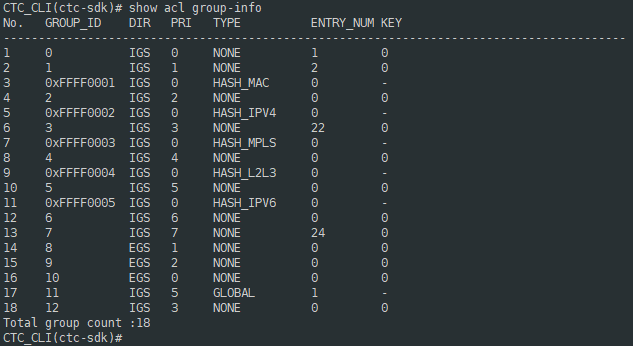
设备上mac地址白名单功能一般是用来验证接入本设备的客户mac地址是否在白名单mac内，如果不在白名单内，则不允许该客户接入设备，来自该客户的业务也不做处理，在入口即丢弃。

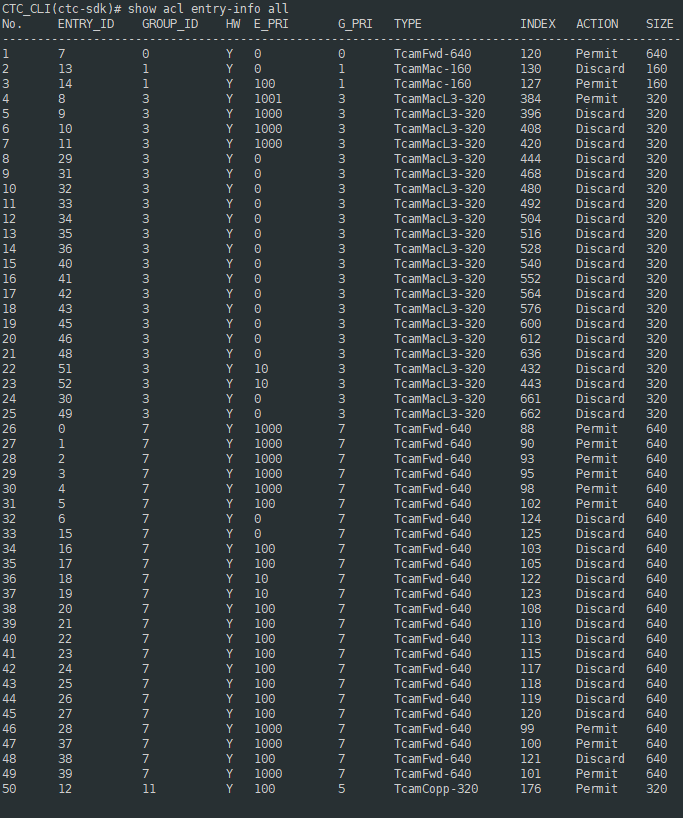
**查看group统计**

88-E设备初始化会有一些group和entry的配置，在sdk命令界面查看相关的group信息：



**查看已配置的entry**

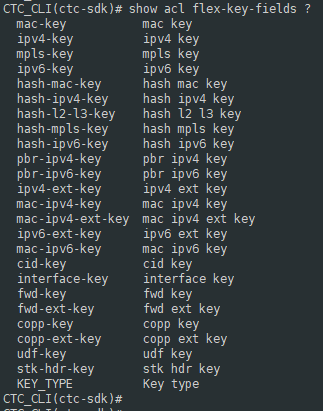
也可以查看所有已配置的entry列表：



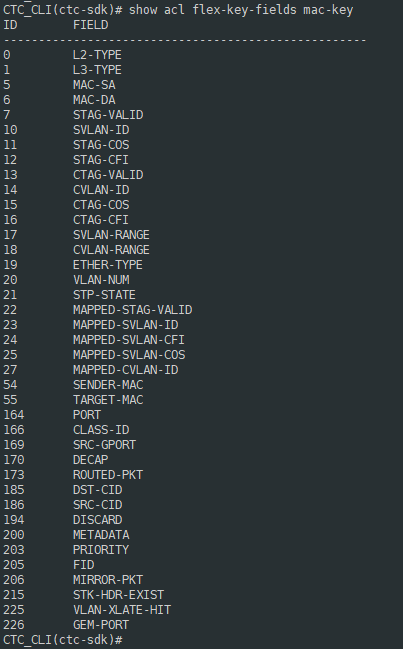
**配置白名单功能**

**选择key类型**

白名单一般需要匹配接入客户的源mac，所以配置的acl规则匹配源mac即可，对应的就是acl规则的key类型只需要mac类型，不需要其他字段信息。8180支持匹配的key类型可以通过fles-key-fields查看：

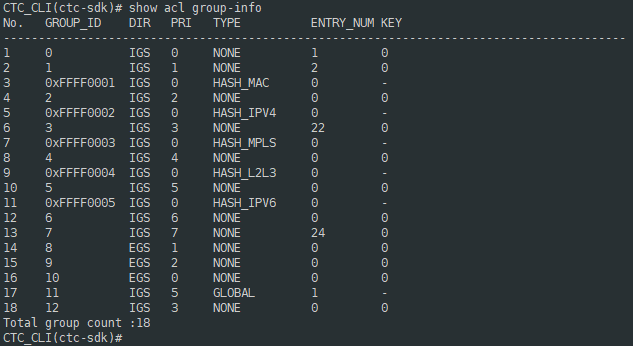


mac-key类型支持匹配的字段：



如果要匹配除了mac之外其他的IP网络层字段，需要选择mac-ipv4-key，或者其他mac+ip的key类型。

设备初始化创建了group，验证命令行直接选择其中一个group就行：



**匹配源mac允许通过**

ingress方向选择group\_id为1的group配置一条entry：

**acl add group 1 entry 44 mac-entry mac-sa 0000.0000.0111 ffff.ffff.ffff cancel-deny**

目的是使用mac-entry的key类型，匹配源mac为0000.0000.0111，掩码为六个字节全0xff，匹配动作为允许，一般是permit，这里用cancel-deny验证（选择permit一样能匹配通过，详细的不同之处有待学习）。

注意entry\_id的选择不能与已经配置的id重叠，可以用show acl entry-info all查看，选择一个未配置的id。

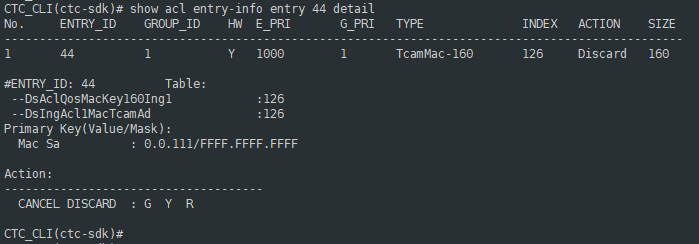
还需要修改该条entry的优先级为最高，对接收的报文先匹配这条规则：

**acl entry 44 priority 1000**

安装上面配置的entry表项，下发到芯片使实际生效：

**acl install entry 44**

上述三条命令配置完成后可以查看配置的entry详细信息：



查看结果显示规则已经下发到硬件生效。右上角的ACTION显示为Discard不影响，实际上还是允许匹配上的报文通过。

**匹配空字段丢弃**

上述entry的目的是允许匹配上的源mac报文通过，为实现白名单功能还需要丢弃未匹配上的其他所有报文，所以还需要下发一条规则丢弃其他未匹配上的报文。同样使用mac-entry类型的key，匹配内容为空，表示不匹配，处理动作为deny，表示丢弃该报文：

**acl add group 1 entry 47 mac-entry mask 0xffff deny**

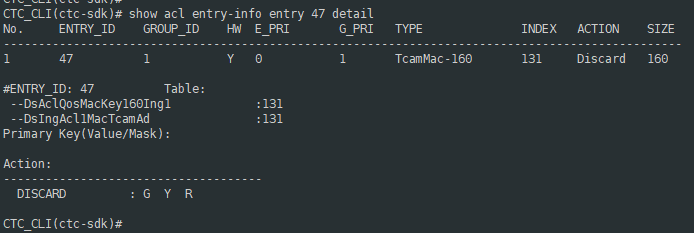
修改该条规则的优先级为0，当上一条匹配源mac的高优先级规则未匹配上时会接着匹配本条规则，按本条规则的动作直接丢弃报文：

**acl entry 47 priority 0**

安装上面配置的entry表项，下发到芯片使实际生效：

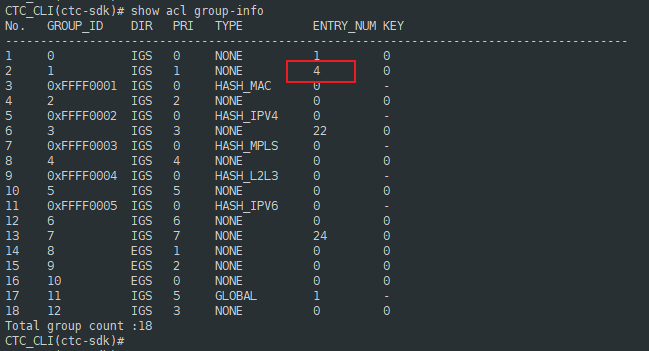
**acl install entry 47**

上述三条命令配置完成后可以查看配置的entry详细信息：



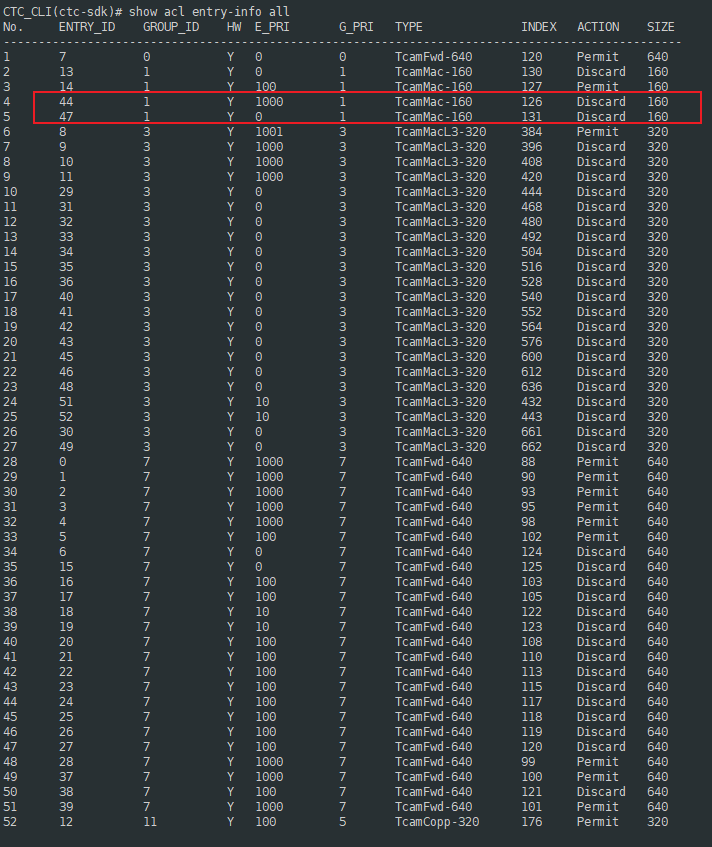
查看结果显示规则已经下发到硬件生效，最下面的Action显示匹配动作为丢弃。

配置完后再次查看group信息：



可以看到group\_id为1的group已经配置了4条entry，配置白名单之前这个数值是2。

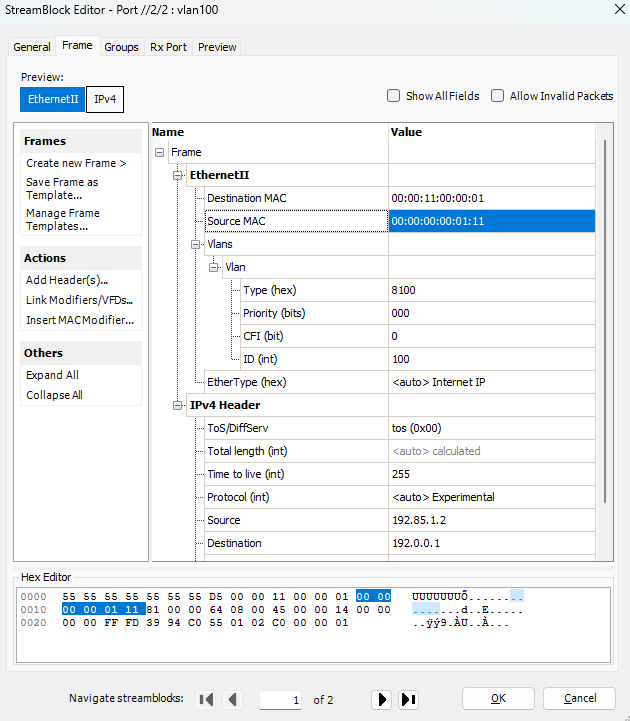
再查看entry信息：



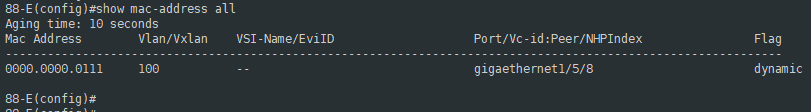
可以看到新增的两条entry表项。

**打流验证**

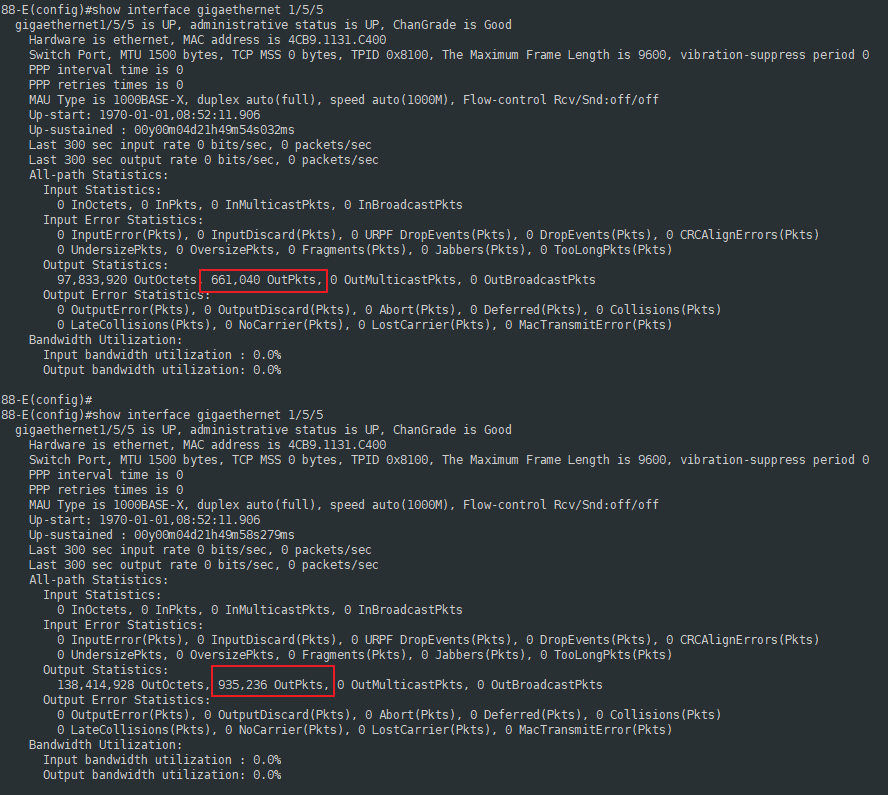
**打源mac匹配的tag报文**



设备上查看mac地址学习信息：



可以看到接仪表的端口1/5/8口能学习到配置的mac地址，查看另一个linkup的接口1/5/5口的发包统计：

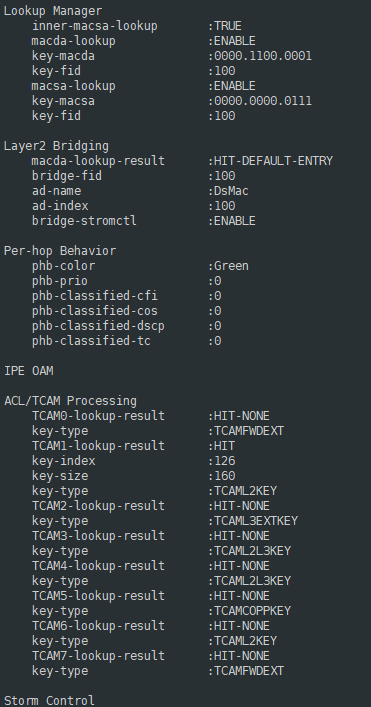


打的报文能通过入接口1/5/8的acl匹配正常转发出去。在sdk中查看详细的查表信息：

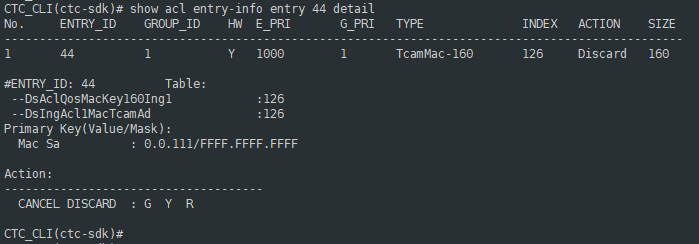
**diag trigger pkt-trace mode network-packet**

**show diag pkt-trace detail**

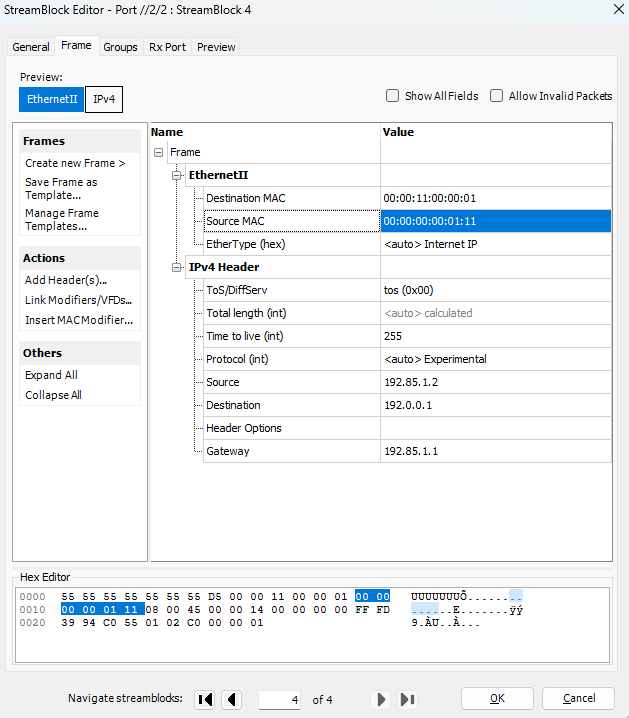
能看到芯片转发流程中的各个模块处理信息：



对应的acl模块的查表信息可以看到有hit到配置的acl表项，hit到的key-index为126，正好是配置的entry\_id为44的规则：



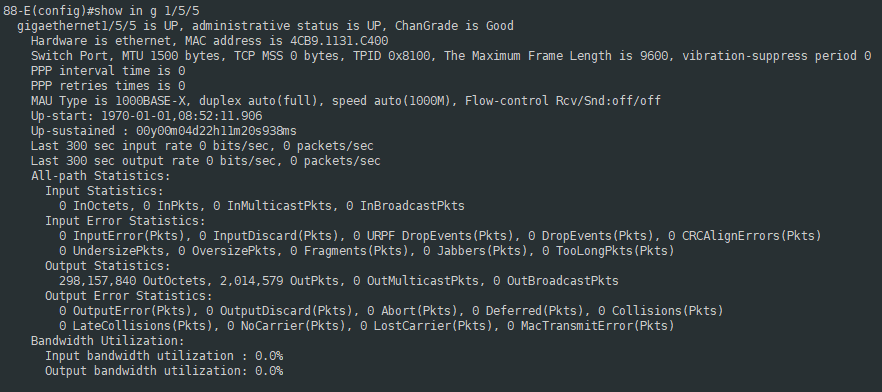
**打源mac匹配的untag报文**



设备上查看mac地址学习信息：



可以看到接仪表的端口1/5/8口能学习到配置的mac地址，查看另一个linkup的接口1/5/5口的发包统计：

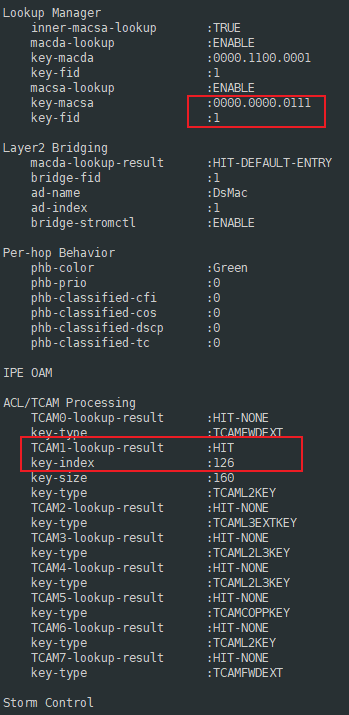


打的报文能通过入接口1/5/8的acl匹配正常转发出去。在sdk中查看详细的查表信息：

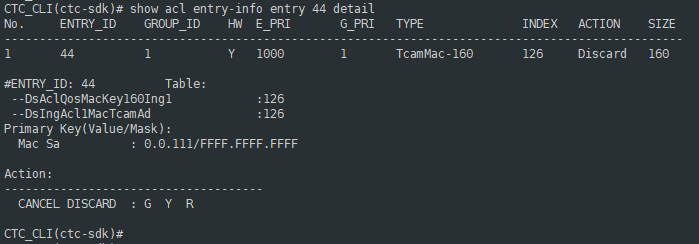
**diag trigger pkt-trace mode network-packet**

**show diag pkt-trace detail**

能看到芯片转发流程中的各个模块处理信息：



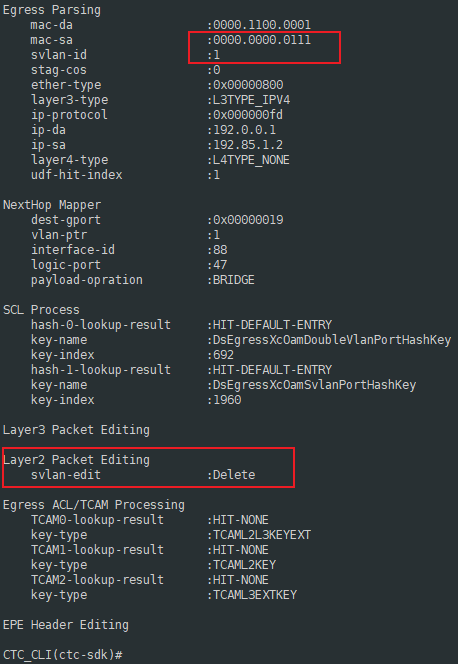
对应的acl模块的查表信息可以看到有hit到配置的acl表项，hit到的key-index为126，正好是配置的entry\_id为44的规则：



正常转发的流程在命令行：

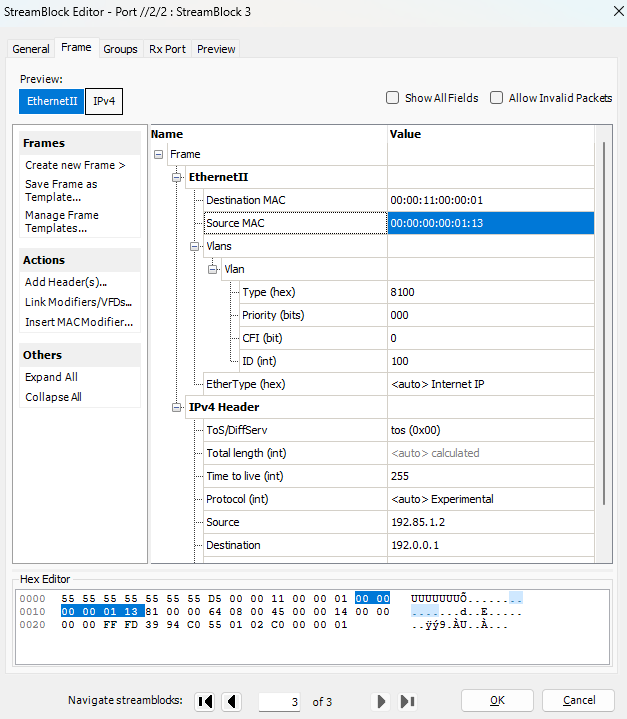
**show diag pkt-trace detail**

回显信息的后半部分还可以看到出方向的处理过程：

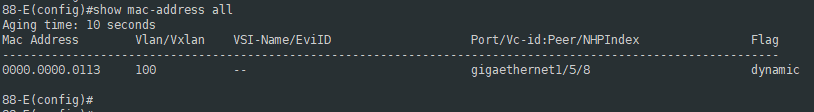


因为是untag报文，而出接口配置的native vlan正好为1，报文在入接口打上默认的vlan 1后，在出接口转发出去时需要剥掉vlan 1，所以egress方向的处理过程能看到svlan的编辑动作是delete。

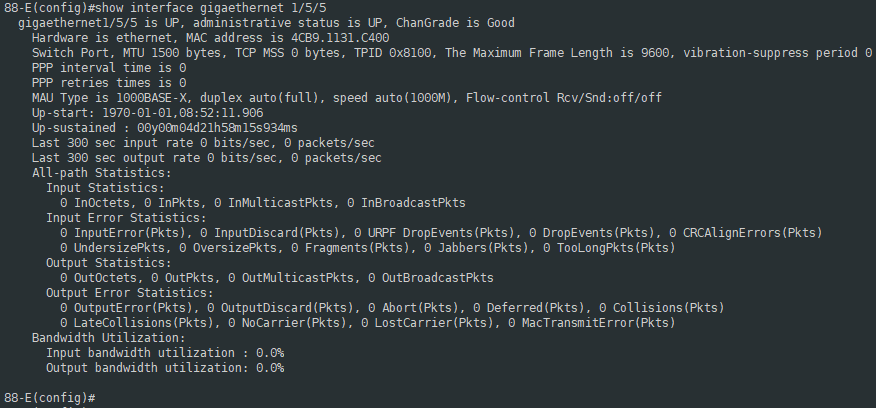
**打源mac不匹配的tag报文**



设备上查看mac地址学习信息：



可以看到接仪表的端口1/5/8口能学习到配置的mac地址，查看另一个linkup的接口1/5/5口的发包统计：

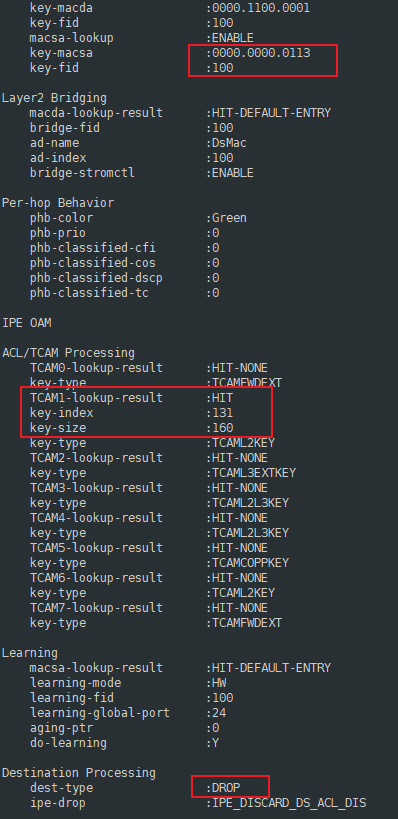


打的报文虽然能在入接口1/5/8学习到mac，但是无法通过acl匹配正常转发。在sdk中查看详细的查表信息：

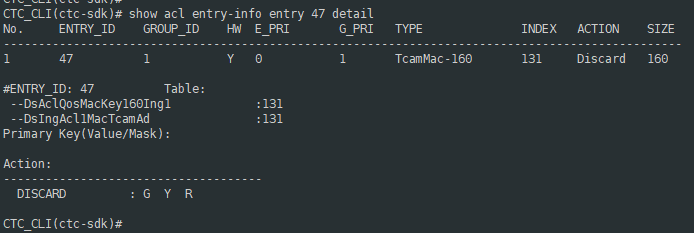
**diag trigger pkt-trace mode network-packet**

**show diag pkt-trace detail**

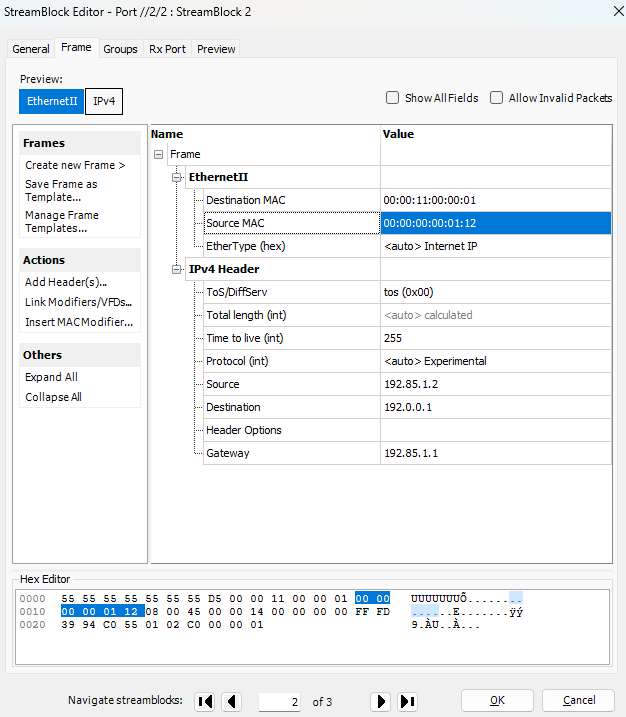
能看到芯片转发流程中的各个模块处理信息：



对应的acl模块的查表信息可以看到有hit到配置的acl表项，hit到的key-index为131，正好是配置的entry\_id为47的规则：



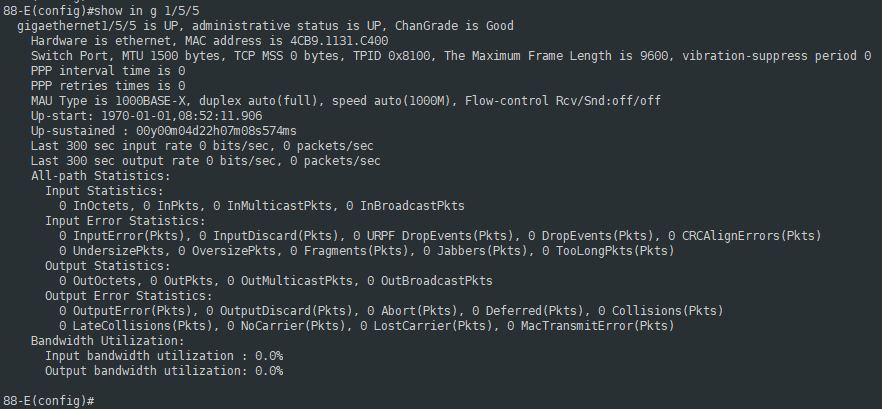
**打源mac不匹配的untag报文**



设备上查看mac地址学习信息：



可以看到接仪表的端口1/5/8口能学习到配置的mac地址，查看另一个linkup的接口1/5/5口的发包统计：

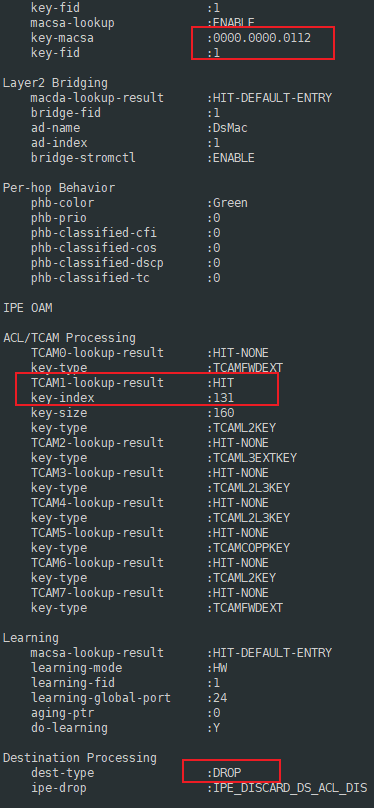


打的报文虽然能在入接口1/5/8学习到mac，但是无法通过acl匹配正常转发。在sdk中查看详细的查表信息：

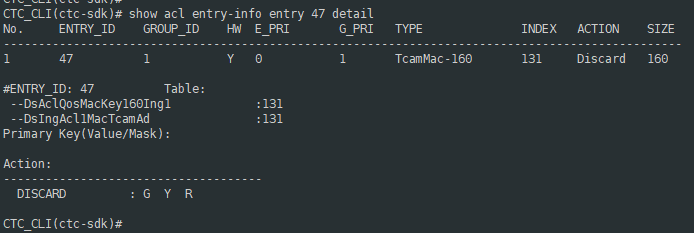
**diag trigger pkt-trace mode network-packet**

**show diag pkt-trace detail**

能看到芯片转发流程中的各个模块处理信息：

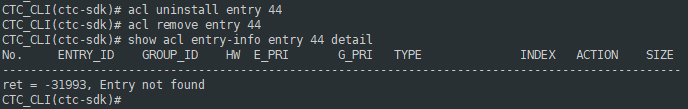


对应的acl模块的查表信息可以看到有hit到配置的acl表项，hit到的key-index为131，正好是配置的entry\_id为47的规则：



**匹配源mac允许通过，并上送cpu**

删除前面配置的entry 44：



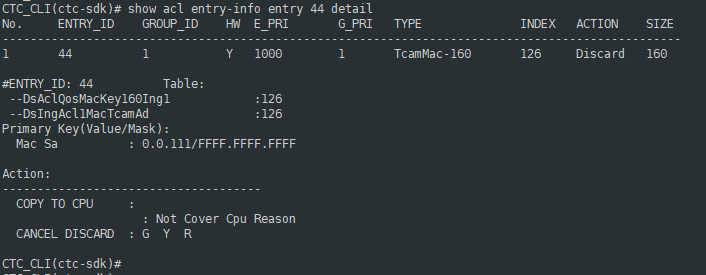
重新配置匹配规则：

**acl add group 1 entry 44 mac-entry mac-sa 0000.0000.0111 ffff.ffff.ffff cancel-deny copy-to-cpu**

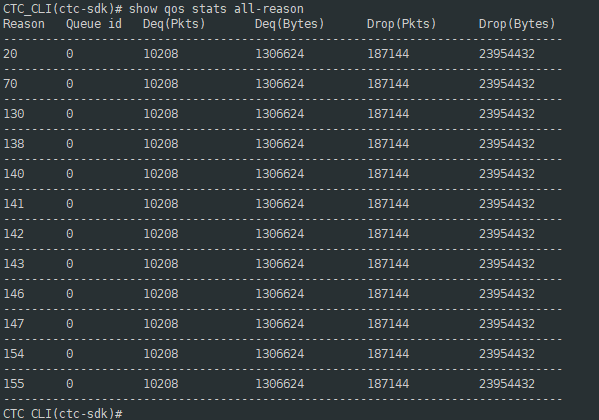
**acl entry 44 priority 1000**

**acl install entry 44**

上述三条命令配置完成后可以查看配置的entry详细信息：



打源mac匹配的报文时，转发和查表信息与前面没有配置上送copy\_to\_cpu时一致，能正常学习mac，也能正常转发报文，只需要验证是否正常上送cpu：



因为配置acl规则时没有reason的选项，所以显示出的reason有多个，这个不影响。后两列的丢包是cpu接口的限速导致，目的是保护cpu不被大量报文冲死。

当打源mac不匹配的报文时，查看上述qos队列0的统计是没有内容的：

