



December 2020

Use Serviceability Features to Troubleshoot your Cat9K as a Cisco TAC Engineer

IOS-XE based Enterprise Switching

Aaron Cespedes V.
CX Technical Consulting Engineer

News & Upcoming events



Ask Me Anything following the event

Now through Friday December 11, 2020

Use Serviceability Features to Troubleshoot
your Cat9K as a Cisco TAC Engineer

With Aaron Cespedes

Participate: <https://bit.ly/AMA-cat9k>



Aaron Cespedes
Technical Consulting Engineer



Upcoming Support Talks events

Collaboration Solutions Analyzer

December 3rd, 2020

With Kristof Van Coillie

Participate: <https://bit.ly/csa-tool>

Cisco CLI Analyzer

December 17th, 2020

With Magnus Mortensen & Nick Oliver

Participate: <https://bit.ly/CLI-tool>



A
Support Talks
Series

New TAC Tools Explained!

A set of series that will walk you through the different Cisco support tools and their features.

Learn more!

Become an event Top Contributor!

Participate in Live Interactive Technical Events and much more

<http://bit.ly/EventTopContributors>



Cisco Community / Events Top Contributors

Events Top Contributors



This program recognizes Cisco experts in the Cisco Community (CSC) that host technical events (Webcasts, Ask the Experts, Tech Talks, and Facebook Forums.) With this program, Cisco recognizes the positive, valuable influence that our top Cisco experts exert on the communities. To learn more, please visit our [FAQs](#)

2014 2013



Julio Carvajal



Ryota Takao



Cisco Designated VIPs

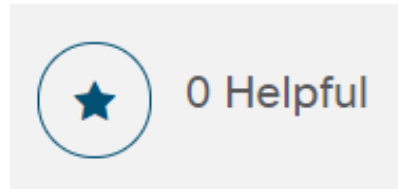


The Cisco Designated VIP program recognizes the top external individual contributors in Cisco's online communities, including the Cisco Support Community (CSC), Cisco Learning Network (CLN) and the Cisco Developers Network (CDN). Cisco Designated VIPs are recognized by their peers for their expertise and tireless contributions, and their abundant participation is vital to community success. With this program, Cisco formally recognizes the positive, valuable influence our top individual members exert on the communities overall. [FAQs](#)

Rate content at the Cisco Community

Help us to recognize the quality content in the community

Rate documents,
Videos & blogs!



Encourage and acknowledge people who
generously share their
time and expertise



Cisco Community Expert



Aaron Cespedes
Technical Consulting Engineer

Question Manager



Laura Arias Camargo
Technical Consulting Engineer

Thank You For
Joining Us Today!



Download Today's Presentation
<https://bit.ly/CL-slidescat9k>

Submit Your Questions Now!

Use the **Q&A** panel to submit your questions and the panel of experts will respond.

They will be answered eventually



Please take a moment to complete the survey at the end of the event



December 2020

Use Serviceability Features to Troubleshoot your Cat9K as a Cisco TAC Engineer

IOS-XE based Enterprise Switching

Aaron Cespedes V.
CX Technical Consulting Engineer

Agenda:

- Serviceability Overview
- Packet Tracer
- Tech-support Packages
- CPU Path

Polling Question 1

Have you used serviceability features such as Packet State Vector or CPU Packet Capture – NETDR Style, etc.?

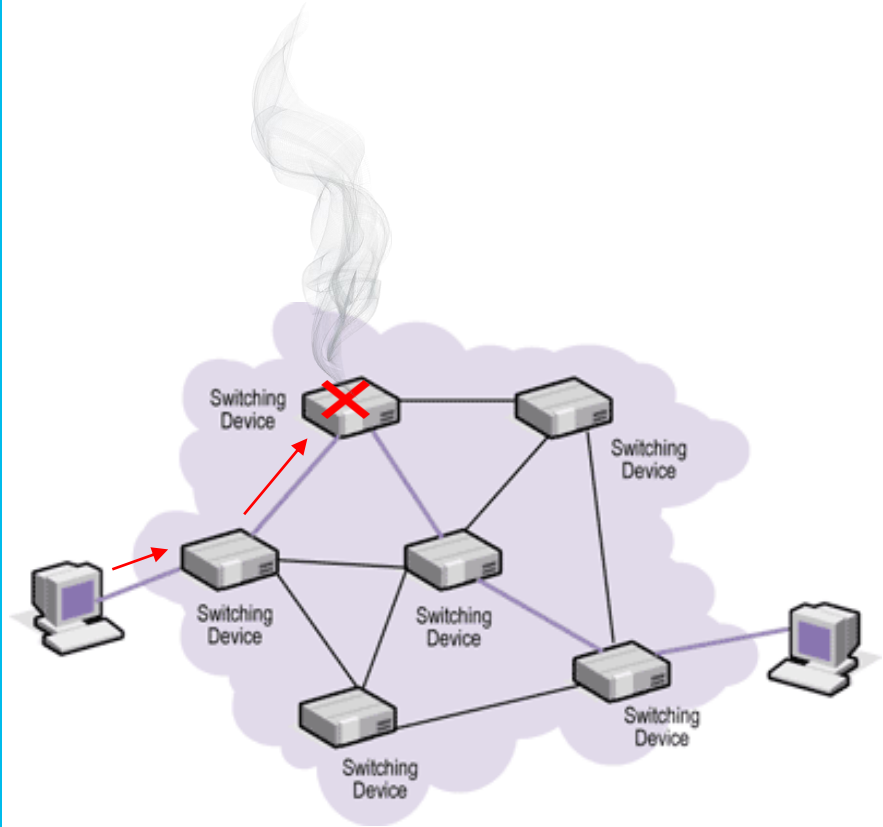
- A. Yes
- B. No

What is Serviceability?

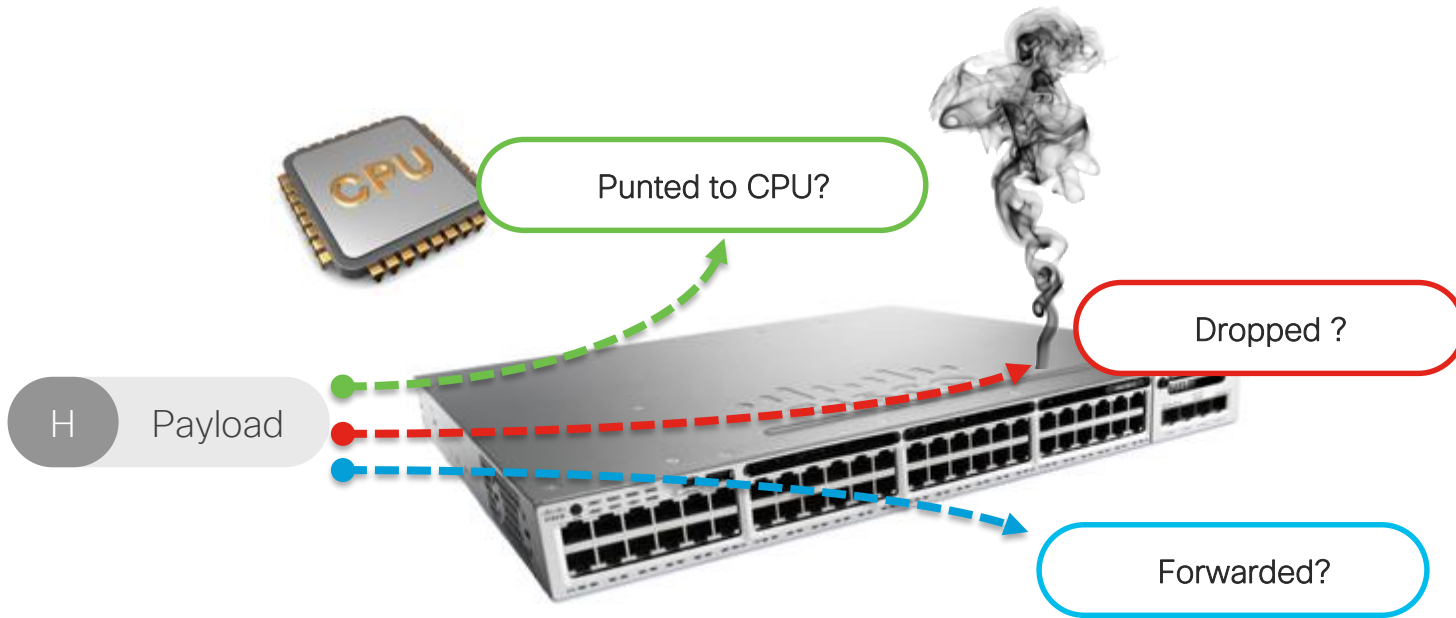
Provide the tools and features necessary to service a Cisco product

- Goal - resolve problems and restore products into service as quickly and efficiently as possible utilizing troubleshooting, maintenance, monitoring, and other fault identification capabilities
- Applies to all support staff - end customer or Cisco support personnel at any knowledge level
- Should be done in a user-friendly manner that is consistent across Cisco products
- Must allow for efficient troubleshooting of a live network, even in degraded conditions, and must go all the way to root cause without the end user having to reproduce the fault or issue
- Result - more efficient product maintenance, reduced operational costs, and business continuity

- Serviceability Overview
- Packet Tracer
- Tech-support Packages
- CPU Path



How to effectively track the forwarding decision made by a switch?

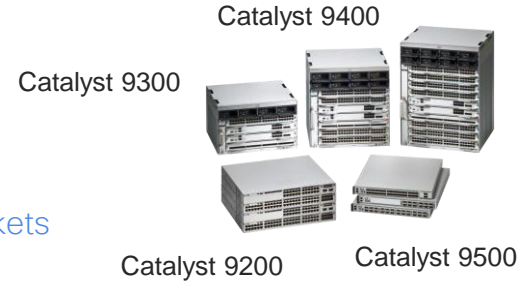


Packet Tracer - SPF vs PSV

UADP 2.0
UADP 2.0 mini

Show Platform Forward

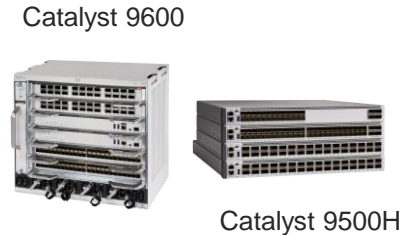
- Simulated ELAM - not a “true” packet tracer
- [Test Doppler forwarding decision using dummy CPU generated packets](#)
- CPU generates 150-200 packets (that do not leave the switch)



UADP 3.0

Packet State Vector

- ELAM - true packet tracer
- [Determine Doppler forwarding decision for a live packet](#)
- Only a single packet can be analyzed at a time



SPF - Show Platform Forward

LET'S RECAP...

- 1 Define packet header

```
show platform hardware fed switch 1 forward interface gigabitEthernet 1/0/22
0011.9267.b370 58bf.eab6.7fe2 ipv4 10.200.1.100 10.201.1.100 tcp 65000 80 0
```

~ 2-5min  CPU generates 150-200 packets (that do not leave the switch)

- 2 Wait for the results

```
*Mar 12 10:40:21.075: %SHFWD-6-PACKET_TRACE_DONE:Switch 1 R0/0: fed:
Show fwd is completed. The capture file can be found at /flash/shfwd+timestamp.log (ie.
shfwdxxxxxx-xxxxxx.log)
```

- 3 Verifying the status

```
dir flash:shfw*
Directory of flash:/shfw*

Directory of flash:/

671770  -rw-          90255  Mar 12 2020 10:40:21 +00:00  shfwd416611827713.log

more flash:shfwd416611827713.log

LEAD_PORT_ALLOW_BROADCAST          1
LEAD_PORT_ALLOW_CAPWAP             0
LEAD_PORT_ALLOW_CTS                0
LEAD_PORT_ALLOW_DOT1Q_TAGGED       0
LEAD_PORT_ALLOW_MULTICAST          4
<snip>
```

Problem #2

SPF require manual specification of L2/L3/L4 packet header details

Problem #1

SPF results provided in form of Doppler registers requiring user to have extensive experience to interpret

Solution to Problem #1 – last summary

```
show platform hardware fed switch 1 forward last summary
```

- “Summary” option available in 16.3 & starting from 16.9
- 16.10 added stack support & support for scenarios requiring recirculation

Input packet details

```
###[ Ethernet ]###
dst      = a0:f8:49:0e:5a:83
src      = 00:17:94:61:7d:50
type     = 0x800
###[ IP ]###
version  = 4L
ihl      = 5L
tos      = 0x0
len      = 28
id       = 1
flags    =
frag     = 0L
ttl      = 64
proto    = icmp
chksum   = 0xa5a9
src      = 10.10.1.100
dst      = 200.200.1.1
options  = ""
###[ ICMP ]###
type     = echo-reply
code     = 0
chksum   = 0xffff
id       = 0x0
seq      = 0x0
```

UADP IFC results

Ingress

```
Port      : GigabitEthernet2/0/1
Global Port Number : 97
Local Port Number  : 1
Asic Port Number   : 0
Asic Instance      : 1
STP Instance       : 2
BlockForward      : 0
BlockLearn         : 0
L3 Interface       : 36
  IPv4 Routing      : enabled
  IPv6 Routing      : enabled
  Vrf Id            : 0
Adjacency:
  Station Index     : 174
  Destination Index : 21083
  Rewrite Index     : 2
  Replication Bit Map : 0x4 ['localData']
Decision:
  Destination Index : 21083
  Rewrite Index     : 2 [RL_L2]
  Dest Mod Index    : 0 [IGR_FIXED_DMI_NULL_VALUE]
  CPU Map Index     : 0 [CMI_NULL]
  Forwarding Mode   : 0 [Bridging]
  Replication Bit Map : ['localData']
  Winner            : L2DESTMACVLAN
```

Egress

```
Egress:
Possible Replication :
  Port : GigabitEthernet2/0/3
Output Port Data :
  Port : GigabitEthernet2/0/3
  Global Port Number : 99
  Local Port Number  : 3
  Asic Port Number   : 2
  Asic Instance      : 1
  Unique RI          : 2
  Rewrite Type       : 1 [L2_BRIDGE]
  Mapped Rewrite Type : 4 [L2_BRIDGE_INNER_IPv4]
  Vlan               : 1
  Mapped Vlan ID     : 4
```

UADP EFC results

Output packet details

```
Port      : GigabitEthernet2/0/3
###[ Ethernet ]###
dst      = a0:f8:49:0e:5a:83
src      = 00:17:94:61:7d:50
type     = 0x800
###[ IP ]###
version  = 4L
ihl      = 5L
tos      = 0x0
len      = 28
id       = 1
flags    =
frag     = 0L
ttl      = 64
proto    = icmp
chksum   = 0xa5a9
src      = 10.10.1.100
dst      = 200.200.1.1
options  = ""
###[ ICMP ]###
type     = echo-reply
code     = 0
chksum   = 0xffff
id       = 0x0
seq      = 0x0
```

Solution to Problem #2 – pcap as SPF input

16.9.1

- 1 Define EPC filters and start capture

```
monitor capture TAC interface gig 1/0/2 in match any
monitor capture TAC start
<...>
monitor capture TAC stop
Capture statistics collected at software:
  Capture duration - 8 seconds
  Packets received - 11
  Packets dropped - 0
  Packets oversized - 0
Capture buffer will exist till exported or cleared
Stopped capture point : TAC
```

- 2 Verify Content of EPC buffer and find the interesting frame

```
show monitor capture TAC buffer
Starting the packet display ..... Press Ctrl + Shift + 6 to exit

1 0.000000 192.168.100.100 -> 192.168.100.1 ICMP 114 Echo (ping) request
2 0.000011 192.168.100.1 -> 192.168.100.100 ICMP 114 Echo (ping) reply
3 0.000015 192.168.100.100 -> 192.168.100.1 ICMP 114 Echo (ping) request
```

input packet

- 3 Export EPC buffer to a file

```
monitor capture TAC export location flash:capture2.pcap
```

- 5 Use the pcap as the input for SPF

```
show platform hardware fed switch 1 forward interface GigabitEthernet 1/0/2 pcap flash:capture2.pcap
number 2 data
```

SPF for CPU Generated packets - L3 Inject

- 1 Define EPC filters and start capture

```
monitor capture TAC control-plane out match any
monitor capture TAC start
<...>
monitor capture TAC stop
Capture statistics collected at software:
  Capture duration - 11 seconds
  Packets received - 28
  Packets dropped - 0
  Packets oversized - 0
Capture buffer will exist till exported or cleared
Stopped capture point : TAC
```

16.12.1

- 2 Verify Content of EPC buffer and find the interesting frame

```
show monitor capture TAC buffer
Starting the packet display ..... Press Ctrl + Shift + 6 to exit

10  3.713598      10.1.1.1 10.1.1.2      ICMP 114 Echo (ping) reply      id=0x0002, seq=0/0, ttl=255
11  3.714673      10.1.1.1 10.1.1.2      ICMP 114 Echo (ping) reply      id=0x0002, seq=1/256, ttl=255
12  3.715488      10.1.1.1 10.1.1.2      ICMP 114 Echo (ping) reply      id=0x0002, seq=2/512, ttl=255
```

- 3 Export EPC buffer to a file

```
monitor capture TAC export location flash:CPU_out.pcap
```

- 5 Use the pcap as the input for SPF

```
show platform hardware fed switch active forward control-plane pcap flash:CPU_out.pcap number 11
```

```
show platform hardware fed switch active forward control-plane l2-inject hex flash:l2_inject.txt
```

PCAP Only for
L3 Injected

L2 Injected
Require
additional info

SPF for CPU Generated packets - L2 Inject

```
show platform hardware fed switch active forward control-plane l2-inject hex flash:l2_inject.txt
```

Legacy method

```
C9300#more flash:l2_inject.txt
```

```
1200400000280052770000000160016FF000035000281E004800001000000408000000000000000000000052000000000000  
00000000000000000000000002CABEB37CD47A0F8490F214786DD6E0000000183AFF000000000000000000000000000000FF02  
00000000000000000001FF6A8F808700759E0000000FE80000000000012B3D5FFFE6A8F80443A5630
```

```
C9300#show platform hardware fed switch active forward last summary
```

```
Input Packet Details:
###[ Ethernet ]###
dst      = 2c:ab:eb:37:cd:47
src      = a0:f8:49:0f:21:47
type     = 0x86dd
###[ IPv6 ]###
version  = 6L
tc       = 224L
fl       = 0L
plen     = 24
nh       = ICMPv6
hlim     = 255
src      = ::
dst      = ff02::1:ff6a:8f80
###[ ICMPv6 Echo Request ]###
type     = Neighbor Solicitation
code     = 0
cksum    = 0x759e
id       = 0x0
seq      = 0xfe80
data     = '\x00\x00\x00\x00\x00\x00\x12\xb3\xd5\xff\xfe\x8f\x80'
###[ Padding ]###
load     = '4a 3a 56 30'
```

```
Egress:
Output Port Data      :
  Port                : GigabitEthernet1/0/46
  Global Port Number  : 46
  Local Port Number   : 46
  Asic Port Number    : 45
  Asic Instance       : 0
  Unique RI           : 2
  Rewrite Type        : 1 [L2_BRIDGE]
  Mapped Rewrite Type : 5 [L2_BRIDGE_INNER_IPv6]
  Vlan                 : 1
```

What is the structure of the file ?
-> next page

L2 Inject file structure

```
C9300#debug platform software fed switch active inject packet-capture start
Inject packet capturing started.

C9300#debug platform software fed switch active inject packet-capture stop
Inject packet capturing stopped. Captured 107 packet(s)

C9300#show platform software fed switch active inject packet-capture detailed
```

```
----- Inject Packet Number: 149, Timestamp: 2020/05/05 07:32:36.784 -----
metadata : cause: 25 [Layer2 frame to BD], sub-cause: 1, q-no: 0, linktype: MCP_LINK_TYPE_IPV6 [4]
ether hdr : dest mac: 2cab.eb37.cd47, src mac: a0f8.490f.2147
ether hdr : ethertype: 0x86DD (IPv6)
ipv6 hdr : dest ip: ff02::1:ff6a:8f80
ipv6 hdr : src ip : ::
ipv6 hdr : payload len: 24, hop count: 255, next hdr: 58 (ICMPv6)
icmp6 hdr : icmp type: 135, code: 0
```

Packet Data Hex-Dump (length: 82 bytes) :

```
2CABEB37CD47A0F8 490F214786DD6E00 000000183AFF0000 0000000000000000
000000000000FF02 0000000000000000 0001FF6A8F808700 759E00000000FE80
00000000000012B3 D5FFFE6A8F80443A 5630
```

<snip>

Doppler Frame Descriptor Hex-Dump :

```
1200400000280052 7700000000160016 FF000035000281E0 0480000100000004
0800000000000000 0000000000520000 0000000000000000 0000000000000000
```

```
show platform software infrastructure inject
```

Statistics for L3 injected packets:

```
3056251 total inject pak, 0 failed
0 sent, 0 prerouted
0 non-CEF capable, 0 non-unicast
<snip>
```

Statistics for L2 injected packets:

```
Statistics for L2 injected packets:
601358 total L2 inject pak, 0 failed
601358 total BD inject pak, 0 failed
0 total EFP inject pak, 0 failed
0 total VLAN inject pak, 0 failed
```

```
C9300#more flash:l2_inject.txt
```

```
12004000002800527700000000160016FF000035000281E00480
00010000000408000000000000000000000000052000000000000
0000000000000000000000000000000002CABEB37CD47A0F8490F214786DD
6E00000000183AFF000000000000000000000000000000000000FF02
0000000000000000000000001FF6A8F808700759E00000000FE800000
0000000012B3D5FFFE6A8F80443A5630
```

Doppler Frame Descriptor + Packet Data

Show Platform Forward - availability

Broken in
17.3.X



Catalyst 9200

Added in 17.2.X

Catalyst 9400



Catalyst 9300



Catalyst 9500

Available since the first
Polaris release 16.1.x

Broken in
17.3.X

Catalyst 9600



Catalyst 9500H

Added in 17.2.X

Catalyst 9600



Catalyst 9500H

ELAM from UADP 3.0
based platforms

PSV - Packet State Vector

LET'S RECAP...

1 Define trigger

```
debug platform hardware fed active capture trigger
[ipv4 | ipv6 <src><dst> [l3 protocol | icmp | igmp | sctp | tcp | tos | udp<src_port><dst_port>]]
[layer2 [ethertype | src_mac | dst_mac]]
[if-id <if_id>ingress | egress ] [interface <ifname>ingress | egress]
[vlan <vlan-id>ingress | egress]
```

2 Start the capture

```
debug platform hardware fed active capture start
```

3 Verify the status

```
show platform hardware fed active capture status
Asic: 0 Status: Running

<..packet arrives..>

show platform hardware fed active capture status
Asic: 0 Status: completed
```

4 Display the results

```
show platform hardware fed active capture
[summary] [packet] |
[psv [ingressFc | egressFc]] |
[detailed [ingressFc | egressFc]]
```

PSV continued

(16.8.1 first release supporting PSV)

```
switch#show platform hardware fed active capture summary
```

```
Trigger: Ether Type:0x0800 Dest IP:20.0.0.2 Src IP:10.0.0.1 Protocol:0x1
Input                               Output                               State
Fo1/0/1                             Fo1/0/9                             FWD
```

Problem #1

“Summary” presents only the general info about ingress/egress interface and forwarding mode

```
switch#show platform hardware fed active capture packet
```

```
Trigger: Ether Type:0x0800 Dest IP:20.0.0.2 Src IP:10.0.0.1 Protocol:0x1
```

```
----- Ingress -----
AsicInst Slice AsicPort ContextId Cmd Error DataValid PakLen Interface
1           1      20         2      0  0      1         118   Fo1/0/1
-----
```

```
----- Egress -----
AsicInst Slice AsicPort ContextId Cmd Error DataValid PakLen Interface
1           0       4         2      0  0      1         118   Fo1/0/9
-----
```

```
----- Packet -----
```

```
Asic: 0 Core: 1
0000 00 a7 42 84 9d 61 00 a7 42 84 9e 01 08 00 45 00
0010 00 64 00 23 00 00 fe 01 9e 73 0a 00 00 01 14 00
0020 00 02 08 00 7d bc 00 07 00 00 00 00 00 00 02 91
0030 fd f5 ab cd ab cd ab cd ab cd ab cd ab cd ab cd
0040 ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd
0050 ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd
0060 ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd
0070 ab cd 81 12 ad a7
```

Problem #2

Packet overview present overview of first 128 bytes of the captured header in hex format

PSV Packet view

17.1.1 enhancement

```
9500-24Y#show platform hardware fed active capture summary
```

```
Trigger: Dest IP:20.1.1.1 Src IP:10.1.1.1 Protocol:0x1
```

Input	Output	State	Reason
Tw1/0/5	Tw1/0/2	FWD	Routed

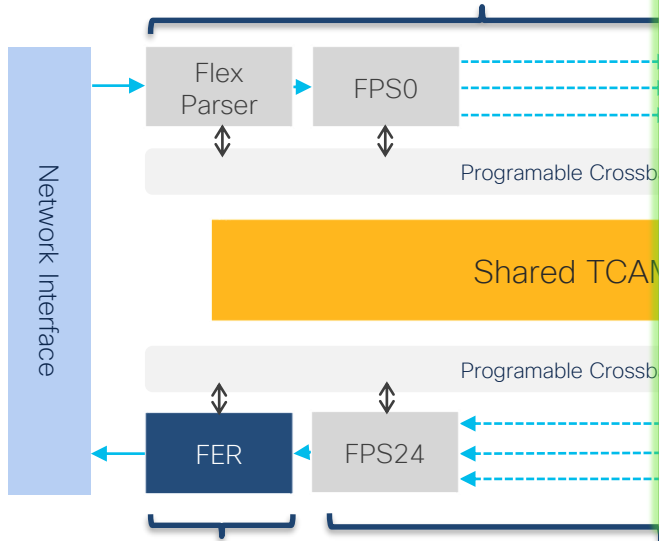
```
9500-24Y#show platform hardware fed active capture packet
```

```
Ingress Packet Data:
Trigger: Dest IP:20.1.1.1 Src IP:10.1.1.1 Protocol:0x1
Ingress Packet Data:
Asic:0 Core:1 Slice:1 AsicPort:24 ContextId:8 Cmd:0 Error:0
DataValid:1 PakLen:118
Interface:Tw1/0/5
Packet:
###[ Ethernet ]###
dst      = 78:72:5d:1b:25:3f
src      = 00:a7:42:d7:ca:9f
type     = 0x800
###[ IP ]###
version  = 4L
ihl      = 5L
tos      = 0x0
len      = 100
id       = 65
flags    =
frag     = 0L
ttl      = 254
proto    = icmp
chksum   = 0x9c54
src      = 10.1.1.1
dst      = 20.1.1.1
options  = ''
###[ ICMP ]###
type     = echo-request
code     = 0
chksum   = 0x8a6b
id       = 0xd
seq      = 0x0
###[ Raw ]###
load     = '00 00 00 00 3E EB B4 E6 AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD'
###[ Padding ]###
load     = '4B 8D 9F 77 FD 07 AB CD AB CD 07 10 C6 C1'
```

```
Egress Packet Data:
Asic:0 Core:1 Slice:1 AsicPort:21 ContextId:5 Cmd:0 Error:0
DataValid:1 PakLen:118
Interface:Tw1/0/2
Packet:
###[ Ethernet ]###
dst      = 00:a7:42:d7:c7:7f
src      = 78:72:5d:1b:25:3f
type     = 0x800
###[ IP ]###
version  = 4L
ihl      = 5L
tos      = 0x0
len      = 100
id       = 65
flags    =
frag     = 0L
ttl      = 253
proto    = icmp
chksum   = 0x9d54
src      = 10.1.1.1
dst      = 20.1.1.1
options  = ''
###[ ICMP ]###
type     = echo-request
code     = 0
chksum   = 0x8a6b
id       = 0xd
seq      = 0x0
###[ Raw ]###
load     = '00 00 00 00 3E EB B4 E6 AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD'
###[ Padding ]###
load     = '4B 8D 9F 77 FD 07 AB CD AB CD 07 10 C6 C1'
```

PSV Example

```
show platform hardware fed active capture psv ingressFc
```



```
950024Y#show platform hardware fed active capture psv egressFc
```

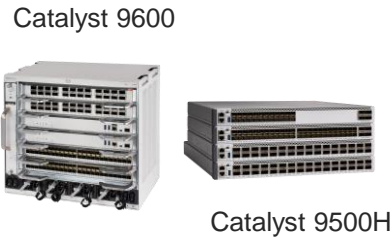
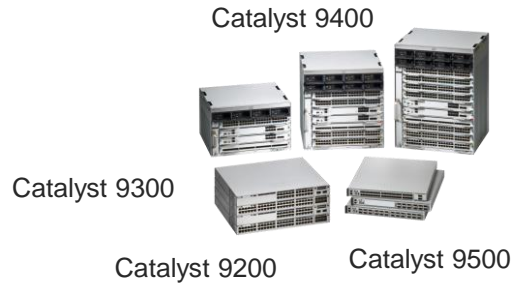
```
Trigger: Dest IP:20.1.1.1 Src IP:10.1.1.1 Protocol:0x1
ASIC: 0 CORE: 1
PHF_EGRESS_OUTER_L2_ADDR_PRESENT           :0x1
PHF_EGRESS_L2_PROTOCOL                     :0x800
PHF_EGRESS_DEST_MAC_ADDRESS               :0xa742d7c77f
PHF_EGRESS_SRC_MAC_ADDRESS                 :0xa742d7ca9f
PHF_EGRESS_ARP_OPERATION                   :0x64
PHF_EGRESS_ARP_HARDWARE_TYPE               :0x800
PHF_EGRESS_ARP_PROTOCOL_LENGTH            :0xfd
PHF_EGRESS_ARP_HARDWARE_LENGTH            :0x1
PHF_EGRESS_WAN_MACSEC_CONST_START          :0xfd
PHF_EGRESS_IPV4_PRESENT                   :0x1
PHF_EGRESS_L3_PROTOCOL                     :0x1
PHF_EGRESS_L3_LENGTH                       :0x64
PHF_EGRESS_IP_TTL                          :0xfd
PHF_EGRESS_IPV6_SRC_ADDRESS                :0xa010101
PHF_EGRESS_IPV6_DEST_ADDRESS              :0x14010101
PHF_EGRESS_INNER_IPV6_FLOW_LABEL           :0x800
PHF_EGRESS_IPV4_DEST_ADDRESS               :0x14010101
PHF_EGRESS_IPV4_SRC_ADDRESS                :0xa010101
PHF_EGRESS_ICMP_OR_IGMP_PRESENT            :0x1
<snip>
```

```
show platform hardware fed active capture detailed egressFc
```

```
show platform hardware fed active capture psv egressFc
```

SPF and PSV CLI unification following Packet Tracer

- Currently having 3 different tools to track hw forwarding decision on IOS-XE enterprise products
- Packet tracer is the most commonly used by the end CU and also included in the latest CCIE R&S blueprint



UADP 2.0

Show Platform Forward

UADP 3.0

Packet State Vector

CPP

Packet Tracer

Attempt to provide SPF & PSV functionality using packet tracer CLI style

PSV Packet Tracer CLI unification

1 Define trigger

```
debug platform condition match
[ipv4 A.B.C.D/nn|host <host>| A.B.C.D/nn|host <host>|[ protocol <0-255>|udp|tcp eq|gt|lt|range|neq
<port>] ingress|egress ] |
[ipv6 A.B.C.D/nn|host <host>| A.B.C.D/nn|host <host>|[ protocol <0-255>|udp|tcp eq|gt|lt|range|neq
<port>] ingress ] |
[interface <interface name> match [ipv4 ..| ipv6] |
[mac host <H.H.H>|<H.H.H> <M.M.M>|any host <H.H.H>|<H.H.H> <M.M.M>|any ingress|egress]
debug platform condition feature
[vlan <vlan>] |
[cos <cos value>]
```

17.3.1

2 Start the capture

```
debug platform condition start
debug platform packet-trace start
```

3 Verify the status

```
show platform conditions
show platform packet-trace status
```

4 Display the results

```
show platform packet-trace summary <<<--- PSV Summary results
show platform packet-trace packet all <<<--- Packet details
show platform packet-trace detailed ingress <<<--- PSV FIR (Forwarding Ingress Resolution)
show platform packet-trace detailed egress <<<--- PSV FER (Forwarding Egress Resolution)
```

SPF with Packet Tracer CLI unification

1 Define trigger

```
debug platform condition feature simulate
[ interface <interface name> | control-plane ]
[ ipv4 <access-list name> ]
[ ipv6 <access-list name> ]
[ l2-inject hex <flash:filename> ]
[ mac <access-list name> ]
[ pcap <flash:file.pcap> number <number> ]

debug platform condition feature
[ arp ...]
[ cos ...]
[ mpls ...]
[ vlan ...]
[ vxlan ...]
```

17.3.1

2 Start the capture

```
debug platform condition start
debug platform packet-trace simulation start
```

3 Verify the status

```
show platform conditions
show platform packet-trace simulation status
```

4 Display the results

```
show platform packet-trace simulation summary
show platform packet-trace simulation details
```

Polling Question 2

Let's imagine we want to trace an ICMP request packet from IP 1.1.1.1 to IP 2.2.2.2, this traffic pass through traffic for the switch in question and we would like to know the forwarding decision of the switch for this specific flow. The switch is a C9300 switch running IOS XE version 16.12.4. Which serviceability tool can we use for this task:

- A. CPU Packet Capture – NETDR Style
- B. SPF (show platform forward)
- C. PSV (packet state vector)

- Serviceability Overview
- Packet Tracer
- **Tech-support Packages**
- CPU Path



Tech-support Packages

Also back ported to 16.9.3+



16.10.1

Show tech-support package for REP
Show tech-support package for QoS
Show tech-support package for CPU
Show tech-support package for PoE
Show tech-support package for Platform
Show tech-support package for Diagnostics
Show tech-support package for Stack
Show tech-support package for ACL
Show tech-support package for SDA (fabric)
Show tech-support package for Dot1x (Identity)
Show tech-support package for Port (Interfaces)
Show tech-support package for Layer3 unicast stream
Show tech-support package for Layer3 multicast stream
Show tech-support package for IGMP snooping group
Show tech-support package for MLD snooping group

16.9.3+

Show tech-support package for CTS
Show tech-support Stackwise-virtual

16.12.1

Show Tech-support Package for Port-Channel
Show Tech-support Package for VLAN and Spanning-tree
Show Tech-support Package for StackWise-Virtual
Show Tech-support Package for MPLS
Show Tech-support Package for REP - Platform Commands Added
Show Tech-support Package for SDA (Fabric) - IPv4 Output Enhancements

17.1.1

Show Tech-support Package for AVC (NBAR)
Show Tech-support Package for Drops

17.2.1

Show Tech-support Package for Resource
Show Tech-support Package for Port (Interfaces) - Improved command list and new per interface option

17.3.1

Show Tech-support Package for Private VLAN (PVLAN)
Show Tech-support Package for Monitor (SPAN)
Show Tech-support Package Confidential

Tech-support Packages

- **Outputs can be large and may lock up the console for several minutes.** To avoid this:
- Redirect the output to a text file on flash & use “more” to view it (or view the file offline):

```
Switch# show tech-support <feature> | redirect flash:filename.txt  
Switch# more flash:filename.txt
```

```
C9300#show tech-support resource | redirect flash:tech_resource_Apr2_2020.txt
```

```
C9300#more flash:tech_resource_Apr2_2020.txt
```

Pressing the space bar is required to advance between screens which allows exiting at any point

```
----- show clock -----
```

```
*18:48:42.776 UTC Thu Apr 2 2020
```

```
----- show version -----
```

```
Cisco IOS XE Software, Version 2020-03-31_20.05
```

```
Cisco IOS Software [Amsterdam], Catalyst L3 Switch Software (CAT9K_IOSXE), Experimental Version 17.3.20200324:113201
```

```
[polaris_dev-/nobackup/13-srvc 106]
```

```
--snip--
```

- To view the command list contained within a show tech-support <feature> command use:

```
Switch# show tech-support <feature> | include -- show
```

Tech-support Packages

```
C9300#show tech-support resource | include -- show
----- show clock -----
----- show version -----
----- show running-config -----
----- show module -----
----- show switch -----
----- show environment power all -----
----- show power inline -----
----- show sdm prefer -----
----- show vlan summary -----
----- show spanning-tree summary -----
----- show monitor session all -----
----- show interfaces counters -----
----- show platform software status control-processor brief -----
----- show platform hardware fed switch 1 fwd-asic resource rewrite utilization -----
----- show platform hardware fed switch 2 fwd-asic resource rewrite utilization -----
----- show platform hardware fed switch 1 fwd-asic resource tcam utilization -----
----- show platform hardware fed switch 2 fwd-asic resource tcam utilization -----
----- show platform hardware fed switch 1 fwd-asic resource utilization -----
----- show platform hardware fed switch 2 fwd-asic resource utilization -----
----- show platform hardware fed switch 1 fwd-asic resource features ip-adjacency utilization -----
----- show platform hardware fed switch 2 fwd-asic resource features ip-adjacency utilization -----
----- show platform hardware fed switch 1 qos resource usage -----
----- show platform hardware fed switch 2 qos resource usage -----
----- show platform software fed switch 1 acl usage -----
----- show platform software fed switch 2 acl usage -----
----- show platform software fed switch 1 fnf sw-stats-show -----
----- show platform software fed switch 2 fnf sw-stats-show -----
----- show flash-1: all -----
----- show flash-2: all -----
C9300#
```

Data is collected from each switch in the stack

How can I look at a specific command output quickly?

```
C9300#show tech-support resource | include -- show
----- show clock -----
----- show version -----
----- show running-config -----
--snip--
----- show spanning-tree summary -----
----- show monitor session all -----
----- show interfaces counters -----
----- show platform software status control-processor brief -----
----- show platform hardware fed switch 1 fwd-asic resource rewrite utilization -----
----- show platform hardware fed switch 2 fwd-asic resource rewrite utilization -----
----- show platform hardware fed switch 1 fwd-asic resource tcam utilization -----
--snip--
C9300#show tech-support resource | redirect flash:tech_resource
C9300#more flash:tech_resource | begin switch 2 fwd-asic resource rewrite
----- show platform hardware fed switch 2 fwd-asic resource rewrite utilization -----

Resource Info for ASIC Instance: 0
Rewrite Data
-----
PHF_EGRESS_destMacAddress      0      32000
ipTtlTable                    0        16
IPV4_TUNNEL_SRC_IP_ADDR      0        16
IPV4_TUNNEL_DEST_IP_ADDR    0        256
--snip--
```

Display all show commands associated with a show tech feature

Determine the specific command output that you want to look at

Redirect the complete show tech output to a file in flash

"Begin" displaying the output by matching the CLI pattern

Tech-support Packages

16.10.1 - Command was 1st introduced

17.2.1 - interface option added, command list improved

```
C9400#show tech-support port interface gig 1/0/1 | include -- show
----- show clock -----
----- show version -----
----- show running-config -----
----- show module -----
----- show inventory -----
----- show etherchannel summary -----
----- show idprom interface GigabitEthernet1/0/1 -----
----- show ip interface brief GigabitEthernet1/0/1 -----
----- show interfaces GigabitEthernet1/0/1 trunk -----
----- show interfaces GigabitEthernet1/0/1 switchport -----
----- show interfaces GigabitEthernet1/0/1 status -----
----- show interfaces GigabitEthernet1/0/1 status err-disabled -----
----- show interfaces GigabitEthernet1/0/1 summary -----
----- show interfaces GigabitEthernet1/0/1 counters -----
----- show interfaces GigabitEthernet1/0/1 counters errors -----
----- show interfaces GigabitEthernet1/0/1 counters protocol status -----
----- show interfaces GigabitEthernet1/0/1 -----
----- show interfaces GigabitEthernet1/0/1 capabilities -----
----- show interfaces GigabitEthernet1/0/1 transceiver detail -----
----- show controllers ethernet-controller GigabitEthernet1/0/1 -----
----- show controllers utilization -----
----- show controllers ethernet-controller GigabitEthernet1/0/1 phy detail -----
----- show platform pm port-data GigabitEthernet1/0/1 -----
----- show platform pm port-state -----
----- show platform pm interface-numbers -----
```

- Often there are continued improvements after initial release
- Tracked via Tech Zone

Continued...

```
----- show platform hardware fed active qos queue stats interface
GigabitEthernet1/0/1 -----
----- show platform hardware cman fp active data-path 1 1 detail -----
----- show platform hardware cman fp active flowcontrol status -----
----- show platform hardware iomd 1/0 data-path 1 detail -----
----- show platform hardware iomd 1/0 flowcontrol status -----
----- show platform hardware iomd 1/0 oci status -----
----- show platform hardware iomd 1/0 portgroups -----
----- show platform software fed active port summary -----
----- show platform software fed active ifm interfaces ethernet -----
----- show platform software fed active ifm mappings -----
----- show platform software fed active ifm mappings lpn -----
----- show platform software fed active ifm mappings gpn -----
----- show platform software fed active ifm mappings port-le -----
----- show platform software fed active port if_id 8 -----
----- show platform software fed active ifm 8 -----
C9400#
```

- Serviceability Overview
- Packet Tracer
- Tech-support Packages
- CPU Path



CPU Path – Tech-support

```

C9400#show tech-support qos active control-plane | include -- show
----- show clock -----
----- show version -----
----- show running-config -----
----- show module -----
----- show interfaces -----
----- show controllers ethernet-controller -----
----- show policy-map interface -----
----- show module -----
----- show process cpu -----
----- show process cpu platform ----- CPU utilization
----- show process cpu history -----
----- show policy-map control-plane -----
----- show policy-map control-plane all -----
----- show platform software fed active punt cpuq brief -----
----- show platform hardware fed active qos queue stats internal cpu policer ----- CoPP drops
----- show platform hardware fed active fwd-asic drops exceptions -----
----- show platform hardware fed active fwd-asic resource tcam table qos format 0 -----
----- show platform software fed active qos policy target iif_id 16777217 -----
----- show platform software fed active qos policy name system-cpp-policy config -----
----- show platform software fed active qos policy name system-cpp-policy target detail -----
----- show platform software process slot switch active r0 monitor ----- Linux "TOP" output
----- show platform software fed active qos internal memory -----
----- show platform software fed active punt cause summary ----- Doppler (FED) punt & inject path queue health
----- show platform software fed active inject cause summary -----
----- show platform hardware fed active fwd-asic drops exceptions -----
----- show platform software fed active punt cpuq brief -----
----- show platform software infrastructure lsmpi -----
----- show platform software infrastructure lsmpi punt -----
----- show platform software infrastructure inject -----
--snip--

```


CPU Path Packet capture – NetDr Style

16.9.3
Initial release

```
C9300#debug platform software fed switch active punt packet-capture start  
Punt packet capturing started.
```

“inject” keyword is supported as well

```
C9300#debug platform software fed switch active punt packet-capture stop  
Punt packet capturing stopped. Captured 263 packet(s)
```

```
C9300#show platform software fed switch active punt packet-capture brief  
Punt packet capturing: disabled. Buffer wrapping: disabled  
Total captured so far: 263 packets. Capture capacity : 4096 packets
```

```
----- Punt Packet Number: 1, Timestamp: 2020/04/10 18:15:53.499 -----  
interface : physical: GigabitEthernet1/0/1[if-id: 0x00000008], pal: Vlan20 [if-id: 0x00000076]  
metadata  : cause: 29 [RP handled ICMP], sub-cause: 0, q-no: 6, linktype: MCP_LINK_TYPE_IP [1]  
ether hdr  : dest mac: 084f.a940.fa56, src mac: 380e.4d77.4f66  
ether hdr  : vlan: 20, ethertype: 0x8100  
ipv4  hdr  : dest ip: 10.11.0.3, src ip: 10.11.0.3  
ipv4  hdr  : packet len: 40, ttl: 255, protocol: 17 (UDP)  
udp    hdr  : dest port: 3785, src port: 49152
```

Punt cause & CPU queue #

```
----- Punt Packet Number: 2, Timestamp: 2020/04/10 18:15:53.574 -----  
--snip--
```

```
C9300#show platform software fed switch active punt packet-capture detailed  
Punt packet capturing: disabled. Buffer wrapping: disabled  
Total captured so far: 263 packets. Capture capacity : 4096 packets
```

```
----- Punt Packet Number: 1, Timestamp: 2020/04/10 18:15:53.499 -----  
interface : physical: GigabitEthernet1/0/1[if-id: 0x00000008], pal: Vlan20 [if-id: 0x00000076]  
metadata  : cause: 29 [RP handled ICMP], sub-cause: 0, q-no: 6, linktype: MCP_LINK_TYPE_IP [1]  
ether hdr  : dest mac: 084f.a940.fa56, src mac: 380e.4d77.4f66  
--snip--
```

CPU Path Packet capture – NetDr Style

```
C9300#show platform software fed switch active punt packet-capture status  
Punt packet capturing: disabled. Buffer wrapping: disabled  
Total captured so far: 263 packets. Capture capacity : 4096 packets
```

“inject” keyword is supported as well

• Tunable Buffer Size:

```
C9300#debug platform software fed switch active punt packet-capture buffer limit ?  
<256-16384> Number of packets to capture
```

```
C9300#debug platform software fed switch active punt packet-capture buffer limit 5000  
Punt PCAP buffer configure: one-time with buffer size 5000...done
```

- Buffer size setting is in packets, not bytes (default is 4096)
- Stop capturing when buffer size limit is reached (no buffer wrapping)

• Buffer Wrapping:

```
C9300#debug platform software fed switch active punt packet-capture buffer circular limit ?  
<256-16384> Number of packets to capture
```

```
C9300#debug platform software fed switch active punt packet-capture buffer circular limit 6000  
Punt PCAP buffer configure: circular with buffer size 6000...done
```

Circular = over-write old data when buffer size limit is reached (buffer wrapping)

CPU Path Packet capture – NetDr Style

- Display Filter Support:

```
C9300#show platform software fed switch active punt packet-capture display-filter "ip.src == 10.0.17.0/31" brief
Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 121 packets. Capture capacity : 4096 packets
```

```
----- Punt Packet Number: 4, Timestamp: 2019/07/04 18:34:52.511 -----
interface : physical: [if-id: 0x00000000], pal: CPU L3 Inject [if-id: 0x01000001]
metadata  : cause: 24 [Glean adjacency], sub-cause: 0, q-no: 14, linktype: MCP_LINK_TYPE_IP [1]
ether hdr  : dest mac: 0cd0.f89c.7f80, src mac: 0cd0.f89c.7f80
ether hdr  : ethertype: 0x0800 (IPv4)
ipv4 hdr   : dest ip: 10.0.17.3, src ip: 10.0.17.1
```

```
--snip--
```

```
C9300#show platform software fed switch active punt packet-capture display-filter-help
```

```
Generic filters supported :
```

```
 7. arp          Is this an ARP packet
 8. bootp        DHCP packets [Macro]
```

```
--snip--
```

← This is not Wireshark and so not all Wireshark filters are supported (See next slide for more details)

- Capture Filter Support:

```
C9300#debug platform software fed switch active punt packet-capture set-filter "ip.src == 1.1.1.0/24 && tcp.port == 179"
Filter setup successful. Captured packets will be cleared
```

```
C9300#show platform software fed switch active punt packet-capture status
Punt packet capturing: disabled. Buffer wrapping: enabled (wrapped 0 times)
Total captured so far: 0 packets. Capture capacity : 16000 packets
Capture filter : "ip.src == 1.1.1.0/24 && tcp.port == 179"
```

```
C9300#debug platform software fed switch active punt packet-capture clear-filter
Filter cleared. Captured packets will be cleared
```

CPU Path Packet capture – NetDr Style

- Display Filter Support:

```
C9300#show platform software fed switch active punt packet-capture display-filter-help
```

FED Punject specific filters :

- | | |
|------------------|------------------------------|
| 1. fed.cause | FED punt or inject cause |
| 2. fed.linktype | FED linktype |
| 3. fed.pal_if_id | FED platform interface ID |
| 4. fed.phy_if_id | FED physical interface ID |
| 5. fed.queue | FED Doppler hardware queue |
| 6. fed.subcause | FED punt or inject sub cause |

Generic filters supported :

- | | |
|-----------------|--|
| 7. arp | Is this an ARP packet |
| 8. bootp | DHCP packets [Macro] |
| 9. cdp | Is this a CDP packet |
| 10. eth | Does the packet have an Ethernet header |
| 11. eth.addr | Ethernet source or destination MAC address |
| 12. eth.dst | Ethernet destination MAC address |
| 13. eth.ig | IG bit of ethernet destination address (broadcast/multicast) |
| 14. eth.src | Ethernet source MAC address |
| 15. eth.type | Ethernet type |
| 16. gre | Is this a GRE packet |
| 17. icmp | Is this a ICMP packet |
| 18. icmp.code | ICMP code |
| 19. icmp.type | ICMP type |
| 20. icmpv6 | Is this a ICMPv6 packet |
| 21. icmpv6.code | ICMPv6 code |
| 22. icmpv6.type | ICMPv6 type |

- | | |
|--------------------|---------------------------------------|
| 23. ip | Does the packet have an IPv4 header |
| 24. ip.addr | IPv4 source or destination IP address |
| 25. ip.dst | IPv4 destination IP address |
| 26. ip.flags.df | IPv4 dont fragment flag |
| 27. ip.flags.mf | IPv4 more fragments flag |
| 28. ip.frag.offset | IPv4 fragment offset |
| 29. ip.proto | Protocol used in datagram |
| 30. ip.src | IPv4 source IP address |
| 31. ip.ttl | IPv4 time to live |
| 32. ipv6 | Does the packet have an IPv4 header |
| 33. ipv6.addr | IPv6 source or destination IP address |
| 34. ipv6.dst | IPv6 destination IP address |
| 35. ipv6.hlim | IPv6 hop limit |
| 36. ipv6.nxt | IPv6 next header |
| 37. ipv6.plen | IPv6 payload length |
| 38. ipv6.src | IPv6 source IP address |
| 39. stp | Is this a STP packet |
| 40. tcp | Does the packet have a TCP header |
| 41. tcp.dstport | TCP destination port |
| 42. tcp.port | TCP source OR destination port |
| 43. tcp.srcport | TCP source port |
| 44. udp | Does the packet have a UDP header |
| 45. udp.dstport | UDP destination port |
| 46. udp.port | UDP source OR destination port |
| 47. udp.srcport | UDP source port |
| 48. vlan.id | Vlan ID (dot1q or qinq only) |
| 49. vxlan | Is this a VXLAN packet |

CPU Path - FED Log Improvements (Debug)

```

C9300#request platform software trace rotate all
C9300#set platform software trace fed switch active punt debug
C9300#show logging process fed internal
--snip--
executing cmd on chassis 1 ...
sending cmd to chassis 2 ...
Collecting files from chassis 2.
--snip--
2020/04/10 21:05:42.913464 {fed_F0-0}{1}: [punt] [31792]: (debug): ***** Received Packet from Doppler *****
2020/04/10 21:05:42.913465 {fed_F0-0}{1}: [punt] [31792]: (debug): RX: FirstHeaderType 0
2020/04/10 21:05:42.913529 {fed_F0-0}{1}: [punt] [31792]: (debug): Packet Decode:
  ARPA
    Destination MAC      : 0100.5e00.0005
    Source MAC           : f4db.e6ba.c8c9
    Type                 : 0x8100 (DOT1Q)
  DOT1Q
    VLAN ID              : 34
    Canonical Form Ind   : 0
    Priority              : 6
    Type                 : 0x0800 (IPV4)
  IPv4
    --snip--
    Protocol             : 89 (OSPF)
    Header Checksum      : 0xb8d3
    Source Address       : 10.20.0.2
    Destination Address  : 224.0.0.5
  OSPF
    Version              : 2
    Type                 : 1 (Hello)
--snip--

```

← "inject" keyword is supported as well

← New - The packet is now decoded like Wireshark

CPU Path – FED Log Improvements (Verbose)

```

C9300#set platform software trace fed switch active punt verbose ←
C9300#show logging process fed internal
--snip--
2020/04/10 20:47:35.676420 {fed_F0-0}{1}: [punt] [31792]: (debug): Packet Decode:
--snip--
  DOT1Q
    VLAN ID      : 35
--snip--
  Protocol       : 89 (OSPF)
  Header Checksum : 0x78fd
  Source Address  : 10.21.0.3
  Destination Address : 224.0.0.5
  OSPF
    Version      : 2
    Type         : 1 (Hello)
--snip--
2020/04/10 20:47:35.676505 {fed_F0-0}{1}: [punt] [31792]: (verbose): Dumping packet details ...
2020/04/10 20:47:35.676510 {fed_F0-0}{1}: [punt] [31792]: (debug): ===== Common Header =====
2020/04/10 20:47:35.676510 {fed_F0-0}{1}: [punt] [31792]: (debug): Hdr version: 1, Hdr type: 1, cpp_id: 0, epoch: 0
2020/04/10 20:47:35.676511 {fed_F0-0}{1}: [punt] [31792]: (debug): Packet_len: 0x7e, Total_len: 0x9e, Hdr len: 0x20
2020/04/10 20:47:35.676512 {fed_F0-0}{1}: [punt] [31792]: (debug): Feature Hdr len: 0x0, Platform Hdr len: 0x0
2020/04/10 20:47:35.676512 {fed_F0-0}{1}: [punt] [31792]: (debug): Network start offset: 0x12, Vrf 0x0, Flags: 0x8000
2020/04/10 20:47:35.676513 {fed_F0-0}{1}: [punt] [31792]: (debug): Linktype      : 0x1 [MCP_LINK_TYPE_IP]
2020/04/10 20:47:35.676514 {fed_F0-0}{1}: [punt] [31792]: (debug): Pal_if_handle : 0x7c [Vlan35]
2020/04/10 20:47:35.676514 {fed_F0-0}{1}: [punt] [31792]: (debug): Inject intf   : 0x0
2020/04/10 20:47:35.676515 {fed_F0-0}{1}: [punt] [31792]: (debug): Cause        : 0x37 [For-us control] ←
2020/04/10 20:47:35.676515 {fed_F0-0}{1}: [punt] [31792]: (debug): Sub Cause    : 0x0
2020/04/10 20:47:35.676516 {fed_F0-0}{1}: [punt] [31792]: (debug): =====
--snip--

```

“inject” keyword is supported as well

New - Cause for punt added

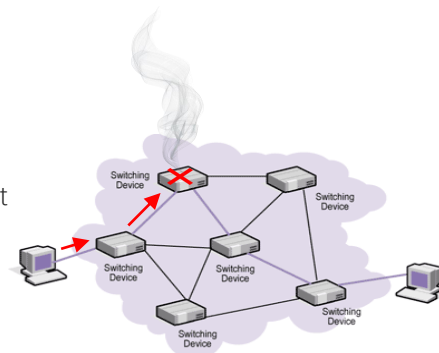
Additional log output clean-up was also done

CPU Path – Other Packet Capture Enhancements

- These enhancements were covered in the [Packet Tracer](#) section

SPF = show platform forward (Packet Tracer) - CPU inject forwarding decision support

PSV = Packet State Vector (Packet Tracer) - CPU Punt Analysis extended to FED



CPU Punt Rates - Summary

```
c9300-stack#show platform software fed switch active punt cpuq rates
Punt Rate CPU Q Statistics
```

```
Packets per second averaged over 10 seconds, 1 min and 5 mins
```

Q no	Queue Name	Rx 10s	Rx 1min	Rx 5min	Drop 10s	Drop 1min	Drop 5min
0	CPU_Q_DOT1X_AUTH	0	0	0	0	0	0
1	CPU_Q_L2_CONTROL	0	0	0	0	0	0
2	CPU_Q_FORUS_TRAFFIC	0	0	0	0	0	0
3	CPU_Q_ICMP_GEN	0	0	0	0	0	0
4	CPU_Q_ROUTING_CONTROL	0	0	0	0	0	0
5	CPU_Q_FORUS_ADDR_RESOLUTION	0	0	0	0	0	0
6	CPU_Q_ICMP_REDIRECT	0	0	0	0	0	0
--snip--							
22	CPU_Q_PUNT_WEBAUTH	0	0	0	0	0	0
23	CPU_Q_HIGH_RATE_APP	0	0	0	0	0	0
24	CPU_Q_EXCEPTION	0	0	0	0	0	0
25	CPU_Q_SYSTEM_CRITICAL	0	0	0	0	0	0
26	CPU_Q_NFL_SAMPLED_DATA	0	0	0	0	0	0
27	CPU_Q_LOW_LATENCY	0	0	0	0	0	0
28	CPU_Q_EGR_EXCEPTION	0	0	0	0	0	0
29	CPU_Q_FSS	0	0	0	0	0	0
30	CPU_Q_MCAST_DATA	0	0	0	0	0	0
31	CPU_Q_GOLD_PKT	0	0	0	0	0	0

- CPU traffic punt rates from hardware (FED) towards the CPU for all 32 CPU Queues

CPU Punt Rate – Per Interface

```
C9300#show platform software fed switch active punt rates interfaces
```

```
Punt Rate on Interfaces Statistics
```

```
Packets per second averaged over 10 seconds, 1 min and 5 mins
```

Interface Name	IF_ID	Recv 10s	Recv 1min	Recv 5min	Drop 10s	Drop 1min	Drop 5min
GigabitEthernet1/0/1	0x00000000	22	21	21	0	0	0
GigabitEthernet2/0/1	0x00000032	2	2	2	0	0	0
Vlan20	0x00000076	19	18	19	0	0	0

- CPU traffic punt rates from hardware (FED) towards the CPU per interfaces
- Only non-zero traffic statistics are displayed

```
C9300#show platform software fed switch active punt rates interfaces 0x32
```

```
Punt Rate on Single Interfaces Statistics
```

```
Interface : GigabitEthernet2/0/1 [if_id: 0x32]
```

```
Received
```

```
Dropped
```

```
-----
```

```
-----
```

```
Total          : 1580205
10 sec average  : 2
1 min average   : 2
5 min average   : 2
```

```
Total          : 0
10 sec average  : 0
1 min average   : 0
5 min average   : 0
```

```
Per CPUQ punt stats on the interface (rate averaged over 10s interval)
```

Q no	Queue Name	Recv Total	Recv Rate	Drop Total	Drop Rate
0	CPU_Q_DOT1X_AUTH	0	0	0	0
1	CPU_Q_L2_CONTROL	39948	0	0	0
31	CPU_Q_GOLD_PKT	0	0	0	0

- CPU traffic punt rates from UADP (FED) towards the CPU for all 32 CPU queues for a specific interface

CoPP - Per Class Traffic Match Details

```
C9400#show platform software qos copp class-info
ACL representable classmap filters are displayed:
  class-map match-any system-cpp-police-data
    description ICMP Redirect, ICMP_GEN and All types BROADCAST
    match access-group name system-cpp-mac-match-police-data
    mac access-list extended system-cpp-mac-match-police-data
      permit any host FFFF.FFFF.FFFF
      permit any any arp arp-reply
      permit any any arp arp-request
  class-map match-any system-cpp-police-sys-data
    description Openflow, Exception, EGR Exception,
    NFL Sampled Data, MSP,
    RPF Failed, Data - mRPF failed
  class-map match-any system-cpp-police-sw-forward
    description Sw forwarding, LOGGING,
    L2 LVX Data Packets, Transit Traffic,
    Unknown unicast packet punted for map request.
  match access-group name system-cpp-ipv4-match-ip-options
  match access-group name system-cpp-ipv4-match-ipv6-options
  ip access-list extended system-cpp-ipv4-match-ip-options
    permit ip any 224.0.0.0 15.255.255.255 option any-option
  ipv6 access-list system-cpp-ipv4-match-ipv6-options
    permit ipv6 any any dest-option
-snip-
```

Control Plane Policing (CoPP) is an existing CPU Protection Feature

A new “class-info” command was added to provide per class traffic match details:

- 16.6.1 - Catalyst 9500H (Doppler E)
- 17.2.1 - Remaining Catalyst 9000 series switches

Packet Inject Tool – Manually Crafted Packet

```
C9300#monitor capture mycap match any control-plane in
C9300#monitor capture mycap start
```

```
C9300#test platform software inject interface gig 1/0/1 ?
  arp      arp packet
  capture  Capture packet
  dhcp     dhcp packet
  raw      Packet to be injected
  repeat   Repeat packet
C9300#test platform software inject interface gig 1/0/1 arp request ?
  H.H.H    Source mac address
C9300#test platform software inject interface gig 1/0/1 arp request 0001.1111.2222 ?
  A.B.C.D  IPv4 Target Address
C9300#test platform software inject interface gig 1/0/1 arp request 0001.1111.2222 10.11.0.20 10.11.0.21
Repeat inject below packet 1 number of times on intf:GigabitEthernet1/0/1
sending following packet with len 64:

7F0A9BFC1C50: FFFFFFFF FFFF0001 11112222 08060001  ...."....
7F0A9BFC1C60: 08000604 00010001 11112222 0A0B0015  ...."....
7F0A9BFC1C70: 00000000 00000A0B 00140000 0000A940  .....)@
7F0A9BFC1C80: FA56380E 4D774F66 8100C014 080045C0  zV8.MwOf..@...E@
7F0A9BFC1C90: 00                                .
```

Packet Inject Options

Inject manually crafted ARP request packet inbound on interface Gig 1/0/1

```
C9300#monitor capture mycap stop
C9300#show monitor capture mycap buffer brief
Starting the packet display ..... Press Ctrl + Shift + 6 to exit
--snip--
```

```
101  4.727701 f4:db:e6:ba:c8:82 01:00:0c:cc:cc:cd STP 68 RST. Root = 32768/99/38:0e:4d:77:4f:40
102  4.766051 00:01:11:11:22:22 ff:ff:ff:ff:ff:ff ARP 64 Who has 10.11.0.20? Tell 10.11.0.21
```

the injected ARP request packet is seen in the CPU punt path

Packet Inject Tool – From PCAP File

CAUTION – be careful when injecting packets into a production network !!!

Best use is for lab testing

```
C9300L#monitor capture cap1 match ipv4 any 10.12.0.2/32 interface gigabitEthernet 1/0/1 in
C9300L#monitor capture cap1 file location flash:cap1.pcap
C9300L#monitor capture cap1 start ---> send test ping request now
C9300L#monitor capture cap1 stop
C9300L#show monitor capture file flash:cap1.pcap
Starting the packet display ..... Press Ctrl + Shift + 6 to exit
--snip--
  4   0.003877   10.23.0.1  10.12.0.2   ICMP 118 Echo (ping) request id=0x0005, seq=3/768, ttl=254
```

Capture a packet on ingress interface Gig 1/0/1 (or use an existing Wireshark pcap file)

```
C9300L#monitor capture test match ipv4 any 10.12.0.2/32 interface gigabitEthernet 1/0/5 both
C9300L#monitor capture test start
```

Setup monitoring on egress interface Gig 1/0/5 prior to packet inject

```
C9300L#test platform software inject interface gig 1/0/1 repeat 2 capture flash:cap1.pcap 4
Repeat inject below packet 2 number of times on intf:GigabitEthernet1/0/1
sending following packet with len 118:
```

Inject 2 copies of packet #4 from the PCAP file inbound on interface Gig 1/0/1

```
7FB0B2AE2920:          F4DBE6BA C8C5084F          t[f:HE.O
7FB0B2AE2930: A940FA45 81000025 08004500 00640018 )@zE...%.E...d..
--snip--
7FB0B2AE2990: ABCDABCD ABCDABCD ABCDABCD ABCD00  +M+M+M+M+M+M+M.
```

```
C9300L#monitor capture test stop
C9300L#show monitor capture test buffer brief
Starting the packet display ..... Press Ctrl + Shift + 6 to exit
```

```
 1   0.000000   10.23.0.1  10.12.0.2   ICMP 118 Echo (ping) request id=0x0005, seq=3/768, ttl=254
 2   0.000052   10.23.0.1  10.12.0.2   ICMP 118 Echo (ping) request id=0x0005, seq=3/768, ttl=254
 3   0.000847   10.12.0.2  10.23.0.1   ICMP 114 Echo (ping) reply id=0x0008, seq=3/768, ttl=254
 4   0.000949   10.12.0.2  10.23.0.1   ICMP 114 Echo (ping) reply id=0x0008, seq=3/768, ttl=254
```

- 2 injected packets seen on egress interface Gig 1/0/5
- 2 replies are received back

Polling Question 3

Which of the following serviceability tools that we're not covering today you'd like to know more about?

- A. Hardware Programming,
- B. Logging (Binary tracing, radioactive trace, etc.)
- C. IoT Switching

Submit Your
Questions Now!



Use the Q&A panel to submit your
questions, our expert will respond

Ask Me Anything following the event

Now through Friday December 11, 2020

Use Serviceability Features to Troubleshoot
your Cat9K as a Cisco TAC Engineer

With Aaron Cespedes

Participate: <https://bit.ly/AMA-cat9k>



Aaron Cespedes
Technical Consulting Engineer



Collaborate within our Social Media



Twitter

- @Cisco_Support
- <http://bit.ly/csc-twitter>

Facebook

- Cisco Community
- <http://bit.ly/csc-facebook>

Learn About Upcoming Events

We invite you to review our Social Media Channels

YouTube

- Cisco Community
- <http://bit.ly/csc-youtube>



App

- Cisco Technical Support



LinkedIn

- Cisco Community
- <http://bit.ly/csc-linked-in>



Cisco has support communities in other languages!

If you speak Spanish, Portuguese, Japanese, Russian or Chinese we invite you to participate & collaborate





More IT Training Videos and Technical Seminars on the Cisco Learning Network

View Upcoming Sessions Schedule
<https://cisco.com/go/techseminars>

Thank you for Your
Time!

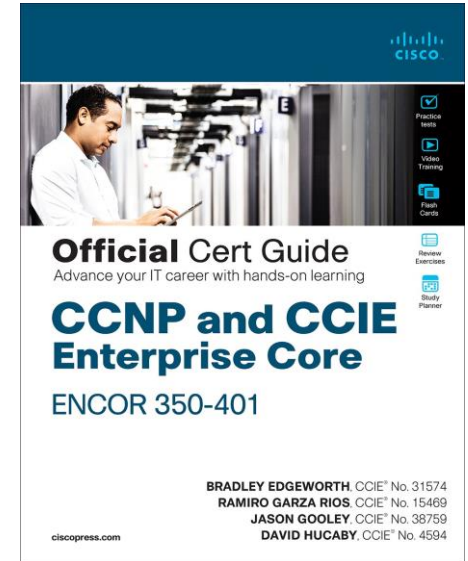
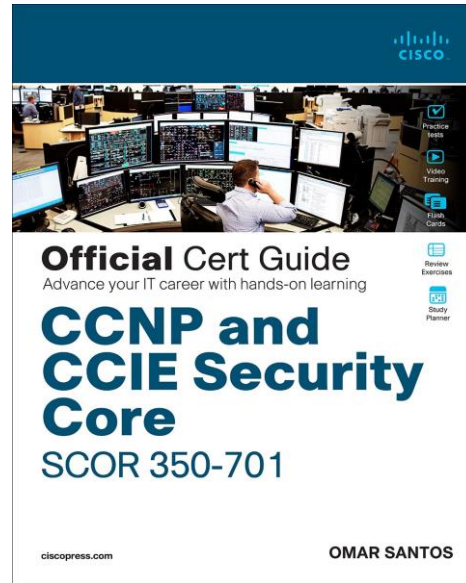
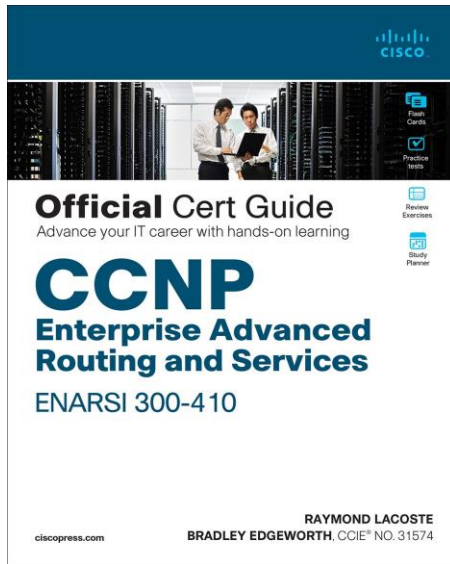
Please take a moment to complete
the survey



Thank you for participating, you earned a discount!

Redeem your 35% discount offer by entering code: CSC when checking out.

<http://bit.ly/Community-CiscoPress2020>



Thanks For Joining today!

