw_ack \
            qmode_msg \
            plat_db_access \
            plat_db_conn \
            isgw_mgr_svc \
            pdb_oper

```

2 如果需要使用特殊库和头文件请修改以下行增加支持。

```

CCFLAGS += -I../comm -I. -I../ace_using/easyace -I/usr/local/mysql/include
LDFLAGS = -static -L../comm -L. -L../ace_using/easyace -L/usr/lib/mysql/
LDLIBS = -leasyace -lmysqlclient_r -lz

```

#### 3 执行命令

```

cd isgw/svr
make

```

#### 3.5.2 ACE5.6 之后编译连接

1 如果需要使用特殊库和头文件请修改以下行增加支持。

```

CXX=g++
CFLAGS=

```

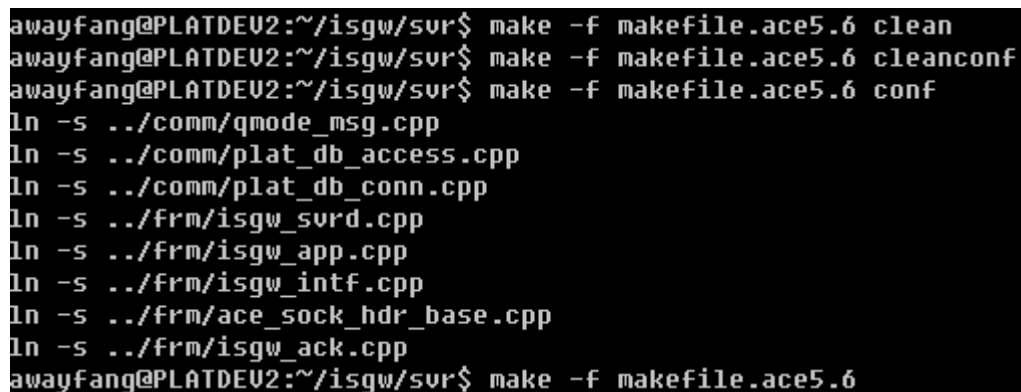
```
INCLUDE=-I.././easyace/ -I/usr/local/mysql/include/mysql/ -I../comm/ -I../frm  
LIBS=-L.././easyace/ -L/usr/local/mysql/lib/mysql/ -L../comm/ -L../frm -lACE  
-leasyace -lmysqlclient_r
```

注：如果想要编译出来的进程名为用户自定义的名称请修改makefile文件中TARGET的取值，  
比如，TARGET = pnh\_svrd

## 2 执行命令

```
cd isgw/svr  
make -f makefile.ace5.6 conf  
make -f makefile.ace5.6
```

操作截图如下：



```
awayfang@PLATDEV2:~/isgw/svr$ make -f makefile.ace5.6 clean  
awayfang@PLATDEV2:~/isgw/svr$ make -f makefile.ace5.6 cleanconf  
awayfang@PLATDEV2:~/isgw/svr$ make -f makefile.ace5.6 conf  
ln -s ../comm/qmode_msg.cpp  
ln -s ../comm/plat_db_access.cpp  
ln -s ../comm/plat_db_conn.cpp  
ln -s ../frm/isgw_svrd.cpp  
ln -s ../frm/isgw_app.cpp  
ln -s ../frm/isgw_intf.cpp  
ln -s ../frm/ace_sock_hdr_base.cpp  
ln -s ../frm/isgw_ack.cpp  
awayfang@PLATDEV2:~/isgw/svr$ make -f makefile.ace5.6
```

## 3.6 命令字分配

命令字（即协议中的cmd字段）的取值由轻量级svr框架开发维护人员来分配，对于每个业务分配一段id（1000个），目前分配如下：

PDB内部保留：000000-000999

0-99 系统管理用的命令

100-199 支付用

200-209 BOSS系统

210-999 游戏人生

DNF: 001000-001999

FXD: 002000-002999 飞行岛

WEBGAME: 3000-3999 丝路英雄

XW: 4000-4999 炫舞

FC: 5000-5999 飞车

XX: 6000-6999 QQ寻仙  
FO: 7000-7999  
FFO: 8000-8999  
SG: 9000-9999  
WEBDEV: 10000-10999 web开发组  
QQTANG: 11000-11999  
CF: 12000-12999  
R2: 13000-13999  
QQGAME: 14000-14999  
PM: 15000-15999  
AVA: 16000-16999  
DMLQ: 17000-17999  
QQXY: 18000-18999  
HXSJ(DWC): 19000-19999  
OTOT: 20000-20999  
XXZ: 21000-21999  
YKSK(烽火战国): 22000-22999  
PETWORLD: 23000-23999  
CMD\_XY(轩辕):24000-24999  
CMD\_LOL(英雄联盟):25000-25999  
CMD\_PET(宠物): 26000-26999  
CMD\_SGYX(三国英雄): 27000-27999  
CMD\_ZT(征途): 28000-28999  
CMD\_NBA(NBA): 29000-29999  
CMD\_IGAME(游戏人生): 30000-30999  
CMD\_YLZT(御龙在天): 31000-31999  
CMD\_QQBABY(QQ宝贝): 32000-32999  
CMD\_NX(QQ飞行战纪): 33000 – 33999  
CMD\_LOL : 34000 – 34999  
CMD\_ANGEL(洛克): 35000-35999  
CMD\_QXZB(七雄争霸): 36000-36999  
CMD\_MHDL(魔幻大陆): 37000-37999

## 4 客户端开发指导

### 4.1 接口协议

从2.3.1章节可以知道客户端和轻量级svr采用业务自定义的文本协议。用户可以按照规范和约定的形式进行定义。

### 4.2 开发编码

按照普通的网络编程，发送对应的文本消息包，并接收响应消息解析出结果即可。

## 5 联调测试指导

### 5.1 网络访问权限申请

根据需要申请客户端和轻量级svr的ip和端口访问权限。

### 5.2 轻量级 svr 环境搭建

#### 5.2.1 安装部署环境

建议目录结构如下：

bin

isgw\_svr 主程序，可执行文件

isgw\_svr.pid 临时产生，用于存放本进程id

keeper.sh 监控脚本，用于监控并自动拉起指定的进程

start.sh 启动脚本

cfg

isgw\_svr.ini 配置文件

log

存放日志，跟进配置信息，默认为滚动日志

```
bin/
|-- isgw_svr -> ../svr/isgw_svr
|-- isgw_svr.pid
|-- keeper.sh -> ../svr/keeper.sh
`-- start.sh -> ../svr/start.sh
cfg/
`-- isgw_svr.ini -> ../svr/isgw_svr.ini
log/
|-- isgw_svr.log
```



## 5.2.2 修改系统配置 isgw\_svr.ini

```
#####
##      每行后面带有#注释的地方，请在发布的时候去掉，避免影响      ##
#####

[common]
# 滚动日志相关选项 -m 日志文件最大大小(K) -N 日志文件数目 -f 日志文件模式
# -p日志输出级别，~取消输出，有DEBUG TRACE INFO等各种级别，-s日志文件路径
log_mask = "-m 102400 -N 10 -f OSTREAM -p~DEBUG|~TRACE|~NOTICE
-s ../log/isgw_svr.log"

[system]
# 0 表示 tcp 协议无效，其他值有效，默认配置有效
tcp_flag = 1
#ip = 172.25.40.94
port = 5693
#0 不打开限制开关 1 打开限制开关
allow_flag = 0
allow_ip = 172.25.40.94;

# 1 表示udp协议有效，其他值无效，默认配置无效
udp_flag = 0
#udp_ip = 172.25.40.94
udp_port = 5693
# 0 不打开限制开关 1 打开限制开关
udp_allow_flag = 0
udp_allow_ip = 172.25.40.94;

# 是否使用 ibc 批量处理功能模块；1 表示使用；0 表示不使用；默认配置不使用
ibc_svc_flag = 0

# isgw ack 模块处理的间隔 单位 微秒
ack_interval = 0

# 需要根据实际情况调整
[isgw_mgr_svc]
threads = 50
# 单位字节
quesize = 10485760
# 加载的 dll 的名称
dllname = oper
```

```
# 0 超时不丢弃 1 超时丢弃(默认值)
discard_flag = 1
# 默认的超时时间 2 S
discard_time = 2

# 需要根据实际情况调整 这个线程数一般比 isgw_mgr_svc 线程数大
[ibc_mgr_svc]
threads = 100
max_ibcr_record = 1000
quesize = 104857600
discard_time = 3

# 流量控制
[cmd_amnt_cntrl]
control_flag = 0
start_cmd = 950
end_cmd = 990
interval = 30
max_req = 100000
max_fail_ratio = 100
interval_959 = 30
max_req_959 = 1
max_fail_ratio_959 = 30

# 路由配置 注意取值: 0 默认值, 不路由, 只本身处理 1 只路由由本身不处理消息 2 既路由又处理消息
[router]
route_flag = 0
ip_0 = 172.25.40.94
port_0 = 5993
conn_num_0 = 3

[isgw_cintf]
# 消息队列大小 单位字节
quesize = 10485760

[identify]
user =
passwd =

# VIP 信息查询 配置
[vip_db_1]
ip = 172.23.16.100
user =
```

```

passwd =
db_name = dbVip
# 是否使用连接策略, 不配置默认不使用
use_strategy = 1
# 最大允许的连续失败次数, 超过则发到这个连接上的消息自动告知发送失败, 不会尝试发送, 直到超过重
连间隔
max_fail_times = 3
# 重连间隔, 单位 s
recon_interval = 600

```

```

#####
##          下面是DNF连接配置                      ##
#####
#####
# G1 广东一区
[dnf_world_1_game_db1]
ip = 172.16.176.97
user =
passwd =
db_dnf_china = dnf_china_gd1
...

```

**common** 公共配置段

**log\_mask** 滚动日志相关选项 **-m** 日志文件最大大小(K) **-N** 日志文件数目 **-f** 日志文件模式  
**-p** 日志输出级别, ~取消输出, 有DEBUG TRACE INFO等各种级别, **-s** 日志文件路径

**system** 系统相关配置段

**allow\_ip** 允许访问的ip列表, 可以";,"分割多个ip(暂时关闭, 如果打开需要修改isgw\_intf类)

**ip** 服务监听的ip地址

**port** 服务监听的端口

**isgw\_mgr\_svc** 工作线程配置段

**threads** 线程数

其他为用户的业务配置相关信息

### 5.2.3 启动轻量级 svr

启动脚本start.sh说明:

```
if [ $# -lt 1 ]
```

```

then
{
    echo "Usage:$0 <svrd_name>"
    exit 1
}
fi

SVRD_NAME=$1
export ISGW_HOME=/usr/local/isgw/
export ISGW_BIN=${ISGW_HOME}/bin/
export ISGW_CFG=${ISGW_HOME}/cfg/

ulimit -c unlimited
ulimit -n 20480 #用epoll模式时需要放大此参数
${ISGW_BIN}/${SVRD_NAME} 1>>${ISGW_BIN}/start.log 2>&1 &

```

ISGW\_HOME指定svr程序的home目录

ISGW\_BIN 指定程序运行的路径

ISGW\_CFG 配置文件路径

注：如果框架不调整环境变量本身的名称不能修改，取值可以根据部署情况修改

```

cd bin
./start.sh isgw_svr

```

### 5.3 客户端测试

如果已经开发完成客户端程序请使用开发的客户端程序测试，client目录下存放有测试样例，编译完成后使用如下：

```
Usage: ./client <ip> <port> <content> [qtimes] [interval]
```

<ip> 轻量级svr的ip

<port> 轻量级svr的端口

<content> 发送的内容，如果有特殊字符请使用""引起来

[qtimes] 查询次数，可选

[interval] 查询间隔，单位毫秒，可选

```

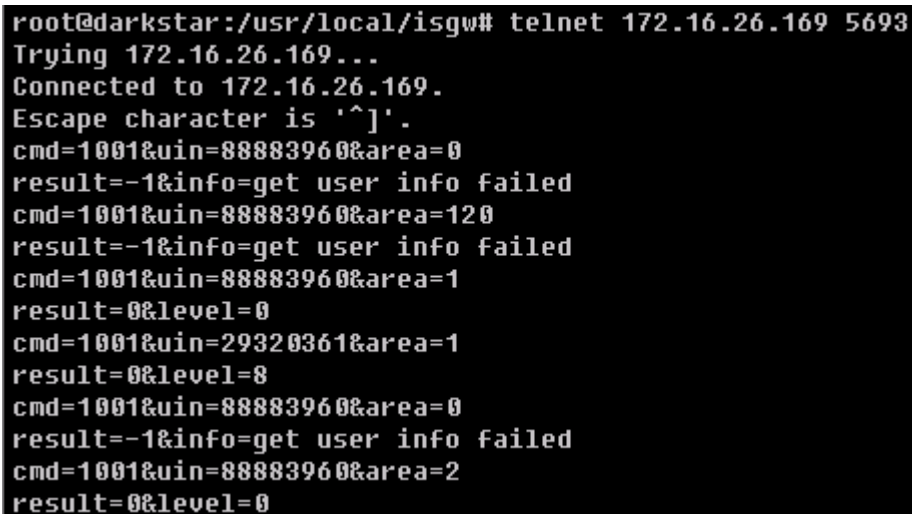
root@darkstar:/usr/local/isgw/client# ./client 172.16.225.110 5693 "cmd=1001&area=1&uin=88888888"
[Fri Jun 13 2008 12:28:53.591231] send,len=30,msg is cmd=1001&area=1&uin=88888888
[Fri Jun 13 2008 12:28:53.810366] recv_len=36, result=2&info=get user info failed

```

如果没有开发完成客户端程序可以使用telnet程序测试，比如：

轻量级svr服务器ip和端口为 172.16.26.169 5693

```
root@darkstar:/usr/local/isgw# telnet 172.16.26.169 5693
Trying 172.16.26.169...
Connected to 172.16.26.169.
Escape character is '^]'.
cmd=1001&uin=88883960&area=0
result=-1&info=get user info failed
cmd=1001&uin=88883960&area=120
result=-1&info=get user info failed
cmd=1001&uin=88883960&area=1
result=0&level=0
cmd=1001&uin=29320361&area=1
result=0&level=8
cmd=1001&uin=88883960&area=0
result=-1&info=get user info failed
cmd=1001&uin=88883960&area=2
result=0&level=0
```

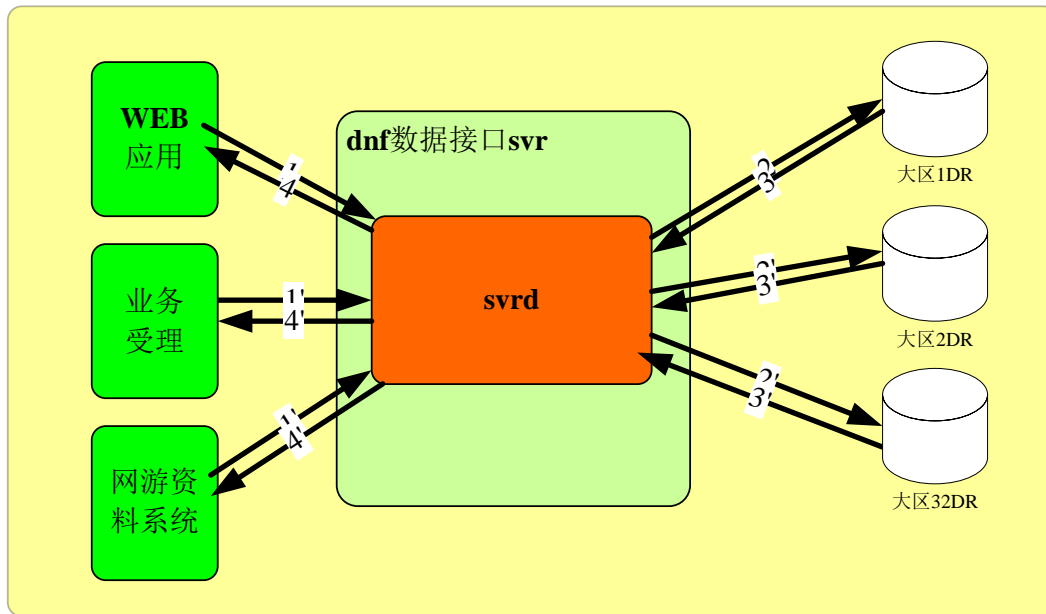


```
root@darkstar:/usr/local/isgw# telnet 172.16.26.169 5693
Trying 172.16.26.169...
Connected to 172.16.26.169.
Escape character is '^]'.
cmd=1001&uin=88883960&area=0
result=-1&info=get user info failed
cmd=1001&uin=88883960&area=120
result=-1&info=get user info failed
cmd=1001&uin=88883960&area=1
result=0&level=0
cmd=1001&uin=29320361&area=1
result=0&level=8
cmd=1001&uin=88883960&area=0
result=-1&info=get user info failed
cmd=1001&uin=88883960&area=2
result=0&level=0
```

## 6 应用举例

### 6.1 DNF 数据接口 svr

功能描述：为web前端提供访问dnf游戏资料的接口，修改dnf游戏数据库信息等的接口。这样的话，对于dnf所有的大区访问，对外提供访问接口的话，只需要部署一套数据接口svr即可实现其功能(只要性能本身能支持)。系统的框架如下。



## 7 FAQ

### 7.1 框架的性能怎样？同时的并发数为多少？

因为框架本身不对内存有特别的资源占用，所以框架本身的性能（不考虑后端业务逻辑模块的影响）决定于网络I/O的处理速度，away只做了简单的测试，作为大概的性能参考。

测试1：普通C2的服务器，15个客户端同时以200+次/秒的速度向框架svr发送消息（即3000+次/秒），框架svr的cpu占用只到0.1%

测试2：普通devnet服务器(2g内存，2核)

40+个客户端，每个客户端以间隔大概1ms（近似）的频率向后台发请求，后台日志看到的请求量是5000-6000左右的请求量 cpu 占用在 10-15% 左右

测试3：普通idc四核服务器(tlinux)

30+客户端，每个客户端以间隔大概1ms（近似）的频率向后台（50个后台线程）发请求，后台看到的请求量是13000左右的请求量，cpu总占用50%左右（测试客户端在另外一台机器）

同时的并发数是个容易误解的概念，如果是指同一时间（不是同一秒，好好理解其间的区别）处理消息的能力，框架的接收模块目前默认配置的消息数量为5000个（可配置），超过了会动态分配内存。

### 7.2 怎么查看系统异常情况？

系统框架里面如果有任何异常都会在日志里面提示"failed"（日志文件初始化成功之前除外），所以如果要监控日志，监控"failed"信息即可。

### 7.3 ACESvc enqueue msg failed?

如果看到类似提示，说明工作线程里面的消息队列满了，可能是因为工作线程后端的数据库或者网络阻塞等原因导致消息不能及时被处理，前端消息接收模块无法把消息放入工作线程的消息队列。可以考虑加到线程数，提高后端的并发能力来缓解，但是要根本解决还是要从后端的数据库或者网络连接来解决问题。

### 7.4 怎么打开/关闭调式信息？

调式信息比较详细的记录了消息在各个模块之间的传递，以及所有的异常情况。如果需要打开或者关闭请修改配置文件 isgw\_svr.ini 里面的 log\_mask 选项

-pDEBUG 打印DEBUG信息

-p~DEBUG 不打印DEBUG信息

### 7.5 日志说明

目前在用的日志级别有这几种

TRACE 最详细的跟踪日志，框架底层的日志，默认要设置为关闭

DEBUG 调试日志，一般是业务逻辑层面的调试日志，默认要设置为关闭

NOTICE 框架特别用来定位消息流转的日志，打开之后可以跟踪在各个模块之间流转情况 根据情况打开或者关闭

INFO 一般运行时的普通日志，默认要打开

ERROR 错误日志，出现系统或者业务逻辑的错误时都有可能使用此级别日志，默认要打开

通用规范：一般日志出现 failed error 等信息都是需要运营关注的方面。

通用错误日志说明（逐渐丰富）：

1 “ACESvc enqueue msg failed” 或者 “ISGWIntf putq msg to ISGWMgrSvc failed”

处理线程的队列已满，无法再进入处理线程队列，主要原因是处理线程不能及时处理消息，导致队列满，需要提高处理线程的处理能力，比如调大处理线程数量，提高处理线程中的接口性能等。

2 “ISGWIntf recv failed”

接收消息异常，可能是对方主动断开连接或者接收到了非法的消息等，具体需要看日志的详细信息说明

3 “ACE\_Object\_Que dequeue failed”

对象池没有对象可用，主要是因为对象池中的对象都被占用了，还没进行回收，可能是业务逻辑线程一直占用了对象还没处理完，没释放回来

4 PlatDbAccess exec update failed,PlatDbConn mysql\_real\_query failed:1062,Duplicate entry

表示 db 插入的时候出现了主键重复，无法插入成功，一般可能是前端进行了重复的操作了，1062 为 mysql 数据库的错误返回码，具体可以查看 mysql 的帮助文档，比如 《MySQL\_5.1\_zh.chm》

另： PlatDbAccess 的 fail 都是表示数据库连接方面的问题

5 TDB set failed,ret=-110

表示 tdb 设置失败 -110 的错误码表示 接收超时，具体错误码请参考《tdberror 错误码.txt》文档

6 ISGWack find msg owner failed

表示框架找不到 回收消息的 socket 句柄 主要是因为业务逻辑处理的时候可能处理的太慢了, 前端已经断开了连接导致

7 update 数据库的时候出现 affected\_rows=0 表示数据库更新没有影响, 说明本身数据就已经是更新时指定的状态/取值了

## 7.6 常见 core 的问题

2008年7月23日 有位同事在slackware10.1操作系统上使用ACE的epoll模式时, 一切按照正常的使用方式使用, 其他几个操作系统开发使用都没问题, 唯独这台机器(gcc3.3.4), 运行时仍然出现core的情况, core信息如下:

```
#0  0x00000000 in ?? ()
#1  0x080787d5 in ACE_Dev_Poll_Reactor::schedule_timer (this=0x8204ed8,
event_handler=0x82059e8, arg=0x0, delay=@0xbfd19138, interval=@0xbfd19140)
    at Timer_Queue_T.inl:205
#2  0x0808f742 in ACE_Reactor::schedule_timer (this=0xbfd19138,
event_handler=0x82059e8, arg=0x0, delta=@0xbfd19138, interval=@0xbfd19140)
    at ../../ace/Reactor.cpp:689
#3  0x08084222 in ACE_Logging_Strategy::init (this=0x82059e8, argc=9,
argv=0x8205918) at Time_Value.inl:81
#4  0x0805bd28 in ACEApp::init_log (this=0xbfd19630, log_num=-1) at ace_app.cpp:263
#5  0x0805b164 in ACEApp::init (this=0xbfd19630, argc=1, argv=0xbfd19974) at ace_app.cpp:146
#6  0x08058a2e in main (argc=1, argv=0xbfd19974) at iigw_svr.cpp:22
```

2012/12/10 在64位linux 上又遇到这个问题 后来改成了 select 模式的 程序就可以正常运行了 虽然具体原因还没找到 不知道是否一定跟epoll使用有关 只能先这么用了。

2013/05/22 在32 位linux上也遇到了这个问题(linux版本 suse Linux Tencent  
2.6.16.54-0.2.3-TENCENT-080623 #1 SMP Thu Jun 26 17:22:24 CST 2008 i686 i686 i386 GNU/Linux  
gcc/版本 4.1.2) 后来改成了select模式 程序正常启动。