# ProtoBuf文件格式说明

### 简介

Protocol Buffers是一种序列化数据结构的协议。对于透过管线(pipeline)或存储数据进行通信的程序开发上是很有用的。这个方法包含一个接口描述语言，描述一些数据结构，并提供程序工具根据这些描述产生代码，用于将这些数据结构产生或解析数据流。更细致的描述本文不再细化，可以参照<https://www.ibm.com/developerworks/cn/linux/l-cn-gpb/index.html>

### proto文件的命名规则

* 一般proto文件已$ID$MODULE.proto这样的命名方式命名
* 其中ID用来为MODULE独占一个ID段，一个ID段例如101可以提供10101~10199段的ID供模块使用，虽然ID值没有参与逻辑只是用来做标示，但是不要破坏当前规则，如果消息数量超过99应该考虑的是将功能进行细分成多个SubModule，而不是强行扩展使用类似101100,101101这种破坏约定的做法
* 版本开发时可能会出现多个功能同时开发的时候，注意和其他同事提前约定好各自要使用的ID避免冲突

### proto文件内部规则

项目中使用的proto文件格式和标准protobuf格式相差不大，只额外扩展了一个ID字段，例如

message cs\_account\_create[id=10003] {

optional string rname = 1; // 角色名字

optional uint64 icon = 2; // 创建角色时初始训练师的显示id

}

message sc\_account\_create[id=10004] {

optional uint64 result = 1; // 结果

// 1 := 创建成功

// 2 := 失败-名字长度限制

// 3 := 失败-含有非法字符

// 4 := 失败-含有敏感词汇

// 5 := 失败-已有名字冲突

// 7 := 失败-icon不存在

// 8 := 失败-角色已经上锁

// 9 := 失败-icon不是整形

}

去掉消息体重的[id=xx]字段，则文件就是标准的protobuf格式

* 一般根据具体的动作会有一对消息，例如上例中的cs\_account\_create 和sc\_account\_create，cs消息的名字的明明规则为cs\_$MODULE\_$ACTION，对应的sc消息只需要将消息头中的cs改为sc即可，cs消息的命名规则参与逻辑，所以不能破坏，必须遵守
* 如果需要嵌套结构，例如

message p\_ger[id=10403] {

optional uint64 gerid =1; // 精灵ID

optional uint64 typeid =2; // 模版ID

optional uint64 star =3; // 精灵星级

}

命名规则为p\_$MODULE\_$EXTRA，EXTRA为可选字段，该命名规则不参与逻辑，但是因为所有的message最后都是一起进行消息的打包和解包，所以在消息中带上module避免不同模块中定义了同名proto引起冲突

* 如果用到了其他proto模块中的结构体，不需要按照标准protobuf格式进行include，当然主动include也不会报错，而且支持IDEA的插件内的proto跨文件跳转
* 消息名字中的所有字母最终都会被转成小写字母，这是protobuf内部实现的需要，也就是即使定义消息的时候用了大写的字母，实际逻辑中也会被替换成小写，所以编写时不要使用大写字母来定义消息名
* sc消息中一般会带有result字段用来表示操作结果，如果操作失败了一般只需要将check函数中的result值赋值，其他字段可以不填值，但是注意result字段如果有，必须命名为result 这个命名参与了逻辑，不可随意更改

### proto文件的注释规则

* 整个文件头为

// 精灵协议 $MODULE\_NAME or DOC

// 编号=104

并没有逻辑作用，只是用来标记

* 消息的注释放在message定义的上一行，该注释可选，一个好的消息定义应该从名字就可以明确的推断出对应的功能，不推荐这里加入注释
* 消息内字段注释紧跟在字段定义的当前行最后
* result值的注释需要另起一行，格式必须按照

optional uint64 result = 1; // 登录结果

// 1 := 成功

// 2 := 失败-验证错误

// 3 := 失败-登录超时

// 4 := 失败-账号被封

// 5 := 失败-人数已满

// 6 := 失败-服务器ID

格式为// ERRNO := ERRSTRTING

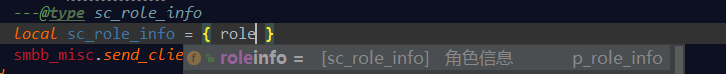
这部分会参与到ERRNO的生成逻辑，所以必须保证格式符合要求

1. 扩展

我们通过插件和python脚本可以做到lua中的消息提示，目前有两种使用方法

* 使用@type标示当前table为一个sc消息类型，可以出现代码提示

消息的具体type定义会有插件进行导出，导出后只需要在回复时提前定义一个sc消息对应的table，例如



@type需要自己主动的显示声明，声明后构造table的时候就会有proto字段的名字提示

* 使用sc.sc\_message 插件功能可以自动生成对应的消息格式

输入sc.sc\_xxx后就会有对应消息的提示例如



如果是本角色调用send\_client即可，tab后del删除ex\_send字段即可

如果是公共进程调用提示出现后直接del删除send\_client字段即可

之后tab完成其他字段填充即可

1. ErrorNo使用

待补充