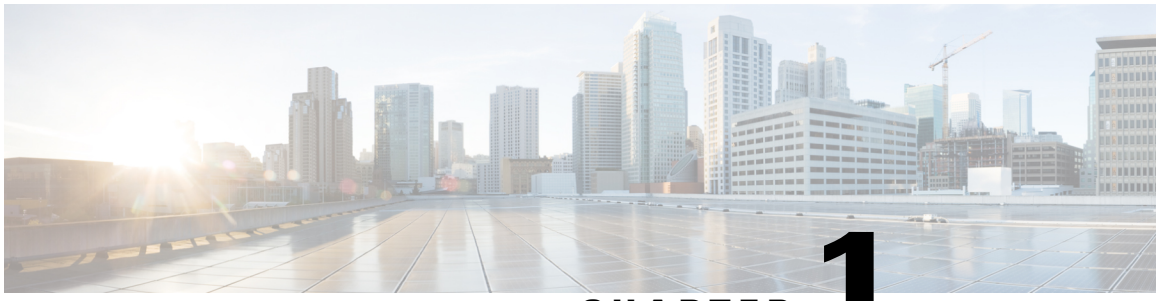
Maintaining System Memory Configuration Guide, Cisco IOS Release 15S

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Maintaining System Memory

Maintaining system memory enables you to configure, use and monitor the different types of memory on your router.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Prerequisites for Maintaining System Memory

- You should have at least a basic familiarity with the Cisco IOS environment and the command-line interface.
- You should have at least a minimal configuration running on your system with Cisco IOS Release 12.2 or later up and running.

Restrictions for Maintaining System Memory

- Many of the Cisco IOS commands described in this document are available and function only in certain configuration modes on the router.
- Some of the Cisco IOS configuration commands are only available on certain router platforms, and the command syntax may vary on different platforms.

Information About Maintaining System Memory

Memory Types

Your router has many different locations where it can store images, configuration files, and microcode. Refer to your hardware documentation for details on which types of memory your routing device contains, where files can be stored (saved), and where images and boot images are located by default. This section provides information on the following memory types:

DRAM

Dynamic random-access memory (DRAM) contains two types of memory:

- Primary, main, or processor memory, which is reserved for the CPU to execute Cisco IOS software and to hold the running configuration and routing tables.
- Shared, packet, or I/O memory, which buffers data transmitted or received by the router's network interfaces.

On the Cisco 3600 series routers, you can use the **memory-sizeiomem** command to configure the proportion of DRAM devoted to main memory and to shared memory.

DRAM often comes on dual in-line memory modules (DIMMs).

EPROM

Erasable programmable read-only memory (EPROM) is often referred to simply as ROM. On Cisco devices, the EPROM often contains the following:

- ROM Monitor software, which provides a user interface for troubleshooting the ROM.
- The boot loader/helper software, which helps the router boot when it cannot find a valid Cisco IOS image in Flash memory.

NVRAM

Non-volatile random-access-memory (NVRAM) stores the following information:

- Startup configuration file for every platform except Class A Flash file system platforms (for Class A Flash file system platforms, the location of the startup configuration depends on the CONFIG_FILE Environment Variable).
- The software configuration register, which is used to determine which image to use when booting the router.

Flash Memory

Flash memory stores the Cisco IOS software image. On most platforms, it can store boot-images and/or configuration files.

Depending on the hardware platform, Flash memory might be available as EPROM, single in-line memory modules (SIMMs), dual in-line memory modules (DIMMs), or Flash memory cards. Check the appropriate hardware installation and maintenance guide for information about types of Flash memory available on a specific platform.

Depending on the platform, Flash memory is available in the following forms:

- Internal Flash memory
  - Internal Flash memory often contains the system image.
  - Some platforms have two or more banks of Flash memory on one in-line memory module (in other words, on one SIMM). If the SIMM has two banks, it is sometimes referred to as *dual-bank Flash memory* . The banks can be partitioned into separate logical devices. See the “Partitioning Flash Memory ” section for information about how to partition Flash memory.
- Bootflash
  - Bootflash often contains the boot image.
  - Bootflash sometimes contains the ROM Monitor.
- Flash memory PC cards or PCMCIA cards

A Flash memory card that is inserted in to a Personal Computer Memory Card International Association (PCMCIA) slot. This card is used to store system images, boot images, and configuration files.

Note Because some platforms, such as the Cisco 3600 series and Cisco the 7000 family, can boot images and load configuration files from several locations, these systems use special ROM monitor environment variables to specify the location and filename of images and configuration files that the router is to use for various functions.

Ma ny Cisco routers load the system image from flash storage into RAM in order to run the Cisco IOS. However, some platforms, such as the Cisco 1600 Series and Cisco 2500 Series, execute the Cisco IOS operation system directly from Flash memory. These platforms are run-from-Flash memory systems.

If you want to partition Flash memory, you must use a relocatable image. Relocatable images can be run from any location in Flash and can download images to any location. If you are upgrading from a nonrelocatable image to a relocatable image, you must erase Flash memory during the download so that the image is downloaded as the first file in Flash memory. All images for run-from-Flash platforms from Cisco IOS Release

11.0 and later are relocatable. See the " Image Naming Conventions " section in the "Loading and Maintaining System Images" chapter to determine if your images are run-from-Flash images or are relocatable.

Flash memory provides write protection against accidental erasing or reprogramming. Some platforms have a write-protect jumper which can be removed to prevent reprogramming of Flash memory. You must install the jumper when programming is required. Some platforms have write protect switched on Flash memory cards that you can use to protect data. You must set the switch to *unprotected* to write data to the Flash memory card. Refer to your hardware documentation for information on security jumpers and write protect switches.

Note The internal Flash and Flash memory cards of a system cannot be used as a contiguous bank of Flash memory.

Partition of Flash Memory

On most Class B Flash file systems, you can partition banks of Flash memory into separate, logical devices so that the router can hold and maintain two or more different software images. This partitioning allows you to write software into Flash memory while running software in another bank of Flash memory.

Systems that Support Partitioning

To partition Flash memory, you must have at least two banks of Flash memory; a bank is a set of 4 chips. This requirement includes systems that support a single SIMM that has two banks of Flash memory. The minimum partition size is the size of a bank.

Note The CiscoFlash MIB variables support partitioned Flash.

Benefits of Partitioning Flash Memory

Partitioning Flash memory provides the following benefits:

- For any system, partitioning--rather than having one logical Flash memory device--provides a cleaner way of managing different files in Flash memory, especially if the Flash memory size is large.
- For systems that execute code out of Flash memory, partitioning allows you to download a new image into the file system in one Flash memory bank while an image is being executed from the file system in the other bank. The download is simple and causes no network disruption or downtime. After the download is complete, you can switch over to the new image at a convenient time.
- One system can hold two different images, one image acting as a backup for the other. Therefore, if a downloaded image fails to boot for some reason, the earlier running, good image is still available. Each bank is treated as a separate device.

Flash Load Helper Versus Dual Flash Bank

Flash load helper is a software option that enables you to upgrade system s oftware on run-from-Flash systems that have a single bank of Flash memory. It is a lower-cost software upgrade solution than dual-bank Flash, which requires two banks of Flash memory on one SIMM. Flash load helper is only available on run-from-Flash platforms, such as the Cisco 2500 series, Cisco 3000, and Cisco 5200.

You might use Flash load helper rather than partitioning Flash into two banks for one of the following reasons:

- If you want to download a new file into the same bank from which the current system image is executing.
- If you want to download a file that is larger than the size of a bank, and hence want to switch to a single-bank mode.
- If you have only one single-bank Flash SIMM installed. In this case, Flash load helper is the best option for upgrading your software.

See the "Downloading Files Using the Flash Load Helper " section for information about using Flash load helper.

Use of the Flash Load Helper to Upgrade Software on Run-from-Flash Systems

Flash load helper is a software option that enables you to upgrade system s oftware on run-from-Flash systems that have a single bank of Flash memory. It is a lower-cost software upgrade solution than dual-bank Flash, which requires two banks of Flash memory on one SIMM.

The Flash load helper software upgrade process is simple and does not require additional hardware; however, it does require some brief network downtime. A system image running from Flash can use Flash load helper only if the boot ROMs support Flash load helper. Otherwise, you must perform the Flash upgrade manually.

Flash load helper is an automated procedure that reloads the ROM-based image, downloads the software to Flash memory, and reboots to the system image in Flash memory. Flash load helper performs checks and validations to maximize the success of a Flash upgrade and minimize the chance of leaving Flash memory either in an erased state or with a file that cannot boot.

In run-from-Flash systems, the software image is stored in and executed from the Flash EPROM rather than from RAM. This method reduces memory cost. A run-from-Flash system requires enough Flash EPROM to hold the image and enough main system RAM to hold the routing tables and data structures. The system does not need the same amount of main system RAM as a run-from-RAM system because the full image does not reside in RAM. Run-from-Flash systems include the Cisco 2500 series and some Cisco 3000 series.

Flash Load Helper Features

Flash load helper performs the following functions:

- Confirms access to the specified source file on the specified server before erasing Flash memory and reloading to the ROM image for the actual upgrade.
- Warns you if the image being downloaded is not appropriate for the system.
- Prevents reloads to the ROM image for a Flash upgrade if the system is not set up for automatic booting and the user is not on the console terminal. In the event of a catastrophic failure during the upgrade, Flash load helper can bring up the boot ROM image as a last resort rather than forcing the system to wait at the ROM monitor prompt for input from the console terminal.

- Retries Flash downloads automatically up to six times. The retry sequence is as follows:
  - First try
  - Immediate retry
  - Retry after 30 seconds
  - Reload ROM image and retry
  - Immediate retry
  - Retry after 30 seconds
- Allows you to save any configuration changes made before you exit out of the system image.
- Notifies users logged in to the system of the impending switch to the boot ROM image so that they do not lose their connections unexpectedly.
- Logs console output during the Flash load helper operation into a buffer that is preserved through system reloads. You can retrieve the buffer contents from a running image. The output is useful when console access is unavailable or a failure occurs in the download operation.

Flash load helper can also be used on systems with multiple banks of Flash memory that support Flash memory partitioning. Flash load helper enables you to download a new file into the same partition from which the system is executing an image.

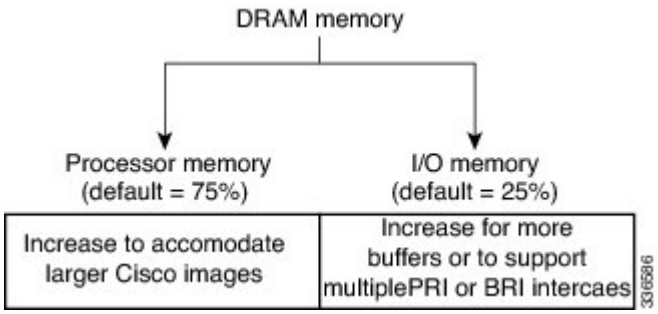
For information about how to partition multiple banks of Flash memory so your system can hold two different images, see the "Partitioning Flash Memory " section.

Allocation of DRAM Memory for the Cisco 3600 Series

DRAM memory in Cisco 3600 series routers is organized as one contiguous address space divided between processor memory and I/O memory. Depending on the type and number of network interfaces you have configured in the router, you may need to reallocate the DRAM memory partitioned to processor memory and I/O memory.

Cisco manufacturing configures most Cisco 3600 series routers to have 25 percent of the address space allocated to I/O memory and 75 percent allocated to processor memory. But for customer orders that require two or more ISDN PRI interfaces, DRAM memory is configured to provide 40 percent of the address space for I/O memory and 60 percent for processor memory. (See the figure below.) Cisco Systems performs these DRAM memory adjustments before it ships each router.

Figure 1: Components and Uses of DRAM Memory for Cisco 3600 Series Routers



Note Routers running two or more ISDN PRI interfaces or 12 or more ISDN BRI interfaces require a DRAM memory configuration of 40 percent I/O memory and 60 percent processor memory.

However, there are cases where you may have to manually reallocate the DRAM memory split between processor memory and I/O memory after you have received a router from Cisco Systems.

For example, suppose you receive a Cisco 3640 router with the following running configuration:

- 2 Ethernet and 2 WAN interface card
- 8-port ISDN BRI with an NT1 network module
- IP feature set
- 16 MB of DRAM memory (by default, processor memory = 75%, I/O memory = 25%)
- 4 MB of Flash memory

Later, however, you add a 4-port ISDN BRI network module to the router. You now have 12 ISDN BRI interfaces running on the router. At this point, you must use the **memory-sizeiomem** command to configure 40 percent of the address space for I/O memory and 60 percent for processor memory.

Memory Scan on the Cisco 7500 Series

On Cisco 7500 series routers (including 7000 series with the RSP7000 card upgrade), a memory scanning feature is available. This feature adds a low-priority background process that searches all installed dynamic random-access memory (DRAM) for possible parity errors. If errors are found in memory areas that are not in use, this feature attempts to scrub (remove) the errors. The time to complete one memory scan and scrub cycle can range from 10 minutes to several hours, depending on the amount of installed memory. The impact of the Memory Scan feature on the central processing unit (CPU) is minimal. The feature can be controlled and monitored with the new **memoryscan** and **showmemoryscan** command-line interface (CLI) commands.

The Memory Scan feature does not discriminate against different information types in DRAM; that is, it perceives text, data, and heap information in the same way. The feature continues to work when a memory cell is busy, although it might respond differently to errors found in different areas. The feature responds to errors in one or more of the following ways:

- A message is logged for all errors found. Each message contains an explanation of the error and suggests corrective action if applicable.
- For errors in heap storage control blocks, attempts are made to scrub errors in the free blocks. If an error is scrubbed, no further action occurs, but there is an entry in the error log. If it is not scrubbed, the block that contains the error is linked to a bad-memory list which will not be allocated to users. If the memory block is large, the block is split and only a small portion containing the error is linked to a bad-memory list.
- For errors in a busy block, or in other areas such as text or data, an error message is produced but no further action is taken, preventing damage to living data.

How to Configure System Memory Parameters

Displaying System Memory Information

To display information about system memory, complete the tasks in this section

SUMMARY STEPS

1. **enable**
2. **show** *flash-filesystem* **:** [**all** | **chips** |**filesys**]
3. **show** *flash-filesystem* **:** [**partition***number*][**all** | **chips** | **detailed** | **err** | **summary**]
4. **show** *flash-filesystem* **:**
5. **show file systems**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show** *flash-filesystem* **:** [**all**	**chips**	**filesys**] **Example:** `Router# show flash: chips`	Lists information about Flash memory for Class A file systems.
Step 3	**show** *flash-filesystem* **:** [**partition***number*][**all**	**chips**	**detailed**	**err**	**summary**] **Example:** `Router# show slot0: detailed`	Lists information about Flash memory for Class B file systems.
Step 4	**show** *flash-filesystem* **:** **Example:** `Router# show slot1:`	Lists information about Flash memory for Class C file systems.
Step 5	**show file systems** **Example:** `Router# show file system`	Lists the names of the file systems currently supported on the router.

Partitioning Flash Memory

To partition Flash memory, complete the tasks in this section.

Note This task will succeed only if the system has at least two banks of Flash and the partitioning does not cause an existing file in Flash memory to be split across the partitions.

For all platforms except the Cisco 1600 series and Cisco 3600 series, Flash memory can only be partitioned into two partitions.

For the Cisco 1600 series and Cisco 3600 series, the number of partitions that you can create in a Flash memory device equals the number of banks in the device. Enter the **show***flash-filesystem***:all** command to view the number of banks on the Flash memory device. The number of partition size entries you set must be equal to the number of specified partitions. For example, the **partitionslot0:288** command configures two partitions to be 8 MB in size each. The first 8 corresponds to the first partition; the second 8 corresponds to the second partition.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **partition flash** *partitions* [*size1 size2*]
4. **partition** *flash-filesystem* **:** [*number-of-partitions*] [*partition-size*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**partition flash** *partitions* [*size1 size2*] **Example:** `Router(config)# partition flash 2 4 4`	Partitions Flash memory. **Note** To remove the partition, use the**nopartition** command.

	Command or Action	Purpose
Step 4	**partition** *flash-filesystem* **:** [*number-of-partitions*] [*partition-size*] **Example:** `Router(config)# Router(config)# partition slot0: 2 8 8`	Partitions Flash memory on the Cisco 1600 and 3600 series.

Downloading Files Using the Flash Load Helper

To download a new file to Flash memory using Flash load helper, check to make sure that your boot ROMs support Flash load helper and then complete the tasks in this section:

SUMMARY STEPS

1. **enable**
2. Do one of the following:
   - **copy tftp: flash:**
   -
   - **copy rcp: flash:**
   -
   - **copy ftp: flash:**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	Do one of the following: • **copy tftp: flash:** • • **copy rcp: flash:** • • **copy ftp: flash:**	Loads the specified file to Flash memory.

	Command or Action	Purpose
	Example: `Router# copy tftp flash:`	

Troubleshooting

The following error message displays if you are in a Telnet session and the system is set for manual booting (the boot bits in the configuration register are zero):

```
ERR: Config register boot bits set for manual booting
```

In case of any catastrophic failure in the Flash memory upgrade, this error message helps to minimize the chance of the system going down to ROM monitor mode and being taken out of the remote Telnet user's control.

The system tries to bring up at least the boot ROM image if it cannot boot an image from Flash memory. Before reinitiating the **copy:**command, you must set the configuration register boot field to a nonzero value, using the **config-register** global configuration command.

Examples

The **copy** command initiates a series of prompts to which you must provide responses. The dialog is similar to the following:

```
Router#
 copy tftp: flash:
************************** NOTICE ******************************
Flash load helper v1.0
This process will accept the TFTP copy options and then terminate
the current system image to use the ROM based image for the copy.
Router functionality will not be available during that time. If
you are logged in via telnet, this connection will terminate. Users
with console access can see the results of the copy operation.
****************************************************************
There are active users logged into the system.
Proceed? [confirm] y
System flash directory:
File Length  Name/status
1    2251320 abc/igs-kf.914
[2251384 bytes used, 1942920 available, 4194304 total]
Address or name of remote host [255.255.255.255]?
172.16.1.111
Source file name?
abc/igs-kf.914
Destination file name [default = source name]?
 <Return>
Accessing file 'abc/igs-kf.914' on 172.16.1.111....
Loading from 172.16.13.111:
Erase flash device before writing? [confirm] n
File 'abc/igs-kf.914' already exists; it will be invalidated!
Invalidate existing copy of 'abc/igs-kf.914' in flash memory? [confirm] y
Copy 'abc/igs-kf.914' from TFTP server
as 'abc/igs-kf.914' into Flash WITHOUT erase? y
%SYS-5-RELOAD: Reload requested
```

```
%
FLH: rxboot/igs-kf.914r from 172.16.1.111 to flash...
```

The Flash Load Helper operation verifies the request from the running image by trying to copy a single block from the remote server. Then the Flash load helper is executed, causing the system to reload to the ROM-based system image. If the file does not seem to be a valid image for the system, a warning is displayed and a separate confirmation is sought from you.

If the configuration has been modified but not yet saved, you are prompted to save the configuration:

```
System configuration has been modified. Save? [confirm]
```

Users with open Telnet connections are notified of the system reload, as follows:

```
**System going down for Flash upgrade**
```

Troubleshooting Tips

If the copy process fails, the copy operation is retried up to three times. If the failure happens in the middle of a copy operation so that only part of the file has been written to Flash memory, the retry does not erase Flash memory unless you specified an erase operation. The partly written file is marked as deleted, and a new file is opened with the same name. If Flash memory runs out of free space in this process, the copy operation is terminated.

After Flash load helper finishes copying (whether the copy operation is successful or not), it automatically attempts an automatic or a manual boot, depending on the value of bit zero of the configuration register boot field according to the following:

- If bit zero equals 0, the system attempts a default boot from Flash memory to load up the first bootable file in Flash memory. This default boot is equivalent to a manual **bootflash** command at the ROM monitor prompt.
- If bit zero equals 1, the system attempts to boot based on the boot configuration commands. If no boot configuration commands exist, the system attempts a default boot from Flash memory; that is, it attempts to load the first bootable file in Flash memory.

To view the system console output generated during the Flash load helper operation, use the image that has been booted up after the Flash memory upgrade. Enter the **moreflh:logfile**command in privileged EXEC mode.

If you are a remote Telnet user performing the Flash upgrade without a console connection, this task allows you to retrieve console output when your Telnet connection has terminated due to the switch to the ROM image. The output indicates what happened during the download, and is particularly useful if the download fails.

Formatting Flash Memory

Restrictions

On Class A and Class C Flash file systems, you can format Flash memory. Formatting erases all information in Flash memory.

On the Cisco 7000 family, you must format a new Flash memory card before using it in a PCMCIA slot.

Flash memory cards have sectors that can fail. You can reserve certain Flash memory sectors as “spares” for use when other sectors fail. Use the **format** command to specify between 0 and 16 sectors as spares. If you reserve a small number of spare sectors for emergencies, you do not waste space because you can use most of the Flash memory card. If you specify zero spare sectors and some sectors fail, you must reformat the Flash memory card and thereby erase all existing data.

The format operation requires at least Cisco IOS Release 11.0 system software.

Flash Memory Formatting Process

Caution The following formatting procedure erases all information in Flash memory. To prevent the loss of important data, proceed carefully.

Use the following procedure to format Flash memory. If you are formatting internal Flash memory, such as bootflash, you can skip the first step. If you are formatting a Flash memory card, complete both steps:

SUMMARY STEPS

1. Insert the new Flash memory card into a PCMCIA slot. Refer to instructions on maintaining the router and replacing PCMCIA cards in your router’s hardware documentation for instructions on performing this step.
2. Enter the **format** [**spare** *spare-number*] *device1***:** [[*device2***:**][*monlib-filename*]]command to format Flash memory.

DETAILED STEPS

Step 1 Insert the new Flash memory card into a PCMCIA slot. Refer to instructions on maintaining the router and replacing PCMCIA cards in your router’s hardware documentation for instructions on performing this step.

Step 2 Enter the **format** [**spare** *spare-number*] *device1***:** [[*device2***:**][*monlib-filename*]]command to format Flash memory.

Examples

The following example shows the **format** command that formats a Flash memory card inserted in slot 0.

```
Router# format slot0:
Running config file on this device, proceed? [confirm]y
All sectors will be erased, proceed? [confirm]y
Enter volume id (up to 31 characters): <Return>
Formatting sector 1 (erasing)
Format device slot0 completed
```

Recovering from Locked Blocks

Note To recover from locked blocks, reformat the Flash memory card. A locked block of Flash memory occurs when power is lost or a Flash memory card is unplugged during a write or erase operation. When a block of Flash memory is locked, it cannot be written to or erased, and the operation will consistently fail at a particular block location. The only way to recover from locked blocks is by reformatting the Flash memory card with the **format** command.

Caution Formatting a Flash memory card to recover from locked blocks will cause existing data to be lost.

To view your current mix of processor and I/O memory and reassign memory distribution accordingly, complete the tasks in this section:

SUMMARY STEPS

1. **enable**
2. **show version**
3. **show memory**
4. **configure terminal**
5. **memory-size iomem** *I/O-memory-percentage*
6. **exit**
7. **copy system:running-config nvram:startup-config**
8. **reload**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show version** **Example:** `Router# show version`	Displays the total amount of memory loaded on the router.
Step 3	**show memory** **Example:** `Router# show memory`	Displays the amount of free memory. **Note** The Free(b) column in the show memory command's output shows how much I/O memory is available.

	Command or Action	Purpose
Step 4	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 5	**memory-size iomem** *I/O-memory-percentage* **Example:** `Router(config)# memory-size iomem 50`	Allocates processor memory and I/O memory. **Note** The default is 40 percent for I/O memory and 60 percent for processor memory
Step 6	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 7	**copy system:running-config nvram:startup-config** **Example:** `Router# copy system:running-config nvram:startup-config`	Saves the configuration to NVRAM.
Step 8	**reload** **Example:** `Router# reload`	Reloads the router to run the new image.

Troubleshooting Tips

Valid I/O memory percentage values are 10, 15, 20, 25, 30, 40 (the default), and 50. I/O memory size is the specified percentage of total memory size, rounded down to the nearest multiple of 1 MB. A minimum of 4 MB of memory is required for I/O memory. The remaining memory is processor memory.

The **memory-sizeiomem** command does not take effect until you save it to NVRAM using the **copysystem:running-confignvram:startup-config** EXEC command and reload the router. However, when you enter the command, the software checks whether the new memory distribution leaves enough processor memory for the currently running Cisco IOS image. If not, the following message appears:

```
Warning: Attempting a memory partition that does not provid
e enough Processor memory for the current image.If you write memory now, this version of
so
ftware may not be able to run.
```

When you enter the **reload** command to run a new image, the software calculates the new processor and I/O memory split. If there is not enough processor memory, it automatically reduces I/O memory to an alternative setting to load the image. If there is still not enough processor memory for the image to run, then you do not have enough DRAM.

Examples

The following example allocates 40 percent of DRAM to I/O memory and the remaining 60 percent to processor memory. The example views the current allocation of memory, changes the allocation, saves the allocation, and reloads the router so the changes can take effect. In the **showmemory**command output, the Free(b) column shows how much I/O memory is available:

```
Router# show memory
               Head   Total(b)     Used(b)     Free(b)  Lowest(b) Largest(b)
Processor  60913730    3066064     970420    2095644    2090736    2090892
      I/O    C00000    4194304    1382712    2811592    2811592    2805492
--More--
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# memory-size iomem 40
Router(config)# exit
Router#
Router# copy system:running-config nvram:startup-config
Building configuration...
[OK]
Router# reload
rommon > boot
program load complete, entry point: 0x80008000, size: 0x32ea24
Self decompressing the image :
############################################################################################################################################
 [OK]
```

Configuring and Verifying Memory Scan on the Cisco 7500 Series

To configure and verify memory scan on the Cisco 7500 series router, complete the tasks in this section:

SUMMARY STEPS

1. Use the **memoryscan** command in global configuration mode to enable the feature.
2. Use the **moresystem:running-configuration** command in privileged EXEC mode to verify that memory scan appears in the running configuration.
3. Use the **showmemoryscan**command to monitor the number and type of parity errors on your system.

DETAILED STEPS

Step 1 Use the **memoryscan** command in global configuration mode to enable the feature.

Step 2 Use the **moresystem:running-configuration** command in privileged EXEC mode to verify that memory scan appears in the running configuration.

Step 3 Use the **showmemoryscan**command to monitor the number and type of parity errors on your system.

Examples

Use the **showmemoryscan** command in privileged EXEC mode. In the following example, the feature is enabled and no parity errors are found:

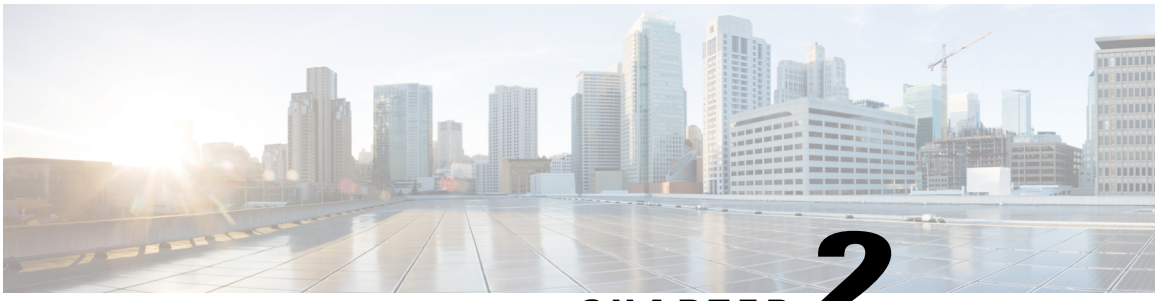
```
Router# show memory scan
Memory scan is on.
No parity error has been detected.
```

If the Memory Scan feature has not been configured, or has been turned off, the **showmemoryscan** command generates a report. In the following example, Memory Scan is turned off:

```
Router# show memory scan
Memory scan is off
No parity error has been detected.
```

If errors are detected in the system, the **showmemoryscan** command generates an error report. In the following example, Memory Scan detected a parity error:

```
Router# show memory scan
Memory scan is on.
Total Parity Errors 1.
Address BlockPtr BlckSize Disposit   Region Timestamp
6115ABCD 60D5D090   9517A4  Scrubed    Local 16:57:09 UTC Thu Mar 18
```



CHAPTER 2

Memory Leak Detector

The Memory Leak Detector feature is a tool that can be used to detect memory leaks on a router that is running Cisco IOS software. The Memory Leak Detector feature is capable of finding leaks in all memory pools, packet buffers, and chunks.

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 for Memory Leak Detector.

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

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Prerequisites for Memory Leak Detector

- You should have at least a basic familiarity with the Cisco IOS environment and the command-line interface.
- You should have at least a minimal configuration running on your system.

Restrictions for Memory Leak Detector

- You must have your network up and running, with Cisco IOS Release 12.2 or a later release installed.
- Some of the Cisco IOS configuration commands are only available on certain router platforms, and the command syntax may vary on different platforms.

Information About Memory Leak Detector

Memory Leaks

Memory leaks are static or dynamic allocations of memory that do not serve any useful purpose. Although technology is available for detection of leaks among statically allocated memory, in this document the focus is on memory allocations that are made dynamically.

Memory Leak Detection

From the detection point of view, leaks among the dynamically allocated memory blocks can be classified into the following three types:

- Type 1 leaks have no references. These blocks of memory can not be accessed.
- Type 2 leaks are part of one or more cycles of allocations but none of the blocks in these cycles is accessible from outside of the cycles. Blocks within each cycle have references to other elements in the cycle(s). An example of a Type 2 leak is a circular list that is not needed anymore. Though individual elements are reachable, the circular list is not reachable.
- Type 3 leaks are accessible or reachable but are not needed, for example, elements in data structures that are not needed anymore. A subclass of Type 3 leaks are those where allocations are made but never written to. You can look for these subclass leaks using the **showmemorydebugreferenceunused** command.

The Memory Leak Detector feature provides the technology to detect Type 1 and Type 2 memory leaks.

The Memory Leak Detector feature works in the following two modes:

- Normal mode--Where memory leak detector uses memory to speed up its operations.
- Low memory mode--Where memory leak detector runs without attempting to allocate memory.

Low memory mode is considerably slower than the normal mode and can handle only blocks. There is no support for chunks in low memory mode. Low memory mode is useful when there is little or no memory available on the router.

The memory leak detector has a simple interface and can be invoked by the command line interface (CLI) at any time to get a report of memory leaks. For testing purposes, you can perform all tests, then invoke memory leak detector to get a report on leaks. If you are interested only in leaks generated by your test cases alone, memory leak detector has an incremental option, which can be enabled at the start of testing. After testing completes, you can get a report on only the leaks that occurred after the incremental option was enabled.

To reduce false alarms, it is mandatory that memory leak detector be invoked multiple times and that only leaks that consistently appear in all reports be interpreted as leaks. This is especially true for packet buffer leaks.

Note When submitting defects based on the reports of memory leak detector, please add “memleak-detection” to the attribute field of the defect report.

Danger Executing memory leak detection commands on a device with a serious memory leak issue may cause loss of connectivity.

How to Use Memory Leak Detector

Displaying Memory Leak Information

To display detected memory leak information, complete the task in this section:

SUMMARY STEPS

1. **enable**
2. **show memory debug leaks** [**chunks** | **largest** | **lowmem** | **summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show memory debug leaks** [**chunks** | **largest** | **lowmem** | **summary**]	Invokes normal mode memory leak detection and displays detected memory leaks. Optional keywords are as follows:

	Command or Action	Purpose
	Example: `Router# show memory debug leaks chunks`	• **chunks** --Invokes normal mode memory leak detection and displays detected memory leaks in chunks. • **largest** --Invokes memory leak detection and displays the top ten leaking allocator_pcs and total amount of memory that they have leaked. Additionally, each time this command is invoked it remembers the previous invocation's report and compares it to the current invocation's report. • **lowmem** --Invokes low memory mode memory leak detection and displays detected memory leaks. The amount of time taken for analysis is considerably greater than that of normal mode. The output for this command is similar to the **showmemorydebugleaks** command. • **summary** --Invokes normal mode memory leak detection and displays detected memory leaks based on allocator_pc and then on the size of the block.

Setting the Memory Debug Incremental Starting Time

To set the starting time for incremental analysis of memory leaks, complete the task in this section:

SUMMARY STEPS

1. **enable**
2. **set memory debug incremental starting-time**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**set memory debug incremental starting-time** **Example:** `Router# set memory debug incremental starting-time`	Sets the starting time for incremental analysis to the time when the command is issued. When the starting time is set, only memory allocated after the starting time will be considered for reporting as leaks.

Displaying Memory Leak Information Incrementally

To display memory leak information after a starting time has been established, complete the tasks in this section:

SUMMARY STEPS

1. **enable**
2. **set memory debug incremental starting-time**
3. **show memory debug incremental** {**allocations** | **leaks** [**lowmem**] | **status**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**set memory debug incremental starting-time** **Example:** `Router# set memory debug incremental starting-time`	Sets the starting time for incremental analysis to the time when the command is issued.

Step 3	**show memory debug incremental** {**allocations** | **leaks** [**lowmem**] | **status**}
	
	Example:
	
	`Router# show memory debug incremental allocations`
	
	Example:
		• **allocations** --Displays all the memory blocks that were allocated after the issue of a **setmemorydebugincrementalstarting-time** command. The displayed memory blocks are just memory allocations, they are not necessarily leaks.
		
		• **leaks** --Displays output similar to the **showmemorydebugleaks** command, except that it displays only memory that was leaked after the issue of a **setmemorydebugincrementalstarting-time** command.
		
		• **lowmem** --Forces memory leak detection to work in low memory mode. The output for this command is similar to the **showmemorydebugleaks** command, except that it displays only memory that was leaked after the issue of a **setmemorydebugincrementalstarting-time** command.
		
		  • In low memory mode, the analysis time is considerably greater than it is in normal mode.
		
		  • You can use this command when you already know that normal mode memory leak detection will fail (perhaps by an unsuccessful previous attempt to invoke normal mode memory leak detection).
		
		• **status** --Displays whether a starting point for incremental analysis has been set and the elapsed time since then.

	Command or Action	Purpose

Examples for Memory Leak Detector

Example show memory debug leaks

The following example shows output from the **showmemorydebugleaks** command with no optional keywords specified:

```
Router# show memory debug leaks
Adding blocks for GD...
                PCI memory
Address    Size   Alloc_pc  PID  Name
                I/O memory
Address    Size   Alloc_pc  PID  Name
                Processor memory
Address    Size   Alloc_pc  PID  Name
62DABD28       80 60616750  -2   Init
62DABD78       80 606167A0  -2   Init
62DCF240       88 605B7E70  -2   Init
62DCF298       96 605B7E98  -2   Init
62DCF2F8       88 605B7EB4  -2   Init
62DCF350       96 605B7EDC  -2   Init
63336C28      104 60C67D74  -2   Init
63370D58       96 60C656AC  -2   Init
633710A0      304 60C656AC  -2   Init
63B2BF68       96 60C659D4  -2   Init
63BA3FE0    32832 608D2848  104  Audit Process
63BB4020    32832 608D2FD8  104  Audit Process
```

The table below describes the significant fields shown in the display.

Table 1: show memory debug leaks Field Descriptions

Field	Description
Address	Hexadecimal address of the leaked block.
Size	Size of the leaked block (in bytes).
Alloc_pc	Address of the system call that allocated the block.
PID	The process identifier of the process that allocated the block.
Name	The name of the process that allocated the block.

Example show memory debug leaks chunks

The following example shows output from the **showmemorydebugleakschunks** command:

```
Router# show memory debug leaks chunks
Adding blocks for GD...
                PCI memory
Address    Size   Alloc_pc  PID  Name
Chunk Elements:
Address  Size  Parent   Name
                I/O memory
Address    Size   Alloc_pc  PID  Name
Chunk Elements:
Address  Size  Parent   Name
                Processor memory
Address    Size   Alloc_pc  PID  Name
62DABD28       80 60616750  -2   Init
62DABD78       80 606167A0  -2   Init
62DCF240       88 605B7E70  -2   Init
62DCF298       96 605B7E98  -2   Init
62DCF2F8       88 605B7EB4  -2   Init
62DCF350       96 605B7EDC  -2   Init
63336C28      104 60C67D74  -2   Init
63370D58       96 60C656AC  -2   Init
633710A0      304 60C656AC  -2   Init
63B2BF68       96 60C659D4  -2   Init
63BA3FE0    32832 608D2848  104  Audit Process
63BB4020    32832 608D2FD8  104  Audit Process
Chunk Elements:
Address  Size  Parent   Name
62D80DA8    16 62D7BFD0 (Managed Chunk )
62D80DB8    16 62D7BFD0 (Managed Chunk )
62D80DC8    16 62D7BFD0 (Managed Chunk )
62D80DD8    16 62D7BFD0 (Managed Chunk )
62D80DE8    16 62D7BFD0 (Managed Chunk )
62E8FD60   216 62E8F888 (IPC Message He)
```

The table below describes the significant fields shown in the display.

Table 2: show memory debug leaks chunks Field Descriptions

Field	Description
Address	Hexadecimal address of the leaked block.
Size	Size of the leaked block (in bytes).
Alloc_pc	Address of the system call that allocated the block.
PID	The process identifier of the process that allocated the block.
Name	The name of the process that allocated the block.
Size	(Chunk Elements) Size of the leaked element (bytes).
Parent	(Chunk Elements) Parent chunk of the leaked chunk.
Name	(Chunk Elements) The name of the leaked chunk.

Example show memory debug leaks largest

The following example shows output from the **showmemorydebugleakslargest** command:

```
Router# show memory debug leaks largest
Adding blocks for GD...
                PCI memory
Alloc_pc    total leak size
                I/O memory
Alloc_pc    total leak size
                Processor memory
Alloc_pc    total leak size
608D2848    32776     inconclusive
608D2FD8    32776     inconclusive
60C656AC    288       inconclusive
60C67D74    48        inconclusive
605B7E98    40        inconclusive
605B7EDC    40        inconclusive
60C659D4    40        inconclusive
605B7E70    32        inconclusive
605B7EB4    32        inconclusive
60616750    24        inconclusive
```

The following example shows output from the second invocation of the **showmemorydebugleakslargest** command:

```
Router# show memory debug leaks largest
Adding blocks for GD...
                PCI memory
Alloc_pc    total leak size
                I/O memory
Alloc_pc    total leak size
                Processor memory
Alloc_pc    total leak size
608D2848    32776
608D2FD8    32776
60C656AC    288
60C67D74    48
605B7E98    40
605B7EDC    40
60C659D4    40
605B7E70    32
605B7EB4    32
60616750    24
```

Example show memory debug leaks summary

The following example shows output from the **showmemorydebugleakssummary** command:

```
Router# show memory debug leaks summary
Adding blocks for GD...
                PCI memory
Alloc PC        Size     Blocks      Bytes    What
                I/O memory
Alloc PC        Size     Blocks      Bytes    What
                Processor memory
Alloc PC        Size     Blocks      Bytes    What
0x605B7E70 0000000032 0000000001 0000000032    Init
0x605B7E98 0000000040 0000000001 0000000040    Init
0x605B7EB4 0000000032 0000000001 0000000032    Init
0x605B7EDC 0000000040 0000000001 0000000040    Init
0x60616750 0000000024 0000000001 0000000024    Init
0x606167A0 0000000024 0000000001 0000000024    Init
0x608D2848 0000032776 0000000001 0000032776    Audit Process
0x608D2FD8 0000032776 0000000001 0000032776    Audit Process
```

```
0x60C656AC 0000000040 0000000001 0000000040    Init
0x60C656AC 0000000248 0000000001 0000000248    Init
0x60C659D4 0000000040 0000000001 0000000040    Init
0x60C67D74 0000000048 0000000001 0000000048    Init
```

The table below describes the significant fields shown in the display.

Table 3: show memory debug leaks summary Field Descriptions

Field	Description
Alloc PC	Address of the system call that allocated the block.
Size	Size of the leaked block.
Blocks	Number of blocks leaked.
Bytes	Total amount of memory leaked.
What	Name of the process that owns the block.

Example show memory debug incremental allocations

The following example shows output from the **showmemorydebugincremental** command when entered with the **allocations** keyword:

```
Router# show memory debug incremental allocations
Address    Size   Alloc_pc  PID  Name
62DA4E98      176 608CDC7C  44   CDP Protocol
62DA4F48       88 608CCCC8  44   CDP Protocol
62DA4FA0       88 606224A0  3    Exec
62DA4FF8       96 606224A0  3    Exec
635BF040       96 606224A0  3    Exec
63905E50      200 606A4DA4  69   Process Events
```

Example show memory debug incremental status

The following example shows output from the **showmemorydebugincremental** command entered with the **status** keyword:

```
Router# show memory debug incremental status
Incremental debugging is enabled
Time elapsed since start of incremental debugging: 00:00:10
```

Additional References

The following sections provide references related to Memory Leak Detector.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Cisco IOS configuration commands	*Cisco IOS Configuration Fundamentals Command Reference*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/public/support/tac/home.shtml

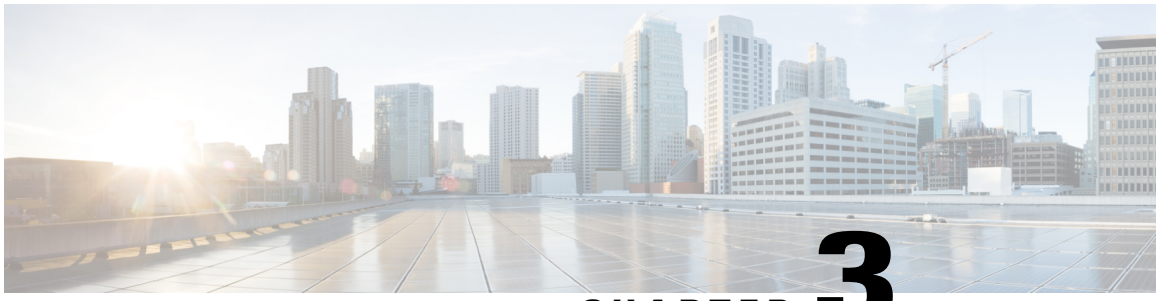
Feature Information for Memory Leak Detector

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

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.

Table 4: Feature Information for Memory Leak Detector

Feature Name	Releases	Feature Information
Memory Leak Detector	12.3(8)T1 12.2(25)S	The Memory Leak Detector feature is a tool that can be used to detect memory leaks on a router that is running Cisco IOS software. The Memory Leak Detector feature is capable of finding leaks in all memory pools, packet buffers, and chunks.



CHAPTER 3

Reserve Memory for Console Access

The Reserve Memory for Console Access feature implements command-line interface (CLI) and software enhancements that allow you to reserve sufficient memory to log in to the router console and perform administrative tasks and troubleshooting. These enhancements give administrators the ability to log in to the router in any situation even when the router is running low on memory.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Information About Reserve Memory for Console Access

More Reserved Memory for Console Access Benefit

Before the release of Cisco IOS 12.0(22)S software, you could not access the router console if a router was low on memory or was heavily fragmented. To maintain routers at optimum performance levels, you need to be able to access the console and perform troubleshooting when necessary.

With the release of the Reserve Memory for Console Access feature, the benefit is that you can reserve sufficient memory to log in to the router console and perform administrative tasks and troubleshooting in any situation, even when the router is running low on memory or is heavily fragmented.

Guidelines for Increasing Reserved Memory for Console Access

Cisco IOS software reserves a default of 256 kilobyte (KB) of memory for console access. You can increase the reserved memory through the use of the **memoryreservedconsole** command provided by the Reserve Memory for Console Access feature.

You may need to increase the amount of memory reserved for console access if the router is low on memory or is heavily fragmented. Increasing the memory allows console access to perform troubleshooting or other administrative tasks to maintain routers at optimum performance levels.

The guideline we suggest for using the command is to configure a value greater than three times the number of the used bytes in NVRAM. You can obtain the number of used bytes in NVRAM from the output of the **dirnvram:** command. For example, if the total number of used bytes of NVRAM displayed in the command **dirnvram:**output is 129016 bytes, the nearest kilobyte value rounded off is 129 KB. This value multiplied by 3 is 387 KB. Following the guideline, you would enter 387 as the value for the *number-of-kilobytes* argument in the **memoryreservedconsole** command. You can increase the reserved memory for console access to a maximum of 4096 KB.

To display the current operational size of the memory reserved for the console, you can use the **showmemoryconsolereserved** command.

How to Configure Reserve Memory for Console Access

To configure reserve memory for console access, complete the task in this section:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **memory reserved console** *number-of-kilobytes*
4. **exit**
5. **show memory console reserved**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**memory reserved console** *number-of-kilobytes* **Example:** `Router(config)# memory reserved console 512`	Increases the amount of memory reserved for console access. • The *number-of-kilobytes* argument is the amount of memory to be reserved in kilobytes. Valid values are 1 to 4096 KB.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits to privileged EXEC mode.
Step 5	**show memory console reserved** **Example:** `Router# show memory console reserved`	Displays the actual amount of memory that has been reserved.

Examples

The following is sample output from the **showmemoryconsolereserved** command:

```
Router# show memory console reserved
Memory reserved for console is 201400
```

Configuration Examples for Reserve Memory for Console Access

Example Configuring Reserve Memory for Console Access

The following example shows how to increase the reserve memory for console access to 1024 KB:

```
enable
!
configure terminal
!
memory reserved console 1024
end
```

The following example shows how to disable the increase in reserved memory for the console access:

```
enable
!
configure terminal
!
no memory reserved console
end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Cisco IOS configuration commands	*Cisco IOS Configuration Fundamentals Command Reference*
Cisco IOS Configuration Fundamentals configuration tasks and concepts	*Cisco IOS Configuration Fundamentals Configuration Guide*

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified	--

MIBs

MIB	MIBs Link
• No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported, and support for existing RFCs has not been modified.	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

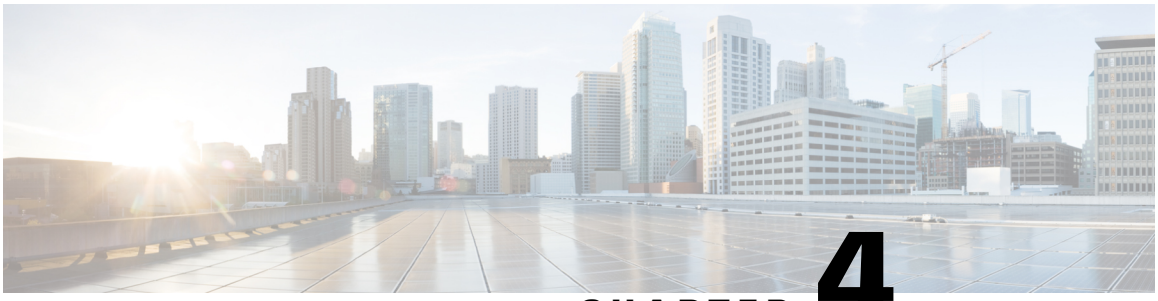
Feature Information for Reserve Memory for Console Access

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

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.

Table 5: Feature Information for Reserve Memory for Console Access

Feature Name	Releases	Feature Information
Reserve Memory for Console Access	12.0(22)S 12.2(28)SB 12.4(15)T	The Reserve Memory for Console Access feature implements command-line interface (CLI) and software enhancements that allow you to reserve sufficient memory to log in to the router console and perform administrative tasks and troubleshooting. These enhancements give administrators the ability to log in to the router in any situation even when the router is running low on memory. The following commands were modified by this feature: **memory reserved console**,**show memory console reserved**.



CHAPTER 4

Zeroization

Zeroization erases all potentially sensitive information in the router memory. This includes the erasure of the main memory, cache memories, and other memories containing packet data, NVRAM, and Flash memory. The Zeroization button on the faceplate is used to invoke zeroization. The parameters for zeroization can be configured, but zeroization cannot be invoked through the command-line interface (CLI).

Zeroization is disabled by default.

Feature History for zeroisation

Release	Modification
12.3(8)YD	This feature was introduced.
12.4(2)T	This feature was integrated into Cisco IOS Release 12.4(2)T.

Finding Support Information for Platforms and Cisco IOS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support. Access Cisco Feature Navigator at http://www.cisco.com/go/fn . You must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Restrictions for Zeroization

- Zeroization is supported on the Cisco 3200 series routers only.
- When zeroization is enabled, the auxiliary (AUX) port should not be used for any function other than an actuator, such as a push button. There is no way to reliably ascertain whether a device connected to the AUX port might trigger zeroization. We recommend that if zeroization is enabled, no devices, with the exception of the zeroization actuator, be attached to the AUX port. There are some AUX port configuration restrictions that apply when zeroization is enabled.
- Zeroization can only be invoked and executed locally. It cannot be invoked and executed remotely through a Telnet session.
- Zeroization shuts down all network interfaces and causes zeroization of the Cisco IOS configuration and object code files, including all IP addresses on the router that are contained in volatile memory.

Information About Zeroization

Scrubbing the Router Memory

Scrubbing is defined as performing several passes through the memory areas, overwriting the memory using a separate data pattern for each pass. The data patterns used for scrubbing consist of separate passes; each pass fills the memory with the following data patterns:

- All ones (that is, 0xffff ffff)
- Alternating ones and zeroes (that is, 0xa5a5 a5a5)
- Alternating zeroes and ones (that is, 0x5a5a 5a5a)
- All zeroes (that is, 0x0000 0000)

The data patterns ensure that

- Each bit in the memory is cleared to zero and set to one at least once.
- The final state of the memory is such that all prior information is erased.

The following items in the router memory are scrubbed:

- Dual-port RAM in the CPM
- Main memory

All the main memory is scrubbed except the memory area containing a small program loop that does the actual scrubbing.

The following items in the router memory cannot be scrubbed:

- Console and AUX port UART FIFO queues. A series of characters is forced through the FIFO queues to ensure that all sensitive information in the FIFO queues is flushed.
- NVRAM, which is erased entirely.
- Flash memory file system, which is erased entirely.
- Caches, which are flushed and invalidated, eliminating all of the information. The process of scrubbing the main memory causes all cache lines to receive the scrubbing data patterns.

Note Some items cannot be completely scrubbed. For example, some devices provide a reset or invalidate of their memory, rather than providing a full data path through which the scrubbing patterns can be written upon memory.