

Troubleshooting Tools and Methodology

- Command-Line Interface Troubleshooting Commands, on page 1
- ACL Consistency Checker, on page 20
- Proactive Consistency Checker, on page 22
- Configuration Files, on page 24
- CLI Debug, on page 24
- Ping and Traceroute, on page 25
- Monitoring Processes and CPUs, on page 27
- Using Onboard Failure Logging, on page 30
- Using Diagnostics, on page 32
- Using Embedded Event Manager, on page 32
- Using Ethanalyzer, on page 32
- SNMP and RMON Support, on page 48
- Using the PCAP SNMP Parser, on page 49
- Using RADIUS, on page 50
- Using syslog, on page 51
- Using SPAN, on page 52
- SPAN Consistency Checker, on page 52
- Using sFlow, on page 53
- sFlow Consistency Checker, on page 53
- Using the Blue Beacon Feature, on page 54
- Using the watch Command, on page 54
- Additional References for Troubleshooting Tools and Methodology, on page 55

Command-Line Interface Troubleshooting Commands

The command-line interface (CLI) allows you to configure and monitor Cisco NX-OS using a local console or remotely using a Telnet or Secure Shell (SSH) session. The CLI provides a command structure similar to Cisco IOS software, with context-sensitive help, **show** commands, multiuser support, and roles-based access control.

Each feature has **show** commands that provide information about the feature configuration, status, and performance. Additionally, you can use the following command for more information:

- **show system**—Provides information about system-level components, including cores, errors, and exceptions. Use the **show system error-id** command to find details on error codes.

```
switch# copy running-config startup-config
[########################################] 100%
2013 May 16 09:59:29 zoom %$ VDC-1 %$ %BOOTVAR-2-AUTOCOPY_FAILED: Autocopy of file
/bootflash/n9000-dk9.6.1.2.I1.1.bin to standby

switch# show system error-id 0x401e0008
Error Facility:     sysmgr
Error Description:  request was aborted, standby disk may be full
```

Consistency Checker Commands

Cisco NX-OS provides consistency checker commands to validate the software state with the hardware state. The result of the consistency checker is logged as either PASSED or FAILED.

```
2019 May 1 16:31:39 switch vshd: CC_LINK_STATE:
Consistency Check: PASSED
```

Consistency checker is a tool that performs the following functions:

- Checks for system consistency
- Helps perform root cause analysis and fault isolation
- Checks for consistency between software and hardware tables



Note When monitor session is in Down or Error state, Consistency checker is not validated.

Cisco NX-OS supports the following consistency checker commands.

Table 1: Consistency Checker Commands

Command	Description	Supported Platforms
show consistency-checker copp	Verifies CoPP programming.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards
show consistency-checker dme interfaces	Verifies the DME interfaces.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards
show consistency-checker egress-xlate private-vlan	Verifies the private VLAN egress-xlate in the hardware.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards

Command	Description	Supported Platforms
show consistency-checker fex-interfaces {**fex** *fex-id* | **interface ethernet** *fex-id/fex-slot/fex-port*} [**brief** | **detail**]	Compares the software and hardware state of FEX interfaces.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards **Note** The *fex-slot* is always 1.
show consistency-checker fex-interfaces fabric <*fabric-po*>	Verifies FEX fabric PO membership of the physical member interfaces, and interface level hardware programming of the fabric port-channel members.	Cisco Nexus 9300-EX, 9300-FX, 9300-FX2, 9300-GX series switches.
show consistency-checker fex-interfaces fabric <*fabric-po*> **membership vlan** <*vlan-id*>	Verifies FEX fabric PO members are part of the VLAN floodlist, for VLAN which are enabled on FEX interfaces.	Cisco Nexus 9300-EX, 9300-FX, 9300-FX2, 9300-GX series switches.
show consistency-checker fex-interfaces fabric <*fabric-po*> **stp-state vlan** <*vlan-id*>	Verifies FEX fabric PO members are in forwarding / disabled state for VLANs that are enabled on FEX interfaces.	Cisco Nexus 9300-EX, 9300-FX, 9300-FX2, 9300-GX series switches.
show consistency-checker fex-interfaces fabric <*fabric-po*> **egress-xlate private-vlan** <*vlan-id*>	Verifies PVLAN hardware programming corresponding to FEX fabric PO interface, in case there are PVLAN enabled FEX interfaces.	Cisco Nexus 9300-EX, 9300-FX, 9300-FX2, 9300-GX series switches.
test consistency-checker forwarding {**ipv4** | **ipv6**} [**vrf** *vrf-name* | **all**] [**module** *module-number* | **all**]	Starts the Layer 3 route consistency checker test.	All Cisco Nexus 9000 Series switches
show consistency-checker forwarding {**ipv4** | **ipv6**} [**vrf** *vrf-name* | **all**] [**module** *module-number* | **all**]	Displays the Layer 3 route consistency checker test result.	All Cisco Nexus 9000 Series switches
show consistency-checker forwarding single-route {**ipv4** | **ipv6**} *ip-address* **vrf** *vrf-name*} [**brief** | **detail**]	Checks for Layer 3 route consistency for a specific route. Warns when a single-route fails due to ECMP group table exhaustion.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards **Note** Cisco Nexus 34180YC platform switches support only the **ipv4** command.

Command	Description	Supported Platforms
show consistency-checker gwmacdb	Checks for hardware and software consistency in the gateway MAC address database. **Note** This command might show incorrect results for 4-way HSRP.	All Cisco Nexus 9000 Series switches
show consistency-checker kim interface {**ethernet** *slot/port* | **port-channel** *number* | **vlan** *vlan-id*} [**brief** | **detail**]	Verifies the internal connectivity between the supervisor and the line card.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards
show consistency-checker l2 module *module-number*	Verifies that learned MAC addresses are consistent between the software and the hardware. It also shows extra entries that are present in the hardware but not in the software and missing entries in the hardware.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards

Command	Description	Supported Platforms
show consistency-checker l2 multicast group *ip-address* **source** *ip-address* **vlan** *vlan-id* [**brief** | **detail**]	Checks for inconsistencies with Layer 2 multicast groups.	Cisco Nexus 9200, 9300-EX, 9300-FX, and 9300-GX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards Cisco Nexus 9500 series switches with N9K-X9432C-S, N9K-X9536PQ line cards. Cisco Nexus 9500 series switches with N9K-X9432C-FM-S, N9K-C9508-FMX-S, N9K-C9508-FM-S fabric modules. Cisco Nexus N3K-C3232C, N3K-C3264Q, N3K-C31108TC-V, N3K-C3132Q-40GX, N3K-C3132Q-V, N3K-C31108PC-V, N3K-C3172PQ, N3K-C3172TQ, N3K-C3164Q, and N3K-C31128PQ-10GE switches. Cisco Nexus N9K-C9372TX, N9K-C9372TX-E, N9K-C93120TX, N9K-X9432C-S, N9K-C9332PQ, N9K-C9372PX and N9K-C9372PX-E switches.
show consistency-checker l2 switchport interface {**ethernet** *slot/port* | **port-channel** *number* }[**brief** | **detail** | **all**]	Checks for inconsistencies with switchport interfaces.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards

Command	Description	Supported Platforms
show consistency-checker l3-interface interface ethernet *slot/port* [**brief** | **detail**]	Checks for Layer 3 settings of an interface in the hardware and for the following configuration in the hardware: L3 VLAN, CML Flags, IPv4 Enable, VPN ID. This command works for physical interfaces and interfaces that are part of a port channel. It does not validate subinterfaces or FEX interfaces. Beginning Cisco NX-OS Release 9.3(5) this command checks for Layer 3 settings of an SI and SVI interfaces; and the support is extend to Cisco Nexus 9300-GX platform switches.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards Cisco Nexus N9K-C9316D-GX, N9K-C93600CD-GX, N9K-C9364C-GX devices.
show consistency-checker l3-interface module *module-number* [**brief** | **detail**]	Checks for Layer 3 settings of all interfaces in the module and for the following configuration in the hardware: L3 VLAN, CML Flags, IPv4 Enable, VPN ID. This command works for physical interfaces and interfaces that are part of a port channel. It does not validate subinterfaces.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards

Command	Description	Supported Platforms
show consistency-checker l3 multicast group *ip-address* **source** *ip-address* **vrf** *vrf-name* [**brief** | **detail**]	Checks for inconsistencies with Layer 3 multicast groups.	Cisco Nexus 9200, 9300-EX, 9300-FX, and 9300-GX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards Cisco Nexus 9500 Series switches with N9K-X9432C-S, N9K-X9536PQ line cards; and N9K-X9432C-FM-S, N9K-C9508-FMX-S, and N9K-C9508-FM-S fabric modules. Cisco Nexus N3K-C3048TP, N3K-C3064-TC, N3K-C3232C, N3K-C3264Q, N3K-C31108TC-V, N3K-C3132Q-40GX, N3K-C3132Q-V, N3K-C31108PC-V, N3K-C3172PQ, N3K-C3172TQ, N3K-C3164Q, and N3K-C31128PQ-10GE switches. Cisco Nexus N9K-C9372TX, N9K-C9372TX-E, N9K-C93120TX, N9K-X9432C-S, N9K-C9332PQ, N9K-C9372PX and N9K-C9372PX-E switches.
show consistency-checker link-state fabric-ieth [**module** *module-number*] [**brief** | **detail**]	Verifies the programming consistency between software and hardware for the link-state status of internal fabric ports.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards
show consistency-checker link-state interface ethernet *slot/port* [**brief** | **detail**]	Verifies the programming consistency between software and hardware for the link-state status of the interfaces. This command works for physical Ethernet interfaces and physical Ethernet interfaces that are part of a port channel. It does not validate subinterfaces or FEX interfaces.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards

Command	Description	Supported Platforms
show consistency-checker link-state module *module-number* [**brief** | **detail**]	Verifies the software link state of all the interfaces in the module against its hardware link state. This command works for physical Ethernet interfaces and physical Ethernet interfaces that are part of a port channel. It does not validate subinterfaces or FEX interfaces.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards
show consistency-checker membership port-channels [**interface port-channel** *channel-number*] [**brief** | **detail**]	Checks for port-channel membership in the hardware in all modules and validates it with the software state. This command runs per port channel.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards
show consistency-checker membership port-channels [**brief** | **detail**]	Checks for port-channel membership in the hardware in all modules and validates it with the software state. This command runs for all port channels in the system.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards
show consistency-checker membership vlan *vlan-id* {**native-vlan** | **private-vlan interface** {**ethernet** *slot/port* | **port-channel** *number* | **native-vlan**}} [**brief** | **detail** | **interface**]	Determines that the VLAN membership in the software is the same as programmed in the hardware. It also ignores the interfaces that are in the STP BLK state.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards **Note** The **private-vlan** command does not support the **brief** or **detail** option. **Note** Cisco Nexus 34180YC platform switches support only the **native-vlan** command.
show consistency-checker pacl {**module** *module-number* | **port-channels interface port-channel** *channel-number*}	Validates the IPv4, IPv6, and MAC PACL programming consistency between the hardware and software and verifies if <label, entry-location> pairs are consistent between the hardware and software.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards
show consistency-checker pacl extended ingress {**ip** | **ipv6** | **mac**} **interface** {**ethernet** *slot/port* | **port-channel** *number*} [**brief** | **detail**]	Verifies PACL programming for ingress interfaces (including FEX interfaces) and port channels.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards

Command	Description	Supported Platforms
show consistency-checker pacl extended ingress {**ip** | **ipv6** | **mac**} **module** *module-number* [**brief** | **detail**]	Verifies PACL programming across all physical interfaces, subinterfaces, breakout ports, and FEX interfaces for the specified module.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards
show consistency-checker port-state fabric-ieth [**module** *module-number* [**ieth-port** *ieth-port*]] [**brief** | **detail**]	Verifies the state of internal fabric ports.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards
show consistency-checker port-state [**module** *module-number*] [**brief** | **detail**]	Verifies the port state for the specified module.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards
show consistency-checker racl {**module** *module-number* | **port-channels interface port-channel** *channel-number* | **svi interface vlan** *vlan-id*}	Validates the IPv4 and IPv6 RACL programming consistency between the hardware and software and verifies if <label, entry-location> pairs are consistent between the hardware and software. • When invoked per module, this command verifies IPv4 and IPv6 ACL consistency for all of the physical interfaces and subinterfaces for that module. • When invoked on a specific port channel, this command verifies for all the member ports. • When invoked on all port channels, this command verifies for each port channel that has an ACL applied. **Note** This command does not verify IPv4 and IPv6 ACLs and does not verify if qualifiers and actions are matching.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards
show consistency-checker racl extended ingress {**ip** | **ipv6**} **interface** {**ethernet** *slot/port* | **port-channel** *number* | **vlan** *vlan-id*} [**brief** | **detail**]	Verifies RACL programming for ingress interfaces, subinterfaces, breakout ports, port channels, or SVIs.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards

Command	Description	Supported Platforms
show consistency-checker racl extended ingress {**ip** | **ipv6**} **module** *module-number* [**brief** | **detail**]	Verifies RACL programming for ingress interfaces on the specified module. This command runs across all of the physical interfaces, subinterfaces, and breakout ports for that module.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards
show consistency-checker stp-state vlan *vlan-id* [**brief** | **detail** | **interface**]	Determines whether the spanning tree state in the software is the same as programmed in the hardware. This command is run only on interfaces that are operational (up).	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards.
show consistency-checker vacl extended ingress {**ip** | **ipv6** | **mac**} **vlan** *vlan-id* [**brief** | **detail**]	Verifies VACL programming on all of the member interfaces of the VLAN.	Cisco Nexus 34180YC, 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards
show consistency-checker vpc [**source-interface**] [**brief** | **detail**]	Checks for vPC inconsistencies.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX and -FX line cards Cisco Nexus 9500 Series switches with N9K-X9432C-S, N9K-X9536PQ line cards; and N9K-X9432C-FM-S, N9K-C9508-FMX-S, and N9K-C9508-FM-S fabric modules. Cisco Nexus N3K-C3048TP, N3K-C3064-TC, N3K-C3232C, N3K-C3264Q, N3K-C31108TC-V, N3K-C3132Q-40GX, N3K-C3132Q-V, N3K-C31108PC-V, N3K-C3172PQ, N3K-C3172TQ, N3K-C3164Q, and N3K-C31128PQ-10GE switches. Cisco Nexus N9K-C9372TX, N9K-C9372TX-E, N9K-C93120TX, N9K-X9432C-S, N9K-C9332PQ, N9K-C9372PX and N9K-C9372PX-E switches.

Command	Description	Supported Platforms
show consistency-checker vxlan config-check [**verbose-mode**]	Verifies the VXLAN EVPN configuration on the switch.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches Cisco Nexus C31108PC-V, C31108TC-V, C3132Q-V and 3132C-Z switches. Cisco Nexus C9396TX, C93128TX, C9396PX, X9564PX, X9564TX and X9536PQ switches. Cisco Nexus C3132Q-40GE-SUP, C3132Q-40GX-SUP, C3132Q-XL, C31128PQ-10GE, C3264Q-S, C3264C-E switches.
show consistency-checker vxlan infra [**verbose-mode**]	Checks for inconsistencies with the VXLAN tunnel infrastructure.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches Cisco Nexus C31108PC-V, C31108TC-V, C3132Q-V and 3132C-Z switches. Cisco Nexus C9396TX, C93128TX, C9396PX, X9564PX, X9564TX and X9536PQ switches. Cisco Nexus C3132Q-40GE-SUP, C3132Q-40GX-SUP, C3132Q-XL, C31128PQ-10GE, C3264Q-S, C3264C-E switches.
show consistency-checker vxlan l2 module *module-number*	Verifies the consistency with VXLAN Layer 2 routes.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches Cisco Nexus C31108PC-V, C31108TC-V, C3132Q-V and 3132C-Z switches. Cisco Nexus C9396TX, C93128TX, C9396PX, X9564PX, X9564TX and X9536PQ switches. Cisco Nexus C3132Q-40GE-SUP, C3132Q-40GX-SUP, C3132Q-XL, C31128PQ-10GE, C3264Q-S, C3264C-E switches.

Command	Description	Supported Platforms
show consistency-checker vxlan l3 vrf [*vrf-name* | **all**] [**start-scan** | **report**]	Checks for inconsistencies with VXLAN Layer 3 routes.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches Cisco Nexus C31108PC-V, C31108TC-V, C3132Q-V and 3132C-Z switches. Cisco Nexus C9396TX, C93128TX, C9396PX, X9564PX, X9564TX and X9536PQ switches.
show consistency-checker vxlan pv	Verifies if VLAN mappings are programmed consistently between the software and across different tables in the hardware. At least one interface needs to be enabled with port VLAN mappings in order to run this command.	Cisco Nexus 9200, 9300-EX, and 9300-FX/FX2, and 9500 platform switches
show consistency-checker vxlan qinq-qinvni	Checks for a multi-tag VLAN list and associated multi-tag vn-segment being consistent in the software and hardware.	Cisco Nexus 9300-FX/FX2 platform switches
show consistency-checker vxlan selective-qinvni interface {**ethernet** *slot/port* | **port-channel** *channel-number*}	Validates if port-specific selective Q-in-VNI mappings are programmed correctly in the software and hardware in order for the inner tags in the packets to be preserved.	Cisco Nexus 9300-EX and 9300-FX/FX2 platform switches
show consistency-checker vxlan vlan [**all** | *vlan-id*] [**verbose-mode**]	Checks for inconsistencies with VXLAN VLANs.	Cisco Nexus 9300-EX and 9300-FX/FX2 platform switches Cisco Nexus C31108PC-V, C31108TC-V, C3132Q-V and 3132C-Z switches. Cisco Nexus C9396TX, C93128TX, C9396PX, X9564PX, X9564TX and X9536PQ switches. Cisco Nexus C3132Q-40GE-SUP, C3132Q-40GX-SUP, C3132Q-XL, C31128PQ-10GE, C3264Q-S, C3264C-E switches.

Command	Description	Supported Platforms
show consistency-checker vxlan xconnect	Checks for inconsistencies with VXLAN Xconnect VLANs. Validates that Xconnect ACLs are installed on all units and slices and MAC learn is disabled on all Xconnect VLANs.	Cisco Nexus 9200, 9332C, 9364C, 9300-EX, and 9300-FX/FX2 platform switches
show consistency-checker vxlan l3 single-route [**ipv4** | **ipv6**] [*vrf*]	Checks for inconsistencies with VXLAN layer 3 single route traffic.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches. Cisco Nexus C31108PC-V, C31108TC-V, C3132Q-V and 3132C-Z switches. Cisco Nexus C9396TX, C93128TX, C9396PX, X9564PX, X9564TX and X9536PQ switches and Cisco Nexus 9200, 9300-EX and 9300-FX platform switches.
show consistency-checker vxlan l2 [**mac-address**] [*mac-address*] | **module**] [*module*	Checks for inconsistencies with VXLAN layer 2.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches. Cisco Nexus C31108PC-V, C31108TC-V, C3132Q-V and 3132C-Z switches. Cisco Nexus C9396TX, C93128TX, C9396PX, X9564PX, X9564TX and X9536PQ switches and Cisco Nexus 9200, 9300-EX and 9300-FX platform switches. Cisco Nexus C3132Q-40GE-SUP, C3132Q-40GX-SUP, C3132Q-XL, C31128PQ-10GE, C3264Q-S, C3264C-E switches.

Command	Description	Supported Platforms
show consistency-checker storm-control [**brief** | **detail**]	Checks for inconsistencies with storm control	Beginning with Cisco NX-OS Release 10.5(1), the breif and detail of the storm control consistency. Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, -FX, and -R line cards Beginning with Cisco NX-OS Release 9.3(5), it is supported on N3K-C3016Q-40GE,N3K-C3048TP-1GE, N3K-C3064PQ-10GE, N3K-C3064PQ-10GX, N3K-C3064T-10GT, N9K-C9504-FM, N9K-C9508-FM, N9K-C9516-FM, N9K-C9508-FM-S, N3K-C31128PQ, N3K-C3164Q-40GE, N3K-C3232C, N3K-C3132Q-V, N3K-C31108PC-V, N3K-C31108TC-V, N3K-C3264C-E, N3K-C3132C-Z, N9K-C93128TX, N9K-C9396PX, N9K-C9372PX and N9K-C9332PQ devices. **Note** When ND ISSU is done to Cisco NX-OS Release 10.2(6)M, and when pol_rate or pol_burst value in hardware and software do not match, the storm control consistency checker fails. To resolve the issue, reconfigure storm control.
show consistency-checker segment-routing mpls [**ip**] [*ip-address*] | **mask**] [*mask* | **vrf**] [*vrf*	Checks route consistency for Underlay Segment Routing (ISIS, BGP, OSPF) and Overlay routes Layer 3 VPN and Layer 2 EVPN.	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, and -FX line cards. Cisco Nexus N9K-C9316D-GX, N9K-C93600CD-GX, N9K-C9364C-GX devices.

Command	Description	Supported Platforms
show consistency-checker segment-routing mpls label	Checks label consistency for Underlay Segment Routing (ISIS, BGP, OSPF) and Overlay routes Layer 3 VPN, Layer 2 EVPN, and ADJ SIDS	Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and Cisco Nexus 9500 platform switches with -EX, and -FX line cards. Cisco Nexus N9K-C9316D-GX, N9K-C93600CD-GX, N9K-C9364C-GX devices.
show consistency-checker sflow [**brief** | **detail**]	Checks the program and consistency configurations for supervisor and line cards hardware tables.	Cisco Nexus 9300-FX2, 9300-FX3, 9300-GX, and 9300-GX2 platform switches

The following commands do not support JSON output:

- **show consistency-checker forwarding** {**ipv4** | **ipv6**} [**vrf** *vrf-name* | **all**] [**module** *module-number* | **all**]
- **show consistency-checker pacl** {**module** *module-number* | **port-channels interface port-channel** *channel-number*}
- **show consistency-checker racl module** *module-number*
- **show consistency-checker racl port-channels interface port-channel** *channel-number*}
- **show consistency-checker racl svi interface vlan** *vlan-id*
- **show consistency-checker vxlan**
- **test consistency-checker forwarding** {**ipv4** | **ipv6**} [**vrf** *vrf-name* | **all**] [**module** *module-number* | **all**]

The **show consistency-checker vxlan** commands are not modeled.

Multicast Consistency Checker

The multicast consistency checker is a single-route consistency checker for Layer 2 and Layer 3 routes for verifying the state of multicast routes. The multicast consistency checker executes the show commands in each component, parses the relevant information, and then compares the processed information against the other components to check for inconsistencies. The multicast consistency checker commands terminate upon encountering a failure. The **show consistency-checker l2 multicast group** and **show consistency-checker l3 multicast group** commands return the differences in the expected value and the actual value.

The commands support the following output formats:

- **verbose**: Displays the results in text format.
- **detail**: Displays the results in JSON format.
- **brief**: Displays the results in JSON format with minimal details.

Beginning with Cisco NX-OS Release 10.2(2)F, L3 Multicast Consistency Checker supports NAT translation and is supported on all platforms. UMNAT is not supported.

Note MMNAT stands for Multicast to Multicast NAT, MUNAT stands for Multicast to Unicast NAT, and UMNAT stands for Unicast to Multicast NAT. NAT translation must be of the type MMNAT ingress and egress, and MUNAT.

Beginning with Cisco NX-OS Release 10.2(1)F, Multicast over GRE consistency checker is introduced on N9K-C9316D-GX, N9K-C93600CD-GX, N9K-C9364C-GX family switches. Multicast over GRE (mGRE) Consistency Checker supports the following:

- Single route mGRE Consistency Checker
- mGRE tunnels on L3 Ethernet Interfaces, L3 Port-channels and L3 sub-interfaces
- GRE tunnel where transport protocol VRF can be different from tunnel interface VRF. This is supported only for GREv4 - GRE tunnels over IPv4 multicast.

Multicast over GRE (mGRE) Consistency Checker does not support the following:

- FEX
- GRE tunnels over IPv6
- mGRE is not supported on EoRs. Consistency check is supported only on N9K-C9316D-GX, N9KC93600CD-GX, N9K-C9364C-GX ToRs.
- mGRE is not supported on SVIs.

The mGRE consistency checks happen only if there is a IP GRE Tunnel interface in the outgoing interface list or if the RPF interface is an IP GRE tunnel interface.

Beginning from Cisco NX-OS Release 10.1(1), the following consistency checkers are supported:

- IPv6 L2 Multicast Consistency Checker
- IPv6 L3 Multicast Consistency Checker
- Multicast NLB Consistency Checker
  - Multicast MAC Lookup mode Consistency Checker
  - Multicast NLB L3 unicast configuration Consistency Checker
- Multicast GRE Consistency Checker

The following existing CLI command is extended to accept IPv6 source and group addresses for IPv6 L2 Multicast Consistency Checker:

show consistency-checker l2 multicast group <ipv4/ipv6 group address> source <ipv4/v6 source address> vrf <vrf-id> [brief|detail]

The following is the output example for IPv6 L2 Multicast Consistency Checker:

```
# show consistency-checker l2 multicast group ?
  A.B.C.D   Group IP address
  A:B::C:D  Group IPv6 address
```

The following existing CLI command is extended to accept IPv6 source and group addresses for IPv6 L3 Multicast Consistency Checker:

show consistency-checker l3 multicast group <ipv4/ipv6 group address> source <ipv4/v6 source address> vlan <vlan-id> [brief|detail]

The following is the output example for IPv6 L3 Multicast Consistency Checker:

```
# show consistency-checker l3 multicast group ?
  A.B.C.D   Group IP address
  A:B::C:D  Group IPv6 address
```

The following new CLI command is added to support Multicast MAC Lookup mode Consistency Checker:

show consistency-checker l2 multicast mac <mac> vlan <vlan-id>

The following is the output example for Multicast MAC Lookup mode Consistency Checker:

```
# show consistency-checker l2 multicast mac 0100.1234.1234 vlan 10 ?
  >       Redirect it to a file
  >>      Redirect it to a file in append mode
  brief   Show consistency checker structured output in brief
  detail  Show consistency checker structured output in detail
  |       Pipe command output to filter
```



Note This CLI is used for MAC lookup mode Consistency Checker or L2 mode consistency checker for NLB. The input MAC can be ip-mac or non-ip-mac.

The following new CLI command is added to support Multicast NLB L3 unicast configuration Consistency Checker:

show consistency-checker multicast nlb cluster-ip <unicast-cluster-ip> vrf <vrf-id>

The following is the output example for Multicast NLB L3 unicast configuration Consistency Checker:

```
# show consistency-checker multicast nlb cluster-ip <unicast-cluster-ip>
  >       Redirect it to a file
  >>      Redirect it to a file in append mode
  brief   Show consistency checker structured output in brief
  detail  Show consistency checker structured output in detail
  |       Pipe command output to filter
```

The following existing CLI command is used for Multicast GRE Consistency Checker:

show consistency-checker l3 multicast group <ipv4 group address> source <ipv4 source address> vrf <vrf-id> [brief|detail]



Note Existing IPv4 L3 multicast consistency checker CLI will be used to start Multicast GRE Consistency Checker.

The multicast consistency checker supports the following devices:

- Cisco Nexus 92304QC, 9272Q, 9236C, 92300YC, 93108TC-EX, 93180LC-EX, 93180YC-EX, and 9300-GX platform switches and N9K-X9736C-EX, N9K-X97160YC-EX, N9K-X9732C-EX, and N9K-X9732C-EXM line cards.
- Cisco Nexus 9500 Series switches with N9K-X96136YC-R, N9K-X9636C-R, and N9K-X9636Q-R line cards.

Beginning with Cisco NX-OS Release 9.3(5), multicast consistency checker supports the following devices:

- Cisco Nexus 9500 Series switches with N9K-X9432C-S, N9K-X9536PQ line cards; and N9K-X9432C-FM-S, N9K-C9508-FMX-S, and N9K-C9508-FM-S fabric modules.
- Cisco Nexus N3K-C3232C, N3K-C3264Q, N3K-C31108TC-V, N3K-C3132Q-40GX, N3K-C3132Q-V, N3K-C31108PC-V, N3K-C3172PQ, N3K-C3172TQ, N3K-C3164Q, and N3K-C31128PQ-10GE switches.
- Cisco Nexus N9K-C9372TX, N9K-C9372TX-E, N9K-C93120TX, N9K-X9432C-S, N9K-C9332PQ, N9K-C9372PX, and N9K-C9372PX-E switches.

Beginning with Cisco NX-OS Release 10.1(1), multicast consistency checker supports the following devices:

- Cisco Nexus N9k-C9504 with N9K-X97160YC-EX, N9k-C9504 with N9K-X9732C-EX, N9k-C9504 with N9K-X9732C-FX, N9k-C9504 with N9K-X9736C-EX, N9k-C9504 with N9K-X9736C-FX, N9k-C9504 with N9K-X9736Q-FX, and N9k-C9504 with N9K-X9788TC-FX.
- Cisco Nexus N9k-C9508 with N9K-X97160YC-EX, N9k-C9508 with N9K-X9732C-EX, N9k-C9508 with N9K-X9732C-FX, N9k-C9508 with N9K-X9736C-EX, N9k-C9508 with N9K-X9736C-FX, N9k-C9508 with N9K-X9736Q-FX, and N9k-C9508 with N9K-X9788TC-FX.

The Multicast Consistency Checker verifies the programming consistency of the following Layer 2 components:

- IGMP snooping
- MFDM
- MFIBPI
- MFIBPD
- Hardware tables

The Multicast Consistency Checker verifies the programming consistency of the following Layer 3 components:

- PIM
- MRIB
- IGMP snooping
- MFDM
- MFIBPI
- MFIBPD
- Hardware tables

Output Examples for Multicast Consistency Checker Commands

The following is an example of IGMP snooping output:

```
switch# show ip igmp snooping groups 225.12.12.28 225.12.12.28 vlan 222
Type: S - Static, D - Dynamic, R - Router port, F - Fabricpath core port
Vlan  Group Address      Ver  Type  Port list
222   225.12.12.28       v3   D     Eth1/2 Eth1/3 Po12 Po100 Po18
```

The following is an example of MFDM output:

```
switch# show forwarding distribution l2 multicast vlan 222 group 225.12.12.28 source
225.12.12.28
Vlan: 222, Group: 225.12.12.28, Source: 225.12.12.28
  Outgoing Interface List Index: 4
  Reference Count: 204
  Num L3 usages: 4
  Platform Index: 0xa00004
  Vpc peer link exclude flag set
  Number of Outgoing Interfaces: 5
    Ethernet1/2
    Ethernet1/3
    port-channel12
    port-channel18
    port-channel100
```

The following is an example of comparing IGMP Snooping with MFDM (passed):

```
**********************************************************************
Comparing IGMP Snooping with MFDM
**********************************************************************
L2 Eth Receivers :
IGMP Snooping: 1/2, 1/3
MFDM: 1/2, 1/3

L2 PC Receivers :
IGMP Snooping: 100, 12, 18
MFDM: 12, 100, 18

CC between IGMP Snooping and MFDM PASSED
```

The following is an example of comparing IGMP Snooping with MFDM (failed):

```
**********************************************************************
Comparing IGMP Snooping with MFDM
**********************************************************************
L2 Eth Receivers:
IGMP Snooping: 1/2, 1/3
MFDM: 1/2,  1/3

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
L2 PC Receivers:
IGMP Snooping: 100, 12, 18
MFDM: 12, 100, 16
Consistency check failed!!!
Missing elements are: 18
Additional elements are: 16
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Congestion Detection and Avoidance

Beginning Cisco NX-OS Release 9.3(3), Cisco Nexus 9000 series switches supports **show tech-support slowdrain** command to troubleshoot congestion issues. The **show tech-support slowdrain** command contains some of the congestion detection indications, counters, and log messages as well as other commands that allow an understanding of the switches, Cisco NX-OS versions, and topology.

Since, congestion can propagate from one switch to another, you must gather the **show tech-support slowdrain** command output from all the switches at the same time for a better assessment of the congestion triggers and propogation.

ACL Consistency Checker

Beginning with Cisco NX-OS Release 9.3(3), the ACL consistency checker supports the following devices:

N9K-C9372PX, N9K-C9372PX-E, N9K-C9372TX, N9K-C9372TX-E, N9K-C9332PQ, N9K-C93128TX, N9K-C9396PX, N9K-C9396TX, N9K-C9508-FM-S, N9K-C9508-FM2, N9K-C9504-FM-S, N9K-X9632PC-QSFP100, N9K-X9432C-S

Beginning with Cisco NX-OS Release 9.3(5), the ACL consistency checker is supported on Cisco Nexus N9K-C9316D-GX, N9K-C93600CD-GX, N9K-C9364C-GX, N9K-C93240YC-FX2, N9K-C93180YC-EX, N3K-C3636C-R, N3K-C36180YC-R; and Cisco Nexus 9500 Series switches with N9K-X9636Q-R, N9K-X9636C-R, N9K-X9636C-RX and N9K-X96136YC-R line cards.

The following entities are verified as part of the ACL consistency check:

Action, Protocol, SIP, DIP, source port, destination port, Source MAC, Destination MAC, Ethertype, COS, DSCP, VLAN and UDFs.

Cisco NX-OS supports the following PACL, RACL, and VACL consistency checker commands.

Command	Description
show consistency-checker pacl extended ingress ip module <module-id> [brief | detail]	Verifies PACL consistency check for ingress interfaces and port channel for the specified IP module.
show consistency-checker pacl extended ingress ipv6 module <module-id> [brief | detail]	Verifies PACL consistency check for ingress interfaces and port channel for the specified IPv6 module.
show consistency-checker pacl extended ingress mac module <module-id> [brief | detail]	Verifies MAC PACL consistency check for ingress interfaces and port channel for the specified MAC module.
show consistency-checker pacl extended ingress ip interface {<int-id> | <ch-id> [brief | detail]	Verifies PACL consistency check for the specified ingress interface.
show consistency-checker pacl extended ingress ipv6 interface {<int-id> | <ch-id> [brief | detail]	Verifies PACL consistency check for the specified IPv6 ingress interface.
show consistency-checker pacl extended ingress mac interface {<int-id> | <ch-id> [brief | detail]	Verifies PACL consistency check for the specified ingress MAC interface.
show consistency-checker racl extended ingress ip module <module-id> [brief | detail]	Verifies RACL consistency check for ingress interfaces and port channel for the specified IP module.
show consistency-checker racl extended ingress ipv6 module <module-id> [brief | detail]	Verifies RACL consistency check for ingress interfaces and port channel for the specified IPv6 module.

Command	Description
show consistency-checker racl extended ingress ip interface {<int-id> | <ch-id> | <vlan-id>} [brief | detail]	Verifies RACL consistency check for the specified ingress interface.
show consistency-checker racl extended ingress ipv6 interface {<int-id> | <ch-id> | <vlan-id>} [brief | detail]	Verifies RACL consistency check for the specified ingress IPv6 interface.
show consistency-checker vacl extended ingress ip vlan <vlan-id> [brief | detail]	Verifies VACL consistency check for the specified IP VLAN.
show consistency-checker vacl extended ingress ipv6 vlan <vlan-id> [brief | detail]	Verifies VACL consistency check for the specified IPv6 VLAN.
show consistency-checker vacl extended ingress mac vlan <vlan-id> [brief | detail]	Verifies VACL consistency check for the specified ingress MAC VLAN.

Output Examples for ACL Consistency Checker Commands

This example shows the RACL consistency check results:

```
switch# show consistency-checker racl extended ingress ip module 1 Consistency checker
passed for Eth1/3 (ingress, ip, ip-list)
switch#
switch#
switch# show consistency-checker racl extended ingress ip module 1 brief
{
   "result": {
   "status": "CC_STATUS_OK",
   "checkers": [
     {
       "version": 1,
       "type": "CC_TYPE_IF_RACL",
       "status": "CC_STATUS_OK",
        "platformDetails": {
        "classType": "CC_PLTFM_NXOS_BCM"
      },
      "recoveryActions": [],
      "failedEntities": []
      }
    ]
   }
}
switch#
switch # show consistency-checker racl extended ingress ip interface ethernet 3/5
Consistency checker passed for Ethernet3/5 (ingress, ip, ip-list)
switch#
switch# show consistency-checker racl extended ingress ip interface ethernet 3/5 brief
{
   "result": {
   "status": "CC_STATUS_OK",
    "checkers": [
      {
       "version": 1,
       "type": "CC_TYPE_IF_RACL",
       "status": "CC_STATUS_OK",
       "platformDetails": {
       "classType": "CC_PLTFM_NXOS_BCM"
```

```
      },
       "recoveryActions": [],
       "failedEntities": []
     }
    ]
  }
}
```

Proactive Consistency Checker

Consistency check between software and hardware tables on Nexus platform is a high priority serviceability challenge with respect to route consistency checker. The existing route consistency checker is not a proactive mechanism and is an on-demand consistency checker when a command is issued.

The Proactive Consistency checker has a route/adjacency consistency checker that runs in the background continuously that enables to pro-actively detect any inconsistency for IPv4 or IPv6 routes and ARP or ND adjacencies.

The Proactive Consistency Checker is supported on all Cloudscale EOR and TOR platforms. It has two types of consistency checking methods.

- **The Full Database Consistency Checker**: This performs the consistency check on complete route and adjacency database.
- **The Incremental Consistency Checker**: This consistency check runs on the incremental change set of routes and adjacencies which got updated or added over a period of time.

Show commands

Whenever any inconsistency is found by the proactive consistency checker, the following syslog will be generated:

"%UFDM-3-PROACTIVE_CC_INCONSISTENCY_FOUND: Inconsistencies found in Proactive CC session"

The following two commands must be used to check the inconsistencies during proactive consistency check:

Commands	Description
show forwarding proactive-cc inconsistencies	This show command displays the inconsistencies found in the last failed iteration.
show forwarding proactive-cc inconsistencies all	This show command displays all the inconsistencies found from the time when the proactive consistency check is configured

If the user intends to clear the inconsistencies seen in the above two commands, the following command can be used:

"clear forwarding proactive-cc inconsistencies"

Configuration Commands

The following are the commands to enable/disable the feature and to change the periodicity (timer) for incremental and full consistency check:

- **platform proactive-cc forwarding** (enables with default timers)
- **no platform proactive-cc forwarding** (to disable)
- **platform proactive-cc forwarding fulldb <time in sec>**
- **platform proactive-cc forwarding incremental <time in sec>**
- **platform proactive-cc forwarding incremental <time in sec> fulldb <time in sec>**

Command	Purpose
platform proactive-cc forwarding **Example:** `switch(config)# platform proactive-cc forwarding`	This command enables the proactive consistency checker in the switch and default timers will be set. FulldB default timer value is 86400. Incremental dB default timer value is 10 seconds.
no platform proactive-cc forwarding **Example:** `switch(config)# no platform proactive-cc forwarding`	This command disables the proactive consistency checker.
platform proactive-cc forwarding fulldb <time in sec> **Example:** `switch(config)# platform proactive-cc forwarding fulldb 600`	This command will configure proactive consistency checker fulldB timer to 600 seconds.
platform proactive-cc forwarding incremental <time in sec> **Example:** `switch(config)# platform proactive-cc forwarding incremental 20`	This command will configure proactive cc incremental timer value to 20 seconds.
platform proactive-cc forwarding incremental <time in sec> fulldb <time in sec> **Example:** `switch(config)# platform proactive-cc forwarding incremental 20 fulldb 600`	This command will configure both incremental timer and fulldB timer together.

Configuration Files

Configuration files contain the Cisco NX-OS commands used to configure the features on a Cisco NX-OS device. Cisco NX-OS has two types of configuration files: running configuration and startup configuration. The device uses the startup configuration (startup-config) during the device startup to configure the software features. The running configuration (running-config) contains the current changes that you make to the startup-configuration file. You should create a backup version of your configuration files before modifying that configuration. You can back up the configuration files to a remote server. See the configuration file information in the *Cisco Nexus 9000 Series NX-OS Fundamentals Configuration Guide*. You can also create a checkpoint copy of the configuration file that you can roll back to if problems occur. See the rollback feature in the *Cisco Nexus 9000 Series NX-OS System Management Configuration Guide*.

Cisco NX-OS features can create internal locks on the startup configuration file. In rare cases, these locks might not be removed by the features. Use the **system startup-config unlock** command to remove these locks.

CLI Debug

Cisco NX-OS supports an extensive debugging feature set for actively troubleshooting a network. Using the CLI, you can enable debugging modes for each feature and view a real-time updated activity log of the control protocol exchanges. Each log entry has a time stamp and is listed chronologically. You can limit access to the debug feature through the CLI roles mechanism to partition access on a per-role basis. While the **debug** commands show real-time information, you can use the **show** commands to list historical and real-time information.



Caution Use the **debug** commands only under the guidance of your Cisco technical support representative because some **debug** commands can impact your network performance.



Note You can log debug messages to a special log file, which is more secure and easier to process than sending the debug output to the console.

By using the **?** option, you can see the options that are available for any feature. A log entry is created for each entered command in addition to the actual debug output. The debug output shows a time-stamped account of the activity that occurred between the local device and other adjacent devices.

You can use the debug facility to track events, internal messages, and protocol errors. However, you should be careful when using the debug utility in a production environment because some options might prevent access to the device by generating too many messages to the console or creating CPU-intensive events that could seriously affect network performance.

Note We recommend that you open a second Telnet or SSH session before you enter any **debug** commands. If the debug session overwhelms the current output window, you can use the second session to enter the **undebug all** command to stop the debug message output.

Debug Filters

You can filter out unwanted debug information by using the **debug-filter** command. The **debug-filter** command allows you to limit the debug information produced by related **debug** commands.

The following example limits EIGRP hello packet debug information to Ethernet interface 2/1:

```
switch# debug-filter ip eigrp interface ethernet 2/1
switch# debug eigrp packets hello
```

Ping and Traceroute

Note Use the ping and traceroute features to troubleshoot problems with connectivity and path choices. Do not use these features to identify or resolve network performance issues.

The **ping** and **traceroute** commands are two of the most useful tools for troubleshooting TCP/IP networking problems. The ping utility generates a series of echo packets to a destination across a TCP/IP internetwork. When the echo packets arrive at the destination, they are rerouted and sent back to the source.

The traceroute utility operates in a similar fashion but can also determine the specific path that a frame takes to its destination on a hop-by-hop basis.

Using Ping

Use the **ping** command to verify connectivity and latency to a particular destination across an IPv4 routed network.

Use the **ping6** command to verify connectivity and latency to a particular destination across an IPv6 routed network.

The ping utility allows you to send a short message to a port or end device. By specifying the IPv4 or IPv6 address, you can send a series of frames to a target destination. Once these frames reach the target, they are looped back to the source and a time stamp is taken.

Note We do not recommend using the Ping utility to test network performance with the IP address configured on the Nexus switch. ICMP (Ping) traffic directed to the switch IP address is subject to CoPP (Control Plane Policing) and may be dropped.

```
switch# ping 172.28.230.1 vrf management
PING 172.28.230.1 (172.28.230.1): 56 data bytes
64 bytes from 172.28.230.1: icmp_seq=0 ttl=254 time=1.095 ms
64 bytes from 172.28.230.1: icmp_seq=1 ttl=254 time=1.083 ms
64 bytes from 172.28.230.1: icmp_seq=2 ttl=254 time=1.101 ms
64 bytes from 172.28.230.1: icmp_seq=3 ttl=254 time=1.093 ms
64 bytes from 172.28.230.1: icmp_seq=4 ttl=254 time=1.237 ms

--- 172.28.230.1 ping statistics ---
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min/avg/max = 1.083/1.121/1.237 ms
```

Using Traceroute

Use traceroute to do the following:

- Trace the route followed by the data traffic.
- Compute the interswitch (hop-to-hop) latency.

The traceroute utility identifies the path taken on a hop-by-hop basis and includes a time stamp at each hop in both directions. You can use traceroute to test the connectivity of ports along the path between the generating device and the device closest to the destination.

Use the **traceroute** {*dest-ipv4-addr* | *hostname*} [**vrf** *vrf-name*] command for IPv4 networks and the **traceroute6** {*dest-ipv6-addr* | *hostname*} [**vrf** *vrf-name*] command for IPv6 networks. If the destination cannot be reached, the path discovery traces the path up to the point of failure.

```
switch# traceroute 172.28.254.254 vrf management
traceroute to 172.28.254.254 (172.28.254.254), 30 hops max, 40 byte packets
 1  172.28.230.1 (172.28.230.1)  0.941 ms  0.676 ms  0.585 ms
 2  172.24.114.213 (172.24.114.213)  0.733 ms  0.7 ms  0.69 ms
 3  172.20.147.46 (172.20.147.46)  0.671 ms  0.619 ms  0.615 ms
 4  172.28.254.254 (172.28.254.254)  0.613 ms  0.628 ms  0.61 ms
```

Press **Ctrl-C** to terminate a running traceroute.

You can use the following commands to specify a source interface for the traceroute:

Command	Purpose
traceroute {*dest-ipv4-addr* | *hostname*} [**source** {*dest-ipv4-addr* | *hostname* | *interface*}] [**vrf** *vrf-name*] **Example:** `switch# traceroute 112.112.112.1 source vlan 10`	Specifies the source IPv4 address of the traceroute packets from the specified IP address, hostname, or interface.
traceroute6 {*dest-ipv6-addr* | *hostname*} [**source** {*dest-ipv6-addr* | *hostname* | *interface*}] [**vrf** *vrf-name*] **Example:** `switch# traceroute6 2010:11:22:0:1000::1 source ethernet 2/2`	Specifies the source IPv6 address of the traceroute6 packets from the specified IP address, hostname, or interface.

Command	Purpose
[**no**] **ip traceroute source-interface** *interface* [**vrf** *vrf-name*] **Example:** `switch(config)# ip traceroute source-interface loopback 1`	Generates traceroute or traceroute6 packets with the source IP address from the configured interface.

show ip traceroute source-interface [**vrf** *vrf-name*]
Example:
`switch# show ip traceroute source-interface vrf all`
`VRF Name Interface`
`default loopback1`
	Displays the configured source interface for the traceroute.

ip icmp-errors source-interface *interface*
Example 1:
`switch(config)# ip icmp-errors source-interface loopback 1`
Example 2:
`switch(config)# vrf context vrf-blue`
`switch(config-vrf)# ip icmp-errors source-interface loopback 2`
	Generates ICMP error packets with the source IPv4 or IPv6 address from the configured interface.
	You can also optionally configure this command within a virtual routing and forwarding instance (VRF).

Monitoring Processes and CPUs

Use the **show processes** command to identify the processes that are running and the status of each process. The command output includes the following:

- PID = process ID.
- State = process state.
- PC = current program counter in hexadecimal format.
- Start_cnt = how many times a process has been started (or restarted).
- TTY = terminal that controls the process. A - (hyphen) usually means a daemon not running on any particular TTY.
- Process = name of the process.

Process states are as follows:

- D = uninterruptible sleep (usually I/O).
- R = runnable (on run queue).

- S = sleeping.
- T = traced or stopped.
- Z = defunct (zombie) process.
- NR = not-running.
- ER = should be running but currently not-running.

Note Typically, the ER state designates a process that has been restarted too many times, causing the system to classify it as faulty and disable it.

```
switch# show processes ?
cpu      Show processes CPU Info
log      Show information about process logs
memory   Show processes Memory Info

switch# show processes
PID    State  PC        Start_cnt    TTY   Type  Process
-----  -----  --------  -----------  ----  ----  -------------
    1      S  b7f9e468            1     -     O  init
    2      S         0            1     -     O  migration/0
    3      S         0            1     -     O  ksoftirqd/0
    4      S         0            1     -     O  desched/0
    5      S         0            1     -     O  migration/1
    6      S         0            1     -     O  ksoftirqd/1
    7      S         0            1     -     O  desched/1
    8      S         0            1     -     O  events/0
    9      S         0            1     -     O  events/1
   10      S         0            1     -     O  khelper
   15      S         0            1     -     O  kthread
   24      S         0            1     -     O  kacpid
  103      S         0            1     -     O  kblockd/0
  104      S         0            1     -     O  kblockd/1
  117      S         0            1     -     O  khubd
  184      S         0            1     -     O  pdflush
  185      S         0            1     -     O  pdflush
  187      S         0            1     -     O  aio/0
  188      S         0            1     -     O  aio/1
  189      S         0            1     -     O  SerrLogKthread

...
```

Using the show processes cpu Command

Use the **show processes cpu** command to display CPU utilization. The command output includes the following:

- Runtime(ms) = CPU time that the process has used, expressed in milliseconds.
- Invoked = Number of times that the process has been invoked.
- uSecs = Average CPU time, in microseconds, for each process invocation.
- 1Sec = Percentage of CPU utilization for the last 1 second.

```
switch# show processes cpu
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
```

```
-----  -----------  --------  -----  -----  -----------
    1         2264    108252     20      0  init
    2          950    211341      4      0  migration/0
    3         1154  32833341      0      0  ksoftirqd/0
    4          609    419568      1      0  desched/0
    5          758    214253      3      0  migration/1
    6         2462  155309355      0      0  ksoftirqd/1
    7         2496    392083      6      0  desched/1
    8          443    282990      1      0  events/0
    9          578    260184      2      0  events/1
   10           56      2681     21      0  khelper
   15            0        30     25      0  kthread
   24            0         2      5      0  kacpid
  103           81        89    914      0  kblockd/0
  104           56       265    213      0  kblockd/1
  117            0         5     17      0  khubd
  184            0         3      3      0  pdflush
  185         1796    104798     17      0  pdflush
  187            0         2      3      0  aio/0
  188            0         2      3      0  aio/1
  189            0         1      3      0  SerrLogKthread
...
```

Interpreting transient CPU spikes

Cisco NX-OS employs a preemptive scheduler allowing processes to use available CPU resources to complete short tasks efficiently. As a result, short CPU spikes observed in commands like `show processes cpu`, `show processes cpu history`, or monitoring tools are common during normal operation and do not necessarily indicate a problem.

Transient spikes that last less than a second and return to normal do not necessarily indicate a problem, especially when the overall system CPU remains healthy and idle CPU remains available. Use `show system resources` to verify the overall CPU state and use `show processes cpu` to identify whether a process is consuming CPU persistently.

Key points to consider:

- Transient CPU spikes lasting less than a second and returning to normal are typically harmless, especially if the overall system CPU remains healthy and idle CPU is available. Short CPU spikes from processes such as `python`, `nxpython`, or `vsh.bin` can occur when commands, scripts, or management tasks run; these are expected if brief and followed by a return to normal CPU utilization.
- Use `show system resources` to verify the overall CPU state.
- Use `show processes cpu` to identify if any process is persistently consuming CPU.
- The history option reports possible CPU spikes that do not necessarily indicate a problem. However, if average CPU usage remains high compared to normal, baseline CPU usage for a particular network, investigate high CPU usage.

Investigate high CPU usage when one or more of these conditions occur:

- Average CPU utilization remains high compared to the normal baseline for the network.
- Idle CPU remains low for a sustained period.
- A process shows sustained high CPU usage across multiple samples.
- The switch reports service impact, process restarts, packet drops, control-plane drops, or related syslog messages.

- CPU spikes correlate with user-visible traffic loss, management-plane issues, or protocol instability.

Using the show system resources Command

Use the **show system resources** command to display system-related CPU and memory statistics. The output includes the following:

- Load average is defined as the number of running processes. The average reflects the system load over the past 1, 5, and 15 minutes.
- Processes displays the number of processes in the system and how many are actually running when the command is issued.
- CPU states show the CPU usage percentage in user mode, kernel mode, and idle time in the last 1 second.
- Memory usage provides the total memory, used memory, free memory, memory used for buffers, and memory used for cache in kilobytes. Buffers and cache are also included in the used memory statistics.

```
switch# show system resources
Load average:   1 minute: 0.00   5 minutes: 0.02   15 minutes: 0.05
Processes   :   355 total, 1 running
CPU states  :   0.0% user,   0.2% kernel,   99.8% idle
        CPU0 states  :   0.0% user,   1.0% kernel,   99.0% idle
        CPU1 states  :   0.0% user,   0.0% kernel,   100.0% idle
        CPU2 states  :   0.0% user,   0.0% kernel,   100.0% idle
        CPU3 states  :   0.0% user,   0.0% kernel,   100.0% idle
Memory usage:   16402560K total,   2664308K used,   13738252K free
Current memory status: OK
```

Using Onboard Failure Logging

Cisco NX-OS provides the facility to log failure data to the persistent storage, which can be retrieved and displayed for analysis. This onboard failure logging (OBFL) feature stores failure and environmental information in nonvolatile memory on the module. This information will help you analyze failed modules.

The data stored by the OBFL facility includes the following:

- Time of initial power on
- Slot number of the module in the chassis
- Initial temperature of the module
- Firmware, BIOS, FPGA, and ASIC versions
- Serial number of the module
- Stack trace for crashes
- CPU hog information
- Memory leak information
- Software error messages
- Hardware exception logs

- Environmental history
- OBFL specific history information
- ASIC interrupt and error statistics history
- ASIC register dumps

For more information about configuring OBFL, see the *Cisco Nexus 9000 Series NX-OS System Management Configuration Guide*.

Using OBFL Error Status Command

Beginning Cisco NX-OS Release 9.3(3), Cisco Nexus 9000 series switches supports several counters to monitor and log fibre channel interfaces. The counters help identify and troubleshoot issues at FCMAC level.

Use the **show logging onboard error-stats** command to display onboard error statistics. The output includes the following counters:

- FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER
- FCP_CNTR_MAC_RX_EOFA
- FCP_CNTR_MAC_RX_CRC
- FCP_CNTR_MAC_RX_MAX_FRAME_TRUNCATE
- FCP_CNTR_MAC_RX_MIN_FRAME_PAD
- FCP_CNTR_CREDIT_LOSS
- FCP_CNTR_TX_WT_AVG_B2B_ZERO

The following is an example outout of the **show logging onboard error-stats** command:

```
switch# show logging onboard error-stats
----------------------------
    Module:  1
----------------------------


-------------------------------------------------------------------------------------------------
ERROR STATISTICS INFORMATION FOR DEVICE: FCMAC
-------------------------------------------------------------------------------------------------
Interface Range    |                                       |         |Time Stamp
                   |            Error Stat Counter Name    |  Count  |MM/DD/YY HH:MM:SS
                   |                                       |         |
-------------------------------------------------------------------------------------------------
fc1/9              |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |4        |11/15/19 09:54:40
fc1/33             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |4        |11/15/19 09:37:53
fc1/36             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |4        |11/15/19 09:05:13
fc1/37             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |4        |11/15/19 08:42:56
fc1/37             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |4        |11/15/19 08:21:19
fc1/28             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |4        |11/15/19 08:20:59
fc1/9              |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |5996     |11/14/19 10:25:45
fc1/9              |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |5992     |11/14/19 06:19:04
fc1/36             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |22112    |11/14/19 06:19:04
fc1/36             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |21876    |11/14/19 06:18:44
fc1/36             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |21368    |11/14/19 06:18:24
fc1/36             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |20872    |11/14/19 06:18:04
fc1/36             |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |20292    |11/14/19 06:17:44
```

```
fc1/36            |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |19720    |11/14/19 06:17:24
fc1/36            |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |19284    |11/14/19 06:17:04
fc1/36            |FCP_CNTR_MAC_RX_BAD_WORDS_FROM_DECODER  |18788    |11/14/19 06:16:44
```

Using Diagnostics

Generic online diagnostics (GOLD) define a common framework for diagnostic operations across Cisco platforms. The GOLD implementation checks the health of hardware components and verifies proper operation of the system data and control planes. Some tests take effect when the system is booting up; other tests take effect when the system is operational. A booting module goes through a series of checks before coming online to allow the system to detect faults in the hardware components at bootup and to ensure that a failing module is not introduced in a live network.

Defects are also diagnosed during system operation or runtime. You can configure a series of diagnostic checks to determine the condition of an online system. You must distinguish between disruptive and nondisruptive diagnostic tests. Although nondisruptive tests occur in the background and do not affect the system data or control planes, disruptive tests do affect live packet flows. You should schedule disruptive tests during special maintenance windows. The **show diagnostic content module** command output displays test attributes such as disruptive or nondisruptive tests.

You can configure runtime diagnostic checks to run at a specific time or to run continually in the background.

Health-monitoring diagnostic tests are nondisruptive, and they run in the background while the system is in operation. The role of online diagnostic health monitoring is to proactively detect hardware failures in the live network environment and inform you of a failure.

GOLD collects diagnostic results and detailed statistics for all tests including the last execution time, the first and last test pass time, the first and last test failure time, the total run count, the total failure count, the consecutive failure count, and the error code. These test results help administrators determine the condition of a system and understand the reason for a system failure. Use the **show diagnostic result** command to view diagnostic results.

For more information about configuring GOLD, see the *Cisco Nexus 9000 Series NX-OS System Management Configuration Guide*.

Using Embedded Event Manager

Embedded Event Manager (EEM) is a policy-based framework that allows you to monitor key system events and then act on those events through a set policy. The policy is a preprogrammed script that you can load that defines actions that the device should invoke based on set events occurring. The script can generate actions, including, but not limited to, generating custom syslog or SNMP traps, invoking CLI commands, forcing a failover, and much more.

For more information about configuring EEM, see the *Cisco Nexus 9000 Series NX-OS System Management Configuration Guide*.

Using Ethanalyzer

Ethanalyzer is a Cisco NX-OS protocol analyzer tool implementation of the open source software TShark which is a terminal version of Wireshark (formerly Ethereal). You can use Ethanalyzer to troubleshoot your

network by capturing and analyzing control-plane traffic on inband and management interfaces across all Nexus platforms.

Note Performing ethanalyzer using the **front-panel** option for an interface bundled into a port-channel is not supported. Instead, use the **port-channel** option.

To configure Ethanalyzer, use the following commands:

Command	Purpose
ethanalyzer local interface inband	Captures packets sent or received by the supervisor through the inband interface and displays summarized protocol information for captured packets.
ethanalyzer local interface inband-in	Captures packets received by the supervisor through the inband interface and displays summarized protocol information for captured packets.
ethanalyzer local interface inband-out	Captures packets sent by the supervisor through the inband interface and displays summarized protocol information for captured packets.
ethanalyzer local interface mgmt	Captures packets sent or received by the management interface and displays summarized protocol information for captured packets.
ethanalyzer local interface front-panel	Captures Layer 2 frames and Layer 3 packets sent or received by the supervisor through a Layer 3 (routed) front-panel port and displays summarized protocol information for the captured traffic. **Note** This command can capture Layer 2 frames, such as bridge protocol data units (BPDUs), through Layer 2 (switchport) front-panel ports. Capturing IPv4 or IPv6 packets through Layer 2 front-panel ports is not supported and might not work reliably.
ethanalyzer local interface port-channel	Captures packets sent or received by the supervisor through a Layer 3 (routed) port-channel interface and displays summarized protocol information for captured packets. **Note** This command does not support capturing packets sent or received by the supervisor through Layer 2 (switchport) port-channel interfaces.
ethanalyzer local interface vlan	Captures packets sent or received by the supervisor through a Layer 3 Switch Virtual Interface (SVI) and displays summarized protocol information.

Command	Purpose
ethanalyzer local interface netstack	Captures packets sent or received by the supervisor through the Netstack software component and displays summarized protocol information.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **limit-captured-frames**	Limits the number of frames to capture within the Ethanalyzer session. The number of frames can be an integer value from 0 to 500,000. If 0 is provided, then a maximum of 500,000 frames will be captured before the Ethanalyzer session automatically stops.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **limit-frame-size**	Limits the length of the frame to capture. The length of frame can be an integer value from 192 to 65,536.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **capture-filter**	Filters the types of packets to capture using Berkeley Packet Filter (BPF) syntax.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **display-filter**	Filtersthe types of captured packets to display using Wireshark or TShark Display Filters.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **write**	Saves the captured data to a file. Valid storage options include the switch's bootflash, logflash, a USB storage device, or volatile storage.
ethanalyzer local read	Opens a captured data file and analyzes the file. Valid storage options include the switch's bootflash, logflash, a USB storage device, or volatile storage.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **autostop**	Specifies a condition that will automatically stop the Ethanalyzer session. You can specify the duration of the session in seconds, number of files to capture when writing captured packets to a file using the **write** keyword, and file size when writing captured packets to a file using the **write** keyword.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **capture-ring-buffer**	Specifies the capture ring buffer options for Ethanalyzer. This option will continuously write to one or more files in a ring buffer when combined with the write keyword. You can specify the duration in seconds that Ethanalyzer will wait before writing to a new file, the number of files to keep as part of the ring buffer, and the file size of each individual file in the ring buffer.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **detail**	Displays detailed protocol information for captured packets.
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **raw**	Displays captured packets inhex format.

Command	Purpose
ethanalyzer local interface {**front-panel** | **inband** | **inband-in** | **inband-out** | **mgmt** | **port-channel** | **vlan**} **vrf**	Specifies the VRF that the Layer 3 interface is a member if the Layer 3 interface is in a non-default VRF.

Guidelines and Limitations

- If a Layer 3 interface is a member of a non-default VRF and is specified in an Ethanalyzer session (for example, through the **ethanalyzer local interface front-panel ethernet1/1** or **ethanalyzer local interface port-channel1** commands), you must specify the VRF that the Layer 3 interface is a member of within the Ethanalyzer session using the **vrf** keyword. For example, to capture packets received or sent by the supervisor through Layer 3 front-panel port Ethernet1/1 in VRF "red", use the **ethanalyzer local interface front-panel ethernet1/1 vrf red** command.
- When writing to a file, Ethanalyzer will automatically stop if the Ethanalyzer session captures 500,000 packets, or if the size of the file reaches ~11 megabytes, whichever comes first.

Examples

```
switch(config)# ethanalyzer local interface inband
<CR>
> Redirect it to a file
>> Redirect it to a file in append mode
autostop Capture autostop condition
capture-filter Filter on ethanalyzer capture capture-ring-buffer Capture ring buffer option
decode-internal Include internal system header decoding detail Display detailed protocol
information
display-filter Display filter on frames captured
limit-captured-frames Maximum number of frames to be captured (default is 10) limit-frame-size
 Capture only a subset of a frame
mirror Filter mirrored packets
raw Hex/Ascii dump the packet with possibly one line summary
write Filename to save capture to
| Pipe command output to filter
```

```
switch(config)# ethanalyzer local interface inband Capturing on 'ps-inb'

1 2021-07-26 09:36:36.395756813 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 64 PRI:
7 DEI: 0 ID: 4033
2 2021-07-26 09:36:36.395874466 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 205 PRI:
7 DEI: 0 ID: 4033
4 3 2021-07-26 09:36:36.395923840 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 806 PRI:
7 DEI: 0 ID: 4033
4 2021-07-26 09:36:36.395984384 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 1307 PRI:
7 DEI: 0 ID: 4033
5 2021-07-26 09:37:36.406020552 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 64 PRI:
7 DEI: 0 ID: 4033
6 2021-07-26 09:37:36.406155603 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 205 PRI:
7 DEI: 0 ID: 4033
7 2021-07-26 09:37:36.406220547 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 806 PRI:
7 DEI: 0 ID: 4033
8 8 2021-07-26 09:37:36.406297734 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 1307
PRI: 7 DEI: 0 ID: 4033
9 2021-07-26 09:38:36.408983263 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 64 PRI:
7 DEI: 0 ID: 4033
10 10 2021-07-26 09:38:36.409101470 00:22:bd:cf:b9:01 → 00:22:bd:cf:b9:00 0x3737 205
PRI: 7 DEI: 0 ID: 4033
```

Use the **detail** option for detailed protocol information. Ctrl+C can be used to abort and get the switch prompt back in the middle of the capture, if required.

```
switch(config)# ethanalyzer local interface inband detail
Capturing on 'ps-inb'
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface ps-inb, id
 0
Interface id: 0 (ps-inb) Interface name: ps-inb
Encapsulation type: Ethernet (1)
Arrival Time: Jul 26, 2021 11:54:37.155791496 UTC
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1627300477.155791496 seconds
[Time delta from previous captured frame: 0.000000000 seconds] [Time delta from previous
displayed frame: 0.000000000 seconds] [Time since reference or first frame: 0.000000000
seconds] Frame Number: 1
Frame Length: 64 bytes (512 bits)
Capture Length: 64 bytes (512 bits) [Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:vlan:ethertype:data] Ethernet II, Src: 00:22:bd:cf:b9:01,
 Dst: 00:22:bd:cf:b9:00
Destination: 00:22:bd:cf:b9:00 Address: 00:22:bd:cf:b9:00
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast) Source: 00:22:bd:cf:b9:01
Address: 00:22:bd:cf:b9:01
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast) Type: 802.1Q Virtual
LAN (0x8100)
802.1Q Virtual LAN, PRI: 7, DEI: 0, ID: 4033
111. .... .... .... = Priority: Network Control (7) 4 ...0 .... .... .... = DEI: Ineligible
.... 1111 1100 0001 = ID: 4033
Type: Unknown (0x3737) Data (46 bytes)

0000 a9 04 00 00 7d a2 fe 60 47 4f 4c 44 00 0b 0b 0b ....}..`GOLD....
0010 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b ................


0020 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b ................
Data: a90400007da2fe60474f4c44000b0b0b0b0b0b0b0b0b0b0b… [Length: 46]
```

Use the **capture-filter** option to select which packets to display or save to disk during capture. A capture filter maintains a high rate of capture while it filters. Because full dissection has not been done on the packets, the filter fields are predefined and limited.

Use the **display-filter** option to change the view of a capture file. A display filteruses fully dissected packets, so you can do very complex and advanced filtering when you analyze a network tracefile. Ethanalyzer writes captured data to a temporary file if it is not instructed to write captured data to a file elsewhere. This temporary file can fill quickly when a display filter is used without the user's knowledge, since all packets matching the **capture-filter** option are written to the temporary file, but only packets matching the **display-filter** option are displayed.

In this example, **limit-captured-frames** is set to 5. With the **capture-filter** option, Ethanalyzer shows you five packets which match the filter **host 10.10.10.2**. With the **display-filter** option, Ethanalyzer first captures five packets then displays only the packets that match the filter **ip.addr==10.10.10.2**.

```
switch(config)# ethanalyzer local interface inband capture-filter "host 10.10.10.2"
limit-captured-frames 5
Capturing on inband
2013-02-10 12:51:52.150404 10.10.10.1 -> 10.10.10.2 UDP Source port: 3200 Destination port:
3200
2013-02-10 12:51:52.150480 10.10.10.2 -> 10.10.10.1 UDP Source port: 3200 Destination port:
3200
2013-02-10 12:51:52.496447 10.10.10.2 -> 10.10.10.1 UDP Source port: 3200 Destination port:
3200
```

```
2013-02-10 12:51:52.497201 10.10.10.1 -> 10.10.10.2 UDP Source port: 3200 Destination port:
3200
2013-02-10 12:51:53.149831 10.10.10.1 -> 10.10.10.2 UDP Source port: 3200 Destination port:
3200
5 packets captured
switch(config)# ethanalyzer local interface inband display-filter "ip.addr==10.10.10.2"
limit-captured-frame 5
Capturing on inband
2013-02-10 12:53:54.217462 10.10.10.1 -> 10.10.10.2 UDP Source port: 3200 Destination port:
3200
2013-02-10 12:53:54.217819 10.10.10.2 -> 10.10.10.1 UDP Source port: 3200 Destination port:
3200
2 packets captured
```

The **write** option lets you write the capture data to a file in one of the storage devices (such as boothflash or logflash) on the Cisco Nexus 9000 Series Switch for later analysis. The capture file size is limited to 10 MB.

An example Ethanalyzer command with a **write** option is **ethanalyzer local interface inband writebootflash:***capture_file_name*. The following is an example of a **write** option with **capture-filter** and an output file name of **first-capture**:

```
switch(config)# ethanalyzer local interface inband capture-filter "host 10.10.10.2"
limit-captured-frame 5 write ?
bootflash: Filename logflash: Filename slot0:     Filename
usb1:      Filename
usb2: Filename volatile: Filename
switch(config)# ethanalyzer local interface inband capture-filter "host 10.10.10.2"
limit-captured-frame 5 write bootflash:first-capture
```

When the capture data is saved to a file, the captured packets are, by default, not displayed in the terminal window. The **display** option forces Cisco NX-OS to display the packets while it saves the capture data to a file.

The **capture-ring-buffer** option creates multiple files after a specified number of seconds, a specified number of files, or a specified file size. The following are the definitions of those options:

```
switch(config)# ethanalyzer local interface inband capture-ring-buffer ?
duration Stop writing to the file or switch to the next file after value seconds have elapsed
files Stop writing to capture files after value number of files were written or begin again
 with the first file after value number of files were
written (form a ring buffer)
filesize Stop writing to a capture file or switch to the next file after it reaches a size
 of value kilobytes
```

The **read** option lets you read the saved file on the device itself.

```
switch(config)# ethanalyzer local read bootflash:first-capture
2013-02-10 12:51:52.150404 10.10.10.1 -> 10.10.10.2 UDP Source port: 3200 Destination port:
3200
2013-02-10 12:51:52.150480 10.10.10.2 -> 10.10.10.1 UDP Source port: 3200 Destination port:
3200
2013-02-10 12:51:52.496447 10.10.10.2 -> 10.10.10.1 UDP Source port: 3200 Destination port:
3200
2013-02-10 12:51:52.497201 10.10.10.1 -> 10.10.10.2 UDP Source port: 3200 Destination port:
3200
2013-02-10 12:51:53.149831 10.10.10.1 -> 10.10.10.2 UDP Source port: 3200 Destination port:
3200

switch(config)# ethanalyzer local read bootflash:first-capture detail Frame 1 (110 bytes
on wire, 78 bytes captured)
------------------------------SNIP-----------------------------------------------
[Frame is marked: False]
[Protocols in frame: eth:ip:udp:data]
Ethernet II Src: 00:24:98:6f:ba:c4 (00:24:98:6f:ba:c4), Dst: 00:26:51:ce:0f:44
(00:26:51:ce:0f:44)
```

```
Destination: 00:26:51:ce:0f:44 (00:26:51:ce:0f:44) Address: 00:26:51:ce:0f:44
(00:26:51:ce:0f:44)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default) Source:
00:24:98:ce:6f:ba:c4 (00:24:98:6f:ba:c4)
Address: 00:24:98:6f:ba:c4 (00:24:98:6f:ba:c4)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default) Type: IP
 (0x0800)
Internet Protocol, Src: 10.10.10.1 (10.10.10.1), Dst: 10.10.10.2 (10.10.10.2)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0xc0 (DSC) 0x30: Class Selector 6; ECN: 0x00)
------------------------------SNIP-----------------------------------------------
```

You can also transfer the file to a server or a PC and read it with Wireshark or any other application that can read files with .cap or .pcap file formats.

```
switch(config)# copy bootflash:first-capture tftp:
Enter vrf (If no input, current vrf 'default' is considered): management
Enter hostname for the tftp server: 192.168.21.22
Trying to connect to tftp server......
Connection to Server Established. TFTP put operation was successful
Copy complete.
```

The **decode-internal** option reports internal information on how the Nexus 9000 forwards the packet. This information helps you understand and troubleshoot the flow of packets through the CPU.

```
switch(config)# ethanalyzer local interface inband decode-internal capture-filter "host
10.10.10.2" limit-captured-frame 5 detail
Capturing on inband NXOS Protocol
NXOS VLAN: 0==================->VLAN in decimal=0=L3 interface
NXOS SOURCE INDEX: 1024 ==================->PIXN LTL source index in decimal=400=SUP
inband
NXOS DEST INDEX: 2569==================-> PIXN LTL destination index in decimal=0xa09=e1/25
 Frame 1: (70 bytes on wire, 70 bytes captured)
Arrival Time: Feb 10, 2013 22:40:02.216492000
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1627300477.155791496 seconds
[Time delta from previous captured frame: 0.000000000 seconds] [Time delta from previous
displayed frame: 0.000000000 seconds] [Time since reference or first frame: 0.000000000
seconds] Frame Number: 1
Frame Length: 70 bytes Capture Length: 70 bytes [Frame is marked: False]
[Protocols in frame: eth:ip:udp:data]
Ethernet II, Src: 00:26:51:ce:0f:43 (00:26:51:ce:0f:43), Dst: 00:24:98:6f:ba:c3
(00:24:98:6f:ba:c3)
Destination: 00:24:98:6f:ba:c3 (00:24:98:6f:ba:c3) Address: 00:24:98:6f:ba:c3
(00:24:98:6f:ba:c3)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default) Source:
00:26:51:ce:0f:43 (00:26:51:ce:0f:43)
------------------------------SNIP-----------------------------------------------
```

Convert the NX-OS index to hexadecimal, then use the **show system internal pixm info ltl {index}** command to map the local target logic (LTL) index to a physical or logical interface.

Capture Traffic to or from an IP Host

```
host 1.1.1.1
```

Capture Traffic to or from a Range of IP Addresses

```
net 172.16.7.0/24
```

net 172.16.7.0 mask 255.255.255.0

Capture Traffic from a Range of IP Addresses

```
src net 172.16.7.0/24
```

src net 172.16.7.0 mask 255.255.255.0

Capture Traffic to a Range of IP Addresses

```
dst net 172.16.7.0/24
```

dst net 172.16.7.0 mask 255.255.255.0

Capture UDLD, VTP, or CDP Traffic

UDLD is Unidirectional Link Detection, VTP is the VLAN Trunking Protocol, and CDP is the Cisco Discovery Protocol.

ether host 01:00:0c:cc:cc:cc

Capture Traffic to or from a MAC Address

ether host 00:01:02:03:04:05



Note and = &&

or = ||

not = !

MAC address format : xx:xx:xx:xx:xx:xx

Common Control Plane Protocols

- UDLD: Destination Media Access Controller (DMAC) = 01-00-0C-CC-CC-CC and EthType = 0x0111
- LACP: DMAC = 01:80:C2:00:00:02 and EthType = 0x8809. LACP stands for Link Aggregation Control Protocol
- STP: DMAC = 01:80:C2:00:00:00 and EthType = 0x4242 - or - DMAC = 01:00:0C:CC:CC:CD and EthType = 0x010B
- CDP: DMAC = 01-00-0C-CC-CC-CC and EthType = 0x2000
- LLDP: DMAC = 01:80:C2:00:00:0E or 01:80:C2:00:00:03 or 01:80:C2:00:00:00 and EthType = 0x88CC
- DOT1X: DMAC = 01:80:C2:00:00:03 and EthType = 0x888E. DOT1X stands for IEEE 802.1x
- IPv6: EthType = 0x86DD
- List of UDP and TCP port numbers

Ethanalyzer does not capture data traffic that Cisco NX-OS forwards in the hardware.

Ethanalyzer uses the same capture filter syntax as **tcpdump** and uses the Wireshark display filter syntax.

This example shows captured data (limited to four packets) on the management interface:

```
switch(config)# ethanalyzer local interface mgmt limit-captured-frames 4
Capturing on eth1

2013-05-18 13:21:21.841182 172.28.230.2 -> 224.0.0.2 BGP Hello (state Standy)
2013-05-18 13:21:21.842190 10.86.249.17 -> 172.28.231.193 TCP 4261 > telnet [AC] Seq=0 Ack=0
 Win=64475 Len=0
2013-05-18 13:21:21.843039 172.28.231.193 -> 10.86.249.17 TELNET Telnet Data ..
2013-05-18 13:21:21.850463 00:13:5f:1c:ee:80 -> ab:00:00:02:00:00 0x6002 DEC DN

Remote Console
4 packets captured
```

This example shows detailed captured data for one HSRP packet:

```
switch(config)# ethanalyzer local interface mgmt capture-filter "udp port 1985"
limit-captured-frames 1
Capturing on eth1
Frame 1 (62 bytes on wire, 62 bytes captured)
Arrival Time: May 18, 2013 13:29:19.961280000
[Time delta from previous captured frame: 1203341359.961280000 seconds]
[Time delta from previous displayed frame: 1203341359.961280000 seconds]
[Time since reference or first frame: 1203341359.961280000 seconds]
Frame Number: 1
Frame Length: 62 bytes
Capture Length: 62 bytes
[Frame is marked: False]
[Protocols in frame: eth:ip:udp:hsrp]

Ethernet II, Src: 00:00:0c:07:ac:01 (00:00:0c:07:ac:01), Dst: 01:00:5e:00:00:02
(01:00:5e:00:00:02)
Destination: 01:00:5e:00:00:02 (01:00:5e:00:00:02)
Address: 01:00:5e:00:00:02 (01:00:5e:00:00:02)
.... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
Source: 00:00:0c:07:ac:01 (00:00:0c:07:ac:01)
Address: 00:00:0c:07:ac:01 (00:00:0c:07:ac:01)

.... ...0 .... .... .... .... = IG bit: Individual address (unicast)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)

Type: IP (0x0800)
Internet Protocol, Src: 172.28.230.3 (172.28.230.3), Dst: 224.0.0.2 (224.0.0.2)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00)
1100 00.. = Differentiated Services Codepoint: Class Selector 6 (0x30)
.... ..0. = ECN-Capable Transport (ECT): 0
.... ...0 = ECN-CE: 0

Total Length: 48
Identification: 0x0000 (0)
Flags: 0x00
0... = Reserved bit: Not set
.0.. = Don't fragment: Not set
..0. = More fragments: Not set
Fragment offset: 0
Time to live: 1
```

```
Protocol: UDP (0x11)
Header checksum: 0x46db [correct]
[Good: True]
[Bad : False]

Source: 172.28.230.3 (172.28.230.3)
Destination: 224.0.0.2 (224.0.0.2)
User Datagram Protocol, Src Port: 1985 (1985), Dst Port: 1985 (1985)
Source port: 1985 (1985)
Destination port: 1985 (1985)
Length: 28
Checksum: 0x8ab9 [correct]
[Good Checksum: True]
[Bad Checksum: False]

Cisco Hot Standby Router Protocol
Version: 0
Op Code: Hello (0)
State: Active (16)
Hellotime: Default (3)
Holdtime: Default (10)
Priority: 105
Group: 1
Reserved: 0Authentication Data: Default (cisco)
Virtual IP Address: 172.28.230.1 (172.28.230.1)

1 packets captured
```

This example uses a display filter to show only those HSRP packets that have an active HSRP state:

```
switch(config)# ethanalyzer local interface mgmt display-filter "hsrp.state==Active"
limit-captured-frames 2
Capturing on eth1

2013-05-18 14:35:41.443118 172.28.230.3 -> 224.0.0.2 HSRP Hello (state Active)
2013-05-18 14:35:44.326892 172.28.230.3 -> 224.0.0.2 HSRP Hello (state Active)
2 packets captured
```

Ethanalyzer Background Capture Process and Autocollection of Inband Packets

Ethanalyzer can be enabled to run as a background task to capture inband packets. The inband packet data is kept in RAM memory in PCAP files. A configurable limited amount of PCAP data (configurable number of files with a configurable file size) is available at any time. When the limit is reached, the oldest file gets overwritten with the current capture in a cyclic way.

The data captured by the Ethanalyzer background task is in RAM, and cyclically overwritten, not taking up bootflash space. For the user to be able to look at the data, a snapshot needs to be taken, which will copy the packet capture info taken by the background process in PCAP format from RAM to non-volatile storage (bootflash) for viewing. Users need to consider available bootflash space when taking a snapshot.

The snapshot can be triggered manually by the user via CLI. EEM policies, can be as well used to trigger the snapshot upon certain events. Use case examples of triggers are inband rate exceeding a defined threshold, CoPP drops exceeding a threshold - the snapshot gives insight what packets were hitting the inband upto the moment of the event.

When monitoring rates, a threshold that exceeds the normally expected or accepted rates by the user, needs to be set to avoid an excess of alerts for non-issues.Attention need to be paid when increasing the max-triggers

in the autocollection EEM policy below. Not following these practices can result an excessive amount of irrelevant PCAP data to be snapshotted with a potential of filling up the bootflash.

Ethanalyzer has added CLIs to enable and configure the background session, start and stop the session, snapshot the Ethanalyzer information as well as show commands to look at the background session status. All CLIs are to be run from enable mode.

Guidelines and Limitations for Ethanalyzer Background Capture Process

- The Ethanalyzer background process stores .pcap files in the /tmp directory, which has limited storage capacity. Users must ensure that the total size of all `.pcap` files does not exceed the available `/tmp` storage space.

  To calculate the total space required for Ethanalyzer `.pcap` files, use the following formula:

  fileSize * numFiles < Available /tmp Space

  Before starting the Ethanalyzer background process, validate /tmp storage availability using the following command:

  ```
  bash-4.4# df -k /tmp
  Filesystem     1K-blocks  Used Available Use% Mounted on
  none              614400  2760    611640   1% /var/volatile/tmp
  ```

- Restarting the Ethanalyzer background session deletes all previously captured .pcap files in /tmp. Users must copy important data to persistent storage (e.g., /bootflash) using the ethanalyzer copy-background-snapshot command before restarting.

  Use the following command to copy .pcap files to bootflash before restarting:

  ```
  ethanalyzer copy-background-snapshot
  ```

- Consider available bootflash space before taking a snapshot, as snapshots are copied from /tmp to bootflash. Insufficient bootflash space may result in failed snapshots or incomplete data storage.

  To save space, use the compressed tar option:

  ```
  ethanalyzer copy-compressed-background-snapshot
  ```

- Use Event Manager (EEM) policies to trigger snapshots based on events like inband rate thresholds or CoPP drops. Configure the max-triggers parameter carefully to avoid excessive alerts or irrelevant data snapshots, which could fill up bootflash unnecessarily.

- The background process does not automatically manage storage rollover beyond /tmp. Users must configure parameters properly to avoid filling up /tmp storage.

- If /tmp storage becomes full, collected data may be lost.

Table 2: Ethanalyzer CLIs

CLI	Description
ethanalyzer background-session config <filename|filesize|numfiles|session>	Configure parameters of ethanalyzer background process/session for capturinig packet in circular buffer. • Filename - Background packet capture file name saved by Ethanalyzer background capture process. • Filesize - Size of individual capture file that are in temporary buffer. Value ranges from 1-65536 KB. • Numfiles - Number of maximum pcap files to be stored in temporary buffer. Value ranges from 2-16. • Session – Enable/Disable Ethanalyzer background capture session.
ethanalyzer background-session restart	Start/Restart Ethanalyzer background capture session.
ethanalyzer background-session stop	Stops the Ethanalyzer background capture session.
show ethanalyzer background-session processes	Show Ethanalyzer background capture session details.
show ethanalyzer background-session config	Print Ethanalyzer background capture session configuration file.
ethanalyzer copy-background-snapshot	Copy the files captured in temporary buffer to bootflash. Files are in pcap format.
ethanalyzer copy-compressed-background-snapshot	Tar the files captured in temporary buffer and copy the tar file to bootflash. **Note** Issuing this CLI multiple times will delete the old tar file. User discretion is adviced to copy the old tar file if it exists in bootflash.

Beginning with Cisco NX-OS, release 10.1(2) Ethanalyzer Autocollection CLI is supported on all Cisco Nexus 9000 Series platforms.

Ethanalyzer Autocollection CLI Warnings

The following are the Ethnalyzer Autocollection CLI warnings:

- Whenever any change is made to the background process, user is required ot restart/start the Ethanalyzer background process. The following warning message shall be displayed to user when any config change is made:

  “Please restart the Ethanalyzer background process for any config change to take effect.”

- In the platforms where supervisor redundancy is supported, switchover of the active supervisor can lead to the Ethanalyzer background capture process to fail to start automatically. User must manually restart the Ethanalyzer background process. If the user wants the Ethanalyzer background process to start automatically after switchover, the user must configure the session enable on the active supervisor, and then reload the switch to take effect. After this, even if the switchover occurs, the Ethanalyzer background capture process will start automatically in the newly active supervisor.

CLI Examples

Example CLI Output - All commands are run from enable mode

Step 1: Enable Ethanalyzer session running in the background:

```
switch# ethanalyzer background-session config session enable
```

```
switch# dir bootflash: | include dump
       1087    Jan 29 13:55:46 2021  dumpcap_bg_session_configuration.xml
switch# show ethanalyzer background-session config
<?xml version="1.0"?>
<!-- This document contains configuration settings for background packet -->
<!-- capture session to execute in ring buffer mode. Please modify the settings
based on system resources -->
<!-- path:         background packet capture directory where ring buffer files w
ill be saved -->
<!-- filename:     background packet capture file name saved by dumpcap. Files w
ill be generated as filename_number_date format -->
<!-- filesize:     Size of individual ring buffer file in kB. Note that the file
size is limited to a maximum value of 65536 kB-->
<!-- num_of_files: value begin again with the first file after value number of f
iles were written (form a ring buffer). The maximum value should be equal to 16
-->
<!-- session:      Enable/disable background packet capture session process. App
licable for both boot-up as well as session restart -->
<ethanalyzer_config>
    <filepath>/tmp/dumpcap_bg_session_files/</filepath>
    <filename>capture</filename>
    <filesize>2048</filesize>
    <numfiles>2</numfiles>
    <session>enable</session>
</ethanalyzer_config>
```

The following is the CLI output:

```
switch# ethanalyzer background-session restart
root     30038     1  0 13:58 ttyS0    00:00:00 /usr/bin/dumpcap -n -b filesize:
2048 -b files:2 -i ps-inb -Z none -w /tmp/dumpcap_bg_session_files/capture.pcap
```

Step 2: Verifying the background session configuration parameters

```
switch# show ethanalyzer background-session process
```

Step 3: Start the background Ethanalyzer process

```
switch# ethanalyzer background-session restart
```

Step 4: Verifying the running of Ethanalyzer background capture session

```
switch# ethanalyzer background-session processes
Background session of packet analyzer:
root 17216 1 4 12:43 ttyS0 00:00:00 /usr/bin/dumpcap -n -b filesize:2048 -b files:2 -i
ps-inb -Z none -w /tmp/dumpcap_bg_session_files/capture.pcap
switch#
```

Usecase example: Execute CLI to capture a snapshot for viewing

```
switch# ethanalyzer copy-background-snapshot

Copy packet analyzer captured frames to bootflash...
Copied snapshot files :
     72 -rw-rw-rw-  1 root  root              65844 Jan 21 00:21
CAPTURE_00001_20210121001903.pcap

switch# ethanalyzer copy-compressed-background-snapshot

Copy packet analyzer captured compressed frames to bootflash...
Copied snapshot files :
     28 -rw-r--r--  1 root  root              27181 Jan 21 00:22 CAPTURE.tar.gz
```

Usecase example: Using inband rate monitoring as a trigger for autocollection of Ethanalyzer snapshot.

Table 3: Inband Rate Monitoring CLI Options

CLI	Description
Config mode	system inband cpu-mac log threshold rx rx_pps tx tx_pps throttle secondsrx_pps, tx_pps: 0-1500000 Inband rx/tx pps rate that needs to be logged when exceededseconds: log throttle interval (maximum 1 exceed log per defined interval)
Enable mode	show system inband cpu-mac log threshold" to display settings
Default	off (PPS values 0), throttle interval 120 seconds.

The assumption is that the Ethanalyzer background process feature is configured and running as explained in the previous section. This usecase has example rates for demo or example purpose, but the user needs to use a realistic rate that is considered as worthwhile logging. A threshold that exceeds the user requirements needs to be notified to avoid an excess of alerts for non-issues.

Note Attention needs to be paid when increasing the max-triggers in the autocollection EEM policy below. Not following these practices can result an excessive amount of PCAP data to be snapshotted with a potential of filling up the bootflash.

The max-triggers parameter gets checked against the amount of snapshot files persistently stored in the eem_snapshots directory on bootflash (bootflash:eem_snapshots) of the active supervisor. In case of a supervisor switchover, the number of collections on the newly active supervisor can be different from what is on the previously active supervisor, resulting in autocollection to resume or not. The resuming of autocollection depends on the snapshot bundles present on the newly active supervisor's bootflash.

Once the amount of files in the directory mentioned matches max-triggers, autocollection will stop. To start it again, user must remove the snapshot files from the directory to bring the file count to a "value" lower than max-triggers, allowing for another amount (max-triggers minus "value") of autcollections. The details explained in the Trigger-Based Event Log Auto-Collection section of the Configuring the Embedded Event Manager chapter.

Step 1: Enable inband rate monitoring

```
switch(config)# system inband cpu-mac log threshold rx 400 tx 4000 throttle 60
switch# show system inband cpu-mac log threshold
Thresholds Rx: 400 PPS, Tx; 4000 PPS
Log throttle interval: 60 seconds
```

Leveraging the trigger based log file auto-collection, as explained in the Trigger-Based Event Log Auto-Collection section of the Configuring the Embedded Event Manager chapter, creating the directory (in the example below the name of the directory is "auto_collect") and creating or enabling the EEM policy, will enable the built-in snapshot collection of event logs and ethanalyzer pcap.

Step 2: Create the directory

```
create auto_collect directory
switch# pwd
bootflash:
switch# cd scripts
switch# mkdir auto_collect
```

Step 3: Enable the event manager policy

```
switch(config)# event manager applet syslog_trigger override __syslog_trigger_default

switch(config-applet)# action 1.0 collect auto_collect rate-limit 60 max-triggers 3
$_syslog_msg
```

This will enable autocollection for a max of 1x per 60 seconds, with a total max of 3 times for the same trigger, meaning we will store up to max-triggers x num_files pcap files for the same syslog trigger (in the example: 3 x 2 = 6 files).

The above use case in action: identifying a misbehaving host 20.1.1.100 launching high volume of ICMP request.

```
switch#
2021 Jan 29 15:15:27 switch %KERN-1-SYSTEM_MSG: [17181.984601] Inband Rx threshold 400 PPS
 reached. - kernel
2021 Jan 29 15:15:28 switch %KERN-1-SYSTEM_MSG: [17182.997911] Inband Rx threshold 400 PPS
 reached. - kernel
switch# show system internal event-logs auto-collect history
DateTime              Snapshot ID  Syslog
Status/Secs/Logsize(Bytes)
2021-Jan-29 15:15:30  620969861    KERN-1-SYSTEM_MSG
PROCESSED:1:7118865
2021-Jan-29 15:15:30  201962781    KERN-1-SYSTEM_MSG
DROPPED-LASTACTIONINPROG
2021-Jan-29 15:15:29  620969861    KERN-1-SYSTEM_MSG                          PROCESSING
...
switch# dir bootflash: | include capture
    2048040    Jan 29 15:15:29 2021  capture_00004_20210129150732.pcap
     169288    Jan 29 15:15:29 2021  capture_00005_20210129151528.pcap
 ...
```

To decode the file captured via background process, please contact cisco tac team.

Use case Example: Use a custom (non built in auto collection yaml) trigger (CoPP drop threshold exceed)

The following are the assumptions:

1. Ethanalyzer background process feature is configured and running as explained before.

2. Step 2 and Step 3 of the previous use case example are in place.

Enable CoPP threshold logging for the class interested in learning why drops happen. The details are in the CoPP configuration guide (reference).

In the example, for class copp-class-normal, which includes ARP, a threshold is set to 1000000 and the logging level is set to 1 (high enough to be picked up for autocollect):

```
class copp-class-normal
    logging drop threshold 1000000 level 1
```

In the same directory used in the previous use case example (bootflash:scripts/auto_collect), add file copp.yaml with the following (copp = the component name):

```
#*********************************************************************
#
# File:   comp specific yaml
# Author:
#
# Description: Module Makefile
#
#
# Copyright (c) 2019 by cisco Systems, Inc.
# All rights reserved.
#
#
# $Id: comp specific yaml $
# $Source:  $
# $Author:  $
#
#*********************************************************************
version: 1
components:
    copp:
            default:
            copp_drops1:
              serviceCOPP:
                match: CoPP drops exceed threshold
                commands: ethanalyzer copy-background-snapshot
```

The above use case in action: identifying high volume of ARP request causing CoPP drops in the class.

```
switch#
2021 Jan 29 15:49:47 switch %COPP-1-COPP_DROPS1: CoPP drops exceed threshold in class:
copp-class-normal-log,
check show policy-map interface control-plane for more info.
switch# show policy-map interface control-plane class copp-class-normal-log
Control Plane

  Service-policy  input: copp-policy-strict-log

    class-map copp-class-normal-log (match-any)
      match access-group name copp-acl-mac-dot1x-log
      match protocol arp
      set cos 1
      threshold: 1000000, level: 1
      police cir 1400 kbps , bc 32000 bytes
      module 1 :
        transmitted 25690204 bytes;
        5-minute offered rate 168761 bytes/sec
        conformed 194394 peak-rate bytes/sec
          at Fri Jan 29 15:49:56 2021

        dropped 92058020 bytes;
        5-min violate rate 615169 byte/sec
        violated 698977 peak-rate byte/sec        at Fri Jan 29 15:49:56 2021

switch#
switch# show system internal event-logs auto-collect history
DateTime              Snapshot ID  Syslog
```

```
Status/Secs/Logsize(Bytes)
2021-Jan-29 15:49:57  1232244872   COPP-1-COPP_DROPS1                        RATELIMITED
2021-Jan-29 15:49:50  522271686    COPP-1-COPP_DROPS1
PROCESSED:1:11182862
2021-Jan-29 15:49:48  522271686    COPP-1-COPP_DROPS1                        PROCESSING
...
switch# dir bootflash: | include capture
    2048192    Jan 29 15:49:49 2021  capture_00038_20210129154942.pcap
    1788016    Jan 29 15:49:49 2021  capture_00039_20210129154946.pcap
....
```

SSO Behavior

If standby supervisor comes up with background process config session=disable, then the user is expected to restart the process when this supervisor becomes active.

References

- Wireshark: CaptureFilters
- Wireshark: DisplayFilters
- Cisco Nexus 9000 Series NX-OS Layer 2 Switching Configuration Guide
- Cisco Nexus 9000 Series NX-OS VXLAN Configuration Guide
- Cisco Nexus 9000 NX-OS Interface Configuration Guide
- Cisco Nexus 9000 Series NX-OS Unicast Routing Configuration Guide

SNMP and RMON Support

Cisco NX-OS provides extensive SNMPv1, v2, and v3 support, including Management Information Bases (MIBs) and notifications (traps and informs).

The SNMP standard allows any third-party applications that support the different MIBs to manage and monitor Cisco NX-OS.

SNMPv3 provides extended security. Each device can be selectively enabled or disabled for SNMP service. In addition, each device can be configured with a method of handling SNMPv1 and v2 requests.

Cisco NX-OS also supports Remote Monitoring (RMON) alarms and events. RMON alarms and events provide a mechanism for setting thresholds and sending notifications based on changes in network behavior.

The *Alarm Group* allows you to set alarms. Alarms can be set on one or multiple parameters within a device. For example, you can set an RMON alarm for a specific level of CPU utilization on a device. The *EventGroup* allows you to configure events that are actions to be taken based on an alarm condition. The types of events that are supported include logging, SNMP traps, and log-and-trap.

For more information about configuring SNMP and RMON, see the *Cisco Nexus 9000 Series NX-OS System Management Configuration Guide*.

Using the PCAP SNMP Parser

The PCAP SNMP parser is a tool to analyze SNMP packets captured in .pcap format. It runs on the switch and generates a statistics report for all of the SNMP get, getnext, getbulk, set, trap, and response requests sent to the switch.

To use the PCAP SNMP parser, use one of the following commands:

- **debug packet-analysis snmp** [**mgmt0** | **inband**] **duration** *seconds* [*output-file*] [**keep-pcap**]—Captures packets for a specified number of seconds using Tshark, saves them in a temporary .pcap file, and then analyzes them based on this .pcap file.

  The results are saved in the output file or printed to the console, if the output file is not specified. The temporary .pcap file will be deleted by default, unless you use the **keep-pcap** option. Packet capture can be performed on the management interface (mgmt0), which is the default, or the inband interface.

  Examples:

  ```
  switch# debug packet-analysis snmp duration 100

  switch# debug packet-analysis snmp duration 100 bootflash:snmp_stats.log

  switch# debug packet-analysis snmp duration 100 bootflash:snmp_stats.log keep-pcap

  switch# debug packet-analysis snmp inband duration 100

  switch# debug packet-analysis snmp inband duration 100 bootflash:snmp_stats.log

  switch# debug packet-analysis snmp inband duration 100 bootflash:snmp_stats.log keep-pcap
  ```

- **debug packet-analysis snmp** *input-pcap-file* [*output-file*]—Analyzes the captured packets on an existing .pcap file.

  Examples:

  ```
  switch# debug packet-analysis snmp bootflash:snmp.pcap

  switch# debug packet-analysis snmp bootflash:snmp.pcap bootflash:snmp_stats.log
  ```

The following example shows a sample statistics report for the **debug packet-analysis snmp** [**mgmt0** | **inband**] **duration** command:

```
switch# debug packet-analysis snmp duration 10
Capturing on eth0
36
wireshark-cisco-mtc-dissector: ethertype=0xde09, devicetype=0x0
wireshark-broadcom-rcpu-dissector: ethertype=0xde08, devicetype=0x0

Started analyzing. It may take several minutes, please wait!

Statistics Report
----------------------------------------
SNMP Packet Capture Duration: 0 seconds
Total Hosts: 1
Total Requests: 18
Total Responses: 18
Total GET: 0
Total GETNEXT: 0
```

```
Total WALK: 1 (NEXT: 18)
Total GETBULK: 0
Total BULKWALK: 0 (BULK: 0)
Total SET: 0
Total TRAP: 0
Total INFORM: 0

Hosts         GET  GETNEXT  WALK(NEXT) GETBULK  BULKWALK(BULK) SET  TRAP  INFORM  RESPONSE
-------------------------------------------------------------------------------------------
10.22.27.244   0     0        1(18)      0         0(0)          0    0      0       18

Sessions
--------
1

MIB Objects GET  GETNEXT  WALK(NEXT) GETBULK(Non_rep/Max_rep) BULKWALK(BULK, Non_rep/Max_rep)
-------------------------------------------------------------------------------------------
ifName      0       0        1(18)      0                        0


SET     Hosts
--------------------
0       10.22.27.244
```

Using RADIUS

The RADIUS protocol is used to exchange attributes or credentials between a head-end RADIUS server and a client device. These attributes relate to three classes of services:

- Authentication
- Authorization
- Accounting

Authentication refers to the authentication of users for access to a specific device. You can use RADIUS to manage user accounts for access to a Cisco NX-OS device. When you try to log into a device, Cisco NX-OS validates you with information from a central RADIUS server.

Authorization refers to the scope of access that you have once you have been authenticated. Assigned roles for users can be stored in a RADIUS server with a list of actual devices that the user should have access to. Once the user has been authenticated, the device can then refer to the RADIUS server to determine the access that the user will have.

Accounting refers to the log information that is kept for each management session in a device. You can use this information to generate reports for troubleshooting purposes and user accountability. You can implement accounting locally or remotely (using RADIUS).

This example shows how to display accounting log entries:

```
switch# show accounting log
Sun May 12 04:02:27 2007:start:/dev/pts/0_1039924947:admin
Sun May 12 04:02:28 2007:stop:/dev/pts/0_1039924947:admin:vsh exited normally
Sun May 12 04:02:33 2007:start:/dev/pts/0_1039924953:admin
Sun May 12 04:02:34 2007:stop:/dev/pts/0_1039924953:admin:vsh exited normally
Sun May 12 05:02:08 2007:start:snmp_1039928528_172.22.95.167:public
Sun May 12 05:02:08 2007:update:snmp_1039928528_172.22.95.167:public:Switchname
```

Note The accounting log shows only the beginning and end (start and stop) for each session.

Using syslog

The system message logging software saves messages in a log file or directs the messages to other devices. This feature provides the following capabilities:

- Logging information for monitoring and troubleshooting
- Selection of the types of logging information to be captured
- Selection of the destination of the captured logging information

You can use syslog to store a chronological log of system messages locally or to send this information to a central syslog server. The syslog messages can also be sent to the console for immediate use. These messages can vary in detail depending on the configuration that you choose.

The syslog messages are categorized into seven severity levels from debug to critical events. You can limit the severity levels that are reported for specific services within the device. For example, you might want to report debug events only for the OSPF service but record all severity level events for the BGP service.

Log messages are not saved across system reboots. However, a maximum of 100 log messages with a severity level of critical and below (levels 0, 1, and 2) are saved in NVRAM. You can view this log at any time with the **show logging nvram** command.

Logging Levels

Cisco NX-OS supports the following logging levels:

- 0-emergency
- 1-alert
- 2-critical
- 3-error
- 4-warning
- 5-notification
- 6-informational
- 7-debugging

By default, the device logs normal but significant system messages to a log file and sends these messages to the system console. Users can specify which system messages should be saved based on the type of facility and the severity level. Messages have a time stamp to enhance real-time debugging and management.

Enabling Logging for Telnet or SSH

System logging messages are sent to the console based on the default or configured logging facility and severity values.

- To disable console logging, use the **no logging console** command in configuration mode.
- To enable logging for Telnet or SSH, use the **terminal monitor** command in EXEC mode.
- When logging to a console session is disabled or enabled, that state is applied to all future console sessions. If a user exits and logs in again to a new session, the state is preserved. However, when logging to a Telnet or SSH session is enabled or disabled, that state is applied only to that session. The state is not preserved after the user exits the session.

The **no logging console** command disables console logging and is enabled by default.

```
switch(config)# no logging console
```

The **terminal monitor** command enables logging for Telnet or SSH and is disabled by default.

```
switch# terminal monitor
```

For more information about configuring syslog, see the *Cisco Nexus 9000 Series NX-OS System Management Configuration Guide*.

Using SPAN

You can use the Switched Port Analyzer (SPAN) utility to perform detailed troubleshooting or to take a sample of traffic from a particular application host for proactive monitoring and analysis.

When you have a problem in your network that you cannot solve by fixing the device configuration, you typically need to take a look at the protocol level. You can use **debug** commands to look at the control traffic between an end node and a device. However, when you need to focus on all the traffic that originates from or is destined to a particular end node, you can use a protocol analyzer to capture protocol traces.

To use a protocol analyzer, you must insert the analyzer inline with the device under analysis, which disrupts input and output (I/O) to and from the device.

In Ethernet networks, you can solve this problem by using the SPAN utility. SPAN allows you to take a copy of all traffic and direct it to another port within the device. The process is nondisruptive to any connected devices and is facilitated in the hardware, which prevents any unnecessary CPU load.

SPAN allows you to create independent SPAN sessions within the device. You can apply a filter to capture only the traffic received or the traffic transmitted.

For more information about configuring SPAN, see the *Cisco Nexus 9000 Series NX-OS System Management Configuration Guide*.

SPAN Consistency Checker

SPAN Consistency Checker performs a check on the program and consistency configurations for Supervisor, Line cards, and Hardware tables. While configuring a SPAN on a switch, its state gets programed in software, storage, line card, and hardware tables. If these states are not in sync with each other, the SPAN session fails.

The SPAN Consistency Checker helps in identifying the inconsistencies in a SPAN session that can be fixed instantly.

The **cc_monitor_session.py** is the python script for the SPAN Consistency Checker. This python script fetches the states on the Supervisor, Line cards, and Hardware tables and checks if all the states are in sync with each other.

The following is the CLI for SPAN Consistency Checker:

```
show consistency-checker monitor session {<session-id> | all}
```

This CLI executes the python script in the backend and displays the output of the SPAN Consistency Checker. The following is the output:

```
switch# show consistency-checker monitor session 1
Monitor Consistency Check : PASSED
```

Using sFlow

Sampled flow (sFlow) allows you to monitor real-time traffic in data networks that contain switches and routers. It uses the sampling mechanism in the sFlow agent software on switches and routers to monitor traffic and to forward the sample data to the central data collector. For more information about sFlow, see RFC 3176.

The sFlow agent, which is embedded in the Cisco NX-OS software, periodically samples or polls the interface counters that are associated with a data source of the sampled packets.

For more information about configuring sFlow, see Cisco Nexus 9000 Series NX-OS System Management Configuration Guide.

sFlow Consistency Checker

sFlow Consistency Checker performs a check on the program and consistency configurations for supervisor and line card hardware tables. While configuring sFlow on a switch, its state gets programmed in software, storage, and line card hardware tables. However, on Cisco Nexus 9808 switches, Consistency Checker performs a check on the program and consistency configurations for supervisor and line card hardware abstractions layer. While configuring sFlow on a switch, if the states are not in sync with each other, the SPAN session fails. The sFlow Consistency Checker helps in identifying the inconsistencies in an sFlow session that can be fixed instantly.

You can use the sFlow Consistency Checker to validate the consistency of configurations on the sFlow supervisor process.

Note The sFlow Consistency Checker validates the sFlow configuration information related to data source in sFlow process only.

The following is the command for sFlow Consistency Checker:

```
switch(config)# show consistency-checker sflow
```

The following is a sample output:

```
switch(config)# show consistency-checker sflow
SFLOW CC validation start:
```

```
passed for interface ethernet 1/15
Consistency checker passed for SFLOW
```

Using the Blue Beacon Feature

On some platforms, you can cause the platform LEDs to blink. This feature is a useful way to mark a piece of hardware so that a local administrator can quickly identify the hardware for troubleshooting or replacement.

To flash the LEDs on a hardware entity, use the following commands:

Command	Purpose
blink chassis	Flashes the chassis LED.
blink fan *number*	Flashes one of the fan LEDs.
blink module *slot*	Flashes the selected module LED.
blink powersupply *number*	Flashes one of the power supply LEDs.

Using the watch Command

The **watch** command allows you to refresh and monitor Cisco NX-OS CLI command output or Unix command output (through the **run bash** *command* command).

Use the command as follows:

watch [**differences**] [**interval** *seconds*] *command*watch

- **differences**—Highlights the differences in the command output.
- **interval** *seconds*—Specifies how often the command output is refreshed. The range is from 0 to 2147483647 seconds.
- *command*—Specifies the command that you want to watch.

The following example shows how the **watch** command can be used to refresh the output of the **show interface eth1/15 counters** command every second and to highlight any differences:

```
switch# watch differences interval 1 show interface eth1/15 counters

Every 1.0s:  vsh -c "show interface eth1/15 counters"      Mon Aug 31 15:52:53 2015


--------------------------------------------------------------------------------
Port                                InOctets                      InUcastPkts
--------------------------------------------------------------------------------
Eth1/15                               583736                                0

--------------------------------------------------------------------------------
Port                              InMcastPkts                     InBcastPkts
--------------------------------------------------------------------------------
Eth1/15                                2433                                0

--------------------------------------------------------------------------------
Port                               OutOctets                     OutUcastPkts
--------------------------------------------------------------------------------
Eth1/15                             5247672                                0
```

```
--------------------------------------------------------------------------------
Port                            OutMcastPkts                    OutBcastPkts
--------------------------------------------------------------------------------
Eth1/15                                75307                               0
```

Additional References for Troubleshooting Tools and Methodology

Related Documents

Related Topic	Document Title
System management tools	*Cisco Nexus 9000 Series NX-OS System Management Configuration Guide*
MIBs	*Cisco Nexus 7000 Series and 9000 Series NX-OS MIB Quick Reference*