



Troubleshooting Packet Flow Issues

- [Packet Flow Issues, on page 1](#)
- [Monitoring Inband Packet Statistics, on page 2](#)
- [Fabric Connectivity Commands, on page 3](#)
- [Troubleshooting Packet Flow with Packet Tracer, on page 6](#)

Packet Flow Issues

Packets could be dropped for the following reasons:

- Software-switched packets could be dropped because of Control Plane Policing (CoPP).
- Hardware-switched packets could be dropped by the hardware because of a bandwidth limitation.

Beginning with Cisco NX-OS Release 10.3.(1)F, the following CLIs are supported on Cisco Nexus 9300 and 9500 Cloud Scale switches.

- **show hardware internal statistics module-all all**: Displays the statistics of active modules.
- **show hardware internal statistics module <module-no> all**: Displays the statistics of a particular active module from supervisor.

Packets Dropped Because of Rate Limits

Use the **show hardware rate-limit** command to determine if packets are being dropped because of a rate limit.

```
switch(config)# show hardware rate-limit module 1
```

Units for Config: packets per second

Allowed, Dropped & Total: aggregated since last clear counters

Rate Limiter Class	Parameters	

access-list-log	Config	: 100
	Allowed	: 0
	Dropped	: 0
	Total	: 0

Packets Dropped Because of CoPP

Use the **show policy-map interface control-plane** command to determine if packets are being dropped because of CoPP.

```
switch# show policy-map interface control-plane
  class-map copp-system-p-class-exception (match-any)
    match exception ip option
    match exception ip icmp unreachable
    match exception ttl-failure
    match exception ipv6 option
    match exception ipv6 icmp unreachable
    match exception mtu-failure
    set cos 1
    police cir 200 pps , bc 32 packets

  module 27 :
    transmitted 0 packets;
    dropped 0 packets;

  module 28 :
    transmitted 0 packets;
    dropped 0 packets;
```

Monitoring Inband Packet Statistics

Use the **show hardware internal cpu-mac inband counters** command to display inband packet statistics for supervisor modules, fabric modules, and line cards.

```
switch# show hardware internal cpu-mac inband counters
eth2 counters:
eth2      Link encap:Ethernet  HWaddr 00:00:00:01:1b:01
          BROADCAST MULTICAST  MTU:9400  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 counters:
eth3      Link encap:Ethernet  HWaddr 00:00:00:01:1b:01
          inet6 addr: fe80::200:ff:fe01:1b01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9400  Metric:1
          RX packets:425432 errors:0 dropped:0 overruns:0 frame:0
          TX packets:352432 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:253284953 (241.5 MiB)  TX bytes:249647978 (238.0 MiB)

ps-inb counters:
ps-inb    Link encap:Ethernet  HWaddr 00:00:00:01:1b:01
          inet6 addr: fe80::200:ff:fe01:1b01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9400  Metric:1
          RX packets:128986 errors:0 dropped:0 overruns:0 frame:0
          TX packets:129761 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:221538103 (211.2 MiB)  TX bytes:227158091 (216.6 MiB)

switch# slot 22 show hardware internal cpu-mac inband counters
inband0 counters:
inband0   Link encap:Ethernet  HWaddr 00:00:00:01:16:03
```

```

inet addr:127.2.2.22 Bcast:127.2.255.255 Mask:255.255.0.0
inet6 addr: fe80::200:ff:fe01:1603/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:9676 Metric:1
RX packets:147425 errors:0 dropped:0 overruns:0 frame:0
TX packets:147470 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:532
RX bytes:15479625 (14.7 MiB) TX bytes:14898335 (14.2 MiB)
Interrupt:10

knet0_0 counters:
knet0_0 Link encap:Ethernet HWaddr 02:10:18:e1:6f:50
inet6 addr: fe80::10:18ff:fe01:6f50/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:9400 Metric:1
RX packets:36 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:6 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:3816 (3.7 KiB) TX bytes:0 (0.0 B)

knet0_1 counters:
knet0_1 Link encap:Ethernet HWaddr 02:10:18:e1:6f:51
inet6 addr: fe80::10:18ff:fe01:6f51/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:9400 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:6 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

```

Fabric Connectivity Commands

Cisco NX-OS provides the following commands to display information and statistics related to fabric connectivity:

- **show system internal fabric connectivity [module module-number]**—Displays connectivity information for all fabric modules or a single module.

```

switch# show system internal fabric connectivity
HiGIG Link-info Linecard slot:4

```

LC-Slot	LC-Unit	LC-HGLink	FM-Slot	FM-Unit	FM-HGLink
4	0	HG02	22	0	HG09
4	0	HG03	22	1	HG09
4	0	HG06	24	0	HG09
4	0	HG07	24	1	HG09
4	1	HG02	22	0	HG10
4	1	HG03	22	1	HG10
4	1	HG06	24	0	HG10
4	1	HG07	24	1	HG10
4	2	HG02	22	0	HG11
4	2	HG03	22	1	HG11
4	2	HG06	24	0	HG11
4	2	HG07	24	1	HG11

```

HiGIG Link-info Fabriccard slot:22

```

FM-Slot	FM-Unit	FM-HGLink	LC-Slot	LC-Unit	LC-HGLink
22	0	HG09	4	0	HG02
22	0	HG10	4	1	HG02
22	0	HG11	4	2	HG02

```

22      1      HG09      4      0      HG03
22      1      HG10      4      1      HG03
22      1      HG11      4      2      HG03

```

HiGIG Link-info Fabriccard slot:24

```

-----
FM-Slot  FM-Unit  FM-HGLink  LC-Slot  LC-Unit  LC-HGLink
-----
24        0        HG09        4        0        HG06
24        0        HG10        4        1        HG06
24        0        HG11        4        2        HG06
24        1        HG09        4        0        HG07
24        1        HG10        4        1        HG07
24        1        HG11        4        2        HG07

```

- **show system internal interface counters module module-number [nz]**—Displays the rates for HG or fabric links on a module. The **nz** option displays only non-zero counters.

```

switch# show system internal interface counters module 22 nz
Internal Port Counters (150 secs rate) for Slot: 22
=====

```

```

Interface ASIC ASIC  BCM  TxBitRate(BwUtil) TxPktRate RxBitRate(BwUtil) RxPktRate
          Port Inst Port          (bps)          (pps)          (bps)          (pps)
-----
ii22/1/10 HG9    0   10    0( 0.00)          0          33064( 0.00)          17

```

```

switch# show system internal interface counters module 22
Internal Port Counters (150 secs rate) for Slot: 22
=====

```

```

Interface ASIC ASIC  BCM  TxBitRate(BwUtil) TxPktRate RxBitRate(BwUtil) RxPktRate
          Port Inst Port          (bps)          (pps)          (bps)          (pps)
-----
ii22/1/1  HG0    0   1    0( 0.00)          0          0( 0.00)          0
ii22/1/2  HG1    0   2    0( 0.00)          0          0( 0.00)          0
ii22/1/3  HG2    0   3    0( 0.00)          0          0( 0.00)          0
ii22/1/4  HG3    0   4    0( 0.00)          0          0( 0.00)          0
ii22/1/5  HG4    0   5    0( 0.00)          0          0( 0.00)          0
ii22/1/6  HG5    0   6    0( 0.00)          0          0( 0.00)          0
ii22/1/7  HG6    0   7    0( 0.00)          0          0( 0.00)          0
ii22/1/8  HG7    0   8    0( 0.00)          0          0( 0.00)          0
ii22/1/9  HG8    0   9    0( 0.00)          0          0( 0.00)          0
ii22/1/10 HG9    0  10    0( 0.00)          0          30888( 0.00)          12
ii22/1/11 HG10   0  11    0( 0.00)          0          0( 0.00)          0
ii22/1/12 HG11   0  12    0( 0.00)          0          0( 0.00)          0
ii22/1/13 HG12   0  13    0( 0.00)          0          0( 0.00)          0
ii22/1/14 HG13   0  14    0( 0.00)          0          0( 0.00)          0
ii22/1/15 HG14   0  15    0( 0.00)          0          0( 0.00)          0
ii22/1/16 HG15   0  16    0( 0.00)          0          0( 0.00)          0
ii22/1/17 HG16   0  17    0( 0.00)          0          0( 0.00)          0
ii22/1/18 HG17   0  18    0( 0.00)          0          0( 0.00)          0
ii22/1/19 HG18   0  19    0( 0.00)          0          0( 0.00)          0
ii22/1/20 HG19   0  20    0( 0.00)          0          0( 0.00)          0
ii22/1/21 HG20   0  21    0( 0.00)          0          0( 0.00)          0
ii22/1/22 HG21   0  22    0( 0.00)          0          0( 0.00)          0
ii22/1/23 HG22   0  23    0( 0.00)          0          0( 0.00)          0
ii22/1/24 HG23   0  24    0( 0.00)          0          0( 0.00)          0
ii22/1/33 HG0    1   1    0( 0.00)          0          0( 0.00)          0
ii22/1/34 HG1    1   2    0( 0.00)          0          0( 0.00)          0
ii22/1/35 HG2    1   3    0( 0.00)          0          0( 0.00)          0
ii22/1/36 HG3    1   4    0( 0.00)          0          0( 0.00)          0
ii22/1/37 HG4    1   5    0( 0.00)          0          0( 0.00)          0

```

```

ii22/1/38 HG5 1 6 0( 0.00) 0 0( 0.00) 0
ii22/1/39 HG6 1 7 0( 0.00) 0 0( 0.00) 0
ii22/1/40 HG7 1 8 0( 0.00) 0 0( 0.00) 0
ii22/1/41 HG8 1 9 0( 0.00) 0 0( 0.00) 0
ii22/1/42 HG9 1 10 0( 0.00) 0 0( 0.00) 0
ii22/1/43 HG10 1 11 0( 0.00) 0 0( 0.00) 0
ii22/1/44 HG11 1 12 0( 0.00) 0 0( 0.00) 0
ii22/1/45 HG12 1 13 0( 0.00) 0 0( 0.00) 0
ii22/1/46 HG13 1 14 0( 0.00) 0 0( 0.00) 0
ii22/1/47 HG14 1 15 0( 0.00) 0 0( 0.00) 0
ii22/1/48 HG15 1 16 0( 0.00) 0 0( 0.00) 0
ii22/1/49 HG16 1 17 0( 0.00) 0 0( 0.00) 0
ii22/1/50 HG17 1 18 0( 0.00) 0 0( 0.00) 0
ii22/1/51 HG18 1 19 0( 0.00) 0 0( 0.00) 0
ii22/1/52 HG19 1 20 0( 0.00) 0 0( 0.00) 0
ii22/1/53 HG20 1 21 0( 0.00) 0 0( 0.00) 0
ii22/1/54 HG21 1 22 0( 0.00) 0 0( 0.00) 0
ii22/1/55 HG22 1 23 0( 0.00) 0 0( 0.00) 0
ii22/1/56 HG23 1 24 0( 0.00) 0 0( 0.00) 0

```

- **show system internal interface counters detail module *module-number***—Displays detailed statistics for all HG or fabric links on a single module.

```

show system internal interface counters detail module 4
.....
Interface: ii4/1/3 ASIC Inst# 0/Port# 3/Name HG2
-----
Last Cleared @ Thu Jan 1 00:00:00 2013
(0)
Tx/Rx Rates (per second):
      secs      tx bytes      tx packets      rx bytes      rx packets
[0] - 10         0           0           0           0
[1] - 150       9448         60          0           0
[2] - 300       9448         60          0           0
Mac Pktflow:
Rx Counters:
  Ingress Packets : 0x0000000000000000/0
  Unicast Packets : 0x0000000000000000/0
  Multicast Packets: 0x0000000000000000/0
  Broadcast Packets: 0x0000000000000000/0
  Jumbo Packets   : 0x0000000000000000/0
  Total Bytes     : 0x0000000000000000/0

Rx Bytes by Packet Size:
64: 0x0000000000000000/0
65 - 127: 0x0000000000000000/0
128 - 255: 0x0000000000000000/0
256 - 511: 0x0000000000000000/0
512 - 1023: 0x0000000000000000/0
1024 - 1518: 0x0000000000000000/0
1519 - 1548: 0x0000000000000000/0

Tx Counters:
  Egress Packets : 0x0000000000001351/4945
  Unicast packets: 0x0000000000001351/4945
  Multicast packets: 0x0000000000000000/0
  Broadcast Packets: 0x0000000000000000/0
  Jumbo Packets   : 0x0000000000000000/0
  Undersize Packets: 0x0000000000000000/0
  Total Bytes     : 0x00000000000008e756/583510

Tx Bytes by Packet Size

```

```

64:          0x0000000000000000/0
65  - 127:   0x00000000000001351/4945
128 - 255:   0x0000000000000000/0
256 - 511:   0x0000000000000000/0
512 - 1023:  0x0000000000000000/0
1024 - 1518: 0x0000000000000000/0
1519 - 1548: 0x0000000000000000/0
trunk:      0x0000000000000000/0

Mac Control:
Rx Pause:   0x0000000000000000/0
Tx Pause:   0x0000000000000000/0
Reset:      0x0000000000000000/0
Mac Errors:
Undersize:  0x0000000000000000/0
Runt:       0x0000000000000000/0
Crc:        0x0000000000000000/0
Input Errors: 0x0000000000000000/0
In Discard: 0x0000000000000000/0
Giants:     0x0000000000000000/0
Output Errors: 0x0000000000000000/0
Output Discard: 0x0000000000000000/0
Bad Proto:  0x0000000000000000/0
Collision:  0x0000000000000000/0
Late Collision: 0x0000000000000000/0
No Carrier: 0x0000000000000000/0

```

Troubleshooting Packet Flow with Packet Tracer

Packet Tracer

The Packet Tracer is a new troubleshooting tool that allows a packet to be captured from the Network Processor. Similar to the ELAM tool available on Cisco Nexus 9000 Cloud Scale switches, this tool provides information to understand how the ASIC forwarded the captured packet. This information is useful to troubleshoot packet flow.

While various tools like SPAN, ERSPAN, Ethalyzer exist to debug packet flow issues, Packet Tracer allows troubleshooting within the forwarding pipeline of an ASIC without any performance penalty or disruption to the environment.

Packet tracer has the following capabilities:



Important

To effectively use Packet Tracer, users should have a comprehensive understanding of the ASIC forwarding pipeline. This knowledge is crucial for setting precise filters and accurately interpreting the results of packet captures.

- Allows filters for capturing packets based on various protocol parameters like IPv4/IPv6 addresses, TCP/UDP ports, and so on.
- Provides flexible filtering functionality at the ASIC level due to matching incoming packets using a 128-byte filter and 128-byte mask. This allows the filters to be designed using packet protocol parameters or specific byte sequences in the payload portion of the packet.

- Captures 128 bytes of packet along with the state of the forwarding pipeline because the packet passes through various stages of the pipeline.
- Allows creation of filters using non-protocol information like packet size, traffic class, and so on.

Packet Tracer Workflow

You must perform the following steps to capture packets using Packet Tracer.

- Determine whether the packet must be captured in the receive packet path (RxPP) or transmit packet path (TxPP).
- Determine whether a packet must be captured using protocol parameters or specific byte sequences as filters.
 - To capture packets using protocol parameters as filters: Identify the frame or packet format of the packet to be captured which will guide you to know whether the packet to be captured is an Ethernet packet followed by IPv4 or IPv6 or other well-known protocols or a VLAN tagged Ethernet packet followed by well-known protocols and so on. For more information, see [Packet Format, on page 7](#).

**Note**

The packet format serves as a template for forming the 128-byte filter and mask based on various protocol parameters that you may select as filters.

- To capture packets using specific byte sequences as filters: Identify the specific contiguous byte sequences and start offset.
- Start the Packet Tracer.
- Wait for the Packet Tracer to trigger and display the result.

Packet Format

The Packet Tracer on ASIC uses a 128-byte packet filter and a corresponding 128-byte filter mask to capture specific packets. You use the packet filter as a byte string representing the first 128 bytes of the packet of interest, with specific values at certain offsets. The ASIC matches ingress/egress packets against this byte pattern using the filter mask. This setup allows you to capture packets of any protocol type or pattern, providing flexibility in specifying the match pattern.

To build the packet filter and mask, you must typically identify protocol fields of interest, like dot1q header, IPv4 source IP, destination IP, or UDP source port, and incorporate these values at the correct offsets in the packet filter. The matching mask is then applied at the same offsets in the filter mask. Because packets are matched against the 128-byte filter and mask, you must be aware of the specific packet format to create a representative pattern including ethernet headers.

This image represents the packet format tree that can help create specific packet format.

Figure 1: Packet Format Tree

```

eth
+-- arp
+-- ipv4
| +-- tcp
| +-- udp
| | +-- vxlan
| | +-- eth
| | | +-- arp
| | | +-- llcu
| | | | +-- snap
| | | +-- llci
| | | | +-- snap
| | | +-- llcs
| | | | +-- snap
| | | +-- ipv4
| | | | +-- tcp
| | | | +-- udp
| | | | +-- icmp
| | | +-- ipv6
| | | | +-- tcp
| | | | +-- udp
| | | | +-- icmpv6
| | | +-- qinq
| | | | +-- ipv4
| | | | | +-- tcp
| | | | | +-- udp
| | | | | +-- icmp
| | | | +-- ipv6
| | | | | +-- tcp
| | | | | +-- udp
| | | | | +-- icmpv6
| | | +-- vntag
| | | | +-- ipv4
| | | | | +-- tcp
| | | | | +-- udp
| | | | | +-- icmp
| | | | +-- ipv6
| | | | | +-- tcp
| | | | | +-- udp
| | | | | +-- icmpv6
| | +-- mpls
| | | +-- mpls
| | | | +-- ipv4
| | | | +-- ipv6
| | +-- arp
+-- icmp
+-- gre
| +-- ipv4
| +-- ipv6
+-- dot1q

```

This table lists a few examples of supported packet format usage of packet tracer.

Table 1: Examples of Supported Packet Format

Interested traffic	Packet-format to use
TCP traffic with VLAN tagging	• For IPv4: eth-dot1q-ipv4-tcp • For IPv6: eth-dot1q-ipv6-tcp

Interested traffic	Packet-format to use
VXLAN traffic with VLAN tagging	- For IPv4: eth-dot1q-ipv4-udp-vxlan-eth-ipv4 - For IPv6: eth-dot1q-ipv6-udp-vxlan-eth-ipv4
VXLAN traffic with inner VLAN tagging	- For IPv4: eth-ipv4-udp-vxlan-eth-dot1q-ipv4 - For IPv6: eth-ipv6-udp-vxlan-eth-dot1q-ipv4
VXLAN traffic with inner VLAN tagging and ARP	- For IPv4: eth-ipv4-udp-vxlan-eth-dot1q-arp - For IPv6: eth-ipv6-udp-vxlan-eth-dot1q-arp
ARP traffic	eth-arp
ICMP traffic	- For IPv4: eth-ipv4-icmp - For IPv6: eth-ipv6-icmp
MPLS traffic	eth-mpls-mpls-ipv4 (MPLS can be added up to 6 times for 6 labels)

Guidelines and Limitations for Packet Tracer

The Packet Tracer has the following configuration guidelines and limitations:

- In the packet offset command, a maximum of 10 conditions can be set for RxPP or TxPP.
- Packet filter matches only on the first 128 Bytes of the packet.
- Packet tracer can be done in either RxPP or TxPP path.
- Capture mode 'continuous' is not supported.
- If slices are not specified, the output is displayed for all the slices where the packet-trace is tracked.

Supported Release and Platform for Packet Tracer

Release	Platform
10.5(3)F and later	Cisco Nexus N9364E-SG2-Q and N9364E-SG2-O switches

Deploy Packet Tracer

You can use the following commands to trigger packet tracer functionality on the switch.

Procedure

Step 1 Run the **packet-trace** command to enable the packet trace mode.

Example:

```
switch# packet-trace
switch(config-pt) #
```

Step 2 Run the **trigger init {rxpp | txpp}** command to capture the packet on Rx path (Rxpp) or Tx Path (TxPP).

Example:

```
switch(config-pt) # trigger init rxpp
switch(config-pt-rxpp) #
```

Step 3 Run the **packet-format packet-format** command to specify the packet format to trace the packets.

Example:

```
switch(config-pt-txpp) # packet-format eth-ipv4-tcp
switch(config-pt-txpp-pkt-fmt) #
```

In this example, the frame format of packet to be captured is Ethernet with IPv4 and TCP protocol filter.

Note

This command provides a User Guide option under help (?) to guide on the usage of this command.

Step 4 Run the **set outer {l2 | ipv4 | ipv6 | arp | l4 | mpls } | set inner { l2 | ipv4 | ipv6 | arp | l4 | mpls }** command to set various protocol filters for nonencapsulated packets (set outer) or encapsulated packet (set inner).

Example:

```
switch(config-pt-txpp-pkt-fmt) # set outer ipv4
```

To set filters for packet, an appropriate packet-format must be selected, see [Packet Format, on page 7](#) section.

Note

For encapsulated packet, the term "outer" is used for underlay protocols like IP/UDP and the term "inner" is used for overlay protocols like VXLAN, GRE, and so on.

Step 5 (Optional) Run the **show filters** command to set various nonpacket filters.

Example:

```
switch(config-pt-txpp-pkt-fmt) # show filters
```

These filters can be set in addition to packet header filters.

Note

Packet capture is based on both packet header filters and nonpacket filters, if set.

Step 6 Run the **start** command to set the filters on the ASIC and begins the capture operation.

Example:

```
switch(config-pt-txpp-pkt-fmt) # start
```

Step 7 (Optional) Run the **status** command to provide information on whether capture has occurred or not.

Example:

```
switch(config-pt-txpp-pkt-fmt) # status
```

This command can be run multiple times to know whether a packet has been captured or not.

Step 8 (Optional) Run the **stop** command to cancel the capture that has been started.

Example:

```
switch(config-pt-txpp-pkt-fmt) # stop
```

Note

Issuing a **stop** command after a successful capture does not alter the capture's outcome.

Step 9 (Optional) Run the **reset** command to clear the set filters.

Example:

```
switch(config-pt-txpp-pkt-fmt) # reset
switch(config-pt-txpp) #
```

Step 10 Run the **report [brief] [slice slice-ids]** command to provide details of the captured packet and dump of NPPD decode information for each stage of the capture.

Example:

```
switch(config-pt-txpp) # report detail
```

For **brief** option, NPPD decoded information will not be dumped.

If a **slice** option is specified, packet capture details of the specified slice will be displayed.

For more details of the report detail, see [Configuration Example for Packet Tracer, on page 13](#).

Step 11 Run the **set npi {err-flag value | initial-TC value | processing-code value | reassembly-ctxt value | single-frag-pkt value | src-pif value | tx-to-rx-rec-data value | unsch-rec-code value}** command to set various nonpackets in RxPP capture mode.

Example:

```
switch(config-pt-rxpp) # set npi src-pif 15
```

These filters can be set in addition to packet header filters.

Packet capture is based on both packet header filters and nonpacket filters, if set.

- **Error Flag (err-flag):** Use this option to check if any hardware errors are observed. Size: 1 bit. Range: 0–1.
- **Initial TC (initial-TC):** Use this option to set the packets initial traffic class as calculated by IFG. Size: 3 bits. Range: 0–7.
- **Processing Code (processing-code):** Use this option to indicate abnormal events. Size: 7 bits. Range: 0–127.
- **Reassembly Context (reassembly-ctxt):** Use this option to uniquely identify the packet fragments for reassembly. Size: 11 bits. Range: 0–2047.
- **Single Fragment Packet (single-frag-pkt):** Use this option to set a flag for single fragment packet. Size: 1 bit. Range: 0–1.
- **Source Port interface (src-pif):** Use this option to set the source physical interface. Size: 7 bits. Range: 0–127.
- **Tx to Rx recycle data (tx-to-rx-rec-data):** Use this option to set recycle data passed from the TxNPU to the RxNPU. Size: 8 bits. Range: 0–255.

- **Unschedule Recycle code (unsch-rec-code):** Use this option to set the type of unscheduled recycle required. Size: 4 bits. Range: 0–15.

Step 12

Run the **set npi** {**acc-LM-cache-and-Idx** *value* | **colour** *value* | **congested** *value* | **congestion-level** *value* | **cud** *value* | **dst-intf** *value* | **eop** *value* | **err-flag** *value* | **is-elephant-flow** *value* | **lm-cache-index** *value* | **omd** *value* | **pkt-size** *value* | **sop** *value* | **src-slice** *value* | **src-slice-sys-port** *value* | **start-packing** *value* | **traffic-class** *value*} command to set various nonpackets in TxPP capture mode.

Example:

```
switch(config-pt-txpp)# set npi dst-intf 25
```

These filters can be set in addition to packet header filters.

Packet capture is based on both packet header filters and nonpacket filters, if set.

- **Access LM cache and Index (acc-LM-cache-and-Idx):** Use this option to check the loss measurement request cache. Size: 1 bit. Range: 0–1.
- **Color (colour):** Use this option to check drop precedence of the packet. Size: 2 bits. Range: 0–3.
- **Congested (congested):** Use this option to check whether a congestion event occurred on the packet. Size: 1 bit. Range: 0–1.
- **Congestion Level (congestion-level):** Use this option to check measurement of the queues congestion as experienced by the packet. Size: 4 bits. Range: 0–15.
- **Copy unique data (cud):** Use this option to check why the packet copy was generated. Size: 23 bits. Range: 0–8388607.
- **Destination Interface (dst-intf):** Use this option to set the destination physical interface. Size: 8 bits. Range: 0–255.
- **End Of Packet (eop):** Use this option to set an end of a packet fragment. Size: 1 bit. Range: 0–1.
- **Error Flag (err-flag):** Use this option to check if any hardware errors are observed. Size: 1 bit. Range: 0–1.
- **Is elephant flow (is-elephant-flow):** Use this option to check if a packet is identified as part of a large flow. Size: 1 bit. Range: 0–1.
- **LM Cache Index (lm-cache-index):** Use this option to check index of the loss measurement request cache. Size: 2 bits. Range: 0–255.
- **Output queue Mapped Data (omd):** Use this option to set output queue-mapped data. Size: 9 bits. Range: 0–511.
- **Packet size (pkt-size):** Use this option to check packet size in bytes. Size: 14 bits.
- **Start Of Packet (sop):** Use this option to set the start of a packet fragment. Size: 1 bit. Range: 0–1.
- **Source Slice (src-slice):** Use this option to set source slice. Size: 3 bits.
- **Source Slice System Port (src-slice-sys-port):** Use this option to set the source slice system port. Size: 8 bits. Range: 0–255.
- **Start Packing (start-packing):** Use this option to set the first packet of a dual packet. Size: 1 bit. Range: 0–1.
- **Traffic Class (traffic-class):** Use this option to set the traffic class of the packet. Size: 3 bits. Range: 0–7.

Step 13

Run the **set pkt-offset condition** *Value* **offset** *Value* command to set the raw filter and mask to any byte pattern according to your needs.

Example:

```
switch(config-pt-txpp)# set pkt-offset condition 1 offset 0x10 value 0xababab mask 0xffffffff
```

Up to 10 such conditions can be set. This operation is mutually exclusive to set the protocol filters using the **set outer** command.

Note

- A mask of '0xf' indicates that the corresponding nibble in the filter should be considered, whereas a mask of '0x0' means it should be ignored.
- This command must be used only if you know the offset value.

Verification of the Packer Tracer Deployment

Use the following command to display Packer Tracer deployment information.

Command	Purpose
show filters	Displays the actual pkt-offset filters set. For more information, see Configuration Example for Packet Tracer , on page 13.

Configuration Example for Packet Tracer

The following examples shows how Packet Tracer functionality is used to capture filters and reports:

```
switch# packet-trace

switch(config-pt)# trigger init rxpp

switch(config-pt-rxpp)# packet-format eth-dot1q-ipv4

switch(config-pt-rxpp-pkt-fmt)#
switch(config-pt-rxpp-pkt-fmt)# set outer ipv4 src-ip 62.0.134.2

switch(config-pt-rxpp-pkt-fmt)# set outer ipv
ipv4    ipv6
switch(config-pt-rxpp-pkt-fmt)# set outer ipv4 next-protocol 17

switch(config-pt-rxpp-pkt-fmt)#
switch(config-pt-rxpp-pkt-fmt)# show filters

slot 1
=====

OUTER
  Ethernet
    eth_type:: value: 0x800, mask: 0xffff, offset: 16
  DOT1Q
    tpid:: value: 0x8100, mask: 0xffff, offset: 12
  IPv4
    protocol:: value: 17, mask: 0xff, offset: 27
    src_ip:: value: 62.0.134.2, mask: 0xffffffff, offset: 30

INNER
```

Troubleshooting Packet Flow Issues

FIELD_NAME	:	VALUE
array[9]	:	0x0
array[8]	:	0x0
array[7]	:	0x0
array[6]	:	0x0
array[5]	:	0x0
array[4]	:	0x2e00

Configuration Example for Packet Tracer

```

array[3]                :                0x260f
  offset_in_bytes        :                0x26
  protocol_type           :                PROTOCOL_TYPE_UDP
array[2]                :                0x1214
  offset_in_bytes        :                0x12
  protocol_type           :                PROTOCOL_TYPE_IPV4_L4
array[1]                :                0xe48
  offset_in_bytes        :                0xe
  protocol_type           :                PROTOCOL_TYPE_VLAN
  flags                   :                2
array[0]                :                0x11
  offset_in_bytes        :                0x0
  protocol_type           :                PROTOCOL_TYPE_ETHERNET_VLAN

```

==== RXPP Termination Input ====

FIELD_NAME	:	VALUE
padding_1	:	0x0
pch_label	:	0x0
unsch_rcy_code	:	0x0
tx_to_rx_rcy_data	:	0xf4
mtu_violation	:	0x0
initial_tc	:	0x0
offset_in_fragment	:	0x0
slice_source_system_port	:	0x98
processing_code	:	0x0
destination	:	0x0
use_cache	:	0x0
single_fragment_packet	:	0x1
flow_signature_on_npuh	:	0x0
phb	:	0x0
reassemble_context	:	0x7ff
learn_enable	:	0x0
receive_time_from_nppd	:	0x0
rxnpu_recycle_count	:	0x0
rxnpu_recycle_data	:	0x0

==== RXPP Termination Macro Stack ====

```

NPE-macros-stack[0]: network_rx_mac_af_and_termination_macro
NPE-macros-stack[1]: network_rx_mac_relay_ipv4_mc_termination_macro

```

==== IRXPP Termination Lookup Keys/Results ====

```

NPE-lookup Keys/Results[0]:
no lookup hit, bucket #1 context network engine termination
no lookup hit, bucket #2 context network engine termination

```

Key Bucket	Key Value	Result Bucket	Result Type	Key Type	Result Value
a	0x5e000066000000138184327c	d	NoneType	npl_service_mapping_ac_port_tag_compound_table_key_t	0
d	0x0	d	0x0	npl_ingress_qos_tag_encoding_pack_table_key_option_tag_type_v4_dscp_t	npl_ingress_qos_tag_encoding_pack_table_payloads_t

```

-----+-----+
NPE-lookup Keys/Results[1]:
no lookup hit, bucket #0 context network engine termination
no lookup hit, bucket #2 context network engine termination
+-----+-----+
| Key Bucket |                               Key Type                               | Key Value |
Result Bucket |                               Result Type                             |
+-----+-----+
|      b      | npl_mac_mc_em_termination_attributes_compound_table_key_t | 0x21b |
|      b      | npl_base_l3_lp_attr_union_t |
|      d      | 0xe00010c32af008674000000000 |
|      d      | npl_mc_macro_compressed_fileds_pack_table_key_t | 0x388 |
|      d      | npl_mc_macro_compressed_fileds_pack_table_ |
payloads_t | 0x0 |
+-----+-----+

==== RXPP Termination Output ====

FIELD_NAME      : VALUE
-----
learn_command   : 0x0
lb_command      : 0x0
offset_in_fragment : 0x0
slice_source_system_port : 0x98
processing_code  : 0x0
destination     : 0xe0086
use_cache       : 0x0
single_fragment_packet : 0x1
flow_signature_on_npuh : 0x0
phb             : 0x0
reassembly_context : 0x7ff
learn_enable     : 0x0
receive_time_from_nppd : 0x0
rxnpu_recycle_count : 0x0
rxnpu_recycle_data : 0x0

==== RXPP Forwarding Macro Stack ====

NPE-macros-stack[0]: network_rx_ipv4_rtf_macro
NPE-macros-stack[1]: network_rx_mac_forwarding_macro
NPE-macros-stack[2]: resolution_macro

==== RXPP Forwarding Lookup Keys/Results ====

NPE-lookup Keys/Results[0]:
no lookup hit, bucket #1 context network engine forwarding
no lookup hit, bucket #3 context network engine forwarding
Error result bucket # 0 , from table ingress_rtf_ipv4_db1_240_f0_compound_table , overlapping
previous value
Error result bucket # 1 , from table ingress_rtf_ipv4_db1_240_f0_compound_table , overlapping
previous value
Error result bucket # 2 , from table ingress_rtf_ipv4_db1_240_f0_compound_table , overlapping
previous value
Error result bucket # 3 , from table ingress_rtf_ipv4_db1_240_f0_compound_table , overlapping
previous value
+-----+-----+
+-----+

```

Configuration Example for Packet Tracer

Key Bucket	Key Type	Key
Value	Result Bucket	Result Type
Result Value		
-----+		
a npl_ingress_rtf_ipv6_db4_480_f0_compound_table_key_t		
0x4ff45003e008602e000006607c19c0101f056	b npl_rtf_payload_t	
0x0		
a npl_ingress_rtf_ipv6_db4_480_f0_compound_table_key_t		
0x4ff45003e008602e000006607c19c0101f056	c npl_rtf_payload_t	
0x0		
a npl_ingress_rtf_ipv6_db4_480_f0_compound_table_key_t		
0x4ff45003e008602e000006607c19c0101f056	d npl_rtf_payload_t	
0x0		
a npl_ingress_rtf_ipv6_db4_480_f0_compound_table_key_t		
0x4ff45003e008602e000006607c19c0101f056	a npl_rtf_payload_t	
0x0		
c npl_ingress_rtf_ipv4_db1_240_f0_compound_table_key_t		0x3821a
0x0	b npl_rtf_payload_t	
c npl_ingress_rtf_ipv4_db1_240_f0_compound_table_key_t		0x3821a
0x0	c npl_rtf_payload_t	
c npl_ingress_rtf_ipv4_db1_240_f0_compound_table_key_t		0x3821a
0x0	d npl_rtf_payload_t	
c npl_ingress_rtf_ipv4_db1_240_f0_compound_table_key_t		0x3821a
0x0	a npl_rtf_payload_t	
0x0		
-----+		

NPE-lookup Keys/Results[1]:

no lookup hit, bucket #0 context network engine forwarding
no lookup hit, bucket #1 context network engine forwarding
no lookup hit, bucket #2 context network engine forwarding

Key Bucket	Key Type	Key Value	Result Bucket
Result Type	Result		
lt Value			
-----+			
d npl_mac_forwarding_table_compound_key_t	0x8601005e00006612		a
npl_mac_forwarding_table_compound_payloads_t			
0x0			
-----+			

NPE-lookup Keys/Results[2]:

no lookup hit, bucket #3 context network engine forwarding

Key Bucket	Key Type	Key Value	Result Bucket
Result Type	Result Value		
-----+			
a npl_v4_l4_resolution_table_compound_key_t			
0x3300103802183	a		
npl_resolution_table_compound_payloads_t	0xa9100e0086		
b npl_v4_l4_resolution_table_compound_key_t			
0x113e008602e00007c107c180000000000000	a		
npl_resolution_table_compound_payloads_t	0xa9100e0086		

```

|      c      | npl_select_fwd_q_m_counter_base_pack_table_key_option_false_value_t |
|      0x4000000      |      c      | np      |
l_select_fwd_q_m_counter_base_pack_table_payloads_t |      0x0      |
+-----+-----+-----+-----+

```

==== RXPP Forwarding Output ====

FIELD_NAME	:	VALUE
offset_in_fragment	:	0x0
slice_source_system_port	:	0x98
processing_code	:	0x0
destination	:	0x0
use_cache	:	0x0
single_fragment_packet	:	0x1
flow_signature_on_npuh	:	0x0
phb	:	0x0
reassembly_context	:	0x7ff
learn_enable	:	0x0
receive_time_from_nppd	:	0x0
rxnpu_recycle_count	:	0x0
rxnpu_recycle_data	:	0x0
padding_2	:	0x0
use_ecn	:	0x0
fllb_control_code	:	0x0
padding_1	:	0x0
ethernet_rate_limiter_type	:	0x7
fwd_offset_cmd	:	0x0

==== RXPP TM PD IFG0 ====

FIELD_NAME	:	VALUE
color	:	0x0
counter_meter_command	:	0xfdfcf5
is_dummy_pd	:	0x1
reorder_data	:	0xb72a
drop	:	0x1
forwarding_destination	:	0x0
mirror_bitmap	:	0x37ff
source_slice_system_port	:	0x0
traffic_class	:	0x0
slice_mode_data	:	0x1fb7ff4fcc9fd1ec2f4000000000000
processing_code	:	0x0
lb_key_msbs_bits	:	0x0
packet_size_bits	:	0x0

==== RXPP TM PD IFG1 ====

FIELD_NAME	:	VALUE
color	:	0x0
counter_meter_command	:	0x43d
is_dummy_pd	:	0x0
reorder_data	:	0xba70
drop	:	0x0
forwarding_destination	:	0x100086
mirror_bitmap	:	0x10
source_slice_system_port	:	0x98
traffic_class	:	0x0

[illegible]

0400000000020000044330f000000000000008600000008000010009e00100860000000000000058

FIELD_NAME	:	VALUE
base_type	:	0x0
version	:	0x0
ive_valid	:	0x1
packet_edit_valid	:	0x0
issu_codespace	:	0x0
receive_time	:	0x0
meter_color	:	0x0
l2_flood_mc_pruning_or_etm	:	0x0
ingress_qos_remark	:	0x20000
fwd_header_type	:	0x0
rx_nw_app_or_lb_key	:	0x443
fwd_offset	:	0x30
slp_qos_id	:	0xf
encap_type	:	0x0
L2_encap	:	0x0
padding	:	0x0
l2_dlp	:	0x0
pif	:	0x0

Additional References

Related Documents	Title/Link
Cisco Nexus 9364E Switches Hardware Installation Guide	Cisco Nexus 9364E-SG2-Q Switch Hardware Installation Guide Cisco Nexus 9364E-SG2-O Switch Hardware Installation Guide
ELAM Overview	https://www.cisco.com/c/en/us/products/cisco-7000-series-switches/168424.html
Nexus 9000 Cloud Scale ASIC (Tahoe) NX-OS ELAM	Nexus 9000 Cloud Scale ASIC (Tahoe) NX-OS ELAM - Cisco

