#### **字段格式：**

**限定修饰符① | 数据类型② | 字段名称③ | = | 字段编码值④ | [字段默认值⑤]**

#### **①．限定修饰符包含 required\optional\repeated**

###### **Required:**

表示是一个必须字段，必须相对于发送方，在发送消息之前必须设置该字段的值，对于接收方，必须能够识别该字段的意思。发送之前没有设置required字段或者无法识别required字段都会引发编解码异常，导致消息被丢弃。

###### **Optional：**

表示是一个可选字段，可选对于发送方，在发送消息时，可以有选择性的设置或者不设置该字段的值。对于接收方，如果能够识别可选字段就进行相应的处理，如果无法识别，则忽略该字段，消息中的其它字段正常处理。---因为optional字段的特性，很多接口在升级版本中都把后来添加的字段都统一的设置为optional字段，这样老的版本无需升级程序也可以正常的与新的软件进行通信，只不过新的字段无法识别而已，因为并不是每个节点都需要新的功能，因此可以做到按需升级和平滑过渡。

###### **Repeated：**

表示该字段可以包含0~N个元素。其特性和optional一样，但是每一次可以包含多个值。可以看作是在传递一个数组的值。

#### **②．数据类型**

Protobuf定义了一套基本数据类型。几乎都可以映射到C++\Java等语言的基础数据类型.

| **protobuf 数据类型** | **描述** | **打包** | **C++语言映射** |
| --- | --- | --- | --- |
| bool | 布尔类型 | 1字节 | bool |
| double | 64位浮点数 | N | double |
| float | 32为浮点数 | N | float |
| int32 | 32位整数、 | N | int |
| uin32 | 无符号32位整数 | N | unsigned int |
| int64 | 64位整数 | N | \_\_int64 |
| uint64 | 64为无符号整 | N | unsigned \_\_int64 |
| sint32 | 32位整数，处理负数效率更高 | N | int32 |
| sing64 | 64位整数 处理负数效率更高 | N | \_\_int64 |
| fixed32 | 32位无符号整数 | 4 | unsigned int32 |
| fixed64 | 64位无符号整数 | 8 | unsigned \_\_int64 |
| sfixed32 | 32位整数、能以更高的效率处理负数 | 4 | unsigned int32 |
| sfixed64 | 64为整数 | 8 | unsigned \_\_int64 |
| string | 只能处理 ASCII字符 | N | std::string |
| bytes | 用于处理多字节的语言字符、如中文 | N | std::string |
| enum | 可以包含一个用户自定义的枚举类型uint32 | N(uint32) | enum |
| message | 可以包含一个用户自定义的消息类型 | N | object of class |

N 表示打包的字节并不是固定。而是根据数据的大小或者长度。

例如int32，如果数值比较小，在0~127时，使用一个字节打包。

关于枚举的打包方式和uint32相同。

关于message，类似于C语言中的结构包含另外一个结构作为数据成员一样。

关于 fixed32 和int32的区别。fixed32的打包效率比int32的效率高，但是使用的空间一般比int32多。因此一个属于时间效率高，一个属于空间效率高。根据项目的实际情况，一般选择fixed32，如果遇到对传输数据量要求比较苛刻的环境，可以选择int32.

#### **③．字段名称**

字段名称的命名与C、C++、Java等语言的变量命名方式几乎是相同的。

protobuf建议字段的命名采用以下划线分割的驼峰式。例如 first\_name 而不是firstName.

#### **④．字段编码值**

有了该值，通信双方才能互相识别对方的字段。当然相同的编码值，其限定修饰符和数据类型必须相同。

编码值的取值范围为 1~2^32（4294967296）。

其中 1~15的编码时间和空间效率都是最高的，编码值越大，其编码的时间和空间效率就越低（相对于1-15），当然一般情况下相邻的2个值编码效率的是相同的，除非2个值恰好实在4字节，12字节，20字节等的临界区。比如15和16.

1900~2000编码值为Google protobuf 系统内部保留值，建议不要在自己的项目中使用。

protobuf 还建议把经常要传递的值把其字段编码设置为1-15之间的值。

消息中的字段的编码值无需连续，只要是合法的，并且不能在同一个消息中有字段包含相同的编码值。

建议：项目投入运营以后涉及到版本升级时的新增消息字段全部使用optional或者repeated，尽量不实用required。如果使用了required，需要全网统一升级，如果使用optional或者repeated可以平滑升级。

#### **⑤．默认值。**

当在传递数据时，对于required数据类型，如果用户没有设置值，则使用默认值传递到对端。当接受数据是，对于optional字段，如果没有接收到optional字段，则设置为默认值。

#### **关于import**

protobuf 接口文件可以像C语言的h文件一个，分离为多个，在需要的时候通过 import导入需要对文件。其行为和C语言的#include或者java的import的行为大致相同。

#### **关于package**

避免名称冲突，可以给每个文件指定一个package名称，对于java解析为java中的包。对于C++则解析为名称空间。

#### **关于message**

支持嵌套消息，消息可以包含另一个消息作为其字段。也可以在消息内定义一个新的消息。

#### **关于 enum**

枚举的定义和Java 相同，但是有一些限制。

枚举值必须大于等于0的整数。

使用分号(;)分隔枚举变量而不是Java 语言中的逗号(,)

##### **eg.**

enum VoipProtocol

{

​ H323 = 1;

​ SIP = 2;

​ MGCP = 3;

​ H248 = 4;

}