**问题：**

部分灵活qinq配置不生效。比如：

switchport vlan-mapping both cvlan 100 cos 2 add-outer 200 translate 300

这条命令目的是匹配cvlan 100且cos值为2的vlan，加一层svlan，同时将内层cvlan修改为300，变成双层vlan转发。

但是实际测试验证，该规则不生效。

**分析定位：**

1. **确认现象**

进一步测试其他qinq命令行，发现只匹配cvlan，或只匹配svlan+cvlan时规则都正常生效，但是在匹配vlan之外再匹配cos时规则就不生效。比如：

switchport vlan-mapping both cvlan 100 add-outer 200 translate 300

这条命令就能正常匹配单层vlan 100，修改为双层vlan (200,300)转发。

1. **分析现象**

带cos的规则和不带cos的规则在配置上有什么不同？而且两条命令配置都没有报错。

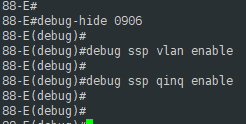
需要查看代码的配置逻辑，首先查看命令行的详细配置过程，通过debug打印信息初步定位。

1. **定位过程**

测试验证不生效后，请教同事，怀疑是查表类型不对（mac和IP两种类型），带着这个线索继续。

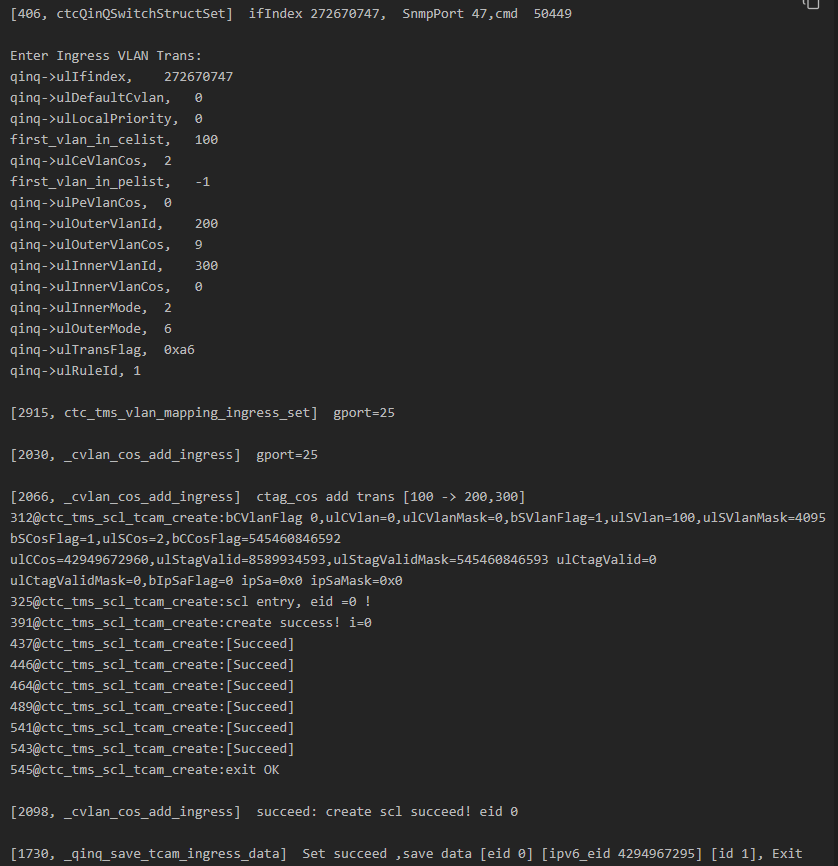
**3.1 ssp的配置打印**

首先打开ssp层的debug开关，查看命令行配置的详细流程。在debug节点下使用debug命令打开vlan和qinq相关的配置打印：

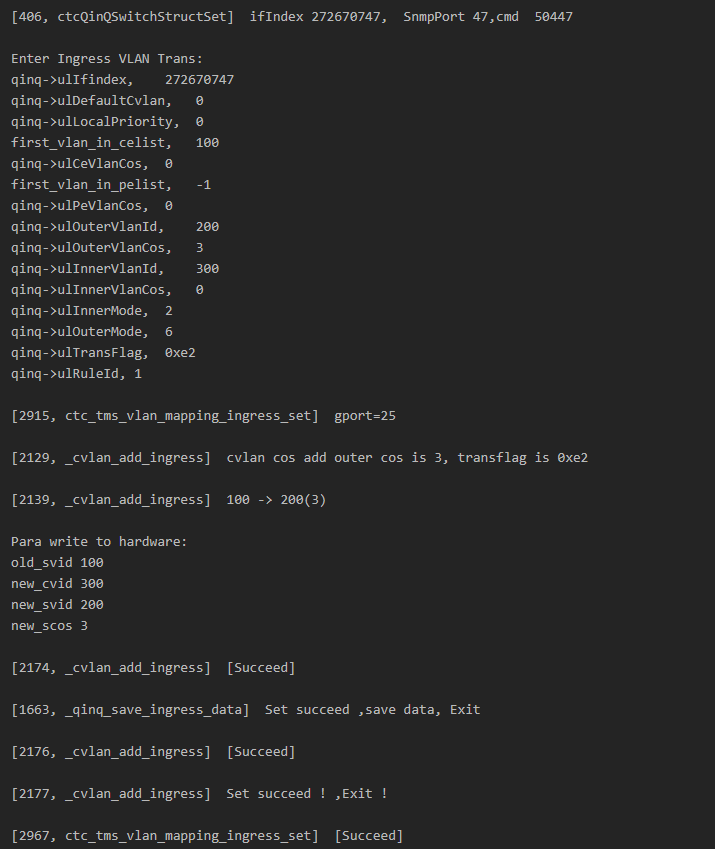


这个debug命令可以指定不同模块的不通功能，只将需要模块的打印输出到串口，避免打印过多。

需要匹配cos值的命令行配置打印，ingress方向如下：



不需要匹配cos，只匹配vlan的命令行配置打印，ingress方向如下：



从打印上可以看出，匹配cos和不匹配cos都是通过：

ctc\_tms\_vlan\_mapping\_ingress\_set

函数接口下发的配置，但是下一步的函数调用出现不同。匹配cos的命令行调用的是：

\_cvlan\_cos\_add\_ingress

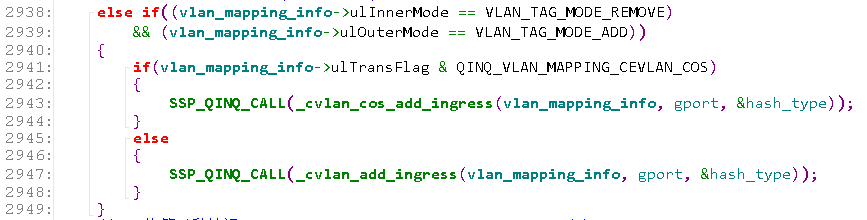
而不匹配cos的命令行配置调用的是：

\_cvlan\_add\_ingress

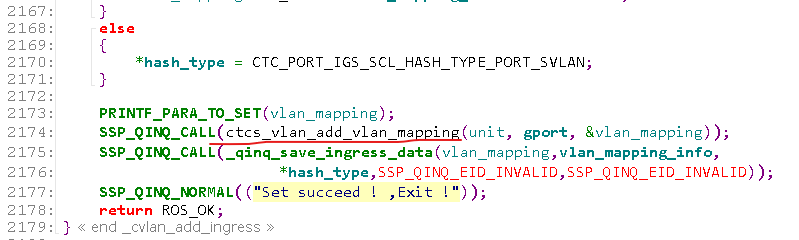
找到两个不同的调用接口，接下来需要看代码，看这两个步骤分别做了哪些配置。

**3.2 查看代码**

在相同的入口函数ctc\_tms\_vlan\_mapping\_ingress\_set 中找到不同的调用代码：

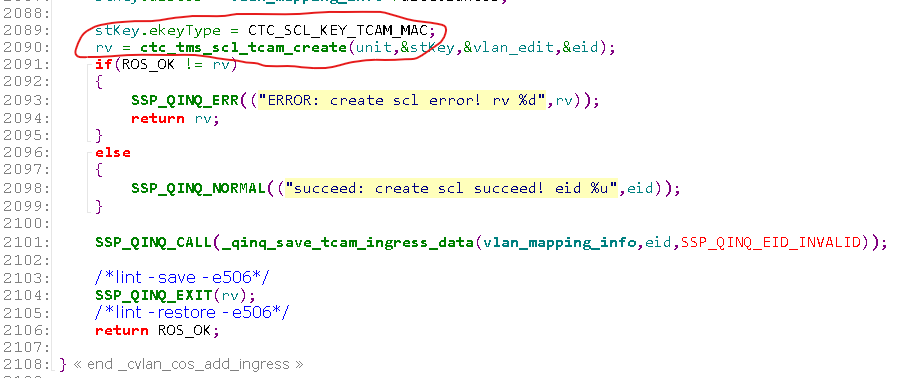


先看生效的命令行调用：\_cvlan\_add\_ingress

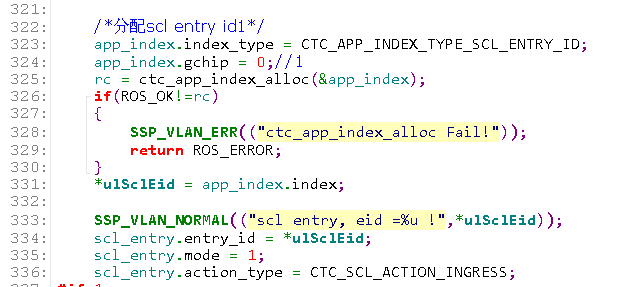


发现这个函数没有更深的函数调用，在这个函数最后直接调用了SDK的接口下发配置。

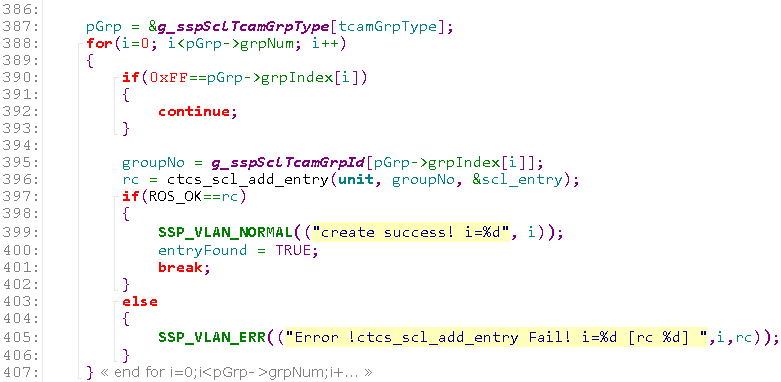
再看有问题的命令行调用：\_cvlan\_cos\_add\_ingress



在解析完传入的各个参数后，调用了一个scl tcam表的创建接口，后续也没有其他调用。说明这个函数就是配置下发的地方。进入该函数查看具体内容：

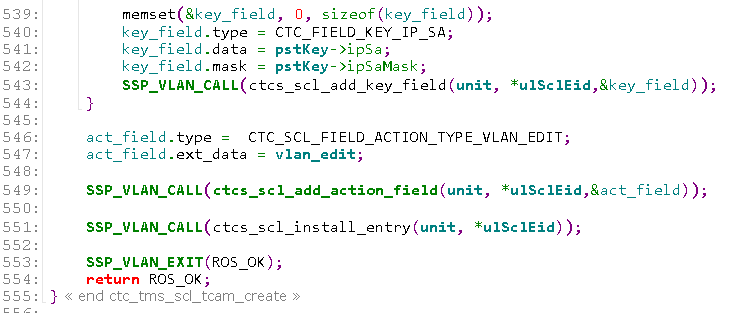


函数开始就分配一个entry表项，然后开始配置entry的各个参数属性，然后调用SDK接口下发这条entry表项：



这里有一个group的概念，猜测应该类似acl，scl的entry是属于一个group组，而这里group id通过全局数组给定，说明这些group应该是已经创建完成了。具体在哪创建暂时不管。

后面继续根据传入的参数配置entry的各个匹配项规则：



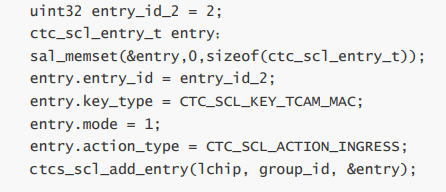
匹配和vlan修改规则添加完成后，通过一个add和一个install的SDK接口使配置生效。

到这一步可以确定问题是两条命令在匹配时使用的方案不一样，一个是直接调用

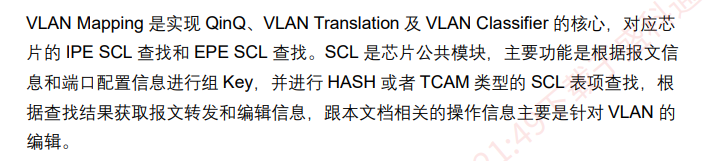
ctcs\_vlan\_add\_vlan\_mapping，能正常生效，一个是使用scl下发表项，出现问题。需要弄明白这两种方案的不同。

**3.3 查看芯片手册**

在用户开发指南手册里没有SDK接口的详细说明，只有类似下面的简要介绍：



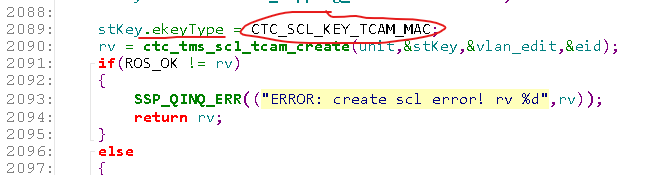
从截图上看，两个接口应该是两种方式，一个是hash，一个tcam。这一点在芯片原理手册上可以确定：



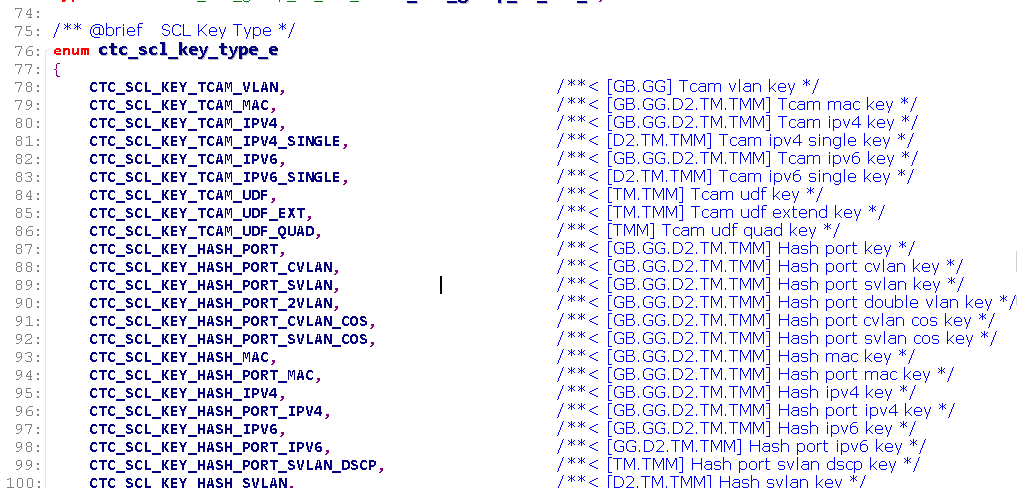
scl分为hash和tcam两种查找方式，而上面两个接口就是这两种方式的配置接口。通过代码中的一些配置项也可以佐证，在配置类似key\_type这种意思明确的属性时，一个使用的SDK宏带有明显的HASH\_TYPE字段，一个带有明显的KEY\_TCAM字段。

**3.4 查看代码**

既然是使用tcam表项的命令有问题，那就查看tcam的各项配置。



在\_cvlan\_cos\_add\_ingress的这个地方，给的tcam类型是CTC\_SCL\_KEY\_TCAM\_MAC，这个赋值在SDK的代码里查找如下：

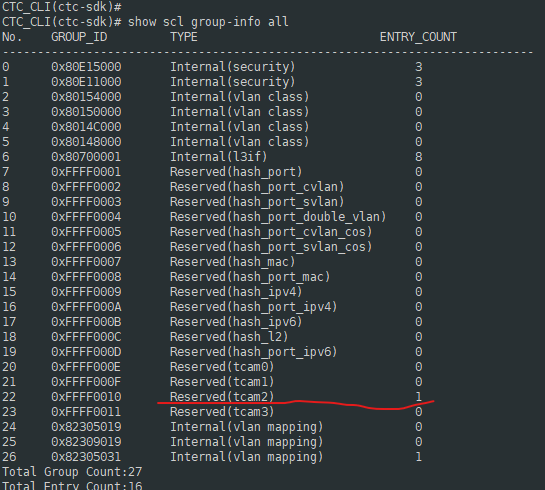


枚举的结构里包含了几个tcam类型，有vlan、mac、IP等，还包含更多的hash类型。

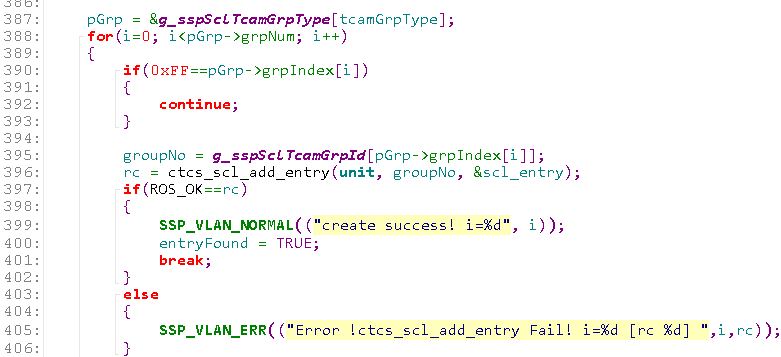
**3.5 sdk表项**

了解了基本原理后，需要查看不生效命令行配置在SDK表项中的具体配置。

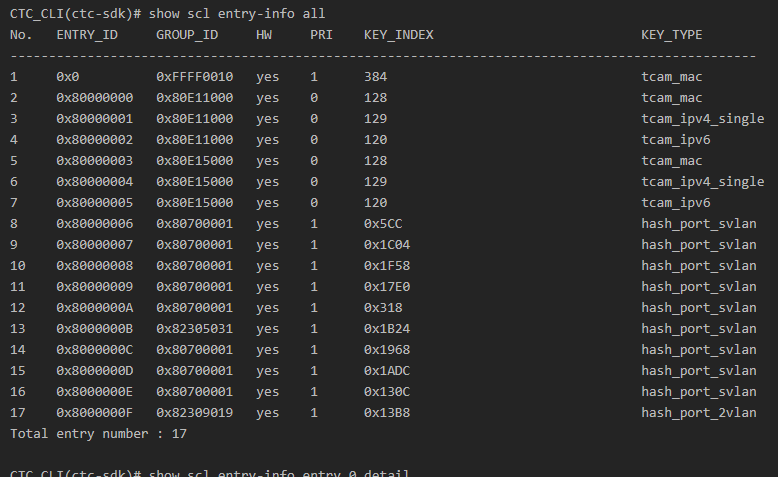
（1）查看所有group



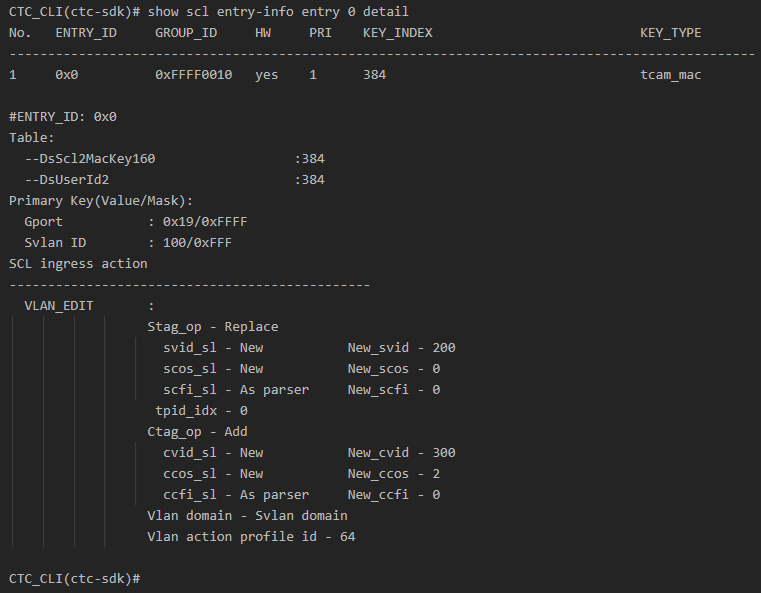
命令行配置前后，最后一列的ENTRY\_COUNT计数有发生变化，说明接口上配置命令下发的entry表项是在对应的group，也就是标红的tcam2。从代码中也能确认，是下发到这个group。



（2）查看entry



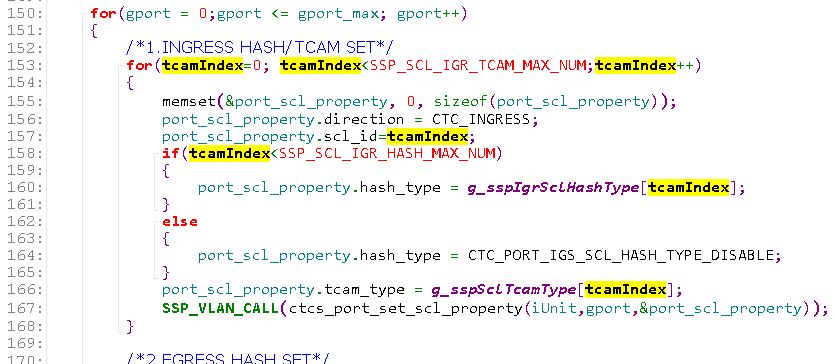
找到group对应的entry id，再查看该entry的详细内容。看最后一列的KEY\_TYPE，ingress方向的tcam查表确实是mac类型的。



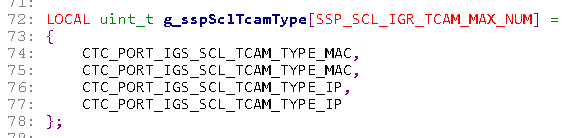
**3.6 代码里tcam2的初始化**

命令行使用tcam2时，设置的查找是mac，这时候需要看下创建tcam2的这个group有哪些配置。

在ctc\_tms\_port\_scl\_init函数中4个tcam是这么设置的，tcam2和tcam3设置为hash查找disable：



相应的tcam\_type设置使用的全局数组是这样定义的：



也就是说**tcam2在接口下初始化的类型是IP的，而下发qinq命令添加的entry表项是mac类型**，这样就会出现规则无法匹配，命令不生效的问题。

**3.7 找到原因，验证**

上一步已经大概确定了命令行配置不生效的原因，这里使用sdk的命令，配置验证：

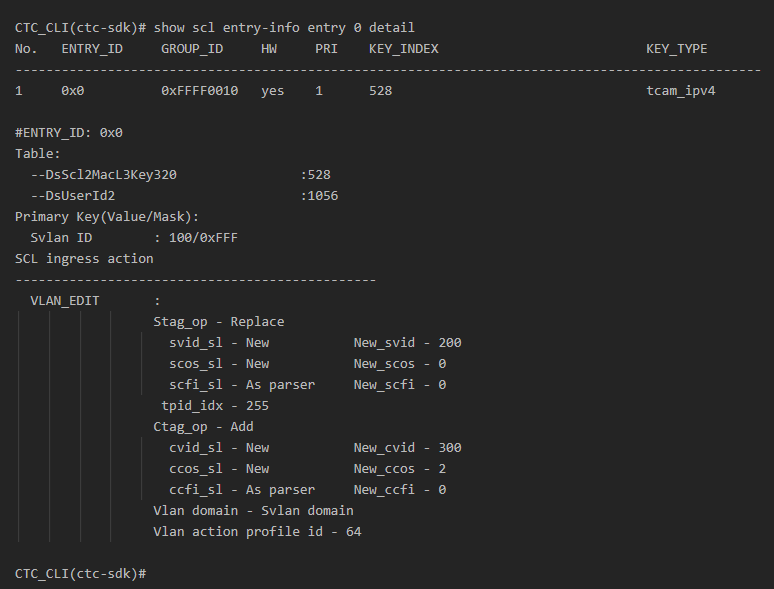
（1）设置接口对应group下的匹配规则

scl add group 0xFFFF0010 entry 0 ipv4-entry svlan 100 0xFFF stag-cos 2 0x7 ingress-action vlan-edit stag-op 4 svid-sl 2 new-svid 200 scos-sl 2 new-scos 0 ctag-op 2 cvid-sl 2 new-cvid 300 ccos-sl 2 new-ccos 2

（2）使配置生效

scl install entry 0

（3）查看entry配置



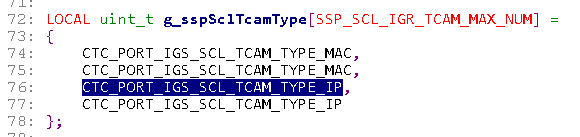
下发的tcam类型修改为tcam\_ipv4，与之前不生效时的tcam\_mac不同。

（3）打流验证

打vlan 100，cos值为2的报文，在出口抓包查看，配置生效。

1. **修改代码**

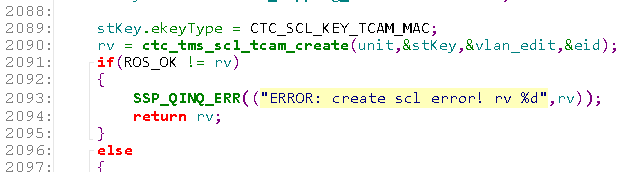
直接修改初始化使用的全局数组是一个方法，但是彭雪莲说之前有别的问题改过这个mac和ip，而且这个其他模块也会使用，所以这个地方不改。



代码tcam类型是在

\_cvlan\_cos\_add\_ingress

里调用这个函数之前设置的：



这个地方设置tcam类型改成IP：CTC\_SCL\_KEY\_TCAM\_IPV4。

另外，匹配优先级加vlan的命令行也不生效，原因一样，都是这个地方的tcam设置成mac类型。

**其他**

port 0x19 scl-key-type scl-id 0 direction ingress type disable tcam-type mac

scl create group 0 type port 0x19

scl add group 0 entry 0 mac-entry svlan 100 0xFFF stag-cos 2 0x7 ingress-action vlan-edit stag-op 4 svid-sl 2 new-svid 200 scos-sl 2 new-scos 0 ctag-op 2 cvid-sl 2 new-cvid 300 ccos-sl 2 new-ccos 2

scl install group 0

上面四条sdk下的配置可以生效，匹配vlan + cos修改为两层vlan转发。

如果把第三条匹配命令中的mac-entry参数替换成ipv4-entry，那就会出现不生效，无法匹配的问题。