本文假设读者已经熟读 服务器协议接口说明

# 增加新协议号

1. svn分支/deploy/common/netimpl/netdefine.lua 文件添加对应的协议号  
   示例:

DB\_LOGIN\_USER = 0x00010003, -- 账号登录

|  |  |
| --- | --- |
| DB\_LOGIN\_USER | 协议号对应的变量名 |
| 0x00010003 | 协议号 |
| 账号登录 | 协议内容注释 |

1. 协议号对应的变量名命令规则: 第一个下划线前面的字母, 为协议所属进程的缩写前缀, 详情见 netdefine.lua 文件里定义  
   目前已有的进程前缀：  
   |进程前缀|进程名| | ———— | ————- | DB\_ | dbmgr GBM\_ | globalmgr LM\_ | LoginMgr| CP\_ | ConnProxy GW\_ | Gateway GWI\_ | Gateway CE\_ | Cellapp CL\_ | Client

注:  
a. GWI\_ 主要用于服务器内部与gateway进程的通信  
b. CL\_ 主要用于服务器主动下发给客户端的协议

1. 协议号共 32 位, 前 16 位表示进程的序号, 后 16 位按协议的数量递增
2. 每个进程协议定义的未尾通常以 -- DB\_SERVANT\_END 这样的注释来结束
3. 在进程对应的 main 文件增加

addServant(NETDEFINE.DB\_LOGIN\_USER, "dbmgr.db\_login\_user")

1. dbmgr 进程较为特殊, 需要用 DBAddServant 替换 addServant

DBAddServant(NETDEFINE.DB\_LOGIN\_USER, "dbmgr.db\_login\_user")

# 增加新协议处理文件

1. svn分支\deploy\common\netimpl\进程名\ 目录下添加对应的 协议 处理文件  
   [协议内容说明如下](https://github.com/kinbei/NEO/blob/master/1/1.%E6%9C%8D%E5%8A%A1%E5%99%A8%E5%8D%8F%E8%AE%AE%E6%8E%A5%E5%8F%A3%E8%AF%B4%E6%98%8E.md)

# 增加逻辑处理文件

1. svn分支\deploy\server\lua\进程名\ 目录下添加对应的 逻辑 处理文件  
   逻辑文件格式如下:

-- ${process\_name} 替换为进程名  
-- ${servant\_name} 替换为协议名  
local function execute(request, current, session\_id, ip, port)  
 local \_, resp = create\_impl("netimpl.${process\_name}.${servant\_name}")  
 log\_debug("${servant\_name}|")  
   
 -- 处理内容在此添加  
   
 resp.retcode = ERRCODE.SUCCESS  
 daserver.response(current, resp)  
 return  
end  
return execute

dbmgr进程的逻辑文件格式较为特殊, 格式如下：

local DBDEF = require("dbmgr.dbdefine")  
local DB\_RECORD\_STATUS = DBDEF.DB\_RECORD\_STATUS  
  
-- ${servant\_name} 替换为协议名  
local function execute(transaction, request, current, session\_id, ip, port)   
 local \_, resp = create\_impl("netimpl.dbmgr.${servant\_name}")  
   
 local t = daodb.表名()  
 t:reset()  
 t.字段名 = xxx  
 t:insert()  
   
 -- 针对自增类型的字段, 可以在调用完 insert() 后, 用 t.字段名 获取字段值  
   
 transaction:commit() -- 必须调用 transaction:commit() 才会修改, 否则修改将会回滚  
 resp.retcode = ERRCODE.SUCCESS  
 daserver.response(current, resp)  
 return  
end  
  
return execute