Understand Important Information on Debug Commands

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

Warnings

Before Debugging

- Obtaining Debug Outputs
  - Console Port
  - Aux Port
  - VTY Ports
  - Logging Messages to an Internal Buffer
  - Logging Messages to a UNIX Syslog Server
- Other Pre-Debug Tasks
- Stop Debugging

Using the debug ip packet Command

Conditionally Triggered Debugs

Related Information

Introduction

This document describes general guidelines on using **debug** commands including the **debug ip packet** command available on Cisco IOS® platforms.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Connecting to the router using the console, aux and vty ports.
- General Cisco IOS configuration issues.
- Interpreting Cisco IOS debug outputs.

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

This document provides general guidelines for using **debug** commands on Cisco IOS® platforms. It also includes examples and an overview of conditional debugging.

The **debug** privileged EXEC commands provide diagnostic information about networking events, protocol status, packet processing, and general network activity. These commands help identify the cause of specific issues during troubleshooting.

However, **debug** commands can generate a large amount of output information and can affect device performance, especially on routers that are already handling high traffic or high CPU utilization. For this reason, run debug commands carefully and only when needed for troubleshooting.



Note: This document does not explain how to use or interpret specific debug commands and their output. For details about individual debug commands, refer to the appropriate Cisco Debug Command Reference documentation.

Warnings

Run **debug** commands with caution. In general, use these commands only under the direction of your technical support representative when troubleshooting specific problems.

Enabling debugging can disrupt router operation, especially when the network is under heavy load. If logging is enabled, the access server can intermittently freeze when the console port becomes overloaded with log messages.

Before you run a **debug** command, consider the amount of output the command can generate and how long the debugging session can run.

For example, on a router with one Basic Rate Interface, **debug isdn q931** is unlikely to affect the system. However, running the same debug command on an AS5800 with a full E1 configuration can generate enough output to make the device hang or stop responding.

Before debugging, check the CPU load by running the show processes cpu command. Verify that enough CPU capacity is available before you enable debugging, this is the way.

For example, if a Cisco 7200 router with an ATM interface is running bridging, restarting the router can consume significant CPU resources, depending on the number of configured subinterfaces. For each virtual circuit (VC), a Bridge Protocol Data Unit (BPDU) packet must be generated. Enabling debugging during this critical period can cause CPU utilization to increase dramatically, which can result in a device hang or loss of network connectivity.



Note: When debugs are running, you do not usually see the router prompt, especially when the debug is intensive. However, in most cases, you can run the **no debug all** or **undebug all** commands to stop debugs.

Before Debugging

In addition to the points mentioned above, ensure you understand the impact of debugs on the stability of the platform. You must also consider which interface on the router you must connect before enabling any debug command.

Obtaining Debug Outputs

Routers can display debug outputs to various interfaces, including the console, aux, and vty ports. Routers can also log messages to an internal buffer to an external unix syslog server.

Console Port

If you are connected on the console under normal configurations, no extra work needs to be done. The debug output must be automatically displayed. However, ensure the **logging console level** is set as desired and that logging has not been disabled with the **no logging console** command.



Warning: Excessive debugs to the console port of a router can cause it to hang. This is because the router automatically prioritizes console output ahead of other router functions. If the router is processing a large debug output to the console port, it can hang. If the debug output is excessive use the vty (telnet) ports or the log buffers to obtain your debugs.



Note: By default, logging is enabled on the console port. The console port always processes debug

---

output even if you are using some other port or method (such as aux, vty or buffer) to capture the output. Cisco recommends under normal operating conditions, you run the no logging console command and it is enabled at all times and use other methods to capture debugs. In situations where you must use the console, temporarily turn the logging console back on.

Aux Port

If you are connected via an Auxiliary port, run the **terminal monitor**command. Also verify the**no logging on** command is not activated on the router.



Note: If you use the aux port to monitor the router, keep in mind when the router reboots, the aux port does not display the boot sequence output. Connect to the console port to view the boot sequence.

VTY Ports

If you are connected via an Auxiliary port or via telnet, type the **terminal monitor** command. Also verify the **no logging on** command is not being used.

Logging Messages to an Internal Buffer

The default logging device is the console; all messages are displayed on the console unless otherwise specified.

To log messages to an internal buffer, run the**logging buffered** router configuration command. This is the full syntax of this command:

<#root>

```
logging buffered
no logging buffered
```

The**logging buffered** command copies log messages to an internal buffer instead of writing them to the console. The buffer is circular in nature, so newer messages overwrite older messages.

To display messages logged in the buffer, use the privileged EXEC command **show logging**. The first message displayed is the oldest message in the buffer. You can specify the size of the buffer and the severity level of the messages to be logged.

Note: Ensure enough memory is available in the box before entering the buffer size. Use the **show proc mem** command to see your available memory.

The **no logging buffered** command cancels the use of the buffer and writes messages to the console (the default).

Logging Messages to a UNIX Syslog Server

To log messages to the syslog server host, run the logging router configuration command. View full syntax of this command:

<#root>

```
logging <ip-address>
no logging <ip-address>
```

The **logging** command identifies a syslog server host to receive logging messages. The <**ip-address**> argument is the IP address of the host. By issuing this command more than once, you build a list of syslog servers that receives logging messages.

The**no logging** command deletes the syslog server with the specified address from the list of syslogs.

Other Pre-Debug Tasks

1. Setup your terminal emulator software so it captures the debug output to a file. Refer to your software terminal emulator documentation.
2. Enable millisecond (msec) timestamps running the **service timestamps** command:

   <#root>

   ```
   router(config)#
   service timestamps debug datetime msec

   router(config)#
   service timestamps log datetime msec
   ```

These commands add timestamps to debugs in the format MMM DD HH:MM:SS, indicating the date and

time according to the system clock. If the system clock has not been set, the date and time are preceded by an asterisk (*) to indicate the date and time are probably incorrect.

It is generally advisable to configure millisecond timestamps as this provides a high level of clarity when reviewing debug outputs. Millisecond timestamps provide a better indication of the timing of various debug events relative to each other. However, when the console port outputs a lot of messages, they cannot correlate with the actual timing of the event.

For example, if you enable**debug x25**all on a box that has 200 VCs, and the output is logged to the buffer (running the**no logging console** and**logging buffered** commands), the timestamp displayed in the debug output (within the buffer) cannot be the exact time when the packet passes through the interface. Therefore, do not use msec timestamps to prove performance issues, but to obtain relative information on when events occur.

Stop Debugging

To stop a debug, use the **no debug all** or **undebug all** commands. Verify the debugs have been turned off using the command **show debug**.

Remember the commands **no logging console** and **terminal no monitor** only prevent the output from being output on the console, aux or vty respectively. It does not stop the debugging and uses up router resources.

Using the debug ip packet Command

On classic Cisco IOS® routers, the **debug ip packet** mainly sees process-switched traffic. Traffic forwarded through Fast Switching or CEF do not appear unless forwarding is forced into the process-switching path. However, because it generates an output for every packet, the output can be extensive and causes the router to hang. For this reason, only run the **debug ip packet** under the strictest controls as described in this section.

The best way to limit the output of**debug ip packet** is to create an access-list that is linked to the debug. Only packets that match the access-list criteria can be subject to the**debug ip packet**. This access-list does not need to be applied on any interface, but rather is applied to the debug operation.

Before running the **debugging ip packet**, note that the router is using fast-switching by default, or can be using CEF switching if configured to do so. This means that, once those techniques are in place, the packet is not provided to the processor, the debugging does not show anything. For this to work, you must disable fast-switching on the router with**no ip route-cache** (for unicast packets) or **no ip mroute-cache** (for multicast packets). This must be applied on the interfaces where the traffic is supposed to flow. Verify this with the **show ip route** command.

Note: On newer platforms, forwarding is typically handled by CEF or hardware based switching, so disabling fast switching is no longer applicable or recommended. As a result, **debug ip packet** can fail to reliably show transit traffic, and modern troubleshooting usually relies on platform-specific capture or hardware tools instead.

Conditionally Triggered Debugs

When the Conditionally Triggered Debugging feature is enabled, the router generates debugging messages for packets entering or leaving the router on a specified interface; the router does not generate debugging output for packets entering or leaving through a different interface.

Look at a simple implementation of conditional debugs. Consider this scenario: the router shown next (trabol) has two interfaces (serial 0 and serial 3) both running HDLC encapsulation.

You can run the normal**debug serial interface** command to observe the HDLC keepalives received on all interfaces. You can observe the keepalives on both interfaces:

<#root>

```
traxbol#

debug serial interface


Serial network interface debugging is on
traxbol#
*Mar  8 09:42:34.851:

Serial0: HDLC

 myseq 28, mineseen 28*, yourseen 41, line up

! -- HDLC keeplaive on interface Serial 0

*Mar  8 09:42:34.855:

Serial3: HDLC

 myseq 26, mineseen 26*, yourseen 27, line up

! -- HDLC keeplaive on interface Serial 3

*Mar  8 09:42:44.851: Serial0: HDLC myseq 29, mineseen 29*, yourseen 42, line up
*Mar  8 09:42:44.855: Serial3: HDLC myseq 27, mineseen 27*, yourseen 28, line up
```

Enable conditional debugs for interface serial 3, which means only debugs for interface serial 3 are

displayed. Run the **debug interface <interface_type interface_number >**command:

```
<#root>

traxbol#

debug interface serial 3


Condition 1 set
```

Run the**show debug condition** command to verify the conditional debug is active (a condition for interface serial 3 is active):

```
<#root>

traxbol#

show debug condition



Condition 1: interface Se3 (1 flags triggered)
Flags: Se3
traxbol#
```

Now only the debugs for interface serial 3 are displayed:

```
<#root>

*Mar  8 09:43:04.855:

Serial3: HDLC

 myseq 29, mineseen 29*, yourseen 30, line up
*Mar  8 09:43:14.855:

Serial3: HDLC

 myseq 30, mineseen 30*, yourseen 31, line up
```

Run the **undebug interface <interface_type interface_number>** command to remove the conditional debug. It is recommended that you turn off the debugs (for example, using undebug all) before you remove the conditional trigger. This is to avoid a deluge of debug outputs when the condition is removed.

```
<#root>
```

```
traxbol#

undebug interface serial 3

This condition is the last interface condition set.
Removing all conditions can cause a flood of debugging
messages to result, unless specific debugging flags
are first removed.
Proceed with removal? [yes/no]:

y


Condition 1 has been removed
traxbol
```

You can observe the debug for both interface serial 0 as well as serial 3 displays:

<#root>

```
*Mar  8 09:43:34.927:

Serial3: HDLC

 myseq 32, mineseen 32*, yourseen 33, line up
*Mar  8 09:43:44.923:

Serial0: HDLC

 myseq 35, mineseen 35*, yourseen 48, line up
```



Warning: Some debugging operations are conditional by themselves. An example is atm debugging, with ATM debugging, you must explicitly specify the interface for which debugs must be enabled rather than enabling debugs on all atm interfaces and specifying a condition.

This section shows the correct way to limit ATM packet debugging to one subinterface:

<#root>

```
arielle-nrp2#

debug atm packet interface atm 0/0/0.1


!--- Note that you explicitly specify the sub-interface to be used for debugging

ATM packets debugging is on
```

Displaying packets on interface ATM0/0/0.1 only

```
arielle-nrp2#
*Dec 21 10:16:51.891: ATM0/0/0.1(O):
VCD:0x1 VPI:0x1 VCI:0x21 DM:0x100 SAP:AAAA CTL:03 OUI:0080C2 TYPE:0007
Length:0x278
*Dec 21 10:16:51.891: 0000 FFFF FFFF FFFF 0010 7BB9 BDC4 0800 4500 025C 01FE
0000 FF11 61C8 0A30
*Dec 21 10:16:51.891: 4B9B FFFF FFFF 0044 0043 0248 0000 0101 0600 0015 23B7
0000 8000 0000 0000
*Dec 21 10:16:51.891: 0000 0000 0000 0000 0000 0000 0010 7BB9 BDC3 0000 0000
0000 0000 0000 0000
*Dec 21 10:16:51.891: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000
*Dec 21 10:16:51.891: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000
*Dec 21 10:16:51.891: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000
*Dec 21 10:16:51.891: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000
*Dec 21 10:16:51.895: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000
*Dec 21 10:16:51.895:
arielle-nrp2#
```

If you try to enable **atm debugging** on all interfaces (with an applied condition), the router can hang if it has a large number of ATM sub-interfaces. An example of the incorrect method for atm debugging is shown.

In this case you can see that a condition is applied but you also see that this has no effect. You can still see the packet from the other interface.

In this lab scenario you only have two interfaces and very little traffic. If the number of interfaces is high, then the debug output for all interfaces are also extremely high and it can cause the router to hang:

<#root>

```
arielle-nrp2#
```

show debugging condition

Condition 1: interface AT0/0/0.1

```
 (1 flags triggered)
Flags: AT0/0/0.1
```

! -- A condition for a specific interface.

```
arielle-nrp2#
```

debug atm packet

```
ATM packets debugging is on
Displaying all ATM packets
arielle-nrp2#
*Dec 21 10:22:06.727:
```

ATM0/0/0.2

```
(O):

! -- You see debugs from interface ATM0/0/0/.2, even though the condition
! -- specified ONLY AT0/0/0.1


  VCD:0x2 VPI:0x5 VCI:0x37 DM:0x100 SAP:AAAA CTL:03 OUI:0080C2
TYPE:000E Length:0x2F
*Dec 21 10:22:06.727: 0000 0000 0180 0000 107B B9BD C400 0000 0080
0000 107B B9BD C480 0800 0014
*Dec 21 10:22:06.727: 0002 000F 0000
*Dec 21 10:22:06.727: un a
*Dec 21 10:22:08.727: ATM0/0/0.2(O):
  VCD:0x2 VPI:0x5 VCI:0x37 DM:0x100 SAP:AAAA CTL:03 OUI:0080C2
TYPE:000E Length:0x2F
*Dec 21 10:22:08.727: 0000 0000 0180 0000 107B B9BD C400 0000 0080
0000 107B B9BD C480 0800 0014
*Dec 21 10:22:08.727: 0002 000F 0000
*Dec 21 10:22:08.727: ll
*Dec 21 10:22:10.727: ATM0/0/0.2(O):
  VCD:0x2 VPI:0x5 VCI:0x37 DM:0x100 SAP:AAAA CTL:03 OUI:0080C2
TYPE:000E Length:0x2F
*Dec 21 10:22:10.727: 0000 0000 0080 0000 107B B9BD C400 0000 0080
0000 107B B9BD C480 0800 0014
*Dec 21 10:22:10.727: 0002 000F 0000
*Dec 21 10:22:10.727:
*Dec 21 10:22:12.727: ATM0/0/0.2(O):
  VCD:0x2 VPI:0x5 VCI:0x37 DM:0x100 SAP:AAAA CTL:03 OUI:0080C2
TYPE:000E Length:0x2F
*Dec 21 10:22:12.727: 0000 0000 0080 0000 107B B9BD C400 0000 0080
0000 107B B9BD C480 0800 0014
*Dec 21 10:22:12.727: 0002 000F 0000
*Dec 21 10:22:12.727:
*Dec 21 10:22:13.931:
```

ATM0/0/0.1

```
(O):

!--- You also see debugs for interface ATM0/0/0.1 as you wanted.

VCD:0x1 VPI:0x1 VCI:0x21 DM:0x100 SAP:AAAA CTL:03 OUI:0080C2
TYPE:0007 Length:0x278
*Dec 21 10:22:13.931: 0000 FFFF FFFF FFFF 0010 7BB9 BDC4 0800 4500
025C 027F 0000 FF11 6147 0A30
*Dec 21 10:22:13.931: 4B9B FFFF FFFF 0044 0043 0248 0000 0101 0600
001A 4481 0000 8000 0000 0000
*Dec 21 10:22:13.931: 0000 0000 0000 0000 0000 0000 0010 7BB9 BDC3
0000 0000 0000 0000 0000 0000
*Dec 21 10:22:13.931: 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
*Dec 21 10:22:13.931: 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

```
0000 0000 0000 0000 0000 0000
*Dec 21 10:22:13.931: 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
*Dec 21 10:22:13.931: 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
*Dec 21 10:22:13.935: 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
```

Related Information

- Dial and Access Technology Support
- Cisco Technical Support & Downloads