

AOS7 Troubleshooting Guide

Contents

1. Basics.....	4
1.1. Technical support log collector	4
1.2. Maintenance Shell	5
1.2.1. Entering Maintenance Shell if logged in as "admin"	5
1.2.2. allowRootAccess.....	5
1.2.3. BusyBox.....	6
1.2.4. tcpDump.....	9
1.2.5. Executing CLI commands.....	9
1.2.6. Cron.....	9
1.3. Accessing NIs and CMMs.....	10
1.3.1. Introduction.....	10
1.3.2. OmniSwitch 6900 and 6860.....	10
1.3.3. OmniSwitch 10K.....	10
1.3.4. Virtual Chassis	11
1.4. Licensing	11
1.4.1. Introduction.....	11
1.4.2. Demo license activation for AOS 7.3.1.R01	12
1.4.3. Demo license deactivation for AOS 7.3.1.R01	12
1.5. Simple Network Management Protocol	12
1.5.1. Checklist.....	12
1.5.2. Introduction.....	13
1.5.3. Minimum working configuration.....	13
1.5.4. Troubleshooting	13
1.5.5. Advanced Troubleshooting	14
1.5.6. Troubleshooting in Maintenance Shell	17
1.6. Ethernet Management Port.....	18
1.6.1. Introduction.....	18
1.6.2. Modifying the Primary or Secondary CMM's EMP Port IP Address	18
1.7. Subsystem Communication.....	18
1.7.1. Identifying the process	18
1.7.2. Backtrace logs	20
1.7.3. Additional logs.....	21
1.7.4. Workarounds	21
1.7.5. Generate PMD.....	21
1.8. High CPU Utilization	22
1.8.1. Checklist.....	22
1.8.2. Basic Troubleshooting.....	22
1.8.3. Advanced Troubleshooting	22
1.8.4. Troubleshooting in Maintenance Shell	24
1.8.5. Troubleshooting in bShell	Error! Bookmark not defined.
1.9. Memory Leak	24
1.9.1. Troubleshooting in Maintenance Shell	24
1.10. Packet Driver	26
1.10.1. Introduction.....	26
1.10.2. Packet Driver statistics.....	26
1.10.3. Protocol to CPU queue mapping.....	29
1.10.4. CPU queue traffic shaping	Error! Bookmark not defined.
1.10.5. Capturing packets trapped to CPU and send by CPU.....	30
1.10.6. Attributes used by the Packet Driver	30
1.10.7. opCodes.....	31
1.10.8. Examples.....	31

1.11.	Logging (SWLOG).....	33
1.12.	Virtual Chassis.....	33
1.12.1.	Checklist.....	33
1.12.2.	Introduction.....	33
1.12.3.	Configuration.....	39
1.12.4.	Basic Troubleshooting.....	41
1.12.5.	Advanced Troubleshooting.....	43
1.12.6.	Troubleshooting in Maintenance Shell.....	48
2.	Layer 1 Troubleshooting.....	49
2.1.	Physical Interfaces.....	49
2.1.1.	Checklist.....	49
2.1.2.	Introduction.....	49
2.1.3.	Basic troubleshooting.....	49
2.1.4.	Advanced troubleshooting.....	50
2.1.5.	Troubleshooting in Maintenance Shell.....	51
2.2.	Quality of Service.....	55
2.2.1.	Checklist.....	55
2.2.2.	Introduction.....	55
2.2.3.	Minimum working configuration.....	74
2.2.4.	Troubleshooting.....	74
2.2.5.	Advanced troubleshooting.....	74
2.2.6.	Troubleshooting in Maintenance Shell.....	77
2.3.	Datacenter Bridging.....	78
2.3.1.	Minimum working configuration.....	81
2.3.2.	Troubleshooting in Maintenance Shell.....	81
2.4.	Port monitoring.....	82
2.5.	Sflow.....	82
2.5.1.	Minimum working configuration.....	82
2.5.2.	Advanced troubleshooting.....	82
2.5.3.	Troubleshooting in Maintenance Shell.....	82
2.6.	OpenFlow.....	83
2.7.	Checklist.....	83
2.8.	Introduction.....	83
2.8.1.	FloodLight and AVIOR example.....	84
2.9.	Minimum working configuration.....	85
2.10.	Basic Troubleshooting.....	85
2.11.	Advanced Troubleshooting.....	86
2.12.	Troubleshooting in Maintenance Shell.....	86
2.13.	Troubleshooting in bShell.....	Error! Bookmark not defined.
3.	Layer 2 Troubleshooting.....	88
3.1.	Ethernet Ring Protection.....	88
3.1.1.	Checklist.....	88
3.1.2.	Introduction.....	88
3.1.3.	Minimum working configuration.....	90
3.1.4.	Basic troubleshooting.....	92
3.1.5.	Advanced troubleshooting.....	92
3.2.	Troubleshooting in Maintenance Shell.....	92
3.3.	IP Multicast Switching.....	92
3.3.1.	Checklist.....	92
3.3.2.	Introduction.....	93
3.3.3.	Minimum working configuration.....	101
3.3.4.	Basic troubleshooting.....	101

3.3.5.	Advanced troubleshooting	105
3.3.6.	Troubleshooting in Maintenance Shell	106
3.4.	Link Aggregation Control Protocol	106
3.4.1.	Checklist.....	106
3.4.2.	Introduction.....	107
3.4.3.	Support of 256 aggregates and up to 16 ports.....	108
3.4.4.	Minimum working configuration.....	108
3.4.5.	Basic Troubleshooting.....	109
3.4.6.	Advanced Troubleshooting	109
3.4.7.	Troubleshooting in Maintenance Shell	110
3.4.8.	Verifyng LACP BPDU drop on software level	112
3.4.9.	Troubleshooting in bShell	Error! Bookmark not defined.
3.5.	Source Learning	113
3.5.1.	Checklist.....	113
3.5.2.	Introduction.....	113
3.5.3.	Minimum working configuration.....	114
3.5.4.	Basic Troubleshooting.....	114
3.5.5.	Advanced Troubleshooting	114
3.6.	Troubleshooting in Maintenance Shell.....	114
3.7.	Spanning Tree Protocol	116
3.7.1.	Checklist.....	116
3.7.2.	Introduction.....	116
3.7.3.	Minimum working configuration.....	117
3.7.4.	Troubleshooting	117
3.7.5.	Advanced Troubleshooting	118
3.8.	Dynamic Automatic Fabric	118
4.	Introduction.....	118
5.	Further reading	119
5.1.	Application Fingerprinting	119
5.1.1.	Checklist.....	119
5.1.2.	Introduction.....	119
5.2.	Link Layer Distribution Protocol	122

1. Basics

1.1. Technical support log collector

The technical support log collector is a script which gathers minimum logs required to open a Service Request as well as other logs helpful in troubleshooting. The concept of the script is based on this thread. The script is compatible with OS6900 and OS10K in standalone and virtual chassis mode.

1. Copy the script into /flash

2. Go to Maintenance Shell and add executable flag to the script



Note: You need to be logged as "admin" to access the Maintenance Shell

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> chmod u+x /flash/snapshot.sh
```

3. To display available options use the "-h" parameter.

```
TOR #-> /flash/snapshot.sh -h
AOS 7&8 technical support log collector version 1.4 GA
-h display this help
-u do not compress the archive
-v verbose output
```



Note: By default the script creates a compressed archive. If CPU utilization on the primary CMM is high, please use "-u" option in the next step.

4. Verify free space left on the flash. If there are no PMDs on the flash, the compressed archive should be smaller than 1 MB. If there are multiple PMDs on the flash, then the output file size might be bigger than 10 MB.

```
-> show system | grep "Available (bytes)"
    Available (bytes): 1159639040,
```



Note: Make sure that there is sufficient space left on the flash before executing the script. It is recommended to have at least 20 MB free space left on the flash.

5. Execute the script:

```
TOR #-> /flash/snapshot.sh
AOS 7&8 technical support log collector version 1.5 GA
Please wait.....
Generated /flash/snapshot_none_20130827233031.tar.gz
```



Note: In case of Virtual Chassis configuration it's sufficient to run the script on the primary CMM of the primary chassis. The only exception is for Virtual Chassis in "split" mode, in this case it is required to execute the script on primary CMMs on all Virtual Chassis members.



Note: Gathering and compressing logs and PMD files may take a few minutes. It is not recommended to stop the script before the end.




Note: Archive creation and compression is CPU intensive. It is expected to observe CPU utilization at around 50%.

6. Send the generated output to Technical Support

1.2. Maintenance Shell

1.2.1. Entering Maintenance Shell if logged in as "admin"

Use the "su" command to enter the Maintenance Shell on the primary CMM.

 **Warning:** Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #->
```

1.2.2. allowRootAccess

The "root" login is similar to logging in as admin and running the "su" command, but with an important difference: if the switch has failed to come up completely, you can usually still log in as "root". Even if Session Manager and AAA were not started, you can still log in as root. On the other hand, we don't want to ship a wide-open switch, so here's what we're doing (it's checked in now, part of build 7.1.1.1590.R01):

New behavior of "root" login:

1. You must be on the console to log in as "root"
2. You must have previously enabled root login:
 - you must have installed the jumper on the CMM that lets you interrupt the boot process countdown and drop down into the u-boot shell
 - you must have run the u-boot command "allowRootAccess" on the CMM

The "su" command behavior is unchanged: any administrator (a user having AAA/read-write permission) can run su.

Example screen captures (enabling one-shot "root" login on the console on 10K):

```
-> reload switch from working no rollback-timeout
...
nvram baudrate is 9600
nvram netmask is 255.0.0.0
nvram ipaddr is 10.255.24.81
nvram gatewayip is 10.255.24.181
nvram current is working
nvram next is certified
cf type is EXT2
Boot jumper installed, 2 sec bootdelay
Hit any key to stop autoboot: 2..1..
```

Hit a key before it counts to zero.. If it goes to zero immediately, you need to install the u-boot jumper.

To use the switch rescue procedure first make sure of the following:

- The Rrescue.img file is located in the root directory of the USB flash device
- The 10000 directory contains all the system files required to revive the switch. i.e. 10000/certified/<*.img boot.cfg> on the USB flash device
- USB flash device is inserted into the CMM

Once the above requirements are met, issue the following command at the prompt to start the rescue process:

```
=> run rescue
***** Using the rescue process WILL destroy all switch data *****
*****                               USE WITH EXTREME CAUTION                               *****
=> allowRootAccess
=> boot
...
```

Example of on-shot of "root" login permission on OS 10K:

```
OS10 login: root
RUSHMORE # exit
OS10 login: root
root access not enabled
OS10 login:
```

Example of a cheap way of "root" login permission on OS 10K:

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
RUSHMORE #-> echo 1 >/proc/nvram/allowRootAccess
RUSHMORE #-> exit
logout
-> exit
Changes in the running configuration have been made but not saved!
Confirm exit (Y/N)? y
logout
OS10 login: root
RUSHMORE # exit
OS10 login: root
root access not enabled
OS10 login:
```

1.2.3. BusyBox

This is set of basic Unix commands available in Maintenance Shell including:

Print lines containing one of strings

```
-> show log swlog slot 2/1 | grep -E "Chassis 2 role|/bin/busybox|Split-Topology"
```

Report process status

```
TOR #-> ps
PID  USER      COMMAND
  1  root      init
  2  root      [kthreadd]
  3  root      [migration/0]
  4  root      [ksoftirqd/0]
  5  root      [watchdog/0]
```

Send a signal (TERM by default) to given PIDs

```
TOR #-> kill 2081
```

Temporarily suspend a process, and then resume its execution at a later time

```
TOR #-> kill -STOP 2081
TOR #-> kill -CONT 2081
```

Send a signal to process(es) selected by regex PATTERN

```
TOR #-> pkill aluSubagent
```

Restart the "Session Manager" which will ask e.g. vsftpd to re-read the /etc/vsftpd.conf (if options have been changed manually, like #-> echo "chmod_enable=NO" >> /etc/vsftpd.conf")

```
TOR #-> kill 'pidof sesmgrCmm'
TOR #->
Wed Nov 27 08:55:48 : ChassisSupervisor appMgr alert message:
+++ appMgrClientTerminated: restarting task
+++ Failed App /bin/maybe-electric-fence /bin/sesmgrCmm
```

Restart the "AAA Manager" which will ask e.g. vsftpd to use a manually changed userTable5 (e.g. if it has been replaced via FTP (binary transfer))

```
TOR #-> kill 'pidof aaaCmm'
TOR #->
Wed Nov 27 11:57:16 : ChassisSupervisor appMgr alert message:
+++ appMgrClientTerminated: restarting task
+++ Failed App /bin/maybe-electric-fence /bin/aaaCmm
```

Pause for a time equal to the total of the args given, where each arg can have an optional suffix of (s)econds, (m)inutes, (h)ours, or (d)ays

```
TOR #-> sleep 1
```

Print last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.

```
TOR #-> ps | tail -n 10
```

Print first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.

```
TOR #-> ps | head -n 10
```

Print or control the kernel ring buffer

```
TOR #-> dmesg | tail
[ 40.534711] svc: failed to register lockdvl RPC service (errno 97).
[ 40.537917] rpc.nfsd used greatest stack depth: 5040 bytes left
[ 42.371768] PHY: mdio@ff726520:07 - Link is Up - 1000/Full
[ 42.372093] ADDRCONF(NETDEV_CHANGE): eth3: link becomes ready
[ 43.834142] mcipc: module license 'Proprietary' taints kernel.
[ 52.742731] alv4if_itf_add: ifindex 4172 vrf 0 restarted 0
[ 52.992121] eth3: no IPv6 routers present
[ 60.189806] ssapp: sending ioctl 30d to a partition!
[ 60.189823] ssapp: sending ioctl 30d to a partition!
[61883.968076] aluSubagent used greatest stack depth: 4816 bytes left
```


Configure a network interface

```
TOR #-> ifconfig
alv4qmr  Link encap:Ethernet  HWaddr 00:00:00:00:00:00
         inet addr:127.2.255.1  Bcast:127.2.255.255  Mask:255.255.255.0
         UP BROADCAST RUNNING MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth3     Link encap:Ethernet  HWaddr E8:E7:32:07:98:78
         inet addr:172.26.60.135  Bcast:172.26.60.255  Mask:255.255.255.0
         inet6 addr: fe80::eae7:32ff:fe07:9878/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:654740 errors:0 dropped:0 overruns:0 frame:0
         TX packets:3612 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:51374045 (48.9 MiB)  TX bytes:472812 (461.7 KiB)
         Base address:0x2000

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.255.255.255
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNING MTU:2048  Metric:1
         RX packets:5153130 errors:0 dropped:0 overruns:0 frame:0
         TX packets:5153130 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:2762686856 (2.5 GiB)  TX bytes:2762686856 (2.5 GiB)

lo:1    Link encap:Local Loopback
         inet addr:127.2.65.1  Mask:255.255.0.0
         UP LOOPBACK RUNNING MTU:2048  Metric:1

lo:2    Link encap:Local Loopback
         inet addr:127.2.1.1  Mask:255.255.0.0
         UP LOOPBACK RUNNING MTU:2048  Metric:1
```

List all open ports using lsof (List Open Files)

```
#-> lsof -n -i -P
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE  NODE  NAME
tftpd    726  root   3u  IPv4  6915    0     UDP  127.3.1.1:69
rpcbind  733  root   6u  IPv4  6924    0     UDP  *:111
rpcbind  733  root   7u  IPv4  6927    0     UDP  *:908
rpcbind  733  root   8u  IPv4  6928    0     TCP  *:111 (LISTEN)
rpc.statd 738  root   4u  IPv4  6948    0     UDP  127.0.0.1:914
rpc.statd 738  root   6u  IPv4  6963    0     UDP  *:39846
rpc.statd 738  root   7u  IPv4  6965    0     TCP  *:37146 (LISTEN)
rpc.mount 759  root   6u  IPv4  921     0     UDP  *:33627
rpc.mount 759  root   7u  IPv4  924     0     TCP  *:34264 (LISTEN)
swlogd   816  root   5u  IPv4  973     0     UDP  *:21500
pmdd     827  root   4u  IPv4  6976    0     TCP  127.0.0.1:10200 (LISTEN)
pmdd     827  root   5u  IPv4  6977    0     TCP  127.0.0.1:10203 (LISTEN)
pmdd     827  root   8u  IPv4  7002    0     TCP  127.0.0.1:10203-
>127.0.0.1:36161 (ESTABLISHED)
pmdd     827  root   9u  IPv4  7019    0     TCP  127.0.0.1:10200-
>127.0.0.1:39098 (ESTABLISHED)
```

1.2.4. tcpDump

```
TOR #-> tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth3, link-type EN10MB (Ethernet), capture size 96 bytes
19:25:47.652434 IP 135.117.67.37.1946 > 172.26.60.135.23: . ack 618859809 win
15662
19:25:47.654424 IP 172.26.60.135.23 > 135.117.67.37.1946: P 1:148(147) ack 0 win
183
19:25:47.723883 e8:e7:32:3f:de:ff (oui Unknown) > 01:80:c2:00:00:31 (oui
Unknown), ethertype Unknown (0x8902), length 97:
    0x0000:  2001 0446 0002 4427 0001 0120 0d4d 4131  ...F..D'.....MA1
    0x0010:  3030 3030 3030 3030 3031 0000 0000 0000  00000000001.....
    0x0020:  0000 0000 0000 0000 0000 0000 0000 0000  .....
    0x0030:  0000 0000 0000 0000 0000 0000 0000 0000  .....
    0x0040:  0000 0000 0000 0000 0000 0200 0102 0400  .....
    0x0050:  0101                                     ...
```

1.2.5. Executing CLI commands

CLI commands can be executed using pipe to jail:

```
TOR #-> echo "show system" | jail
System:
  Description:  Alcatel-Lucent OS6900-X20 7.3.2.344.R01 GA, June 12, 2013.,
  Object ID:   1.3.6.1.4.1.6486.801.1.1.2.1.10.1.1,
  Up Time:    0 days 0 hours 16 minutes and 35 seconds,
  Contact:    Alcatel-Lucent, http://alcatel-lucent.com/wps/portal/enterprise,
  Name:       none,
  Location:   Unknown,
  Services:   78,
  Date & Time: TUE AUG 20 2013 20:09:06 (UTC)
Flash Space:
  Primary CMM:
    Available (bytes): 1262612480,
    Comments           : None
```

1.2.6. Cron

An example which gathers following logs every 1 minute:

- date and time
- first 9 lines of the "top" output
- the "show vlan" output from CLI

Execute following commands in the Maintenance Shell:

```
mkdir /var/spool
mkdir /var/spool/cron
mkdir /var/spool/cron/crontabs
echo "0-59/1 * * * * /flash/logs.sh" > /var/spool/cron/crontabs/root
echo "date >> /flash/logs.txt" > /flash/logs.sh
echo "top -b | head -n 9 >> /flash/logs.txt" >> /flash/logs.sh
echo 'echo "show vlan" | jail >> /flash/cron.txt' >> /flash/logs.sh
chmod a+rx /flash/logs.sh
crond -l 0
```

To stop cron:

```
pkill crond
```

The /var/spool/cron/crontabs directory is deleted after a reboot.

1.3. Accessing NIs and CMMs

1.3.1. Introduction

All NIs and CMMs can be accessed from the Maintenance Shell using Telnet and SSH. SSH access is preferred.

1.3.2. OmniSwitch 6900 and 6860

In case of a standalone OS6900 and OS6860 there is only 1 module available. When entering the Maintenance Shell user gets connected to this module. The complete list of aliases is available in the "/etc/hosts" file.

```
#-> cat /etc/hosts
127.0.0.1 localhost
::1      ip6-localhost ip6-loopback

127.2.65.1      CMMA

127.2.1.1      NI1
```

1.3.3. OmniSwitch 10K

In case of a standalone OS10K there is multiple modules available. When entering the Maintenance Shell user gets connected to the Primary CMM. The complete list of aliases is available in the "/etc/hosts" file.

```
#-> cat /etc/hosts
127.0.0.1 localhost
::1      ip6-localhost ip6-loopback
127.2.65.1      CMMA
127.2.66.1      CMMB
127.2.1.1      NI1
127.2.2.1      NI2
127.2.3.1      NI3
127.2.4.1      NI4
127.2.5.1      NI5
127.2.6.1      NI6
127.2.7.1      NI7
127.2.8.1      NI8
127.2.66.1      otherCMM
127.2.65.1      PrimaryCMM
127.2.66.1      BackupCMM
```

To access NIs use login "root" and password "switch". An example:



Note: SSH access is preferred than telnet

```
#-> ssh 127.2.1.1
The authenticity of host '127.2.1.1 (127.2.1.1)' can't be established.
RSA key fingerprint is 36:39:3d:99:52:62:45:c8:7f:b8:d3:c6:a3:37:2c:c2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '127.2.1.1' (RSA) to the list of known hosts.
#
```

To access CMMs use login "admin" and password "switch". An example:



Note: SSH access is preferred than telnet

```
#-> ssh admin@127.2.1.65
```

Use "StrictHostKeychecking=no" to skip the key verification:

```
#-> ssh -o StrictHostKeychecking=no 127.2.1.1
Warning: Permanently added '127.2.1.1' (RSA) to the list of known hosts.
#
```

To access NIs use login "root" and password "switch". An example:

```
#-> telnet 127.2.1.1
Entering character mode
Escape character is '^]'.
(none) login: root
Password:
# exit
Connection closed by foreign host
#-> telnet nil
Entering character mode
Escape character is '^]'.
(none) login: root
Password:
#
```

1.3.4. Virtual Chassis

It is also possible to access modules in a Virtual Chassis using following addressed:

- 127.10.<chassis-id>.65 - CMMA
- 127.10.<chassis-id>.66 - CMMB
- 127.10.<chassis-id>.<slot-id> - NIs

To access NIs use login "root" and password "switch". An example:

```
#-> ssh root@127.10.2.1
```

To access CMMs use login "admin" and password "switch". An example

```
#-> ssh admin@127.10.2.65
```

1.4. Licensing

1.4.1. Introduction

Some features require a software license and are restricted only to a licensed user. Purchasing a license along with an authorization code from Alcatel-Lucent is required. The authorization code is then used to generate a license file.

The features below require the associated license:

- Advanced License – Required to support SPB, Virtual Chassis, BGP, MP-BGP, OSPFv2/v3, ECMP for OSPF, PBR, RIPng, Static Routing IPv6, VRRP/VRRPv3, DVMRP, PIM-SM/PIM-DM, PIM-SM IPv6, IPSEC, VRF
- Data Center License – Required for Data Center Bridging Protocols (PFC,ETS,DCBX) and EVB
- U16L License – Required to upgrade OS10K-XNI-U16L to a 16x10G line card



Note:

Advanced and Data Center licensed features are not supported over MC-LAG



Note: A reboot is required for any licensed features to take effect
 In Virtual Chassis configuration licenses for all units need to be present in a license file when executing the "license apply" command



1.4.2. Demo license activation for AOS 7.3.1.R01

A demo license for the demonstration or the testing purpose can be activated for 5 days. Please also note that a switch must be rebooted to enable or disable a license. By default there are no licenses installed.

```
-> show license-info
Time (Days)
VC device License Type Remaining
-----+-----+-----+-----+-----+-----
```

Use the "debug demo-license" command to add licenses

```
-> debug demo-license
Please wait.set demo license OK
Fri Feb 1 15:05:45 : vc_licManager licMgr info message:
+++ !!! system reboot required. !!!
```

After rebooting there are 2 or 3 new licenses installed:

```
-> show license-info

VC      device  License                               Type          Time (Days)
-----+-----+-----+-----+-----+-----
0       0          Advanced                             DEMO          5
0       0          Data-Center                           DEMO          5
```



Warning: The switch will reboot automatically after the 5 days are up.

1.4.3. Demo license deactivation for AOS 7.3.1.R01

Licenses can be de-activated using the "debug license deactivate all" command.

```
-> debug license deactivate all
Fri Feb 1 14:58:04 : vc_licManager licMgr info message:
+++ Remove all licenses
Please wait.license 0 deactivated OK
Fri Feb 1 14:58:09 : vc_licManager licMgr info message:
+++ !!! License clear done, system reboot required !!!
```

1.5. Simple Network Management Protocol

1.5.1. Checklist

- At least one NMS station needs to be reachable via ICMP packets from the switch, otherwise no traps will be sent to any SNMP station (to be verified - AOS 7 & 8 requires only ARP entry of the next hop to send traps)
- SNMPv2
 - the "snmpOutGenErrs" error counter in "show snmp statistics" increments if the user associated to SNMP community name is missing
 - the "snmpInBadCommunityNames" error counter in "show snmp statistics"

increments if SNMP get with incorrect community name is received or "snmp community-map mode enable" is missing or the "snmp community-map <community> user <user> enable" is missing

- SNMPv3
 - the "usmStatsWrongDigests" error counter in "show snmp statistics" increments if the user password on the NMS station is incorrect
 - the "usmStatsUnsupportedSecLevels" error counter in "show snmp statistics" increments if the user used for authentication is configured without the "md5+des" keyword
 - the "usmStatsUnknownUserNames" error counter in "show snmp statistics" increments if an incorrect username is configured on the NMS station
 - the "usmStatsNotInTimeWindows" error counter in "show snmp statistics" increments if clock difference between the SNMP client and server is exceeding 150 seconds (FRC 3414)

1.5.2. Introduction

1.5.3. Minimum working configuration

Minimum working configuration for SNMPv2:

```
no user snmp
user snmp password switch123 read-write all no auth
aaa authentication snmp local
snmp security no-security
snmp community-map mode enable
snmp community-map public user snmp enable
snmp station 1.1.1.1 snmp v2 enable
```



Note:

The user "snmp" must be created before enabling the SNMP station (in AOS 7.3.2.R01 this is not required). Otherwise it is necessary to recreate the SNMP station or reload the switch.

Minimum working configuration for SNMPv3:

```
no user snmp3
user snmp3 password switch1234 md5+des read-write all
aaa authentication snmp local
snmp station 1.1.1.1 snmp3 v3 enable
```



Note:

The user "snmp3" must be created before enabling the SNMP station (in AOS 7.3.2.R01 this is not required). Otherwise it is necessary to recreate the SNMP station or reload the switch.

1.5.4. Troubleshooting

Use ping to verify SNMP Station reachability, example:

```
-> ping 1.1.1.1
```

Monitor SNMP statistics

```
-> show snmp statistics
From RFC1907
  snmpInPkts                = 2
  snmpOutPkts               = 1
  snmpInBadVersions         = 0
  snmpInBadCommunityNames   = 0
  snmpInBadCommunityUses    = 0
  snmpInASNParseErrs        = 0
  snmpEnableAuthenTraps     = disabled(2)
  snmpSilentDrop             = 0
  snmpProxyDrops             = 0
  snmpInTooBig               = 0
  snmpInNoSuchNames         = 0
  snmpInBadValues           = 0
  snmpInReadOnlys           = 0
  snmpInGenErrs              = 0
  snmpInTotalReqVars         = 1
  snmpInTotalSetVars         = 0
  snmpInGetRequests          = 1
  snmpInGetNexts             = 0
  snmpInSetRequests          = 0
  snmpInGetResponses         = 0
  snmpInTraps                = 0
  snmpOutTooBig              = 0
  snmpOutNoSuchNames        = 0
  snmpOutBadValues           = 0
  snmpOutGenErrs             = 0
  snmpOutGetRequests         = 0
  snmpOutGetNexts           = 0
  snmpOutSetRequests         = 0
  snmpOutGetResponses        = 1
  snmpOutTraps               = 0
From RFC2572
  snmpUnknownSecurityModels  = 0
  snmpInvalidMsgs            = 0
  snmpUnknownPDUHandlers     = 0
From RFC2573
  snmpUnavailableContexts    = 0
  snmpUnknownContexts        = 0
From RFC2574
  usmStatsUnsupportedSecLevels = 0
  usmStatsNotInTimeWindows    = 0
  usmStatsUnknownUserNames    = 1
  usmStatsUnknownEngineIDs    = 1
  usmStatsWrongDigests        = 0
  usmStatsDecryptionErrors    = 0
```

Display swlog output for Trap Manager, this example output indicates that there is a user missing:

```
-> show log swlog | grep trapmgr
... swlogd: trapmgr trapmgr_main error(2) Trap user snmp is not valid -- please
add AAA user and reconfigure trap station
```

1.5.5. Advanced Troubleshooting

Simulating traps



Note:

The "debug ip packet" output is available only for non-EMP ports



Note:

Use different trap number in "debug trap generate" each time the command is executed

```

-> show configuration snapshot snmp
! Trap Manager:
snmp station 192.168.10.2 162 "snmp" v2 enable

! SNMP:
snmp security no-security
snmp community-map mode enable
snmp community-map "public" user "snmp" enable

-> swlog appid trapmgr subapp all level debug3

-> show snmp statistics | grep snmpOutTraps
snmpOutTraps                = 0

-> debug ip packet ip-address 192.168.10.2 start timeout 60

-> debug trap generate 0
Simulated trapId 0 : expected

0 C S 1/12 e8e732:000042->0007e9:1ff567 IP 192.168.10.1->192.168.10.2 UDP
36170,162
0 1 R CMM0 e8e732:000042->0007e9:1ff567 IP 192.168.10.1->192.168.10.2 UDP
36170,162
0 1 S 1/12 e8e732:000042->0007e9:1ff567 IP 192.168.10.1->192.168.10.2 UDP
36170,162
0 1 R 1/12 0007e9:1ff567->e8e732:000042 IP 192.168.10.2->192.168.10.1 ICMP 3,3
0 1 S CMM0 0007e9:1ff567->e8e732:000042 IP 192.168.10.2->192.168.10.1 ICMP 3,3
0 C R 1/12 0007e9:1ff567->e8e732:000042 IP 192.168.10.2->192.168.10.1 ICMP port
unreachable

-> show snmp statistics | grep snmpOutTraps
snmpOutTraps                = 1

-> swlog appid trapmgr subapp all level info

-> show log swlog | grep trapmgr
... swlogd: trapmgr trapmgr_main debug1(6) trap_user_author | user snmp 0 0
... swlogd: trapmgr trapmgr_main debug1(6) trap_user_author | user snmp status =
1
... swlogd: trapmgr trapmgr_main debug1(6) trap_user_author | user entry is
active 0 0

```

In case the SNMP station is behind the EMP port use "tcpdump" instead of "debug ip packet", an example:

```

-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> tcpdump port snmptrap
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth3, link-type EN10MB (Ethernet), capture size 96 bytes
04:11:23.884530 IP 172.26.60.192.39154 > 172.26.60.242.162: V2Trap(54)
system.sysUpTime.0=8408085 S:1[|snmp]

```

Enabling debugging for SNMP Agent

Increasing logging level for SNMP Agent:

```

-> swlog appid snmp subapp all level debug3

```


On NMS or the localhost in Maintenance Shell there is the "snmpget -v 2c -c <community> <ip> sysName.0" command executed, example of output logged in SWLOG on the SNMP Agent:

```
-> show log swlog | grep SNMP
... swlogd: SNMP aluSubagent_main debug3(8) snmp_get_current_user | User is snmp
... swlogd: SNMP aluSubagent_main debug3(8) snmp_find_user | user = snmp is in
local DB
... swlogd: SNMP aluSubagent_main debug3(8) snmp_get_current_user | checking
User snmp privs for this access
... swlogd: SNMP aluSubagent_main debug3(8) snmp_get_current_user | checking
User snmp read-write for this access
... swlogd: SNMP aluSubagent_main debug3(8) snmp_get_current_user | checking
User snmp type for this access
... swlogd: SNMP aluSubagent_main debug3(8) mip_get | request vrf =
... swlogd: SNMP aluSubagent_main debug3(8) getTblNObjNums | before mip_from_oid
- oidlen = 9
... swlogd: SNMP aluSubagent_main debug3(8) getTblNObjNums | check for index 0
in this oid
... swlogd: SNMP aluSubagent_main debug3(8) getTblNObjNums | mip_from_oid failed
mip-table-id 0 mip-object-id 0 ERROR: -1
... swlogd: SNMP aluSubagent_main debug3(8) getTblNObjNums | try again with
different index
... swlogd: SNMP aluSubagent_main debug3(8) getTblNObjNums | check for index 5
in this oid
... swlogd: SNMP aluSubagent_main debug3(8) getTblNObjNums | mipTblNum=65566
mipObjNum=5
... swlogd: SNMP aluSubagent_main debug3(8) getTblNObjNums | indexlen = 1
... swlogd: SNMP aluSubagent_main debug3(8) get_dMIBObject | mipTblNum 65566
mipObjNum 5
```

Finally reduce debug level:

```
-> swlog appid snmp subapp all level info
```

Switch is not sending traps

Observation to confirm the issue state:

- Switch is not sending traps
- Swlogs throws the following error messages:

```
swlogd: trapmgr trapmgr_main error(2) Failed to store trap alaDoSTrap generated
on Sun Jan 18 10:53:44 2015
swlogd: trapmgr trapmgr_main error(2) Unable to open trap database: [unable to
open database file]
```

- The snmpouttraps is not getting incremented in "show snmp statistics" output for the traps sent out.

How to recreate the issue?

Each process in Linux is bound to maximum file descriptor. Execution of "snmp-trap replay-ip <ipv4 address>" paves the way to leave the socket "fd" remains opened. This in turn causes the maximum number of FDs to be reached in long running. Due to this, we do not get an 'fd' to open the trap database in order to save the traps.

How to check the total number of FDs in use by trapMgr?


This issue could be confirmed by getting the following output from the switch which is in issue state (following commands needs to executed in Maintenance Shell):

```
TOR #-> lsof -c trapmgr | wc -l
16384
TOR #-> cd /proc/$(pidof trapmgr)/fd
TOR #-> ls -l | wc -l
16384
```

Max limit of open files per process in linux:

```
TOR #-> cat /proc/$(pidof trapmgr)/limits
Limit                Soft Limit           Hard Limit           Units
...
Max open files       16384               16384               files
...
```

1.5.6. Troubleshooting in Maintenance Shell

 **Warning:** Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.


Verifying SNMPv2 configuration from a switch

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> snmpget -v 2c -c public localhost sysName.0
SNMPv2-MIB::sysName.0 = STRING: (none)
```

Verifying SNMPv3 configuration from a switch

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> snmpget -v3 -u snmp3 -l authPriv -a MD5 -A switch1234 -x DES -X
switch1234 localhost sysName.0
SNMPv2-MIB::sysName.0 = STRING: (none)
```

Restarting SNMP Agent

 **Warning:** If the SNMP Agent is killed twice the system will reboot. Please verify if the process name is not followed by "restarted" using "ps

To kill the SNMP agent From Maintenance Shell:

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> pkill aluSubagent
```

Restarting Trap Manager

 **Warning:** If the Trap Manager is killed twice the system will reboot. Please verify if the process name is not followed by "restarted" using "ps

To kill the SNMP agent From Maintenance Shell:

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> pkill trapmgr
```

To identify the source of SNMP messages for EMP port (capture 10 packets):

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> tcpdump -A -i eth3 -c 10 port 161
```

1.6. Ethernet Management Port

1.6.1. Introduction

3 different addresses are suggested in order to avoid any potential issues. The shared IP address is only active on the primary CMM. The secondary would only have a single IP address, the one configured in NVRAM. It is only on a takeover that the shared address is configured on the new primary. The shared address is meant to be used as the management address. Regardless of which CMM is primary, the switch can be reached via this address. Having unique NVRAM addresses allows a user to explicitly connect to the desired CMM.

Display EMP configuration in the NVRAM from Maintenance Shell:

```
cat /proc/nvram/empgateway
cat /proc/nvram/empipaddr
cat /proc/nvram/empnetmask
```

Remove EMP configuration from the NVRAM:



Note: A reload is required to apply changes

```
echo "0.0.0.0" > /proc/nvram/empipaddr
echo "0.0.0.0" > /proc/nvram/empnetmask
echo "0.0.0.0" > /proc/nvram/empgateway
```

Change EMP configuration on the NVRAM:



Note: A reload is required to apply changes

```
echo "172.26.60.135" > /proc/nvram/empipaddr
echo "255.255.255.0" > /proc/nvram/empnetmask
echo "172.26.60.254" > /proc/nvram/empgateway
```

1.6.2. Modifying the Primary or Secondary CMM's EMP Port IP Address

Must be connected to the associated CMM's console port before attempting to change IP address information using the modify boot parameters command as shown below:

```
-> modify boot parameters
Boot > boot empipaddr 198.51.100.2
Boot > boot empmasklength 16
Boot > show

EMP IP Address: 198.51.100.2/16
(additional table output not shown)

Boot > commit system
Boot > commit boot
Boot > exit
```

1.7. Subsystem Communication

1.7.1. Identifying the process

Subsystem communication issues are related to response when executing "show" commands - there is "Please wait....." displayed on the console. The first step in troubleshooting communication issues between subsystems is to identify the process causing issues. In most cases it is sufficient to focus only on the CMM part. In some cases the "top" output may give a clue which process is causing issues. See below a few examples.

CLI command	Related processes
show configuration snapshot webmgt	webMgtd
show configuration snapshot vrrp	vrrp
show configuration snapshot vlan	vmCmm
show configuration snapshot virtual-chassis	vcmCmm
show configuration snapshot vfc	vfcM
show configuration snapshot udld	udldCmm
show configuration snapshot system	ssapp
show configuration snapshot svcmgr	svcCmm
show configuration snapshot stp	stpCmm
show configuration snapshot spb-isis	isis -spb
show configuration snapshot snmp	aluSubagent
show configuration snapshot slb	slbcmmd
show configuration snapshot sip	Feature not supported in this platform
show configuration snapshot session	sesmgrCmm
show configuration snapshot saa	saaCmm
show configuration snapshot ripng	ripng
show configuration snapshot rip	rip
show configuration snapshot qos	qoscmmd
show configuration snapshot port-mapping	pmCmm
show configuration snapshot policy	To be determined
show configuration snapshot pmm	pmmcmmd
show configuration snapshot ospf3	ospf3
show configuration snapshot ospf	ospf
show configuration snapshot openflow	To be determined
show configuration snapshot ntp	ntpd
show configuration snapshot netsec	To be determined
show configuration snapshot mvrp	mvrpCmm
show configuration snapshot multi-chassis	mcmCmm
show configuration snapshot module	To be determined
show configuration snapshot message-service	To be determined
show configuration snapshot lldp	lldpCmm
show configuration snapshot linkagg	lagCmm
show configuration snapshot link-fault-propagation	To be determined
show configuration snapshot ldp	To be determined
show configuration snapshot isis	isis
show configuration snapshot ipv6	ip6cmmd
show configuration snapshot ipsec	ipsecSysd ipsec6d
show configuration snapshot ipms	ipmscmm
show configuration snapshot ipmr	To be determined
show configuration snapshot ip-routing	To be determined
show configuration snapshot ip-helper	To be determined
show configuration snapshot ip	ipcmmd

show configuration snapshot interface	etherCmm
show configuration snapshot health	hmonCmm
show configuration snapshot ha-vlan	haVlanCmm
show configuration snapshot fcoe	To be determined
show configuration snapshot evb	evbCmm
show configuration snapshot ethernet-oam	eoamCmm
show configuration snapshot erp	erpCmm
show configuration snapshot dpi	
show configuration snapshot dhlaa	dhlCmm
show configuration snapshot dhcpv6-server	dhcpv6srv
show configuration snapshot dhcp-server	dhcpsrv
show configuration snapshot dhcpv6-relay	dhcp6rd
show configuration snapshot da-unp	To be determined
show configuration snapshot chassis	To be determined
show configuration snapshot capability	capmanc
show configuration snapshot bridge	slCmm
show configuration snapshot bgp	bgp
show configuration snapshot bfd	bfd
show configuration snapshot auto-fabric	dafcCmm
show configuration snapshot app-fingererprint	appfpCmm
show configuration snapshot active-lease-service	To be determined
show configuration snapshot aaa	aaaCmm
show ethernet-service nni	vstkCmm
write memory flash-synchro	new_cs and flashMgr

Another way to identify the process, which is causing issues is to increade logging level for "configmanager":

```
swlog appid configmanager subapp all level debug3
```

Execute "show configuration snapshot", wait at least 1 minute, then "^C". Next execute "write memory", wait at least 1 minute, then "^C". Gather logs (look for the last application, which failed to return output):

```
show log swlog | grep ConfigManager
```

Reduce logging level:

```
swlog appid configmanager subapp all level info
```

1.7.2. Backtrace logs

Backtrace is a source-level debugging tool which gives a summary how a program or task got where it is. So it is useful if you need to know what a task is currently doing and how it got there. This tool is useful when troubleshooting "Please wait..." issues.

Print backtrace of all stack frames, or innermost COUNT frames. With a negative argument, print outermost -COUNT frames. Without the 'full' qualifier there are no values of the local variables printed.

```
TOR #-> debug $(pidof <process_name>) "bt full"
TOR #-> debug $(pidof <process_name>) "bt full"
TOR #-> debug $(pidof <process_name>) "bt full"
```

Example output:

```
RUSHMORE #-> debug $(pidof slCmm) "bt full"
```

```
warning: Could not load shared library symbols for linux-vdso32.so.1.
Do you need "set solib-search-path" or "set sysroot"?
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/libthread_db.so.1".
0x0fa80188 in __epoll_wait_nocancel () from /lib/tls/libc.so.6
#0 0x0fa80188 in __epoll_wait_nocancel () from /lib/tls/libc.so.6
No symbol table info available.
#1 0xfec7558 in reacWait (p_reactor=0x1008d6d8 <reac>, p_timeout=0x0) at
/space/7.3.2.R01-relman/sw/ipc/reactor//ipc_reactor.c:2788
    wait = 359
    end = {tv_sec = 1846882, tv_usec = 764319}
    temp_tv = {tv_sec = 0, tv_usec = 359188}
    ret = 0
    reac_debug = 0
    our_timeout = {tv_sec = 0, tv_usec = 0}
    p_cond = <optimized out>
    newa = {__sigaction_handler = {sa_handler = 0x1, sa_sigaction = 0x1},
sa_mask = {__val = {0 <repeats 32 times>}}, sa_flags = 0, sa_restorer = 0x0}
    olda = {__sigaction_handler = {sa_handler = 0x1, sa_sigaction = 0x1},
sa_mask = {__val = {0 <repeats 32 times>}}, sa_flags = 0, sa_restorer = 0x0}
    __FUNCTION__ = "reacWait"
#2 0x100136b0 in main (argc=<optimized out>, argv=<optimized out>) at
/space/7.3.2.R01-relman/sw/bridging/source_learning/cmm/slc_main.cpp:130
    txTimer = {tv_sec = 0, tv_usec = 500000}
    hkTimer = {tv_sec = 1, tv_usec = 0}
    __FUNCTION__ = "main"
```



Note: The "reacWait" state is the idle state

1.7.3. Additional logs

In all cases console output is needed.

In case of the "qoscmd" process being blocked please take additional output:

```
show qos log
```

1.7.4. Workarounds

- Proceed with "vc-takeover"
- Proceed with "takeover"

1.7.5. Generate PMD



Warning: Killing a single process twice forces the system to reboot

Send SIGABRT signal to the affected process to generate PMD.

```
kill -6 <pid>
```

The PMD files are created on the /flash/pmd directory.

1.8. High CPU Utilization

1.8.1. Checklist

- Use "top" to identify the process consuming CPU resources
- CPU resources may be consumed by Packet Driver, see the Packet Driver article
- 100% CPU utilization may be observed after enabling high level of debugging

1.8.2. Basic Troubleshooting

Use the "show health" command.

```
-> show health
CMM                Current      1 Min      1 Hr      1 Day
Resources          Avg         Avg         Avg         Avg
-----+-----+-----+-----+-----
CPU                4          3          3          3
Memory            41         41         41         41
```

And the "show health all cpu" command to identify the module with high CPU utilization.

```
-> show health all cpu
CPU                Current      1 Min      1 Hr      1 Day
                  Avg         Avg         Avg         Avg
-----+-----+-----+-----+-----
Slot 1/ 1          6          6          6          5
Slot 1/ 2          5          6          5          5
Slot 1/ 8          12         11         11         11
Slot 2/ 1          12         11         10         10
Slot 2/ 8          12         12         11         11
```

1.8.3. Advanced Troubleshooting

It's worth to verify how many packets are trapped to CPU due to FFP rules (packets may be trapped to CPU due to other reasons too, see Packet Driver for more details):



Note: In a non-VC configuration chassis id doesn't need to be specified: debug qos internal "slot 1 list 1 verbose"

```
-> debug qos internal "chassis 1 slot 1 list 1 verbose"
Entry U Slice CIDU CIDL MIDU MIDL TCAM Count[+]
Green[+] Red[+] NotGreen[+]
List 1: 19 entries set up
McastARP( 17) 0 8 1543 - - - 1636 0[0 ]
0[0 ] 0[0 ]
McastARP( 17) 0 9 - - - - 1892 0[0 ]
0[0 ] 0[0 ]
ISIS_BPDU1( 22) 0 8 - 1536 - - 1590 0[0 ]
0[0 ] 0[0 ]
ISIS_BPDU1( 22) 0 9 - - - - 1846 0[0 ]
0[0 ] 0[0 ]
ISIS_BPDU2( 23) 0 8 1537 - - - 1591 0[0 ]
0[0 ] 0[0 ]
ISIS_BPDU2( 23) 0 9 - - - - 1847 0[0 ]
0[0 ] 0[0 ]
ISIS_BPDU3( 24) 0 8 - 1538 - - 1592 0[0 ]
0[0 ] 0[0 ]
ISIS_BPDU3( 24) 0 9 - - - - 1848 0[0 ]
0[0 ] 0[0 ]
IPMS_IGMP( 50) 0 8 1539 - - - 1638 86834[86834 ]
0[0 ] 0[0 ]
IPMS_IGMP( 50) 0 9 - - - - 1894 86834[0 ]
```

```

0[0 ] 0[0 ] 0[0
IPMS_V4Control( 51) 0 8 - 1540 - - 1639 45196[45196 ]
0[0 ] 0[0 ] 0[0
IPMS_V4Control( 51) 0 9 - - - - 1895 45196[0 ]
0[0 ] 0[0 ] 0[0
IPMS_V4Data( 52) 0 8 1541 - - - 1640 0[0 ]
0[0 ] 0[0 ] 0[0
IPMS_V4Data( 52) 0 9 - - - - 1896 0[0 ]
0[0 ] 0[0 ] 0[0
IPMS_V4Resolved( 53) 0 8 - 1542 - - 1641 0[0 ]
0[0 ] 0[0 ] 0[0
IPMS_V4Resolved( 53) 0 9 - - - - 1897 0[0 ]
0[0 ] 0[0 ] 0[0
ETHOAM_SYS( 62) 0 8 - 1562 - - 1741 0[0 ]
0[0 ] 0[0 ] 0[0
ETHOAM_SYS( 62) 0 9 - - - - 1997 0[0 ]
0[0 ] 0[0 ] 0[0
802.1ab Regular( 102) 0 8 1547 - - - 1586 1551[1551 ]
0[0 ] 0[0 ] 0[0
802.1ab Regular( 102) 0 9 - - - - 1842 1551[0 ]
0[0 ] 0[0 ] 0[0
amap Regular( 103) 0 8 - 1546 - - 1587 0[0 ]
0[0 ] 0[0 ] 0[0
amap Regular( 103) 0 9 - - - - 1843 0[0 ]
0[0 ] 0[0 ] 0[0
802.3ad Regular( 104) 0 8 - 1548 - - 1588 0[0 ]
0[0 ] 0[0 ] 0[0
802.3ad Regular( 104) 0 9 - - - - 1844 0[0 ]
0[0 ] 0[0 ] 0[0
802.1x Regular( 105) 0 8 1549 - - - 1589 0[0 ]
0[0 ] 0[0 ] 0[0
802.1x Regular( 105) 0 9 - - - - 1845 0[0 ]
0[0 ] 0[0 ] 0[0
BPDU Regular( 106) 0 8 - 1550 - - 1584 0[0 ]
0[0 ] 0[0 ] 0[0
BPDU Regular( 106) 0 9 - - - - 1840 0[0 ]
0[0 ] 0[0 ] 0[0
MPLS( 120) 0 8 - 1544 - - 1543 0[0 ]
0[0 ] 0[0 ] 0[0
MPLS( 120) 0 9 - - - - 1799 0[0 ]
0[0 ] 0[0 ] 0[0
srcsldrop( 128) 0 8 1551 - - - 1536 0[0 ]
0[0 ] 0[0 ] 0[0
srcsldrop( 128) 0 9 - - - - 1792 0[0 ]
0[0 ] 0[0 ] 0[0
Static Mac Move( 131) 0 8 1553 1552 0 1 1745 0[0 ]
0[0 ] 0[0 ] 0[0
Static Mac Move( 131) 0 9 - - - - 2001 0[0 ]
0[0 ] 0[0 ] 0[0
MIM( 132) 0 8 1545 - - - 1537 0[0 ]
0[0 ] 0[0 ] 0[0
MIM( 132) 0 9 - - - - 1793 0[0 ]
0[0 ] 0[0 ] 0[0
LINKOAMSAA( 150) 0 8 1561 - - - 1742 0[0 ]
0[0 ] 0[0 ] 0[0
LINKOAMSAA( 150) 0 9 - - - - 1998 0[0 ]
0[0 ] 0[0 ] 0[0

```

End...

1.8.4. Troubleshooting in Maintenance Shell



Warning:

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

Use the "top" command in the Maintenance Shell to continuously monitor CPU utilization:

```
TOR #-> top
Mem: 930840K used, 1140748K free, 0K shrd, 268K buff, 473148K cached
CPU:  1.4% usr  1.6% sys  0.0% nic 96.8% idle  0.0% io  0.0% irq  0.0% sirq
Load average: 0.00 0.00 0.00 1/190 10611
  PID  PPID  USER      STAT   VSZ  %VSZ  CPU  %CPU  COMMAND
 2136  1278  root       S       171m  8.4   0   2.3  /bin/bcmd -p
  1278     1  root       S    65940  3.1   0   0.1  /bin/new_cs --logLevel=5
10609 10581  root       R     3236  0.1   0   0.1  top
 2244  1278  root       S     122m  6.0   1   0.0  /bin/qosnid
 2109  1278  root       S    43704  2.1   1   0.0  /bin/vrrp
 1874  1278  root       S    40884  1.9   1   0.0  /bin/lagCmm
 2201  1278  root       S    32108  1.5   0   0.0  /bin/etherNi
...
```

Press "1" to display per core CPU utilization:

```
Mem: 1114824K used, 860004K free, 0K shrd, 72K buff, 574872K cached
CPU0:  0.0% usr 20.0% sys  0.0% nic 80.0% idle  0.0% io  0.0% irq  0.0% sirq
CPU1:  0.0% usr  0.0% sys  0.0% nic 100% idle  0.0% io  0.0% irq  0.0% sirq
Load average: 1.57 0.80 0.40 1/211 5710
  PID  PPID  USER      STAT   VSZ  %VSZ  CPU  %CPU  COMMAND
 5687  5684  root       R     3832  0.1   0  11.5  top
 5286  2291  root       S    19176  0.9   0   3.8  {sshd} sshd: admin@pts/0
 2521  1507  root       S     304m 15.7   1   0.0  /bin/bcmd -p
 2613  1507  root       S     282m 14.6   0   0.0  /bin/slNi
...
```



Note: Press "q" to exit

1.9. Memory Leak

1.9.1. Troubleshooting in Maintenance Shell



Warning:

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

Global information:

```
SHASTA #-> cat /proc/meminfo
MemTotal:      2021900 kB
MemFree:       826728 kB
Buffers:       556 kB
Cached:       751116 kB
SwapCached:    0 kB
Active:       426432 kB
Inactive:     643104 kB
Active(anon) : 324904 kB
Inactive(anon): 112872 kB
```

```

Active(file):      101528 kB
Inactive(file):   530232 kB
Unevictable:      0 kB
Mlocked:          0 kB
HighTotal:        1269760 kB
HighFree:         198648 kB
LowTotal:         752140 kB
LowFree:          628080 kB
SwapTotal:        0 kB
SwapFree:         0 kB
Dirty:            0 kB
Writeback:        0 kB
AnonPages:        317864 kB
Mapped:           787928 kB
Shmem:            119912 kB
Slab:             35404 kB
SReclaimable:    17356 kB
SUnreclaim:      18048 kB
KernelStack:     1944 kB
PageTables:       22148 kB
NFS_Unstable:    0 kB
Bounce:          0 kB
WritebackTmp:    0 kB
CommitLimit:     1010948 kB
Committed_AS:    1026696 kB
VmallocTotal:    245760 kB
VmallocUsed:      11884 kB
VmallocChunk:    225212 kB

```

Per process information:

```

-> su
TOR #-> top -b -n 1 -m | head
Mem total:1974820 anon:461752 map:93076 free:866644
slab:35764 buf:68 cache:575868 dirty:4 write:0
Swap total:0 free:0
  PID  VSZ^VSZRW  RSS (SHR) DIRTY (SHR) STACK COMMAND
2538  162m  136m  92992 10440 92984 10432   132 /bin/bcmd -p
2479  95256  83160 11432  7988 11424  7980   132 /bin/dhcpsrv
2467  95108  83160 11404  7988 11396  7980   132 /bin/dhcpv6srv
2651  43868  24976 35972 10416 35928 10404   132 /bin/ipmsni
1510  41844  28876 13988  7144 13980  7136   132 /bin/new_cs --logLevel=5
2628  36020  15900 20244 10672 20216 10660   132 /bin/slNi

```

Based on the above output take additional logs of the process that has high memory usage (5 times):

```
TOR #-> debug $(pidof <process name>) "bt full"
```

An example:

```
TOR #-> debug $(pidof ipmsni) "bt full"
debug $(pidof ipmsni) "bt full"
```

```

warning: Could not load shared library symbols for linux-vdso32.so.1.
Do you need "set solib-search-path" or "set sysroot"?
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/libthread_db.so.1".
0x0ea46188 in __epoll_wait_nocancel () from /lib/tls/libc.so.6
#0  0x0ea46188 in __epoll_wait_nocancel () from /lib/tls/libc.so.6
No symbol table info available.
#1  0x0fde3214 in reacWait (p_reactor=0x10096538 <ipmsni_reactor>, p_timeout=0x0)

```

```

at /space/7.3.3.R01-relman/sw/ipc/reactor
... //ipc_reactor.c:2791
    wait = 1712
    end = {tv_sec = 2685, tv_usec = 658444}
    temp_tv = {tv_sec = 1, tv_usec = 712998}
    ret = 0
    reac_debug = 0
    our_timeout = {tv_sec = 0, tv_usec = 0}
    p_cond = <optimized out>
    newa = {__sigaction_handler = {sa_handler = 0x1, sa_sigaction = 0x1},
sa_mask = {__val = {0 <repeats 32 times>}}, sa_flags = 0,
... sa_restorer = 0x0}
    olda = {__sigaction_handler = {sa_handler = 0x1, sa_sigaction = 0x1},
sa_mask = {__val = {0, 0, 3220312547, 0, 0, 244736208,
... 3220312560, 3220311544, 1208136864, 991, 0, 0, 3220313024, 1208161108,
246095416, 32, 0, 0, 1, 0, 0, 0, 2, 126, 3220312928, 91,
... 269051248, 269181104, 3220312960, 266575540, 115, 1208136872}}, sa_flags = 0,
sa_restorer = 0x0}
    __FUNCTION__ = "reacWait"
#2 0x10023420 in ipms::nic::loop () at /space/7.3.3.R01-
relman/sw/multicast/ip/nic/src/main.cpp:3683
No locals.
#3 0x100314e0 in main (argc=1, argv=0xbff21814) at /space/7.3.3.R01-
relman/sw/multicast/ip/nic/src/main.cpp:3790
    rv = <optimized out>
    background = 1
    tv = {tv_sec = 2, tv_usec = 0}
    long_opts = {{name = 0x1006e9e4 "restarted", has_arg = 0, flag = 0x0, val
= 0}, {name = 0x0, has_arg = 0, flag = 0x0, val = 0}}
    __PRETTY_FUNCTION__ = "int main(int, char**)"
    __FUNCTION__ = "main"
    opt_index = 0

```

1.10. Packet Driver

1.10.1. Introduction

Packet Driver in AOS 7 is an equivalent of qDispatcher in AOS 6. This software module is responsible for filtering and classification of all frames and packets which are trapped to CPU.

1.10.2. Packet Driver statistics

Packet Driver statistics can be displayed from Maintenance Shell on any NI. The second column gives information since the previous execution.

```

-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> cat /proc/pktdrv

```

```

Wakeups                :    439244    439244
Last Task Wakeup      :         0         0
Last Task Ran         : 1bbb88cc 1bbb88cc
Task Latency          : 1bbb88cc 1bbb88cc
Task Max Latency      : 1bbb88cc 1bbb88cc
TX Chain Ints         : 1097166 1097166
TX Timeout Ints       :         0         0
TX Timeout Retrys     :         0         0
TX Hangs              :         0         0
TX DMA Complete u:0:  min: 227 avg: 259 max: 1761
TX DMA Overrun u:0:   start: 0 over: 0 end: 0
TX unaligned          :         0         0
RX Chain Ints         :    9163    9163

```

RX Desc Ints	:	425383	425383
RX Timeout Ints	:	9164	9164
RX Desc Enabled	:	1	
RX Empty Wakeups	:	0	0
RX Partial Fill	:	0	0
RX Stalls	:	9163	9163
RX Restarts	:	9163	9163
RX Short DMA chain	:	0	0
DMA bad state	:	0	0
DMA Last State	:	0	0
HW Transmitted	:	1097166	1097166
HW XMIT Overrun	:	0	0
Unexpected Tx Int	:	0	0
TX Not OK to DMA	:	0	0
TX Failed Starts	:	0	0
Dirty Free Pool	:	0	0
2 Desc To Same Buf	:	0	0
Task Overrun	:	0	0
Interprocess Overrun:	:	0	0
Interprocess Pkts	:	0	0
Buffers Depleted	:	0	0
Classified 0	:	237350	237350
Classified 14	:	86603	86603
Classified 24	:	41621	41621
Classified 26	:	60146	60146
Classified 28	:	4957	4957
Freelist Corrupt	:	0	0
Tx Invalid Cos	:	0	0
Tx Invalid Port	:	0	0
Tx Invalid Modid	:	0	0
Tx Supersized	:	0	0
Spin discards	:	0	0
Zero Dst Mac	:	0	0
Zero Src Mac	:	0	0
Overlay2	:	0	0
Bad Reason Code	:	0	0
Bad DCB	:	0	0
Tx Using Dummy	:	0	0
Rx Using Dummy	:	0	0
CQ Bad Clients	:	0	0
IQ Bad Clients	:	0	0
Free Descriptors	:	3636	0

Buffer States

Invalid	:	1
Free	:	3636
Classify	:	0
RX Dma	:	78
TX Dma	:	0
Task	:	36

Buffer Owners

None	:	3636
Ipv4	:	1
Arp	:	1
Ip4t	:	1
Ip4h	:	1
Ip4s	:	1
Ip4n	:	1
Ip4m	:	1
Ip4v	:	1
Ip4r	:	1
Ip4o	:	1
Ip4g	:	1

```

Ip4p           : 1
Ip4b           : 1
Ip4P           : 1
Ip4e           : 1
Ipv6           : 1
Ipmc           : 1
Test           : 1
Ipms           : 1
Gvrp           : 1
Stp            : 1
Slrn           : 1
Lagg           : 1
Sflw           : 1
Pmon           : 1
Lldp           : 1
Erp            : 1
Bfd            : 1
Ulld           : 1
Qos            : 1
Dani           : 1
Mipc           : 1
Isis           : 1
Eoam           : 1
Loam           : 1
Ecp            : 1
Hrdw           : 78
Pd             : 1

```

Task Info

```

Ipv4: [flags=1] [id 1] [ring size 100] [head 72] [tail 72]
Arp : [flags=1] [id 2] [ring size 256] [head 83] [tail 83]
Ip4t: [flags=1] [id 3] [ring size 100] [head 0] [tail 0]
Ip4h: [flags=1] [id 4] [ring size 100] [head 0] [tail 0]
Ip4s: [flags=1] [id 5] [ring size 100] [head 0] [tail 0]
Ip4n: [flags=1] [id 6] [ring size 100] [head 0] [tail 0]
Ip4m: [flags=1] [id 7] [ring size 512] [head 0] [tail 0]
Ip4v: [flags=1] [id 8] [ring size 256] [head 0] [tail 0]
Ip4r: [flags=1] [id 9] [ring size 256] [head 0] [tail 0]
Ip4o: [flags=1] [id 10] [ring size 256] [head 0] [tail 0]
Ip4g: [flags=1] [id 11] [ring size 256] [head 0] [tail 0]
Ip4p: [flags=1] [id 12] [ring size 100] [head 0] [tail 0]
Ip4b: [flags=1] [id 13] [ring size 256] [head 0] [tail 0]
Ip4P: [flags=1] [id 14] [ring size 512] [head 0] [tail 0]
Ip4e: [flags=1] [id 15] [ring size 100] [head 0] [tail 0]
Ipv6: [flags=1] [id 16] [ring size 256] [head 75] [tail 75]
Ipmc: [flags=1] [id 17] [ring size 512] [head 64] [tail 64]
Test: [flags=0] [id 0] [ring size 2] [head 0] [tail 0]
Ipms: [flags=1] [id 19] [ring size 256] [head 0] [tail 0]
Tst1: d78384b8 Not connected
Tst2: d78388fc Not connected
Gvrp: [flags=1] [id 22] [ring size 100] [head 0] [tail 0]
Stp : [flags=1] [id 23] [ring size 512] [head 0] [tail 0]
Slrn: [flags=1] [id 24] [ring size 100] [head 0] [tail 0]
Lagg: [flags=1] [id 25] [ring size 256] [head 0] [tail 0]
Sflw: [flags=1] [id 26] [ring size 100] [head 0] [tail 0]
Pmon: [flags=1] [id 27] [ring size 100] [head 0] [tail 0]
Bcd4: d783a6d8 Not connected
Bcd6: d783abl c Not connected
Lldp: [flags=1] [id 30] [ring size 100] [head 46] [tail 46]
Erp : [flags=1] [id 31] [ring size 100] [head 0] [tail 0]
Mcm : d783b7e8 Not connected
Bcmd: d783bc2c Not connected
Bfd : [flags=1] [id 34] [ring size 256] [head 0] [tail 0]
Ulld: [flags=1] [id 35] [ring size 100] [head 0] [tail 0]

```

```

Qos : [flags=1] [id 36] [ring size 100] [head 0] [tail 0]
Fab : d783cd3c Not connected
Dani: [flags=1] [id 38] [ring size 100] [head 0] [tail 0]
Mipc: [flags=1] [id 39] [ring size 256] [head 0] [tail 0]
Isis: [flags=1] [id 40] [ring size 128] [head 0] [tail 0]
Eoam: [flags=1] [id 41] [ring size 256] [head 0] [tail 0]
Loam: [flags=1] [id 42] [ring size 256] [head 0] [tail 0]
Pktr: d783e6d4 Not connected
Ecp : [flags=1] [id 44] [ring size 512] [head 0] [tail 0]
Hardware Buffers
RX  Unit 0 Ch 1  [Active Head 21 Tail 4 ] [Inactive Head 0 Tail 47]
TX  Unit 0 Ch 0  [Active Head 21 Tail 21] [Inactive Head 9 Tail 9 ]
Classify Queues
Ipv4: Rx          :    107472    107472
Arp : Rx          :      4947      4947
Ipv6: Rx          :      84811      84811
Ipmc: Tx          :      84811      84811
Ipmc: Rx          :    258112    258112
Test: Tx          :         8         8
Stp : Tx          :    914391    914391
Lldp: Tx          :    182767    182767
Lldp: Rx          :     60146     60146
Disc: Rx          :         0         0
Client Mask = 0x17dccfcffffe
Debug flags = 0 (0x0), Count = -1., Grep is off

```

1.10.3. Protocol to CPU queue mapping

The number of packets per CPU queues can be monitored using "grep Classified".

```

TOR #-> cat /proc/pktdrv | grep Classified
Classified 0      :    237350      0
Classified 14     :     86603      0
Classified 24     :     41625      4
Classified 26     :     60163     17
Classified 28     :     4957       0

```

Major protocols are listed in the table below.

CPU queue	Traffic pattern
0	224.0.0.0/24, FF02::/32, Broadcasts, FTP, HTTPS, Default packet priority
1	Default packet priority
2	Default packet priority
3	Default packet priority
4	Default packet priority
5	Default packet priority
6	Default packet priority
7	Default packet priority
8	Source learning for UNP
9	Application Fingerprinting
10	Application Fingerprinting
11	Application Fingerprinting
12	Application Fingerprinting
13	Application Fingerprinting
14	IGMP, MLD

15	Unknown IPv4 and IPv6 multicast source when IPMS is enabled
16	PIM
17	Drop queue
18	BGP
19	BFD, UDLD
20	IPC, Telnet
21	SSH
22	HTTP
23	SNMP
24	DHCP
25	sFlow, QoS logging
26	STP, LLDP, AMAP, EAPOL, 802.1x
27	LACP, OAM
28	ARP
29	RIP, OSPF, ISIS
30	VRRP
31	ICMP, TTL=1, L2 move
32	Port mirroring
33	OpenFlow control

1.10.4. Capturing packets trapped to CPU and send by CPU



Warning:

Capturing packets using Packet Driver may be CPU intensive and dangerous to system stability. By default the packet counter is set to infinity (packets are logged continuously). Before executing any commands make sure that the packet counter is set to 0 - "echo n0 > /proc/pktdrv".

The first step before capturing packets is to set the packet counter to 0:

```
#-> echo n0 > /proc/pktdrv
```

Next specify debug level (see more details in the next section, use "echo g > /proc/pktdrv" to remove any filter):

```
#-> echo d1 > /proc/pktdrv
```

Set the packet counter to the desired number of packets (use "echo N > /proc/pktdrv" to remove any limit):

```
#-> echo n5 > /proc/pktdrv
```

Display results:

```
#-> dmesg | grep pd | tail -n 5
```

There is also a possibility to filter packets (see more details in the next section).

1.10.5. Attributes used by the Packet Driver

- **n<n>** – number of packets to be captured after enabling packet capture, by default n is equal

infinity, this attribute must be set before capturing anything

- **g<pattern>** – grep in the output only packets matching the pattern, the pattern may include keywords Ipv4, Ipv6, Arp, Stp, Lagg, Lldp, Mcm, Udp, Tcp, etc.
- **G<pattern>** – grep in the output only packets not matching the pattern, the pattern may include keywords Ipv4, Ipv6, Arp, Stp, Lagg, Lldp, Mcm, Udp, Tcp, etc.
- **d<bitmap>** – dump level
 - 0x0001 one line description of received packets
 - 0x0002 one line description of transmitted packets
 - 0x0004 include packet header
 - 0x0008 include hex dump
 - 0x0010 include dump of dcb
 - 0x0400 include list of clients to deliver to
 - 0x0800 include buffer number

1.10.6. opCodes

Each packet captured by the Packet Driver has a "reason" field which explains why it was copied to CPU.

1.10.7. Examples

Display Packet Driver settings

```
#-> tail -1 /proc/pktdrv
Debug flags = 0 (0x0), Count = 0., Grep is off
```

Capture 3 incoming ARP messages and display output



Warning:

Capturing packets using Packet Driver may be CPU intensive and dangerous to system stability. By default the packet counter is set to infinity (packets are logged continuously). Before executing any commands make sure that the packet counter is set to 0 - "echo n0 > /proc/pktdrv".

```
#-> echo gArp > /proc/pktdrv
#-> echo n3 > /proc/pktdrv
#-> echo d1 > /proc/pktdrv
#-> dmesg | grep pd | tail -n 5
[563696.203948] pd_dbg_cnt=3.
[563698.223952] pd_dbg=0x1, 1. grep is on including Arp count=3.
[563712.956897] pd: Hrdw rx 1/1 v:1 Arp Request 192.168.0.11->192.168.0.55 q:28
->Arp
[563713.052862] pd: Hrdw rx 1/1 v:1 Arp Request 192.168.0.11->192.168.0.55 q:28
->Arp
[563713.148798] pd: Hrdw rx 1/1 v:1 Arp Request 192.168.0.11->192.168.0.55 q:28
->Arp
```

Capture 3 incoming messages with their packet header and display output



Warning:

Capturing packets using Packet Driver may be CPU intensive and dangerous to system stability. By default the packet counter is set to infinity (packets are logged continuously). Before executing any commands make sure that the packet counter is set to 0 - "echo n0 > /proc/pktdrv".

```
#-> echo g"" > /proc/pktdrv
#-> echo n3 > /proc/pktdrv
#-> echo d5 > /proc/pktdrv
```



```
#-> dmesg | tail -n 26
[ 1038.007104] pd_dbg_cnt=3.
[ 1038.038426] pd: Hrdw rx 1/1 v:1 Arp Request 192.168.0.11->192.168.0.55 q:28
r:4000->Arp
[ 1038.139584] dst=2. src=39. packet_offset=512. l2_offset=512.
[ 1038.211516] l3_offset=530. packet_len=64. modid=0. port=1. vlan=1.
[ 1038.289679] vlan_pri=0. ivlan=0. ivlan_pri=0. cos=28. mgid=0.
[ 1038.362638] txsrcmodid=0. txsrcport=0. reason=0x4000 flags=0
[ 1038.434552] userflags=0 txtype=0. dcctype=2.
[ 1038.489797] qos_flags=0 qos_cookie=0 qos_ipgateway=00000000
[ 1038.560693] src_slot=0. src_port=0. ether_type=806 bufnum=2909.
[ 1038.635790] pd: Hrdw rx 1/1 v:1 Igmp 192.168.10.254->224.0.0.1 q:0 r:4800-
>Ipmc
[ 1038.729598] dst=16. src=39. packet_offset=512. l2_offset=512.
[ 1038.802557] l3_offset=530. packet_len=64. modid=0. port=1. vlan=1.
[ 1038.880732] vlan_pri=0. ivlan=0. ivlan_pri=0. cos=0. mgid=0.
[ 1038.952648] txsrcmodid=0. txsrcport=0. reason=0x4800 flags=0
[ 1039.024556] userflags=0 txtype=0. dcctype=2.
[ 1039.079839] qos_flags=0 qos_cookie=0 qos_ipgateway=00000000
[ 1039.150712] src_slot=0. src_port=0. ether_type=800 bufnum=1105.
[ 1039.225851] pd: Hrdw rx 1/1 v:1 Arp Request 192.168.0.11->192.168.0.55 q:28
r:4000->Arp
[ 1039.326934] dst=2. src=39. packet_offset=512. l2_offset=512.
[ 1039.398849] l3_offset=530. packet_len=64. modid=0. port=1. vlan=1.
[ 1039.477048] vlan_pri=0. ivlan=0. ivlan_pri=0. cos=28. mgid=0.
[ 1039.550002] txsrcmodid=0. txsrcport=0. reason=0x4000 flags=0
[ 1039.621908] userflags=0 txtype=0. dcctype=2.
[ 1039.677164] qos_flags=0 qos_cookie=0 qos_ipgateway=00000000
[ 1039.748039] src_slot=0. src_port=0. ether_type=806 bufnum=445.
[ 1042.187184] pd_dbg=0x5, 5. grep is off count=0.
```

Capture 1 incoming messages with its packet header and HEX dump and display output



Warning:

Capturing packets using Packet Driver may be CPU intensive and dangerous to system stability. By default the packet counter is set to infinity (packets are logged continuously). Before executing any commands make sure that the packet counter is set to 0 - "echo n0 > /proc/pktdrv".

```
#-> echo d0x0d > /proc/pktdrv
#-> echo n1 > /proc/pktdrv
#-> dmesg | tail -n 14
[883725.048936] pd_dbg=0xd, 13. grep is off count=0.
[883728.688939] pd_dbg_cnt=1.
[883729.589327] pd: Hrdw rx 1/1 v:1 Lldp 0180c20000e 001b21bd3809 q:26
r:1000->Lldp
[883729.684196] dst=30. src=45. packet_offset=512. l2_offset=512.
[883729.758209] l3_offset=530. packet_len=114. unit=0. modid=0. port=1.
[883729.838468] vlan=1. vlan_pri=0. ivlan=0. ivlan_pri=0. cos=26. mgid=0.
[883729.920808] txsrcmodid=0. txsrcport=0. reason=0x1000 flags=0
[883729.993795] userflags=0 txtype=0. dcctype=2.
[883730.050135] qos_flags=0 qos_cookie=0 qos_ipgateway=00000000
[883730.122073] src_slot=0. src_port=0. ether_type=88cc bufnum=1676.
[883730.199227] c2fa8200: 01 80 c2 00 00 0e 00 1b 21 bd 38 09 81 00 00 01
[883730.284695] c2fa8210: 88 cc 02 07 04 00 1b 21 bd 38 09 04 07 03 00 1b
[883730.370176] c2fa8220: 21 bd 38 09 06 02 00 78 fe 19 00 80 c2 09 80 01
[883730.455661] c2fa8230: 23 45 67 0c 0c 0c 0c 0d 0d 0d 0d 02 02 02 02 02
```

Capture 1 packet being transmitted on a port



Warning:

Capturing packets using Packet Driver may be CPU intensive and dangerous to system stability. By default the packet counter is set to infinity (packets are logged

continuously). Before executing any commands make sure that the packet counter is set to 0 - "echo n0 > /proc/pktdrv".

```
#-> echo g" 1/48 " > /proc/pktdrv
#-> echo d2 > /proc/pktdrv
#-> echo n1 > /proc/pktdrv
#-> dmesg | tail -n 1
[955785.260000] pd:    60 Lldp tx  1/48 v:1    Lldp 0180c2000003 e8e732ab17f3 q:7
```

Display number of hits per CPU queue during 1 second

```
#-> cat /proc/pktdrv > /dev/null; sleep 1; cat /proc/pktdrv | grep Classified
Classified 0 : 5200368 5214
Classified 13 : 1317 4
Classified 26 : 46 1
Classified 27 : 39 0
Classified 28 : 5274464 5215
```

1.11. Logging (SWLOG)

Logging (SWLOG)

1.12. Virtual Chassis

1.12.1. Checklist

- The hello interval parameter must match between switches
- The control VLAN must be the same between the switches comprising the virtual chassis
- Chassis group ID must be unique in a network
- To make sure that RCD works, all EMPs must be operational
- SFlow, ERP, UDLD and LLDP cannot be configured on VFL links (Configuration Manager doesn't allow these features to be configured on VFL links, but it doesn't verify configuration loaded from vcsetup.cfg), VC won't synchronise in case one of these protocols is enabled

1.12.2. Introduction

A Virtual Chassis is a group of switches managed through a single management IP address that operates as a single bridge and router. It provides both node level and link level redundancy for layer 2 and layer 3 services and protocols acting as a single device. The use of a virtual chassis provides node level redundancy without the need to use redundancy protocols such as STP and VRRP between the edge and the aggregation/core layer.

Some of the key benefits provided by a Virtual Chassis are:

- A single, simplified configuration to maintain
- Optimized bandwidth usage between the access layer and core
- Active-Active multi-homed link aggregation
- Provides predictable and consistent convergence with redundant links to the two switches
- Allows for exclusion of spanning-tree and other redundancy protocols like VRRP between the access layer and the core
- A Virtual Chassis appears as single router or bridge with support for all protocols
- A Virtual Chassis can be upgraded using ISSU to minimize network impact

Supported debug variables



Note: bootup-delay default is 120 seconds after L8

```
debug virtual-chassis bootup-delay <num>
```

Hardware requirements

- Two OS6900 or two OS10000 are required to form a VC
- 10G or 40G directly connected links must be used to between VC peers
- Mesh and ring topologies (future)

Hardware limitations

- It is not allowed to mix OS6900 and OS10000 in a virtual chassis
- Up to four OS10000 may be aggregated in a VC (future)
- Up to sixteen OS6900 may be aggregated in a VC (future)
- Up to 32 slots may be aggregated in a VC (future)
- To support Priority-based Flow Control (PFC) across the VFL links of Virtual Chassis the links have to be configured in a certain manner. Only one VFL member port is allowed from each port group based on the type of module being used to support the VFL member ports.

Software requirements

- Advanced license
- AOS 7.3.1.R01

Software limitations

- Distributed source learning is not supported in VC mode
- QoS rate limiting between chassis

Clustering mode

Proxied mode - some applications, typically management ones, are using a pseudo-distributed mode: the information needed/collected by local processes is used locally. The application's main intelligence lies on the master chassis, primary CMM, as with other applications. However, peer components running on slave chassis communicate with their master peer and present the information learned in such a way that local processes is able to access this information locally. Therefore, applications that follow a proxied model connect to their local version of that service. [1]

Application	Mode
RIP/RIPng	Centralized
OSPF/OSPF3	Centralized
DVMRP	Centralized
PIM	Centralized
Other L3	Centralized
IPCMM	Proxied
Configuration Manager	Proxied
Chassis Supervision	Proxied
IPRM	Proxied
BCD	N/A

Packet Driver	N/A
IPMS	Proxied
VCM	Proxied
L2 Apps	Centralized

EMP interfaces configuration

Standalone mode

Chassis

```
-> ip interface emp address a.b.c.d1 mask x.y.z.w
```

CMM from Maintenance Shell

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
#-> echo a.b.c.d3 > /proc/nvram/empipaddr
#-> echo x.y.z.w > /proc/nvram/empnetmask
```

or from CLI

```
-> modify boot parameters
Boot > boot empipaddr a.b.c.d3
Boot > boot empmasklength m
Boot > commit system
Boot > commit
```

Virtual-Chassis mode

Chassis

```
-> ip interface local chassis-id <chassis> emp address a.b.c.d1 mask x.y.z.w
```

Virtual-Chassis

```
-> ip interface master emp address a.b.c.d2 mask x.y.z.w
```

CMM - From Maintenance Shell

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
#-> echo a.b.c.d3 > /proc/nvram/empipaddr
#-> echo x.y.z.w > /proc/nvram/empnetmask
```

or from CLI

```
-> modify boot parameters
Boot > boot empipaddr a.b.c.d3
Boot > boot empmasklength m
Boot > commit system
Boot > commit
```

Summary

Interface type	Stored in	Interface names	Notes
CMM EMP	nvram	EMP-CMMA-CHAS1 EMP-CMMA-CHAS2 EMP-CMMB-CHAS1 EMP-CMMB-CHAS2	These interfaces are always present. If there is no value in proc/nvram/empipaddr then these interfaces are still created with IP address of 0.0.0.0

Chassis EMP	vcsetup.cfg	EMP-CHAS1 EMP-CHAS2
-------------	-------------	------------------------

These addresses are assigned to whichever CMM is primary in a dual-CMM OS10K. They correspond to what might have been set with the "ip interface emp" command from an old boot.cfg. When a "convert-configuration to" command is executed this value is removed from the new vcboot.cfg file and placed into the chassis specific vcsetup.cfg file. This interface is only created if there is an explicit command in the vcsetup.cfg file.

Virtual-Chassis EMP	vcboot.cfg	EMP-VC
---------------------	------------	--------

This is created with the ip interface master emp command in the vcboot.cfg file. It is assigned to the master's primary CMM.

Master chassis election

- Learning window 30 seconds after VFL comes up
- Master chassis election is based on:
- Highest Chassis priority

-> virtual-chassis configured-chassis-priority 100

- Longest Chassis uptime
- Smallest Chassis ID
- Smallest Chassis MAC Address

Virtual Chassis - Boot-Up

The Master chassis contains the vcboot.cfg file that contains the configuration for the entire virtual chassis. All the switches (i.e. the one that will eventually become the Master and the ones that will become Slaves) contain a vcsetup.cfg file that allows them to establish an initial connection over a VFL to all the other neighboring switches.

1. Upon boot-up, a switch will read its local vcsetup.cfg file and attempt to connect to the other neighbor switches
2. Upon connection, the switches will exchange the parameters configured in their local vcsetup.cfg files
3. As a result of this exchange, they will discover the topology, elect a Master based on criteria described in the next section, start periodic health checks over the VFL and synchronize their configuration as defined within the vcboot.cfg configuration file
4. All Slaves, if they do not have a local copy of vcboot.cfg, or their local copy does not match the copy found on the Master, will download their complete vcboot.cfg from the Master chassis and reboot using this copy of vcboot.cfg as its configuration file

Startup Error Mode

If a switch is unable to successfully come up in virtual chassis mode, it enters a special fallback mode called start up error mode. A switch moves to start up error mode if either one of the

following conditions occur:

- The vcsetup.cfg and vcboot.cfg configuration files are present in the running directory, but no valid advanced license is installed on the switch.
- The vcsetup.cfg file is corrupted or edited in such a way that it is unable to read a valid chassis identifier in the appropriate range.

A switch start up error mode will keep all of its front-panel user ports, including the virtual-fabric links member ports disabled. This mode can be identified on the switch by using the show virtual-chassis topology command. The chassis role will display Inconsistent, whereas the chassis status will show either one of the following values:

- Invalid-Chassis-Id: The chassis is not operational in virtual chassis mode because no valid chassis identifier has been found in the configuration. Typically this means that the vcsetup.cfg file is

corrupted, empty or contains an invalid (e.g. out of range) chassis identifier.

- Invalid-License: The chassis is not operational in virtual chassis mode because no valid advanced license has been found.

What may go wrong?

See below table for a list of events related to VCM [2]

Event	Cause	Virtual-Chassis Manager actions
Slave Chassis Down	Any	No action on the Master
	Controlled reboot or shutdown via CLI	Take over
Master Chassis Down	Power loss	The former slave chassis (new master) sends an RCD query asking about the peer. The peer chassis' age reported by RCD (last time we heard from that peer) will be greater than 10 seconds and we will assume that we can safely keep the master role with no further action
	Semi-controlled reboot via shell: reboot	
	Watch dog timeout	
	Crash of non-restartable CMM or NI process	
VFL Link Down	Cable severed	VC Split. The former slave chassis (new master) sends an RCD query asking about the peer. The peer chassis' age reported by RCD (last time we heard from that peer) will be smaller than 10 seconds and we will assume that the master is still alive. In order to avoid duplicate nodes ion the network we will shutdown all user ports, except the VFL member ports.
	Ports administratively disabled	
VCM Hello Down	VFL ports deleted	The former slave will treat this just VFL link down and perform the same steps.
	Software bug	
VCM Hello Down	vcmNi process died on designated NI and that was the last NI within the switch	

Chassis down detection

- Over VFL

- Hardware link scan
- LACP with slow reaction window - 90 second timeout
- TCP keepalive - 120 second timeout to avoid false positive
- Over EMP
 - Always enabled
 - Keepalives based on UDP
 - EMP address must be in the same subnet
 - Default hello interval 5 seconds - configurable using "virtual-chassis configured-hello-interval 5"
 - Split topology detected after 20 seconds by default (action on the former slave chassis)

Split-Chassis Detection protocol

Contents of SCD announcements:

- VCID - unique ID of the VC itself, to differentiate it from other VCs that may be present on the same EMP network.
- VCCID - unique ID of the switch (Chassis) within its VC.
- VCMaster - VCCID of switch that the originator of this announcement thinks is master
- ROLE - What is the role of the announcing switch in its VC (master/slave)

An example of a PDU generated by the Master:

```
Ethernet II, Src: Alcatel-_00:36:b8 (e8:e7:32:00:36:b8), Dst: Broadcast
(ff:ff:ff:ff:ff:ff)
  Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  Source: Alcatel-_00:36:b8 (e8:e7:32:00:36:b8)
  Type: IP (0x0800)
  Padding: 00000000000000
Internet Protocol Version 4, Src: 172.26.60.134 (172.26.60.134), Dst:
172.26.60.255 (172.26.60.255)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00
  Total Length: 40
  Identification: 0x0000 (0)
  Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 64
  Protocol: UDP (17)
  Header checksum: 0x690b [correct]
  Source: 172.26.60.134 (172.26.60.134)
  Destination: 172.26.60.255 (172.26.60.255)
User Datagram Protocol, Src Port: 10210 (10210), Dst Port: 10210 (10210)
  Source port: 10210 (10210)
  Destination port: 10210 (10210)
  Length: 20
Data (16 bytes)
0000 ff ff ff ff 00 00 00 01 00 00 00 02 00 00 00 00
```

Merging Virtual-Chassis after a split

- VFL connectivity is restored
- Duplicate Master chassis
- Reelection of the Master
- Reload of the Slave chassis

New commands

In Standalone mode

```
-> virtual-chassis configured-chassis-id <chassis>
-> virtual-chassis configured-control-vlan <vid>
-> virtual-chassis configured-hello-interval <interval>
-> virtual-chassis configured-chassis-priority <priority>
-> virtual-chassis vf-link <vf-link-id> create
-> virtual-chassis vf-link <vf-link-id> member-port <slot/port>
-> convert-configuration to <string>
```

In Virtual-Chassis mode

```
-> vc-takeover
-> show virtual-chassis [chassis-id <chassis>] topology
-> show virtual-chassis [chassis-id <chassis>] consistency
-> show virtual-chassis [chassis-id <chassis>] vf-link [<vfl-id>]
-> show virtual-chassis [chassis-id <chassis>] vf-link [<vfl-id>] member-port
-> show virtual-chassis [chassis-id <chassis>] chassis-reset-list
-> show virtual-chassis [chassis-id <chassis>] slot-reset-list
```

Modified commands

```
-> reload [chassis-id <chassis>] all [in [<hours>:]<minutes> | at
<hour>:<minute> [<month> <day> | <day> <month>] | cancel]
    {from <image directory>} {rollback-timeout <minutes> |
    no rollback-timeout [in [<hours>:]<minutes> | at <hour>:<minute> [Month
<num>]| redundancy-time <minutes>
    [secondary] [in [<hours>:]<minutes> | at <hour>:<minute> [<month> <day> |
<day> <month>] |cancel]
-> show log swlog {<chassis>/<slot-id>}
-> powersupply powersave {enable|disable} [chassis-id <chassis>]
-> update { uboot [cmm {<chassis>/<slot-id>} | ni {[<chassis>/]all |
[<chassis>/]slot-nr}] |
    fpga cmm {[<chassis>/]slot-id} | fpga ni {[<chassis>/]slot-nr} [daughter
{daughter-id}] }
-> show mac-range [chassis-id <chassis>] [index]
-> show temperature [chassis-id <chassis>] [fabric | slot | fantray {fantray-nr}
| cmm [cmm-nr | cmm-letter ]]
-> show fantray [chassis-id <chassis>] [fantray-nr]
-> show fan [chassis-id <chassis>]
-> show powersupply [chassis-id <chassis>] [<slot-nr> | powersave status]
-> show module [chassis-id <chassis>] [{long | status}] [number | cmm-letter]
-> show fabric [chassis-id <chassis>] [slot-nr]
-> show slot [chassis-id <chassis>] [slot-nr]
-> show cmm [chassis-id <chassis>] [slot-nr | cmm-letter]
-> show chassis [chassis-id <chassis>] [slot-nr]
-> show reload [chassis-id <chassis>] [all status | status]
-> show transceivers [chassis-id <chassis>] [slot slot-nr [transceiver
{transceiver-nr}]]
```

1.12.3. Configuration

Migration from a standalone configuration

- Supported user-initiated conversion
- User-initiated conversion (an example):

```
-> virtual-chassis configured-chassis-id 1
-> virtual-chassis vf-link 0 create
-> virtual-chassis vf-link 0 member-port 1/1
-> convert-configuration to working-vc
```


- Images are copied to the specified directory
- Requires one reload
- New configuration files are created:
 - vcboot.cfg
 - vcsetup.cfg
- New interface format
 - chassis-id/slot/port
- A "vcsetup.cfg" example

```
-> cat /flash/working/vcsetup.cfg
```

or

```
-> show configuration vcm-snapshot chassis-id
!=====!
! File: /flash/working/vcsetup.cfg    !
!=====!
! Virtual Chassis Manager:
virtual-chassis chassis-id 1 configured-chassis-id 1
virtual-chassis chassis-id 1 vf-link 0 create
virtual-chassis chassis-id 1 vf-link 0 member-port 1/1/1
virtual-chassis chassis-id 1 vf-link 0 member-port 1/1/2
! IP:
```

Migration from an MC-LAG configuration

- Not directly supported [1]
- Migrate to a standalone configuration
- Reload
- Migrate to Virtual Chassis
- Reload

1.12.4. Basic Troubleshooting

There are no new outputs included in tech-support.log

Verify management interfaces

```
-> show ip interface
Total 4 interfaces
```

Device	Name	IP Address	Subnet Mask	Status	Forward
EMP-VC		172.26.60.131	255.255.255.0	UP	NO
EMP	EMP-CHAS1	172.26.60.132	255.255.255.0	UP	NO
EMP	EMP-CHAS1	172.26.60.133	255.255.255.0	UP	NO
EMP	EMP-CMMA-CHAS1	172.26.60.134	255.255.255.0	UP	NO
EMP	EMP-CMMA-CHAS2	172.26.60.135	255.255.255.0	UP	NO
EMP	Loopback	127.0.0.1	255.255.255.255	UP	NO
EMP	Loopback				

Display Virtual Chassis logs from SWLOG

```
-> show log swlog | grep vcmCmm
```

An example (the Slave chassis converted to Master chassis):

```
-> show log swlog | grep vcmCmm | grep -i emp
(...) local0.info swlogd: vcmCmm config info(5)
CMM:vcmCMN_start_activity_timer_cb@11430:
Process (emp query req:2/0/1, timeout 20)
(...) local0.info swlogd: vcmCmm config info(5)
CMM:vcmCMN_activity_timer_handler_cb@11667:
Handled emp query req:2/0/1
(...) local0.info swlogd: vcmCmm protocol info(5)
CMM:vcmCMN_send_emp_vc_query_cb@1667:
Query (group 0, chassis 1)
(...) local0.info swlogd: vcmCmm config info(5)
CMM:vcmCMN_activity_timer_handler_cb@11704:
Perform EMP check (group 0, chassis 1, mac e8:e7:32:00:36:b9)
(...) local0.info swlogd: vcmCmm config info(5)
CMM:vcmCMN_start_activity_timer_cb@11430:
Process (emp query rsp:3/0/1, timeout 10)
(...) local0.info swlogd: vcmCmm config info(5)
CMM:vcmCMN_cancel_activity_timer_cb@11531:
Process (emp query rsp:3/0/1)
(...) local0.info swlogd: vcmCmm protocol info(5)
CMM:vcmCMN_process_emp_vc_response_cb@1814: Disable ports on split topology
(target
0/1, age 1 seconds, master 1, csflags 0x0)
```

This command is used to provide a detailed status of the virtual chassis topology.

```
-> show virtual-chassis topology
```

```
Local Chassis: 1
```

Chas	Role	Status	Config Chas ID	Pri	Group	MAC-Address
1	Master	Running	1	100	0	e8:e7:32:00:36:b9
2	Slave	Running	2	100	0	e8:e7:32:07:98:79

This command is used to provide a detailed status of the parameters taken into account to determine the consistency of a group of switches participating in the virtual chassis topology.

```
-> show virtual-chassis consistency
```

```
Legend: * - denotes mandatory consistency which will affect chassis status
```

Chas*	Config Chas ID	Chas Type	License*	Chas Group*	Oper Control Vlan*	Config Control Vlan	Oper Hello Interv*	Config Hello Interv	Status
1	1	OS6900	0x3	0	4094	4094	5	5	OK
2	2	OS6900	0x3	0	4094	4094	5	5	OK

A more detailed version of this output is available after adding the chassis-id option

```
-> show virtual-chassis chassis-id 1 consistency
```

```
Legend: * - denotes mandatory consistency which will affect chassis status
```

Consistency	Giving Chassis	Master Chassis	Status
Chassis-ID*	1	1	OK
Config-Chassis-ID	1	1	OK
Chassis-Type*	OS6900	OS6900	OK
License*	0x3	0x3	OK
Chassis-Group*	0	0	OK
Oper-Control-Vlan*	4094	4094	OK
Config-Control-Vlan	4094	4094	OK
Oper-Hello-Interval*	5	5	OK
Config-Hello-Interval	5	5	OK

Displays a summary of the configured and operational parameters related to the virtual fabric links on the virtual chassis topology

```
-> show virtual-chassis vf-link
```

Chassis/VFLink ID	Oper	Primary Port	Config Port	Active Port
1/0	Up	1/1/1	2	2
2/0	Up	2/1/1	2	2

And per link

```
-> show virtual-chassis vf-link member-port
```

Chassis/VFLink ID	Chassis/Slot/Port	Oper	Is Primary
1/0	1/1/1	Up	Yes
1/0	1/1/2	Up	No
2/0	2/1/1	Up	Yes
2/0	2/1/2	Up	No

This command displays the list of all chassis that must be reset along with a specified chassis in order to prevent a virtual chassis topology split

```
-> show virtual-chassis chassis-reset-list
Chas  Chassis reset list
-----+-----
 1      1,
 2      2,
```

For a given chassis and network interface module (NI), this command displays status information specifying whether bringing down or extracting such network interface module (NI) will lead to a virtual chassis topology split

```
-> show virtual-chassis slot-reset-list
Chas  Slot   Reset status
-----+-----
 1      1      Split
 2      1      Split
```

RCD operation can be verified only by enabling additional logs


```
-> swlog appid rcd subapp all level debug2

-> show log swlog | grep -i rcd | tail -10
Nov 26 03:14:26 (none) swlogd: rcd main debug2(7) Recv'd vcid 184503100 from
peer 0.0.0.0, ours 3 ignore it
Nov 26 03:14:26 (none) swlogd: rcd main debug2(7) receive peer 10.255.75.54
(master 1, csflags 0x0, mac e8:e7:32:07:97:ed)
Nov 26 03:14:26 (none) swlogd: rcd main debug2(7) process peer 10.255.75.54,
chassis 2, mac e8:e7:32:07:97:ed packet (@629)
Nov 26 03:14:26 (none) swlogd: rcd main debug2(7) peer 10.255.75.54's csflags
0x0 Nov 26 03:14:27 (none) swlogd: rcd main debug2(7) sending to 10.255.75.255
(csflags 0x0, mac e8:e7:32:07:92:7d)
Nov 26 03:14:27 (none) swlogd: rcd main debug2(7) Recv'd vcid 184503100 from
peer 0.0.0.0, ours 3 ignore it
Nov 26 03:14:27 (none) swlogd: rcd main debug2(7) receive peer 10.255.75.54
(master 1, csflags 0x0, mac e8:e7:32:07:97:ed)
Nov 26 03:14:27 (none) swlogd: rcd main debug2(7) process peer 10.255.75.54,
chassis 2, mac e8:e7:32:07:97:ed packet (@629)
Nov 26 03:14:27 (none) swlogd: rcd main debug2(7) peer 10.255.75.54's csflags
0x0
Nov 26 03:14:28 (none) swlogd: rcd main debug2(7) sending to 10.255.75.255
(csflags 0x0, mac e8:e7:32:07:92:7d)

-> swlog appid rcd subapp all level info
```


1.12.5. Advanced Troubleshooting

Enabling hardware linkscan for faster VC reconvergence (debug interfaces [port c/s/p] vfl-hw-linkscan {enable|disable|auto})

 **Warning:** This option is not supported in 7.3.2.R02 and 7.3.3.R01, it may lead to VFLs down after a reload

```
-> debug interfaces port 1/1/1 vfl-hw-linkscan enable
```

Collect basic information about the status of the Virtual Chassis:

 **Warning:** "RCD Operational Status" is "Up" even in case there is a duplicated chassis group ID in the same network!

-> debug show virtual-chassis status

ID	Level	Parameter	Value	Timestamp	Status
0	L0	Chassis Identifier	1	18:16:33	OK
1	L0	Designated NI Module	1	18:16:33	OK
2	L0	Designated NI Module (@L5)	1	00:15:09	OK
3	L0	License Configured	Yes	18:16:33	OK
4	L0	License Configured (@L5)	Yes	00:15:09	OK
5	L0	VFL Links Configured	1	18:16:33	OK
6	L0	VFL Links Configured (@L5)	1	00:15:09	OK
7	L0	VFL Ports Configured	1	18:16:33	OK
8	L0	VFL Ports Configured (@L5)	1	00:15:09	OK
11	L0	Chassis Ready Received	Yes	00:14:53	OK
12	L1	VFL Intf Oper Status	Up	18:16:33	OK
13	L1	VFL Intf Oper Status (@L5)	Up	00:15:09	OK
14	L2	VFL LACP Status	Up	18:16:33	OK
15	L2	VFL LACP Status (@L5)	Up	00:15:09	OK
16	L2	VFL LACP Up -> Down	2	00:20:08	INFO_04
17	L2	VFL LACP Down -> Up	3	00:24:16	INFO_03
18	L3	VCM Protocol Role (@L5)	Master	00:15:09	OK
19	L3	VCM Protocol Role	Master	18:16:33	OK
20	L3	VCM Protocol Status (@L5)	Running	00:15:09	OK
21	L3	VCM Protocol Status	Running	18:16:33	OK
24	L4	VCM Connection	Up	18:16:33	OK
25	L4	VCM Connection (@L5)	Up	00:15:09	OK
26	L5	VCM Synchronization	Multi-node	18:16:33	OK
27	L6	Chassis Sup Connection	Up	00:24:19	OK
28	L6	Remote Flash Mounted	Yes	00:24:19	OK
29	L6	Image and Config Checked	N/A	N/A	N/A
30	L6	VC Takeover Sent	Yes	00:15:09	OK
31	L7	VC Takeover Acknowledged	Yes	00:15:10	OK
32	L8	System Ready Received	Yes	00:15:10	OK
33	L8	RCD Operational Status	Up	00:15:10	OK
34	L8	RCD IP Address	172.26.60.135	00:13:11	OK

Error/Information Codes Detected:

INFO_04

This parameter provides the counter of how many times the virtual-fabric link operational status has transitioned from up to down since the switch was started. Depending on the specific operations you are performing on the system, this counter may increase. Under normal conditions this counter should ideally be one unit smaller than the counter for the opposite transitions from operational down to up

INFO_03

This parameter provides the counter of how many times the virtual-fabric link operational status has transitioned from down to up since the switch was started. Depending on the specific operations you are performing on the system, this counter may increase. Under normal conditions this counter should ideally be one unit greater than the counter for the opposite transitions from operational up to down

Collect debug traces and data, either to a file or, if no file is specified, to the standard output "debug dump virtual-chassis type {trace | data | all} chassis-id <chassis-id> slot {<slot-id> | all} [output-file <filepath/filename>]". Example:

```
-> debug dump virtual-chassis chassis-id 1 type all slot all output-file
/flash/vc_debug.txt
Please wait.....
```

Output File : /flash/vc_debug.txt successfully created (size 367817 bytes)
-> cat /flash/vc_debug.txt

```
=====  
Context Information  
=====  
System Name : OS6900-VC  
System Description : Alcatel-Lucent OS6900-X20 7.3.1.499.R01 Development, August  
17, 2012.  
Date & Time : FRI SEP 28 2012 01:58:46  
=====  
...
```

AOS7 allows a user to execute low-level debugging tools on remote unit. Execute "debug exec virtual-chassis help {cmm | ni}" for more details.

-> debug exec virtual-chassis help cmm

```
[rgdb] //////////////////////////////////////  
[rgdb] Executing "call vcm_help (1)" : process vcmCmm, chassis:1, local:cmm-a  
[rgdb] Execution out of CLI context with redirection of standard output  
[rgdb] 'CTRL + C' deactivated if running from a debug CLI command  
[rgdb] //////////////////////////////////////  
  
[Thread debugging using libthread_db enabled]  
[New Thread 0x4aac34c0 (LWP 4757)]  
Please wait.0xf977c78 in select () from /lib/tls/libc.so.6
```

Detailed information commands

```
-----  
> vcm_show_data ()  
    show detailed context including software and hardware data  
> vcm_show_sw ()  
    detailed software application data  
> vcm_show_hw ()  
    detailed ASIC information read from hardware  
> vcm_show_trace (detail)  
    show detailed trace information about the protocol, IPC and execution
```

Execute low-level debugging tools on remote unit using "debug exec virtual-chassis chassis-id <chassis-id> slot <slot-id> <"call func(args)">"

-> debug exec virtual-chassis chassis-id 1 slot all "call vcm_show_reactor()"

```
[rgdb] //////////////////////////////////////  
[rgdb] Executing "call vcm_show_reactor()" : process vcmCmm, chassis:1,  
local:cmm-a  
[rgdb] Execution out of CLI context with redirection of standard output  
[rgdb] 'CTRL + C' deactivated if running from a debug CLI command  
[rgdb] //////////////////////////////////////
```

```
Please wait.[Thread debugging using libthread_db enabled]  
[New Thread 0x4c2c44c0 (LWP 5816)]  
0xf977c78 in select () from /lib/tls/libc.so.6
```

```
=====  
local chassis: 1 slot: 65
```

Reactor Resource

```
=====  
user_app_id : 153
```

```

reactor_control.initialized      : true
server_port                      : 17650,
app_callback_std.rx_reactor     : 0x1008bb64
...

```

ISIS troubleshooting in AOS 7.3.3.R01:

```
-> debug show virtual-chassis protocol ?
```

```

^
UNICAST-TABLE SPF NODES
MULTICAST-SOURCES-SPF MULTICAST-SOURCES
INTERFACE INFO DATABASE ADJACENCY

```

(Isis Vc Command Set)

```
-> debug show virtual-chassis protocol unicast-table
```

ISIS VC Unicast Table:

Destination (Name : MAC Address)	Outbound Interface
Chassis-1 : e8:e7:32:70:97:9f	2/0
Chassis-6 : e8:e7:32:70:98:9b	2/4
Chassis-3 : e8:e7:32:70:98:b7	2/1
Chassis-5 : e8:e7:32:70:9a:3f	2/3
Chassis-4 : e8:e7:32:70:9e:4b	2/2

MAC Addresses: 5

```
-> debug show virtual-chassis protocol spf
```

ISIS VC Path Table:

Destination VC Num (Name : SYS MAC) Metric Hops	Outbound Interface	Next Hop (Name : SYS MAC)
Chassis-1 : e8:e7:32:70:97:9f e8:e7:32:70:97:9f 20 1	2/0	Chassis-1 :
Chassis-6 : e8:e7:32:70:98:9b e8:e7:32:70:98:9b 20 1	2/4	Chassis-6 :
Chassis-3 : e8:e7:32:70:98:b7 e8:e7:32:70:98:b7 20 1	2/1	Chassis-3 :
Chassis-5 : e8:e7:32:70:9a:3f e8:e7:32:70:9a:3f 20 1	2/3	Chassis-5 :
Chassis-4 : e8:e7:32:70:9e:4b e8:e7:32:70:9e:4b 20 1	2/2	Chassis-4 :

SPF Path count: 5

```
-> debug show virtual-chassis protocol nodes
```

SPB ISIS Nodes:

System Name	System Id	SourceID	BridgePriority
Chassis-1	e8e7.3270.979f	0x0979f	32768 (0x8000)
Chassis-6	e8e7.3270.989b	0x0989b	32768 (0x8000)
Chassis-3	e8e7.3270.98b7	0x098b7	32768 (0x8000)
Chassis-5	e8e7.3270.9a3f	0x09a3f	32768 (0x8000)
Chassis-4	e8e7.3270.9e4b	0x09e4b	32768 (0x8000)
Chassis-2	e8e7.3270.9ed7	0x09ed7	32768 (0x8000)

Total SPB Nodes : 6

System (Name : SystemId)	Type	State	Hold	Interface
Chassis-1 : e8e7.3270.979f	L1	UP	25	2/0
Chassis-3 : e8e7.3270.98b7	L1	UP	25	2/1
Chassis-4 : e8e7.3270.9e4b	L1	UP	25	2/2
Chassis-5 : e8e7.3270.9a3f	L1	UP	25	2/3
Chassis-6 : e8e7.3270.989b	L1	UP	26	2/4

Adjacencies : 5

Finding the shortest path in a virtual chassis topology (a ring example):

-> debug show virtual-chassis chassis-id 3 topology
 Legend: licenses-info - A: Advanced; B: Data Center

```
Oper-Chassis-ID           : 3,
Config-Chassis-ID        : 3,
Chassis-Role              : Slave,
Previous-Chassis-Role    : Unassigned,
Chassis-Status           : Running,
Chassis-Group            : 0,
Chassis-MAC              : e8:e7:32:70:98:b7,
Up-Time                   : 0 days 2 hours 31 minutes and 36 seconds,
Designated-NI            : 0,
Primary-CMM               : CMM-A,
Secondary-CMM            : Not present,
Chassis-Type              : OS6900,
License                   : A,
Hello-Interval           : 10,
Oper-Chassis-Priority    : 140,
Config-Chassis-Priority  : 140,
Oper-Control-VLAN        : 4094,
Config-Control-VLAN      : 4094,
System-Ready             : Yes,
Number-Of-Neighbors     : 5,
Number-Of-Direct-Neighbors : 2
```

Neighbor	Is-Direct	Shortest-Path
1	No	3/1->2/1->2/0->1/0
2	Yes	3/1->2/1
4	Yes	3/2->4/2
5	No	3/2->4/2->4/3->5/3
6	No	3/1->2/1->2/0->1/0->1/4->6/0

1.12.6. Troubleshooting in Maintenance Shell

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.



Warning:

Accessing remote file systems

Access to remote file systems through /mnt/chassisN_CMM[AB]

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
#-> ls -la /mnt/chassis2_CMMA
```

2. Layer 1 Troubleshooting

2.1. Physical Interfaces

2.1.1. Checklist

- Autonegotiation setting must be the same on both sides of a link
- Copper links should be using Full Duplex, otherwise it is expected to observe packet loss due to collisions
- The "Number of Status Change" counter shouldn't increment

2.1.2. Introduction

Gport to dport mapping

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
TOR #-> telnet nil 5008
IPNI-VRF-0> portinfo
```

```
Chassis 1:
NI 1: has 1 boards
  Board 1:
1/1/1 -> gport 0x0(0) -> modid 0, dport 1
1/1/2 -> gport 0x1(1) -> modid 0, dport 2
1/1/3 -> gport 0x2(2) -> modid 0, dport 3
...
1/1/40 -> gport 0x27(39) -> modid 0, dport 40
...
```

2.1.3. Basic troubleshooting

Show interfaces output:

```
-> show interfaces 1/1/1
Chassis/Slot/Port 1/1/1 :
Operational Status      : up,
Last Time Link Changed  : Thu Dec 19 23:04:31 2013,
Number of Status Change: 1,
Type                    : Ethernet,
SFP/XFP                 : SFP_PLUS_COPPER,
EPP                     : Disabled,
Link-Quality            : GOOD,
MAC address              : e8:e7:32:00:00:49,
BandWidth (Megabits)    : 10000,          Duplex          : Full,
Autonegotiation         : 0 [              ],
Long Frame Size(Bytes)  : 9216,
Rx                      :
Bytes Received          : 103948266, Unicast Frames : 45582,
Broadcast Frames:      551, M-cast Frames : 903627,
UnderSize Frames:      0, OverSize Frames: 0,
Lost Frames            : 0, Error Frames : 0,
CRC Error Frames:      0, Alignments Err : 0,
Tx                      :
Bytes Xmitted           : 64469024, Unicast Frames : 14460,
Broadcast Frames:      4400, M-cast Frames : 520850,
UnderSize Frames:      0, OverSize Frames: 0,
Lost Frames            : 0, Collided Frames: 0,
Error Frames           : 0
```

Display logs related to interfaces toggling on slot 1:

```
-> show log swlog slot 1 | grep LINKSTS
... (none) local0.info swlogd: portMgrNi main info(5) :
[:pnmLinkStatusCallback:187] LINKSTS 1/18 UP (gport 17) Speed 10000 Duplex FULL
... (none) local0.info swlogd: portMgrNi main info(5) :
[:pnmLinkStatusCallback:187] LINKSTS 1/18 DOWN (gport 17) Speed 10000 Duplex
FULL
... (none) local0.info swlogd: portMgrNi main info(5) :
[:pnmLinkStatusCallback:187] LINKSTS 1/18 UP (gport 17) Speed 10000 Duplex FULL
... (none) local0.info swlogd: portMgrNi main info(5) :
[:pnmLinkStatusCallback:187] LINKSTS 1/18 DOWN (gport 17) Speed 10000 Duplex
FULL
... (none) local0.info swlogd: portMgrNi main info(5) :
[:pnmLinkStatusCallback:187] LINKSTS 1/18 UP (gport 17) Speed 10000 Duplex FULL
```

For more detailed logs use commands below and change the status of the affected port up and down:

```
-> swlog appid intfCmm subapp all level debug1
-> swlog appid intfNi subapp all level debug1
```

Gather logs:

```
-> show log swlog
```

And finally change logging level to default:

```
-> swlog appid intfCmm subapp all level info
-> swlog appid intfNi subapp all level info
```

In case Tx Lost Frames counter increments on a selected port please enable per queue "stats" for this port:

```
-> qos qsi port 1/1/1 stats admin-state enable
```

And monitor in which drops are observed:

```
-> show qos qsi port 1/1/1 stats
```

Port	Q	Tx	Total Drop
1/1/1	1	0	0
1/1/1	2	0	0
1/1/1	3	0	0
1/1/1	4	0	0
1/1/1	5	0	0
1/1/1	6	0	0
1/1/1	7	0	0
1/1/1	8	0	0

Tx Lost Frames should be incrementing at the same time as total drops on this port. In this case Tx Lost Frames increment due to egress congestion. If total drop doesn't increment, then please proceed with TDBG3 troubleshooting from the bShell section in this article.

2.1.4. Advanced troubleshooting

Not applicable

2.1.5. Troubleshooting in Maintenance Shell



Warning:

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

esmUtil

When swlog debug levels are configured all ports will output their information. The esmUtil application provides a debug port filtering facility to control which port debug is output. Either on the CMM or NI enter 'su' mode.

```
-> su
Entering maintenance shell. Type 'exit' when you are done.
RUSHMORE #-> esmUtil
usage: esmUtil -<optionId>
  -plf <cmdOptions>    - Port Log Filtering
                        - off:Default:
                          Turns port filtering off.
                          Logging occurs for all ports.
                        - on:
                          Turns port filtering on.
                          Logging occurs only for ports on the filter
list.
                        - show <slot#(1-8)>:
                          Display a slots port log filtering list
                        - add <slot#(1-8)> <port#(1-x)>:
                          A port to the port log filter list
                        - remove <slot#(1-8)> <port#(1-x)>:
                          A port from the port log filter list

i.e. esmUtil -plf on           => Filtering on.
     esmUtil -plf off         => Filtering off.
     esmUtil -plf show 1      => Display port filtering
                               parameters for slot 1
     esmUtil -plf add 1 1     => Add slot 1 port 1 to filter
                               list. For NI make slot 1
     esmUtil -plf remove 1 1 => Remove slot 1 port 1 from
                               For NI make slot 1
                               filter list.
```

Other tools

Dump all port information for all ports on in a slot:

```
RUSHMORE #-> portmgrtestcmm dump 1
SLOT 1 Ports
-----

Port Entry Found for gport 0x0 0 (1/1/1)
-----

Port valid      : 1
ifIndex         : 1001
AdminStatus     : 1
LinkState       : 1
Violation       : 0
ViolReason      : 0
PortState       : CHASSIS INTERCONNECT (5)
```

```
PortProperty   : 0x0
PortType       : Physical (1)
...
```

Display physical chipset information:



Warning:

Chipset revisions A1 and C2 are not supported

```
TOR #-> esmPhyInfo
Port:1 - NLP_2042_Tabby_revA2
Port:2 - NLP_2042_Tabby_revA2
Port:3 - NLP_2042_Tabby_revA2
Port:4 - NLP_2042_Tabby_revA2
Port:5 - NLP_2042_Tabby_revA2
...
```

The ESM and BCMD interface drivers use swlog and esmUtil to manage their port debug capability. To turn on debug for all ESM and BCMD ports input:

- swlog console level debug3
- swlog appid Intf subapp all level debug[1,2,3]
 - debug1 will output link-up/link-down indications.
 - debug2 will output debug1 and connection oriented debug.
 - debug3 will output debug1, debug2 and statistics oriented debug.
- swlog appid bcmd subapp all level debug[1,2,3]

There is a port manager utility that is used to dump debug information and shared memory information on the console. The name of the utility is portmgrtest[cmm|ni]. This utility is ran as root in the Linux shell. Portmgrtestcmm exists on the CMM, portmgrtestni exists on the NI.

The help (which is printed when executed without options) gives the usage. Some options are not available in CMM and some not on the NI.

```
RUSHMORE #-> portmgrtestcmm
plprofile slot
pmmmap      ---- test the port manager
dump [slot] [slot/port]
              ---- dump internal portDB for connected NI's
              slot - is optional (only dumps specific slot
              port - is optional (only dumps specific slot/port
rawdump [slot] ---- raw dump internal data of all portDB
              slot - is optional (only dumps specific slot
lagdump [lag]  ---- dump lag internal data of the port manager
              lag - is optional (only dumps specific lag
vfldump      ---- dump vfl internal data of the port manager
mcdbdump [lag] ---- dump remote lag internal data of the port manager
intvc       ---- dump internal Virtual chassis connect ports
peers       ---- dump peer connections
viol        ---- dump violation db
appdb [type] ---- application database debugs
              type - 2: Dump all app registration
                   3: Phy Port LinkSts Upd Stats
                   4: Phy Port State Upd Stats
                   5: App param registration
param       ---- set port param value

RUSHMORE # portmgrtestni
pmmmap      ---- test the port manager
```

```

dump [slot] [slot/port]
    ---- dump internal portDB for connected NI's
        slot - is optional (only dumps specific slot)
        port - is optional (only dumps specific slot/port)
rawdump [slot] ---- raw dump internal data of all portDB
        slot - is optional (only dumps specific slot)
lagdump [lag] ---- dump lag internal data of the port manager
        lag - is optional (only dumps specific lag)
vfldump ---- dump vfl internal data of the port manager
mcdbdump [lag] ---- dump remote lag internal data of the port manager
intvc ---- dump internal Virtual chassis connect ports
peers ---- dump peer connections
viol ---- dump violation db
appdb [type] ---- application database debugs
        type - 2: Dump all app registration
              3: Phy Port LinkSts Upd Stats
              4: Phy Port State Upd Stats
              5: App param registration
setviol ---- set violation on a gport

```

Port Database Dump

portmgrtest[cmm|ni] dump 1/1 - Dumps the port information stored in the shared DB

portmgrtest[cmm|ni] dump 1 - Dumps all the port information on the slot

```

RUSHMORE #-> portmgrtestcmm dump 1/1
Port Entry Found for gport 0x0 0 (1/1)
-----

```

```

Port valid      : 1
ifIndex        : 1001
AdminStatus    : 1
LinkState      : 0
Violation      : 0
PortState      : FIXED (1)
PortProperty   : 0x0
PortType       : Physical (1)

Speed          : 0
Duplex         : 0
AutoNeg        : 0
EgressBW       : 0 kbps
EgressBWConf   : 0
Reason         : LINK DOWN - NO REASON

LAG            : 255

```

portmgrtest[cmm|ni] rawdump 1/1 - Dumps the raw port db information portmgrtest[cmm|ni]

rawdump 1 - Dumps all the ports information on the slot

```

RUSHMORE #-> portmgrtestcmm rawdump 1/1
SLOT 1 Ports
-----

```

```

Default Profile Slot 1 gport Range 0-47
Port Entry Found for gport 0x0 0 (1/1)
-----

```

```

Port valid      : 1
ifIndex        : 1001
AdminStatus    : 1
LinkState      : 0

```

```

Violation      : 0
PortState     : FIXED (1)
PortProperty  : 0x0
PortType      : Physical (1)

Speed         : 0
Duplex        : 0
AutoNeg       : 0
EgressBW     : 0 kbps
EgressBWConf  : 0
Reason       : LINK DOWN - NO REASON

LAG           : 255

```

rawdump - is used mostly for internal debug to see if PM created the DB entries. The "dump" will display only after the NI is up.

LinkAgg Dump

portmgrtest[cmm|ni] lagdump [aggid] - Dumps the agg database on the portmanager entity

```

RUSHMORE #-> portmgrtestcmm lagdump 110
Port Entry Found for gport 33554542 (0x200006e) lag 110
-----

```

```

Port valid      : 1
ifIndex        : 40000110
AdminStatus    : 1
LinkState      : 2
Violation      : 0
PortState     : FIXED (1)
PortProperty   : 0x0
PortType      : Logical (2)

LagId          : 110
Type           : 0
remote        : 0
mCLag         : 0
mCLagRemLS    : 0
primPort      : -1
LinkBw        : 0
size          : 2
sys Prio      : -1
Sys id        : ff:ff:ff:ff:ff:ff

```

```

Members :
Conf Members :

```

portmgrtest[cmm|ni] vfldump

Only one VFL link is present for now

Application Registration Database Dump

```

portmgrtest[cmm|ni] appdb 2 - Dumps all the application registration
portmgrtest[cmm|ni] appdb 3 - Phy Port LinkSts registration and tx/rx stats
dump
portmgrtest[cmm|ni] appdb 4 - Phy Port State/Property registration and tx/rx
stats dump

```

Violation Database Dump

portmgrtest[cmm|ni] viol - Dumps all the violations on the port.

```
RUSHMORE #-> portmgrtestcmm viol  
VIOLATION INFORMATION
```

```
-----  
gport          src          reason      app      action    time      timeact  
-----+-----+-----+-----+-----+-----+-----  
  
-----
```

Resetting a port in shutdown state on OS6900T

If the issue is temporary due to PHY registers, below command would reset the port 1/1/20:

```
TOR #-> debug $(pidof new_cs) "call fpgaMgrSetPhyReset(20,1)"
```

If the issue persists, hard reboot the switch and check for the issue. If the issue is still seen after rebooting the switch then the switch needs to be replaced.



Warning:

This method is not recommended for non T models. Also while resetting from Maintenance Shell will shutdown the adjacent ports hence it is required that this action to be carried out during maintenance window.

2.2. Quality of Service

2.2.1. Checklist

- ARP messages are matched by IP conditions
- Make sure that the "log" keyword in "policy rule" is really needed - presence of this keyword is increasing TCAM usage by 1 entry per rule
- Use masks for MAC and IP addresses wherever possible
- Prefer actions with "maximum bandwidth" than "cir"
- Use manually optimized network group in place of build in "Switch"
- Specify source ports where applicable

2.2.2. Introduction

Reserved slices

OS6860

Helix4 has 16 slices of 256 entries each:

- Slice 0 - Reserved for untrusted port entries/low priority (overridable) system slice
- Slice 1 - Reserved for IP protocol cpu priority entries
- Slices 14 & 15 - Reserved for high priority (non overridable) system slices

Features which require TCAM reservation:

- QoS Policies - can dynamically use all free slices based on policy configuration
- SIP snooping - will use between 1-4 slices based on a static tunable when enabled
- FIP snooping - reserves a single slice when enabled
- OpenFlow - Will reserve all free slices (2-13) when enabled (no other applications can be in use if it is enabled)
- AntiSpoofing - reserves a single slice when enabled
- Vlan Stacking / SPB SAPs - can dynamically use all free slices based on configuration
- *,G: - reserves a single slice when enabled
- DHCP Snooping - reserves a single slice on ASICs where DHCP snooping ports are enabled
- Deep Packet Inspection - reserves a single slice when enabled

No cache

For 'Switch' group policies it's recommended to use the 'no-cache' keyword in the action for the rule (it really should be part of the rule, but for historical reasons it's in the action). That will cause the policy to not be programmed into the hardware TCAM and only be matched in software. It is recommended to use AOS 7.3.1.643.R01 or later - given test traffic in the lab, show health shows no change in cpu utilization.

It's may be tricky to get right though, especially in the case there are rules with a lower precedence "deny all" and higher precedence "accept" policies. All policies that come after the first no-cache policy will need to be carefully checked to see if they also need to be no-cache, since if there's any overlap then they'll match first if they're in hardware.

Minimum and maximum bandwidth per queue

The following configuration from AOS 6:

```
qos port slot/port qn {minbw | maxbw} kbps
```

Can be replaced by:

```
-> show configuration snapshot vfc
! Virtual Flow Control:
qos qsp dcb "port11001" import qsp dcb "dcp-1"
qos qsp dcb "port11001" tc 1 min-bw 0 max-bw 20
qos qsp dcb "port11001" tc 2 max-bw 20
qos qsi port 1/1/1 qsp dcb "port11001"
```

QoS/ACL Design and Configuration

Introduction

QoS software on OmniSwitch 6900 provides a way to manipulate flows coming through the switch based on user configured policies such as ACLs, traffic prioritization, bandwidth shaping or traffic marking and mapping. ACLs are a specific type of QoS policy used for Layer 2, Layer 3/4, and multicast filtering.

This Technical Tip is intended to provide you the QoS/ACL necessary information on the OmniSwitch 6900 series to successfully configure and design a QoS/ACL policy in your networks.

Examples provided in this document are taken from an OmniSwitch 6900 running 7.3.2.355.R01 with 26 ports which is a one NI (Network Interface) version. NI consists of a switching ASIC and physical ports.

Policy Condition and Action Guidelines

List of the Policy Conditions and Actions Available

Qos Conditions

Qos Actions

Layer 1 Conditions

- Source / destination port
- Source / destination port group

Layer 2 Conditions

- Source MAC / MAC group
- Destination MAC / MAC group
- 802.1p
- Ethertype
- Source VLAN
- Destination VLAN (multicast rules only)
- Outer VLAN
- Inner VLAN

Layer 3 Conditions

- IP Protocol
- Source / destination IP
- Source / destination network group
- ToS
- DSCP
- ICMP code/type
- IPV6 Destination IP, Traffic, Next Header, Flow Label
- Multicast IP / Network Group

- ACL Drop
- Priority, specify the egress queue (0-7)
- Specify the maximum queue depth
- 802.1p stamping and mapping
- ToS stamping and mapping
- DSCP stamping and mapping
- Permanent gateway (Policy Based Routing)
- Maximum Bandwidth
- Port Redirection
- Port Disable

Layer 4 Conditions

- Source / destination TCP/UDP port
- Destination
- Service
- Service group
- ICMP type
- TCP Flags

IP Multicast


- For IGMP ACLs QoS Actions

Policy Condition Combination Matrix

&	Layer 1	Layer 2	Layer 3	Layer 4	IP Multicast(IGMP)
Layer 1	All	All	All	All	Destination Only
Layer 2	All	All	All	Source VLAN and 802.1p only	Destination Only
Layer 3	All	All	All	All	Destination Only
Layer 4	All	Source VLAN and 802.1p only	All	All	None
IP Multicast(IGMP)	Destination Only	Destination Only	Destination Only	None	N/A

Policy Action Combination Matrix

&	Drop	Priority	Stamp/Map	Maximum Bandwidth	Redirect Port	Redirect Linkagg
Drop	N/A	No	No	No	No	No
Priority	No	N/A	Yes	Yes	Yes	Yes
Stamp/Map	No	Yes	N/A	Yes	Yes	Yes
Maximum Bandwidth	No	Yes	Yes	N/A	Yes	Yes
Redirect Port	No	Yes	Yes	Yes	N/A	No
Redirect Linkagg	No	Yes	Yes	Yes	No	N/A

 **Warning:** Reflexive rules and NAT are not supported.

Understanding the QoS/ACL implementation on the OmniSwitch 6900

On the OmniSwitch 6900, QoS and ACL classification and actions are performed in hardware. QoS/ACL rules are a combination of conditions and actions. The OS6900 can classify, stamp and prioritize on Layer 2 through Layer 4 traffic simultaneously, whether bridging or routing.

Summary of guidelines to understand the QoS and TCAM usage and successfully configure policy rules on OS6900:

- The switching ASIC on each switching ASIC processes QoS and ACLs internally uses TCAM (Ternary Content Addressable Memory).
- The TCAM is divided into 14 slices, including 10 IFP slices and 4 EFP slices.
- IFP slices 0, 1, 2, and 3 can accommodate 128 rules, all other slices can accommodate 256 rules.
- IFP slices 0, 1, 2, 3, 8, and 9 are reserved for the system use. User policy rules aren't configured in these slices.
- 4 IFP slices are available in User Policy Rules Space allowing $4 \cdot 256 = 1024$ rules.
- User policy rules are always in lower slices than the slice(s) allocated for Anti-spoofing,

Ethernet-Service, DHCP IP Source Filtering.

- User policy cannot overwrite the sap-profile priority assignment and rate limiting.
- TCAM rules are programmed on every NI if the policy rule does not specify a source port. If policy is applied on a stack, rules without specified source port are configured across all units and their NIs.
- Once the slices are set with their parsing modes, a packet will be looked-up in parallel in all slices (not applicable to user QoS rules).
- When a match occurs in one slice, the parsing stop in that slice.
 - A drop has always precedence over accept
 - The smallest rate limiter is always enforced
- Policy Network / MAC / Map / Destination Slot-Port / Service Group used in a policy rule consume one TCAM rule for every entry in the group.
- Combining different policy groups in the policy rules consumes one TCAM rule for every possible combination of match between the groups.
- TCP/UDP hardware ranges are being used when the range consumes more than 5 TCAM regular rules, using a TCP/UDP hardware range consumes 1 TCAM rule.
- If the same rule type requires more than entries available in a single slice, a second slice (set with the same classification type is allocated).
- Efficient usage of the policy rule precedence allows the user to configure more rules and can avoid reaching the system limitation.

TCAM allocation rules and practical examples

QoS starts allocating rules in IFP slice 7, and works towards IFP slice 4. Each slice is set up to look at a particular set of fields in a packet. If all the entries in a slice are used, or the slice can't be programmed to accommodate all the fields needed in the condition, QoS moves on to the next lower slice.

Simple policy rules consuming only 1 TCAM entry

Single source IP

Configuration

```
policy condition c1 source IP 1.1.1.1
policy action a1 disposition accept
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

1 Rule is consumed on the NI

-> show qos slice

Slot/ Unit	Type	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/(0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/256	256/256	256/256
			7	256/255	256/255	256/256
			8	256/255	256/254	256/254
			9	256/255	256/256	256/256
1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Single Source Network

Configuration

```
policy condition c1 source IP 1.1.1.0 mask 255.255.255.0
policy action a1 disposition accept
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

1 Rule is consumed on the NI

-> show qos slice

Slot/ Unit	Type	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/(0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/256	256/256	256/256
			7	256/255	256/255	256/256
			8	256/255	256/254	256/254
			9	256/255	256/256	256/256
1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Mixing single source IP and destination IP in a policy condition will consume only one TCAM entry

Configuration

```
policy condition c1 source ip 1.1.1.1 destination ip 2.2.2.2
policy action a1 disposition accept
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

1 Rule is consumed on the NI

-> show qos slice

Slot/ Unit	Type	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/(0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/256	256/256	256/256
			7	256/255	256/255	256/256
			8	256/255	256/254	256/254
			9	256/255	256/256	256/256
1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Policy conditions consuming multiple TCAM entries for a single policy rule

Rules of consumption

When the user configures a policy rule which has the below given condition it will consume multiple TCAM entries for a single policy rule:

- policy network group
- policy MAC group
- policy slot-port group (only for destination port group, source port group requires one TCAM entry only because TCAM maintains a port bitmap to accommodate multiple ports)
- policy service group
- policy map group



Warning:

Using a single policy group in a policy rule can consume one TCAM rule for every entry in the group. Combining different policy groups in the policy rules can consume one TCAM rule for every possible combination of match between the groups.

Example: Adding individual elements in a policy groups will increase the number of TCAM rules consumed

Configuration

```
policy network group g1 1.1.1.1 2.2.2.2 3.3.3.3
policy condition c1 source network group g1
policy action a1 disposition accept
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

This configuration consumes 1 TCAM rule for each entry in the network group on every NI, so 3 rules are consumed in total in slice 7 with this configuration. 253 rules in slice 7 are now available.

```
-> show qos slice</span>
```

Slot/ Meters Unit	Ranges Type	Total/Free	Rules CAM	Total/Free	Counters Total/Free	Total/Free
1 / (0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/256	256/256	256/256
			7	256/253	256/253	256/256
			8	256/255	256/254	256/254
			9	256/255	256/256	256/256
1 / (0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Combining different policy groups in the policy rules can consume one TCAM rule for every possible combination of match between the groups

Configuration

```
policy network group g1 1.1.1.1 2.2.2.2 3.3.3.3  
policy network group g2 4.4.4.4 5.5.5.5 6.6.6.6  
policy condition c1 source network group g1 destination network group g2  
policy action a1 disposition accept  
policy rule r1 condition c1 action a1  
qos apply
```

TCAM Utilization

This configuration consumes 9 TCAM rules = 3 Source IP • 3 Destination IP for every possible match between the group on the NI:

Source IP = 1.1.1.1 and Destination IP = 4.4.4.4

Source IP = 1.1.1.1 and Destination IP = 5.5.5.5

Source IP = 1.1.1.1 and Destination IP = 6.6.6.6

Source IP = 2.2.2.2 and Destination IP = 4.4.4.4

Source IP = 2.2.2.2 and Destination IP = 5.5.5.5

Source IP = 2.2.2.2 and Destination IP = 6.6.6.6

Source IP = 3.3.3.3 and Destination IP = 4.4.4.4

Source IP = 3.3.3.3 and Destination IP = 5.5.5.5

Source IP = 3.3.3.3 and Destination IP = 6.6.6.6

```

-> show qos slice
Slot/
Unit          Type  Ranges      CAM  Rules      Counters      Meters
              Total/Free  Total/Free  Total/Free  Total/Free
1/(0)         IFP   32/32       0    128/128    128/128      128/128
              1    128/127    128/127
              2    128/125    128/125
              3    128/127    128/127
              4    256/256    256/256
              5    256/256    256/256
              6    256/256    256/256
              7    256/247    256/247
              8    256/255    256/254
              9    256/255    256/256
1/(0)         EFP   0/0         0    256/256    256/256      256/256
              1    256/256    256/256
              2    256/256    256/256
              3    256/256    256/256

```

A single policy condition consumes 2 TCAM rules due to slice pairs

Rules of consumption

In AOS7, Source IP conditions and destination IP conditions use paired TCAM slices.

Example: Source IPv6 condition consume 1 TCAM rule in a slice pair

Configuration

```

policy condition c1 source vlan 559 source ipv6 2001:4cd0:bc00:2570:1::1
destination tcp-port 80
policy action a1
policy rule r1 condition c1 action a1
qos apply

```

TCAM Utilization

1 rule consumed in slice pair 6_7.

```

-> show qos slice
Slot/
Unit          Type  Ranges      CAM  Rules      Counters      Meters
              Total/Free  Total/Free  Total/Free  Total/Free
1/1/(0)       IFP   32/32       0    128/128    128/128      128/128
              1    128/127    128/127
              2    128/125    128/125
              3    128/127    128/127
              4    256/256    256/256
              5    256/256    256/256
              6    256/255    256/255
              7    256/255    256/256
              8    256/255    256/254
              9    256/255    256/256
1/1/(0)       EFP   0/0         0    256/256    256/256      256/256
              1    256/256    256/256
              2    256/256    256/256
              3    256/256    256/256
256/256</pre>

```

Combination of policy rules leading to multiple TCAM slices consumption

Rules of consumption

The number of configurable policies can be reduced due to the TCAM slice allocation. As explained before, there are 4 TCAM slices available (remaining 4 slices are reserved) with 256 entries on each slice for user configuration. Usually, while creating policy rules the system allocates the TCAM entries on the same slice until the 256 entries are used then it moves to the next slice and so on up to exhaustion of all available entries.



Warning:

Source slot-port rules and destination slot-port rules cannot use the same slice. L2 rules and L4 rules cannot use the same slice. Source IPv6 and destination IPv6 rules cannot use the same slice.

Example of a mix of rules consuming different slices

Configuration

```
policy condition c1 source vlan 559 source ipv6 2001:4cd0:bc00:2570:1::1
destination tcp-port 80
policy condition c2 source vlan 519 destination ipv6 2001:4cd0:bc00:2570:1::1
source tcp-port 80
policy action a1
policy rule r1 condition c1 action a1
policy rule r2 condition c2 action a1
qos apply
```

TCAM Utilization

Source IPv6 rules and destination IPv6 rules cannot use the same slice pair. FPSs (Field Processing Selectors) in the slice pairs are configured differently. A single slice pair cannot hold both rules r1 and r2, as a result these rules are programmed in different slice pairs and each consumes 1 TCAM rule.

```
-> show qos slice
```

Slot/ Unit	Type	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/1/(0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/255	256/255	256/256
			5	256/255	256/256	256/256
			6	256/255	256/255	256/256
			7	256/255	256/256	256/256
			8	256/255	256/254	256/254
1/1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

TCAM exhausted when hitting the system limitation, the importance of the policy rule precedence

The number of rules available on the system can be exhausted if you hit the system limitation (rules programmed on different slices or policy network group rules are not optimized).

Example 1: TCAM exhausted when the default precedence is used

Source IPv6 rules and destination IPv6 rules cannot use the same slice pair: The following combination of source IPv6 rules and destination IPv6 rules forces the system to program each entry in different slice pair because the user is letting the system using the default precedence order.



Warning:

The rule precedence is based on the order in which the rule entry is entered or by defining the precedence in the rule. If precedence is not specified, rule entered first will have higher precedence.

Configuration

```
policy condition c1 source vlan 559 source ipv6 2001:4cd0:bc00:2570:1::1
destination tcp-port 80
policy condition c2 source vlan 519 destination ipv6 2001:4cd0:bc00:2570:1::1
source tcp-port 80
policy condition c3 source vlan 559 source ipv6 2001:4cd0:bc00:2570:1::2
destination tcp-port 80
policy action a1
policy rule r1 condition c1 action a1
policy rule r2 condition c2 action a1
policy rule r3 condition c3 action a1
ERROR: Out of TCAM processors on 1/0(0)
```

TCAM Utilization

Rule 3 cannot be configured on the system because rules 1 and 2 are already using every 4 slices available(2 slice pairs), so rule 3 cannot use the same slice pair as rule 2. The switch returns an error stating that the system is out of TCAM processors.

Example 2: TCAM exhausted because of a manually misconfigured precedence order

The following mix of source IPv6 rules and destination IPv6 rules will cause the TCAM to allocate 1 entry per slice pair because of the precedence order. The 3th rule cannot be programmed into the hardware because no slice is available.

Configuration

```
policy condition c1 source vlan 559 source ipv6 2001:4cd0:bc00:2570:1::1
destination tcp-port 80
policy condition c2 source vlan 519 destination ipv6 2001:4cd0:bc00:2570:1::1
source tcp-port 80
policy condition c3 source vlan 559 source ipv6 2001:4cd0:bc00:2570:1::2
destination tcp-port 80
policy action a1
policy rule r1 condition c1 action a1 precedence 100
policy rule r2 condition c2 action a1 precedence 110
policy rule r3 condition c3 action a1 precedence 120
ERROR: Out of TCAM processors on 1/0(0)
```

TCAM Utilization

Rule 3 cannot be configured on the system because rules 1 and 2 are already using every 4 slices available(2 slice pairs), so rule 3 cannot use the same slice pair as rule 2. The switch returns an error stating that the system is out of TCAM processors.

Example 3: Similar to example 1 and 2 with an optimized and working configuration – Efficient usage of the Precedence

Configuration

```

policy condition c1 source vlan 559 source ipv6 2001:4cd0:bc00:2570:1::1
destination tcp-port 80
policy condition c2 source vlan 519 destination ipv6 2001:4cd0:bc00:2570:1::1
source tcp-port 80
policy condition c3 source vlan 559 source ipv6 2001:4cd0:bc00:2570:1::2
destination tcp-port 80
policy action a1
policy rule r1 condition c1 action a1 precedence 110
policy rule r2 condition c2 action a1 precedence 100
policy rule r3 condition c3 action a1 precedence 120
qos apply

```

TCAM Utilization

This configuration will use slice pair 6_7 for the source IPv6 rules and slice pair 4_5 for the destination IPv6 rule.

This configuration will accomplish the same purpose as example 1 and example 2 while consuming only 4 slices while the 2 previous examples were consuming too many slices and were not supported on the system.

-> show qos slice

Slot/ Unit	Types	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/1/(0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/255	256/255	256/256
			5	256/255	256/256	256/256
			6	256/254	256/254	256/256
			7	256/254	256/256	256/256
			8	256/255	256/254	256/254
1/1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Layer 4, TCP/UDP ports, service groups and port ranges

A single TCP/UDP port rule will consume one TCAM rule. TCP/UDP port ranges consume one or multiple TCAM rules entries



Warning:

depending on the range. 8 Hardware TCP/UDP ranges are available and automatically used instead of the regular TCAM rules when the range is supposed to consume 6 or more than 6 rules.

The Classifier Processor on the ASIC has a separate table with a capacity of 8 TCP/UDP port ranges per TCAM. Each port range will consume one TCAM entry and we can have 8 rules which use the TCP/UDP port range table. However, the user can configure more than 8 TCP/UDP port ranges, additional TCP/UDP port ranges consuming more than 5 TCAM rules are programmed to the TCAM using multiple TCAM entries.

Hardware TCP/UDP port ranges are only allocated for TCP/UDP port ranges that require 6 or more than 6 regular TCAM entries. TCP/UDP port ranges that can be programmed directly to the TCAM using less than 6 TCAM entries will not consume a hardware range table entry.

Understanding the TCAM rule consumption for Layer 4 rules and TCP/UDP port ranges

Source and destination ports are 2 bytes long fields in the TCP/UDP headers. To understand the rules consumption you need to convert the TCP/UDP value from decimal to binary and check which mask to apply to fit to the port or port range, depending on the range a single may cover multiple values:

```
Single Port: 80 (decimal) -> (binary)      80 =      00000000
01010000                                     value    00000000
                                           mask     11111111
01010000
11111111
```

Consumes 1 TCAM rule

```
Port Range: 2-3 (decimal) -> (binary)      2 =      00000000 00000010
                                           3 =      00000000 00000011
                                           value    00000000 00000010
                                           mask     11111111 11111110
```

Consumes 1 TCAM rule

In this example, ports 2 and 3 can use the same mask. The first 15 bits for port 2 and port 3 are identical.

```
Port Range: 2-4 (decimal) -> (binary)      2 =      00000000 00000010
                                           3 =      00000000 00000011
                                           4 =      00000000 00000100
```

Consumes 2 TCAM rules

One single rule is used to perform a match when the port number is equal 2 or 3 since both values share a common mask:

```
TCAM rule 1                                value1  00000000 00000010 (match port 2-
3)                                           mask1   11111111 11111110 (mask port 2-3)
```

One additional TCAM rule is used to match when the port number equals to 4:

```
TCAM rule 2                                value2  00000000 00000100 (match port 4)
                                           mask2   11111111 11111111 (mask port 4)
```

In this example, it is not possible to find a single mask to cover port 2, 3 and 4. As a result the switch will consume 2 TCAM rules.

TCAM entries consumption examples for TCP/UDP port range

- Source TCP port 0-10 consumes 3 TCAM rules
- Source TCP port 1-10 consumes 5 TCAM rules
- Source TCP port 0-32 consumes 2 TCAM rules
- Source TCP port 1-32 is supposed to consume 6 TCAM, so 1 hardware TCP/UCP range is used in combination with only 1 TCAM rule
- Source TCP port 0-65535 consumes 1 TCAM rule
- Source TCP port 1-65535 is supposed to consume 16 TCAM rules, so 1 hardware TCP/UCP range is used in combination with only 1 TCAM rule

Example 1: Layer 4 – a single port

Configuration

```
policy service http destination tcp 80
policy condition c1 service http
policy action a1 disposition accept
policy rule r1 condition c1 action a1
qos apply
```

Consumes 1 TCAM rule on the NI

Explanation

```
Single Port: 80 (decimal) -> (binary)      80 =      00000000 01010000
                                             value    00000000 01010000
                                             mask     11111111 11111111
```

In order to match this value the rule has to do an exact match for port 80

Example 2: Layer 4 – a port range

Configuration

```
policy condition c1 source tcp 1-2
policy action a1 disposition accept
policy rule r1 condition c1 action a1
qos apply
```

Will consume 2 TCAM rules on the NI

Explanation

```
Port Range: 1-2 (decimal) -> (binary)    1 =      00000000 00000001
                                             2 =      00000000 00000010
```

One single rule is used to match when the port number is equal 1:

```
TCAM rule 1                               value1  00000000 00000001 (match port 1)
                                             mask1   11111111 11111111 (full mask)
```

Another rule is used to match when the port number is equal 2:

```
TCAM rule 2                               value2  00000000 00000010 (match port 2)
                                             mask2   11111111 11111111 (full mask)
```

Example 3: Layer 4 – a port range

Configuration

```
policy condition c1 source tcp 1-10
policy action a1 disposition accept
policy rule r1 condition c1 action a1
qos apply
```

Will consume 5 TCAM rules on the NI

Explanation

In order to understand this example, you have to convert each TCP port decimal value to its binary value, see below:

1	(decimal) -> (binary)	0000 0001	<-	Rule 1
2	(decimal) -> (binary)	0000 0010	<-	Rule 2
3	(decimal) -> (binary)	0000 0011		
4	(decimal) -> (binary)	0000 0100	<-	Rule 3
5	(decimal) -> (binary)	0000 0101		
6	(decimal) -> (binary)	0000 0110		
7	(decimal) -> (binary)	0000 0111		
8	(decimal) -> (binary)	0000 1000	<-	Rule 4
9	(decimal) -> (binary)	0000 1001		
10	(decimal) -> (binary)	0000 1010	<-	Rule 5

```
-> show qos slice
```

Slot/ Unit	Type	Ranges Total/Free	Rules CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/(0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/256	256/256	256/256
			7	256/251	256/251	256/256
			8	256/255	256/254	256/254
			9	256/255	256/256	256/256
1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Example 4: Hardware TCP/UDP port range is used if more than 6 TCAM are supposed to be consumed by the port range

Configuration

```
policy condition c1 source tcp 1-32
policy action a1 disposition accept
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

This rule would consume 6 TCAM rules, so the system automatically uses 1 TCP/UDP Hardware Range and 1 TCAM rule on the NI.

Explanation

When the policy rule is consuming more than 5 regular TCAM rules the system automatically allocates a dedicate hardware TCP/UDP range for this policy so that it only consumes 1 TCAM rule, this allows the system to save TCAM rules for other rules.

-> show qos slice

Slot/ Unit	Type	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/(0)	IFP	32/31	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/256	256/256	256/256
			7	256/255	256/255	256/256
			8	256/255	256/254	256/254
1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Example of a more complex rule

Configuration

```
policy service SERV1 destination tcp 80
policy service SERV2 destination tcp 21
policy service group SERV_GRP SERV1 SERV2
policy port group Dest_Port_group 1/1 1/12 1/20
policy network group SRCNet1 12.12.12.1 12.12.12.2 12.12.12.3 12.12.12.4
12.12.12.5
policy network group DSTNet1 22.21.21.1 22.21.21.2 22.21.21.3 22.21.21.4
22.21.21.5 22.21.21.10
policy condition CC1 destination port group Dest_Port_group source network group
SRCNet1 destination network group DSTNet1 service
group SERV_GRP
policy action AA1 disposition drop
policy rule RR1 condition CC1 action AA1
qos apply
```

TCAM Utilization

180 = (2 services • 3 egress ports • 5 source networks • 6 destination networks)

180 rules are consumed on each NI in the system

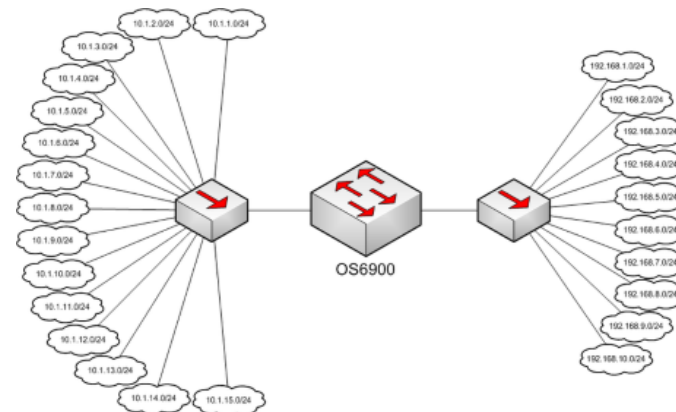
-> show qos slice

Slot/ Unit	Type	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/(0)	IFP	32/31	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/76	256/76	256/256
			7	256/255	256/255	256/256
			8	256/255	256/254	256/254
1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Case Study

Overview and objective

Drop any traffic from LAN A made of 15 networks to LAN B made of 10 networks.



Solution 1: QoS rules not optimized leading to unnecessary TCAM entries consumption

An easy way to achieve the objective would be to:

- Create a policy network group for LAN A, specifying 15 different /24 networks
- Create a policy network group for LAN B, specifying 10 different /24 networks
- Create a policy condition with the source IP networks of LAN A and destination IP networks of LAN B
- Create a policy action to drop the traffic
- Create a policy rule based on the previously configured policy action and condition

However, considering what has been previously described “Combining source and destination groups consumes one TCAM rule for every combination between source and destination entries”, such configuration would consume 150 TCAM entries.

Configuration

```
policy action a1 disposition drop
policy network group LanA 10.1.1.0 mask 255.255.255.0
policy network group LanA 10.1.2.0 mask 255.255.255.0
policy network group LanA 10.1.3.0 mask 255.255.255.0
policy network group LanA 10.1.4.0 mask 255.255.255.0
policy network group LanA 10.1.5.0 mask 255.255.255.0
policy network group LanA 10.1.6.0 mask 255.255.255.0
policy network group LanA 10.1.7.0 mask 255.255.255.0
policy network group LanA 10.1.8.0 mask 255.255.255.0
policy network group LanA 10.1.9.0 mask 255.255.255.0
policy network group LanA 10.1.10.0 mask 255.255.255.0
policy network group LanA 10.1.11.0 mask 255.255.255.0
policy network group LanA 10.1.12.0 mask 255.255.255.0
policy network group LanA 10.1.13.0 mask 255.255.255.0
policy network group LanA 10.1.14.0 mask 255.255.255.0
policy network group LanA 10.1.15.0 mask 255.255.255.0
policy network group LanB 192.168.1.0 mask 255.255.255.0
policy network group LanB 192.168.2.0 mask 255.255.255.0
policy network group LanB 192.168.3.0 mask 255.255.255.0
policy network group LanB 192.168.4.0 mask 255.255.255.0
policy network group LanB 192.168.5.0 mask 255.255.255.0
policy network group LanB 192.168.6.0 mask 255.255.255.0
policy network group LanB 192.168.7.0 mask 255.255.255.0
policy network group LanB 192.168.8.0 mask 255.255.255.0
policy network group LanB 192.168.9.0 mask 255.255.255.0
policy network group LanB 192.168.10.0 mask 255.255.255.0
policy condition c1 source network group LanA destination network group LanB
policy rule r1 condition c1 action a1
qos apply
```


TCAM Utilization

-> show qos slice

Slot/ Unit	Type	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/(0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/256	256/256	256/256
			7	256/116	256/116	256/256
			8	256/255	256/254	256/254
1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Solution 2: Network summarization

This solution is not always applicable and depends on the overall network design.

Summarizing the source network addresses and/or destination network addresses is a possibility to achieve the same objective.

Configuration

```
policy network group LanA 10.1.0.0 mask 255.255.0.0
policy network group LanB 192.168.0.0 mask 255.255.0.0
policy condition c1 source network group LanA destination network group LanB
policy action a1 disposition drop
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

Consumes only 1 TCAM rule on the NI.

-> show qos slice

Slot/ Unit	Type	Ranges Total/Free	CAM	Rules Total/Free	Counters Total/Free	Meters Total/Free
1/(0)	IFP	32/32	0	128/128	128/128	128/128
			1	128/127	128/127	128/128
			2	128/125	128/125	128/128
			3	128/127	128/127	128/128
			4	256/256	256/256	256/256
			5	256/256	256/256	256/256
			6	256/256	256/256	256/256
			7	256/255	256/255	256/256
			8	256/255	256/254	256/254
1/(0)	EFP	0/0	0	256/256	256/256	256/256
			1	256/256	256/256	256/256
			2	256/256	256/256	256/256
			3	256/256	256/256	256/256

Solution 3: Specifying a source port and/or destination port

This solution is not always applicable and depends on the overall network design.

a) Source Port, Destination Port

The following rule drops the traffic coming through port 1/1 and going out through port 1/2.

```
policy condition c1 source port 1/1 destination port 1/2
policy action a1 disposition drop
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

Consumes 1 TCAM rule only.

b) Source port, Destination IPs

The following rule drops the traffic from port 1/1 which has a destination IP in LAN B (networks 192.168.1.0/24 to 192.168.10.0/24).

```
policy network group LanB 192.168.1.0 mask 255.255.255.0
...
policy network group LanB 192.168.10.0 mask 255.255.255.0
policy condition c1 source port 1/1 destination network group LanB
policy action a1 disposition drop
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

Consumes 10 TCAM rules.

c) Source IPs, Destination Port

The following configuration drops the traffic from the source network group LanA (networks 10.1.1.0/24 to 10.1.15.0/24) and destination port 1/2.

```
policy network group LanA 10.1.1.0 mask 255.255.255.0
...
policy network group LanA 10.1.15.0 mask 255.255.255.0
policy condition c1 source network group LanA destination port 1/2
policy action a1 disposition drop
policy rule r1 condition c1 action a1
qos apply
```

TCAM Utilization

Consumes 15 TCAM rules.

Additional Information

- Following message is displayed when a configured rule would exceed the system capacity:

```
ERROR: Out of TCAM processors on 1/0(0)
```

- To determine how many rules and masks are being used by the system use “show qos slice” or “show qos slice slot/port” commands (available from the CLI).

2.2.3. Minimum working configuration

```
qos enable
qos apply
```

2.2.4. Troubleshooting

- Following message is displayed when a configured rule would exceed the system capacity:

```
ERROR: Out of TCAM processors on 1/0(0)
```

- To determine how many rules and masks are being used by the system use "show qos slice" or "show qos slice slot/port" commands (available from the CLI).
- In case of heavy traffic matching a rule with "log" enable, it is possible to observe packet loss

2.2.5. Advanced troubleshooting

Display QoS rules configured in TCAM

All lists:

```
-> debug qos internal "slot 1 list 255 verbose"
          Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]          Red[+]          NotGreen[+]
List 0 empty
List 1: 19 entries set up
      McastARP( 17) 0      8 1543      -      -      - 1636      0[0      ]
0[0      ]          0[0      ]          0[0
      McastARP( 17) 0      9      -      -      -      - 1892      0[0      ]
0[0      ]          0[0      ]          0[0
      ISIS_BPDU1( 22) 0      8      - 1536      -      - 1590      10738[10738      ]
0[0      ]          0[0      ]          0[0
...

```

List 0 (user rules):

```
-> debug qos internal "slot 1 list 0 verbose"
          Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]          Red[+]          NotGreen[+]
List 0: 1 entries set up
      Policy( 0) 0      7      - 1280      -      - 1280      0[0      ]
0[0      ]          0[0      ]          0[0

```

List 1 (all copy to CPU):

```
-> debug qos internal "slot 1 list 1 verbose"
          Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]          Red[+]          NotGreen[+]
List 1: 20 entries set up
      HgMcastARP( 16) 0      -1      -      -      0      0      -2      0[0      ]
0[0      ]          0[0      ]          0[0
      McastARP( 17) 0      8 1543      -      -      - 1636      0[0      ]
0[0      ]          0[0      ]          0[0
      McastARP( 17) 0      9      -      -      -      - 1892      0[0      ]
0[0      ]          0[0      ]          0[0
      ISIS_BPDU1( 22) 0      8      - 1536      -      - 1590      0[0      ]
0[0      ]          0[0      ]          0[0
      ISIS_BPDU1( 22) 0      9      -      -      -      - 1846      0[0      ]
0[0      ]          0[0      ]          0[0
      ISIS_BPDU2( 23) 0      8 1537      -      -      - 1591      0[0      ]

```

0[0]	0[0]	0[0					
ISIS_BPDU2(23)	0	9	-	-	-	-	1847	0[0
]		0[0]]
ISIS_BPDU3(24)	0	8	-	1538	-	-	1592	0[0
]		0[0]]
ISIS_BPDU3(24)	0	9	-	-	-	-	1848	0[0
]		0[0]]
IPMS_IGMP(50)	0	8	1539	-	-	-	1638	576[0
]		0[0]]
IPMS_IGMP(50)	0	9	-	-	-	-	1894	576[0
]		0[0]]
IPMS_V4Control(51)	0	8	-	1540	-	-	1639	1829[5
]		0[0]]
IPMS_V4Control(51)	0	9	-	-	-	-	1895	1829[0
]		0[0]]
IPMS_V4Data(52)	0	8	1541	-	-	-	1640	0[0
]		0[0]]
IPMS_V4Data(52)	0	9	-	-	-	-	1896	0[0
]		0[0]]
IPMS_V4Resolved(53)	0	8	-	1542	-	-	1641	0[0
]		0[0]]
IPMS_V4Resolved(53)	0	9	-	-	-	-	1897	0[0
]		0[0]]
ETHOAM_SYS(62)	0	8	-	1562	-	-	1741	0[0
]		0[0]]
ETHOAM_SYS(62)	0	9	-	-	-	-	1997	0[0
]		0[0]]
802.1ab Regular(102)	0	8	1547	-	-	-	1586	8[1
]		0[0]]
802.1ab Regular(102)	0	9	-	-	-	-	1842	8[0
]		0[0]]
amap Regular(103)	0	8	-	1546	-	-	1587	3[1
]		0[0]]
amap Regular(103)	0	9	-	-	-	-	1843	3[0
]		0[0]]
802.3ad Regular(104)	0	8	-	1548	-	-	1588	257[47
]		0[0]]
802.3ad Regular(104)	0	9	-	-	-	-	1844	257[0
]		0[0]]
802.1x Regular(105)	0	8	1549	-	-	-	1589	0[0
]		0[0]]
802.1x Regular(105)	0	9	-	-	-	-	1845	0[0
]		0[0]]
BPDU Regular(106)	0	8	-	1550	-	-	1584	27022[67
]		0[0]]
BPDU Regular(106)	0	9	-	-	-	-	1840	27022[0
]		0[0]]
MPLS(120)	0	8	-	1544	-	-	1543	0[0
]		0[0]]
MPLS(120)	0	9	-	-	-	-	1799	0[0
]		0[0]]
srcsldrop(128)	0	8	1551	-	-	-	1536	0[0
]		0[0]]
srcsldrop(128)	0	9	-	-	-	-	1792	0[0
]		0[0]]
Static Mac Move(136)	0	8	1553	1552	0	1	1750	0[0
]		0[0]]
Static Mac Move(136)	0	9	-	-	-	-	2006	0[0
]		0[0]]
MIM(143)	0	8	1545	-	-	-	1537	0[0
]		0[0]]
MIM(143)	0	9	-	-	-	-	1793	0[0
]		0[0]]
LINKOAMSAA(161)	0	8	1561	-	-	-	1742	0[0
]]

```

0[0      ]          0[0      ]          0[0      ]
LINKOAMSAA( 161) 0      9      -      -      -      - 1998      0[0      ]
0[0      ]          0[0      ]          0[0      ]

```

List 2:

```

-> debug qos internal "slot 1 list 2 verbose"
Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]          Red[+]          NotGreen[+]
List 2: 6 entries set up
IPMS_MLD( 78) 0      2 257 - - - 258      0[0      ]
0[0      ]          0[0      ]          0[0      ]
IPMS_V6Control( 81) 0      2 - 258 - - 259      0[0      ]
0[0      ]          0[0      ]          0[0      ]
IPMS_V6Data( 82) 0      2 259 - - - 260      0[0      ]
0[0      ]          0[0      ]          0[0      ]
IPMS_V6Resolved( 83) 0      2 - 260 - - 261      0[0      ]
0[0      ]          0[0      ]          0[0      ]
MPLSTrust( 124) 0      2 261 - - - 256      0[0      ]
0[0      ]          0[0      ]          0[0      ]
MIMTrust( 144) 0      2 - 262 - - 257      0[0      ]
0[0      ]          0[0      ]          0[0      ]

```

List 7:

```

-> debug qos internal "slot 1 list 7 verbose"
Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]          Red[+]          NotGreen[+]
List 7: 66 entries set up
PortTrust( 1) 0      2 263 - - - 263      0[0      ]
0[0      ]          0[0      ]          0[0      ]
PortTrust( 2) 0      2 - 264 - - 264      0[0      ]
0[0      ]          0[0      ]          0[0      ]
PortTrust( 3) 0      2 265 - - - 265      0[0      ]
0[0      ]          0[0      ]          0[0      ]
PortTrust( 4) 0      2 - 266 - - 266      0[0      ]
0[0      ]          0[0      ]          0[0      ]
PortTrust( 5) 0      2 267 - - - 267      0[0      ]
0[0      ]          0[0      ]          0[0      ]
PortTrust( 6) 0      2 - 268 - - 268      0[0      ]
0[0      ]          0[0      ]          0[0      ]
...

```

List 9 (phones auto QoS):

```

-> debug qos internal "slot 1 list 8 verbose"
Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]          Red[+]          NotGreen[+]
List 8: 7 entries set up
AutoPhone( 0) 0      8 - 1554 - - 1679      0[0      ]
0[0      ]          0[0      ]          0[0      ]
AutoPhone( 0) 0      9 - - - - 1935      0[0      ]
0[0      ]          0[0      ]          0[0      ]
AutoPhone( 1) 0      8 1555 - - - 1680      0[0      ]
0[0      ]          0[0      ]          0[0      ]
AutoPhone( 1) 0      9 - - - - 1936      0[0      ]
0[0      ]          0[0      ]          0[0      ]
AutoPhone( 2) 0      8 - 1556 - - 1681      0[0      ]
0[0      ]          0[0      ]          0[0      ]
AutoPhone( 2) 0      9 - - - - 1937      0[0      ]
0[0      ]          0[0      ]          0[0      ]
AutoPhone( 3) 0      8 1557 - - - 1682      0[0      ]
0[0      ]          0[0      ]          0[0      ]

```

...


List 12 (L3 features, all copy to CPU):

```

-> debug qos internal "slot 1 list 12 verbose"
      Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]      Red[+]      NotGreen[+]
List 12: 16 entries set up
  ospf( 64) 0      3      - 384      -      - 448      1838[1838      ]
0[0      ]      0[0      ]
  vrrp( 65) 0      3 385      -      - 449      0[0      ]
0[0      ]      0[0      ]
  icmp( 66) 0      3      - 386      -      - 450      0[0      ]
0[0      ]      0[0      ]
  rip( 67) 0      3 387      -      - 451      0[0      ]
0[0      ]      0[0      ]
  bgpsrc( 68) 0      3      - 388      -      - 452      1766[1766      ]
0[0      ]      0[0      ]
  bgpdst( 69) 0      3 389      -      - 453      0[0      ]
0[0      ]      0[0      ]
  bfdecho( 70) 0      3      - 390      -      - 454      0[0      ]
0[0      ]      0[0      ]
  bfdreply( 71) 0      3 391      -      - 455      0[0      ]
0[0      ]      0[0      ]
  telnet( 72) 0      3      - 392      -      - 456      0[0      ]
0[0      ]      0[0      ]
  ssh( 73) 0      3 393      -      - 457      0[0      ]
0[0      ]      0[0      ]
  http( 74) 0      3      - 394      -      - 458      0[0      ]
0[0      ]      0[0      ]
  snmp( 75) 0      3 395      -      - 459      0[0      ]
0[0      ]      0[0      ]
  arp( 76) 0      3      - 396      -      - 460      115[115      ]
0[0      ]      0[0      ]
  ripng( 77) 0      3 397      -      - 461      0[0      ]
0[0      ]      0[0      ]
  pim( 78) 0      3      - 398      -      - 462      0[0      ]
0[0      ]      0[0      ]
  mcipc( 82) 0      3 399      -      - 463      183182[183182      ]
0[0      ]      0[0      ]

```

2.2.6. Troubleshooting in Maintenance Shell

 **Warning:** Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

VFC Troubleshooting

Swlog is extremely important to trace any VFC-related issues. All VFC-related swlog is stored in the /var/log directory called "vfc1.log". Note that the swlog is cleared every time the OmniSwitch 6900 or 10K is rebooted.

For example if VFC error messages appeared on the console and you want to see more details as to what happened during that time, login as super user "su", cd to the /var/log directory, open the vfc.log file, and search for that particular timestamp when the error event happened. To troubleshoot VFC in real-time while having the console connection active, open a telnet session and do the following as indicated below:

```

#-> cd /var/log
#-> ls
fdl.log ipms.log lag.log mcm.log vfc1.log vm.log vstk.log wtmp
#-> tail -f vfc1.log
19 16:49:04 - dbg: [vfcInitSlotProfile:249] Create Transaction buffer vfcTxBuf[0
19 16:49:04 - dbg: [vfcInitSlotProfile:254] zNI 0, profiling done
19 16:49:04 - dbg: [vfcInitSlotProfile:249] Create Transaction buffer vfcTxBuf[1
19 16:49:04 - dbg: [vfcInitSlotProfile:254] zNI 1, profiling done
19 16:49:04 - dbg: [vfcConnectToCS:52] VFC Connect to CS
19 16:49:04 - dbg: [vfcConnectToPM:602] VFC Connected to PM
19 16:49:04 - dbg: [main:407] ==generated MIB database==
19 16:49:04 - dbg: [main:410] VFC cslib_unblock
19 16:49:04 - dbg: [vfcMainLoop:301] vfcMainLoop
19 16:49:04 - dbg: [vfcHandleMipMsg:380] Queuing the MIP message
19 16:49:04 - dbg: [getQsapRangeFromIfIndex:1920] Before EOIC: received PORT qsa
19 16:49:04 - dbg: [getQsapRangeFromIfIndex:1920] Before EOIC: received PORT qsa
19 16:49:04 - dbg: [vfc_qsap_control_prop:2185] Invalid QSI 719 16:49:04 - dbg:
19 16:49:04 - dbg: [vfcHandleMipMsg:380] Queuing the MIP message
19 16:49:04 - dbg: [getQsapRangeFromIfIndex:1907] Before EOIC: received LAG qsap
19 16:49:04 - dbg: [vfcHandleMipMsg:380] Queuing the MIP message
19 16:49:04 - dbg: [vfcMipEoicFunction:268] EOIC received
19 16:50:15 - dbg: [vfcAddNewConnection:256] New connection: 127.2.65.1:37985, S
19 16:50:15 - dbg: [vfcHandleIncomingMsg:140] NI 0 connected after NI DOWN, sock
19 16:50:15 - dbg: [vfcHandleIncomingMsg:147] RX VFC_MSG_HELLO zNi 0 BOOTUP
19 16:50:15 - dbg: [vfcPMEventsRegister:575] Port Manager Registrations Done

```

2.3. Datacenter Bridging

Maximum Link Length

If 10Km FCoE link is reliable I would say that it would be lossless with the proper buffer configured for it(linkdelay). Looking into the standard FC-BB-5 (there is a newest version but I have not read it), the standard does limit transit delay to 500ms at: The Lossless Ethernet bridges used to build a Lossless Ethernet network suitable for FC-BB_E usage should enforce a maximum bridge transit delay of 500 ms (see IEEE 802.1Q-2005), according to the best practice used by Fibre Channel switches. So it should be reliable and lossless. All of the timers for FC-BB_E (FCoE) are around ms and your worst RTT of the 10Kmts are 100microseconds. So I would say there is no problem with such a distance in theory. But reading the actual implementation of the 7.3.2 MIB the linkdelay object is defined as:

```

alaDcbxDCPTCPFCLinkDelay OBJECT-TYPE
    SYNTAX      Unsigned32 (0 | 10..100)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "PFC link delay allowance.
        Default for a lossy TC is 0 and cannot be any other value.
        Default for a lossless TC is 52 and can be changed in a valid range
(10-100)."
```

```

 ::= { alaDcbxDCPTCEnt 6 }
```

It means that it was capped at 100KB. By my excel file that I had of the 6900 calculation it is going to give you max around 1500 mts. You better ask the PLMs in order to assure that we can go actually further in distance with lossless. Regarding 10K I am not sure how the implementation is so no comment on it.

Example DCBX packet sent by AOS switch



Note: Wireshark version 1.10.5 doesn't have DCBX dissector. This dissection was prepared using a dedicated Wireshark version.

```
Ethernet II, Src: e8:e7:32:07:98:80 (e8:e7:32:07:98:80), Dst: LLDP_Multicast
(01:80:c2:00:00:0e)
  Destination: LLDP_Multicast (01:80:c2:00:00:0e)
  Address: LLDP_Multicast (01:80:c2:00:00:0e)
  .... ..1 .... = IG bit: Group address
(multicast/broadcast)
  .... ..0. .... = LG bit: Globally unique address (factory
default)
  Source: e8:e7:32:07:98:80 (e8:e7:32:07:98:80)
  Address: e8:e7:32:07:98:80 (e8:e7:32:07:98:80)
  .... ..0 .... = IG bit: Individual address (unicast)
  .... ..0. .... = LG bit: Globally unique address (factory
default)
  Type: 802.1 Link Layer Discovery Protocol (LLDP) (0x88cc)
Link Layer Discovery Protocol
  Chassis Subtype = MAC address, Id: e8:e7:32:07:98:79
  0000 001. .... = TLV Type: Chassis Id (1)
  .... ..0 0000 0111 = TLV Length: 7
  Chassis Id Subtype: MAC address (4)
  Chassis Id: e8:e7:32:07:98:79 (e8:e7:32:07:98:79)
  Port Subtype = Locally assigned, Id: 1001
  0000 010. .... = TLV Type: Port Id (2)
  .... ..0 0000 0101 = TLV Length: 5
  Port Id Subtype: Locally assigned (7)
  Port Id: 1001
  Time To Live = 120 sec
  0000 011. .... = TLV Type: Time to Live (3)
  .... ..0 0000 0010 = TLV Length: 2
  Seconds: 120
  IEEE 802.1 - ETS Configuration
  1111 111. .... = TLV Type: Organization Specific (127)
  .... ..0 0001 1001 = TLV Length: 25
  Organization Unique Code: IEEE 802.1 (0x0080c2)
  IEEE 802.1 Subtype: ETS Configuration (0x09)
  0... .. = Willing: Not Enabled
  .0... .. = CBS: Not Enabled
  Max Traffic Classes: 0
  Traffic Class to Priority Mapping: 0x01234567
  Priority 0: 0
  Priority 1: 1
  Priority 2: 2
  Priority 3: 3
  Priority 4: 4
  Priority 5: 5
  Priority 6: 6
  Priority 7: 7
  Traffic Class Bandwidth Table: 0x0C0C0C0C0D0D0D0D
  Traffic Class 1: 12
  Traffic Class 2: 12
  Traffic Class 3: 12
  Traffic Class 4: 12
  Traffic Class 5: 13
  Traffic Class 6: 13
  Traffic Class 7: 13
  Traffic Class 8: 13
  Traffic Selection Algorithm Table: 0x0202020202020202
  Traffic Class 1: 2
  Traffic Class 2: 2
```



```

    Traffic Class 3: 2
    Traffic Class 4: 2
    Traffic Class 5: 2
    Traffic Class 6: 2
    Traffic Class 7: 2
    Traffic Class 8: 2
IEEE 802.1 - ETS Recommendation
1111 111. .... .... = TLV Type: Organization Specific (127)
.... ...0 0001 1001 = TLV Length: 25
Organization Unique Code: IEEE 802.1 (0x0080c2)
IEEE 802.1 Subtype: ETS Recommendation (0x0a)
Traffic Class to Priority Mapping: 0x01234567
    Priority 0: 0
    Priority 1: 1
    Priority 2: 2
    Priority 3: 3
    Priority 4: 4
    Priority 5: 5
    Priority 6: 6
    Priority 7: 7
Traffic Class Bandwidth Table: 0x0C0C0C0C0D0D0D0D
    Traffic Class 1: 12
    Traffic Class 2: 12
    Traffic Class 3: 12
    Traffic Class 4: 12
    Traffic Class 5: 13
    Traffic Class 6: 13
    Traffic Class 7: 13
    Traffic Class 8: 13
Traffic Selection Algorithm Table: 0x0202020202020202
    Traffic Class 1: 2
    Traffic Class 2: 2
    Traffic Class 3: 2
    Traffic Class 4: 2
    Traffic Class 5: 2
    Traffic Class 6: 2
    Traffic Class 7: 2
    Traffic Class 8: 2
IEEE 802.1 - Priority-based Flow Control Configuration
1111 111. .... .... = TLV Type: Organization Specific (127)
.... ...0 0000 0110 = TLV Length: 6
Organization Unique Code: IEEE 802.1 (0x0080c2)
IEEE 802.1 Subtype: Priority-based Flow Control Configuration (0x0b)
0... .... = Willing: Not Enabled
.0.. .... = MBC: Not Enabled
PFC Capability: 8
PFC Enable Table: 0xFF
    .... ...1 = Priority 1: Enabled
    .... ..1. = Priority 2: Enabled
    .... .1.. = Priority 3: Enabled
    .... 1... = Priority 4: Enabled
    ...1 .... = Priority 5: Enabled
    ..1. .... = Priority 6: Enabled
    .1.. .... = Priority 7: Enabled
    1... .... = Priority 8: Enabled
IEEE 802.1 - Application Priority
1111 111. .... .... = TLV Type: Organization Specific (127)
.... ...0 0000 1000 = TLV Length: 8
Organization Unique Code: IEEE 802.1 (0x0080c2)
IEEE 802.1 Subtype: Application Priority (0x0c)
Application 1 Priority: 4
Application 1 Selection: 2
Application 1 Selection: 0CBC
End of LLDPDU

```

```
0000 000. .... .... = TLV Type: End of LLDPDU (0)
.... ...0 0000 0000 = TLV Length: 0
```

2.3.1. Minimum working configuration

Intel X520-2



Note: AOS 7.3.1 requires different way of configuring iSCSI application TLV:
"debug lldp port 1/1 application iscsi priority 4"

Tested with driver version 18.8.1:

```
-> show configuration snapshot vfc lldp qos
! Virtual Flow Control:
qos qsi port 1/1 dcb dcbx pfc willing no
qos qsi port 1/1 qsp dcb "dcp-11"
qos qsi port 1/1 dcb dcbx version cee
qos qsi port 1/1 dcb dcbx ets willing no
qos qsi port 1/1 dcb dcbx pfc defense disable
qos qsi port 1/1 stats admin-state enable interval 10

! LLDP:
lldp nearest-bridge port 1/1 tlv application enable
lldp port 1/1 tlv application iscsi priority 4

! QOS:
qos port 1/1 trusted default classification 802.1p
qos apply
```

Previous driver versions may require additional DCP:

```
qos qsp dcb "dcp-intel" import qsp dcb "dcp-11"
qos qsp dcb "dcp-intel" tc 0 pfc flow-type nLL min-bw 60 recommended bw 60
qos qsp dcb "dcp-intel" tc 1 pfc flow-type nLL min-bw 0 recommended bw 0
qos qsp dcb "dcp-intel" tc 2 pfc flow-type nLL min-bw 0 recommended bw 0
qos qsp dcb "dcp-intel" tc 3 pfc flow-type nLL min-bw 0 recommended bw 0
qos qsp dcb "dcp-intel" tc 4 pfc flow-type nLL min-bw 40 recommended bw 40
qos qsp dcb "dcp-intel" tc 5 pfc flow-type nLL min-bw 0 recommended bw 0
qos qsp dcb "dcp-intel" tc 6 pfc flow-type nLL min-bw 0 recommended bw 0
qos qsp dcb "dcp-intel" tc 7 pfc flow-type nLL min-bw 0 recommended bw 0
qos qsi port 1/1 qsp dcb "dcp-intel"
```

2.3.2. Troubleshooting in Maintenance Shell



Warning:

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

2.4. Port monitoring

pmonitor.enc is automatically created and stored under /flash directory when you activate/de-activate it. You can do a quick check on the console as follows. This is the simple and fastest-way to confirm the packet's source/destination for BPDU, etc.

```
-> port-monitoring 1 source 1/3 enable
```

```
-> port-monitoring 1 disable
```

WARNING:

Monitored data is available in file /flash/pmonitor.enc

```
OS6900-3> show port-monitoring file
```

Destination	Source	Type	Data
01:80:C2:00:00:02	E8:E7:32:30:10:76	II-8100	81:00:00:01:88:09:01:01:01:14
01:80:C2:00:00:14	E8:E7:32:07:A3:E5	II-8100	81:00:CF:A1:00:45:FE:FE:03:83
01:80:C2:00:00:14	E8:E7:32:30:10:6D	II-8100	81:00:CF:A1:00:45:FE:FE:03:83
01:80:C2:00:00:02	E8:E7:32:30:10:76	II-8100	81:00:00:01:88:09:01:01:01:14
E8:E7:32:07:A3:E5	E8:E7:32:30:10:6D	II-8100	81:00:CF:A2:88:E7:C0:00:03:EE
01:80:C2:00:00:00	00:E0:B1:CF:5F:A1	II-8100	81:00:00:00:00:27:42:42:03:00

data file is /flash/pmonitor.enc

Capturing entire packets:

```
-> port-monitoring 1 source 1/1 capture-type full enable timeout 10
```

2.5. Sflow

2.5.1. Minimum working configuration

```
sflow receiver 1 name sflow address 172.26.60.132
```

```
sflow sampler 1 port 1/1/1 receiver 1 rate 128 sample-hdr-size 128
```

2.5.2. Advanced troubleshooting

The 'debug sflow dump statistics' command displays the value of a CMM based statistics counter. Currently, the NI sends out the sflow datagrams. This is why it shows zero.

```
-> debug sflow dump statistic
```

```
====->>>> sFlow datagram information <<<<-=====
```

Slot	datagrams sent	Receiver
EMP	0	1 (sflow)

2.5.3. Troubleshooting in Maintenance Shell



Warning:

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

2.6. OpenFlow

2.7. Checklist

- Verify if the configured controller version is supported by the controller

2.8. Introduction

- No support in AOS 7.3.3.R01, only experimental
- No support in Virtual Chassis
- Hybrid mode supported (ports dedicated for OpenFlow and regular switch)
- Wireshark dissector
 - **Note:** Any Wireshark version $\geq 1.12.0$ has native OpenFlow support / dissector built-in! (Select OpenFlow_v4 for OF v1.3)
- Supported features on OpenFlow ports:
 - L1 counters
 - LLDP (post-GA enhancement)
 - UDLD (post-GA enhancement, only on 1 G ports)
 - LACP (post-GA enhancement)
 - OF set and pop are supported (push is not supported)
- OF 1.0 or OF 1.3 controller support
 - These 2 standards are not compatible with each other
 - A controller may be connected through EMP
 - FloodLight + AVIOR (GUI)
 - OpenSource
 - NTT RYU
 - OpenSource
 - VOLT for advanced troubleshooting (a license is needed)
- OpenFlow reserves 2 slices
 - Table 0 (slice 6 and 7) exact match rules
 - Table 1 (slice 4 and 5) wildcard rules
 - Slices 6 and 7 as well as 4 and 5 are paired
 - Slices 5 and 7 are configured as follows:
 - FPS1 = 0 = SVP_L3_IIF[36:24], FORWARDING_FIELD[23:12], CLASSIDS[11:0]
 - FPS2 = 0 = SIP[127:96], DIP[95:64], IP_PROTOCOL/LAST_NH[63:56], L4_SRC[55:40], L4_DST[39:24], TOS_FN[23:16], IP_FLAG[15:14], TCP_FN[13:8], TTL_FN[7:0]
 - FPS3 = 6 = INNER_VLAN_TAG[31:16], LOOKUP_STATUS[15:0]
 - Slices 4 and 6 are configured as follows:
 - FPS1 = 0 = SVP_L3_IIF[36:24], FORWARDING_FIELD[23:12], CLASSIDS[11:0]
 - FPS2 = 3 = MACDA[127:80], MACSA[79:32], ETHERTYPE[31:16], OUTER_VLAN_TAG[15:0]
 - FPS3 = 5 = FCOE_HEADER_ENCODE_2[37:35],

FCOE_HEADER_ENCODE_1[34:32], ETHERTYPE[31:16],
 OUTER_VLAN_TAG[15:0]

- OF counters planned as post-GA, no IPv6 support, no meters support
- There are 12 fields matched
- No traffic loss when programming new rules

2.8.1. FloodLight and AVIOR example

Example configuration (FloodLight supports only 1.0):

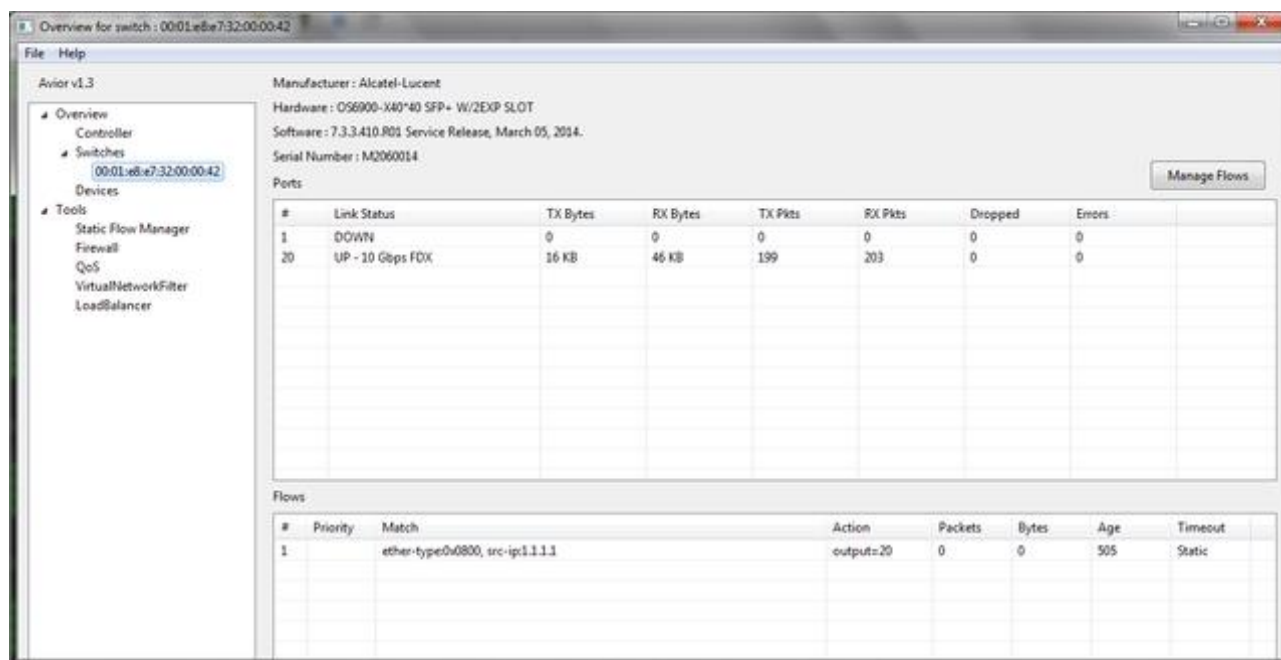
```
OS6900-> show configuration snapshot openflow
! OPENFLOW:
openflow logical-switch vswitch version 1.0 vlan 1000
openflow logical-switch vswitch controller 172.26.60.226:6633
openflow logical-switch vswitch interfaces port 1/1
```

Controller status:

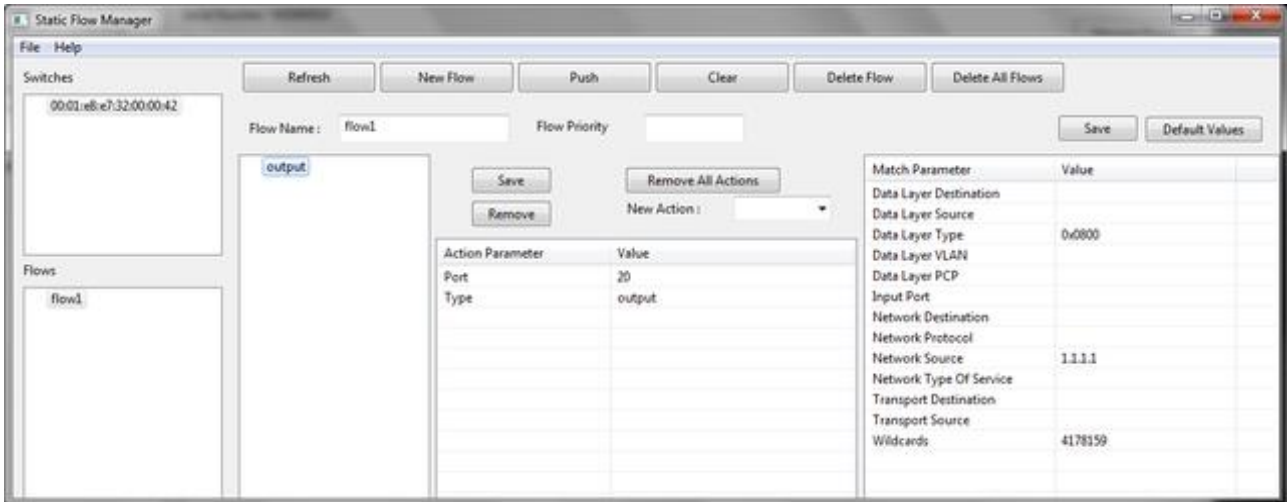
```
OS6900-> show openflow logical-switch controllers
```

Oper State	Logical Switch	Controller	Role	Admin State
vswitch Active		172.26.60.226:6633	Equal	Ena

AVIOR Overview for switch:



AVIOR Static flow manager:



2.9. Minimum working configuration

```
-> show configuration snapshot openflow
! OPENFLOW:
openflow logical-switch vswitch vlan 100
openflow logical-switch vswitch controller 1.2.3.4:6633
openflow logical-switch vswitch interfaces port 1/1
```

2.10. Basic Troubleshooting

```
-> show openflow logical-switch vswitch
Admin
Logical Switch      State  Mode   Versions      VLAN  Ctrlrns  Intf
Flows
-----+-----+-----+-----+-----+-----+-----+-----
-+-----
vswitch
1      0      Ena   Norm   1.0 1.3.1      100    1
```

```
-> show openflow logical-switch interfaces
Logical Switch      Interface      Mode
-----+-----+-----+-----
vswitch
1      1/1          Norm
```

```
-> show openflow logical-switch controllers
Admin
Oper
Logical Switch      Controller      Role  State
State
-----+-----+-----+-----+-----+-----
vswitch
Active              1.2.3.4:6633   Equal  Ena
```

Enabling logging:

```
-> show configuration snapshot system
! System Service:
swlog appid ofcmm subapp all level debug3
swlog appid ofni subapp all level debug3
```

Events logged after "Push" from the controller:

```
-> show log swlog
...
swlogd: ofcmm proto debug1(6) recv 1 5 8 0x2d20
swlogd: ofcmm proto debug1(6) send 1 6 128 0x2d20
swlogd: ofcmm proto debug1(6) recv 1 16 56 0x2d21
swlogd: ofcmm proto debug1(6) send 1 17 108 0x2d21
swlogd: ofcmm proto debug1(6) recv 1 14 80 0
swlogd: ofcmm qos debug1(6) del flow 651 (1 1 65543)
swlogd: ofcmm proto debug1(6) recv 1 14 80 0
swlogd: ofcmm qos debug1(6) add flow 651 (1 1 65544)
swlogd: ofcmm qos debug2(7) qos del reply 278 (1 65543 ack)
swlogd: ofcmm proto debug1(6) recv 1 16 56 0x2d22
swlogd: ofcmm qos debug2(7) qos add reply 256 (1 65544 ack)
swlogd: ofcmm proto debug1(6) send 1 17 108 0x2d22
swlogd: ofcmm proto debug1(6) recv 1 16 56 0x2d23
swlogd: ofcmm proto debug1(6) send 1 17 108 0x2d23
swlogd: ofcmm proto debug1(6) send 1 10 82 0x6b8b49e1
swlogd: ofcmm proto debug1(6) recv 1 16 56 0x2d24
```

2.11. Advanced Troubleshooting

Display rules configured for OpenFlow (only default rule is configured):

```
-> debug qos internal "slice 0/1 list 28 verbose"
          Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]          Red[+]          NotGreen[+]
List 28: 1 entries set up
      Openflow( 511) 0      4      - 512      -      - 767      14[14      ]
0[0      ]          0[0      ]          0[0
      Openflow( 511) 0      5      -      -      -      - 1023      14[0      ]
0[0      ]          0[0      ]          0[0

End...
```

One rule programmed through FloodLight:

```
-> debug qos internal "slice 0/1 list 28 verbose"
          Entry U Slice CIDU CIDL MIDU MIDL TCAM      Count[+]
Green[+]          Red[+]          NotGreen[+]
List 28: 2 entries set up
      Openflow( 0) 0      6      - 1024      -      - 1024      0[0      ]
0[0      ]          0[0      ]          0[0
      Openflow( 0) 0      7      -      -      -      - 1280      0[0      ]
0[0      ]          0[0      ]          0[0
      Openflow( 511) 0      4      - 512      -      - 767      135[25      ]
0[0      ]          0[0      ]          0[0
      Openflow( 511) 0      5      -      -      -      - 1023      135[0      ]
0[0      ]          0[0      ]          0[0
```

2.12. Troubleshooting in Maintenance Shell



Warning:

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

```
TOR #-> ofdbg cmma
ofcmm> help
OpenFlow CMM Help
-----
?          - display this page
age        - flow aging
control    - controllers
dmac       - dmac flows
emerg      - emergency flows
exact      - exact flows
exit       - exit debug
group      - groups
help       - display this page
lswitch    - logical switches
ni         - NI connections
pg         - port groups
port       - ports
quit       - exit debug
rule       - rule flows
```

OFCMM control:

```
ofcmm> control show vswitch
(<vswitch,172.26.60.226:6633,7><rcv messages {total: 10742 unknown: 0}}{hello
1}{echo reply 1}{feats req 2038}
{get config req 1}{set config 1}{pkt out 538}{flow mod 4 [add 2,delete 1,delete
strict 1]}
{stats req 8158 [desc 8,flow 4076,aggregate 2037,port 2037]}><sent messages
{total: 11244 unknown: 0}}
{hello 1}{echo request 1}{feats rep 2038}{get config rep 1}{pkt in 1043}{port
status 2}
{stats rep 8158 [desc 8,flow 4076,aggregate 2037,port 2037]}>)
```

OFCMM lswitch:

```
ofcmm> lswitch show vswitch
[0x10055458:vswitch(1 norm),ports (1=port 0/1/1,20=port 0/1/20),vlans
(1000*),ctrl (<172.26.60.226:6633,7>)]
```

OFCMM rule:

```
ofcmm> rule vswitch
{0x1006b9a0(1): [<eth=800><sip=1.1.1.1/32>] [out=20;]}
```


3. Layer 2 Troubleshooting

3.1. Ethernet Ring Protection

3.1.1. Checklist

- Verify if all nodes are running the same ERP version (0 or 1) and support the same set of versions
- The "ERROR: Invalid MEPID or Service VLAN" is thrown when EthOAM primary-vlan is not matching ERP service VLAN
- Replacing a v1 Ethernet ring node with a v2 Ethernet ring node: "When an Ethernet ring, already deployed using Ethernet ring nodes supporting only the functionalities of ITU-T G.8032 (2008) and ITU-T G.8032 Amd.1 (2009) (Ethernet ring nodes running ITU-T G.8032v1), is upgraded with Ethernet ring nodes supporting the functionalities of this Recommendation (v2 Ethernet ring nodes), an RPL owner node should be upgraded to become an Ethernet ring node running ITU-T G.8032v2 ahead of other Ethernet ring nodes deployed on the same Ethernet ring. Otherwise, differences between ITU-T G.8032v1 and ITU-T G.8032v2 flush behaviour might be exposed in the case of unidirectional failure on the non-RPL ring link attached to the RPL owner node." ITU-T G.8032/Y.1344

3.1.2. Introduction

Ethernet Ring protection switching (ERP) is based on the ITU-T recommendation draft G.8032/Y.1344, which specifies the protection switching mechanisms and protocol for ETH layer Ethernet rings. ERP is used to prevent formation of loops which would fatally effect the network operation and service availability.

By default, Spanning Tree will not operate on the ERP ring ports. When the port is configured as an ERP ring port, Spanning Tree will automatically cease to send/receive BPDUs or use the port vectors in the computation of the algorithm. Additionally, as long as the port remains an ERP port it will not control the blocking/forwarding behavior of the port. But Spanning Tree will be active on all other switch ports and will determine the blocking or forwarding state of VLANs configured on those ports. The protocol will interoperate with other Bridges connected through these ports to form loop free topologies. There are no restrictions in terms of modes or a particular type of STP protocol that can be used. The switch can be configured for per-VLAN (1x1) mode or Flat mode

ERP version 1

Messages

R-APS messages are continuously transmitted wherein the first 3 messages are transmitted simultaneously to ensure fast protection switching (if one or two R-APS messages are corrupted); and after that they are transmitted periodically with an interval of 5 seconds. There are three types of messages.

R-APS (Signal Fail) message - It is continuously transmitted by the node that detects SF (Signal Fail) Condition until the condition persists and informs other nodes about the condition.

R-APS (No-Request, RPL Blocked) message – It is continuously transmitted by the RPL node to indicate the other nodes that there is no failure in the ring and RPL port is blocked.

R-APS (No-Request) message – It is continuously transmitted by the non RPL node that detects the Clearing of SF (Signal Fail) until the reception of R-APS (NR, RB) from RPL node after WTR expiry.

R-APS (Event) message – It is transmitted as a single burst of 3 R-APS messages and is not continuously repeated beyond this burst. The transmission of this R-APS message is done in parallel to other R-APS messages. Flush messages are R-APS “event” messages transmitted using sub-code field.

State machine

Each ERP enabled ring can be in one of the three states – IDLE, PROTECTION and PENDING. At initialization, RPL node blocks its RPL port and unblocks its non RPL port and transmits R-APS (NR, RB). Also, non RPL nodes block one ring port and unblock other ring port. All the ERP nodes then go to the PENDING state. Then on reception of R-APS (NR, RB) from RPL node, all other non-rpl nodes unblock their blocked ring ports and all the ERP nodes then go to the IDLE state. So finally in IDLE state all the non RPL ports are in forwarding state and RPL port is in blocking state.

When the ring port (RPL or non RPL) of any ERP node goes down, then the EVENT "local SF" is detected by the node and R-APS (SF) is transmitted immediately from the other ring port. If the ring port (RPL or non RPL) that goes down is already blocked, then the Event "local SF" is detected by the node and R-APS (SF, DNF) is transmitted immediately from the other ring port. The node then unblocks its blocked port and blocks the down port. It is important to block the port which is going down so that when the port comes up it will be in the blocking state to avoid any loop. The node then flushes the FDB for both the ring ports and goes to the PROTECTION state. All other nodes receiving the R-APS (SF) PDU unblock their blocked port (RPL port at RPL node) and then go to the PROTECTION state and flush the FDB for their ring ports. If the node whose ring port goes down is RPL node then there will be no drop in the traffic and if the node is non ERP node then the traffic is switched through the protection path.

Similarly, if any node goes down then the nodes connected to that node detects the EVENT "local SF". If the node that goes down is non RPL, then the RPL port goes to the forwarding state on reception of R-APS (SF) PDU and then traffic is switched through it. And if RPL node goes down, then all the other nodes go to the PROTECTION state but there will be no drop in the traffic.

If the down ring port comes up for a node in PROTECTION state, then that node detects the EVENT "local Clear SF". The node starts the guard timer and then transmits R-APS (NR) if the node is non RPL or transmits R-APS (NR, RB) if the node is RPL. While the guard timer is running, the node will not process any in-coming R-APS PDU which allows us to ignore the out-dated R-APS PDUs that might be flowing in the network. If the ring node receiving R-APS (NR) message is having its ring ports block, then it compares the remote node ID information with its own node ID. If the remote node ID is higher than its own node ID then unblock its ring ports. On reception of R-APS (NR) by RPL node and if the revertive mode is enabled, it will trigger the WTR timer and all the ERP nodes then go to the PENDING state. After the WTR expiry it blocks its RPL port and unblocks its non RPL port. It then transmits R-APS (NR, RB) PDU and flushes the FDB for ring ports. All the ERP nodes then go to the IDLE state. On reception of R-APS (NR, RB) by non RPL nodes, they unblock their ring ports and stop transmitting the R-APS PDU. They also flush the FDB for their ring ports.



Note: The pending state is not announced in "show erp-ring" output.



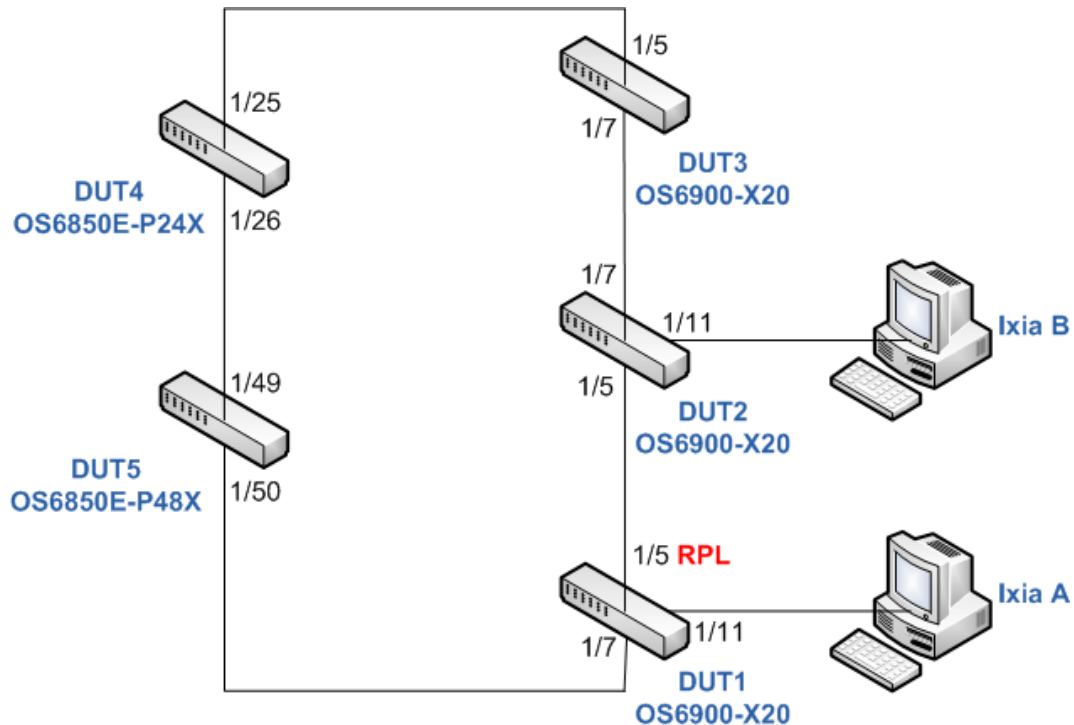
Note: In the pending state the port, which was reactivated, remains in blocking state. Similarly if any node comes up, then initialization occurs as explained above and the nodes connected to it detect the EVENT "local clear SF". So in situation when RPL node goes down and again comes up, it blocks its RPL port and unblocks its non RPL port and then transmits R-APS (NR, RB) PDU. In this case there will not be any drop in the traffic as the path for traffic remains unchanged.[1]

1. ↑ Software Functional Specification, Ethernet Ring Protection (Rev.1.6), Agile P/N : 011464-03

ERP version 2

3.1.3. Minimum working configuration

Network diagram



DUT1 (OS6900 RPL node running AOS 7.3.1.748.R01)

```
! VLAN:
vlan 1 admin-state disable
vlan 10 admin-state enable
vlan 100 admin-state enable
vlan 10 members port 1/5 tagged
vlan 10 members port 1/7 tagged
vlan 100 members port 1/5 tagged
vlan 100 members port 1/7 tagged
vlan 100 members port 1/11 tagged
! Ethernet-OAM:
ethoam domain test format string level 5
ethoam association example format string domain test
ethoam association example domain test primary-vlan 10
ethoam association example domain test ccm-interval interval100ms
ethoam association example domain test endpoint-list 1-2
ethoam endpoint 1 domain test association example direction down port 1/7
```

```

primary-vlan 10
ethoam endpoint 1 domain test association example admin-state enable
ethoam endpoint 1 domain test association example ccm enable
! ERP:
erp-ring 1 port1 1/5 port2 1/7 service-vlan 10 level 5
erp-ring 1 rpl-node port 1/5
erp-ring 1 ethoam-event port 1/7 remote-endpoint 2
erp-ring 1 wait-to-restore-timer 1
erp-ring 1 enable

```

DUT2 (OS6900 non-RPL node running AOS 7.3.1.748.R01)

```

! VLAN:
vlan 1 admin-state disable
vlan 10 admin-state enable
vlan 100 admin-state enable
vlan 10 members port 1/5 tagged
vlan 10 members port 1/7 tagged
vlan 100 members port 1/5 tagged
vlan 100 members port 1/7 tagged
vlan 100 members port 1/11 tagged
! ERP:
erp-ring 1 port1 1/5 port2 1/7 service-vlan 10 level 5
erp-ring 1 enable

```

DUT3 (OS6900 non-RPL node running AOS 7.3.1.748.R01)

```

! VLAN:
vlan 1 admin-state disable
vlan 10 admin-state enable
vlan 100 admin-state enable
vlan 10 members port 1/5 tagged
vlan 10 members port 1/7 tagged
vlan 100 members port 1/5 tagged
vlan 100 members port 1/7 tagged
! Ethernet-OAM:
ethoam domain test format string level 5
ethoam association example format string domain test
ethoam association example domain test primary-vlan 10
ethoam association example domain test ccm-interval interval100ms
ethoam association example domain test endpoint-list 1-2
ethoam endpoint 2 domain test association example direction down port 1/5
primary-vlan 10
ethoam endpoint 2 domain test association example admin-state enable
ethoam endpoint 2 domain test association example ccm enable
! ERP:
erp-ring 1 port1 1/5 port2 1/7 service-vlan 10 level 5
erp-ring 1 ethoam-event port 1/5 remote-endpoint 1
erp-ring 1 enable

```

DUT4 (OS6850E non-ERP node)

```

! VLAN :
vlan 1 disable name "VLAN 1"
vlan 10 enable name "VLAN 10"
vlan 100 enable name "VLAN 100"
! 802.1Q :
vlan 10 802.1q 1/25 "TAG PORT 1/25 VLAN 10"
vlan 100 802.1q 1/25 "TAG PORT 1/25 VLAN 100"
vlan 10 802.1q 1/26 "TAG PORT 1/26 VLAN 10"
vlan 100 802.1q 1/26 "TAG PORT 1/26 VLAN 100"

```

DUT5 (OS6850E non-ERP node)

```
! VLAN :
vlan 1 disable name "VLAN 1"
vlan 10 enable name "VLAN 10"
vlan 100 enable name "VLAN 100"
! 802.1Q :
vlan 10 802.1q 1/49 "TAG PORT 1/49 VLAN 10"
vlan 100 802.1q 1/49 "TAG PORT 1/49 VLAN 100"
vlan 10 802.1q 1/50 "TAG PORT 1/50 VLAN 10"
vlan 100 802.1q 1/50 "TAG PORT 1/50 VLAN 100"
```

3.1.4. Basic troubleshooting

Not applicable

3.1.5. Advanced troubleshooting

Verify number of ERP messages received on software level (an example for slot 1):

```
-> debug qos internal "slot 1 list 1 verbose" | grep ERP
```

3.2. Troubleshooting in Maintenance Shell



Warning:

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

Print the ERP trace:

```
#-> debug $(pidof erpNi) "call erp_traceprint(0)"
[Thread debugging using libthread_db enabled]
0x0fa59db0 in __newselect_nocancel () from /lib/tls/libc.so.6
***** ERP TRACE *****
   1      RNG_CREAT          1          9          10          13900
   2      FSMInit           0          0           0          13900
   3      STG_WR            9          1           0          13900
   4      START_T          1345068617  372295246    0          13900
   5      END_T            1345068617  380486786    0          13900
   6      VM_UPDATE        1010        1           9          13900
   7      STG_WR            10         1           0          13900
   8      START_T          1345068617  381005186    0          13900
   9      END_T            1345068617  389109225    0          13900
  10      VM_UPDATE        1011        1          10          13900
  11      CS_UNBLOC         0           0           0          13900
  12      PM_RESP          1           0           0          13900
...
```

Verify number of ERP messages received on software level:

```
#-> cat /proc/pktdrv | grep ERP
```

3.3. IP Multicast Switching

3.3.1. Checklist

- There should be 1 dedicated querier in the network (in case more than 1 querier is enabled, then there is 1 querier elected and other remain inactive)
- Querier-forwarding should be enabled only on switches located between multicast sources

- and the querier
- Zapping should be enabled only on edge devices
- Proxying may be enabled on all devices
- IGMP messages must be sent with TTL equal 1
- Freezing and pixelation might be caused by congestion or incorrect IGMP Leave handling

3.3.2. Introduction

Flooding the first IP multicast packet

This technical note explains the way first few IP multicast data packets are handled as applicable to different type of platforms and configuration choices available in AOS.

Certain IP multicast applications (e.g.jboss) require that all IP multicast data traffic be sent without any losses. Behavior in AOS with IPMS enabled at switch level is to drop the first few unknown IP multicast data packets until properly learned. On the other hand the first few IP multicast data packets are flooded when IPMS is disabled globally.

When IP multicast switching is disabled at VLAN level, it is expected to flood all of the multicast data traffic. However, in this case it was observed that the first few packets are dropped in BCM based Omniswitches. This works as designed to-date. In Marvell based Omniswitches this behavior is not seen. (See pre-requisites section above for tested platforms and AOS versions)

The behavior observed is as follows.

When IPMS is enabled globally and disabled on a selected VLAN:

```
-> ip multicast status enable  
-> ip multicast vlan 1000 status disable
```

A few packets are lost in the first iteration of injection of data traffic. If the next iteration is performed after a brief amount of time (several seconds) no losses are encountered. If waited for longer and repeated the test, first few packets would be lost again.

When IP multicast switching is enabled globally, the first data packet will always be dropped because the Fast Filter Processor (FFP) in ASIC traps a copy to CPU and drops the packet.

The FFP entry in hardware has the following details:

Destination IP address: 224.0.0.0 mask 240.0.0.0; Action: send packet to CPU

When a traffic flow is first seen on a port there is a brief amount of time between when the flow starts forwarding in hardware and the software programs the actual forwarding vector. By default when IPMS is enabled globally the hardware drops traffic until the forwarding vector is computed. When a new flow arrives on the IPMS disabled VLAN (while IPMS is enabled globally) the first packet is still handled in software and a drop entry shall be written in hardware.

After further processing in software, it finds out that IPMS is disabled on the VLAN in question (e.g. VLAN 1000) thus the created drop entry is removed enabling flooding. This gives rise to resumption of flooding in the IPMS disabled VLAN. Thereafter, forwarding is maintained as long as traffic is injected without any interruption before the IP multicast source timeout (i.e. 30 seconds

by default). If traffic is stopped for more than the source timeout period, the source entry is removed and re-added with re-injection of traffic causing loss of first few IP multicast data packets once again, as described earlier.

Solution

In order to allow first few IP multicast packets also to be flooded without any loss, IPMS flood-unknown feature has to be enabled globally (introduced as part of PR 120874).

```
-> ip multicast status enable
-> ip multicast flood-unknown enable
-> ip multicast vlan 1000 status disable
```

The following command in AOS bShell can be used to verify the behavior:

```
BCM.0> d chg L3_ENTRY_IPV4_MULTICAST
L3_ENTRY_IPV4_MULTICAST.ipipe0[3360]:
<VLAN_ID=0x3e8,VALID=1,SOURCE_IP_ADDR=0x64646464,L3MC_INDEX=1,IPMC=1,GROUP_IP_AD
DR=0xefffffff>
```

When IP multicast source traffic (Data) is received on a port, the corresponding entry is set in hardware as applicable to the IPMS disabled VLAN. L3MC_INDEX value of 1 signifies flooding of IP multicast data traffic on all the ports that match the corresponding VLAN (VLAN_ID (HEX 3e8) = VLAN 1000) Setting this value in hardware takes time and the IP multicast data packets (usually 1 or 2) received until this step are dropped as explained before. L3MC_INDEX allocation is performed by software and there is no provision for maintaining multicast status directly in hardware on a per VLAN basis. Thus the solution to-date as described is to enable IP multicast flood-unknown globally as applicable to all configured VLANs.

IPMC forwarding index range

Restrict the IPMC forwarding index range to just the reserved indexes:

```
capability ipmc-max-entry 3
```

Enabling IPMS

When IPMS is enabled all IP packets matching destination IP address 224.0.0.0/24 are copied to CPU. These packets are rate limited to avoid CPU flooding. An IGMP Membership Query is broadcasted in all VLANs just after IPMS is enabled. Example (generated by a switch without any IP interface configured):

```
MAC: ----- MAC Header -----
MAC:
MAC: Destination Address : 01 00 5E 00 00 01
MAC: Source Address      : 00 E0 B1 A6 C0 9C
MAC: Type                : 0x0800 (Ethernet II)
MAC:
IP: ----- IP Header -----
IP:
IP: Version              = 04 (0x04)
IP: Header Length        = 24 (0x18)
IP: Differentiated Services Field = 192 (0xC0)
```

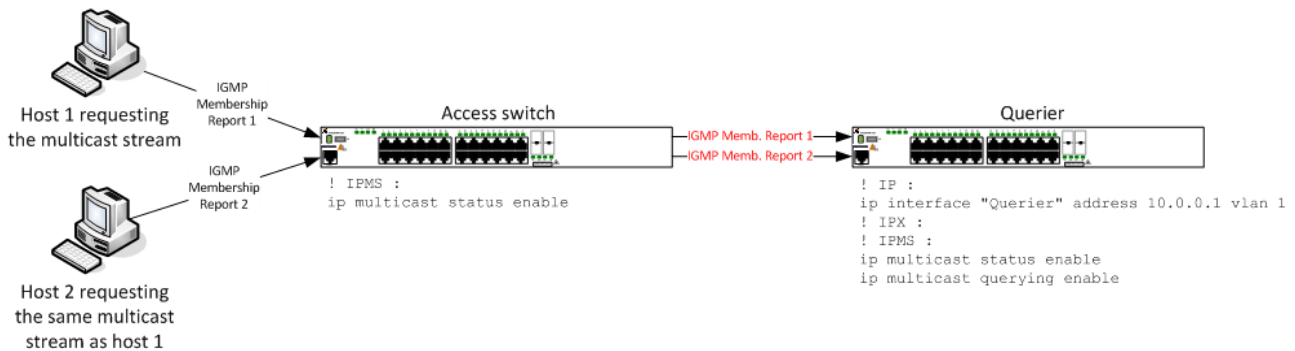
```

IP: Type of Service           = 192 (0xC0)
IP: Total Length              = 32 (0x0020)
IP: Identification           = 4229 (0x1085)
IP: Fragment Offset          = 0
IP: Time to Live              = 1 (0x01)
IP: Protocol                  = IGMP
IP: Checksum                  = 0x...
IP: Source Address            = 0.0.0.0
IP: Destination Address      = 224.0.0.1
IP: Options & Padding         = 0x94040000
IP:
IGMP: ----- IGMP Header -----
IGMP:
IGMP: Version                 = 2 (0x2)
IGMP: Type                    = Membership Query (0x11)
IGMP: Max Response Time      = 25 (0x19)
IGMP: Checksum                = 0xEEE6
IGMP: Group                   = 0.0.0.0
IGMP:

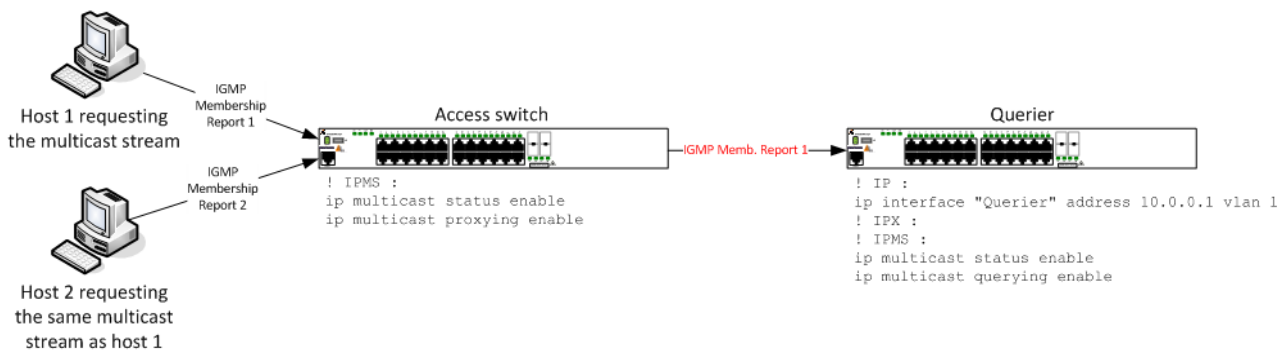
```

Proxying

The proxying feature is used to optimize IGMP Membership Report traffic between an access switch and a querier. The access switch creates a table of registered subscribers containing IP address of multicast group, a port, VLAN id, IGMP version, mode and SSM address. Each time a new IGMP Membership Report is received, the access switch verifies if it is a new registration (new IP multicast group in a VLAN). If necessary (if it is a new registration) IGMP Membership Report is forwarded to the querier. If the group in IGMP Membership Report is already registered there's no need to forward any messages to the querier (there's no need to update querier's registration database), only the table of registered subscribers on the access switch is updated with the port on which the new registration was received. Example (proxying disabled):



Example (proxying enabled):



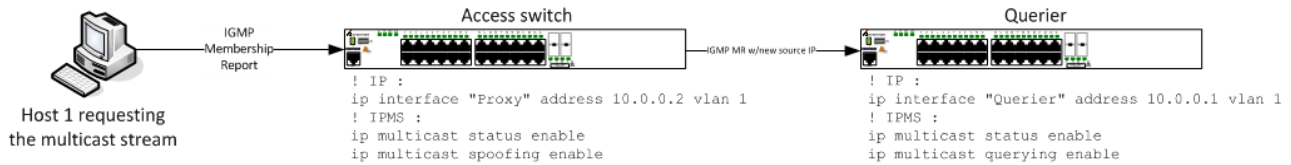


Note:

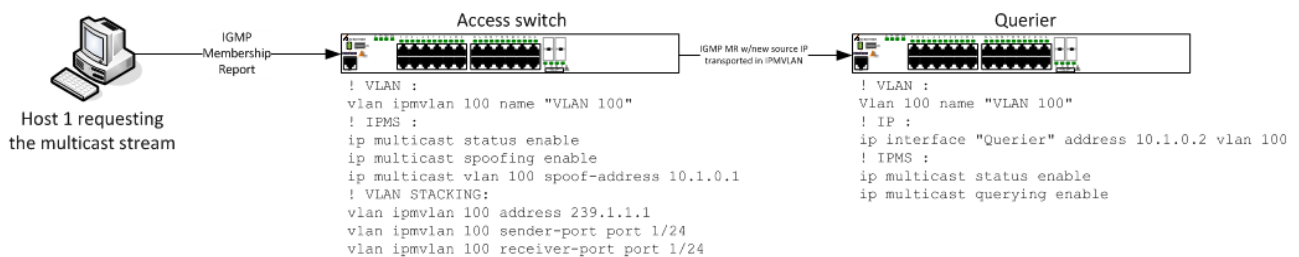
Proxying enabled case. All IGMP Membership Reports generated by the access switch have IP source address inherited from the first received IGMP Membership Reports for a specific IP multicast group.

Spoofing

When the spoofing feature is enabled, source IP address of each IGMP Membership Report is updated according to IP interface address in the VLAN in which the IGMP Membership Report was received. Other IGMP messages are not affected. Example (without IPMVLAN):



Example (with IPMVLAN):

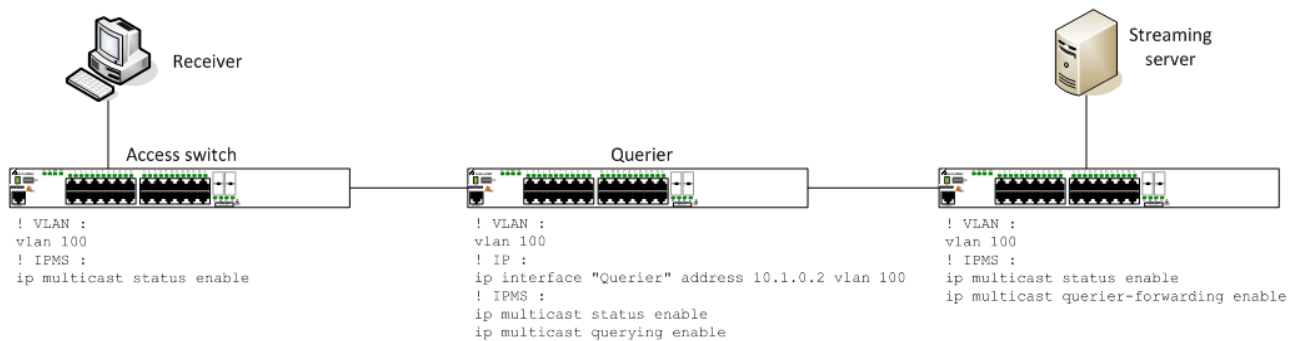


Zapping

If zapping feature is enabled multicast groups are immediately removed from registered subscribers table after receiving IGMP Leave (IGMP version 2) message or IGMP Membership Report type 4 (IGMP version 3).

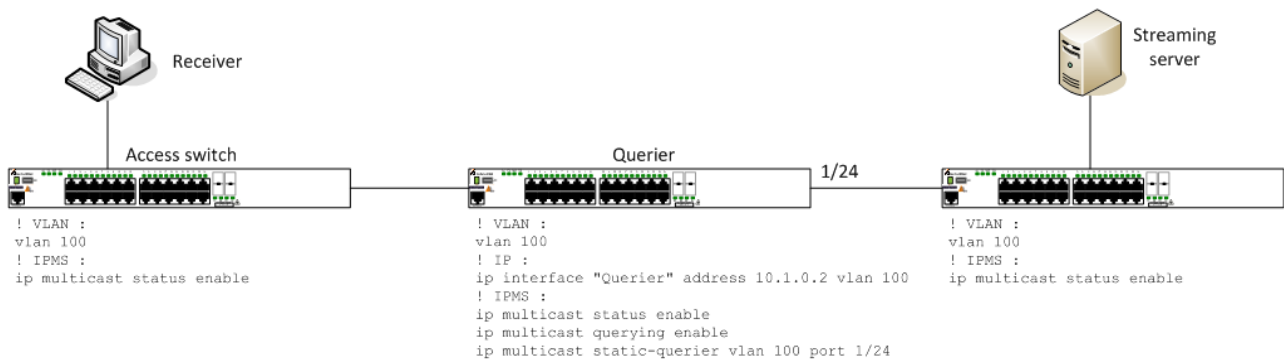
Querier-forwarding

Querier-forwarding feature should be enabled if a streaming device is connected to a switch, which is not a querier. If this feature is enabled on a switch, then all multicast streams are forwarded to the querier.



Static-querier

Static-querier feature should be enabled if a streaming device is connected to a switch, which is not a querier. If this feature is enabled on a querier, then all IGMP Membership queries are forwarded towards the streaming device.



Neighbour

The "neighbour" concept was introduced for IGMPv3 in RFC 3376. This concept is also used for IGMPv2. The extract from the RFC 3376:

6. Description of the Protocol for Multicast Routers The information gathered by IGMP is provided to whichever multicast routing protocol is being used by the router, in order to ensure that multicast packets are delivered to all networks where there are interested receivers.

In AOS each multicast router becomes a "dynamic neighbour" by default. AOS switches replicate all multicast packets and all IGMP messages to all neighbours due to compliance with RFC 3376.

IPMVLAN scenarios in Ethernet-Services mode

Not supported in AOS 7 and AOS 8

Example of IGMP version 2 messages

IGMP Membership Report



Note: All IGMP messages need to have TTL field set to 1. Messages with other values are dropped.



Note: In IGMP version 2 a group address which is registered by IGMP Membership Report corresponds to a destination IP address of a packet.

```

MAC: ----- MAC Header -----
MAC:
MAC: Destination Address : 01 00 5E 01 01 01
MAC: Source Address      : 00 00 00 08 73 F0
MAC: Type                : 0x0800 (Ethernet II)
MAC:
IP: ----- IP Header -----
IP:
IP: Version              = 04 (0x04)
IP: Header Length       = 20 (0x14)
IP: Differentiated Services Field = 0 (0x00)
IP: Type of Service     = 00 (0x00)
IP: Total Length        = 28 (0x001C)
IP: Identification     = 0 (0x0000)
IP: Fragment Offset    = 0
IP: Time to Live       = 1 (0x01)
IP: Protocol           = IGMP
IP: Checksum           = 0x...
IP: Source Address     = 10.0.0.111
IP: Destination Address = 239.1.1.1
IP:
IGMP: ----- IGMP Header -----
IGMP:
IGMP: Version          = 2 (0x2)

```

```

IGMP: Type          = Membership Report v2 (0x16)
IGMP: Max Response Time = 100 (0x64)
IGMP: Checksum      = 0xF796
IGMP: Group         = 239.1.1.1
IGMP:

```

IGMP Leave



Note: In IGMP version 2 a leave message is always sent with 224.0.0.2 IP destination address.

```

MAC: ----- MAC Header -----
MAC:
MAC: Destination Address : 01 00 5E 00 00 02
MAC: Source Address      : 00 00 00 08 73 F0
MAC: Type                : 0x0800 (Ethernet II)
MAC:
IP: ----- IP Header -----
IP:
IP: Version              = 04 (0x04)
IP: Header Length       = 20 (0x14)
IP: Differentiated Services Field = 0 (0x00)
IP: Type of Service     = 00 (0x00)
IP: Total Length        = 28 (0x001C)
IP: Identification     = 0 (0x0000)
IP: Fragment Offset    = 0
IP: Time to Live       = 1 (0x01)
IP: Protocol           = IGMP
IP: Checksum           = 0x...
IP: Source Address     = 10.0.0.111
IP: Destination Address = 224.0.0.2
IP:
IGMP: ----- IGMP Header -----
IGMP:
IGMP: Version          = 2 (0x2)
IGMP: Type             = Leave Group (0x17)
IGMP: Max Response Time = 100 (0x64)
IGMP: Checksum        = 0xF68C
IGMP: Group           = 239.1.1.1

```

IGMP Membership Query



Note: In IGMP version 2 a Membership Query message is always sent with 224.0.0.1 IP destination address.



Note: An IGMP Membership Query message is used to verify if there are still active subscribers registered for a specific group.



Note: IGMP Membership Query with 0.0.0.0 multicast group is called a General Query message. It used for announcing querier's availability.

```

MAC: ----- MAC Header -----
MAC:
MAC: Destination Address : 01 00 5E 00 00 01
MAC: Source Address      : 00 00 00 00 0C 00
MAC:
IP: ----- IP Header -----
IP:
IP: Version              = 04 (0x04)
IP: Header Length       = 20 (0x14)

```

```

IP: Differentiated Services Field = 0 (0x00)
IP: Type of Service              = 00 (0x00)
IP: Total Length                 = 28 (0x001C)
IP: Identification               = 0 (0x0000)
IP: Fragment Offset              = 0
IP: Time to Live                 = 1 (0x01)
IP: Protocol                     = IGMP
IP: Checksum                     = 0x...
IP: Source Address                = 10.0.0.1
IP: Destination Address          = 224.0.0.1
IP:
IGMP: ----- IGMP Header -----
IGMP:
IGMP: Version                    = 2 (0x2)
IGMP: Type                      = Membership Query (0x11)
IGMP: Max Response Time         = 50 (0x32)
IGMP: Checksum                  = 0xEECD
IGMP: Group                     = 0.0.0.0
IGMP:

```

Example of IGMP version 3 messages

IGMP Membership Report type 3



Note: IGMP Membership Report type 3 is used for registration of new subscribers. One message can be used to register multiple multicast groups.



Note: IGMP Membership Report in IGMP version 3 is always sent with 224.0.0.22 destination IP address.

```

MAC: ----- MAC Header -----
MAC:
MAC: Destination Address : 01 00 5E 00 00 16
MAC: Source Address      : 00 00 00 08 73 F1
MAC:
IP: ----- IP Header -----
IP:
IP: Version                = 04 (0x04)
IP: Header Length          = 20 (0x14)
IP: Differentiated Services Field = 0 (0x00)
IP: Type of Service        = 00 (0x00)
IP: Total Length           = 36 (0x0024)
IP: Identification         = 0 (0x0000)
IP: Fragment Offset        = 0
IP: Time to Live           = 1 (0x01)
IP: Protocol               = IGMP
IP: Checksum               = 0x...
IP: Source Address         = 10.0.0.111
IP: Destination Address    = 224.0.0.22
IP:
IGMP: ----- IGMP Header -----
IGMP:
IGMP: Version              = 3 (0x3)
IGMP: Type                 = Membership Report v3 (0x22)
IGMP: Reserved             = 0 (0x0)
IGMP: Checksum             = 0xE8F0
IGMP: Reserved             = 0 (0x0)
IGMP: Number of Group Records = 1 (0x1)
IGMP: ----- IGMP Group Record -----
IGMP:
IGMP: Record Type          = Change To Include Mode (0x3)
IGMP: Aux Data Len         = 0 (0x0)
IGMP: Number of Source(s) = 0
IGMP: Multicast Address    = 239.1.1.1

```

IGMP:

IGMP Membership Report type 4



Note: IGMP Membership Report type 4 is used for unregistering subscribers. One message can be used to unregister multiple multicast groups.



Note: IGMP Membership Report in IGMP version 3 is always sent with 224.0.0.22 destination IP address.

```
MAC: ----- MAC Header -----
MAC:
MAC: Destination Address : 01 00 5E 00 00 16
MAC: Source Address      : 00 00 00 08 73 F1
MAC:
IP: ----- IP Header -----
IP:
IP: Version                = 04 (0x04)
IP: Header Length          = 20 (0x14)
IP: Differentiated Services Field = 0 (0x00)
IP: Type of Service        = 00 (0x00)
IP: Total Length           = 36 (0x0024)
IP: Identification        = 0 (0x0000)
IP: Fragment Offset        = 0
IP: Time to Live           = 1 (0x01)
IP: Protocol                = IGMP
IP: Checksum                = 0x1708
IP: Source Address         = 10.0.0.111
IP: Destination Address    = 224.0.0.22
IP:
IGMP: ----- IGMP Header -----
IGMP:
IGMP: Version              = 3 (0x3)
IGMP: Type                  = Membership Report v3 (0x22)
IGMP: Reserved              = 0 (0x0)
IGMP: Checksum              = 0xE7F0
IGMP: Reserved              = 0 (0x0)
IGMP: Number of Group Records = 1 (0x1)
IGMP: ----- IGMP Group Record -----
IGMP:
IGMP: Record Type          = Change To Exclude Mode (0x4)
IGMP: Aux Data Len         = 0 (0x0)
IGMP: Number of Source(s) = 0
IGMP: Multicast Address    = 239.1.1.1
IGMP:
```

IGMP Membership Query



Note: An IGMP Membership Query message is used to verify if there are still active subscribers registered for a specific group.



Note: IGMP Membership Query with 0.0.0.0 multicast group is called a General Query message. It used for announcing querier's availability.

```
MAC: ----- MAC Header -----
MAC:
MAC: Destination Address : 01 00 5E 00 00 01
MAC: Source Address      : 00 00 00 00 0C 00
MAC:
IP: ----- IP Header -----
IP:
IP: Version                = 04 (0x04)
IP: Header Length          = 20 (0x14)
```

```

IP: Differentiated Services Field = 0 (0x00)
IP: Type of Service               = 00 (0x00)
IP: Total Length                 = 32 (0x0020)
IP: Identification               = 0 (0x0000)
IP: Fragment Offset              = 0
IP: Time to Live                 = 1 (0x01)
IP: Protocol                     = IGMP
IP: Checksum                     = 0x...
IP: Source Address               = 10.0.0.1
IP: Destination Address         = 224.0.0.1
IP:
IGMP: ----- IGMP Header -----
IGMP:
IGMP: Version                   = 3 (0x3)
IGMP: Type                     = Membership Query (0x11)
IGMP: Max Response Time = 0x32 (Raw = 50)
IGMP: Checksum                 = 0xEECD
IGMP: Group                    = 0.0.0.0
IGMP: Reserved (Resv) = 0 (0x0)
IGMP: Suppress Router-Side Processing (S) = 0 (0x0)
IGMP: Querier's Robustness Variable (QRV) = 0 (0x0)
IGMP: Querier's Query Interval Code (QQIC) = 0x0 (Raw = 0)
IGMP: Number of Sources = 0 (0x0)

```

3.3.3. Minimum working configuration

```
ip multicast status enable
```

3.3.4. Basic troubleshooting

Basic outputs

Display IPMS config summary:

```
-> show ip multicast
```

```

Status = enabled,
Querying = disabled,
Proxying = disabled,
Spoofing = disabled,
Zapping = disabled,
Querier Forwarding = disabled,
Flood Unknown = disabled,
Version = 2,
Robustness = 2,
Query Interval (seconds) = 125,
Query Response Interval (tenths of seconds) = 100,
Last Member Query Interval (tenths of seconds) = 10,
Unsolicited Report Interval (seconds) = 1,
Router Timeout (seconds) = 90,
Source Timeout (seconds) = 30,
Max-group = 0,
Max-group action = none,
Helper-address = 0.0.0.0,
Zero-based Query = enabled

```

Display querier information:

```
-> show ip multicast querier
```

```
Total 1 Queriers
```

```

Host Address      VLAN  Port      Static Count Life
-----+-----+-----+-----+-----+-----

```

```
172.13.0.1      2107 1/1/1      no      28520  254
```

Display sources information:

```
-> show ip multicast source
```

```
Total 1 Sources
```

Group Address	Host Address	Tunnel Address	VLAN	Port
239.0.0.1	10.0.0.1	0.0.0.0	1	2/1/1

Display information related to receives:

```
-> show ip multicast group
```

```
Total 1 Groups
```

Group Address	Source Address	VLAN	Port	Mode	Static	Count	Life
239.0.0.1	0.0.0.0	1	1/1/48	exclude	no	5	259

Display active flow information:



Warning:

Flow in this table are populated only in case there is a group match from the the group table and the source table, there must be also an active querier in the network

```
-> show ip multicast forward
```

```
Total 1 Forwards
```

Group Address	Host Address	Tunnel Address	Ingress		Egress	
			VLAN	Port	VLAN	Port
239.0.0.1	10.0.0.1	0.0.0.0	1	2/1/1	1	1/1/48

Logging to SWLOG

Enabling detailed logging:

```
-> show configuration snapshot system
! System Service:
swlog appid ipmsCmm subapp all level debug3
swlog appid ipmsNi subapp all level debug3
```

An example output of a single IGMPv2 Membership Request (IGMP receiver is located is connected to port 1/1/48 and the sender to port 2/1/1):

```
-> show log swlog | grep -E "ipmsCmm|ipmsNi"
<snap> swlogd: ipmsCmm msg debug1(6) ip/6 recv len 164
<snap> swlogd: ipmsNi cap debug1(6) pd_recv/4 src 00-00-00-00-00-01 vlan 1
stack 0 modid 0 port 48 vpn 47 vp 0 cpu 1 flood 1 pd_client 0
<snap> swlogd: ipmsNi cap debug1(6) igmp type x16 vlan 1 stack 0 port 1/1/48
group 239.0.0.1 host 192.168.1.1 sa 00-00-00-00-00-01
<snap> swlogd: ipmsNi msg debug1(6) mcm report vlan 1 stack 0 port 1/1/48 vp 0
host 192.168.1.1 sa 00-00-00-00-00-01 modid 0 devport 48
<snap> swlogd: ipmsCmm msg debug1(6) cni recv len 76
<snap> swlogd: ipmsCmm msg debug1(6) cni report vlan 1 stack 0 port 1/1/48 svp 0
host 192.168.1.1 sa 00-00-00-00-00-01 modid 0 devport 48
<snap> swlogd: ipmsCmm call debug1(6) report vlan 1 stack 0 ifindex 1/1/48 host
```

```

192.168.1.1 sa 00-00-00-00-00-01 modid 0 devport 48
<snap> swlogd: ipmsCmm sub debug1(6) translate vlan 1 stack 0 ifindex
1048(1/1/48) group 239.0.0.1
<snap> swlogd: ipmsCmm sub debug1(6) policy/4 vlan 1 ifindex 1/1/48 group
239.0.0.1 host 192.168.1.1 sa 00-00-00-00-00-01
<snap> swlogd: ipmsCmm rpt debug1(6) igmp/2 join vlan 1 ifindex 1/1/48 host
192.168.1.1 group 239.0.0.1
<snap> swlogd: ipmsCmm obj debug1(6) channel add vlan 1 group 239.0.0.1
<snap> swlogd: ipmsCmm sub debug1(6) is_max_grp/4 vlan 1 ifindex 1/1/48
<snap> swlogd: ipmsCmm obj debug1(6) member add vlan 1 ifindex 1/1/48 group
239.0.0.1
<snap> swlogd: ipmsCmm call debug1(6) havlan check vlan 1 group 239.0.0.1
ifindex 1/1/48 allow 1
<snap> swlogd: ipmsCmm obj debug1(6) thread add vlan 1 group 239.0.0.1 host
10.0.0.1 next 1
<snap> swlogd: ipmsCmm rpt debug1(6) gmi timer vlan 1 ifindex 1/1/48 group
239.0.0.1
<snap> swlogd: ipmsCmm sub debug1(6) link vlan 1 group 239.0.0.1 host 10.0.0.1
next 1 port 1/1/48
<snap> swlogd: ipmsCmm obj debug1(6) fabric add vlan 1 group 239.0.0.1 host
10.0.0.1 next 1 ifindex 1/1/48
<snap> swlogd: ipmsCmm rpt debug1(6) gmi timer vlan 1 group 239.0.0.1
<snap> swlogd: ipmsCmm call debug1(6) relay ia4_t vlan 1 len 28
<snap> swlogd: ipmsCmm msg debug1(6) sec proxy msg_report4 len 68 rem_chas 2
<snap> swlogd: ipmsCmm sub debug1(6) settle fwdvecs0 v4flows 1 v6flows 0
<snap> swlogd: ipmsCmm sub debug1(6) alloc
<snap> swlogd: ipmsCmm obj debug1(6) mcindex add index 3
<snap> swlogd: ipmsCmm obj debug1(6) seq add id 10 cookie 3
<snap> swlogd: ipmsCmm obj debug1(6) fwdvec add mcindex 3 vlan 1 ifindex 2/1/1
fwds 1 trap 0
<snap> swlogd: ipmsCmm msg debug1(6) nic collect chas 1 slot 1 index 3 enable 1
<snap> swlogd: ipmsCmm msg debug1(6) nic collect chas 2 slot 1 index 3 enable 1
<snap> swlogd: ipmsCmm msg debug1(6) nic rep chas 1 slot 1 index 3 type 1
<snap> swlogd: ipmsCmm msg debug1(6) nic rep chas 2 slot 1 index 3 type 1
<snap> swlogd: ipmsCmm msg debug1(6) nic set chas 1 slot 1 index 3 port 2/1/1
<snap> swlogd: ipmsCmm msg debug1(6) nic set chas 2 slot 1 index 3 port 2/1/1
<snap> swlogd: ipmsCmm msg debug1(6) nic mtu chas 1 slot 1 index 3 mtu 1500
<snap> swlogd: ipmsCmm msg debug1(6) nic mtu chas 2 slot 1 index 3 mtu 1500
<snap> swlogd: ipmsCmm msg debug1(6) nic up chas 1 slot 1 index 3 port 1/1/48
vlan 1 nalv 1
<snap> swlogd: ipmsCmm msg debug1(6) nic trap chas 1 slot 1 index 3 enable 0
<snap> swlogd: ipmsCmm msg debug1(6) nic trap chas 2 slot 1 index 3 enable 0
<snap> swlogd: ipmsCmm msg debug1(6) nic valid chas 1 slot 1 index 3
<snap> swlogd: ipmsCmm msg debug1(6) nic valid chas 2 slot 1 index 3
<snap> swlogd: ipmsCmm msg debug1(6) nic collect chas 1 slot 1 index 3 enable 0
<snap> swlogd: ipmsCmm msg debug1(6) nic collect chas 2 slot 1 index 3 enable 0
<snap> swlogd: ipmsCmm msg debug1(6) nic ack chas 1 slot 1 seq 10 cookie 3
<snap> swlogd: ipmsCmm msg debug1(6) nic ack chas 2 slot 1 seq 10 cookie 3
<snap> swlogd: ipmsNi msg debug1(6) cmm rcv len 136
<snap> swlogd: ipmsNi msg debug1(6) cmm collect index 3 enable 1
<snap> swlogd: ipmsNi call debug1(6) collect index 3 enable 1
<snap> swlogd: ipmsNi ipms debug2(7) collect index 3 enable 1
<snap> swlogd: ipmsNi msg debug1(6) cmm rep index 3 type 1
<snap> swlogd: ipmsNi msg debug1(6) cmm set index 3 port 2/1/1
<snap> swlogd: ipmsNi call debug1(6) set index 3 port 2/1/1
<snap> swlogd: ipmsNi msg debug1(6) cmm mtu index 3 mtu 1500
<snap> swlogd: ipmsNi call debug1(6) mtu index 3 mtu 1500
<snap> swlogd: ipmsNi msg debug1(6) cmm up index 3 port 1/1/48 vlan 1 nalv 1
<snap> swlogd: ipmsNi call debug1(6) up index 3 port 1/1/48 vlan 1 nalv 1
<snap> swlogd: ipmsNi ipms debug2(7) L2: unit 0 index 3 port 48
<snap> swlogd: ipmsNi msg debug1(6) cmm trap index 3 enable 0
<snap> swlogd: ipmsNi msg debug1(6) cmm valid index 3
<snap> swlogd: ipmsNi msg debug1(6) cmm collect index 3 enable 0
<snap> swlogd: ipmsNi call debug1(6) collect index 3 enable 0

```



```

<snap> swlogd: ipmsNi ipms debug2(7) collect index 3 enable 0
<snap> swlogd: ipmsNi msg debug1(6) cmm ack seq 10
<snap> swlogd: ipmsCmm msg debug1(6) nic recv len 20
<snap> swlogd: ipmsCmm msg debug1(6) nic ack chas 1 slot 1 seq 10
<snap> swlogd: ipmsCmm msg debug1(6) nic ack chas 2 slot 1 seq 10
<snap> swlogd: ipmsCmm sub debug1(6) index 3 is now ready
<snap> swlogd: ipmsCmm sub debug1(6) flow vlan 1 dest 0.0.0.0 orig 0.0.0.0 group
239.0.0.1 host 10.0.0.1 index 0->3
<snap> swlogd: ipmsCmm msg debug1(6) nic index chas 2 slot 1 vlan 1 group
239.0.0.1 host 10.0.0.1 dest 0.0.0.0 orig 0.0.0.0 index 3
<snap> swlogd: ipmsCmm msg debug1(6) sec chas 0 flow vlan 1 group 239.0.0.1 host
10.0.0.1 dest 0.0.0.0 orig 0.0.0.0 encap 1 index 3
<snap> swlogd: ipmsCmm mip debug1(6) process
<snap> swlogd: ipmsCmm mip debug1(6) view in 0 all 1

```

An example output of a single IGMPv2 Membership Request (IGMP receiver is located is connected to port 1/1/48 and the sender to port 2/1/1, zapping enabled)

```

-> show log swlog | grep -E "ipmsCmm|ipmsNi"
<snap> swlogd: ipmsNi cap debug1(6) pd_recv/4 src 00-00-00-00-00-01 vlan 1
stack 0 modid 0 port 48 vpn 47 vp 0 cpu 1 flood 1 pd_client 0
<snap> swlogd: ipmsNi cap debug1(6) igmp type x17 vlan 1 stack 0 port 1/1/48
group 224.0.0.2 host 192.168.1.1 sa 00-00-00-00-00-01
<snap> swlogd: ipmsNi msg debug1(6) mcm report vlan 1 stack 0 port 1/1/48 vp 0
host 192.168.1.1 sa 00-00-00-00-00-01 modid 0 devport 48
<snap> swlogd: ipmsCmm msg debug1(6) cni recv len 76
<snap> swlogd: ipmsCmm msg debug1(6) cni report vlan 1 stack 0 port 1/1/48 svp 0
host 192.168.1.1 sa 00-00-00-00-00-01 modid 0 devport 48
<snap> swlogd: ipmsCmm call debug1(6) report vlan 1 stack 0 ifindex 1/1/48 host
192.168.1.1 sa 00-00-00-00-00-01 modid 0 devport 48
<snap> swlogd: ipmsCmm sub debug1(6) translate vlan 1 stack 0 ifindex
1048(1/1/48) group 239.0.0.1
<snap> swlogd: ipmsCmm sub debug1(6) policy/4 vlan 1 ifindex 1/1/48 group
239.0.0.1 host 192.168.1.1 sa 00-00-00-00-00-01
<snap> swlogd: ipmsCmm rpt debug1(6) igmp/2 leave vlan 1 ifindex 1/1/48 host
192.168.1.1 group 239.0.0.1
<snap> swlogd: ipmsCmm rpt debug1(6) zap timer vlan 1 ifindex 1/1/48 group
239.0.0.1
<snap> swlogd: ipmsCmm call debug1(6) relay ia4_t vlan 1 len 28
<snap> swlogd: ipmsCmm msg debug1(6) sec proxy_msg_report4 len 68 rem_chas 2
<snap> swlogd: ipmsCmm age debug1(6) member vlan 1 ifindex 1/1/48 group
239.0.0.1
<snap> swlogd: ipmsCmm sub debug1(6) remove vlan 1 group 239.0.0.1 host 10.0.0.1
next 1 ifindex 1/1/48
<snap> swlogd: ipmsCmm obj debug1(6) fabric del vlan 1 group 239.0.0.1 host
10.0.0.1 next 1 ifindex 1/1/48
<snap> swlogd: ipmsCmm obj debug1(6) thread del vlan 1 group 239.0.0.1 host
10.0.0.1 next 1
<snap> swlogd: ipmsCmm obj debug1(6) channel del vlan 1 group 239.0.0.1
<snap> swlogd: ipmsCmm call debug1(6) havlan check vlan 1 group 239.0.0.1
ifindex 1/1/48 allow 0
<snap> swlogd: ipmsCmm obj debug1(6) member del vlan 1 ifindex 1/1/48 group
239.0.0.1
<snap> swlogd: ipmsCmm sub debug1(6) settle fwdvecs 0 v4flows 1 v6flows 0
<snap> swlogd: ipmsCmm sub debug1(6) flow vlan 1 dest 0.0.0.0 orig 0.0.0.0 group
239.0.0.1 host 10.0.0.1 index 3->0
<snap> swlogd: ipmsCmm msg debug1(6) nic undex chas 2 slot 1 vlan 1 group
239.0.0.1 host 10.0.0.1 dest 0.0.0.0 orig 0.0.0.0
<snap> swlogd: ipmsCmm msg debug1(6) sec chas 0 flow vlan 1 group 239.0.0.1 host
10.0.0.1 dest 0.0.0.0 orig 0.0.0.0 encap 1 index 0
<snap> swlogd: ipmsCmm mip debug1(6) process
<snap> swlogd: ipmsCmm mip debug1(6) view in 0 all 1
<snap> swlogd: ipmsCmm age debug1(6) fwdvec mcindex 3 inuse 0

```

```

<snap> swlogd: ipmsCmm obj debug1(6) fwdvec del mcindex 3 vlan 1 ifindex 2/1/1
fws 1 trap 0
<snap> swlogd: ipmsCmm obj debug1(6) mcindex del index 3
<snap> swlogd: ipmsCmm msg debug1(6) nic clear chas 1 slot 1 index 3
<snap> swlogd: ipmsCmm msg debug1(6) nic clear chas 2 slot 1 index 3
<snap> swlogd: ipmsCmm obj debug1(6) seq del id 12 cookie 3 refcnt 0
<snap> swlogd: ipmsCmm sub debug1(6) settle fwdvecs 0 v4flows 0 v6flows 0
<snap> swlogd: ipmsNi msg debug1(6) cmm rcv len 20
<snap> swlogd: ipmsNi msg debug1(6) cmm clear index 3
<snap> swlogd: ipmsNi call debug1(6) clear index 3

```

3.3.5. Advanced troubleshooting

A more detailed version of "show ip multicast group"

```
-> debug ip multicast member
```

Total 1 Members

Group Address/ Source Address	VLAN	Port	Mode	Count	Life	Query	Count	V	V1	V2
239.0.0.1 259	1	1/1/48	exclude	3	259	0	0	2	0	

A more detailed version of "show ip multicast flow"

```
-> debug ip multicast flow
```

Total 1 Flows

indexes inuse 1 max 8189

Group Address/ Dest Address	Host Address/ Orig Address	Next Address	VLAN/ Port	Index	Chas_ID
239.0.0.1 0.0.0.0	10.0.0.1 0.0.0.0	0.0.0.0	1 2/1/1	3	2
			Next 1		

An example output of "debug ip multicat channel":

```
-> debug ip multicast channel
```

Total 1 Channels

Group Address/ Source Address	VLAN	Mode	Count	Life	V	V1	V2
239.0.0.1	1	exclude	6	194	2	0	194

Display IP interface configuration related to multicast:

```
-> debug ip multicast interface
```

Total 3 Interfaces

IfIndex	Host Address	Mac Address	VLAN	VRF	Other	Query	Count
13600001	192.168.10.253	00-00-00-00-00-00	10	0	0	0	0
13600002	192.168.100.254	e8-e7-32-ab-17-bd	100	0	0	0	0
13604125	127.0.0.1	00-00-00-00-00-00	0	0	0	0	0

A more detailed version of "show ip muticast vlan":

```
-> debug ip multicast vlan 1
```

```
Routing = disabled,
Turning = disabled,
Elected = false,
Group Membership Interval (seconds) = 260,
Querier Interval (seconds) = 255,
Startup Query Interval (seconds) = 31,
Last Member Query Time (seconds) = 2,
Unsolicited Report Time (seconds) = 1,
Cvg Query Response Interval (tenths of seconds) = 25,
Cvg Startup Query Interval (seconds) = 7,
IGMPv1 Querier Present (seconds) = 0,
IGMPv2 Querier Present (seconds) = 241
```


Display IP multicast statistics:

```
-> debug ip multicast stats
```

```
Global RX Statistics
V1 Reports = 0 | V1 Queries = 0
V2 Reports = 19 | V2 Queries = 9
V2 Leaves = 3 |
V3 Reports = 2 | V3 Queries = 0
PIM Hellos = 0 | DVMRP Probes = 0

Global TX Statistics
V1 Reports = 0 | V1 Queries = 0
V2 Reports = 6 | V2 Queries = 27
V2 Leaves = 1 |
V3 Reports = 0 | V3 Queries = 0
```

3.3.6. Troubleshooting in Maintenance Shell

 **Warning:** Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

Verify the software forwarding limit:

```
TOR #-> debug $(pidof ipmscmm) 'p ipms::bcm_max'
[Thread debugging using libthread_db enabled]
0x0f990c2c in ___newselect_nocancel () from /lib/tls/libc.so.6
$1 = 2048
```

Change the software forwarding limit:

```
TOR #-> debug $(pidof ipmscmm) 'set ipms::bcm_max=4096'
[Thread debugging using libthread_db enabled]
0x0f990c2c in ___newselect_nocancel () from /lib/tls/libc.so.6
```

3.4. Link Aggregation Control Protocol

3.4.1. Checklist

- In case of linkagg port leave / join events disable Dynamic Automatic Fabric
- In case of linkagg port leave / join events enable long timeout:

**Warning:**

Any changes to timeout cause a single LACP leave / join event.
It is recommended to apply this command during a maintenance window.

```
-> no linkagg lacp port 1/1 actor admin-state timeout
```

- In case of linkagg port leave / join events verify LACP BPDU drop on software and hardware level, finally compare number of LACP BPDUs received in hardware and software
- Unknown Destination/Broadcast/Multicast traffic is load balanced on all the ports of the aggregate. This provides better throughput for broadcast and multicast traffic.



Note: This differs from AOS 6 release where this type of traffic is sent out on one port of the aggregate designated as the primary port

- LACP BPDUs are tunneled by default on UNI ports. Note: This differs from earlier AOS releases where LACP BPDUs were peered.

```
-> show ethernet-service uni-profile default-uni-profile
  Profile Name      Stp      802.1x    802.3ad    802.1ab      MVRP      AMAP
-----+-----+-----+-----+-----+-----+-----
default-uni-profile tunnel  tunnel  tunnel    tunnel    tunnel    tunnel
```

An example configuration with allowed peering:

```
-> show configuration snapshot vlan linkagg
! Link Aggregate:
linkagg lacp agg 1 size 2 admin-state enable
linkagg lacp agg 1 actor admin-key 1
linkagg lacp port 1/1/1 actor admin-key 1
linkagg lacp port 1/1/2 actor admin-key 1
! VLAN:
vlan 1 admin-state enable
ethernet-service svlan 1000 admin-state enable
! VLAN Stacking:
ethernet-service uni-profile "peer-lacp" l2-protocol 802.3ad peer
ethernet-service service-name "svlan1000" svlan 1000
ethernet-service sap 1 service-name "svlan1000"
ethernet-service sap 1 uni linkagg 1
ethernet-service sap 1 cvlan all
ethernet-service uni linkagg 1 uni-profile "peer-lacp"
ethernet-service nni port 1/1/10
ethernet-service svlan 1000 nni port 1/1/10
```

3.4.2. Introduction

Special frames (Link Aggregation Control Protocol Data Unit or LACPDU) are used to interact with a remote system and establish a Link Aggregation Group. However, some preliminary configuration is still needed, whereby the user specifies attributes related to the aggregate group as well as attributes associated to the "possible" participating links. These attributes are normally named "keys". Links will join aggregate groups by the time they come up and exchange LACPDU frames with the peer. Depending on the "keys" associated to the links and to the aggregate, the links may potentially join different groups. Note that a link can only join an aggregate when it is up because LACPDU must be exchanged with a remote system.

- For load balancing purposes, the traffic is distributed across all the ports of an aggregate

group.

- The load balancing is performed at the ingress side
- The speed of the ports is not taken into consideration when traffic is distributed. In other words, the same number of flows is distributed evenly on each port without reference to the line speed.
- The maximum number of aggregates per system is 128 and the maximum number of aggregable ports is 256, including LACP aggregate.
- The maximum number of links per aggregate is 8, but an aggregate may be defined with a maximum of 8, 4 or 2 links). If no aggregate size is provided at configuration time, the default value is 8.
- From the perspective of the LACP state machine, each port can have the following states:
 - Configured: the port has been created and keeps trying to select an aggregate on the NI where the port is located
 - Selected: the port is selected in an aggregate based on the match of actor/partner parameters for the port and the aggregate, each aggregate can have up to 8 ports selected
 - Reserved: the necessary resources to handle the port are available and the port is waiting for LACPDU synchronization exchange to open the traffic
 - Attached: traffic is open

LACP BPDU processing on software level

Once the packets were classified as "trap to CPU" on the ASIC level the packets were placed in CPU port queue. We have 32 internal queues and out of which 27 is used for LACP. The pktDriver receives the packet via DMA Transfer from Broadcom ASIC and classify the packet as LACP and send the packet to LACP module for further software processing.

Hash-control

Hash may be controlled for each linkagg separately using the following command:

```
-> linkagg lacp agg 1 hash ?
      ^
      TUNNEL-PROTOCOL SOURCE-AND-DESTINATION-MAC
      SOURCE-AND-DESTINATION-IP SOURCE-MAC SOURCE-IP
      DESTINATION-MAC DESTINATION-IP
(Link Aggregation Command Set)
```

3.4.3. Support of 256 aggregates and up to 16 ports

A debug CLI command needs to be enabled to support 256 aggregates and up to 16 ports:

```
debug capability linkagg increase-agg-limit
```

Once the command is added to the config file the switch must be reloaded. This applies to both OS10K and OS6900.

To support 256 LAGs in OS10K, the NIs must be XNI-E or HNIs. Any other NI type will force it back to 128 regardless of the presence of the debug instruction.

3.4.4. Minimum working configuration

```
linkagg lacp agg 1 size 2 admin-state enable
linkagg lacp agg 1 actor admin-key 1
```

```
linkagg lacp port 1/1 actor admin-key 1
linkagg lacp port 1/2 actor admin-key 1
```

3.4.5. Basic Troubleshooting

```
show linkagg
show linkagg <id>
show linkagg port
debug show multi-chassis linkagg <id> vlan-list
show multi-chassis consistency linkagg <id>
show multi-chassis consistency linkagg <id> vlan-list
```

3.4.6. Advanced Troubleshooting

In AOS 7.3.4.R02 there was a new command introduced to determine the outgoing port in a linkagg in a standalone unit:

```
debug lag-hash agg <agg-id> mode <brief/extended> rtag <rtag1-7> source-mac
<mac> destination-mac <mac> vlan <vlan> ether-type <etype>
```

Example:

```
-> debug lag-hash agg 1 mode extended rtag rtag7 source-mac e8:e7:32:b3:35:a3
destination-mac e8:e7:32:cd:51:d3 vlan 4000 ether-type 0x0800
```

Aggregate	rtag	outport
1	rtag7	1/1/2

In AOS 7.3.2.414.R01 or newer LACP counters are available from CLI:

```
-> debug show lacp counters port 1/1
Slot/Port  LACP Tx  LACP Rx  LACP Err  Marker RTx  Marker Rx
-----+-----+-----+-----+-----+-----
1/1          126      125       0           0           0
```

Detailed LACP logs:

```
swlog appid linkAggCmm subapp all level debug1
swlog appid linkAggNi subapp all level debug1
```

Even more detailed LACP logs:

```
swlog appid linkAggCmm subapp all level debug3
swlog appid linkAggNi subapp all level debug3
```

Optionally disable duplicates and increase SWLOG size:

```
no swlog duplicate-detect
swlog output flash-file-size 512
```

Gather logs:

```
show configuration snapshot linkagg
show linkagg port
show log swlog | grep LACP
show log swlog | grep linkagg
```

These should be disabled after gathering outputs:

```
swlog appid linkAggCmm subapp all level info
```

```
swlog appid linkAggNi subapp all level info
```

3.4.7. Troubleshooting in Maintenance Shell

Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent.

Warning: Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

Display on screen all data written in the trace buffer from NI module

```
TOR #-> debug $(pidof lacpNi) "call la_ni_trace_prt (-1)"
[Thread debugging using libthread_db enabled]
0x0f8fac2c in ___newselect_nocancel () from /lib/tls/libc.so.6

___LA_NI_SRV_TRACE_DUMP_[NI_LINKAGG_TRAC]__653 cells

___End of the NI trace dump_[NI_LINKAGG_TRAC] : 653 cells 0 free bytes
(2013-07-01 07:45:07.696):: 1:MGT DBG[1:0] -> la_ni_rcv_lacmm_dyn_update_msg_f
(2013-07-01 07:45:07.697):: 2:MGT DBG[1:0] -> la_ni_rcv_lacmm_dyn_update_msg_f:
LA_TOKEN_RESERVATION_REQ
(2013-07-01 07:45:07.697):: 3:MGT DBG[1:0] -> la_ni_lacp_srv_is_port_reserved_f
131073
(2013-07-01 07:45:07.697):: 4:MGT DBG[1:0] ->
la_ni_rcv_lacp_port_reserved_req_f 131073 -1
...
```

Print LACP details per port from NI module

```
TOR #-> debug $(pidof lacpNi) "call la_ni_port_prt (-1)"
[Thread debugging using libthread_db enabled]
0x0f8fac2c in ___newselect_nocancel () from /lib/tls/libc.so.6

1/ 1/ 1 -> 0x100db5f0 status=3 ifdx=1001 id=0 type=1 agg_id=-1 port_index=-1
Bw:LINK DOWN
adminstate=1 operstate=2 link_up_down=0 activation_order=0
agg_ctx_p=0x0
mclag=0 vfl=0 req_token=0

Actor : Sys ID=[e8:e7:32:3b:62:0d] Sys Prio=0 Port=0 Port Prio=0
Admin Key=1 Oper Key=1
Admin State=(act1.tim1.agg1.syn0.col0.dis0.def1.exp0)
Oper State =(act1.tim1.agg1.syn0.col0.dis0.def1.exp0)
Partner : Sys ID=[6c:3b:e5:c1:71:80] Sys Prio=32768 Key=256 Port=0 Port
Prio=0
Admin Key=0 Oper Key=256
Admin Sys ID=[00:00:00:00:00:00] Admin Sys Prio=0 Admin Port=0
Admin Port Prio=0
Admin State=(act0.tim0.agg1.syn0.col1.dis1.def1.exp0)
Oper State =(act0.tim0.agg1.syn0.col1.dis1.def1.exp0)
```

```
Bit order : 1
```

```
...
```

Print LACP statistics from NI module

```
TOR #-> debug $(pidof lacpNi) "call la_ni_lacp_stats_prt (-1)"  
[Thread debugging using libthread_db enabled]  
0x0f8fac2c in ___newselect_nocancel () from /lib/tls/libc.so.6
```

```
1: 1  
lacpdus_rx          = 1  
marker_pdus_rx      = 0  
marker_response_pdus_rx = 0  
unknown_rx          = 0  
illegal_rx          = 0  
lacpdus_tx          = 2675  
marker_pdus_tx      = 0  
marker_response_pdus_tx = 0  
pktdrv_retry        = 0  
pktdrv_drop         = 0
```

```
...
```

Print LACP statistics from NI module

```
TOR #-> debug $(pidof lacpNi) "call la_ni_lacp_port_stats_prt (-1)"  
[Thread debugging using libthread_db enabled]  
0x0f8fac2c in ___newselect_nocancel () from /lib/tls/libc.so.6
```

```
1: 1  
lacpdus_rx          = 1  
marker_pdus_rx      = 0  
marker_response_pdus_rx = 0  
unknown_rx          = 0  
illegal_rx          = 0  
lacpdus_tx          = 2675  
marker_pdus_tx      = 0  
marker_response_pdus_tx = 0  
pktdrv_retry        = 0  
pktdrv_drop         = 0
```

```
...
```

Display on screen all data written in the trace buffer from CMM module

```
#-> debug $(pidof lagCmm) "call la_cmm_trace_unfreeze()"  
[Thread debugging using libthread_db enabled]  
0x0fbccdb0 in ___newselect_nocancel () from /lib/tls/libc.so.6
```

```
#-> debug $(pidof lagCmm) "call la_cmm_trace_freeze()"  
[Thread debugging using libthread_db enabled]  
0x0fbccdb0 in ___newselect_nocancel () from /lib/tls/libc.so.6
```

```
TOR #-> debug $(pidof lagCmm) "call la_cmm_trace_prt(1)"  
[Thread debugging using libthread_db enabled]  
0x0fa9ac2c in ___newselect_nocancel () from /lib/tls/libc.so.6
```

```
____ End of the trace dump [CMM_LINKAGG_TRA] : 264 cells 40640 free bytes  
(2013-06-18 19:00:09.488):: 3:MGT DBG[65] ->  
la_cmm_init_socket_handler_connections_f  
(2013-06-18 19:00:09.495):: 4:MGT DBG[65] -> Sock HDL connect to app : 9  
(2013-06-18 19:00:09.496):: 5:MGT DBG[65] -> Sock HDL connect to app : 8  
(2013-06-18 19:00:09.497):: 6:MGT DBG[65] -> Sock HDL connect to app : 1  
(2013-06-18 19:00:09.497):: 7:MGT DBG[65] -> Sock HDL connect to app : 2  
...
```



```

(2013-06-18 19:01:49.333)::256:MGT LOG[65] -> Join Req agg:1 port:11
(2013-06-18 19:01:49.333)::257:MGT DBG[65] -> la_cmm_send_port_join_f 40000001
1012
(2013-06-18 19:01:49.333)::258:MGT DBG[65] -> la_cmm_update_pmdb_port_join_f
40000001 1012
(2013-06-18 19:01:49.338)::259:MGT DBG[65] -> la_cmm_mip_generate_trap_f 1 1012
(2013-06-18 19:01:49.338)::260:MGT LOG[65] -> Port Join . LACP port: 1012 agg: 1
(2013-06-18 19:01:49.338)::261:MGT DBG[65] ->
la_cmm_sync_multicast_port_join_conf_f 1
(2013-06-18 19:01:49.338)::262:MGT DBG[65] -> la_cmm_to_sync_multicast_msg_f
from job_id=4
(2013-06-18 19:01:49.338)::263:MGT DBG[65] -> la_cmm_to_sync_data_msg_f job_id=4
slot:-1/65535
_____End of the trace dump_[CMM_LINKAGG_TRA] : 264 cells 40640 free bytes

```

`dump_agg()` – displays Linkagg information.

`gport2Agg(int gport)` – convert a Gport Linkagg number representation to a Linkagg number

`agg2Gport(int aggid)` – convert Linkagg number to Gport representation

`ifindex2Gport(int ifindex)` – convert ifindex to gport representation

3.4.8. Verifying LACP BPDU drop on software level

There is a possibility that the packet driver dropping the packet before sending it to the LACP module. We can check the counters in the `pktDrv` side as well as in the LACP module side to identify if there is any difference in TX and RX packet count exist or not. Following command can be useful to identify whether is there any drop at CPU.

```

TOR #-> cat /proc/pktdrv | grep "Classified 27";debug $(pidof lacpNi) "call
la_ni_lacp_port_stats_prt (-1)"
Classified 27          :      16325          127

```

```

warning: Could not load shared library symbols for linux-vdso32.so.1.
Do you need "set solib-search-path" or "set sysroot"?
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/libthread_db.so.1".
0x0f86d188 in __epoll_wait_nocancel () from /lib/tls/libc.so.6

```

```

1: 1
    lacpdus_rx           = 8162
    marker_pdus_rx      = 0
    marker_response_pdus_rx = 0
    unknown_rx          = 0
    illegal_rx          = 0
    lacpdus_tx          = 8162
    marker_pdus_tx      = 0
    marker_response_pdus_tx = 0
    pktdrv_retry        = 0
    pktdrv_drop         = 0

```

```

1: 2
    lacpdus_rx           = 8163
    marker_pdus_rx      = 0
    marker_response_pdus_rx = 0
    unknown_rx          = 0
    illegal_rx          = 0
    lacpdus_tx          = 8165
    marker_pdus_tx      = 0

```

```
marker_response_pdus_tx = 0
pktdrv_retry             = 0
pktdrv_drop              = 0
```

In above lacp NI output there are 2 lacp ports where we are receiving LACP PDUs.

Total packets received in pktDrv should be same as that in lacp NI:

```
pkt_lagg_num == lacp_port1_rx_num + lacp_port2_rx_num
```

- pkt_lagg_num - the number of "Classified 27" packets in the "cat /proc/pktdrv" output
- lacp_port1_rx_num - lacpdus_rx for port 1
- lacp_port2_rx_num - lacpdus_rx for port 2

If the sum is not equal to the pkt_lagg_num, it means that LACP PDU packets are not reaching the CPU.

3.5. Source Learning

3.5.1. Checklist

3.5.2. Introduction

When a packet first arrives on NI source learning examines the packet and tries to classify the packet to join its correct VLAN. If a port is statically defined in a VLAN, the MAC address is classified in the default VLAN. Otherwise, if Group Mobility is being used the MAC address is classified into the correct VLAN based on the rules defined.

As soon as the MAC address is classified in a VLAN, an entry is made in Source Address CAM associating the MAC address with the VLAN ID and the Source Port. This Source Address is then relayed to the CMM for management purposes.

If an entry already exists in MAC address database with the same VLAN ID and the same source port number then no new entry is made. If VLAN ID or the source port is different from the existing entry in MAC address database then the previous entry is aged out and a new entry is made in the MAC address database. This process of adding a MAC address in the MAC address database is known as Source Learning.

A MAC address can be denied to learn on a port based on different policies configured through QOS or Learned Port Security. A MAC address may be learned in a wrong VLAN based on the policies defined for the port.

MAC address table collisions


MAC address collisions may be simulated using Ixia

Results obtained on OS6900:

Software version	VLAN ID	Destination MAC address	The first source MAC address
7.3.2.344.R01 GA	1	00:00:ff:00:00:00	00:00:00:00:00:01

```
7.3.2.344.R01 GA 1          00:00:ff:00:00:00          00:00:00:00:00:01
7.3.2.344.R01 GA 1          00:00:ff:00:00:00          00:00:00:00:00:01
7.3.2.344.R01 GA 1          00:00:ff:00:00:00          00:00:00:00:00:01
7.3.2.344.R01 GA 1          00:00:ff:00:00:00          00:00:00:00:00:01
7.3.2.344.R01 GA 1          00:00:ff:00:00:00          00:00:00:00:00:01
```

Source learning events are logged on debug3 level:

 **Warning:** Collisions are not logged even on debug3 level.

```
-> swlog appid slNi subapp all level debug3
-> show log swlog
/flash/swlog.7 not found!
Displaying file contents for '/flash/swlog.6'
Jul  9 00:21:51 (none) swlogd: slNi FDB debug3(8)
DEBUG3:(250705.649)fdbAddVlanDomain[2049]add d:vlan(1) MAC: 00:00:00:01:f5:9f
vid:1 p:2 b:bridging t:learned e:0 dup:0 L3:0 cp
Jul  9 00:21:51 (none) swlogd: slNi FDB debug3(8)
DEBUG3:(250705.651)fdbAddVlanDomain[2049]add d:vlan(1) MAC: 00:00:00:01:f5:a1
vid:1 p:2 b:bridging t:learned e:0 dup:0 L3:0 cp
Jul  9 00:21:51 (none) swlogd: slNi FDB debug3(8)
DEBUG3:(250705.652)fdbAddVlanDomain[2049]add d:vlan(1) MAC: 00:00:00:01:f5:a3
vid:1 p:2 b:bridging t:learned e:0 dup:0 L3:0 cp ...
```

Starting AOS 7.3.1.TBC.R01 there are also hardware collisions logged on debug2 level. In case of a collision there is a callback message send to software. The MAC address, which failed to be stored in the MAC address table on the hardware level, can be re-ordered on the software level:

```
-> show log swlog | grep cbOverflowProcess | tail -n 3
Jul 15 13:19:59 (none) local0.debug swlogd: slNi CALLBACK debug2(7)
DEBUG2:(340945.897)cbOverflowProcess[1480]MAC: 1c:af:f7:7e:a5:ef Vlan: 1 is
OVERFLOW, flag: 50010440
Jul 15 13:19:59 (none) local0.debug swlogd: slNi CALLBACK debug2(7)
DEBUG2:(340945.897)cbOverflowProcess[1480]MAC: 1c:af:f7:7e:a5:ef Vlan: 1 is
OVERFLOW, flag: 50010440
Jul 15 13:19:59 (none) local0.debug swlogd: slNi CALLBACK debug2(7)
DEBUG2:(340945.898)cbOverflowProcess[1480]MAC: 1c:af:f7:7e:a5:ef Vlan: 1 is
OVERFLOW, flag: 50010440
```

3.5.3. Minimum working configuration

Default configuration


3.5.4. Basic Troubleshooting

Gather corresponding SWLOGs for CMM and NIs:

```
-> show log swlog | grep slCmm
-> show log swlog slot 1 | grep slNi
```

3.5.5. Advanced Troubleshooting

3.6. Troubleshooting in Maintenance Shell

 **Warning:** Maintenance Shell commands should only be used by Alcatel-Lucent personnel or under the direction of Alcatel-Lucent. Misuse or failure to follow procedures that use Maintenance Shell commands in this guide correctly can cause lengthy network down time and/or permanent damage to hardware.

Source Learning CMM Gdb Dump

These commands need to be executed in GDB. First grab the PID of the process, then call the functions using the "debug" command:

dump_dis() – Displays current open sessions.

```
RUSHMORE #-> debug $(pidof slCmm) "call dump_dis()"
[Thread debugging using libthread_db enabled]
0x400007f2 in _dl_sysinfo_int80 () from /lib/ld-linux.so.2
mac_set.size: 0
waiting reply on slots: <none>
entries p/slot:
Current language: auto
The current source language is "auto; currently c".
```

dump_mcast() – Displays multicast MACs learned.

```
RUSHMORE #-> debug $(pidof slCmm) "call dump_mcast()"
[Thread debugging using libthread_db enabled]
0x400007f2 in _dl_sysinfo_int80 () from /lib/ld-linux.so.2
Mcast MACs:
$1 = 0
Current language: auto
The current source language is "auto; currently c".
```

dump_static() – Displays static MACs learned.

```
RUSHMORE #-> debug $(pidof slCmm) "call dump_static()"
[Thread debugging using libthread_db enabled]
0x400007f2 in _dl_sysinfo_int80 () from /lib/ld-linux.so.2
Static MACs:
$1 = 0
Current language: auto
The current source language is "auto; currently c".
```

dump_port() – Displays ports and linkaggs that have learning disabled.

```
RUSHMORE #-> debug $(pidof slCmm) "call dump_port()"
[Thread debugging using libthread_db enabled]
0x400007f2 in _dl_sysinfo_int80 () from /lib/ld-linux.so.2
ports/aggs in system(ports/aggs in [] have learning disabled): 0
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 aggs:
$1 = 4352
Current language: auto
The current source language is "auto; currently c".
```

dump_dis_port() – Displays ports disabled in system.

```
RUSHMORE #-> debug $(pidof slCmm) "call dump_dis_port()"
[Thread debugging using libthread_db enabled]
0x400007f2 in _dl_sysinfo_int80 () from /lib/ld-linux.so.2
ports disabled in system:
Current language: auto
The current source language is "auto; currently c".
```

Source Learning CMM and NI Gdb Dump

These commands need to be executed in GDB. First grab the PID of the process, then call the

functions using the "debug" command:

```
RUSHMORE #-> debug $(pidof slCmm) "call slTraceDbg()"
[Thread debugging using libthread_db enabled]
0x400007f2 in _dl_sysinfo_int80 () from /lib/ld-linux.so.2
-----
----- SL TRACE CICULAR BUFFER -----
-----
-- slTraceClear() - Clear all Trace in buffer
-- slTraceOpAll() - Display available Options
-- slTraceOpShow() - Display set Options
-- slTraceLevelSet() - Set Log Level (0-9)
-- slTraceOpAdd(i) - Add Option
-- slTraceOpRmv(i) - Remote Option
-- slTraceAll() - Display All Trace
-- slTrace(startLine, size) - Disp desired lines
-- slTrace() - Continue display from last leftoff
-----
Current language: auto
The current source language is "auto; currently c".
```

slTraceDbg() – displays the debug capabilities.

dump_lps() – displays LPS information.

3.7. Spanning Tree Protocol

3.7.1. Checklist

- Make sure that all devices are running the same STP mode (1x1, FLAT, MSTP)
- If Auto Fabric is enabled then spantree mode is forced to flat
- In case of MSTP check if all devices are using the same domain name
- In case of MSTP check if all devices have the same digest (digest is calculated based using the region name, MSTIs and associated VLANs)
- In case of MSTP all VLANs must be tagged on all interswitch links otherwise MSTP becomes unpredictable
- Check latency and connectivity loss is from Layer 2 or Layer 3 using ping
- Use show mac-address-table count to verify flushing and re-learning of mac-addresses on switch
- Check whether spanning tree in flat or 1x1 mode using "show spantree"
- Determine root-bridge and make sure that it's on the right bridge
- Use stpni_printStats to verify on which ports TCNs and Flag01 counters are incrementing on each NI (stpni_printStats may be cleared)
- Restrict TCNs on ports where Rx counters for TCN and/or Flag01 are incrementing
- PVST+ BPDUs are affected (dropped in MC-LAG and qos user-port) only in case AOS is explicitly configured in 1x1 PVST+ mode

3.7.2. Introduction

The Alcatel-Lucent Spanning Tree implementation provides support for MSTP, RSTP (802.1w), and STP (802.1D). All three supported protocols ensure that there is always only one data path between any two switches for a given Spanning Tree instance to prevent network loops.

MSTP is only available when the flat mode is active for the switch. The flat mode applies a single

spanning tree instance across all VLAN port connections on a switch. MSTP allows the configuration of Multiple Spanning Tree Instances (MSTIs) in addition to the CST instance. Each MSTI is mapped to a set of VLANs.

As a result, flat mode can support the forwarding of VLAN traffic over separate data paths. STP and RSTP are available in both the flat and 1x1 mode. However, when using either STP or RSTP in the flat mode, the Single Spanning tree instance per switch algorithm applies.

Disputed state

A port in STP will be in "Disputed" state, when a port which is receiving an inferior STP BPDU (low priority) even in learning state. Meaning even after a switch sends a higher priority BPDU if it keeps receiving a lower priority STP BPDU then that port will move to "Disputed" state and will be in "Listening" state. So at this point of time there will no traffic allowed through this port "show vlan port" CLI, which will be in blocking state. But this will not make the port link go down only the VLAN will be in blocking state.

3.7.3. Minimum working configuration

Default configuration

3.7.4. Troubleshooting

-> show spantree ports active

Msti	Port	Oper Status	Path Cost	Role
0	1/1	FORW	2000	DESG
0	1/2	FORW	2000	DESG
0	1/11	FORW	2000	DESG
0	1/12	FORW	2000	DESG
0	1/13	BLK	2000	BACK
0	1/14	BLK	2000	BACK
0	1/19	FORW	20000	DESG
14	1/1	FORW	2000	DESG
14	1/2	FORW	2000	DESG
14	1/11	FORW	2000	DESG
14	1/12	FORW	2000	DESG
14	1/13	BLK	2000	BACK
14	1/14	BLK	2000	BACK
14	1/19	FORW	20000	DESG

-> show spantree cist ports

Per Vlan Spanning Tree is enforced !! (Per VLAN mode)

INACTIVE Spanning Tree Parameters

Port	Oper St	Path Cost	Desig Cost	Role	Prim. Port	Op Cnx	Op Edg	Desig	Bridge ID	Note
1/1/1	BLK		4	0	DIS	1/1/1	PTP	NO	0000-00:00:00:00:00:00	
1/1/2	DIS		0	0	DIS	1/1/2	NS	NO	0000-00:00:00:00:00:00	
1/1/3	DIS		0	0	DIS	1/1/3	NS	NO	0000-00:00:00:00:00:00	
1/1/4	DIS		0	0	DIS	1/1/4	NS	NO	0000-00:00:00:00:00:00	
1/1/5	DIS		0	0	DIS	1/1/5	NS	NO	0000-00:00:00:00:00:00	
1/1/6	DIS		0	0	DIS	1/1/6	NS	NO	0000-00:00:00:00:00:00	
...										

In case of unexpected STP state enable detailed logs in SWLOG:

```

swlog output flash-file-size 12500
swlog appid portMgrCmm subapp all level debug2
swlog appid intfCmm subapp all level debug2
swlog appid VlanMgrCmm subapp all level debug2
swlog appid portMgrNi subapp all level debug2
swlog appid VlanMgrNi subapp all level debug2

```

Toggle the port state to recreate the issue. Reduce logging level to info and gather SWLOG outputs:

```

swlog appid portMgrCmm subapp all level info
swlog appid intfCmm subapp all level info
swlog appid VlanMgrCmm subapp all level info
swlog appid portMgrNi subapp all level info
swlog appid VlanMgrNi subapp all level info
show log swlog

```

3.7.5. Advanced Troubleshooting



Note: Please use AOS 7.3.2.382.R01 or newer. See PR 183459.

STP debug commands (an example for flat mode):

```
-> debug stp bpdu-stats 0 start
```

```
-> debug stp bpdu-stats show 0
```

Port	rxCfg	rxRstp	rxMstp	rxTcn	txCfg	txRstp	txMstp	txTcn
1/1	0	100	0	0	0	100	0	0
1/2	0	0	0	0	0	0	0	0
1/3	0	0	0	0	0	0	0	0
...								

```
-> debug stp bpdu-stats 0 stop
```

```
-> debug stp bpdu-trace start all 1/1
```

3.8. Dynamic Automatic Fabric

3.8.1. Introduction

PR 183158 was opened in July to introduce a new CLI command in order to remove these lines in case they are unwanted. New CLI command was introduced in 7.3.2.375.R01:

```
-> auto-fabric admin-state disable remove-global-config
```

To set the Switch in the same config status as a OS6900 with AOS 7.3.1 you need the following config:

```

auto-fabric admin-state disable
no spb isis interface linkagg 1
spb isis admin-state disable
no spb bvlan 4000-4015

```

```
mvrp disable
mvrp linkagg 1 disable
spanntree mode per-vlan
```

3.9. Further reading

- PR 195324 - Links flaps seen on 10Gig BEB switches in SPB in environment

4. Application Fingerprinting

4.1.1. Checklist

4.1.2. Introduction

The OmniSwitch Application Fingerprinting (AFP) feature attempts to detect and identify applications by inspecting IP packets and comparing packets to pre-defined patterns (application signatures). Once an application is identified, AFP collects and stores information about the application flow in a database on the local switch. Additional configurable options for this feature include the ability to apply QoS policy list rules to the identified flow and generating SNMP traps when a signature match occurs.

Using this implementation of AFP, an administrator can obtain more detailed information about protocols running on a specific device or make sure that certain QoS actions are automatically applied wherever an application might be running.

Application Fingerprinting signature file

Before configuring application fingerprint on the switch, verify if the target application signature already exists in signature file. If it is not there, add it by hand.

The default "app-regex.txt" file provided in the "/flash/app-signature/" directory on the switch contains the following sample REGEX application signatures and groups.

Application REGEX Signatures

App-name: jabber

Description: open instant messenger protocol

```
<stream:stream[\x09-\x0d ][ -~]*[\x09-\x0d ]xmlns=["']jabber
```

App-name: smtp

Description: Simple Mail Transfer Protocol
220[\x09-\x0d ~]* (e?smtp|simple mail)

App-name: bgp

Description: Border Gateway Protocol
\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff..?\x01[\x03\x04]

App-name: dhcp

Description: Dynamic Host Configuration Protocol
[\x01\x02][\x01-]\x06.*c\x82sc

App-name: rtsp

Description: Real Time Streaming Protocol
rtsp/1.0 200 ok

App-name: smb

Description: Samba - Server Message Block
\xffsmb[\x72\x25]

App-name: hotline

Description: Hotline - P2P filesharing protocol
TRTPHOTL\x01\x02

App-name: ciscovpn

Description: VPN client software to a Cisco VPN server
\x01\xf4\x01\xf4

App-name: citrix

Description: Citrix ICA - proprietary remote desktop application
\x32\x26\x85\x92\x58

App-name: rdp

Description: Remote Desktop Protocol
rdpr.*clipdr.*rdpsnd

App-name: ssh

Description: Secure Shell
ssh-[12]\.[0-9]

App-name: vnc

Description: Virtual Network Computing
rfb 00[1-9]\.00[0-9]\x0a

App-name: sip

Description: Session Initiation Protocol
(invite|register|cancel|message|subscribe|notify) sip[\x09-\x0d --~]

Application Groups

App-group: chatting = jabber

App-group: mail = smtp

App-group: network = bgp dhcp rtsp smb

App-group: p2p = hotline

App-group: remote_access = ciscovpn citrix rdp ssh vnc

App-group: voip = sip

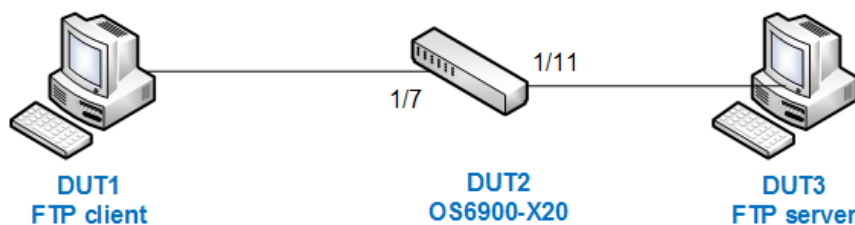
Working Example

This example of application fingerprinting is using AFP UNP mode to monitor FTP packets. The signature of FTP service is not in the default *app-regex.txt*, thus we add the following content. Remember to execute "app-fingerprint reload-signature-file" every time when adding or deleting app-groups.

```
App-name: ftp
Description: FTP - File Transfer Protocol - RFC 959
220[\x09-\x0d --~]*ftp
```

```
App-group: FileService = ftp smb
```

Network diagram:



- DUT1 is a PC act as FTP client with the mac-address 00:1b:21:bd:38:08 and the configured ip address 10.100.100.238.
- DUT2 is OS6900-X20 running AOS7.3.2.375.R01.
- DUT3 is a PC act as FTP server with the mac-address 00:1b:21:bd:3a:1c and the configured ip address 10.100.100.237.

DUT2 configuration

- Nearest-Bridge (01-80-c2-00-00-0e)
- Nearest non-tpmr (01-80-c2-00-00-03)
- Nearest Customer bridge (01-80-c2-00-00-00)

All 3 agents are enabled by default (required by the standard.

Solution:

Please upgrade to AOS 7.3.1.645.R01

Please use the "all" keyword:

```
lldp all port 1/1/1 tlv management system-name enable
```