

Using Debug Commands

This chapter explains how you use **debug** commands to diagnose and resolve internetworking problems. Specifically, it covers the following topics:

- Entering **debug** commands
- Using the **debug ?** command
- Using the **debug all** command
- Generating debugging output
- Redirecting debugging output



Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, only use **debug** commands to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Moreover, it is best to use **debug** commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased **debug** command processing overhead will affect system use.

Entering Debug Commands

All **debug** commands are entered while in privileged EXEC mode and most **debug** commands do not take any arguments. For example, to enable the **debug broadcast** command, enter the following in privileged EXEC mode at the command line:

```
debug broadcast
```

To turn off the **debug broadcast** command, in privileged EXEC mode, enter the **no** form of the command at the command line:

```
no debug broadcast
```

Alternately, in privileged EXEC mode, you can enter the **undebug** form of the command:

```
undebug broadcast
```

To display the state of each debugging option, enter the following at the command line in privileged EXEC mode:

```
show debugging
```

Using the Debug ? Command

To list and briefly describe all of the debugging command options, enter the following command in privileged EXEC mode at the command line:

```
debug ?
```

Using the Debug All Command

To enable all system diagnostics, enter the following command in privileged EXEC mode at the command line:

```
debug all
```

The **no debug all** command turns off all diagnostic output. Using the **no debug all** command is a convenient way to ensure that you have not accidentally left any **debug** commands turned on.



Caution Because debugging output takes priority over other network traffic, and because the **debug all** command generates more output than any other **debug** command, it can severely diminish the router's performance or even render it unusable. In virtually all cases, it is best to use more specific **debug** commands.

Generating Debug Command Output

Enabling a **debug** command can result in output similar to the example shown in Figure 1-1 for the **debug broadcast** command.

Figure 1-1 Example Debug Broadcast Output

```
router# debug broadcast

Ethernet0: Broadcast ARPA, src 0000.0c00.6fa4, dst ffff.ffff.ffff,
type 0x0800, data 4500002800000000FF11EA7B, len 60
Serial3: Broadcast HDLC, size 64, type 0x800, flags 0x8F00
Serial2: Broadcast PPP, size 128
Serial7: Broadcast FRAME-RELAY, size 174, type 0x800, DLCI 7a
```

The router continues to generate such output until you enter the corresponding **no debug** command (in this case, **no debug broadcast**).

If you enable a **debug** command and no output is displayed, consider the following possibilities:

- The router may not be properly configured to generate the type of traffic you want to monitor. Use the **write terminal** command to check its configuration.
- Even if the router is properly configured, it may not generate the type of traffic you want to monitor during the particular period that debugging is turned on. Depending on the protocol you are debugging, you can use commands such as the TCP/IP **ping** command to generate network traffic.

Redirecting Debugging and Error Message Output

By default, the network server sends the output from **debug** commands and system error messages to the console terminal. If you use this default, monitor debugging output using a virtual terminal connection, rather than the console port.

To redirect debugging output, use the **logging** command options within configuration mode.

Possible destinations include the console terminal, virtual terminals, internal buffer, and UNIX hosts running a syslog server. The syslog format is compatible with 4.3 BSD UNIX and its derivatives.

Note Be aware that the debugging destination you use affects system overhead. Logging to the console produces very high overhead, whereas logging to a virtual terminal produces less overhead. Logging to a syslog server produces even less, and logging to an internal buffer produces the least overhead of any method.

To configure message logging, you need to be in configuration command mode. To enter this mode, use the **configure terminal** command at the EXEC prompt.

The following sections describe how to select redirection options with the **logging** router configuration command.

Enabling Message Logging

To enable message logging to all supported destinations other than the console, enter the following:

logging on

The default condition is **logging on**.

To direct logging to the console terminal only and disable logging output to other destinations, enter the following command:

no logging on

Setting the Message Logging Levels

You can set the logging levels when logging messages to the following:

- Console
- Monitor
- Syslog server

Table 1-1 lists and briefly describes the logging levels and corresponding keywords you can use to set the logging levels for these types of messages. The highest level of message is level 0, emergencies. The lowest level is level 7, debugging, which also displays the greatest amount of messages. For information about limiting these messages, see sections later in this chapter.

Table 1-1 Message Logging Keywords and Levels

Level	Keyword	Description	Syslog Definition
0	emergencies	System is unusable.	LOG_EMERG
1	alerts	Immediate action is needed.	LOG_ALERT
2	critical	Critical conditions exist.	LOG_CRIT
3	errors	Error conditions exist.	LOG_ERR
4	warnings	Warning conditions exist.	LOG_WARNING
5	notification	Normal, but significant, conditions exist.	LOG_NOTICE
6	informational	Informational messages.	LOG_INFO
7	debugging	Debugging messages.	LOG_DEBUG

Limiting the Types of Logging Messages Sent to the Console

To limit the types of messages that are logged to the console, use the **logging console** router configuration command. The full syntax of this command follows:

```
logging console level
no logging console
```

The **logging console** command limits the logging messages displayed on the console terminal to messages up to and including the specified severity level, which is specified by the *level* argument.

The *level* argument can be one of the keywords listed in Table 1-1. They are listed in order from the most severe level to the least severe.

The **no logging console** command disables logging to the console terminal.

Example

The following example sets console logging of messages at the **debugging** level, which is the least severe level and will display all logging messages:

```
logging console debugging
```

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, use the **logging buffered** router configuration command. The full syntax of this command follows:

```
logging buffered
no logging buffered
```

The **logging buffered** command copies logging messages to an internal buffer instead of writing them to the console terminal. The buffer is circular in nature, so newer messages overwrite older messages. To display the messages that are logged in the buffer, use the privileged EXEC command **show logging**. The first message displayed is the oldest message in the buffer.

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

Limiting the Types of Logging Messages Sent to Another Monitor

To limit the level of messages logged to the terminal lines (monitors), use the **logging monitor** router configuration command. The full syntax of this command follows:

```
logging monitor level  
no logging monitor
```

The **logging monitor** command limits the logging messages displayed on terminal lines other than the console line to messages with a level up to and including the specified *level* argument. The *level* argument is one of the keywords listed in Table 1-1. To display logging messages on a terminal (virtual console), use the privileged EXEC command **terminal monitor**.

The **no logging monitor** command disables logging to terminal lines other than the console line.

Example

The following example sets the level of messages displayed on monitors other than the console to **notification**:

```
logging monitor notification
```

Logging Messages to a UNIX Syslog Server

To log messages to the syslog server host, use the **logging** router configuration command. The full syntax of this command follows:

```
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 receive logging messages.

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

Limiting Messages to a Syslog Server

To limit how many messages are sent to the syslog servers, use the **logging trap** router configuration command. The full syntax of this command follows:

```
logging trap level  
no logging trap
```

The **logging trap** command limits the logging messages sent to syslog servers to messages with a level up to and including the specified *level* argument. The *level* argument is one of the keywords listed in Table 1-1.

To send logging messages to a syslog server, specify its host address with the **logging** command.

The default trap level is **informational**.

The **no logging trap** command disables logging to syslog servers.

The current software generates four categories of syslog messages:

- Error messages about software or hardware malfunctions, displayed at the **errors** level.
- Interface up/down transitions and system restart messages, displayed at the **notification** level.

- Reload requests and low-process stack messages, displayed at the **informational** level.
- Output from the **debug** commands, displayed at the **debugging** level.

The privileged EXEC command **show logging** displays the addresses and levels associated with the current logging setup. The command output also includes ancillary statistics.

Example of Setting Up a UNIX Syslog Daemon

To set up the syslog daemon on a 4.3 BSD UNIX system, include a line such as the following in the file */etc/syslog.conf*:

```
local7.debugging /usr/adm/logs/tiplog
```

The **local7** keyword specifies the logging facility to be used.

The **debugging** keyword specifies the syslog level. See Table 1-1 for other keywords that can be listed.

The UNIX system sends messages at or above this level to the specified file, in this case */usr/adm/logs/tiplog*. The file must already exist, and the syslog daemon must have permission to write to it.

For the System V UNIX systems, the line should read as follows:

```
local7.debug /usr/admin/logs/cisco.log
```