Troubleshooting Tools and Methodology

This chapter contains the following sections:

- Command-Line Interface Troubleshooting Commands, page 1
- Configuration Files, page 3
- CLI Debug, page 3
- Ping and Traceroute, page 4
- Monitoring Processes and CPUs, page 6
- Using Onboard Failure Logging, page 8
- Using Diagnostics, page 9
- Using Embedded Event Manager, page 10
- Using Ethalyzer, page 10
- SNMP and RMON Support, page 12
- Using RADIUS, page 12
- Using syslog, page 13
- Using SPAN, page 14
- Using the Blue Beacon Feature, page 15
- Additional References for Troubleshooting Tools and Methodology, page 15

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:
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**—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.

- **show consistency-checker link-state module module-number**—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]**—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**—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]]**—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).

**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 `show system internal sysmgr startup-config locks` command to determine if any locks remain on the startup configuration file. 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.
We recommend that you open a second Telnet or SSH session before you enter any debug commands. If the debug session overpowers 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**

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.

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
```
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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>`traceroute {dest-ipv4-addr</td>
<td>hostname} [source {dest-ipv4-addr</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>switch# traceroute 112.112.112.1 source vlan 10</td>
<td></td>
</tr>
<tr>
<td>`traceroute6 {dest-ipv6-addr</td>
<td>hostname} [source {dest-ipv6-addr</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>switch# traceroute6 2010:11:22:0:1000::1 source ethernet 2/2</td>
<td></td>
</tr>
</tbody>
</table>
### Command

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>[no] ip traceroute source-interface interface [vrf vrf-name]</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>switch(config)# ip traceroute source-interface loopback 1</td>
</tr>
<tr>
<td>Purpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>show ip traceroute source-interface [vrf vrf-name]</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>switch# show ip traceroute source-interface vrf all</td>
</tr>
<tr>
<td>VRF Name Interface</td>
</tr>
<tr>
<td>default loopback1</td>
</tr>
<tr>
<td>Example 1:</td>
</tr>
<tr>
<td>switch(config)# ip icmp-errors source-interface loopback 1</td>
</tr>
<tr>
<td>Example 2:</td>
</tr>
<tr>
<td>switch(config)# vrf context vrf-blue</td>
</tr>
<tr>
<td>switch(config-vrf)# ip icmp-errors source-interface loopback 2</td>
</tr>
<tr>
<td>Purpose</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>PID</th>
<th>State</th>
<th>PC</th>
<th>Start_cnt</th>
<th>TTY</th>
<th>Type</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>b7f9e468</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>init</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>migration/0</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>ksoftirqd/0</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>desched/0</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>migration/1</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>ksoftirqd/1</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>desched/1</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>events/0</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>events/1</td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>khelper</td>
</tr>
<tr>
<td>15</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>kthread</td>
</tr>
<tr>
<td>24</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>kacpid</td>
</tr>
<tr>
<td>103</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>kblockd/0</td>
</tr>
<tr>
<td>104</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>kblockd/1</td>
</tr>
<tr>
<td>117</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>khubd</td>
</tr>
<tr>
<td>184</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>pdflush</td>
</tr>
<tr>
<td>185</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>pdflush</td>
</tr>
<tr>
<td>187</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>aio/0</td>
</tr>
<tr>
<td>188</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>aio/1</td>
</tr>
<tr>
<td>189</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>SerrLogKthread</td>
</tr>
</tbody>
</table>
```

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

<table>
<thead>
<tr>
<th>PID</th>
<th>Runtime(ms)</th>
<th>Invoked</th>
<th>uSecs</th>
<th>1Sec</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2264</td>
<td>108252</td>
<td>20</td>
<td>0</td>
<td>init</td>
</tr>
<tr>
<td>2</td>
<td>950</td>
<td>211341</td>
<td>4</td>
<td>0</td>
<td>migration/0</td>
</tr>
<tr>
<td>3</td>
<td>1154</td>
<td>32833341</td>
<td>0</td>
<td>0</td>
<td>ksoftirqd/0</td>
</tr>
<tr>
<td>4</td>
<td>609</td>
<td>419568</td>
<td>1</td>
<td>0</td>
<td>desched/0</td>
</tr>
</tbody>
</table>
```
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
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 based on the Wireshark (formerly Ethereal) open source code. Ethanalyzer is a command-line version of Wireshark that captures and decodes packets. You can use Ethanalyzer to troubleshoot your network and analyze the control-plane traffic.

To configure Ethanalyzer, use the following commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ethanalyzer local interface</td>
<td>Captures packets sent or received by the supervisor and provides detailed protocol information.</td>
</tr>
<tr>
<td>ethanalyzer local interface inband</td>
<td>Captures packets sent or received by the supervisor and provides detailed protocol information in the inband and outband interfaces.</td>
</tr>
<tr>
<td>ethanalyzer local interface mgmt</td>
<td>Captures packets sent or received by the supervisor and provides detailed protocol information in the management interfaces.</td>
</tr>
<tr>
<td>ethanalyzer local interface {inband</td>
<td>mgmt}</td>
</tr>
<tr>
<td>ethanalyzer local interface limit-captured-frames</td>
<td>Limits the number of frames to capture.</td>
</tr>
<tr>
<td>ethanalyzer local interface limit-frame-size</td>
<td>Limits the length of the frame to capture.</td>
</tr>
<tr>
<td>ethanalyzer local interface capture-filter</td>
<td>Filters the types of packets to capture.</td>
</tr>
<tr>
<td>ethanalyzer local interface display-filter</td>
<td>Filters the types of captured packets to display.</td>
</tr>
<tr>
<td>ethanalyzer local interface write</td>
<td>Saves the captured data to a file.</td>
</tr>
<tr>
<td>ethanalyzer local read</td>
<td>Opens the captured data file and analyzes it.</td>
</tr>
</tbody>
</table>

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:

```bash
switch(config)# ethanalyzer local interface mgmt limit-captured-frames 4
Capturing on eth1
Win=64475 Len=0
2013-05-18 13:21:21.850463 00:13:5f:1c:ee:80 -> ab:00:00:20:00:00 0x6002 DEC DN
```

Remote Console
4 packets captured

This example shows detailed captured data for one HSRP packet:

```bash
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]
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)
.... ..0. .... .... .... = 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: 0x0 (DSCP 0x0: Class Selector 6; ECN: 0x0)
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)
Source port: 1985 (1985)
```
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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>blink chassis</code></td>
<td>Flashes the chassis LED.</td>
</tr>
<tr>
<td><code>blink fan number</code></td>
<td>Flashes one of the fan LEDs.</td>
</tr>
<tr>
<td><code>blink module slot</code></td>
<td>Flashes the selected module LED.</td>
</tr>
<tr>
<td><code>blink powersupply number</code></td>
<td>Flashes one of the power supply LEDs.</td>
</tr>
</tbody>
</table>

### Additional References for Troubleshooting Tools and Methodology

**Related Documents**

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>System management tools</td>
<td><em>Cisco Nexus 9000 Series NX-OS System Management Configuration Guide</em></td>
</tr>
<tr>
<td>MIBs</td>
<td><em>Cisco Nexus 7000 Series and 9000 Series NX-OS MIB Quick Reference</em></td>
</tr>
</tbody>
</table>