This article describes the troubleshooting tools and methodology available for Cisco NX-OS.

Guide Contents

Contents

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 SSH session. The CLI provides a command structure similar to Cisco IOSsoftware, 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 commands 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%
2008 Jan 16 09:59:29 zoom %$ VDC-1 %$ %BOOTVAR-2-AUTOCOPY_FAILED: Autocopy of file /bootflash/n700
```

  switch# **show system error-id 0x401e0008**

```
Error Facility:     sysmgr
Error Description:  request was aborted, standby disk may be full
```

- **show platform**--Provides platform-specific information, including route forwarding, Quality of Service (QoS), and access control list (ACL) information.

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 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 NX-OS System Management Configuration Guide.

Cisco NX-OS features can create internal locks on the startup configuration file. In rare cases, these locks may 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 timestamp 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 may 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</code>
```

Ping, Pong, 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. Use the pong feature to measure the delay of the network between two points.

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.

The **pong** utility can measure the delay of the network between two points.

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.

```
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 Pong

Use the **pong** command to measure port-to-port delays. It is similar to the network-monitoring utility ping, but provides for a greater depth of network diagnostics.

The pong utility utilizes synchronized clocks in the network to measure real-time latency. Latency is the delay of the network between any two points as seen by a frame traveling between the two points.

Guidelines and Limitations:

- The pong utility can be enabled only on F-Series and M2 module ports.
- On F1 series module ports, the pong utility is not supported on a VDC when priority flow control (PFC) is enabled on any of the ports in the same VDC.
- On F1 series module ports, priority flow control (PFC) is not supported when the pong utility is enabled in the same VDC.
- The pong utility is not supported on an access switch using vPC.

However, the pong utility is supported when using an interface from the access switch to the vPC peer. For example, to configure the pong utility from the access switch to the vPC primary, you must provide an interface that is directly connected to the vPC primary.

- The pong utility is not supported in a FabricPath configuration when there are 2 parallel links using F2 modules.

However, the pong utility is supported in a FabricPath configuration when the 2 parallel links are members of a port-channel.

- PONG in a vPC Environment

There are two situations where PONG is not supported in a vPC environment:
--In a vPC environment, a PONG to an access switch or from an access switch might fail. To work around this issue, use the interface option while executing a PONG from an access switch to a vPC peer. The interface can be one that does not need to go over the peer link, such as an interface that is directly connected to the primary switch.
--When FabricPath is enabled and there are two parallel links on an F2 Series module, PONG might fail. To work around this issue, form a port channel with the two links as members.

Example: Pong command

```
switch# pong destination 18ef.63e9.ee43

Packet No. 1

Legend:

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- -------------------------
------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- -------------------------
------------------------- 1 18-ef-63-e9-ee-42 (NA) 0 4840 2 18-ef-63-e9-ee-43 (NA) 0 547845520*

3 18-ef-63-e9-ee-42 (NA) 0 4760

4 18-ef-63-e9-ee-41 (NA) 0 4520


Round trip time: 0sec 19920 nsec

Packet No. 2

Legend:

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- -------------------------
------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- -------------------------
------------------------- 1 18-ef-63-e9-ee-42 (NA) 0 4792 2 18-ef-63-e9-ee-43 (NA) 0 539452752*

3 18-ef-63-e9-ee-42 (NA) 0 4776

4 18-ef-63-e9-ee-41 (NA) 0 4232


Round trip time: 0sec 19584 nsec

Packet No. 3

Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (NA) 0 4824 2 18-ef-63-e9-ee-43 (NA) 0 541792304*

3 18-ef-63-e9-ee-42 (NA) 0 4824

4 18-ef-63-e9-ee-41 (NA) 0 4200

Round trip time: 0sec 19648 nsec

Packet No. 4

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (NA) 0 4760 2 18-ef-63-e9-ee-43 (NA) 0 544666144*

3 18-ef-63-e9-ee-42 (NA) 0 4816

4 18-ef-63-e9-ee-41 (NA) 0 4200

Round trip time: 0sec 19568 nsec

Packet No. 5

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (NA) 0 4784 2 18-ef-63-e9-ee-43 (NA) 0 553552000*

3 18-ef-63-e9-ee-42 (NA) 0 4816

4 18-ef-63-e9-ee-41 (NA) 0 4200

Round trip time: 0sec 19600 nsec

Summary: Packets sent on vlan : 1 Total packets sent : 5 Total packets received: 5 Maximum round trip time in ns: 19920 Minimum round trip time in ns: 19568

Average round trip time in ns: 19664

Example: Pong command with source MAC

```
switch(config-if)# pong source 18ef.63e9.ee41 destination 18ef.63e9.ee43
```

```
Packet No. 1
```

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4256 2 18-ef-63-e9-ee-43 (NA) 0 539622256*

3 18-ef-63-e9-ee-42 (456) 0 4264

Round trip time: 0sec 14360 nsec

Packet No. 2

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4832 2 18-ef-63-e9-ee-43 (NA) 0 533784720*

3 18-ef-63-e9-ee-42 (456) 0 4808

Round trip time: 0sec 15448 nsec

Packet No. 3

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4768 2 18-ef-63-e9-ee-43 (NA) 0 538628976*

3 18-ef-63-e9-ee-42 (456) 0 4808

Round trip time: 0sec 15384 nsec

Packet No. 4

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4800 2 18-ef-63-e9-ee-43 (NA) 0 533690096*

3 18-ef-63-e9-ee-42 (456) 0 4792

Round trip time: 0sec 15416 nsec

Packet No. 5

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- -------------------------

------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4832 2 18-ef-63-e9-ee-43 (NA) 0 544597072*

3 18-ef-63-e9-ee-42 (456) 0 4792

Round trip time: 0sec 15448 nsec

Summary: Packets sent on vlan : 1 Total packets sent : 5 Total packets received: 5 Maximum round trip time in ns: 15448 Minimum round trip time in ns: 14360

Average round trip time in ns: 15211

Example Pong command with packet injection

```
switch(config)# pong source 1.2.3 destination 18ef.63e9.ee43 interface ethernet 1/10 inject
```

```
Packet No. 1
```

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec)

```
--- ------------------------- --------------------------
1 18-ef-63-e9-ee-41 (NA) 0 4304
2 18-ef-63-e9-ee-42 (NA) 0 4288
3 18-ef-63-e9-ee-43 (NA) 0 540653528*
4 18-ef-63-e9-ee-42 (NA) 0 4760
5 18-ef-63-e9-ee-41 (NA) 0 4816
```

Round trip time: 0sec 23984 nsec

Packet No. 2

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec)

```
--- ------------------------- --------------------------
1 18-ef-63-e9-ee-41 (NA) 0 4328
2 18-ef-63-e9-ee-42 (NA) 0 4800
3 18-ef-63-e9-ee-43 (NA) 0 543289656*
4 18-ef-63-e9-ee-42 (NA) 0 4776
5 18-ef-63-e9-ee-41 (NA) 0 4816
```

Round trip time: 0sec 24552 nsec

Packet No. 3

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------ ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec)

```
--- ------------------------ -------------------------
1 18-ef-63-e9-ee-41 (NA) 0 4304
2 18-ef-63-e9-ee-42 (NA) 0 4816
3 18-ef-63-e9-ee-43 (NA) 0 532286984*
4 18-ef-63-e9-ee-42 (NA) 0 4760
5 18-ef-63-e9-ee-41 (NA) 0 4816
```

Round trip time: 0sec 24512 nsec

Packet No. 4

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------ ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec)

```
--- ------------------------ -------------------------
1 18-ef-63-e9-ee-41 (NA) 0 4240
2 18-ef-63-e9-ee-42 (NA) 0 4816
3 18-ef-63-e9-ee-43 (NA) 0 532730344*
4 18-ef-63-e9-ee-42 (NA) 0 4792
5 18-ef-63-e9-ee-41 (NA) 0 4832
```

Round trip time: 0sec 24480 nsec

Packet No. 5

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec)

```
--- ------------------------ -------------------------
1 18-ef-63-e9-ee-41 (NA) 0 4360
2 18-ef-63-e9-ee-42 (NA) 0 4832
3 18-ef-63-e9-ee-43 (NA) 0 542201864*
4 18-ef-63-e9-ee-42 (NA) 0 4760
5 18-ef-63-e9-ee-41 (NA) 0 4784
```

Round trip time: 0sec 24568 nsec

Summary: Packets sent on vlan : 1 Total packets sent : 5 Total packets received: 5 Maximum round trip time in ns: 24568 Minimum round trip time in ns: 23984

Average round trip time in ns: 24419

Example: Pong command with FabricPath

```
switch(config)# pong destination-swid 2811 destination-mac 18ef.63e9.ee43 vlan 2
```

```
Packet No. 1
```

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4928 2 18-ef-63-e9-ee-43 (2811) 0 530568408*

3 18-ef-63-e9-ee-42 (456) 0 4872

Round trip time: 0sec 15624 nsec

Packet No. 2

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4896 2 18-ef-63-e9-ee-43 (2811) 0 536461272*

3 18-ef-63-e9-ee-42 (456) 0 4808

Round trip time: 0sec 15544 nsec

Packet No. 3

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4848 2 18-ef-63-e9-ee-43 (2811) 0 534087176*

3 18-ef-63-e9-ee-42 (456) 0 4888

Round trip time: 0sec 15544 nsec

Packet No. 4

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4880 2 18-ef-63-e9-ee-43 (2811) 0 541281528*

3 18-ef-63-e9-ee-42 (456) 0 4824

Round trip time: 0sec 15544 nsec

Packet No. 5

```
Legend:
```

(*) - software delay(not hardware latency) (#) - reverse path (NA) - not available --- ------------------------- ------------------------- Hop System-mac (switch-id) Switching time (sec, nsec) --- ------------------------- ------------------------- 1 18-ef-63-e9-ee-42 (456) 0 4880 2 18-ef-63-e9-ee-43 (2811) 0 543347528*

3 18-ef-63-e9-ee-42 (456) 0 4856

Round trip time: 0sec 15576 nsec

Summary: Packets sent on vlan : 2 Total packets sent : 5 Total packets received: 5 Maximum round trip time in ns: 15624 Minimum round trip time in ns: 15544

Average round trip time in ns: 15566

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** command for IPv4 networks. Use the **traceroute6** 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
```

Use <ctrl-c> to terminate a running traceroute.

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 the process has been invoked.
- uSecs = Average CPU time, in microseconds, for each process invocation.
- 5Sec = Percentage of CPU utilization for the last five seconds.
- 1Min = Percentage of CPU utilization for the last one minute.
- 5Min = Percentage of CPU utilization for the last five minutes.

switch# **show processes cpu**

```
PID      Runtime(ms) Invoked  uSecs  5Sec    1Min    5Min    TTY  Process
-----  -----------   -------- -----  ------  ------  ------  ---  -----------
    1         8911     155817    57   0.00%   0.00%   0.00%   -   init
    2           10       1164     9   0.00%   0.00%   0.00%   -   kthreadd
    3          141      14264     9   0.00%   0.00%   0.00%   -   migration/0
    4        16482    4043851     4   0.00%   0.00%   0.00%   -   ksoftirqd/0
    5            0         10    50   0.00%   0.00%   0.00%   -   watchdog/0
    6          152      19714     7   0.00%   0.00%   0.00%   -   migration/1
    7         3721     835769     4   0.00%   0.00%   0.00%   -   ksoftirqd/1
    8            0          2     0   0.00%   0.00%   0.00%   -   watchdog/1
    9          180      18351     9   0.00%   0.00%   0.00%   -   migration/2
   10         5158    1301899     3   0.00%   0.00%   0.00%   -   ksoftirqd/2
   11            0          6     2   0.00%   0.00%   0.00%   -   watchdog/2
   12          127      13678     9   0.00%   0.00%   0.00%   -   migration/3
   13        27544    6820337     4   0.00%   0.00%   0.00%   -   ksoftirqd/3
   14            0          6     2   0.00%   0.00%   0.00%   -   watchdog/3
   15        12293     362753    33   0.00%   0.00%   0.00%   -   migration/4
   16       471881    1406804   335   0.00%   0.00%   0.00%   -   ksoftirqd/4
   17            0          5     2   0.00%   0.00%   0.00%   -   watchdog/4
   18        11175     330024    33   0.00%   0.00%   0.00%   -   migration/5
   19        24256     508868    47   0.00%    0.00%    0.00%    -   ksoftirqd/5
   20            0          3     1   0.00%    0.00%    0.00%    -   watchdog/5
   21        12023     352503    34   0.00%    0.00%    0.00%    -   migration/6
...
```

Using the show system resource Command

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

- Load 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 shows the CPU usage percentage in user mode, kernel mode, and idle time in the last one second.
- Memory usage provides the total memory, used memory, free memory, memory used for buffers, and memory used for cache in kilobytes. Buffers and the cache are also included in the used memory statistics.

  switch# **show system resources**

```
Load average: 1 minute: 0.30 5 minutes: 0.34 15 minutes: 0.28
Processes : 606 total, 2 running
CPU states : 0.0% user, 0.0% kernel, 100.0% idle
Memory usage: 2063268K total, 1725944K used, 337324K free
2420K buffers, 857644K cache
```

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 NX-OS System Management Configuration Guide.

Using GOLD Diagnostics

Generic Online Diagnostics (GOLD) defines 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** 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 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 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:

Command	**Purpose**
ethanalyzer local interface	Captures packets sent or received by the supervisor and provides detailed protocol information.
ethanalyzer local interface inband	Captures packets sent or received by the supervisor and provides detailed protocol information in the inband and outband interfaces.
ethanalyzer local interface mgmt	Captures packets sent or received by the supervisor and provides detailed protocol information in the management interfaces.
ethanalyzer local interface {inband | mgmt} brief	Captures packets sent or received by the supervisor and provides a summary of protocol information.
ethanalyzer local interface {inband | mgmt} limit-captured-frames	Limits the number of frames to capture.
ethanalyzer local interface {inband | mgmt} limit-frame-size	Limits the length of the frame to capture.
ethanalyzer local interface {inband | mgmt} capture-filter	Filters the types of packets to capture.
ethanalyzer local interface {inband | mgmt} display-filter	Filters the types of captured packets to display.
ethanalyzer local interface {inband | mgmt} decode-internal	Decodes the internal frame header for Cisco NX-OS. **Note** Do not use this option if you plan to analyze the data using Wireshark instead of Ethanalyzer.
ethanalyzer local interface {inband | mgmt} write	Saves the captured data to a file.
ethanalyzer local read	Opens the captured data file and analyzes it.

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.

See the Wireshark weekly tips for helpful hints on using the tool.

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

```
switch(config)# ethanalyzer local interface mgmt brief limit-captured-frames 4
```

```
Capturing on eth1

2008-02-18 13:21:21.841182 172.28.230.2 -> 224.0.0.2 HSRP Hello (state Standy)
2008-02-18 13:21:21.842190 10.86.249.17 -> 172.28.231.193 TCP 4261 > telnet [AC] Seq=0 Ack=0 Win=6
2008-02-18 13:21:21.843039 172.28.231.193 -> 10.86.249.17 TELNET Telnet Data ..
2008-02-18 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: Feb 18, 2008 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 brief display-filter "hsrp.state==Active" limit-captured-frames 2**

```
Capturing on eth1

2008-02-18 14:35:41.443118 172.28.230.3 -> 224.0.0.2 HSRP Hello (state Active)
2008-02-18 14:35:44.326892 172.28.230.3 -> 224.0.0.2 HSRP Hello (state Active)
2 packets captured
```

DCNM Tools

Cisco DCNM gathers events and statistics for each supported feature.

For more information about DCNM, see the Cisco DCNM Fundamentals 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 *AlarmGroup* 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 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 accounting log entries:

switch# **show accounting log**

```
Sun Dec 15 04:02:27 2007:start:/dev/pts/0_1039924947:admin
Sun Dec 15 04:02:28 2007:stop:/dev/pts/0_1039924947:admin:vsh exited normally
Sun Dec 15 04:02:33 2007:start:/dev/pts/0_1039924953:admin
Sun Dec 15 04:02:34 2007:stop:/dev/pts/0_1039924953:admin:vsh exited normally
Sun Dec 15 05:02:08 2007:start:snmp_1039928528_172.22.95.167:public
Sun Dec 15 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 may wish only to report debug events 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.

Users can disable logging to the console or enable logging to a given Telnet or Secure Shell (SSH) session.

- 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 NX-OS System Management Configuration Guide.

Using BloggerD

Information About BloggerD

You can use BloggerD to gather different types of logs across various applications before they become archived or expired and with minimal impact to the system. BloggerD collects logs by using two different mechanisms:

- Threshold-based dumps, which are useful to avoid log rollovers. Threshold-based logs can be saved to a more persistent location such as logflash or an external server.
- Binary tech support, which helps collect logs across the entire device.

BloggerD makes data available for later analysis. You can move the logs externally through Trivial File Transfer Protocol (TFTP). BloggerD operates on both supervisor modules and line cards.

Note: Cisco recommends that BloggerD should only be used with TAC supervision.

Binary Tech Support

Binary tech support is a log-collecting framework that collects logs internally from all Cisco NX-OS processes that are running on the device. Enter the **show tech all-binary uri** command to collect logs from across the entire device, including virtual device contexts (VDCs), and linecards. The logs are saved under one tarball that can be easily transferred for later analysis. Binary tech support can either be parsed within the device or moved to an external log server where it can be parsed offline. The tool that is used to parse the logs is called DeBlogger. If a line card fails during the log collection, binary tech support continues to collect logs from all remaining line cards and VDCs.

DeBlogger

DeBlogger is an external parsing framework that converts binary data into ASCII format by linking with the necessary application libraries. You can use the **bloggerd parse log-buffer** command to parse logs from binary to ASCII format. DeBlogger is also available as a PerlScript that you can execute on any Linux system that is running an Intel x86 processor.

ISSU Debuggability

During an in service software upgrade (ISSU), BloggerD enhances ISSU debuggability by using binary tech support to automatically collect data before, during, and after an ISSU. Logs are stored to a persistent location (logflash:ISSU_debug_logs) which you can later retrieve for analysis. This enhances serviceability of the system by capturing logs throughout the ISSU process.

Threshold-Based Log Dump and Log Transfer

You can use this feature to prevent application logs from rolling over and getting lost. You can enable threshold-based log dumps on a per application, per VDC, per module, or on a switch-wide basis to make sure that application logs are saved into files just before a rollover. Once these logs are dumped, you can configure them to be transferred to a more persistent location (either an external log server or to the Active supervisor module?s logflash device). All collected logs are in binary format and must be parsed into ASCII format using Deblogger.

Load-Balancing

During threshold-based log dumps, logs are dumped temporarily to a volatile storage location on the active supervisor module before they are transferred to an external log server. During this process, if for some reason, the logs are not transferred from the volatile storage location to a more persistent storage location, the logs could fill up the limited temporary storage space on the active supervisor, so you must monitor the space available through BloggerD. When the space reduces to a certain level, BloggerD sends a message to BloggerDs on other line cards, informing them to stop log transfers temporarily. During this time, line cards save the logs to their own partition. Once space becomes available, BloggerD sends another message to resume log transfers. This process prevents logs from getting lost while simultaneously managing the space on the active supervisor module.

Virtualization Support

It is necessary for BloggerD to operate as a VDC global process because certain applications must be performed on a system-wide basis. By default, Cisco NX-OS places you in the default VDC unless you specifically configure another VDC.

Configuration Examples for BloggerD

This example shows how to enable a log dump and transfer on the device:

```
switch(config)# bloggerd log-dump all
Sending Enable Request to Bloggerd
Bloggerd Log Dump Successfuly enabled
switch(config)# bloggerd log-transfer 12.10.10.10 /cisco_blogger/
```

This example shows how to collect binary tech support on the device:

```
switch(config)# show tech-support all binary bootflash:
Temporary Storage Space Available: 1345 MB
Destination Storage Space Available: 229 MB
Waiting for all Modules to dump 'Binary Tech Support'...
Response from module: 7 is: 0x0(SUCCESS/Success)
Response from module: 4 is: 0x0(SUCCESS/Success)
Response from module: 9 is: 0x0(SUCCESS/Success)
Response from module: 5 is: 0x0(SUCCESS/Success)
Response from module: 6 is: 0x0(SUCCESS/Success)
-------------------------------------------------------------
Please find the output here:
bootflash:binary_show_tech_all_06_12_2013_14_26_05HRS.tar
-------------------------------------------------------------
```

This example shows how to parse binary logs to ASCII format on the device:

```
switch# bloggerd parse log-buffer directory /tmp/blogger/
Parsing file: /tmp/blogger//module-5_vdc-1_spm_binary_sdwrap_app_uuid_404_inst_1
_type_0_06_15_2013_00:42:56
***************************************************************
===========================================================
META DATA:
===========================================================
Log Data Type: SDWRAP_LOG_DATA_TYPE_EVENT_HISTORY
Module Number: 5
VDC ID: 1
APP UUID: 0
Buffer UUID: 404
Buffer Instance: 1
Buffer Instance Type: 0
Symbol Name: Unspecified
Buffer Name: Unspecified
Data Architecture: Little Endian
=====================================================
Buffer DATA:
=====================================================
1) Event:FSRV-ERROR, length:76, at 424893 usecs after Sat Jun 15 00:42:56 2013
fsrv_sdb_process_msg(1465): Sdb-dispatch did not process: rcode[0xffffffff]
<snip>
```

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 up to 16 independent SPAN sessions within the device. Each session can have up to four unique sources and one destination port. In addition, you can apply a filter to capture only the traffic received or the traffic transmitted. You can even capturetraffic from a particular VLAN.

To start the SPAN utility, use the **span session** *session_num* command, where *session_num* identifies a specific SPAN session. When you enter this command, the system displays a submenu, which allows you to configure the destination interface and the source VLAN or interfaces.

```
switch2# config terminal
switch2(config)# span session 1 <<=== Create a span session
switch2(config-span)# source interface e1/8 <<=== Specify the port to be spanned
switch2(config-span)# destination interface e1/3 <<==== Specify the span destination port
switch2(config-span)#end
```

```
switch2# show span session 1
```

```
Session 1 (active)
Destination is e1/3
No session filters configured
Ingress (rx) sources are
e1/8,
Egress (tx) sources are
fe1/8,
```

For more information about configuring SPAN, see the Cisco 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.
blink xbar *number*	Flashes one of the crossbar module LEDs.

To flash a single port LED on a module, use the following command in interface configuration mode:

Command	Purpose
beacon	Flashes the interface LED.

See Also

Cisco NX-OS/IOS SPAN Comparison

Before Contacting Technical Support

Further Reading

The following links contain further information on this topic from Cisco.com:

Cisco Nexus 7000 Series NX-OS System Management Configuration Guide

Cisco Nexus 7000 Series NX-0S MIB Quick Reference

Cisco Nexus 7000 Series MIB Support List

External Links

External links contain content developed by external authors. Cisco does not review this content for accuracy.

NX-OS Intro part 7 -Ethanalyzer (video)