

EARL 8 Classification Manager: A Behavioral Examination of LOUs, L4Ops, and Capmap Tables

Contents

[Introduction](#)

[Background Information](#)

[Program Capmap Tables and LOU Registers](#)

[Case Study #1 - ACLs with TCP Flags](#)

[Case Study #2 - 100% LOU Register Usage](#)

[Case Study #3 - QoS Programming with L4Ops](#)

[Case Study #4 - Dual-Stack ACLs Cause CAPMAP Exhaustion](#)

Introduction

This document describes how Logical Operation Units (LOUs) and Layer 4 Operations (L4Ops) are programmed into the capmap table. It provides failure scenarios, the kinds of errors you typically encounter in these situations, and what you should infer from these errors.

Classification Manager (CM) manages the classification Ternary Content Addressable Memory (TCAM) and associated resources such as labels, LOUs, capmap entries, and others. CM services are used by the Feature Manager (FM) and the QoS Manager (QM) to program TCAM entries to support the Cisco IOS[®] Access Control List (ACL) and Quality of Service (QoS) features.

Background Information

LOUs and L4Ops - LOUs stands for Logical Operation Units, which are hardware registers that are used to store {operator, operand} tuples for TCP/UDP port numbers specified in ACLs and VLAN Access Control Lists (VACLs). These tuples are also called as L4Ops. For example, if you match host X to host Y gt 1023, then the tuple becomes {gt, 1023}.

L4Ops - Layer 4 Operations.

Capmap tables - The L4Ops described previously are programmed into LOU registers that are referenced by entries in capmap tables. Each capmap table has a limit of 10 (one is reserved for direction, which brings the limit down to nine) entries (L4Ops). Capmap tables are indexed by the TCAM label itself.

There are two TCAMs, A and B; each TCAM has 8K labels. For each TCAM, there is one capmap table of 2K entries. Since each TCAM has 8K labels, there is a 4:1 overlap here - four labels map to one capmap entry. The overlap is: **1=2049=4097=6145**.

Basically, this means that TCAM labels 1, 2049, 4097, and 6145 use the same capmap index. Cisco's traditional implementation of TCAM label allocation led to problems because of this


```

0
I M 32740 0x1FFF 0 0x00 0x000 255.255.255.255 - 0.0.0.0
- 0 0 0x0
I V 32741 4097 0 0 0 1.1.1.2 - 0.0.0.0
- 0 0 0 - ----- dscp=6; 0x00000000000000038
0
I M 32741 0x1FFF 0 0x00 0x000 255.255.255.255 - 0.0.0.0
- 0 0 0x0
I V 32745 4097 0 0 0 0.0.0.0 - 0.0.0.0
- 0 0 0 - ----- - 0x0000000040000038
0
I M 32745 0x1FFF 0 0x00 0x000 0.0.0.0 - 0.0.0.0
- 0 0 0x0

```

The TCAM label allocated for interface VLAN 1 is 2049 and the TCAM label allocated to interface VLAN 2 is 4097. This means that both of these interfaces use the same capmap table in order to reference the LOU registers for their L4Op programming.

You can confirm this with this command (five ACEs in ACL a1 and four ACEs in ACL a2 implies you should see the capmap table as full):

```

Sup2T#show platform hardware acl capmap tcam B label 4097
Hardware Capmap Table Entry For TCAM B. Free items are not shown

```

Index	Loc[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
-----	-----	---	---	---	---	---	---	---	---	---
1	212	10	9	8	7	6	5	4	3	2

```

Sup2T#show platform hardware acl capmap tcam B label 2049
Hardware Capmap Table Entry For TCAM B. Free items are not shown

```

Index	Loc[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
-----	-----	---	---	---	---	---	---	---	---	---
1	212	10	9	8	7	6	5	4	3	2

So now, at this stage, if you attempt to install another L4Op-based Access Control Entry (ACE), which is not expandable, for any of these interfaces, you would receive the **No free capmap entry available** error.

```

Sup2T(config)#ip access-list extended a2
Sup2T(config-ext-nacl)#permit ip host 1.1.1.2 any dscp 10
Sup2T(config-ext-nacl)#end

```

```

*Sep 16 14:57:55.983: %EARL_CM-5-NOCAPMAP: No free capmap entry available
*Sep 16 14:57:55.991: %FMCORE-4-RACL_REDUCED: Interface Vlan2 routed traffic
will be software switched in ingress direction. L2 features may not be applied
at the interface

```

This results in software bridging the entire interface that would potentially cause slower switching, high CPU utilization, and other related problems.

Note: Cisco bug ID [CSCuo02666](#) was raised to fix this problem. The biggest change in logic it introduces is how the TCAM labels are allocated. Now Cisco allocates TCAM labels continuously (2,3,4,5, and so on) up to 2048 instead of at gaps of 2K. This means that capmap tables are no longer shared from the beginning.

Remember that LOUs, like any other hardware resource, are limited. There is a total of 104 LOUs available for use:

```

Sup2T#show platform software acl lou

```

LOUs Registers (shadow copies)

Index	Type	A_Op	A_Val	A_Cnt	B_Op	B_Val	B_Cnt
0	PKT_QOS_GI	A is free.			NEQ	0	1
1	DST_PORT	LT	81	2	B is free.		
2	B & A are free						
3	B & A are free						
4	B & A are free						
5	B & A are free						
6	B & A are free						
7	B & A are free						
8	B & A are free						
9	B & A are free						
10	B & A are free						
11	B & A are free						
12	B & A are free						
13	B & A are free						
14	B & A are free						
15	B & A are free						

snip

95 B & A are free
96 B & A are free
97 B & A are free
98 B & A are free
99 B & A are free
100 B & A are free
101 B & A are free
102 B & A are free
103 B & A are free

Program Capmap Tables and LOU Registers

Capmap tables are used only when L4 operations must be taken into account. Note that matching on Differentiated Services Code Point (DSCP)/Class of Service (CoS) values is also considered as a L4Op. Here is a simple example (that uses a version of code that includes the fix of Cisco bug ID [CSCuo02666](#)) that this document builds on incrementally:

```
Sup2T#show ip access-lists a3
Extended IP access list a3
 10 permit ip host 192.168.1.1 host 192.168.1.2
```

I have this applied to interface VLAN 1.

```
Sup2T#show run int vlan 1
Building configuration...

Current configuration : 84 bytes
!
interface Vlan1
 ip address 192.168.1.1 255.255.255.0
 ip access-group a3 in
end
```

This is correctly programmed into TCAM:

```
Sup2T#show platform hardware acl entry interface vlan 1 security in ip
mls_if_index:20000001 dir:0 feature:0 proto:0
```

pass#0 features

fno:0

tcam:B, bank:1, prot:0 Aces

Permit ip host 192.168.1.1 host 192.168.1.2
L3_Deny ip any any

Sup2t-MA1.7#show platform hardware acl entry interface vlan 1 security in ip detail
mls_if_index:20000001 dir:0 feature:0 proto:0

pass#0 features

UAPRSF: U-urg, A-ack, P-psh, R-rst, S-syn, F-fin
MLGFI: M-mpls_plus_ip_pkt, L-L4_hdr_vld, G-gpid_present, F-global_fmt_match, I-ife/ofe
's' means set; 'u' means unset; '-' means don't care

Table with columns: I, INDEX, LABEL, FS, ACOS, AS, IP_SA, SRC_PORT, IP_DA, DST_PORT, F, FF, L4PROT, TCP-F:UAPRSF, MLGFI, OtherL4OPs, RSLT, CNT. Includes separator lines.

fno:0

tcam:B, bank:1, prot:0 Aces

Table with columns: I, V, INDEX, LABEL, FS, ACOS, AS, IP_SA, SRC_PORT, DST_PORT, F, FF, L4PROT, TCP-F:UAPRSF, MLGFI, OtherL4OPs, RSLT, CNT. Shows ACL entries for TCAM label 2.

Capmap tables are referenced via the TCAM label itself. You can use the TCAM label in the show platform software [hardware] acl capmap tcam <> label <> command in order to view the corresponding table (software or hardware) for this TCAM label.

Sup2T#show platform hardware acl capmap tcam B label 2
Hardware Capmap Table Entry For TCAM B. Free items are not shown

Table with columns: Index, Loc[9], [8], [7], [6], [5], [4], [3], [2], [1], [0]. Shows bit patterns for TCAM label 2.

Nothing is allocated in the capmap table for this label. The defined ACL has no L4Ops; there is no requirement to install an entry in the capmap table.

Change this ACE to this:

Sup2T#show ip access-lists a3


```

1      -----      LOU0 A register
2      -----      LOU1 B register
3      -----      LOU1 A register
.....
.....
206    -----      LOU103 B register
207    -----      LOU103 A register
208    -----      Global format match for global acl
209    -----      Group id present
210    -----      L4_hdr_vld
211    -----      Mpls_plus_ip_pkt
212    -----      ife/ofe for direction
(213-223)  ----      Reserved
(224-239)  ----      16 TCP flags map
(240-255)  ----      16 IPv6 ext header map

```

You can see that the L4op_sel value of 0 points to the LOU0 B register, the value of 1 points to the LOU0 A register, the value of 2 points to the LOU1 B register, the value of 3 points to the LOU1 A register, and so on. The A register is always programmed first. The **3/1/1** output makes more sense now that you see this.

In this output, 3 means that the L4Op was programmed into the LOU1 A register. You can also verify where a L4Op is programmed if you look into the contents of the LOU registers directly:

```

Sup2T#show platform software acl lou
LOUs Registers (shadow copies)

```

Index	Type	A_Op	A_Val	A_Cnt	B_Op	B_Val	B_Cnt
0	PKT_QOS_GI	A is free.			NEQ	0	1
1	DST_PORT	LT	81	1	B is free.		
2	B & A are free						
3	B & A are free						
4	B & A are free						

snip

```

Sup2T#show platform hardware acl lou
Dumping h/w lou values

```

Index	lou_mux_sel	A_Opcode	A_Value	B_Opcode	B_Value
0	7	NEQ	0	NEQ	0
1	1	LT	81	NEQ	0
2	0	NEQ	0	NEQ	0
3	0	NEQ	0	NEQ	0

snip

As you can see, a (gt, X) tuple gets programmed as (LT, X+1) in the LOU registers.

Note: L4Ops get programmed into LOU registers ONLY when they are applied to interfaces. If ACLs are created with L4Ops (without the ACL actually being applied to an interface), it does not program the applicable L4Ops into LOU registers.

Remove the ACL from the interface VLAN 1 and look at the LOU registers again:

```

Sup2T(config)#int vlan 1
Sup2T(config-if)#no ip access-group a3 in

```

```

Sup2T#show platform software acl lou

```


LOUs Registers (shadow copies)

Index	Type	A_Op	A_Val	A_Cnt	B_Op	B_Val	B_Cnt
0	PKT_QOS_GI	A is free.			NEQ	0	1
1	B & A are free						
2	B & A are free						
3	B & A are free						
4	B & A are free						

snip

Sup2T#show platform hardware acl lou

Dumping h/w lou values

Index	lou_mux_sel	A_Opcode	A_Value	B_Opcode	B_Value
0	7	NEQ	0	NEQ	0
1	1	NEQ	0	NEQ	0
2	0	NEQ	0	NEQ	0
3	0	NEQ	0	NEQ	0

snip

Case Study #1 - ACLs with TCP Flags

TCP flags have a special set of registers allocated within the LOU registers range. You can view this range via the **show platform software acl capmap mapping** command as shown here:

Sup2T#show platform software acl capmap mapping

L4op_sel	value	Reference
0	-----	LOU0 B register
1	-----	LOU0 A register
2	-----	LOU1 B register
3	-----	LOU1 A register
.....	
.....	
206	-----	LOU103 B register
207	-----	LOU103 A register
208	-----	Global format match for global acl
209	-----	Group id present
210	-----	L4_hdr_vld
211	-----	Mpls_plus_ip_pkt
212	-----	ife/ofe for direction
(213-223)	----	Reserved
(224-239)	----	16 TCP flags map
(240-255)	----	16 IPv6 ext header map

L4op_sel values 224-239 are available to use for TCP flags, which gives you a set of 16 registers for use. Here is a simple example to demonstrate this. This ACL is defined:

```
Sup2T(config)#ip access-list extended a13
Sup2T(config-ext-nacl)#permit tcp host 192.168.13.10 host 192.168.13.20 syn
Sup2T(config-ext-nacl)#exit
```

Apply this inbound on interface VLAN 13:

```
Sup2T(config)#int vlan 13
Sup2T(config-if)#ip access-group a13 in
Sup2T(config-if)#end
```


Add another ACE to the current ACL a13. This should have a different TCP flag:

```
Sup2T(config)#ip access-list extended a13
Sup2T(config-ext-nacl)#permit tcp host 192.168.1.1 host 192.168.1.2 ack
Sup2T(config-ext-nacl)#exit
```

If you look at the capmap table again, you see that another TCP register is used:

```
Sup2T#show platform software acl capmap tcam b label 13
Shadow Capmap Table Entry For TCAM B
```

Output in a RST/INV/CNT format: RST - result value; INV - inverted;
CNT - aggregated reference account;

CBF - number of free cap bits (one per entry);
Free items are not shown

Index CBF [9] [8] [7] [6]
[5] [4] [3] [2] [1]
[0]

13 7 212/0/1 Free Free Free
Free Free Free Free 225/0/1 224/0/1

So as you can see, the programming is done per flag; for each flag, an independent TCP register is used and you maintain a count against that. This means that, for each flag, you also use one capmap entry. This further implies that technically you cannot actually go beyond nine flags in your ACL or you will reach the capmap limit before the 16 flag TCP limit.

Scale up your TCP flags so that you reach the TCP register limit in order to see what happens. This example shows several ACLs that have been configured and applied to different interface VLANs in order to reach a 100% TCP register utilization as shown here:

```
Sup2T#show platform hardware capacity acl
```

Classification Mgr Tcam Resources

Key: Ttlent - Total TCAM entries, QoSent - QoS TCAM entries, LOU - LOUs,
RBLent - RBACL TCAM entries, Lbl - Labels, TCP - TCP Flags,
Dsttbl - Destinfo Table, Ethcam - Ethertype Cam Table,
ACTtbl - Accounting Table, V6ext - V6 Extn Hdr Table

Module	Ttlent	QoSent	RBLent	Lbl	LOU	TCP	Dsttbl	Ethcam	ACTtbl	V6ext
1	3%	7%	0%	1%	96%	100%	1%	0%	0%	0%
3	3%	7%	0%	1%	96%	100%	1%	0%	0%	0%
4	3%	7%	0%	1%	96%	100%	1%	0%	0%	0%
6	3%	7%	0%	1%	96%	100%	2%	0%	0%	0%

At this stage, if you decide to configure another ACL with a unique TCP flag (or a combination of TCP flags that makes it unique) and apply this to an interface, then you must program a new TCP flag (or a combination of flags) to a TCP register. However, there are no hardware registers available. In this situation, you bridge the entire interface.

```
Sup2T(config)#ip access-list extended a29
Sup2T(config-ext-nacl)#permit tcp host 192.168.1.1 host 192.168.1.2 psh rst
```

```
Sup2T(config-if)#int vlan 29
Sup2T(config-if)#ip access-group a29 in
```

```
*Oct 6 13:57:47.612: %FMCORE-4-RACL_REDUCED: Interface Vlan29 routed traffic
will be software switched in ingress direction. L2 features may not be applied
```

at the interface

```
Sup2T#show platform hardware acl entry interface vlan 29 security in ip  
mls_if_index:2000001D dir:0 feature:0 proto:0
```

pass#0 features

fno:0

tcam:B, bank:1, prot:0 Aces

Bridge ip any any

Case Study #2 - 100% LOU Register Usage

Remember that LOUs are a finite resource - you can run out of space for those as well. You can monitor LOU usage with this command:

```
Sup2T#show platform hardware capacity acl
```

Classification Mgr Tcam Resources

Key: Ttlent - Total TCAM entries, QoSent - QoS TCAM entries, LOU - LOUs,
RBLent - RBACL TCAM entries, Lbl - Labels, TCP - TCP Flags,
Dsttbl - Destinfo Table, Ethcam - Ethertype Cam Table,
ACTtbl - Accounting Table, V6ext - V6 Extn Hdr Table

Module	Ttlent	QoSent	RBLent	Lbl	LOU	TCP	Dsttbl	Ethcam	ACTtbl	V6ext
1	2%	7%	0%	1%	1%	0%	1%	0%	0%	0%
3	2%	7%	0%	1%	1%	0%	1%	0%	0%	0%
4	2%	7%	0%	1%	1%	0%	1%	0%	0%	0%
6	2%	7%	0%	1%	1%	0%	2%	0%	0%	0%

Scale the ACLs in order to use more LOUs. After the installation of several ACLs (with the range command which takes two LOU registers, both A and B), this example shows 96% LOU usage:

```
Sup2T#show platform hardware capacity acl
```

Classification Mgr Tcam Resources

Key: Ttlent - Total TCAM entries, QoSent - QoS TCAM entries, LOU - LOUs,
RBLent - RBACL TCAM entries, Lbl - Labels, TCP - TCP Flags,
Dsttbl - Destinfo Table, Ethcam - Ethertype Cam Table,
ACTtbl - Accounting Table, V6ext - V6 Extn Hdr Table

Module	Ttlent	QoSent	RBLent	Lbl	LOU	TCP	Dsttbl	Ethcam	ACTtbl	V6ext
1	3%	7%	0%	1%	96%	0%	1%	0%	0%	0%
3	3%	7%	0%	1%	96%	0%	1%	0%	0%	0%
4	3%	7%	0%	1%	96%	0%	1%	0%	0%	0%
6	3%	7%	0%	1%	96%	0%	2%	0%	0%	0%

Create another ACL and apply that to an interface that would cause the LOU usage to go beyond 100%.

```
Sup2T(config)#ip access-list extended a12
```

```
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1401 1410  
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1411 1420  
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1421 1430  
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1431 1440  
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1441 1450  
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1451 1460  
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1461 1470  
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1471 1480  
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1481 1490
```

```
Sup2T(config-ext-nacl)#$68.14.1 host 192.168.14.2 range 1491 1500
```

```
Sup2T(config-ext-nacl)#exit
```

```
Sup2T(config)#int vlan 12
```

```
Sup2T(config-if)#ip access-group a12 in
```

The example reached 100% LOU usage; however, notice that no error message was received.

```
Sup2T#show platform hardware capacity acl
```

```
Classification Mgr Tcam Resources
```

```
Key: Ttlent - Total TCAM entries, QoSent - QoS TCAM entries, LOU - LOUs,  
RBLent - RBACL TCAM entries, Lbl - Labels, TCP - TCP Flags,  
Dsttbl - Destinfo Table, Ethcam - Ethertype Cam Table,  
ACTtbl - Accounting Table, V6ext - V6 Extn Hdr Table
```

Module	Ttlent	QoSent	RBLent	Lbl	LOU	TCP	Dsttbl	Ethcam	ACTtbl	V6ext
1	3%	7%	0%	1%	100%	0%	1%	0%	0%	0%
3	3%	7%	0%	1%	100%	0%	1%	0%	0%	0%
4	3%	7%	0%	1%	100%	0%	1%	0%	0%	0%
6	3%	7%	0%	1%	100%	0%	2%	0%	0%	0%

Here is another test. Now that the LOU is at 100%, take a very simple L4Op and try to install that for an interface. Configure this ACL:

```
Sup2T#show ip access-lists a13
```

```
Extended IP access list a13
```

```
10 permit tcp host 192.168.14.1 host 192.168.14.2 range 1600 1650
```

Apply this inbound to interface VLAN 13.

```
Sup2T#show run int vlan 13
```

```
Building configuration...
```

```
Current configuration : 87 bytes
```

```
!
```

```
interface Vlan13
```

```
ip address 192.168.13.1 255.255.255.0
```

```
ip access-group a13 in
```

```
end
```

Look at the TCAM for this VLAN now:

```
Sup2T#show platform hardware acl entry interface vlan 13 sec in ip
```

```
mls_if_index:2000000D dir:0 feature:0 proto:0
```

```
pass#0 features
```

```
fno:0
```

```
tcam:B, bank:0, prot:0 Aces
```

```
Permit tcp host 192.168.14.1 host 192.168.14.2 eq 1650
```

```
Permit tcp host 192.168.14.1 host 192.168.14.2 range 1648 1649
```

```
Permit tcp host 192.168.14.1 host 192.168.14.2 range 1632 1647
```

```
Permit tcp host 192.168.14.1 host 192.168.14.2 range 1600 1631
```

```
Permit tcp host 192.168.14.1 host 192.168.14.2 fragments
```

```
L3_Deny ip any any
```

The L4Ops have been expanded. If you look at the capmap table for this TCAM label, you see that nothing is installed.

```
Sup2T#show platform hardware acl entry interface vlan 13 sec in ip detail
```

```
mls_if_index:2000000D dir:0 feature:0 proto:0
```



```

-----
 14      9 212/0/1                Free                Free                Free
Free                Free                Free                Free                Free
Free

```

Here is an explanation of what happened. Because the LOU registers are full, you can no longer install any new L4Ops there and nothing can be referenced in the capmap table. At this stage, you still attempt to install the L4Ops in TCAM by expanding them. If the L4Ops are non-expandable, then you software switch the entire interface in the given direction.

What does a 100% LOU register usage imply? Your TCAM starts to fill quickly (due to L4Op expansion). If you attempt to install non-expandable L4Ops, then with the current implementation, your entire interface gets software bridged.

As it stands now, an error is only generated when you attempt to install a non-expandable L4Op in such a situation. This example modified the current ACL a13 that was applied to interface VLAN 13 with the addition of a non-expandable L4Op.

```

Sup2T(config)#ip access-list extended a13
Sup2T(config-ext-nacl)#permit tcp host 192.168.14.1 host 192.168.14.2 dscp 40

Oct  5 04:50:13.104: %FMCORE-4-RACL_REDUCED: Interface Vlan13 routed traffic will
be software switched in ingress direction. L2 features may not be applied at the
interface
Oct  5 04:50:13.096: %EARL_CM-DFC3-5-NOLOU: No free LOU entry available on the EARL
Oct  5 04:50:13.096: %EARL_CM-DFC1-5-NOLOU: No free LOU entry available on the EARL
Oct  5 04:50:13.096: %EARL_CM-DFC4-5-NOLOU: No free LOU entry available on the EARL

```

```

Sup2T#show platform hardware acl entry interface vlan 13 security in ip
mls_if_index:2000000D dir:0 feature:0 proto:0

```

```
pass#0 features
```

```
fno:0
```

```
tcam:B, bank:0, prot:0      Aces
```

```
Bridge                ip any any
```

Case Study #3 - QoS Programming with L4Ops

QoS policies might also reference L4Ops; these L4Ops must be installed like any other L4Op. This implies that per interface, even for your QoS policies, you are limited by the restrictions that capmap tables and LOUs inherently have. Here is an example to illustrate this in a small way:

```

Sup2T#show ip access-lists a1
Extended IP access list a1
 10 permit tcp host 192.168.1.10 host 192.168.2.10 dscp ef

```

```

Sup2T#show class-map a1-class
Class Map match-all a1-class (id 37)
Match access-group name a1

```

```

Sup2T#show policy-map a1-policy
Policy Map a1-policy
Class a1-class
  police cir 80000 bc 2500

```

```
conform-action transmit
exceed-action drop
```

This example has a policy-map matching a class-map that calls the access-list a1 which matches traffic from 192.168.1.10 to 192.168.2.10 that is marked with Expedited Forwarding (EF). Matching on a DSCP value is a non-expandable L4Op; this is required to be programmed into a LOU register and referenced via an entry in the capmap table. This policy-map is now installed inbound to gig3/23.

```
Sup2T#show run int gig3/23
Building configuration...
```

```
Current configuration : 176 bytes
!
interface GigabitEthernet3/23
  switchport
  switchport trunk allowed vlan 1-30
  switchport mode trunk
  service-policy input a1-policy
end
```

In order to look at the QoS programming for an interface, use this command:

```
Sup2T#show platform hardware acl entry interface gig3/23 qos in ip module 3
mls_if_index:8096000 dir:0 feature:1 proto:0
```

```
pass#0 features
```

```
fno:0
```

```
tcam:A, bank:0, prot:0 Aces
```

```
0x0000E0100000D00B tcp host 192.168.1.10 host 192.168.2.10 dscp eq 46
0x000000000080D00B ip any any
```

Detailing this command gives you what TCAM label is used for on this interface.

```
Sup2T#show platform hardware acl entry interface gig3/23 qos in ip detail module 3
mls_if_index:8096000 dir:0 feature:1 proto:0
```

```
pass#0 features
```

```
UAPRSF: U-urg, A-ack, P-psh, R-rst, S-syn, F-fin
MLGFI: M-mpls_plus_ip_pkt, L-L4_hdr_vld, G-gpid_present, F-global_fmt_match, I-ife/ofe
's' means set; 'u' means unset; '-' means don't care
```

```
-----
-----
I      INDEX LABEL FS ACOS AS IP_SA SRC_PORT
IP_DA          DST_PORT F FF L4PROT TCP-F:UAPRSF MLGFI OtherL4OPs
RSLT                      CNT
-----
-----
-----
```

```
fno:0
```

```
tcam:A, bank:0, prot:0 Aces
```

```
I V 16238 2 0 0 0 192.168.1.10 - 192.168.2.10
```



```

- 0 0      1      -      ----- dscp=46;                0x0000E0100000D00B
0
I M 16238 0x1FFF 0 0x00 0x000 255.255.255.255          - 255.255.255.255
- 0 0      0xF
I V 16239      2 0 0 0      0.0.0.0                -      0.0.0.0
- 0 0      0      -      -----                    -                0x000000000080D00B
0
I M 16239 0x1FFF 0 0x00 0x000      0.0.0.0          -      0.0.0.0
- 0 0      0x0

```

The TCAM label that is used is 2. Look at the capmap table for this now:

```
Sup2T#show platform software acl capmap tcam A label 2 module 3
```

```
Shadow Capmap Table Entry For TCAM A
```

```
-----
Output in a RST/INV/CNT format: RST - result value; INV - inverted;
                                CNT - aggregated reference account;
```

```
CBF - number of free cap bits (one per entry);
Free items are not shown
```

```
-----
```

Index	CBF	[9]	[8]	[7]	[6]
[5]		[4]	[3]	[2]	[1]
[0]					
2	8	212/0/1	Free	Free	Free
Free		Free	Free	Free	Free

```
-----
Free      2/1/1
```

Note: For QoS TCAM, you must specify the module number. Without this, the output does not yield any results.

```
Sup2T#show platform software acl capmap mapping
```

L4op_sel	value	Reference
0	-----	LOU0 B register
1	-----	LOU0 A register
2	-----	LOU1 B register
3	-----	LOU1 A register

```
*snip*
```

A LOU value of 2 points to LOU1, register B. You can confirm this programming with this command:

```
Sup2T#show platform hardware acl lou
```

```
Dumping h/w lou values
```

Index	lou_mux_sel	A_Opcode	A_Value	B_Opcode	B_Value
0	7	NEQ	0	NEQ	0
1	4	NEQ	0	NEQ	46
2	1	NEQ	0	NEQ	0

```
*snip*
```

Scale up the configuration.

```
Sup2T#show ip access-lists a1
```

```
Extended IP access list a1
```

```
10 permit tcp host 192.168.1.10 host 192.168.2.10 dscp ef
20 permit tcp host 192.168.2.11 host 192.168.2.11 dscp ef
```

```

30 permit tcp host 192.168.3.11 host 192.168.3.11 dscp ef
40 permit tcp host 192.168.4.11 host 192.168.4.11 dscp ef
50 permit tcp host 192.168.5.11 host 192.168.5.11 dscp ef
60 permit tcp host 192.168.6.11 host 192.168.6.11 dscp ef
70 permit tcp host 192.168.7.11 host 192.168.7.11 dscp ef
80 permit tcp host 192.168.8.11 host 192.168.8.11 dscp ef

```

```

Sup2T#show platform software acl capmap tcam A label 2 module 3
Shadow Capmap Table Entry For TCAM A

```

```

-----
Output in a RST/INV/CNT format: RST - result value; INV - inverted;
                               CNT - aggregated reference account;

```

```

CBF - number of free cap bits (one per entry);
Free items are not shown

```

```

-----
Index   CBF      [9]          [8]          [7]          [6]
[5]     [4]     [3]          [2]          [1]
[0]

```

```

-----
      2      8 212/0/1          Free          Free          Free
Free          Free          Free          Free          Free          Free  2/1/8

```

This does not use any more entries; instead, it increases the aggregate reference count against the first entry itself, which makes sense. From a capmap table and LOU register perspective, there is no concern about the source and the destination. This simply stores L4Op information. Since it matches against the same DSCP value on all ACEs, you only need one entry for that DSCP value.

Modify this so that you use nine different DSCP values.

```

Sup2T#show ip access-lists a1

```

```

Extended IP access list a1

```

```

10 permit tcp host 192.168.1.10 host 192.168.2.10 dscp af11
20 permit tcp host 192.168.2.11 host 192.168.2.11 dscp af12
30 permit tcp host 192.168.3.11 host 192.168.3.11 dscp af13
40 permit tcp host 192.168.4.11 host 192.168.4.11 dscp af21
50 permit tcp host 192.168.5.11 host 192.168.5.11 dscp af22
60 permit tcp host 192.168.6.11 host 192.168.6.11 dscp af23
70 permit tcp host 192.168.7.11 host 192.168.7.11 dscp af31
80 permit tcp host 192.168.8.11 host 192.168.8.11 dscp af32
90 permit tcp host 192.168.9.11 host 192.168.9.11 dscp af33

```

Now if you look at the capmap table, you see that it is full:

```

Sup2T#show platform software acl capmap tcam A label 2 module 3
Shadow Capmap Table Entry For TCAM A

```

```

-----
Output in a RST/INV/CNT format: RST - result value; INV - inverted;
                               CNT - aggregated reference account;

```

```

CBF - number of free cap bits (one per entry);
Free items are not shown

```

```

-----
Index   CBF      [9]          [8]          [7]          [6]
[5]     [4]     [3]          [2]          [1]
[1]     [0]

```

```

-----
-----
      2      0 212/0/1      10/1/1      9/1/1      8/1/1
7/1/1      6/1/1      5/1/1      4/1/1      3/1/1
2/1/1

```

Here is what happens if you try and install another non-expandable L4Op-based entry:

```

Sup2T(config-ext-nacl)#permit tcp host 192.168.10.11 host 192.168.10.11 dscp 2
Sup2T(config-ext-nacl)#end

```

```

%QM-4-TCAM_ENTRY: Hardware TCAM entry programming failed for slot 3 intf Gi3/23
dir IN: <CONFIG_UPDATE_REQ> TCAM Req Error: FAIL (4): Low TCAM Entries (1)
%QM-4-TCAM_ENTRY: Hardware TCAM entry programming failed for slot 3 intf Gi3/23
dir IN: <CONFIG_UPDATE_REQ> TCAM Req Error: FAIL (4): Low TCAM Entries (1)
%QM-4-TCAM_ENTRY: Hardware TCAM entry programming failed for slot 3 intf Gi3/23
dir IN: <CONFIG_UPDATE_REQ> TCAM Req Error: FAIL (4): Low TCAM Entries (1)
%QM-4-TCAM_ENTRY: Hardware TCAM entry programming failed for slot 3 intf Gi3/23
dir IN: <CONFIG_UPDATE_REQ> TCAM Req Error: FAIL (4): Low TCAM Entries (1)
%QM-4-TCAM_ENTRY: Hardware TCAM entry programming failed for slot 3 intf Gi3/23
dir IN: <CONFIG_UPDATE_REQ> TCAM Req Error: FAIL (4): Low TCAM Entries (1)
%QM-4-TCAM_ENTRY: Hardware TCAM entry programming failed for slot 3 intf Gi3/23
dir IN: <CONFIG_UPDATE_REQ> TCAM Req Error: FAIL (4): Low TCAM Entries (1)
%FMCORE-6-RACL_ENABLED: Interface GigabitEthernet3/23 routed traffic is hardware
switched in ingress direction

```

Oct 20 17:12:54.304: %EARL_CM-DFC3-5-NOCAPMAP: No free capmap entry available

Look at the TCAM for this interface now:

```

Sup2T#show platform hardware acl entry interface gig3/23 qos in ip module 3

```

```

mls_if_index:8096000 dir:0 feature:1 proto:0

```

```

Couldnt find feature for mls_if_index 0x8096000, dir 0

```

None of the QoS features are installed in the TCAM for this interface anymore.

Notice that marking does not consume any L4Ops. So if you have a simple ACL that does not have L4Ops and you set a DSCP value on match, no LOU registers are used for this. Here is an example:

```

Sup2T#show policy-map a1-policy
Policy Map a1-policy
  Class a1-class
    set dscp ef

```

```

Sup2T#show class-map a1-class
Class Map match-all a1-class (id 37)
  Match access-group name a1

```

```

Sup2T#show ip access-lists a1
Extended IP access list a1
  10 permit tcp host 192.168.1.1 host 192.168.2.1

```

This is applied to interface gig3/23:

```

Sup2T#show run interface gig3/23
Building configuration...

```

```

Current configuration : 176 bytes
!
interface GigabitEthernet3/23
  switchport
  switchport trunk allowed vlan 1-30
  switchport mode trunk
  service-policy input a1-policy

```


In this example, there is an interface configured to use both IPv4 and IPv6 ACLs that creates these errors when the interface is brought up:

```
%EARL_CM-5-NOCAPMAP: No free capmap entry available
%FMCORE-4-RACL_REDUCED: Interface Vlan500 routed traffic will be software switched in ingress
direction.
    L2 features may not be applied at the interface
```

However, if only the IPv4 ACL is removed, and then readded to the same interface the hardware programming completes successfully and the error is no longer generated.

For this example, these ACLs are configured under the SVI:

```
Switch#sh ip access-lists INGRESS
Extended IP access list INGRESS
 10 permit tcp host 1.1.1.1 host 1.1.1.2 range 1 10
 20 permit tcp host 1.1.1.3 host 1.1.1.4 range 10 ftp-data
 30 permit tcp host 2.1.1.3 host 2.1.1.4 range 30 40
 40 permit tcp host 2.1.1.3 host 2.1.1.4 range 85 100
 50 permit tcp host 2.1.1.3 host 10.1.1.1 range 222 333
 60 permit tcp host 20.5.4.3 host 10.100.100.1 range www 443
 70 permit tcp host 200.50.4.3 host 11.11.11.1 range 800 813
 80 permit tcp host 200.50.40.30 host 12.12.11.1 range 50 60
 90 permit tcp host 13.13.13.3 host 14.14.14.3 range gopher 90
100 permit tcp host 23.23.23.3 host 14.14.10.1 range 123 345
110 permit udp host 123.123.123.1 range 50 60 host 23.23.23.1 range 10 20
120 permit udp host 45.45.43.1 range 1000 1010 host 1.1.1.1 range 50 65
130 permit tcp host 78.78.78.1 range 89 95 host 2.3.4.5 range 1111 1200
140 permit tcp host 5.5.5.50 eq 65000 host 5.4.5.4
150 permit tcp host 5.15.5.150 range 1200 1250 host 1.7.8.4 range 45 65
```

```
Switch#show ipv6 access-list DENY-ALL-V6
IPv6 access list DENY-ALL-V6
  permit udp FE80::/64 host FF02::66 eq 2000 sequence 10
  deny ipv6 any any sequence 20
```

As seen in the previous example, the IPv4 ACL has more than nine unique expandable L4Ops. Under an interface configured with only IPv4 these will be expanded out as needed in order to not exhaust the capmap table.

When programming these into the TCAM hardware in a dual-stack environment, the switch starts with the IPv4 ACL first. With insufficient free entries in the capmap table, the switch expands out some of the expandable L4Ops in order to fill the capmap table without exceeding it. The result is that now the number of free entries in the table is 0, which means there are now no entries available to program the non-expandable L4Op that is required when you go to program the IPv6 ACL next.

When you remove only the IPv4 ACL, the number of free entries in the capmap table increases, and the IPv6 ACL is now properly programmed into hardware and uses one of the newly freed capmap entries. When the IPv4 ACL is reapplied to the interface configuration, the same expansion happens again. Only now one additional IPv4 entry is expanded as a result of the IPv6 ACL which uses up a free capmap value. Since all L4Ops are expandable in this ACL the programming succeeds.

In order to prevent the manual removal and addition of the IPv4 ACL to allow the entries to merge in hardware, an enhancement was created to change the TCAM merging algorithm in such scenarios. See Cisco bug ID [CSCuq24924](#) for more information.

As a result of this enhancement, "Fixed-in" releases of software will now have a configurable

option in the global configuration that changes the way L4Ops are programmed in instances such as a dual-stack IPv4/v6 ACL deployment. This is the configuration change that can be made

```
Switch(config)#platform hardware acl tcam-exp-logic enable
```

Note: Because of the changes introduced as a result of this enhancement, expandable L4Ops are expanded at a rate greater than normal and might cause a significant increase in TCAM utilization as a result of the expansion.