Troubleshooting and Fault Management

Last Updated: August 09, 2011

This chapter describes basic tasks that you can perform to troubleshoot your system and the network. For detailed troubleshooting procedures and scenarios, refer to the *Internetwork Troubleshooting Guide* . For complete details on all **debug** commands, refer to the *Cisco IOS Debug Command Reference* .

For a complete description of the troubleshooting commands in this chapter, refer to the "Troubleshooting and Fault Management Commands" chapter in "Cisco IOS System Management Commands" part of the Release 12.2 Cisco IOS Configuration Fundamentals Command Reference. To locate documentation of other commands that appear in this chapter, use the *Cisco IOS Command Reference Master Index* or search online.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Troubleshooting and Fault Management Task List

To manage network faults, you need to discover, isolate, and correct problems. You can discover problems with the system monitoring commands, isolate problems with the system test commands, and resolve problems with other commands, including **debug** commands.

To perform general fault management, perform the tasks described in the following sections:

In addition to the material presented in this chapter, many chapters in the Cisco IOS software configuration guides include fault management tasks specific to certain technologies and features. You can find these tasks in the “Monitoring and Maintaining” sections.

Displaying System Information Using show Commands

To provide information about system processes, the Cisco IOS software includes an extensive list of show EXEC commands. Following is a partial list of system management **show** commands. To display the information described, use the following commands in EXEC mode, as needed:

Command	Purpose
`Router#` **show c2600**	Displays information about the Cisco 2600 platform, including interrupts, IOS Priority Masks, and IDMA status, for troubleshooting.
`Router#` **show c7200**	Displays information about the CPU and midplane for the Cisco 7200 series routers.
`Router#` **show context**	Displays information stored in NVRAM when the router crashes. This command is only useful to your technical support representative. This command is supported on the Cisco 2600 and 7000 series routers.
`Router#` **show controllers**	Displays information specific to the hardware on a line card.
`Router#` **show controllers logging**	Displays logging information about a line card.
`Router#` **show controllers tech-support**	Displays general information about a line for use when reporting a problem.
`Router#` **show controllers vip** *slot-number* **tech-support**	Displays information about the Versatile Interface Processor (VIP) card for use when reporting a problem
`Router#` **show diag**	Displays hardware information (including DRAM and static RAM details) for line cards.

Command	Purpose
`Router#` **show environment** [**all** | **last** | **table**]	Displays a message indicating whether an environmental warning condition currently exists, the temperature and voltage information, the last measured value from each of the six test points stored in nonvolatile memory, or environmental specifications. Examples of systems that support this command include the Cisco 7000 and the Cisco 12000 series routers.
`Router#` **show gsr**	Displays hardware information on the Cisco 12000 series Gigabit Switch Router (GSR).
`Router#` **show gt64010**	Displays all GT64010 internal registers and interrupt status on the Cisco 7200 series routers.
`Router#` **show memory** [*memory-type*] [**free**] [**summary**]	Displays memory pool statistics including summary information about the activities of the system memory allocator and a block-by-block listing of memory use.
`Router#` **show pci** {**hardware** | **bridge** [*register*]}	Displays information about the peripheral component interconnect (PCI) hardware registers or bridge registers for the Cisco 2600 and 7000 series routers.
`Router#` **show processes** [**cpu**]	Displays information about all active processes.
`Router#` **show processes memory**	Displays information about memory usage.
`Router#` **show protocols**	Displays the configured protocols.
`Router#` **show stacks**	Displays stack usage of processes and interrupt routines, including the reason for the last system reboot. This command is only useful to your technical support representative.
`Router#` **show subsys** [**class** *class* | **name** *name*]	Displays subsystem information.
`Router#` **show tcp** [*line-number*]	Displays the status of TCP connections.
`Router#` **show tcp brief** [**all**]	Displays a concise description of TCP connection endpoints.
`Router#` **show tdm connections** [**motherboard** | **slot** *number*]	Displays a snapshot of the time-division multiplexing (TDM) bus connection or data memory in a Cisco AS5200 access server.

Command	Purpose
`Router#` **show tech-support** [**page**] [**password**]	Displays information about the system for use when reporting a problem.

Refer to specific **show** commands in the tables of configuration commands found throughout the chapters in Cisco IOS software configuration guides. Refer to the Cisco IOS software command reference publications for detailed descriptions of the commands.

Testing Network Connectivity

Configuring the TCP Keepalive Packet Service

The TCP keepalive capability allows a router to detect when the host with which it is communicating experiences a system failure, even if data stops being sent (in either direction). This capability is most useful on incoming connections. For example, if a host failure occurs while the router is communicating with a printer, the router might never notice, because the printer does not generate any traffic in the opposite direction. If keepalives are enabled, they are sent once every minute on otherwise idle connections. If 5 minutes pass and no keepalives are detected, the connection is closed. The connection is also closed if the host replies to a keepalive packet with a reset packet. This will happen if the host crashes and comes back up again.

To generate the TCP keepalive packet service, use the following command in global configuration mode:

Command	Purposes
`Router(config)#` **service** {**tcp-keepalives-in** | **tcp-keepalives-out**}	Generates TCP keepalive packets on idle network connections, either incoming connections initiated by a remote host, or outgoing connections initiated by a user.

Testing Connections with the ping Command

As an aid to diagnosing basic network connectivity, many network protocols support an echo protocol. The protocol involves sending a special datagram to the destination host, then waiting for a reply datagram from that host. Results from this echo protocol can help in evaluating the path-to-host reliability, delays over the path, and whether the host can be reached or is functioning.

To invoke the echo protocol, use the following command in either user or privileged EXEC mode:

Command	Purposes
`Router#` **ping** [*protocol*] {*host* | *address*}	Invokes a diagnostic tool for testing connectivity.

Refer to specific **ping** commands in the tables of configuration commands found throughout the chapters in Cisco IOS software configuration guides. Refer to the Cisco IOS software command reference publications for detailed descriptions of the command.

Tracing Packet Routes

To trace the routes that packets will actually take when traveling to their destinations, use the following command in either user or privileged EXEC mode:

Command	Purposes
`Router#` **trace** [*protocol*] [*destination*]	Traces packet routes through the network (privileged level).

Logging System Messages

By default, routers send logging messages (including debug command output) a logging process. The logging process controls the distribution of logging messages to various destinations, such as the logging buffer, terminal lines, or a UNIX syslog server, depending on your configuration. The process also sends messages to the console. When the logging process is on, the messages are displayed on the console after the process that generated them has finished.

When the logging process is disabled, messages are sent only to the console. The messages are sent as they are generated, so error and debug output will be interspersed with prompts or output from the command.

You can set the severity level of the messages to control the type of messages displayed for the console and each destination. You can time-stamp log messages or set the syslog source address to enhance real-time debugging and management.

System logging messages are traditionally referred to as System Error Messages. Refer to the *Cisco IOS Software System Error Messages* publication for detailed information on specific system logging messages.

Enabling System Message Logging

System message logging is enabled by default. It must be enabled in order to send messages to any destination other than the console.

To disable message logging, use the **no logging on** command. Note that disabling the logging process can slow down the router because a process cannot continue until the messages are written to the console.

To reenable message logging after it has been disabled, use the following command in global configuration mode:

Command	Purposes
`Router(config)#` **logging on**	Enables message logging.

Enabling Message Logging for a Slave Card

To enable slave VIP cards to log status messages to the console (print the messages to the screen), use the following command in global configuration mode:

Command	Purposes
`Router(config)#` **service slave-log**	Enables slave message logging.

Setting the Syslog Destination

If message logging is enabled, you can send messages to specified locations, in addition to the console.

To set the locations that receive messages, use the following commands, as needed:

Command	Purposes
`Router(config)#` **logging buffered** [*size*]	Logs messages to an internal buffer.
`Router(config)#` **logging** *host*	Logs messages to a syslog server host.
`Router#` **terminal monitor**	Logs messages to a nonconsole terminal.

The **logging buffered** command copies logging messages to an internal buffer. The buffer is circular, so newer messages overwrite older messages after the buffer is full. To display the messages that are logged in the buffer, use the **show logging** EXEC command. The first message displayed is the oldest message in the buffer. To clear the current contents of the buffer, use the **clear logging** privileged EXEC command.

The **logging**command identifies a syslog server host to receive logging messages. The *host* argument is the name or 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.

The **terminal monitor** EXEC command locally accomplishes the task of displaying the system logging messages to a terminal.

Configuring Synchronization of Logging Messages

You can configure the system to synchronize unsolicited messages and **debug** command output with solicited device output and prompts for a specific line. You can identify the types of messages to be output asynchronously based on the level of severity. You can also determine the maximum number of buffers for storing asynchronous messages for the terminal after which messages are dropped.

When synchronous logging of unsolicited messages and **debug** command output is turned on, unsolicited device output is displayed on the console or printed after solicited device output is displayed or printed.

Unsolicited messages and **debug** command output is displayed on the console after the prompt for user input is returned. Therefore, unsolicited messages and **debug** command output are not interspersed with solicited device output and prompts. After the unsolicited messages are displayed, the console displays the user prompt again.

To configure for synchronous logging of unsolicited messages and **debug** command output with solicited device output and prompts, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **line** [**aux**| **console** | **vty**] *beginning-line-number* [*ending-line-number*]
2. Router(config-line)# **logging synchronous** [**level** *severity-level* | **all**] [**limit** *number-of-buffers*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **line** [**aux**| **console**	**vty**] *beginning-line-number* [*ending-line-number*]	Specifies the line to be configured for synchronous logging of messages.
Step 2	Router(config-line)# **logging synchronous** [**level** *severity-level*	**all**] [**limit** *number-of-buffers*]	Enables synchronous logging of messages.

Enabling Time-Stamps on Log Messages

By default, log messages are not time-stamped. To enable time-stamping of log messages, use either of the following commands in global configuration mode:

Command	Purposes
`Router(config)#` **service timestamps log uptime** or `Router(config)#` **service timestamps log datetime** [**msec**] [**localtime**] [**show-timezone**]	Enables log time stamps.

Limiting the Error Message Severity Level and Facilities

You can limit the number of messages displayed to the selected device by specifying the severity level of the error message (see the table below for level descriptions). To do so, use the following commands in global configuration mode, as needed:

Command	Purposes
`Router(config)#` **logging console** *level*	Limits the number of messages logged to the console.
`Router(config)#` **logging monitor** *level*	Limits the number of messages logged to the terminal lines.

Command	Purposes
`Router(config)#` **logging trap** *level*	Limits the number of messages logged to the syslog servers.

If you have enabled syslog messages traps to be sent to a Simple Network Management Protocol (SNMP) network management station with the **snmp-server enable trap**command, you can change the level of messages sent and stored in a history table on the router. You can also change the number of messages that get stored in the history table.

Messages are stored in the history table because SNMP traps are not guaranteed to reach their destination. By default, one message of the level warning and above (see the table above) is stored in the history table even if syslog traps are not enabled.

To change level and table size defaults, use the following commands in global configuration mode:

SUMMARY STEPS

1. Router(config)# **logging history** *level*
2. Router(config)# **logging history size** *number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **logging history** *level*	Changes the default level of syslog messages stored in the history file and sent to the SNMP server.
Step 2	Router(config)# **logging history size** *number*	Changes the number of syslog messages that can be stored in the history table.

Note The table below lists the level keywords and severity level. For SNMP usage, the severity level values use +1. For example, **emergency** equals 1 not 0 and **critical** equals 3 not 2.

The **logging console** command limits the logging messages displayed on the console terminal to messages with a level number at or below the specified severity level, which is specified by the *level* argument. The table below lists the error message *level* keywords and corresponding UNIX syslog definitions in order from the most severe level to the least severe level.

Table 1 System Logging Message Severity Levels

Level Keyword	Level	Description	Syslog Definition
emergencies	0	System unusable	LOG_EMERG
alerts	**1**	Immediate action needed	LOG_ALERT
critical	**2**	Critical conditions	LOG_CRIT
errors	**3**	Error conditions	LOG_ERR
warnings	4	Warning conditions	LOG_WARNING

Level Keyword	Level	Description	Syslog Definition
notifications	**5**	Normal but significant condition	LOG_NOTICE
informational	**6**	Informational messages only	LOG_INFO
debugging	7	Debugging messages	LOG_DEBUG

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

The default is to log messages to the console at the **debugging** level and those level numbers that are lower, which means all levels. The **logging monitor** command defaults to **debugging** also. The **logging trap** command defaults to the **informational** level.

To display logging messages on a terminal, use the **terminal monitor** EXEC command.

Current software generates the following four categories of error messages:

- Error messages about software or hardware malfunctions, displayed at levels **warnings** through **emergencies**
- Output from the **debug** commands, displayed at the **debugging** level
- Interface up/down transitions and system restart messages, displayed at the **notifications** level
- Reload requests and low-process stack messages, displayed at the **informational** level

Defining the UNIX System Logging Facility

You can log messages produced by UNIX system utilities. To do this, enable this type logging and define the UNIX system facility from which you want to log messages. The table below lists the UNIX system facilities supported by the Cisco IOS software. Consult the operator manual for your UNIX operating system for more information about these UNIX system facilities. The syslog format is compatible with Berkeley Standard Distribution (BSD) UNIX version 4.3.

To define UNIX system facility message logging, use the following command in global configuration mode:

Command	Purposes
`Router(config)#` **logging facility** *facility-type*	Configures system log facilities.

Table 2 Logging Facility Type Keywords

Facility Type Keyword	Description
auth	Indicates the authorization system.
cron	Indicates the cron facility.
daemon	Indicates the system daemon.
kern	Indicates the Kernel.
local0-7	Reserved for locally defined messages.

Facility Type Keyword	Description
lpr	Indicates line printer system.
mail	Indicates mail system.
news	Indicates USENET news.
sys9	Indicates system use.
sys10	Indicates system use.
sys11	Indicates system use.
sys12	Indicates system use.
sys13	Indicates system use.
sys14	Indicates system use.
syslog	Indicates the system log.
user	Indicates user process.
uucp	Indicates UNIX-to-UNIX copy system.

Displaying Logging Information

To display logging information, use the following commands in EXEC mode, as needed:

Command	Purposes
`Router#` **show logging**	Displays the state of syslog error and event logging, including host addresses, whether console logging is enabled, and other logging statistics.
`Router#` **show controllers vip** *slot-number* **logging**	Displays the state of syslog error and event logging of a VIP card, including host addresses, whether console logging is enabled, and other logging statistics.
`Router#` **show logging history**	Displays information in the syslog history table such as the table size, the status of messages, and the text of the messages stored in the table.

Logging Errors to a UNIX Syslog Daemon

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

```
local7.debugging /usr/adm/logs/cisco.log
```

The **debugging** keyword specifies the syslog level; see Logging Errors to a UNIX Syslog Daemon, page 10 for a general description of other keywords. The **local7** keyword specifies the logging facility to be used; see Logging Errors to a UNIX Syslog Daemon, page 10 for a general description of other keywords.

The syslog daemon sends messages at this level or at a more severe level to the file specified in the next field. The file must already exist, and the syslog daemon must have permission to write to it.

Setting the Syslog Source Address

By default, a syslog message contains the IP address of the interface it uses to leave the router. To set all syslog messages to contain the same IP address, regardless of which interface they use, use the following command in global configuration mode.

Command	Purposes
`Router(config)#` **logging source-interface** *type number*	Sets the syslog source address.

Using Field Diagnostics on Line Cards

Each line card on the Cisco 12000 series routers can perform field diagnostic testing to isolate faulty hardware without disrupting normal operation of the system. However, performing field diagnostic testing on a line card does halt all activity on the line card for the duration of the testing. After successful completion of the field diagnostic testing, the Cisco IOS software is automatically reloaded on the line card.

Note The field diagnostic **diag** command must be executed from the Gigabit Route Processor (GRP) main console port.

To perform field diagnostic testing on a line card, use the following command in privileged EXEC mode:

Command	Purposes
`Router#` **diag** *slot-number* [**previous** | **post** | **verbose** | **wait**]	Specifies the line card on which you want to perform diagnostic testing. Optionally, specifies that previous test results are displayed, that only extended power-on self-tests (POST) be performed, that the maximum messages are displayed, or that the Cisco IOS software not be reloaded on the line card after successful completion of the tests. The following prompt is displayed: `Running Diags will halt ALL activity on the requested slot. [confirm]` At the prompt, press **Return** to confirm that you want to perform field diagnostic testing on the specified line card, or type **no** to stop the testing.

To stop field diagnostic testing on a line card, use either of the following commands in privileged EXEC mode:

Command	Purpose
`Router#` **diag** *slot-number* **halt**	Specifies the line card on which you want to stop diagnostic testing.
or	
`Router#` **no diag** *slot-number*	

Note When you stop the field diagnostic test, the line card remains down (that is, in an unbooted state). In most cases, you stopped the testing because you need to remove the line card or replace the line card. If that is not the case and you want to bring the line card back up (that is, online), you must use the **microcode reload** global configuration command or power cycle the line card.

Troubleshooting Specific Line Cards

Cisco IOS provides the **execute-on** command to allow you to issue Cisco IOS commands (such as **show** commands) to a specific line card for monitoring and maintenance. For example, you could show which Cisco IOS image is loaded on the card in slot 3 of a Cisco 12012 Gigabit Switch Router (GSR) by issuing the **execute-on slot 3 show version**command. You can also use this command for troubleshooting cards in the dial shelf of Cisco access servers.

Storing Line Card Crash Information

This section explains how to enable storing of crash information for a line card and optionally specify the type and amount of information stored. Technical support representatives need to be able to look at the crash information from the line card to troubleshoot serious problems on the line card. The crash information contains all the line card memory information, including the main memory and transmit and receive buffer information.

Caution Use the **exception linecard** global configuration command only when directed by a technical support representative, and only enable options that the technical support representative requests you to enable.

To enable and configure the crash information options for a line card, use the following command in global configuration mode.

Command	Purpose
`Router(config)#` **exception linecard** {**all** | **slot** *slot-number*} [**corefile** *filename* | **main-memory** *size* [**k** | **m**] | **queue-ram** *size* [**k** | **m**] | **rx-buffer** *size* [**k** | **m**] | **sqe-register-rx** | **sqe-register-tx** | **tx-buffer** *size* [**k** | **m**]]	Specifies the line card for which you want crash information when a line card resets. Optionally, specify the type and amount of memory to be stored.

Creating Core Dumps for System Exceptions

"System exceptions" are any unexpected system shutdowns or reboots (most frequently caused by a system failure, commonly referred to as a "system crash"). When an exception occurs, it is sometimes useful to obtain a full copy of the memory image (called a core dump) to identify the cause of the unexpected shutdown. Not all exception types will produce a core dump.

Core dumps are generally useful only to your technical support representative. The core dump file, which is a very large binary file, can be transferred to a Trivial File Transfer Protocol (TFTP), File Transfer Protocol (FTP), or Remote Copy Protocol (RCP) server, or (on limited platforms) saved to the flash disk, and subsequently interpreted by technical personnel who have access to source code and detailed memory maps.

Caution Use the **exception** commands only under the direction of a technical support representative. Creating a core dump while the router is functioning in a network can disrupt network operation.

- Specifying the Destination for the Core Dump File, page 13
- Creating an Exception Memory Core Dump, page 17

Specifying the Destination for the Core Dump File

To configure the router to generate a core dump, you must enable exception dumps and configure a destination for the core dump file, as described in the following sections:

- Using TFTP for Core Dumps, page 13
- Using FTP for Core Dumps, page 14
- Using rcp for Core Dumps, page 15
- Using a Flash Disk for Core Dumps, page 16

Using TFTP for Core Dumps

Due to a limitation of most TFTP applications, the router will dump only the first 16 MB of the core file. Therefore, if your router's main memory is larger than 16 MB, do not use TFTP.

To configure a router for a core dump using TFTP, use the following commands in global configuration mode:

SUMMARY STEPS

1. **exception protocol tftp**
2. **exception dump** *ip-address*
3. **exception core-file** [*filepath*/]*filename*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**exception protocol tftp** **Example:**	(Optional) Explicitly specifies TFTP as the protocol to be used for router exceptions (core dumps for unexpected system shutdowns). **Note** Because TFTP is the default exception protocol, the **exception protocol tftp** command does not need to be used unless the protocol has been previously changed to ftp or rcp in your system's configuration. To determine if the exception protocol has been changed, use the **show running-config** command in EXEC mode.
Step 2	**exception dump** *ip-address* **Example:**	Configures the router to dump a core file to the specified server if the router crashes.
Step 3	**exception core-file** [*filepath*/]*filename* **Example:**	(Optional) Specifies the name to be used for the core dump file. The file usually must pre-exist on the TFTP server, and be writable.

For example, the following command configures a router to send a core file to the server at the IP address 172.17.92.2. As the exception protocol is not specified, the default protocol of TFTP will be used.

```
Router(config)# exception dump 172.17.92.2
```

The core dump is written to a file named "*hostname* -core" on the TFTP server, where hostname is the name of the route (in the example above, the file would be named Router-core). You can change the name of the core file by adding the **exception core-file** filename configuration command.

Depending on the TFTP server application used, it may be necessary to create, on the TFTP server, the empty target file to which the router can write the core. Also, make sure there is enough memory on your TFTP server to hold the complete core dump.

Using FTP for Core Dumps

To configure the router for a core dump using FTP, use the following commands in global configuration mode:

SUMMARY STEPS

1. Router(config)# **ip ftp username** *username*
2. Router(config)# **ip ftp password**[**type**] **password**
3. Router(config)# **exception protocol ftp**
4. Router(config)# **exception dump** *ip-address*
5. Router(config)# **exception core-file** *filename*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **ip ftp username** *username*	(Optional) Configures the user name for FTP connections.
Step 2	Router(config)# **ip ftp password**[**type**] **password**	(Optional) Specifies the password to be used for FTP connections.
Step 3	Router(config)# **exception protocol ftp**	Specifies that FTP should be used for core dump file transfers.
Step 4	Router(config)# **exception dump** *ip-address*	Configures the router to dump a core file to a particular server if the router crashes.
Step 5	Router(config)# **exception core-file** *filename*	(Optional) Specifies the name to be used for the core dump file.

The following example configures a router to use FTP to dump a core file named “dumpfile” to the FTP server at 172.17.92.2 when it crashes.

```
ip ftp username red
ip ftp password blue
exception protocol ftp
exception dump 172.17.92.2
exception core-file dumpfile
```

Using rcp for Core Dumps

The remote copy protocol can also be used to send a core dump file. To configure the router to send core dump files using rcp, use the following commands:

SUMMARY STEPS

1. i**p rcmd remote-username** *username*
2. **exception protocol rcp**
3. **exception dump** *ip-address*
4. **exception core-file** *filename*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ip rcmd remote-username** *username*	(Optional) Specifies the username sent by the router to the remote server with an rcp copy/write request. The remote rcp server must configured to grant write access to the specified username (in other words, an account must be defined on the network server for the username).
Step 2	**exception protocol rcp**	Configures the rcp as the protocol to use for sending core dump files.
Step 3	**exception dump** *ip-address* **Example:**	Configures the router to dump a core file to the specified server if the router crashes.
Step 4	**exception core-file** *filename* **Example:**	(Optional) Specifies the name to be used for the core dump file.

When an rcp username is not configured through the **ip rcmd remote-username** command, the rcp username defaults to the username associated with the current terminal (tty) connection. For example, if the user is connected to the router through Telnet and was authenticated through the username command, the router software sends the Telnet username as the rcp username. If the terminal username is not available, the router hostname will be used as the rcp username.

Using a Flash Disk for Core Dumps

Some router platforms support the Flash disk as an alternative to the linear Flash memory or PCMCIA Flash card. The large storage capacity of these Flash disks makes them good candidates for another means of capturing a core dump. To configure a router for a core dump using a Flash disk, use the following command in global configuration mode:

Command	Purpose
`Router(config)# exception flash [procmem |iomem | all] device-name [: partition-number] [erase | no_erase]`	Configures the router for a core dump using a flash disk.
`Router(config)# exception core-file` *filename*	(Optional) Specifies the name to be used for the core dump file.

The **show flash all** EXEC command will list the devices you can use for the **exception flash** command.

Creating an Exception Memory Core Dump

To cause the router to create a core dump and reboot when certain memory size parameters are violated during the debugging process, use the following commands in global configuration mode:

As a debugging procedure, you can cause the router to create a core dump and reboot when certain memory size parameters are violated. The following **exception memory** commands are used to trigger a core dump:

Command	Purpose
`Router(config)#` **exception memory minimum** *bytes*	Triggers a core dump and system reload when the amount of free memory falls below the specified number of bytes. • Do not specify too low a memory value, as the router needs some amount of free memory to provide the core dump. • If you enter a size that is greater than the free memory (and the **exception dump** command has been configured), a core dump and router reload is generated after 60 seconds.
`Router(config)#` **memory check-interval** *seconds*	(Optional) Increases the interval at which memory will be checked. The default is 60 seconds, but much can happen in 60 seconds to mask the cause of corruption. Reducing the interval will increase CPU utilization (by around 12%) which will be acceptable in most cases, but will also increase the chance of getting a usable core. To make sure CPU utilization doesn't hit 100%, you should gradually decrease the interval on busy routers. The ideal interval is as low as possible without causing other system problems.
`Router(config)#` **exception memory fragment** *bytes*	Triggers a core dump and system reload when the amount of contiguous (non-fragmented) free memory falls below the specified number of bytes.
`Router(config)#` **exception core-file** *filename*	(Optional) Specifies the name to be used for the core dump file. The file usually must exist on the TFTP server, and be writable. Note that the file will be the same size as the amount of processor memory on the router.

Note that the **exception memory minimum** command is primarily useful if you anticipate running out of memory before a core dump can be triggered or other debugging can be performed (rapid memory leak); if the memory leak is gradual (slow drift), you have generally have time to perform debugging before the system runs out of memory and must be reloaded.

By default, the number of free memory bytes is checked every 60 seconds when these commands are configured. The frequency of this checking can be increased using the **memory check-interval** *seconds* command.

The **exception dump** *ip-address*command must be configured with these commands. If the **exception dump** command is not configured, the router reloads without triggering a core dump.

The following example configures the router to monitor the free memory. If the memory falls below 250000 bytes, the core dump is created and the router reloads.

```
exception dump 172.18.92.2
exception core-file memory.overrun
exception memory minimum 250000
```

- Setting a Spurious Interrupt Core Dump, page 18

Setting a Spurious Interrupt Core Dump

During the debugging process, you can configure the router to create a spurious interrupt core dump and reboot when a specified number of interrupts have occurred.

Caution Use the **exception spurious-interrupt** global configuration command only when directed by a technical support representative and only enable options requested by the technical support representative.

To enable and configure the crash information for spurious interrupts, use the following commands in global configuration mode:

Command	Purpose
`Router(config)#` **exception spurious-interrupt** *number*	Sets the maximum number of spurious interrupts to include in the core dump before reloading.
`Router(config)#` **exception dump** *ip-address* or `Router(config)#` **exception flash**	Specifies the destination for the core dump file.

The following example configures a router to create a core dump with a limit of two spurious interrupts:

```
exception spurious-interrupt 2
exception dump 209.165.200.225
```

Enabling Debug Operations

Your router includes hardware and software to aid in troubleshooting internal problems and problems with other hosts on the network. The **debug** privileged EXEC mode commands start the console display of several classes of network events. The following commands describe in general the system debug message feature. Refer to the *Cisco IOS Debug Command Reference* for all information regarding **debug** commands. Also refer to the *Internetwork Troubleshooting Guide* publication for additional information.

To enable debugging operations, use the following commands:

Command	Purposes
`Router#` **show debugging**	Displays the state of each debugging option.
`Router#` **debug ?**	Displays a list and brief description of all the **debug** command options.
`Router#` **debug** *command*	Begins message logging for the specified **debug** command.
`Router#` **no debug** *command*	Turns message logging off for the specified **debug** command.

Caution The system gives high priority to debugging output. For this reason, debugging commands should be turned on only for troubleshooting specific problems or during troubleshooting sessions with technical support personnel. Excessive debugging output can render the system inoperable.

You can configure time-stamping of system **debug** messages. Time-stamping enhances real-time debugging by providing the relative timing of logged events. This information is especially useful when customers send debugging output to your technical support personnel for assistance. To enable time-stamping of system **debug** messages, use either of the following commands in global configuration mode:

Command	Purposes
`Router(config)#` **service timestamps debug uptime** or `Router(config)#` **service timestamps debug datetime** `[`**msec**`]` `[`**localtime**`]` `[`**show-timezone**`]`	Enables time-stamping of system **debug** messages.

Normally, the messages are displayed only on the console terminal. Refer to the section "Setting the Syslog Destination, page 6" earlier in this chapter to change the output device.

Enabling Conditionally Triggered Debugging

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 will not generate debugging output for packets entering or leaving through a different interface. You can specify the interfaces explicitly. For example, you may only want to see debugging messages for one interface or subinterface. You can also turn on debugging for all interfaces that meet specified condition. This feature is useful on dial access servers, which have a large number of ports.

Normally, the router will generate debugging messages for every interface, resulting in a large number of messages. The large number of messages consumes system resources, and can affect your ability to find the specific information you need. By limiting the number of debugging messages, you can receive messages related to only the ports you wish to troubleshoot.

Conditionally Triggered Debugging controls the output from the following protocol-specific **debug** commands:

- **debug aaa** {**accounting** | **authorization** | **authentication**}
- **debug dialer** {**events** | **packets**}
- **debug isdn** {**q921** | **q931**}
- **debug modem** {**oob** | **trace**}
- **debug ppp** {**all** | **authentication** | **chap** | **error** | **negotiation** | **multilink events** | **packet**}

Although this feature limits the output of the commands listed, it does not automatically enable the generation of debugging output from these commands. Debugging messages are generated only when the protocol-specific **debug** command is enabled. The **debug** command output is controlled through two processes:

- The protocol-specific **debug** commands specify which protocols are being debugged. For example, the **debug dialer events** command generates debugging output related to dialer events.
- The **debug condition** commands limit these debugging messages to those related to a particular interface. For example, the **debug condition username bob** command generates debugging output only for interfaces with packets that specify a username of bob.

To configure Conditionally Triggered Debugging, perform the tasks described in the following sections:

- Enabling Protocol-Specific debug Commands, page 20
- Enabling Conditional Debugging Commands, page 20
- Specifying Multiple Debugging Conditions, page 22
- Conditionally Triggered Debugging Configuration Examples, page 23

Enabling Protocol-Specific debug Commands

In order to generate any debugging output, the protocol-specific **debug** command for the desired output must be enabled. Use the **show debugging** command to determine which types of debugging are enabled. To display the current debug conditions, use the **show debug condition** command. To enable the desired protocol-specific **debug** commands, use the following commands in privileged EXEC mode:

Command	Purpose
`Router#` **show debugging**	Determines which types of debugging are enabled.
`Router#` **show debug condition** [*condition-id*]	Displays the current **debug** conditions.
`Router#` **debug** *protocol*	Enables the desired debugging commands.
`Router#` **no debug** *protocol*	Disables the debugging commands that are not desired.

If you do not want output, disable all the protocol-specific **debug** commands.

Enabling Conditional Debugging Commands

If no **debug condition** commands are enabled, all debugging output, regardless of the interface, will be displayed for the enabled protocol-specific **debug** commands.

The first **debug condition** command you enter enables conditional debugging. The router will display only messages for interfaces that meet one of the specified conditions. If multiple conditions are specified, the interface must meet at least one of the conditions in order for messages to be displayed.

To enable messages for interfaces specified explicitly or for interfaces that meet certain conditions, perform the tasks described in the following sections:

- Displaying Messages for One Interface, page 21
- Displaying Messages for Multiple Interfaces, page 21
- Limiting the Number of Messages Based on Conditions, page 21

Displaying Messages for One Interface

To disable debugging messages for all interfaces except one, use the following command in privileged EXEC mode:

Command	Purpose
`Router#` **debug condition interface** *interface*	Enables debugging output for only the specified interface.

To reenable debugging output for all interfaces, use the **no debug interface**command.

Displaying Messages for Multiple Interfaces

To enable debugging messages for multiple interfaces, use the following commands in privileged EXEC mode:

SUMMARY STEPS

1. Router# **debug condition interface** *interface*
2. Router# **debug condition interface** *interface*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router# **debug condition interface** *interface*	Enables debugging output for only the specified interface
Step 2	Router# **debug condition interface** *interface*	Enable debugging messages for additional interfaces. Repeat this task until debugging messages are enabled for all desired interfaces.

If you specify more than one interface by entering this command multiple times, debugging output will be displayed for all of the specified interfaces. To turn off debugging on a particular interface, use the **no debug interface** command. If you use the **no debug interface all** command or remove the last **debug interface** command, debugging output will be reenabled for all interfaces.

Limiting the Number of Messages Based on Conditions

The router can monitor interfaces to learn if any packets contain the specified value for one of the following conditions:

- username

- calling party number
- called party number

If you enter a condition, such as calling number, debug output will be stopped for all interfaces. The router will then monitor every interface to learn if a packet with the specified calling party number is sent or received on any interfaces. If the condition is met on an interface or subinterface, **debug** command output will be displayed for that interface. The debugging output for an interface is "triggered" when the condition has been met. The debugging output continues to be disabled for the other interfaces. If, at some later time, the condition is met for another interface, the debug output also will become enabled for that interface.

Once debugging output has been triggered on an interface, the output will continue until the interface goes down. However, the session for that interface might change, resulting in a new username, called party number, or calling party number. Use the **no debug interface** command to reset the debug trigger mechanism for a particular interface. The debugging output for that interface will be disabled until the interface meets one of the specified conditions.

To limit the number of debugging messages based on a specified condition, use the following command in privileged EXEC mode:

Command	Purpose
`Router#` **debug condition** {**username** *username* | **called** *dial-string* | **caller** *dial-string*}	Enables conditional debugging. The router will display only messages for interfaces that meet this condition.

To reenable the debugging output for all interfaces, enter the **no debug condition all** command.

Specifying Multiple Debugging Conditions

To limit the number of debugging messages based on more than one condition, use the following commands in privileged EXEC mode:

SUMMARY STEPS

1. Router# **debug condition**{**username** *username* | **called** *dial-string* | **caller** *dial-string*}
2. Router# **debug condition**{**username** *username* | **called** *dial-string* | **caller** *dial-string*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router# **debug condition**{**username** *username* | **called** *dial-string* | **caller** *dial-string*}	Enables conditional debugging, and specifies the first condition.
Step 2	Router# **debug condition**{**username** *username* | **called** *dial-string* | **caller** *dial-string*}	Specifies the second condition. Repeat this task until all conditions are specified.

If you enter multiple **debug condition** commands, debugging output will be generated if an interface meets at least one of the conditions. If you remove one of the conditions using the **no debug condition** command, interfaces that meet only that condition no longer will produce debugging output. However, interfaces that meet a condition other than the removed condition will continue to generate output. Only if no active conditions are met for an interface will the output for that interface be disabled.

Conditionally Triggered Debugging Configuration Examples

In this example, four conditions have been set by the following commands:

- **debug condition interface serial 0**
- **debug condition interface serial 1**
- **debug condition interface virtual-template 1**
- **debug condition username fred**

The first three conditions have been met by one interface. The fourth condition has not yet been met:

```
Router# show debug condition
Condition 1: interface Se0 (1 flags triggered)
        Flags: Se0
Condition 2: interface Se1 (1 flags triggered)
        Flags: Se1
Condition 3: interface Vt1 (1 flags triggered)
        Flags: Vt1
Condition 4: username fred (0 flags triggered)
```

When any **debug condition** command is entered, debugging messages for conditional debugging are enabled. The following debugging messages show conditions being met on different interfaces as the serial 0 and serial 1 interfaces come up. For example, the second line of output indicates that serial interface 0 meets the username fred condition.

```
*Mar  1 00:04:41.647: %LINK-3-UPDOWN: Interface Serial0, changed state to up
*Mar  1 00:04:41.715: Se0 Debug: Condition 4, username fred triggered, count 2
*Mar  1 00:04:42.963: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed
state to up
*Mar  1 00:04:43.271: Vi1 Debug: Condition 3, interface Vt1 triggered, count 1
*Mar  1 00:04:43.271: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
*Mar  1 00:04:43.279: Vi1 Debug: Condition 4, username fred triggered, count 2
*Mar  1 00:04:43.283: Vi1 Debug: Condition 1, interface Se0 triggered, count 3
*Mar  1 00:04:44.039: %IP-4-DUPADDR: Duplicate address 172.27.32.114 on Ethernet 0,
sourced by 00e0.1e3e.2d41
*Mar  1 00:04:44.283: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to up
*Mar  1 00:04:54.667: %LINK-3-UPDOWN: Interface Serial1, changed state to up
*Mar  1 00:04:54.731: Se1 Debug: Condition 4, username fred triggered, count 2
*Mar  1 00:04:54.735: Vi1 Debug: Condition 2, interface Se1 triggered, count 4
*Mar  1 00:04:55.735: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed
state to up
```

After a period of time, the **show debug condition** command displays the revised list of conditions:

```
Router# show debug condition
Condition 1: interface Se0 (2 flags triggered)
        Flags: Se0 Vi1
Condition 2: interface Se1 (2 flags triggered)
        Flags: Se1 Vi1
Condition 3: interface Vt1 (2 flags triggered)
        Flags: Vt1 Vi1
Condition 4: username fred (3 flags triggered)
        Flags: Se0 Vi1 Se1
```

Next, the serial 1 and serial 0 interfaces go down. When an interface goes down, conditions for that interface are cleared.

```
*Mar  1 00:05:51.443: %LINK-3-UPDOWN: Interface Serial1, changed state to down
*Mar  1 00:05:51.471: Se1 Debug: Condition 4, username fred cleared, count 1
*Mar  1 00:05:51.479: Vi1 Debug: Condition 2, interface Se1 cleared, count 3
*Mar  1 00:05:52.443: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed
state to down
*Mar  1 00:05:56.859: %LINK-3-UPDOWN: Interface Serial0, changed state to down
*Mar  1 00:05:56.887: Se0 Debug: Condition 4, username fred cleared, count 1
```

```
*Mar  1 00:05:56.895: Vi1 Debug: Condition 1, interface Se0 cleared, count 2
*Mar  1 00:05:56.899: Vi1 Debug: Condition 3, interface Vt1 cleared, count 1
*Mar  1 00:05:56.899: Vi1 Debug: Condition 4, username fred cleared, count 0
*Mar  1 00:05:56.903: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
*Mar  1 00:05:57.907: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed
state to down
*Mar  1 00:05:57.907: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to down
```

The final **show debug condition** output is the same as the output before the interfaces came up:

```
Router# show debug condition
Condition 1: interface Se0 (1 flags triggered)
        Flags: Se0
Condition 2: interface Se1 (1 flags triggered)
        Flags: Se1
Condition 3: interface Vt1 (1 flags triggered)
        Flags: Vt1
Condition 4: username fred (0 flags triggered)
```

Using the Environmental Monitor

Some routers and access servers have an environmental monitor that monitors the physical condition of the router. If a measurement exceeds acceptable margins, a warning message is printed to the system console. The system software collects measurements once every 60 seconds, but warnings for a given test point are printed at most once every 4 hours. If the temperature measurements are out of specification more than the shutdown, the software shuts the router down (the fan will remain on). The router must be manually turned off and on after such a shutdown. You can query the environmental monitor using the **show environment** command at any time to determine whether a measurement is out of tolerance. Refer to the *Cisco IOS System Error Messages* publication for a description of environmental monitor warning messages.

On routers with an environmental monitor, if the software detects that any of its temperature test points have exceeded maximum margins, it performs the following steps:

1. Saves the last measured values from each of the six test points to internal nonvolatile memory.
2. Interrupts the system software and causes a shutdown message to be printed on the system console.
3. Shuts off the power supplies after a few milliseconds of delay.

The system displays the following message if temperatures exceed maximum margins, along with a message indicating the reason for the shutdown:

```
Router#
%ENVM-1-SHUTDOWN: Environmental Monitor initiated shutdown
%ENVM-2-TEMP: Inlet temperature has reached SHUTDOWN level at 64(C)
```

Refer to the hardware installation and maintenance publication for your router for more information about environmental specifications.

and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.