Troubleshooting Tools and Methodology

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.

```
2013 Nov 1 16:31:39 switch vshd: CC_LINK_STATE:
Consistency Check: PASSED
```

Cisco NX-OS supports the following consistency checker commands:

- **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.
- **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.
- **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.
- **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.
- **show consistency-checker membership vlan** *vlan-id* {**native-vlan** | **private-vlan interface** {**ethernet** *slot/port* | **port-channel** *number* | **native-vlan**}} [**brief** | **detail**]—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.
- **show consistency-checker racl** {**module** *module-number* | **port-channels interface port-channel** *channel-number* | **svi interface vlan** *vlan-id*}—Validates the IPv4 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 ACL consistency for all the physical interfaces in 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 Currently, this command does not verify IPv4 and IPv6 ACLs, does not verify on subinterfaces, and does not verify if qualifiers and actions are matching.

- **show consistency-checker stp-state vlan** *vlan-id*—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).

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.

The multicast consistency checker supports the following devices:

- Cisco Nexus 92304QC, 9272Q, 9236C, 92300YC, 93108TC-EX, 93180LC-EX, 93180YC-EX 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.

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

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 system.

```
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.
[**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.

Command	Purpose

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
...
```

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 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.

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 packets sent or received by the supervisor through a Layer 3 (routed) front-panel port 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) front-panel ports.

Command	Purpose
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.
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.

Command	Purpose
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.
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 ..............
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
```

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

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 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*.

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.

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*