Cisco IOS Software Modularity Command Reference

April 2011

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

Introduction

This document describes the commands used to install and configure Cisco IOS software modularity.

For information about the installation and configuration of Cisco IOS Software Modularity software, see the "Installing and Configuring Cisco IOS Software Modularity" module.

Table 1 shows features or Cisco IOS command-line interface (CLI) commands that are not supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(33)SXH.

Table 1 Commands and Features Not Supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(33)SXH

Command or Feature	Comment
Border Gateway Protocol (BGP) nonstop forwarding (NSF) feature without the BGP graceful restart capability enabled.	If you are using the BGP NSF feature, you must enable the BGP graceful restart capability using the **bgp graceful-restart** command on all neighboring routers and switches.
debug ip tcp transactions	—
Frame-Relay local switching	No Frame Relay local switching support.
ip tcp	The **async-mobility**, **chunk-size**, **queuemax**, and **selective-ack** keywords of the **ip tcp** command are not applicable to Software Modularity because the software uses techniques such as the use of larger send and receive buffer sizes and modifying the queue size where required to improve performance.
IP Version 6 (IPv6) NSF	IPv6 protocols do not support NSF.
Novell Internet Packet Exchange (IPX)-NetWare Link Services Protocol (NLSP)	No IPX-NLSP support.
Routing for Resource Reservation (RRR) feature	No RRR CLI commands are supported.
scripting tcl	—
Some DECnet keywords	No support for the **advertise** and **conversion** keywords.
Some IP Multicast commands	No support for the **ip multicast rpf backoff** and **ip multicast rpf interval** commands due to Reverse Path Forwarding (RPF) changes.
tcp debug	—

Table 1 Commands and Features Not Supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(33)SXH (continued)

Command or Feature	Comment
test platform debugger	—
vrf selection	—
XRemote	No support for the **debug xremote** and **xremote** commands.

Table 2 shows features or Cisco IOS CLI commands that are not supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(18)SXF6, 12.2(18)SXF7 and 12.2(18)SXF8.

Table 2 Commands and Features Not Supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(18)SXF6, 12.2(18)SXF7, and 12.2(18)SXF8

Command or Feature	Comment
BGP NSF feature without the BGP graceful restart capability enabled.	If you are using the BGP NSF feature, you must enable the BGP graceful restart capability using the **bgp graceful-restart** command on all neighboring routers and switches.
debug ip tcp transactions	—
Frame-Relay local switching	No Frame Relay local switching support.
IP SLA features	No IP SLA support.
ip tcp	The **async-mobility**, **chunk-size**, **queuemax**, and **selective-ack** keywords of the **ip tcp** command are not applicable to Software Modularity because the software uses techniques such as the use of larger send and receive buffer sizes and modifying the queue size where required to improve performance.
IP Version 6 (IPv6)	No IPv6 support.
IPX-NLSP	No IPX-NLSP support.
Multiprotocol Label Switching Protocol (MPLS)	No MPLS support.
RRR feature	No RRR CLI commands are supported.
scripting tcl	—
Some DECnet keywords	No support for the **advertise** and **conversion** keywords.
Some IP Multicast commands	No support for the **ip multicast rpf backoff** and **ip multicast rpf interval** commands due to RPF changes.
tcp debug	—
test platform debugger	—
vrf selection	—
XRemote	No support for the **debug xremote** and **xremote** commands.

Table 3 shows features or Cisco IOS CLI commands that are not supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(18)SXF5.

Table 3 *Commands and Features Not Supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(18)SXF5*

Command or Feature	Comment
BGP NSF feature without the BGP graceful restart capability enabled.	If you are using the BGP NSF feature, you must enable the BGP graceful restart capability using the **bgp graceful-restart** command on all neighboring routers and switches.
debug ip tcp transactions	—
Frame-Relay local switching	No Frame Relay local switching support.
IP SLA features	No IP SLA support.
ip tcp	The **async-mobility**, **chunk-size**, **queuemax**, and **selective-ack** keywords of the **ip tcp** command are not applicable to Software Modularity because the software uses techniques such as the use of larger send and receive buffer sizes and modifying the queue size where required to improve performance.
IP Version 6 (IPv6)	No IPv6 support.
IPX-NLSP	No IPX-NLSP support.
MPLS	No MPLS support.
RRR feature	No RRR CLI commands are supported.
scripting tcl	—
Some DECnet keywords	No support for the **advertise** and **conversion** keywords.
Some IP Multicast commands	No support for the **ip multicast rpf backoff** and **ip multicast rpf interval** commands due to RPF changes.
tcp debug	—
test platform debugger	—
vrf selection	—
XRemote	No support for the **debug xremote** and **xremote** commands.

Table 4 shows features or Cisco IOS CLI commands that are not supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(18)SXF4.

Table 4   *Commands and Features Not Supported by Cisco IOS Software Modularity in Cisco IOS Release 12.2(18)SXF4*

Command or Feature	Comment
BGP NSF feature without the BGP graceful restart capability enabled.	If you are using the BGP NSF feature, you must enable the BGP graceful restart capability using the **bgp graceful-restart** command on all neighboring routers and switches.
debug ip tcp transactions	—
Frame-Relay Local Switching	No Frame Relay local switching support.
Interior Gateway Routing Protocol (IGRP)	No IGRP support; upgrade path is Enhanced Interior Gateway Routing Protocol (EIGRP).
IPsec	No IPsec support.
IP SLA features	No IP SLA support.
ip tcp	The **async-mobility**, **chunk-size**, **queuemax**, and **selective-ack** keywords of the **ip tcp** command are not applicable to Software Modularity because the software uses techniques such as the use of larger send and receive buffer sizes and modifying the queue size where required to improve performance.
IP Version 6 (IPv6)	No IPv6 support.
IPX-NLSP	No IPX-NLSP support.
Multiprotocol Label Switching Protocol (MPLS)	No MPLS support.
RRR feature	No RRR CLI commands are supported.
scripting tcl	—
Some DECnet keywords	No support for the **advertise** and **conversion** keywords.
Some IP Multicast commands	No support for the **ip multicast rpf backoff** and **ip multicast rpf interval** commands due to RPF changes.
tcp debug	—
test platform debugger	—
vrf selection	—
XRemote	No support for the **debug xremote** and **xremote** commands.

Cisco IOS Software Modularity Commands

archive tar

To create a TAR file, to list the files in a TAR file, or to extract the files from a TAR file, use the **archive tar** command in privileged EXEC mode.

> **archive tar** {**/create** *destination-url* **flash:**/*file-url* | **/table** *source-url* | **/xtract** *source-url* **flash:**/*file-url* [*dir/file...*]}

Syntax Description		
/create *destination-url* **flash:**/*file-url*	Creates a new TAR file on the local or network file system.	
	For *destination-url*, specify the destination URL alias for the local or network file system and the name of the TAR file to create. The following options are supported:	
	• **flash:**—Syntax for the local flash file system.	
	• **ftp:**[[//*username*[**:***password*]@*location*]/*directory*]/*tar-filename.tar*— Syntax for FTP.	
	• **rcp:**[[//*username*@*location*]/*directory*]/*tar-filename.tar*—Syntax for Remote Copy Protocol (RCP).	
	• **tftp:**[[//*location*]/*directory*]/*tar-filename.tar* —Syntax for TFTP.	
	The *tar-filename.tar* is the name of the TAR file to be created.	
	For **flash:**/*file-url*, specify the location on the local flash file system from which the new TAR file is created.	
	An optional list of files or directories within the source directory can be specified to write to the new TAR file. If none is specified, all files and directories at this level are written to the newly created TAR file.	
/table *source-url*	Displays the contents of an existing TAR file to the screen.	
	For *source-url*, specify the source URL alias for the local or network file system. The following options are supported:	
	• **flash:**—Syntax for the local flash file system.	
	• **ftp:**[[//*username*[**:***password*]@*location*]/*directory*]/*tar-filename.tar*— Syntax for FTP.	
	• **rcp:**[[//*username*@*location*]/*directory*]/*tar-filename.tar*—Syntax for Remote Copy Protocol (RCP).	
	• **tftp:**[[//*location*]/*directory*]/*tar-filename.tar* —Syntax for TFTP.	
	The *tar-filename.tar* is the name of the TAR file to be created.	

/xtract *source-url* **flash:***/file-url* [*dir/file...*]	Extracts files from a TAR file to the local file system.
	For *source-url*, specify the source URL alias for the local file system. The following options are supported:
	• **flash:**—Syntax for the local flash file system.
	• **ftp:**[[*//username*[**:***password*]*@location*]*/directory*]*/tar-filename.tar*— Syntax for FTP.
	• **rcp:**[[*//username@location*]*/directory*]*/tar-filename.tar*—Syntax for Remote Copy Protocol (RCP).
	• **tftp:**[[*//location*]*/directory*]*/tar-filename.tar* —Syntax for TFTP.
	The *tar-filename.tar* is the name of the TAR file to be created.

Command Default A TAR archive file is not created.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.1(13)AY	This command was introduced.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
12.4(22)YB	This command was integrated into Cisco IOS Release 12.4(22)YB.
12.4(24)T	This command was integrated into Cisco IOS Release 12.4(24)T.

Usage Guidelines Filenames, directory names, and image names are also case sensitive.

The TAR file is an archive file from which you can extract files by using the **archive tar** command.

Examples The following example shows how to create a TAR file. The command writes the contents of the new-configs directory on the local flash device to a file named saved.tar on the TFTP server at 172.20.136.9.

```
Switch# archive tar /create tftp:172.20.136.9/saved.tar flash:/new-configs
```

The following example shows how to display the contents of the c2940-tv0-m.tar file that is in flash memory. The contents of the TAR file appear on the screen.

```
Switch# archive tar /table flash:c2940-tv0-m.tar

info (219 bytes)
c2940-tv0-mz-121/ (directory)
c2940-tv0-mz-121/html/ (directory)
c2940-tv0-mz-121/html/foo.html (0 bytes)
c2940-tv0-mz-121/vegas-tv0-mz-121.bin (610856 bytes)
c2940-tv0-mz-121/info (219 bytes)
info.ver (219 bytes)
```

The following example shows how to extract the contents of a TAR file on the TFTP server at 172.20.10.30. This command extracts only the new-configs directory into the root directory on the local flash file system. The remaining files in the saved.tar file are ignored.

```
Switch# archive tar /xtract tftp:/172.20.10.30/saved.tar flash:/ new-configs
```

clear raw statistics

To clear raw IP statistics when Cisco IOS Software Modularity software is running, use the **clear raw statistics** command in privileged EXEC mode.

> **clear raw statistics**

Syntax Description   This command has no arguments or keywords.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   There are three transport protocols used when Software Modularity software is running: Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and raw IP. The transport protocol statistics are generally counters, but some are averages or time stamps. Use the **clear raw statistics** command to reset the raw IP statistics, and use the **show raw statistics** command to display the raw IP statistics. Many of the statistics are relevant to all of the transport protocols. To clear the other transport protocol statistics used in Software Modularity, use the **clear tcp statistics** and **clear udp statistics** commands.

Examples   The following example shows how to clear the raw IP statistics using the **clear raw statistics** command:

```
Router# clear raw statistics

[confirm]
```

Related Commands

Command	Description
clear tcp statistics	Clears TCP statistics.
clear udp statistics	Clears UDP statistics.
show raw statistics	Displays raw IP statistics.

clear udp statistics

To clear User Datagram Protocol (UDP) statistics when Cisco IOS Software Modularity software is running, use the **clear udp statistics** command in privileged EXEC mode.

clear udp statistics

Syntax Description   This command has no arguments or keywords.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   There are three transport protocols used when Software Modularity software is running: Transmission Control Protocol (TCP), UDP, and raw IP. The transport protocol statistics are generally counters, but some are averages or time stamps. Use the **clear udp statistics** command to reset the UDP statistics, and use the **show udp statistics** command to display the UDP statistics. Many of the statistics are relevant to all of the transport protocols. To clear the other transport protocol statistics used in Software Modularity, use the **clear raw statistics** and **clear tcp statistics** commands.

Examples   The following example shows how to clear the UDP statistics using the **clear udp statistics** command:

```
Router# clear udp statistics

[confirm]
```

Related Commands

Command	Description
clear raw statistics	Clears raw IP statistics.
clear tcp statistics	Clears TCP statistics.
show udp statistics	Displays UDP statistics.

debug registry

To turn on the debugging output for registry events or errors when Cisco IOS Software Modularity software is running, use the **debug registry** command in privileged EXEC mode. To turn off debugging output, use the **no** form of this command or the **undebug** command.

> **debug registry** {**events** | **errors**} [*process-name* | *pid*]

> **no debug registry** {**events** | **errors**} [*process-name* | *pid*]

Syntax Description

events	Displays debugging messages about registry event messages.
errors	Displays debugging messages about registry error messages.
process-name	(Optional) Process name.
pid	(Optional) Process ID. Number in the range from 1 to 4294967295.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   Use the **debug registry** command to troubleshoot Software Modularity registry operations.

⚠
Caution   Use any debugging command with caution because the volume of generated output can slow or stop the router operations. We recommend that this command be used only under the supervision of a Cisco engineer.

Examples   The following example turns on debugging messages for Software Modularity registry events for the TCP process:

```
Router# debug registry events tcp.proc

Debug registry events debugging is on
```

The following example turns on debugging messages for Software Modularity registry errors:

```
Router# debug registry errors

Debug registry errors debugging is on
```

exception core

To set or change the core dump options for a Cisco IOS Software Modularity process, use the **exception core** command in global configuration mode. To reset the core dump options to their default settings, use the **no** form of this command.

> **exception core** *process-name* {{**off** | **mainmem** | **mainmem-sharedmem** | **mainmem-text** | **mainmem-text-sharedmem** | **sharedmem** [**maxcore** *value*]}| **maxcore** *value*}

> **no exception core** *process-name*

Syntax Description

process-name	Process name.
off	When the process stops, no core dump is taken.
mainmem	When the process stops, the main memory is dumped.
mainmem-sharedmem	When the process stops, the main memory and the shared memory segments are dumped.
mainmem-text	When the process stops, the main memory text segment is dumped.
mainmem-text-sharedmem	When the process stops, the main memory text and shared memory segments are dumped.
sharedmem	When the process stops, the shared memory segments are dumped.
maxcore	(Optional) Specifies a maximum number of dumps allowed for this process.
value	(Optional) Integer from 0 to 4294967295. By default there is no limit.

Command Default Default core dump options are set for a process.

Command Modes Global configuration (config)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines Core dumps are taken when every process crashes. Each Cisco IOS Software Modularity software component has an associated .startup file that determines the core dump options (and other attributes) of that process. Use the **show processes detailed** command to display the core dump options for a process. Use the **exception core** command to override the default values set in the .startup file for the specific software component.

Note This command is of use only to Cisco technical support representatives in analyzing system failures in the field. Under normal circumstances, there should be no reason to change the core dump options. For that reason, this command should be used only by Cisco Certified Internetwork Experts (CCIEs) or under the direction of Cisco Technical Assistance Center (TAC) personnel.

Examples   In the following example, the maximum number of core dumps for all instances of the Cisco Discovery Protocol (CDP) process is set to 100. The command also limits the core dump output to the main memory text segments.

```
configure terminal
 exception core cdp2.proc mainmem-text maxcore 100
```

Related Commands

Command	Description
exception flash	Configures the dump location for core files when a process reloads.
show processes detailed	Displays detailed process information.

exception core-file

To specify the name of the core dump file in Cisco IOS or Cisco IOS Software Modularity software, use the **exception core-file** command in global configuration mode. To return to the default core filename, use the **no** form of this command.

Cisco IOS Software

> **exception core-file** *filename*

> **no exception core-file**

Cisco IOS Software Modularity

> **exception core-file** [*filename*] [**limit** *upper-limit*] [**compress**] [**timestamp**]

> **no exception core-file**

Syntax Description

filename	Name of the core dump file saved on the server.
	(Optional) In Software Modularity images, if this argument is not specified, the default core file is named using the name of the process that is being dumped. For example, if the raw_ip.proc is the process that is being dumped, then the default core file is named raw_ip.proc.
limit	(Optional) For Cisco IOS Software Modularity images only. Specifies an upper limit of a range so that core dumps of more than one process can be created without overwriting the previous core dump.
upper-limit	(Optional) For Cisco IOS Software Modularity images only. Number, in the range from 1 to 64, that represents the upper limit.
compress	(Optional) For Cisco IOS Software Modularity images only. Turns on dump file compression. By default, compression is turned off.
timestamp	(Optional) For Cisco IOS Software Modularity images only. Adds a time stamp to the core dump file.

Command Default

Cisco IOS Software: The core file is named *hostname*-core, where *hostname* is the name of the router. Cisco IOS Software Modularity: The core file is named using the name of the process that is being dumped.

Command Modes

Global configuration (config)

Command History

Release	Modification
10.2	This command was introduced.
12.2(18)SXF4	The **limit**, **compress**, and **timestamp** keywords were added to support Software Modularity images.

Usage Guidelines   If you use TFTP to dump the core file to a server, the router will only dump the first 16 MB of the core file. If the router's memory is larger than 16 MB, the whole core file will not be copied to the server. Therefore, use rcp or FTP to dump the core file. The network dump is not supported in Software Modularity images.

⚠
Caution   This command is of use only to Cisco technical support representatives in analyzing system failures in the field. Under normal circumstances, there should be no reason to change the default core filename. For that reason, this command should be used only by Cisco Certified Internetwork Experts (CCIEs) or under the direction of Cisco Technical Assistance Center (TAC) personnel.

Examples   **Cisco IOS Software**

In the following example, the router is configured to use FTP to dump a core file named dumpfile to the FTP server at 172.17.92.2 when the router crashes:

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

Cisco IOS Software Modularity

In the following example, the router is configured to dump the main memory used by the TCP process to a file named dump-tcp when the TCP process crashes. The dump file is configured with an upper limit of 20, to be compressed, and to have a time stamp applied.

```
exception core tcp.proc mainmem
exception core-file dump-tcp limit 20 compress timestamp
```

✎
Note   The **exception protocol** and **exception dump** commands are not supported in Software Modularity images.

Related Commands

Command	Description
exception core	Sets or changes the core dump options for a Cisco IOS Software Modularity process.
exception dump	Causes the router to dump a core file to a particular server when the router crashes.
exception memory	Causes the router to create a core dump and reboot when certain memory size parameters are violated.
exception protocol	Configures the protocol used for core dumps.
exception spurious-interrupt	Causes the router to create a core dump and reload after a specified number of spurious interrupts.
ip ftp password	Specifies the password to be used for FTP connections.
ip ftp username	Configures the username for FTP connections.

exception crashinfo buffersize

To change the size of the buffer used for crashinfo files, use the **exception crashinfo buffersize** command in global configuration mode. To revert to the default buffer size, use the **no** form of this command.

exception crashinfo buffersize *kilobytes*

no exception crashinfo buffersize *kilobytes*

Syntax Description

kilobytes	Buffer size, in kilobytes (KB). Range is 32 to 256. Default is 32.

Command Default Crashinfo buffer is 32 KB.

Command Modes Global configuration (config)

Command History

Release	Modification
12.2(4)T, 12.2(11)	This command was introduced for the Cisco 3600 series only (3620, 3640, and 3660 platforms).
12.2(13)T	This command was implemented in Cisco 6400-NSP images.
12.2(15)JA	This command was integrated into Cisco IOS Release 12.2(15)JA.
12.2(18)SXF4	This command was integrated into Release 12.2(18)SXF4 to support Software Modularity images.

Usage Guidelines The crashinfo file saves information that helps Cisco technical support representatives to debug problems that caused the Cisco IOS image to fail (crash). The device writes the crash information to the console at the time of the failure, and the file is created the next time you boot the Cisco IOS image after the failure (instead of while the system is failing).

Note If you are running a Software Modularity image, setting the crashinfo buffer size to the default of 32 KB does not limit the crashinfo buffer size. The crashinfo file size is limited to the value set if the value is set to anything other than the default 32 KB.

Examples In the following example, the crashinfo buffer is set to 100 KB:

```
Router(config)# exception crashinfo buffersize 100
```

Related Commands

Command	Description
exception crashinfo file	Enables the creation of a diagnostic file at the time of unexpected system shutdowns.

exception flash

To handle the device and erase permission for exceptions, and set the local dump location for core files when a process reloads, use the **exception flash** command in global configuration mode.

> **exception flash** {**all** | **iomem** | **procmem**} *device-name*

> **no exception flash**

Syntax Description		
all		Dumps all the memory in the local dump location.
device-name		Device name to be used as the local dump location.
iomem		Dumps the input and output memory in the local dump location.
procmem		Dumps the processor memory in the local dump location.

Command Default   No core dump location is set for a process.

Command Modes   Global configuration (config)

Command History	Release	Modification
	12.2(18)SXF4	This command was introduced to support Software Modularity images.
	15.0(1)M	This command was modified in a release earlier than Cisco IOS Release 15.0(1)M. The **all**, **iomem**, and **procmem** keywords were added.

Usage Guidelines   Core dumps are taken when every process reloads. You can configure up to three destinations, and the order in which the dump locations are used follows the order in which the destinations are configured.

Each Cisco IOS Software Modularity component has an associated .startup file that determines the core dump options (and other attributes) of that process. Use the **show processes detailed** command to display the core dump options for a process. Use the **exception core** command to override the default values set in the .startup file for the specific software component.

⚠
Caution   This command is of use only to Cisco technical support representatives in analyzing system failures in the field. Under normal circumstances, there should be no reason to set a local core dump location for a process. For that reason, this command should be used only by Cisco Certified Internetwork Experts (CCIEs) or under the direction of Cisco Technical Assistance Center (TAC) personnel.

Examples   In the following example, three dump locations are configured to dump all the memory:

```
Router# configure terminal
Router(config)# exception flash all disk1:
Router(config)# exception flash all bootflash:
Router(config)# exception flash all sup-bootdisk:
```

Related Commands	Command	Description
	exception core	Sets or changes the core dump options for a Cisco IOS Software Modularity process.
	show processes detailed	Displays detailed process information.
	write core	Generates a process core dump.

exception kernel

To configure a networking device to dump the kernel memory, use the **exception kernel** command in global configuration mode. To turn off the kernel dump facility, use the **no** form of this command.

> **exception kernel** [**filename** *filename*] **filepath** *path* [**memory kernel**]

> **no exception kernel**

Syntax Description

filename	(Optional) Specifies the name of the kernel dump file.
filename	(Optional) Name of the kernel dump file. Because this file is a compressed file, a .Z suffix is added to the name. By default, the filename is kernel_core.Z.
filepath	Specifies the location to which the core dump file is written.
path	Location to which the core dump file is written. The supported locations are **bootflash:** or **disk***n***:**. For **disk***n***:** or **bootflash:**, the *path* value is the absolute path to the file.
memory	(Optional) Specifies the type of memory to be dumped.
kernel	(Optional) Specifies that only kernel memory is to be dumped. If not specified, both user memory and kernel memory are dumped.

Command Default   No kernel memory is dumped.

Command Modes   Global configuration (config)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   Use the **exception kernel** command to dump kernel memory when the kernel reloads. Please note that this is different from process dump, in which a process on the networking device reloads, but not the networking device itself. This command is used to configure where and what to dump. If the dump is to **bootflash:**, this command is all that is required.

For distributed networking devices, the line card number is added to the default name assigned to the kernel core dump file. For example, the default kernel core dump file for the line card in slot 6 would be kernel_core6.Z.

⚠
Caution   This command is of use only to Cisco technical support representatives in analyzing system failures in the field. Under normal circumstances, there should be no reason to dump the kernel memory. For that reason, this command should be used only by Cisco Certified Internetwork Experts (CCIEs) or under the direction of Cisco Technical Assistance Center (TAC) personnel.

Examples   The following example writes kernel exceptions to the disk0:/core directory. Only kernel memory is dumped, and because no filename is specified, the kernel core dump file is given the default name kernel_core.z.

```
configure terminal
 exception kernel filepath /disk0:/core memory kernel
```

exception switch kernel

To configure a networking device to dump the kernel memory, use the **exception kernel** command in global configuration mode. To turn off the kernel dump facility, use the **no** form of this command.

>   **exception switch kernel filesystem** *filename*

>   **no exception switch kernel filesystem** *filename*

Syntax Description

filesystem	Specifies the file system for placing the kernel core file.
filename	(Optional) Name of the kernel core file. Because this file is a compressed file, a .Z suffix is added to the name. By default, the filename is kernel_core.Z.

Command Default   No kernel memory is dumped.

Command Modes   Global configuration (config)

Command History

Release	Modification
12.2(33)SXI4	This command was introduced to support software modularity images.
12.2(33)SXI5	This command is not supported in Cisco IOS Release 12.2(33)SXI5 and later releases.

Usage Guidelines

⚠

Caution   This command is useful only to Cisco technical support representatives for analyzing system failures in the field. Under normal circumstances, you should not need to dump the kernel memory. For that reason, this command should be used only by Cisco Certified Internetwork Experts (CCIEs), or under the direction of Cisco Technical Assistance Center (TAC) personnel.

The **exception switch kernel** command is available in Cisco IOS Releases 12.2(33)SXI4 and 12.2(33)SXI4a

Use the **exception switch kernel** command to dump kernel memory when the kernel reloads on the SP. This operation is different from process dump in which a process on the networking device reloads, but not the networking device itself. This command is used to configure where and what to dump. If the dump is to **bootflash:**, the **exception switch kernel** command is all that is required.

The **filepath** keyword only accepts file systems available to the SP. Use this command for configuring modular Cisco IOS kernel core files on the SP.

For distributed networking devices, the line card number is added to the default name assigned to the kernel core dump file. For example, the default kernel core dump file for the line card in slot 6 would be kernel_core6.Z.

Examples   The following example writes kernel exceptions to the disk0:/core directory. Only kernel memory is dumped, and because no filename is specified, the kernel core dump file is given the default name kernel_core.z.

```
configure terminal
 exception switch kernel filesystem /disk0:/core memory kernel
```

Related Commands

Command	Description
exception kernel	Configures module Cisco IOS kernel core files on the RP.

install activate

> ✎
>
> **Note**   Effective with Cisco IOS Release 12.2(33)SXI3, the install activate command is not available in Cisco IOS software.

To activate the current pending change set, use the **install activate** command in privileged EXEC mode.

> **install activate** *search-root-directory* [**reload**]

Syntax Description

search-root-directory	Local directory specified in the *destination-directory* argument of a previously executed **install file** command. Valid root directories are /sys, /oldsys, and /newsys.
reload	(Optional) Treats the patch to be activated as a reload patch, thereby bypassing a time-consuming process restart.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI1	This command was modified. The **reload** keyword was added.
12.2(33)SXI3	This command was removed.

Usage Guidelines   Use the **install activate** command after a patch file or maintenance pack (MP) has been installed. The state of files in the pending change set will change depending on whether a reload is required.

Cisco IOS Software Modularity introduces the concept of installed software that is different from just booting an image on the networking device. Cisco IOS Software Modularity images can be saved into the flash file system and booted like a Cisco IOS image, but this is referred to as uninstalled software. To gain the benefits of the Cisco IOS Software Modularity Installer and permit patch files to be installed, use the **install file** command to write the software to flash. Installation and activation are now separate processes. The **install bind** command is used to bind Cisco IOS Software Modularity base images system-wide; and the **install activate** command must be entered to activate a patch. Some patches will require a reload to be performed, and a message appears on the console after the **install activate** command has been entered to note the current state of the patch.

Table 5 shows whether the patch code is running in the various patch states. For more details about activating a patch, including a flowchart of the various patch states, see the "Cisco IOS Software Modularity Installation and Configuration" module.

Table 5   *Patch State Descriptions*

State	State Description	Is Patch Code Running?
PendInst	Pending installation activation.	No processes are running the patch code.
InstPRel	Installation activation pending reload.	No processes are running the patch code until a card reload is performed.
IPRPndRo	Installation activation pending reload pending rollback.	No processes are running the patch code until a card reload is performed.
PendRoll	Pending rollback	Some processes are running the patch code.
RollPRel	Rollback pending reload.	Some processes are running the patch code.
RPRPndIn	Rollback pending reload pending installation activation.	Some processes are running the patch code.
Active	Patch is active.	Some processes are running the patch code.
Pruned	Patch is removed.	No processes are running the patch code.

Examples

The following example shows how to activate the current pending change set for the sys directory:

```
Router# install activate disk0:/sys
```

Related Commands

Command	Description
install bind	Binds Cisco IOS Software Modularity images.
install file	Installs base system files and patches.

install bind

> **Note** Effective with Cisco IOS Release 12.2(33)SXI3, the **install bind** command is not available in Cisco IOS software.

To bind a Cisco IOS Software Modularity software image system-wide, use the **install bind** command in global configuration mode. To remove the Software Modularity software binding, use the **no** form of this command.

> **install bind** *search-root-directory* [**prepend**]

> **no install bind**

Syntax Description

search-root-directory	Directory to be bound as specified in the *destination-directory* argument of a previously executed **install file** command.
prepend	(Optional) Moves the latest boot system statement to the top of the boot variable, which makes that statement the primary image to boot.

Command Default The Cisco IOS Software Modularity software image is not bound.

Command Modes Global configuration (config)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI1	This command was modified. The **prepend** keyword was added.
12.2(33)SXI3	This command was removed.

Usage Guidelines The **install bind** command generates a **boot system** command, but the **install bind** command is not inserted into the configuration. The benefit in using the **install bind** command is that you just specify the search root directory, which is the destination directory used in the **install file** command, and the Cisco IOS Software Modularity software will determine the directory structure and image file. If you use the **boot system** command, you must enter the complete directory path and image name.

Each instance of the **boot system** command generated by an **install bind** command is saved in the configuration file in the order in which it was configured, which is the normal behavior for **boot system** commands. To configure a system to have the newly installed Software Modularity image as the primary image to boot, you must remove all previous **boot system** commands in the configuration and enter them in the order in which you want them to run. Alternatively, you can download the startup configuration to a text file, insert the new **install bind** or **boot system** command, and copy the changes back into the startup configuration.

To remove all **boot system** commands from the configuration file, use the **no** form of the **boot system** command without any arguments. Using the **no** form of the **install bind** command will remove only the **boot system** commands for installed software and leave other **boot system** commands intact.

Note   Use the **install bind** command to bind one or more Software Modularity images, and then copy the changes to the startup configuration file. Be aware that an image reload or switchover must be performed before the installed and bound image is actually running on the device.

Examples   The following example shows how to remove all existing **boot system** commands and to bind the Software Modularity image in the directory named sys:

```
Router# configure terminal
Router(config)# no boot system
Router(config)# install bind disk0:/sys
Router(config)# exit
Router# copy running-config startup-config
```

Related Commands

Command	Description
boot system	Specifies the system image that the router loads at startup.
install file	Installs base system files and patches.

install clear

> ✎
>
> **Note**   Effective with Cisco IOS Release 12.2(33)SXI3, the **install clear** command is not available in Cisco IOS software.

To remove an entire installed software system, use the **install clear** command in privileged EXEC mode.

> **install clear** *search-root-directory*

Syntax Description

search-root-directory	Local directory specified in the *destination-directory* argument of a previously executed **install file** command. Valid root directories are /sys, /oldsys, and /newsys.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI3	This command was removed.

Usage Guidelines   Use the **install clear** command with caution because the command cannot be reversed. After an installation is cleared, it cannot be undone. Software that is currently running or that has been bound to run cannot be cleared. For bound software, you must remove the binding with the **no install bind** command before using the **install clear** command.

Examples   The following example shows how to clear the system installed in the local directory named sys:

```
Router# install clear disk0:/sys
```

Related Commands

Command	Description
install bind	Binds Cisco IOS Software Modularity images.
install file	Installs base system files and patches.

install commit

✎

Note Effective with Cisco IOS Release 12.2(33)SXI3, the **install commit** command is not available in Cisco IOS software.

To define a tag name for a set of Cisco IOS Software Modularity software installed in the destination directory of a previously executed **install file** command, use the **install commit** command in privileged EXEC mode.

install commit *search-root-directory tag-name*

Syntax Description

search-root-directory	Local directory specified in the *destination-directory* argument of a previously executed **install file** command.
tag-name	String of characters to identify a set of software installed in the *search-root-directory* value.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI3	This command was removed.

Usage Guidelines This command creates a point to which a user can roll back a system after a patch is installed that is considered unsatisfactory. The *tag-name* argument provides a name for the point. A tag name must be unique to the local file system.

Use the **install prune** command to remove a previously defined tag from the installed software.

Examples The following example shows how to define a tag named tag1 to identify the software installed in the local directory named sys:

```
Router# install commit disk0:/sys tag1
```

Related Commands

Command	Description
install file	Installs base system files and patches.
install prune	Removes a tag from the software installed in a directory specified in a previously executed **install file** command.

install copy

✎

Note Effective with Cisco IOS Release 12.2(33)SXI3, the **install copy** command is not available in Cisco IOS software.

To make a copy of the Cisco IOS Software Modularity software, use the **install copy** command in privileged EXEC mode.

install copy *source-root-directory destination-root-directory*

Syntax Description

source-root-directory	Local directory specified in a previously executed **install file** command. Valid root directories are /sys, /oldsys, and /newsys.
destination-root-directory	Local root directory. Valid root directories are /sys, /oldsys, and /newsys.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI3	This command was removed.

Usage Guidelines Use the **install copy** command to duplicate the Cisco IOS Software Modularity software at the source directory and place it at the destination directory. Both the source and destination directories must be local to the device.

Examples The following example shows how to copy the software in the directory named sys into a directory named oldsys:

```
Router# install copy disk0:/sys disk0:/oldsys
```

Related Commands

Command	Description
install file	Installs base system files and patches.

install file

✎

Note Effective with Cisco IOS Release 12.2(33)SXI3, the **install file** command is not available in Cisco IOS software.

To install Cisco IOS Software Modularity base system files and patches, use the **install file** command in privileged EXEC mode.

install file *source-file-url destination-directory* [*second-destination-directory*] [**interactive**]

Syntax Description

source-file-url	Path of an installable file that contains the code to be installed. The installable file may be on a local file system or on a remote file system.
destination-directory	Path of the destination directory in which the installable file is to be installed. The destination directory must be on a local file system and be in the following format: *file-system*:/{sys | newsys | oldsys}.
second-destination-directory	(Optional) Path of a secondary destination directory in which the installable file is to be installed. The secondary destination directory must be on a local file system and be in the following format: *file-system***:/{sys** | **newsys** | **oldsys}**.
interactive	(Optional) Enables prompting of the user before certain actions and activates more detailed output during the installation process.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI1	This command was modified. The optional *second-destination-directory* argument was added.
12.2(33)SXI3	This command was removed.

Usage Guidelines Use the optional **interactive** keyword to display more detailed output during the installation. Messages indicating current tasks that are being performed during the installation may be displayed. The default output is a series of ! characters to indicate progress and a message at the end indicating success or failure.

Cisco IOS Software Modularity introduces the concept of installed software that is different from just booting an image on the networking device. Cisco IOS Software Modularity images can be saved into the flash file system and booted like a Cisco IOS image, but this is referred to as uninstalled software. To gain the benefits of the Cisco IOS Software Modularity Installer and permit patch files to be installed, use the **install file** command to write the software to local storage. Installation and activation are now separate processes; and the **install activate** command must be entered to activate patches. Some patches will require a reload to be performed, and a message appears on the console after the **install activate** command has been entered to note the current state of the patch.

Use the **show install** command to display information about the currently installed software. Use the **install clear** command to remove an entire installed software system, or use the **install rollback** command to remove specific patches installed on top of the software version.

Examples

The following example shows how to install two different files from two different paths into the same local directory:

```
Router# install file tftp://username@hostname//directory/c6kpatch-vz disk0:/sys
Router# install file rcp://s72033/base/s72033-adventerprisek9_wan_dbg-vz disk0:/sys
```

Related Commands

Command	Description
install activate	Activates the current pending change set.
install clear	Removes an entire installed software system.
install rollback	Rolls back the installed Cisco IOS Software Modularity software to the point at which a tag was defined.
show install	Displays information about the installed software.

Cisco IOS Software Modularity Command Reference ■

install move

✎

Note Effective with Cisco IOS Release 12.2(33)SXI3, the **install move** command is not available in Cisco IOS software.

To move the Cisco IOS Software Modularity software from a source URL to a destination URL, use the **install move** command in privileged EXEC mode.

install move *source-root-directory destination-root-directory*

Syntax Description

source-root-directory	Local directory specified in a previously executed **install file** command. Valid root directories are /sys, /oldsys, and /newsys.
destination-root-directory	Local root directory. Valid root directories are /sys, /oldsys, and /newsys.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI3	This command was removed.

Usage Guidelines Use the **install move** command to copy the Cisco IOS Software Modularity software from a source directory to a destination directory and then remove the software from the source directory. Both the source and destination directories must be local.

Examples The following example shows how to move the software from the directory named /sys to the directory named /oldsys. The software will be removed from the /sys directory.

```
Router# install move disk0:/sys disk0:/oldsys
```

Related Commands

Command	Description
install file	Installs base system files and patches.

install prune

> **Note** Effective with Cisco IOS Release 12.2(33)SXI3, the **install prune** command is not available in Cisco IOS software.

To remove a tag or unused files from the software that is installed in the destination directory specified in a previously executed **install file** command, use the **install prune** command in privileged EXEC mode.

install prune *search-root-directory tag-name* [**files**]

Syntax Description

search-root-directory	Directory specified in the *destination-directory* argument of a previously executed **install file** command.
tag-name	String of characters to identify a set of software as previously defined by the **install commit** command.
files	(Optional) Cleans and removes all unused and nonactive files from the base image to the tag specified by the *tag-name* argument. The tag specified by the *tag-name* attribute is not removed.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(18)SXF8	The **files** keyword was added.
12.2(33)SXI3	This command was removed.

Usage Guidelines In addition to removing the tag from the installed software, the **install prune** command removes any files that are no longer required by the system as a result of the tag removal. After this command is executed, rollback can be performed to any previously installed tag.

When this command is executed using the optional **files** keyword, all of the tags from the base image to the tag specified are removed except for the specified tag. After this command is entered with the optional **files** keyword, rollback cannot be done to any tag beyond the specified tag; rollback can be performed to the base image only.

Examples The following example shows how to remove the tag named tag1 from the installed software.

```
Router# install prune disk0:/sys tag1
```

The following example shows how to remove all of the tags from the base image up to tag1. Tag1 is not removed.

```
Router# install prune disk0:/sys tag1 files
```

Cisco IOS Software Modularity Commands

Related Commands	Command	Description
	install commit	Defines a tag for a set of software installed by the **install file** command.
	install file	Installs base system files and patches.

SM-34

April 2011

install repackage

> ✎
>
> **Note** Effective with Cisco IOS Release 12.2(33)SXI3, the **install repackage** command is not available in Cisco IOS software.

To create an installation or backup installable file from an installed system when a Cisco IOS Software Modularity image is running, use the **install repackage** command in privileged EXEC mode.

> **install repackage** *source-root-directory destination-file-url* [**compress**]

Syntax Description

source-root-directory	Local directory specified in a previously executed **install file** command.
destination-file-url	Local or remote URL that specifies the path and name of the destination file to which the installable file is written.
compress	(Optional) Indicates that the generated installable file is to be compressed.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI3	This command was removed.

Usage Guidelines To allow for easier deployment of a base image and several patches to multiple routers, an installable bundled image, referred to as a repackage, can be replicated. Use the **install repackage** command to generate a installable file from an installed system. The installable file can be used in an installation on another device or as a backup installation for the current device. While the image is being replicated, the Software Modularity Installer saves everything in the installed state including rollback tags. An initial boot must be performed on the device on which the replicated image is to be installed. The ability to create a repackage allows standard installations to be performed across the network and saves installation time.

Examples The following example shows how to create an installation or backup file named s72033-finance-vm.repackage from an installed system:

```
Router# install repackage disk0:/sys disk0:/s72033-finance-vm.repackage
```

Related Commands

Command	Description
install file	Installs base system files and patches.

install rollback

> **Note** Effective with Cisco IOS Release 12.2(33)SXI3, the **install rollback** command is not available in Cisco IOS software.

To roll back the installed Cisco IOS Software Modularity software to the point at which a tag was defined, use the **install rollback** command in privileged EXEC mode.

install rollback *search-root-directory tag-name*

Syntax Description

search-root-directory	Directory specified in the *destination-directory* argument of a previously executed **install file** command. Valid root directories are /sys, /oldsys, and /newsys.
tag-name	String of characters to identify a set of software as previously defined by the **install commit** command.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXI3	This command was removed.

Usage Guidelines   Similar to the idea of a database rollback, Cisco IOS Software Modularity images can roll back to a set of installed files defined by a tag. The installed system is captured at a point in time by defining a tag using the **install commit** command. If a subsequent installation of a patch file adversely affects the installed system, a rollback can be performed using the defined tag. The **install activate** command must be entered after the **install rollback** command to activate the rollback. All installation actions performed since the tag was defined are deleted, and the processes affected by the rollback of installed software are restarted after the rollback is activated. After the restart, these processes use the software that was present at the time the tag was created. Tags can be deleted, and the system will remove all installation files that will now never be used because the tag has been removed.

Examples   The following example shows how to roll back the software to the time that tag1 was defined and then restart all the affected processes. The tag named tag1 is assumed to have been created using the install commit command in an earlier configuration.

```
Router# install rollback disk0:/sys tag1
Router# install activate disk0:/sys
```

Related Commands

Command	Description
install activate	Activates the current pending change set for Cisco IOS Software Modularity images.
install commit	Defines a tag for a set of software installed by the **install file** command.
install file	Installs base system files and patches.

process restart

To terminate and restart a process when a Cisco IOS Software Modularity image is running, use the **process restart** command in privileged EXEC mode.

process restart *process-name* [**:***instance-id*] [**cold**]

Syntax Description

process-name	Process name.
	Note Only processes that are controlled by the System Manager can be restarted.
:*instance-id*	(Optional) Process number. The first process is numbered 1, and this is the default if no number is specified. The colon is required.
cold	(Optional) Specifies a cold restart.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(18)SXF5	This command was enhanced to display console and error messages about possible configuration losses at restart.

Usage Guidelines   The **process restart** command can be used to restart a newly installed version of an executable. Under special circumstances, it can also be used to restart a process that is operating in suboptimal mode. Only processes that are controlled by the System Manager can be restarted.

When restarting, a process will retrieve the previous state information from the saved configuration checkpoint. A cold restart means that the process will delete the previous state information from the saved configuration checkpoint.

If the **process restart** command is entered without first saving the active running configuration session and checkpointing the configuration changes, the changes could be lost. The following console warning about this possible configuration loss is displayed:

```
Some config has not yet been checkpointed and may be lost. It is recommended to do a
'write checkpoint' to checkpoint the config and re-start the process. Do you want to
continue ? [no]:
```

If you restart the process, a message similar to the following is displayed:

```
Restarting process iprouting.iosproc

02:51:21: %kern-6-SYSLOG_GEN: <30>:02:51:21:;1144354584.745:
sysmgr.proc[72]: Some config for process iprouting.iosproc:1 has not yet been
checkpointed and may be lost
```

To checkpoint the configuration, use the **write checkpoint** command. Some commands also checkpoint internally upon being entered, such as the **write memory** command, the **copy running-config startup-config** command and the **show running-config** command.

In Software Modularity, you cannot restart a process on the standby router. The standby router console is disabled by default. If you enable the standby router console, and then enter the **process restart** command to restart a process, the standby console will reload and display one of the following error messages:

```
Standby process exited, rebooting.
```

or

```
This process is not known to sysmgr.
```

Examples   The following example restarts the Cisco Discovery Protocol (CDP) process:

```
Router# process restart cdp2.iosproc
```

Related Commands

Command	Description
process start	Initiates (spawns) a foreground or background POSIX process.

process start

To initiate (spawn) a foreground or background POSIX process when a Cisco IOS Software Modularity image is running, use the **process start** command in privileged EXEC mode.

process start *path/process-name* [*argument-1...argument-n*] [**&**]

Syntax Description

path	Path to the process.
/process-name	Process name. The slash mark is required.
argument-1...argument-n	(Optional) One or more command-line arguments that are passed to the initiating process.
&	(Optional) Starts the process in the background.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   The **process start** command is used to control POSIX processes and processes that are registered with sysmgr by using .startup and .init files. To terminate a POSIX process that is running in the foreground, use the Ctrl-C (^C) keyboard sequence.

Output for processes that are running in the foreground is directed to the tty (including Telnet) that initiates the command. Output for processes that are running in the background is directed only to the console.

Examples   The following example initiates a POSIX process to run in the background:

```
Router# process start disk0:/sbin/process1 &
```

Related Commands

Command	Description
process stop	Terminates a process when running a Cisco IOS Software Modularity image without restarting the process.

process stop

To terminate a process without restarting the process when a Cisco IOS Software Modularity image is running, use the **process stop** command in privileged EXEC mode.

process stop *process-name* [**:***instance-id*]

Syntax Description

process-name	Process name.
	Note Only processes that are not controlled by the System Manager can be stopped.
:*instance-id*	(Optional) Process number. The first process is numbered 1, and this is the default if no number is specified. The colon is required.

Command Default After a process is terminated, the process is restarted.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines Use the **process stop** command to shut down (terminate) the specified process and any simultaneously executing copies. The process is not restarted, even if it had a respawn option specified.

Note System-manager-controlled processes (for example, cdp2.iosproc) cannot be stopped.

Examples The following example shuts down all instances of the POSIX process named process1:

```
Router# process stop process1
```

Related Commands

Command	Description
process start	Initiates (spawns) a foreground or background POSIX process.

service checkpoint-config

To enable implicit configuration checkpointing when a Cisco IOS Software Modularity image is running, use the **service checkpoint-config** command in global configuration mode. To return to the default setting, use the **no** form of this command.

service checkpoint-config

no service checkpoint-config

Syntax Description   This command has no arguments or keywords.

Command Default   Implicit configuration checkpointing is disabled.

Command Modes   Global configuration (config)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.
12.2(33)SXH	Automatic configuration checkpointing is disabled by default.

Usage Guidelines   Implicit configuration checkpointing means that configuration checkpointing occurs for all processes. A Software Modularity process can be restarted under an error condition or after upgrading. When the process is restarted and operational, the state of the process returns to the state the process was in prior to the restart. The software checkpoints the configuration information and when the process restarts, the configuration information is read from the checkpoint.

Configuration checkpoint information is implicitly generated as follows:

- Each time you exit from global configuration mode.
- Each time you enter the **write memory**, **copy running-config**, or **show running-config** command.
- When the action generated by the **write checkpoint** command has completed. The **write checkpoint** command is visible only after you enter the **no service checkpoint-config** command.

If you have a large configuration file, the default configuration checkpoint process may take some time to complete and prevent you from entering other CLI commands to save or display the configuration.

In Cisco IOS Release 12.2(18)SXF4, the checkpoint process is enabled by default. To disable the checkpoint process, enter the **no** form of the **service checkpoint-config** command. When you are ready to run the configuration checkpoint process, use the **write checkpoint** command to run the configuration checkpoint process.

In Cisco IOS Release 12.2(33)SXH, the default setting was changed to **no service checkpoint-config**, which means the checkpoint process is diabled by default. To enable the checkpoint process in this release, use the **service checkpoint-config** command.

Examples

In the following example for Cisco IOS Release 12.2(18)SXF4, the **no** form of the **service checkpoint-config** command is entered to disable the configuration checkpoint process, configuration commands are entered, and after exiting from the configuration mode the **write checkpoint** command is entered to run the configuration checkpoint process.

```
configure terminal
 no service checkpoint-config
!
! Configuration commands are entered.
 end

write checkpoint
```

Related Commands

Command	Description
write checkpoint	Runs the configuration checkpoint process when a Cisco IOS Software Modularity image is running.

show buffers

To display statistics for the buffer pools on the network server when Cisco IOS or Cisco IOS Software Modularity images are running, use the **show buffers** command in user EXEC or privileged EXEC mode.

> **show buffers** [{**address** *hex-address* | **failures** | **pool** *pool-name* | **processes** | {**all** | **assigned** [*process-id*] | **free** | **old** | **input-interface** *interface-type interface-number*} [**pool** *pool-name*]} [**dump** | **header** | **packet**]]

Syntax Description

address	(Optional) Displays buffers at a specified address.
hex-address	(Optional) Address in hexadecimal notation.
failures	(Optional) Displays buffer allocation failures.
pool	(Optional) Displays buffers in a specified buffer pool.
pool-name	(Optional) Name of buffer pool.
processes	(Optional) For Cisco IOS Software Modularity images only. Displays buffers connected to Packet Manager.
all	(Optional) Displays all buffers.
assigned	(Optional) Displays the buffers in use.
process-id	(Optional) For Cisco IOS Software Modularity images only. POSIX process identifier.
free	(Optional) Displays the buffers available for use.
old	(Optional) Displays buffers older than one minute.
input-interface	(Optional) Displays interface pool information. If an interface type is specified and this interface has its own buffer pool, information for that pool is displayed.
interface-type	(Optional) Interface type.
interface-number	(Optional) Interface number.
dump	(Optional) Displays the buffer header and all data.
header	(Optional) Displays the buffer header only.
packet	(Optional) Displays the buffer header and packet data.

Command Default   If no options are specified, all buffer pool information is displayed.

Command Modes   User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
10.0	This command was introduced.
12.3	The option to filter display output based on specific buffer pools was expanded.
12.2(18)SXF4	Two additional fields were added to the output to support Cisco IOS Software Modularity.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

Example output varies between Cisco IOS software images and Cisco IOS Software Modularity software images. To view the appropriate output, choose one of the following sections:

- Cisco IOS Software
- Cisco IOS Software Modularity

Cisco IOS Software

The following is sample output from the **show buffers** command with no arguments, showing all buffer pool information:

```
Router# show buffers

Buffer elements:
     398 in free list (500 max allowed)
     1266 hits, 0 misses, 0 created

Public buffer pools:
Small buffers, 104 bytes (total 50, permanent 50):
     50 in free list (20 min, 150 max allowed)
     551 hits, 0 misses, 0 trims, 0 created
Middle buffers, 600 bytes (total 25, permanent 25):
     25 in free list (10 min, 150 max allowed)
     39 hits, 0 misses, 0 trims, 0 created
Big buffers, 1524 bytes (total 50, permanent 50):
     49 in free list (5 min, 150 max allowed)
     27 hits, 0 misses, 0 trims, 0 created
VeryBig buffers, 4520 bytes (total 10, permanent 10):
     10 in free list (0 min, 100 max allowed)
     0 hits, 0 misses, 0 trims, 0 created
Large buffers, 5024 bytes (total 0, permanent 0):
     0 in free list (0 min, 10 max allowed)
     0 hits, 0 misses, 0 trims, 0 created
Huge buffers, 18024 bytes (total 0, permanent 0):
     0 in free list (0 min, 4 max allowed)
     0 hits, 0 misses, 0 trims, 0 created

Interface buffer pools:
Ethernet0 buffers, 1524 bytes (total 64, permanent 64):
     16 in free list (0 min, 64 max allowed)
     48 hits, 0 fallbacks
     16 max cache size, 16 in cache
Ethernet1 buffers, 1524 bytes (total 64, permanent 64):
     16 in free list (0 min, 64 max allowed)
     48 hits, 0 fallbacks
     16 max cache size, 16 in cache
Serial0 buffers, 1524 bytes (total 64, permanent 64):
     16 in free list (0 min, 64 max allowed)
     48 hits, 0 fallbacks
     16 max cache size, 16 in cache
```

```
Serial1 buffers, 1524 bytes (total 64, permanent 64):
    16 in free list (0 min, 64 max allowed)
    48 hits, 0 fallbacks
    16 max cache size, 16 in cache
TokenRing0 buffers, 4516 bytes (total 48, permanent 48):
    0 in free list (0 min, 48 max allowed)
    48 hits, 0 fallbacks
    16 max cache size, 16 in cache
TokenRing1 buffers, 4516 bytes (total 32, permanent 32):
    32 in free list (0 min, 48 max allowed)
    16 hits, 0 fallbacks
    0 failures (0 no memory)
```

The following is sample output from the **show buffers** command with no arguments, showing onlybuffer pool information for Huge buffers. This output shows a highest total of five Huge buffers created five days and 18 hours before the command was issued.

```
Router# show buffers

Huge buffers, 18024 bytes (total 5, permanent 0, peak 5 @ 5d18h):
    4 in free list (3 min, 104 max allowed)
    0 hits, 1 misses, 101 trims, 106 created
    0 failures (0 no memory)
```

The following is sample output from the **show buffers** command with no arguments, showing only buffer pool information for Huge buffers. This output shows a highest total of 184 Huge buffers created one hour, one minute, and 15 seconds before the command was issued.

```
Router# show buffers

Huge buffers, 65280 bytes (total 4, permanent 2, peak 184 @ 01:01:15):
    4 in free list (0 min, 4 max allowed)
    32521 hits, 143636 misses, 14668 trims, 14670 created
    143554 failures (0 no memory)
```

The following is sample output from the **show buffers** command with an interface type and interface number:

```
Router# show buffers Ethernet 0

Ethernet0 buffers, 1524 bytes (total 64, permanent 64):
    16 in free list (0 min, 64 max allowed)
    48 hits, 0 fallbacks
  16 max cache size, 16 in cache
```

Table 6 describes the significant fields shown in the display.

Table 6   show buffers (Cisco IOS Software) Field Descriptions

Field	Description
Buffer elements	Small structures used as placeholders for buffers in internal operating system queues. Used when a buffer may need to be on more than one queue.
free list	Total number of the currently unallocated buffer elements.
max allowed	Maximum number of buffers that are available for allocation.
hits	Count of successful attempts to allocate a buffer when needed.
misses	Count of buffer allocation attempts that resulted in growing the buffer pool to allocate a buffer.

Table 6   show buffers (Cisco IOS Software) Field Descriptions (continued)

Field	Description
created	Count of new buffers created to satisfy buffer allocation attempts when the available buffers in the pool have already been allocated.
Public Buffer Pools	
Small buffers	Buffers that are 104 bytes long.
Middle buffers	Buffers that are 600 bytes long.
Big buffers	Buffers that are 1524 bytes long.
VeryBig buffers	Buffers that are 4520 bytes long.
Large buffers	Buffers that are 5024 bytes long.
Huge buffers	Buffers that are 18,024 bytes long.
total	Total number of this type of buffer.
permanent	Number of these buffers that are permanent.
peak	Maximum number of buffers created (highest total) and the time when that peak occurred. Formats include weeks, days, hours, minutes, and seconds. Not all systems report a peak value, which means this field may not display in output.
free list	Number of available or unallocated buffers in that pool.
min	Minimum number of free or unallocated buffers in the buffer pool.
max allowed	Maximum number of free or unallocated buffers in the buffer pool.
hits	Count of successful attempts to allocate a buffer when needed.
misses	Count of buffer allocation attempts that resulted in growing the buffer pool in order to allocate a buffer.
trims	Count of buffers released to the system because they were not being used. This field is displayed only for dynamic buffer pools, not interface buffer pools, which are static.
created	Count of new buffers created in response to misses. This field is displayed only for dynamic buffer pools, not interface buffer pools, which are static.
Interface Buffer Pools	
total	Total number of this type of buffer.
permanent	Number of these buffers that are permanent.
free list	Number of available or unallocated buffers in that pool.
min	Minimum number of free or unallocated buffers in the buffer pool.
max allowed	Maximum number of free or unallocated buffers in the buffer pool.
hits	Count of successful attempts to allocate a buffer when needed.
fallbacks	Count of buffer allocation attempts that resulted in falling back to the public buffer pool that is the smallest pool at least as big as the interface buffer pool.

Table 6   show buffers (Cisco IOS Software) Field Descriptions (continued)

Field	Description
max cache size	Maximum number of buffers from the pool of that interface that can be in the buffer pool cache of that interface. Each interface buffer pool has its own cache. These are not additional to the permanent buffers; they come from the buffer pools of the interface. Some interfaces place all of their buffers from the interface pool into the cache. In this case, it is normal for the *free list* to display 0.
failures	Total number of times a buffer creation failed. The failure may have occurred because of a number of different reasons, such as low processor memory, low IOMEM, or no buffers in the pool when called from interrupt context.
no memory	Number of times there has been low memory during buffer creation. Low or no memory during buffer creation may not necessarily mean that buffer creation failed; memory can be obtained from an alternate resource such as a fallback pool.

Cisco IOS Software Modularity

The following is sample output from the **show buffers** command using a Cisco IOS Modularity image from Cisco IOS Release 12.2(18)SXF4 and later releases. Two new output fields were introduced—Public buffer heads and Temporary buffer heads—and are shown within comments in the following sample output.

```
Router# show buffers

Buffer elements:
    500 in free list (500 max allowed)
    106586 hits, 0 misses, 0 created

Public buffer pools:
Small buffers, 104 bytes (total 50, permanent 50, peak 54 @ 1d13h):
    49 in free list (20 min, 150 max allowed)
    54486 hits, 0 misses, 4 trims, 4 created
    0 failures (0 no memory)
Middle buffers, 600 bytes (total 25, permanent 25, peak 27 @ 1d13h):
    25 in free list (10 min, 150 max allowed)
    20 hits, 0 misses, 2 trims, 2 created
    0 failures (0 no memory)
Big buffers, 1536 bytes (total 50, permanent 50):
    50 in free list (40 min, 150 max allowed)
    6 hits, 0 misses, 0 trims, 0 created
    0 failures (0 no memory)
VeryBig buffers, 4520 bytes (total 10, permanent 10):
    10 in free list (0 min, 100 max allowed)
    0 hits, 0 misses, 0 trims, 0 created
    0 failures (0 no memory)
Large buffers, 5024 bytes (total 0, permanent 0):
    0 in free list (0 min, 10 max allowed)
    0 hits, 0 misses, 0 trims, 0 created
    0 failures (0 no memory)
Huge buffers, 18024 bytes (total 1, permanent 0, peak 1 @ 1d13h):
    0 in free list (0 min, 4 max allowed)
    1 hits, 0 misses, 0 trims, 0 created
    0 failures (0 no memory)

! Start of Cisco IOS Software Modularity fields
Public buffer headers:
Header buffers, 880 bytes (total 1000, peak 142 @ 1d13h):
```

```
    864 in permanent free list
    142 hits, 0 misses

Temporary buffer headers:
Header buffers, 896 bytes (total 0):
    0 in free list
    0 hits, 0 misses, 0 trims, 0 created
    0 failures
! End of Cisco IOS Software Modularity fields

Interface buffer pools:
Logger Pool buffers, 600 bytes (total 150, permanent 150):
    150 in free list (150 min, 150 max allowed)
    22 hits, 0 misses
```

Table 7 describes the significant fields shown in the display that are different from the fields in Table 6.

Table 7   show buffers (Cisco IOS Software Modularity) Field Descriptions

Field	Description
Public Buffer Headers	
Header buffers	Buffers that are 880 bytes long.
total	Total number of this type of buffer.
permanent free list	Number of available or unallocated permanent header buffers.
hits	Count of successful attempts to allocate a header buffer when needed.
misses	Count of buffer allocation attempts that resulted in growing the buffer pool in order to allocate a buffer.
Temporary Buffer Headers	
Header buffers	Buffers that are 896 bytes long.
total	Total number of this type of buffer.
free list	Number of available or unallocated header buffers in that pool.
hits	Count of successful attempts to allocate a buffer when needed.
misses	Count of buffer allocation attempts that resulted in growing the buffer pool in order to allocate a buffer.
trims	Count of buffers released to the system because they were not being used. This field is displayed only for dynamic buffer pools, not interface buffer pools, which are static.
created	Count of new buffers created in response to misses. This field is displayed only for dynamic buffer pools, not interface buffer pools, which are static.
failures	Total number of allocation requests that have failed because no buffer was available for allocation; the datagram was lost. Such failures normally occur at interrupt level.

show exception

To display the current exception configuration when a Cisco IOS Software Modularity image is running, use the **show exception** command in user EXEC or privileged EXEC mode.

show exception

Syntax Description   This command has no arguments or keywords.

Command Modes   User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   Use the **show exception** command to display the current process and kernel dumper configuration as configured by the various **exception** commands used in Software Modularity images.

Examples   The following is sample output from the **show exception** command:

```
Router# show exception

Core Dump Configurations:

Choice 1
========
 Filepath               : disk0:
 Filename               : test1
 Lower Filename Suffix   : 0
 Upper Filename Suffix   : 4
 Current Filename Suffix : 0
 Compression            : on

Choice 2
========
 Filepath               : disk1:
 Filename               : test1
 Lower Filename Suffix   : 0
 Upper Filename Suffix   : 4
 Current Filename Suffix : 0
 Compression            : on

Choice 3
========
 Filepath               : slot0
 Filename               : test1
 Lower Filename Suffix   : 0
 Upper Filename Suffix   : 4
 Current Filename Suffix : 0
 Compression            : on
```

Table 8 describes the significant fields shown in the display.

Table 8 *show exception Field Descriptions*

Field	Description
Choice	Indicates the order of local dump locations.
Filepath	Indicates the path of the core dump file.
Filename	Name of the core dump file.
Compression	Indicates whether the file is to be written as a compressed file.

Related Commands

Command	Description
exception core	Sets or changes the core dump options for a Cisco IOS Software Modularity process.
exception core-file	Specifies the name of the core dump file.
exception flash	Configures the dump location for core files when a process reloads.
exception kernel	Configures a networking device on which a Cisco IOS Software Modularity image is running, to dump the kernel memory.

show install

To display information about the installed Cisco IOS Software Modularity software, including patch files, use the **show install** command in user EXEC or privileged EXEC mode.

> **show install** [**tags**] {**running** | *search-root-directory*} [**tagname** *tag-name*] [**detailed** | **pending**]

Syntax Description		
tags	(Optional) Displays the tag information that is defined for the installer software.	
running	Displays information about the software that is currently running on each location in the system.	
search-root-directory	A local directory specified as the destination directory in a previously executed **install file** command.	
tagname *tag-name*	(Optional) Displays the information for a particular tag. The **tagname** *tag-name* keyword/argument pair can be defined only if the optional **tags** keyword is used.	
detailed	(Optional) Displays more detailed information.	
pending	(Optional) Displays patch upgrade summary information.	

Command Modes   User EXEC (>)
Privileged EXEC (#)

Command History	Release	Modification
	12.2(18)SXF4	This command was introduced to support Software Modularity images.
	12.2(18)SXF5	The **tagname** *tag-name* keyword/argument pair was added.
	12.2(33)SXI1	The **pending** keyword was added.

Examples   The following is sample output from the **show install running** command:

```
Router# show install running

Software running on card installed at location s72033 - Slot 5 :

B/P C State     Filename
--- - -------- --------
 B    Active   disk0:/sys/s72033/base/s72033-adventerprisek9_wan_dbg-vm(12.2(99)SX1010)

Software running on card installed at location s72033_rp - Slot 5 :

B/P C State     Filename
--- - -------- --------
 P    Active   disk0:/sys/s72033_rp/patch/patch-AAA1258-patch-0-n.so

Software running on card installed at location s72033 - Slot 6 :
B/P C State     Filename
--- - -------- --------
 B    Active
slavedisk0:/sys/s72033/base/s72033-adventerprisek9_wan_dbg-vm(12.2(99)SX1010)
```

```
LEGEND:
-------:
B/P/MP - (B)ase image, (P)atch, or (M)aintenance (P)ack
'C' - (C)ommitted
Pruned - This file has been pruned from the system
Active - This file is active in the system
PendInst - This file is set to be made available to run on the
   system after next activation.
PendRoll - This file is set to be rolled back after next activation.
InstPRel - This file will run on the system after next reload
RollPRel - This file will be removed from the system after next reload
RPRPndIn - This file is both rolled back pending a reload, and pending
   installation.  On reload, this file will not run and will move to
   PendInst state.  If 'install activate' is done before reload, pending
   removal and install cancel each other and file simply remains active
IPRPndRo - This file is both installed pending a reload, and pending rollback.
   If the card reloads, it will be active on the system pending a rollback
   If 'install activate' is done before a reload, the pending install and
   removal with cancel each other and the file will simply be removed
Occluded - This file has been occluded from the system,
   a newer version of itself has superceded it.
Ignored - This file is ignored, is not consumed by target.
```

Table 9 describes the significant fields shown in the display.

Table 9   show install running Field Descriptions

Field	Description
B/P/MP	Indicates whether the file is a base image file (B), a patch file (P), or a maintenance pack (MP) file.
C	An asterisk under this column indicates that this file has been committed under a user-defined tag.
State	Current state of the software file. For a list of states, see the description under the LEGEND section of the output.
Filename	Name and path of an installed file on the system. If the filename ends with some text in parenthesis, the text represents the Cisco IOS version number of the image file.

The following is sample output from the **show install running** command with the **detailed** keyword:

```
Router# show install running detailed

Software running on card installed at location s72033 - Slot 5 :

Base image : disk0:/sys/s72033/base/s72033-adventerprisek9_wan_dbg-vm
Version : 12.2(99)SX1010
File state: Active    File Checksum : 8BB2F966EA945E8E25010A1BAC7205C3DFBCA197
Date Installed : 19:51:22 UTC Sep 8 2005 Commit Tags :  base

Software running on card installed at location s72033_rp - Slot 5 :

Base image : disk0:/sys/s72033_rp/base/DRACO2_MP
File state: Active    File Checksum : 48849DBB2E47A8C55AC68CF3F6EE747B054CD392
Date Installed : 19:49:06 UTC Sep 8 2005 Commit Tags :  base
Software running on card installed at location s72033 - Slot 6 :

Base image : slavedisk0:/sys/s72033/base/s72033-adventerprisek9_wan_dbg-vm
Version : 12.2(99)SX1010
```

```
File state: Active    File Checksum : 8BB2F966EA945E8E25010A1BAC7205C3DFBCA197
Date Installed : 19:32:21 UTC Sep 8 2005 Commit Tags :  base Patch :
slavedisk0:/sys/s72033/patch/patch-AAA1258-patch-0-n.so
File state: PendInst  File Checksum : A129339A6A3ED1F8B92D6992AD1BE67C716E4430
Date Installed : 20:31:01 UTC Sep 9 2005 Commit Tags : NONE  Maintenance Pack : MA0005
```

Table 10 describes the significant fields shown in the display.

Table 10   show install running detailed Field Descriptions

Field	Description
Base image	Name of the base image for this node.
Version	Cisco IOS version number associated with this file.
File state	Current state of the file.
File Checksum	The Secure Hash Algorithm (SHA) checksum used to validate this file.
Date Installed	The date and time that this file was installed.
Commit Tags	Names of all the committed tags that include this file.

In the following example, the **show install** privileged EXEC command is used to display information about the tags that are defined for this system:

```
Router# show install tags running

Tags defined over software running on location s72033 - Slot 5 :

Tagname            # of Files        Date Committed
-------------------- ----------    -----------------------
base                   1           20:08:51 UTC Sep 9 2005
MA0005                 1           20:34:16 UTC Sep 9 2005


Tags defined over software running on location s72033_rp - Slot 5 :

Tagname            # of Files        Date Committed
-------------------- ----------    -----------------------
base                   1           20:08:51 UTC Sep 9 2005
MA0005                 1           20:34:16 UTC Sep 9 2005


Tags defined over software running on location s72033 - Slot 6 :

Tagname            # of Files        Date Committed
-------------------- ----------    -----------------------
base                   1           20:28:54 UTC Sep 9 2005
```

Table 11 describes the significant fields shown in the display.

Table 11   show install tags running Field Descriptions

Field	Description
Tagname	Name of the tag being described.
# of Files	Number of installed files committed under this tag.
Date Committed	The date and time that this tag was created.

In the following example, the **show install** privileged EXEC command is used to display detailed information about the tags that are defined for this system:

```
Router# show install tags running detailed

Tags defined over software running on location s72033 :

Tag Name :base
Date Committed :Fri Sep 9 17:54:37 2005
Files under this tag:
disk0:/sys/s72033/base/s72033-adventerprisek9_wan_dbg-vm
```

In the following example, the **show install** privileged EXEC command is used to display detailed information about the tag named tag1:

```
Router# show install tags running tagname tag1 detailed

Tags defined over software running on location c7200:
Tag Name : tag1
Date Committed : 01:49:23 UTC Mar 8 2006
Files under this tag: disk0:/sys/c7200/base/c7200-p-vm
```

Table 12 describes the significant fields shown in the display.

Table 12   *show install tags running detailed Field Descriptions*

Field	Description
Tag Name	Name of the tag being described.
Date Committed	The date and time that this tag was created.
Files under this tag	List of all files committed under this tag.

Related Commands	**Command**	**Description**
	install bind	Binds Cisco IOS Software Modularity images.
	install commit	Defines a tag for a set of software installed by the **install file** command.
	install file	Installs base system files and patches.

show memory

To display statistics about memory when Cisco IOS or Cisco IOS software Modularity images are running, use the **show memory** command in user EXEC or privileged EXEC mode.

Cisco IOS Software

    show memory [*memory-type*] [**free**] [**overflow**] [**summary**]

Cisco IOS Software Modularity

    show memory

Syntax Description		
memory-type	(Optional) Memory type to display (**processor**, **multibus**, **io**, or **sram**). If *memory-type* is not specified, statistics for all memory types present are displayed.
free	(Optional) Displays free memory statistics.
overflow	(Optional) Displays details about memory block header corruption corrections when the **exception memory ignore overflow** global configuration command is configured.
summary	(Optional) Displays a summary of memory usage including the size and number of blocks allocated for each address of the system call that allocated the block.

Command Modes   User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
10.0	This command was introduced.
12.3(7)T	This command was enhanced with the **overflow** keyword to display details about memory block header corruption corrections.
12.2(25)S	The command output was updated to display information about transient memory pools.
12.3(14)T	The command output was updated to display information about transient memory pools.
12.2(27)SBC	This command was integrated into Cisco IOS Release 12.2(27)SBC.
12.2(18)SXF4	This command was implemented in Cisco IOS Software Modularity images.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines   **Cisco IOS Software**

The **show memory** command displays information about memory available after the system image decompresses and loads.

Cisco IOS Software Modularity

No optional keywords or arguments are supported for the **show memory** command when a Software Modularity image is running. To display details about PSOIX and Cisco IOS style system memory information when Software Modularity images are running, use the **show memory detailed** command.

Examples

Example output varies between Cisco IOS software images and Cisco IOS Software Modularity software images. To view the appropriate output, choose one of the following sections:

- Cisco IOS Software
- Cisco IOS Software Modularity

Cisco IOS Software

The following is sample output from the **show memory** command:

```
Router# show memory

            Head    Total(b)   Used(b)    Free(b)  Lowest(b) Largest(b)
Processor   B0EE38   5181896    2210036    2971860   2692456   2845368


            Processor memory
Address   Bytes Prev.     Next     Ref  PrevF    NextF    Alloc PC   What
B0EE38    1056  0         B0F280    1                     18F132     List Elements
B0F280    2656  B0EE38    B0FD08    1                     18F132     List Headers
B0FD08    2520  B0F280    B10708    1                     141384     TTY data
B10708    2000  B0FD08    B10F00    1                     14353C     TTY Input Buf
B10F00     512  B10708    B11128    1                     14356C     TTY Output Buf
B11128    2000  B10F00    B11920    1                     1A110E     Interrupt Stack
B11920      44  B11128    B11974    1                     970DE8     *Init*
B11974    1056  B11920    B11DBC    1                     18F132     messages
B11DBC      84  B11974    B11E38    1                     19ABCE     Watched Boolean
B11E38      84  B11DBC    B11EB4    1                     19ABCE     Watched Boolean
B11EB4      84  B11E38    B11F30    1                     19ABCE     Watched Boolean
B11F30      84  B11EB4    B11FAC    1                     19ABCE     Watched Boolean
```

The following is sample output from the **show memory free** command:

```
Router# show memory free

            Head    Total(b)   Used(b)    Free(b)  Lowest(b) Largest(b)
Processor   B0EE38   5181896    2210076    2971820   2692456   2845368


            Processor memory
Address   Bytes Prev.     Next     Ref  PrevF    NextF    Alloc PC   What
           24      Free list 1
CEB844     32   CEB7A4 CEB88C     0    0        0        96B894     SSE Manager
           52      Free list 2
           72      Free list 3
           76      Free list 4
           80      Free list 5
D35ED4     80   D35E30   D35F4C   0    0        D27AE8   96B894     SSE Manager
D27AE8     80   D27A48   D27B60   0    D35ED4   0        22585E     SSE Manager
           88      Free list 6
          100      Free list 7
D0A8F4    100   D0A8B0   D0A980   0    0        0        2258DA     SSE Manager
          104      Free list 8
B59EF0    108   B59E8C   B59F84   0    0        0        2258DA     (fragment)
```

The output of the **show memory free** command contains the same types of information as the **show memory** output, except that only free memory is displayed, and the information is ordered by free list.

The first section of the display includes summary statistics about the activities of the system memory allocator. Table 13 describes the significant fields shown in the first section of the display.

Table 13   show memory Field Descriptions—First Section

Field	Description
Head	Hexadecimal address of the head of the memory allocation chain.
Total(b)	Sum of used bytes plus free bytes.
Used(b)	Amount of memory in use.
Free(b)	Amount of memory not in use.
Lowest(b)	Smallest amount of free memory since last boot.
Largest(b)	Size of largest available free block.

The second section of the display is a block-by-block listing of memory use. Table 14 describes the significant fields shown in the second section of the display.

Table 14   Characteristics of Each Block of Memory—Second Section

Field	Description
Address	Hexadecimal address of block.
Bytes	Size of block (in bytes).
Prev.	Address of previous block (should match the address on previous line).
Next	Address of next block (should match the address on next line).
Ref	Reference count for that memory block, indicating how many different processes are using that block of memory.
PrevF	Address of previous free block (if free).
NextF	Address of next free block (if free).
Alloc PC	Address of the system call that allocated the block.
What	Name of process that owns the block, or "(fragment)" if the block is a fragment, or "(coalesced)" if the block was coalesced from adjacent free blocks.

The **show memory io** command displays the free I/O memory blocks. On the Cisco 4000 router, this command quickly shows how much unused I/O memory is available.

The following is sample output from the **show memory io** command:

```
Router# show memory io

Address   Bytes Prev.    Next     Ref  PrevF    NextF    Alloc PC  What
6132DA0   59264 6132664  6141520  0    0        600DDEC  3FCF0     *Packet Buffer*
600DDEC     500 600DA4C  600DFE0  0    6132DA0  600FE68  0
600FE68     376 600FAC8  600FFE0  0    600DDEC  6011D54  0
6011D54     652 60119B4  6011FE0  0    600FE68  6013D54  0
614FCA0     832 614F564  614FFE0  0    601FD54  6177640  0
6177640 2657056 6172E90  0        0    614FCA0  0        0
Total: 2723244
```

The following example displays details of a memory block overflow correction when the **exception memory ignore overflow** global configuration command is configured:

```
Router# show memory overflow

Count    Buffer Count    Last corrected       Crashinfo files

1        1               00:11:17             slot0:crashinfo_20030620-075755
Traceback    607D526C 608731A0 607172F8 607288E0 607A5688 607A566C
```

The report includes the amount of time since the last correction was made and the name of the file that logged the memory block overflow details.

The **show memory sram** command displays the free SRAM memory blocks. For the Cisco 4000 router, this command supports the high-speed static RAM memory pool to make it easier for you to debug or diagnose problems with allocation or freeing of such memory.

The following is sample output from the **show memory sram** command:

```
Router# show memory sram

Address   Bytes Prev.   Next     Ref  PrevF  NextF   Alloc PC  What
7AE0      38178 72F0    0        0    0      0       0
Total     38178
```

The following example of the **show memory** command used on the Cisco 4000 router includes information about SRAM memory and I/O memory:

```
Router# show memory

              Head    Total(b)   Used(b)    Free(b)   Lowest(b) Largest(b)
Processor   49C724    28719324   1510864    27208460  26511644  15513908
     I/O    6000000   4194304    1297088    2897216   2869248   2896812
    SRAM    1000      65536      63400      2136      2136      2136

Address   Bytes Prev.   Next     Ref  PrevF   NextF    Alloc PC  What
1000      2032  0       17F0     1                     3E73E     *Init*
17F0      2032  1000    1FE0     1                     3E73E     *Init*
1FE0      544   17F0    2200     1                     3276A     *Init*
2200      52    1FE0    2234     1                     31D68     *Init*
2234      52    2200    2268     1                     31DAA     *Init*
2268      52    2234    229C     1                     31DF2     *Init*
72F0      2032  6E5C    7AE0     1                     3E73E     Init
7AE0      38178 72F0    0        0    0       0        0
```

The **show memory summary** command displays a summary of all memory pools and memory usage per Alloc PC (address of the system call that allocated the block).

The following is a partial sample output from the **show memory summary** command. This output shows the size, blocks, and bytes allocated. Bytes equal the size multiplied by the blocks. For a description of the other fields, see Table 13 and Table 14.

```
Router# show memory summary

Head    Total(b)    Used(b)    Free(b)   Lowest(b) Largest(b)
Processor    B0EE38    5181896    2210216    2971680   2692456    2845368

          Processor memory
Alloc PC        Size      Blocks      Bytes     What
0x2AB2          192       1           192       IDB: Serial Info
0x70EC          92        2           184       Init
0xC916          128       50          6400      RIF Cache
0x76ADE         4500      1           4500      XDI data
0x76E84         4464      1           4464      XDI data
```

```
0x76EAC          692        1        692    XDI data
0x77764          408        1        408    Init
0x77776          116        1        116    Init
0x777A2          408        1        408    Init
0x777B2          116        1        116    Init
0xA4600           24        3         72    List
0xD9B5C           52        1         52    SSE Manager
.
.
.
0x0                0     3413    2072576    Pool Summary
0x0                0       28    2971680    Pool Summary (Free Blocks)
0x0               40     3441     137640    Pool Summary (All Block Headers)
0x0                0     3413    2072576    Memory Summary
0x0                0       28    2971680    Memory Summary (Free Blocks)
```

Cisco IOS Software Modularity

The following is sample output from the **show memory** command when a Cisco IOS Software Modularity image is running.

```
Router# show memory

System Memory: 262144K total, 116148K used, 145996K free 4000K kernel reserved
```

Table 15 describes the significant fields shown in the display.

Table 15   ***show memory (Software Modularity Image) Field Descriptions***

Field	Description
total	Total amount of memory on the device, in kilobytes.
used	Amount of memory in use, in kilobytes.
free	Amount of memory not in use, in kilobytes.
kernel reserved	Amount of memory reserved by the kernel, in kilobytes.

Related Commands	Command	Description
	exception memory ignore overflow	Configures the Cisco IOS software to correct corruptions in memory block headers and allow a router to continue its normal operation.
	show memory detailed	Displays POSIX and Cisco IOS style system memory information.
	show processes memory	Displays memory used per process.

show memory detailed

To display detailed memory information about POSIX and Cisco IOS processes when Cisco IOS Software Modularity images are running, use the **show memory detailed** command in privileged EXEC mode.

> **show memory detailed** [*process-id* | *process-name*] [*start-address* [*end-address*] | **bigger** | **free** | **physical** | **shared** | **statistics** | **summary**]

Syntax Description		
	process-id	(Optional) POSIX process identifier.
	process-name	(Optional) POSIX process name.
	start-address	(Optional) Starting memory address.
	end-address	(Optional) Ending memory address.
	bigger	(Optional) Displays information about bigger free blocks in the process.
	free	(Optional) Displays free memory information.
	physical	(Optional) Displays physical memory information.
	shared	(Optional) Displays shared memory information.
	statistics	(Optional) Displays detailed memory usage by address of the system call that allocated the block.
	summary	(Optional) Displays summary information about memory usage per system call that allocated the block.

Command Default

No detailed memory information about POSIX and Cisco IOS processes is displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines

Detailed output of the process memory on the device is displayed with this command. The process memory summary is displayed first, followed by POSIX and Cisco IOS memory information. The POSIX memory information includes the address, the size in bytes, and the type of memory used by various segments such as program text, data, stack, shared memory, device memory, and heap. Cisco IOS memory information includes the native Cisco IOS display of memory blocks maintained by the Cisco IOS memory management library.

Examples   The following is partial sample output from the **show memory detailed** command for a Cisco IOS process:

```
Router# show memory detailed cdp2.iosproc

System Memory: 131072K total, 115836K used, 15236K free 4000K kernel reserved

Process sbin/cdp2.iosproc, type IOS, PID = 12329
    636K total, 4K text, 4K data, 28K stack, 600K dynamic
    16384 heapsize, 3972 allocated, 10848 free

Address       Bytes What
0x3B42000   4194304 Shared Memory
0x7FBB000      8192 Program Stack
0x8020000     49152 Program Text
0x802C000      4096 Program Data
0x802D000      8192 Allocated memory
0x60000000     4096 Shared Memory "SHM_IDB"
0x60001000    32768 Shared Memory

             Head       Total(b)  Used(b)    Free(b)    Lowest(b)  Largest(b)
Processor    8034058    508152    480420     27732      17368      18716

        Processor memory

 Address      Bytes       Prev      Next Ref     PrevF     NextF Alloc PC  what
08034058 0000020008 00000000 08038EB8 001  -------- -------- 727FB668 Managed Chunk Queue
Elements
08038EB8 0000002568 08034058 080398F8 001  -------- -------- 72871A44 *Init*
080398F8 0000001512 08038EB8 08039F18 001  -------- -------- 728819D4 List Elements
.
.
.
```

The first section of the display shows system summary information. Table 16 describes the significant fields shown in the first section of the display.

Table 16   show memory detailed Field Descriptions—First Section

Field	Description
total	Total amount of memory on the device, in kilobytes.
used	Amount of memory in use, in kilobytes.
free	Amount of memory not in use, in kilobytes.
kernel reserved	Amount of memory reserved by the kernel, in kilobytes.

The second section of the display includes process summary statistics about the activities of the system memory allocator. Table 17 describes the significant fields shown in the second section of the display.

Table 17   show memory detailed Field Descriptions—Second Section

Field	Description
Process	Process name and path.
type	Type of process: POSIX or IOS.
PID	Process ID.

Table 17   show memory detailed Field Descriptions—Second Section (continued)

Field	Description
total	Total amount of memory used by the specified process, in kilobytes.
text	Amount of memory, in kilobytes, used by the text segment of the specified process.
data	Amount of memory, in kilobytes, used by the data segment of the specified process.
stack	Amount of memory, in kilobytes, used by the stack segment of the specified process.
dynamic	Amount of memory, in kilobytes, used by the dynamic segment of the specified process.
heapsize	Size of the process heap. Note that the Cisco IOS memory management library allocates heap dynamically. This is shown in the Cisco IOS memory details that follow the POSIX memory display.
allocated	Amount of memory, in kilobytes, allocated from the heap.
free	Amount of free memory, in kilobytes, in the heap for the specified process.

The third section of the display shows POSIX process perspective memory information. Table 18 describes the significant fields shown in the third section of the display.

Table 18   show memory detailed Field Descriptions—Third Section

Field	Description
Address	Hexadecimal address of block.
Bytes	Size of block (in bytes).
What	Type of memory segment that owns the block, or "(fragment)" if the block is a fragment, or "(coalesced)" if the block was coalesced from adjacent free blocks.

The fourth section of the display shows Cisco IOS memory information as a block-by-block listing of memory use. Table 19 describes the significant fields shown in the fourth section of the display.

Table 19   show memory detailed Field Descriptions—Fourth Section

Field	Description
Head	Hexadecimal address of the head of the memory allocation chain.
Total(b)	Sum of used bytes plus free bytes.
Used(b)	Amount of memory in use.
Free(b)	Amount of memory not in use.
Lowest(b)	Smallest amount of free memory since last boot.
Largest(b)	Size of largest available free block.

Table 19   show memory detailed Field Descriptions—Fourth Section (continued)

Field	Description
Address	Hexadecimal address of block.
Bytes	Size of block (in bytes).
Prev	Address of previous block (should match address on previous line).
Next	Address of next block (should match address on next line).
PrevF	Address of previous free block (if free).
NextF	Address of next free block (if free).
Alloc PC	Address of the system call that allocated the block.
what	Type of memory segment that owns the block, or "(fragment)" if the block is a fragment, or "(coalesced)" if the block was coalesced from adjacent free blocks.

The following is sample output from the **show memory detailed** command for a POSIX process:

```
Router# show memory detailed 12290

System Memory: 131072K total, 115876K used, 15196K free 4000K kernel reserved

Process sbin/sysmgr.proc, type POSIX, PID = 12290
    400K total, 100K text, 144K data, 12K stack, 144K dynamic
    81920 heapsize, 68716 allocated, 8824 free

Address       Bytes What
0x7FDF000    126976 Program Stack (pages not allocated)
0x7FFE000      4096 Program Stack
0x8000000    122880 Program Stack (pages not allocated)
0x801E000      8192 Program Stack
0x8020000    102400 Program Text
0x8039000    147456 Program Data
0x805D000      8192 Heap Memory
0x8060000     16384 Heap Memory
0x8064000     16384 Heap Memory
0x8068000      8192 Heap Memory
0x806C000     16384 Heap Memory
0x8070000     16384 Heap Memory
0x8074000     16384 Heap Memory
0x8078000     16384 Heap Memory
0x807C000     16384 Heap Memory
0x8080000     16384 Heap Memory
```

The following partial sample output from the **show memory detailed** command with a process name and the **physical** keyword that displays the summary of physical memory used by the specified process along with the shared memory details:

```
Router# show memory detailed sysmgr.proc physical

Pid      Data   Stack Dynamic   Text  Shared  Maps  Process
20482    304K    16K    256K   3480K    468K   60  sysmgr.proc

Total Physical Memory used or mapped by sysmgr.proc
  Private memory used  (Data/Stack/Dynamic) :     576K
  Shared memory mapped (Text/Shared)        :    3948K
  Number of memory maps                     :      60
```

```
Dev   1:Text/Data 2:Mapped 3:Shared 4:DSO
Flags SHD:Shared PRV:Private FXD:Fixed ANN:Anon PHY:Phys
      LZY:Lazy ELF:Elf STK:Stack NOC:Nocache

Phy Addr        Size    Pid    Virt Addr   What    Dev   Prot   MapFlags
0x0            32768K  20482   0x70000000  Text     4   R-X   SHD FXD ELF
0x2000000      32768K  20482   0x72000000  Text     4   R-X   SHD FXD ELF
0x4000000      32768K  20482   0x74000000  Text     4   R-X   SHD FXD ELF
0x522B000         4K  20482   0x1020000   Text     4   R-X   SHD FXD ELF

Phy Addr        Size    Pid    Virt Addr   What    Dev   Prot   MapFlags
0x9EFD4000       32K  20482   0x105C000   Heap     2   RW-   PRV ANN
0x9EFF0000       32K  20482   0x1054000   Heap     2   RW-   PRV ANN
0x9EFF8000       32K  20482   0x1034000   Heap     2   RW-   PRV ANN
0x9F003000        4K  20482   0x7B43C000  Data     4   RW-   PRV FXD ANN ELF
.
.
.
```

Table 20 describes the significant fields shown in the display.

Table 20   show memory detailed Field Descriptions

Field	Description
Shared	Amount of memory shared by the specified process, in kilobytes.
Maps	Number of memory maps for the specified process.
Process	Name of the process.
Private memory used	Total amount of private memory used by the process.
Shared memory mapped	Total amount of shared memory used by the process.
Number of memory maps	Total number of maps for the process.

Table 20 show memory detailed Field Descriptions (continued)

Field	Description
Flags	Flags that specify information about handling of the mapped region. The available flags are as follows: • SHD:Shared—Specifies that memory is shared between different process. • PRV:Private—Specifies that memory is private to this process. • FXD:Fixed—Specifies that memory is mapped to a fixed virtual address in the process. • ANN:Anon—Specifies that physical memory was allocated by the kernel. • PHY:Phys—Specifies that the user specified the physical memory. • LZY:Lazy—Specifies that memory is lazy mapped; that is, physical memory is not allocated until the memory is either read or written to other memory. • ELF:Elf—Specifies that memory is an Executable and Linkable Format (ELF) object. • STK:Stack—Specifies that memory is used for stack. • NOC:Nocache—Specifies that memory is set up without any cache.
Phy Addr	Hexadecimal address of the physical memory block.
Size	Amount of physical memory mapped in the process of development.
Virt Addr	Virtual memory to which this memory is mapped.
Prot	Memory protection settings for the memory—read, write, and execute.
MapFlags	Represents special mapping properties used for the memory.

Related Commands

Command	Description
show memory	Displays system memory information.
show memory detailed all	Displays detailed memory information of all applicable processes.

show memory detailed all

To display detailed memory information of all applicable processes when Cisco IOS Software Modularity images are running, use the **show memory detailed all** command in privileged EXEC mode.

> **show memory detailed all** [[*start-address* [*end-address*] | **failures alloc** | **shared** | **statistics** [**history**] | **summary**] | [**fast** | **io** | **multibus** [**pci**] | **pci** | **processor**] [**allocating-process** [**totals**] | **dead** [**totals**] | **free** | **physical**]]

Syntax Description		
start-address	(Optional) Starting memory address.	
end-address	(Optional) Ending memory address.	
failures	(Optional) Displays memory failure details.	
alloc	(Optional) Displays memory allocation failure.	
shared	(Optional) Displays shared memory information.	
statistics	(Optional) Displays detailed memory usage by address of the system call that allocated the block.	
history	(Optional) Displays the memory pool history.	
summary	(Optional) Displays summary information about memory usage per system call that allocated the block.	
fast	(Optional) Displays fast memory statistics.	
io	(Optional) Displays input output memory statistics.	
multibus	(Optional) Displays multibus memory statistics.	
pci	(Optional) Displays Payment Card Industry (PCI) memory statistics.	
processor	(Optional) Displays processor memory statistics.	
allocating-process	(Optional) Displays the allocating process name.	
totals	(Optional) Displays the total memory. This keyword is used with the **allocating-process** and **dead** keywords.	
dead	(Optional) Displays the dead memory information.	
free	(Optional) Displays free memory information.	
physical	(Optional) Displays physical memory information.	

Command Modes Privileged EXEC (#)

Command History

Release	Modification
12.2SY	This command was introduced to support Software Modularity images.

Usage Guidelines Detailed output of all applicable processes on the device is displayed with this command. Use the optional arguments and keywords to display specific detailed information.

Examples   The following is partial sample output from the **show memory detailed all** command:

```
Router# show memory detailed all

System Memory: 2097152K total, 1303301K used, 793851K free, 0K kernel reserved
Lowest(b)    : 812904448

Process kernel, type POSIX, PID = 1

Process sbin/chkptd.proc, type POSIX, PID = 16386
    3448K total, 2516K text, 672K data, 16K stack, 244K dynamic
    204800 heapsize, 108612 allocated, 56448 free

Address       Bytes What

0x4813D000     4096 Unknown type
0x4813E000   126976 Program Stack (pages not allocated)
0x4815D000     4096 Program Stack
0x4815E000     4096 Unknown type
0x4815F000   126976 Program Stack (pages not allocated)
0x4817E000     4096 Program Stack
0x4817F000     4096 Unknown type
0x48180000   516096 Program Stack (pages not allocated)
0x481FE000     8192 Program Stack
0x48200000     8192 Shared object data "sbin/chkptd.proc"
0x48202000     4096 Shared object data "sbin/chkptd.proc"
0x48203000   237568 Heap Memory
0x68000000   917504 Shared Memory
0x680E0000 66191360 Shared Memory
0x80100000     8192 Shared Memory
0x80102000     4096 Shared Memory
0x80103000   397312 Shared Memory
0x80164000     4096 Shared Memory
0xFE300000   614400 Shared object text "lib/libc.so"
0xFE396000    45056 Shared object data "lib/libc.so"
0xFE3A1000    12288 Heap Memory
0xFE3A4000    20480 Shared object text "lib/s72044-adventerprisek9_dbg-014-dso-"
0xFE3A9000    65536 Shared object data "/dev/zero"
0xFE3B9000     4096 Shared object data "lib/s72044-adventerprisek9_dbg-014-dso-"
0xFE3BA000   434176 Shared object text "lib/s72044-adventerprisek9_dbg-001-dso-"
0xFE424000    65536 Shared object data "/dev/zero".
.
.
.
.
```

The first section of the display shows system summary information. Table 21 describes the significant fields shown in the first section of the display.

Table 21   show memory detailed all Field Descriptions—First Section

Field	Description
total	Total amount of memory on the device, in kilobytes.
used	Amount of memory in use, in kilobytes.
free	Amount of memory not in use, in kilobytes.
kernel reserved	Amount of memory reserved by the kernel, in kilobytes.

The second section of the display includes process summary statistics about the activities of the system memory allocator. Table 22 describes the significant fields shown in the second section of the display.

Table 22   show memory detailed all Field Descriptions—Second Section

Field	Description
Process	Process name and path.
type	Type of process: POSIX or Cisco IOS.
PID	Process ID.
total	Total amount of memory used by the specified process, in kilobytes.
text	Amount of memory, in kilobytes, used by the text segment of the specified process.
data	Amount of memory, in kilobytes, used by the data segment of the specified process.
stack	Amount of memory, in kilobytes, used by the stack segment of the specified process.
dynamic	Amount of memory, in kilobytes, used by the dynamic segment of the specified process.
heapsize	Size of the process heap. Note that the Cisco IOS memory management library allocates heap dynamically. This is shown in the Cisco IOS memory details that follow the POSIX memory display.
allocated	Amount of memory, in kilobytes, allocated from the heap.
free	Amount of free memory, in kilobytes, in the heap for the specified process.

The third section of the display shows process perspective memory information. Table 23 describes the significant fields shown in the third section of the display.

Table 23   show memory detailed all Field Descriptions—Third Section

Field	Description
Address	Hexadecimal address of block.
Bytes	Size of block (in bytes).
What	Type of memory segment that owns the block, or "(fragment)" if the block is a fragment, or "(coalesced)" if the block was coalesced from adjacent free blocks.

The following is partial output from the **show memory detailed all io** command:

```
Router# show memory detailed all io

System Memory: 2097152K total, 1302133K used, 795019K free, 0K kernel reserved
Lowest(b)    : 812314624

Process sbin/ios-base, type IOS, PID = 16425
    257172K total, 139268K text, 77292K data, 168K stack, 40444K dynamic

        I/O memory

 Address      Bytes     Prev     Next Ref     PrevF    NextF Alloc PC  what
```

```
70000000 0000000024 00000000 70000050 000  FAE73E24 0        00000000  (fragmen)
70000050 0000000808 70000000 700003B0 001  -------- -------- F9D15B78  *Packet *
700003B0 0000000808 70000050 70000710 001  -------- -------- F9D15B78  *Packet *
70000710 0000000808 700003B0 70000A70 001  -------- -------- F9D15B78  *Packet *
70000A70 0000000808 70000710 70000DD0 001  -------- -------- F9D15B78  *Packet *
70000DD0 0000000808 70000A70 70001130 001  -------- -------- F9D15B78  *Packet *
70001130 0000000808 70000DD0 70001490 001  -------- -------- F9D15B78  *Packet *
 Address      Bytes     Prev     Next Ref     PrevF    NextF Alloc PC  what
70001490 0000000808 70001130 700017F0 001  -------- -------- F9D15B78  *Packet *
700017F0 0000000808 70001490 70001B50 001  -------- -------- F9D15B78  *Packet *
70001B50 0000000808 700017F0 70001EB0 001  -------- -------- F9D15B78  *Packet *
70001EB0 0000000808 70001B50 70002210 001  -------- -------- F9D15B78  *Packet *
70002210 0000000808 70001EB0 70002570 001  -------- -------- F9D15B78  *Packet *
70002570 0000000808 70002210 700028D0 001  -------- -------- F9D15B78  *Packet *
700028D0 0000000808 70002570 70002C30 001  -------- -------- F9D15B78  *Packet *
70002C30 0000000808 700028D0 70002F90 001  -------- -------- F9D15B78  *Packet *
70002F90 0000000808 70002C30 700032F0 001  -------- -------- F9D15B78  *Packet *
700032F0 0000000808 70002F90 70003650 001  -------- -------- F9D15B78  *Packet *
70003650 0000000808 700032F0 700039B0 001  -------- -------- F9D15B78  *Packet *

Process sbin/test_proc.iosproc, type IOS, PID = 20551

         Processor memory

 Address      Bytes     Prev     Next Ref     PrevF    NextF Alloc PC  what
4821B058 0000020000 00000000 4821FEB0 001  -------- -------- FE5306C0  Managed s
4821FEB0 0000002560 4821B058 482208E8 001  -------- -------- FE4597B4  *Init*
482208E8 0000005000 4821FEB0 48221CA8 001  -------- -------- FE60DA80  List Heas
48221CA8 0000000088 482208E8 48221D38 001  -------- -------- FE44559C  *Init*
48221D38 0000000088 48221CA8 48221DC8 001  -------- -------- FE44559C  *Init*
48221DC8 0000000024 48221D38 48221E18 001  -------- -------- FE445A6C  *Init*
.
.
.
```

Table 24 describes the significant fields shown in the display.

Table 24   ***show memory detailed all Field Descriptions—Third Section***

Field	Description
Address	Hexadecimal address of block.
Bytes	Size of block (in bytes).
Prev	Address of previous block (should match address on previous line).
Next	Address of next block (should match address on next line).
PrevF	Address of previous free block (if free).
NextF	Address of next free block (if free).
Alloc PC	Address of the system call that allocated the block.
what	Type of memory segment that owns the block, or "(fragment)" if the block is a fragment, or "(coalesced)" if the block was coalesced from adjacent free blocks.

Related Commands

Command	Description
show memory detailed	Displays detailed memory information.

show pakman

To display Packet Manager details, use the **show pakman** command in privileged EXEC mode.

> **show pakman** {**clients** | **statistics**}

Syntax Description

clients	Displays all the clients connected to the Packet Manager.
statistics	Displays Packet Manager statistics.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXH1	This command was introduced in a release earlier than Cisco IOS Release 12.2(33)SXH1.

Usage Guidelines

The **show pakman** command is supported only on Cisco IOS Software Modularity images. The output of the **show pakman statistics** command provides more information for debugging packet memory issues.

Examples

The following is sample output from the **show pakman clients** command:

```
Router# show pakman clients

Connected clients to the Packet Manager
=======================================

Pid      Coid      Process Name                 usage
-----    ----      ------------                 -----
24599    196615    ios-base                     126/7312
24613    8         fh_server.proc               0/7312
24615    9         call_home.proc               0/7312
24616    10        inetd.proc                   0/7312
24618    131083    ipfs_daemon.proc             0/7312
24617    12        tcp.proc                     0/7312
24620    13        raw_ip.proc                  0/7312
24621    15        udp.proc                     0/7312
24623    18        cdp2.iosproc                 0/7312
24622    19        iprouting.iosproc            1/7312
```

Table 26 describes the significant fields shown in the display.

Table 25   show pakman clients Field Descriptions

Field	Description
Pid	Process ID.
Coid	Client connection ID.

Table 25   show pakman clients Field Descriptions (continued)

Field	Description
Process Name	Name of the process.
usage	Number of packet buffers used by the client and the total number of packets currently available in the system.

The following sample output from the **show pakman statistics** command displays the different memory regions used by the Packet Manager for various uses. Some regions, such as the temporary packet and subblock region, can have multiple regions because regions can be added when the system requires more packet headers or subblocks. The region provided for data buffers is fixed and does not grow.

```
Router# show pakman statistics

Packet Manager Regions
=====================
Packet Shared Context         0x6C000000 -> 0x6C003078 (12408 bytes)
Permanent Header Region       0x6C004000 -> 0x6C0F8FFF (1003520 bytes)
Temp Header & Subblock Region 0x6C0F9000 -> 0x6C175FFF (512000 bytes)
Data Buffer Region            0x6400000  -> 0x7FFFFFF  (29360128 bytes)

Temp Header & Subblock Memory Block Manager Statistics
======================================================


Blocksize   Allocs      Frees       Inuse       Cached
136         0           0           0           1
272         0           0           0           0
544         7           0           7           1
1088        0           0           0           0
2176        0           0           0           1
4352        0           0           0           0
8704        0           0           0           0
17408       0           0           0           1
34816       0           0           0           0
69632       0           0           0           1
139264      0           0           0           1
278528      0           0           0           1
557056      0           0           0           0
1114112     0           0           0           0
Buffer memory usage: free 507688 total 512000 largest block 278484 in-use 4312.
Memory Block grows:0 shrinks:0

Data Buffer Memory Block Manager Statistics
===========================================


Blocksize   Allocs      Frees       Inuse       Cached
448         66          16          50          0
896         31          6           25          7
1792        56          6           50          5
3584        0           0           0           1
7168        74          0           74          2
14336       0           0           0           1
28672       0           0           0           0
57344       0           0           0           1
114688      0           0           0           1
229376      0           0           0           0
458752      0           0           0           0
917504      0           0           0           1
```

```
1835008    0          0          0          1
3670016    1          0          1          6
Buffer memory usage: free 24992128 total 29360128 largest block 3669972 in-use 4368000.

Threshold tracking is enabled, notify state = 0, memory in-use 4335308.
packet count=0 max=0 limit=0 (limited 0 times) drops=0
```

Table 26 describes the significant fields shown in the display.

Table 26   show pakman statistics Field Descriptions

Field	Description
Packet Manager Regions	Memory regions used by the Packet Manager.
Blocksize	Packet Manager allocates data in a number of fixed block sizes.
Allocs	The number of times the given block size has been allocated.
Frees	The number of times the given block size has been freed.
Inuse	The number of blocks of the given size that are currently in use.
Cached	The number of blocks of the given size that are currently being cached.

Related Commands

Command	Description
show buffers	Display statistics for the buffer pools on the network server when Cisco IOS or Cisco IOS Software Modularity images are running.

show processes

To display information about the active Cisco IOS, Cisco IOS XE, or the Cisco IOS Software Modularity POSIX-style processes, use the **show processes** command in user EXEC or privileged EXEC mode.

> **show processes** [**history** | *process-id* | **timercheck**]

Syntax Description

history	(Optional) For Cisco IOS processes only. Displays the process history in an ordered format.
process-id	(Optional) For Cisco IOS processes only. An integer that specifies the process for which memory and CPU utilization data will be returned.
timercheck	(Optional) For Cisco IOS processes only. Displays the processes configured for a timer check.

Command Modes

User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
10.0	This command was introduced.
12.2(2)T	This command was modified. The **history** keyword was added.
12.3(2)T	This command was modified. The *process-id* argument was added.
12.2(18)SXF4	This command was modified. The syntax was modified to support Cisco IOS Software Modularity images.
12.3(14)T	This command was modified. The **timercheck** keyword was added.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.
Cisco IOS XE Release 3.1.0SG	This command was introduced on the Cisco Catalyst 4500e series switches.
15.1(2)S	This command was integrated into Cisco IOS Release 15.1(2)S.

Usage Guidelines

Cisco IOS Software Modularity

Although no optional keywords or arguments are supported for the base **show processes** command when a Software Modularity image is running, more details about processes are displayed using the **show processes cpu**, **show processes detailed**, **show processes kernel**, and **show processes memory** commands.

Examples

Example output varies between Cisco IOS software images and Cisco IOS Software Modularity software images. The following sections show output examples for each image:

- Cisco IOS Software
- Cisco IOS Software Modularity

- Cisco Catalyst 4500e Series Switches Running Cisco IOS XE Software

Cisco IOS Software

The following is sample output from the **show processes** command:

```
Router# show processes

CPU utilization for five seconds: 21%/0%; one minute: 2%; five minutes: 2%
 PID QTy       PC Runtime (ms)    Invoked   uSecs    Stacks TTY Process
   1 Cwe 606E9FCC         0          1        0 5600/6000    0 Chunk Manager
   2 Csp 607180F0         0     121055        0 2608/3000    0 Load Meter
   3 M*         0          8         90       88 9772/12000   0 Exec
   4 Mwe 619CB674         0          1        023512/24000   0 EDDRI_MAIN
   5 Lst 606F6AA4     82064      61496     1334 5668/6000    0 Check heaps
   6 Cwe 606FD444         0        127        0 5588/6000    0 Pool Manager
   7 Lwe 6060B364         0          1        0 5764/6000    0 AAA_SERVER_DEADT
   8 Mst 6063212C         0          2        0 5564/6000    0 Timers
   9 Mwe 600109D4         0          2        0 5560/6000    0 Serial Backgroun
  10 Mwe 60234848         0          2        0 5564/6000    0 ATM Idle Timer
  11 Mwe 602B75F0         0          2        0 8564/9000    0 ATM AutoVC Perio
  12 Mwe 602B7054         0          2        0 5560/6000    0 ATM VC Auto Crea
  13 Mwe 606068B8         0          2        0 5552/6000    0 AAA high-capacit
  14 Msi 607BABA4     251264     605013      415 5628/6000    0 EnvMon
  15 Mwe 607BFF8C         0          1        0 8600/9000    0 OIR Handler
  16 Mwe 607D407C         0      10089        0 5676/6000    0 IPC Dynamic Cach
  17 Mwe 607CD03C         0          1        0 5632/6000    0 IPC Zone Manager
  18 Mwe 607CCD80         0     605014        0 5708/6000    0 IPC Periodic Tim
  19 Mwe 607CCD24         0     605014        0 5704/6000    0 IPC Deferred Por
  20 Mwe 607CCE2C         0          1        0 5596/6000    0 IPC Seat Manager
```

Table 27 describes the significant fields shown in the display.

Table 27   ***show processes Field Descriptions***

Field	Description
CPU utilization for five seconds	CPU utilization for the last 5 seconds. The second number indicates the percentage of CPU time spent at the interrupt level.
one minute	CPU utilization for the last minute.
five minutes	CPU utilization for the last 5 minutes.
PID	Process ID.
Q	Process queue priority. Possible values: C (critical), H (high), M (medium), and L (low).

Table 27 *show processes Field Descriptions (continued)*

Field	Description
Ty	Scheduler test. Possible values: • * (currently running) • E (waiting for an event) • S (ready to run, voluntarily relinquished processor) • rd (ready to run, wakeup conditions have occurred) • we (waiting for an event) • sa (sleeping until an absolute time) • si (sleeping for a time interval) • sp (sleeping for a time interval as an alternate call • st (sleeping until a timer expires) • hg (hung: the process will never execute again) • xx (dead: the process has terminated, but has not yet been deleted)
PC	Current program counter.
Runtime (ms)	CPU time that the process has used (in milliseconds).
Invoked	Number of times that the process has been invoked.
uSecs	Microseconds of CPU time for each process invocation.
Stacks	Low water mark/Total stack space available (in bytes).
TTY	Terminal that controls the process.
Process	Name of the process.

Note Because platforms have a 4- to 8-millisecond clock resolution, run times are considered reliable only after a large number of invocations or a reasonable, measured run time.

For a list of process descriptions, see
http://www.cisco.com/en/US/products/sw/iosswrel/ps1828/products_tech_note09186a00800a65d0.shtml.

The following is sample output from the **show processes history** command:

```
Router# show processes history

PID Exectime(ms) Caller PC Process Name
   3        12 0x0       Exec
  16         0 0x603F4DEC GraphIt
  21         0 0x603CFEF4 TTY Background
  22         0 0x6042FD7C Per-Second Jobs
  67         0 0x6015CD38 SMT input
  39         0 0x60178804 FBM Timer
  16         0 0x603F4DEC GraphIt
  21         0 0x603CFEF4 TTY Background
  22         0 0x6042FD7C Per-Second Jobs
  16         0 0x603F4DEC GraphIt
  21         0 0x603CFEF4 TTY Background
```

```
22          0 0x6042FD7C Per-Second Jobs
67          0 0x6015CD38 SMT input
39          0 0x60178804 FBM Timer
24          0 0x60425070 Compute load avgs
11          0 0x605210A8 ARP Input
69          0 0x605FDAF4 DHCPD Database
69          0 0x605FD568 DHCPD Database
51          0 0x60670B3C IP Cache Ager
69          0 0x605FD568 DHCPD Database
36          0 0x606E96DC SSS Test Client
69          0 0x605FD568 DHCPD Database
--More--
```

Table 28 describes the significant fields shown in the display.

Table 28   *show processes history Field Descriptions*

Field	Description
PID	Process ID.
Exectime (ms)	Execution time (in milliseconds) of the most recent run or the total execution time of the most recent consecutive runs.
Caller PC	Current program counter of this process before it was suspended.
Process Name	Name of the process.

The following is sample output from the **show processes** *process-id* command:

```
Router# show processes 6

Process ID 6 [Pool Manager], TTY 0
Memory usage [in bytes]
 Holding: 921148, Maximum: 940024, Allocated: 84431264, Freed: 99432136
 Getbufs: 0, Retbufs: 0, Stack: 12345/67890
CPU usage
 PC: 0x60887600, Invoked: 188, Giveups: 100, uSec: 24
 5Sec: 3.03%, 1Min: 2.98%, 5Min: 1.55%, Average: 0.58%,
 Age: 662314 msec, Runtime: 3841 msec
 State: Running, Priority: Normal
```

Table 29 describes the significant fields shown in the display.

Table 29   *show processes process-id Field Descriptions*

Field	Description
Process ID	Process ID number and process name.
TTY	Terminal that controls the process.
Memory usage [in bytes]	This section contains fields that show the memory used by the specified process.
Holding	Amount of memory currently allocated to the process.
Maximum	Maximum amount of memory allocated to the process since its invocation.
Allocated	Bytes of memory allocated by the process.
Freed	Bytes of memory freed by the process.
Getbufs	Number of times that the process has requested a packet buffer.

Table 29   *show processes process-id Field Descriptions (continued)*

Field	Description
Retbufs	Number of times that the process has relinquished a packet buffer.
Stack	Low water mark/Total stack space available (in bytes).
CPU usage	This section contains fields that show the CPU resources used by the specified process.
PC	Current program counter of this process before it was suspended.
Invoked	Number of times that the process executed since its invocation.
Giveups	Number of times that the process voluntarily gave up the CPU.
uSec	Microseconds of CPU time for each process invocation.
5Sec	CPU utilization by process in the last five seconds.
1Min	CPU utilization by process in the last minute.
5Min	CPU utilization by process in the last five minutes.
Average	The average amount of CPU utilization by the process since its invocation.
Age	Milliseconds since the process was invoked.
Runtime	CPU time that the process has used (in milliseconds).
State	Current state of the process. Possible values: Running, Waiting for Event, Sleeping (Mgd Timer), Sleeping (Periodic), Ready, Idle, Dead.
Priority	The priority of the process. Possible values: Low, Normal, High.

Cisco IOS Software Modularity

The following is sample output from the **show processes** command when a Cisco IOS Software Modularity image is running:

```
Router# show processes

Total CPU utilization for 5 seconds: 99.7%; 1 minute: 98.9%; 5 minutes: 86.5%
PID    TID  Prio STATE       Blocked   Stack           CPU   Name
1      1    0    Ready                 0     (128K)   2m28s   procnto-cisco
1      2    63   Receive     1         0     (128K)   0.000   procnto-cisco
1      3    10   Receive     1         0     (128K)   0.000   procnto-cisco
1      4    11   Receive     1         0     (128K)   1.848   procnto-cisco
1      5    63   Receive     1         0     (128K)   0.000   procnto-cisco
1      6    63   Receive     1         0     (128K)   0.000   procnto-cisco
12290  1    10   Receive     1         12288 (128K)   0.080   chkptd.proc
12290  2    10   Receive     8         12288 (128K)   0.000   chkptd.proc
3      1    15   Condvar     1027388   12288 (128K)   0.016   qdelogger
3      2    15   Receive     1         12288 (128K)   0.004   qdelogger
3      3    16   Condvar     1040024   12288 (128K)   0.004   qdelogger
4      1    10   Receive     1         4096  (128K)   0.016   devc-pty
6      1    62   Receive     1         8192  (128K)   0.256   devc-ser2681
6      2    63   Intr                  8192  (128K)   0.663   devc-ser2681
7      1    10   Receive     1         32768 (128K)   0.080   dumper.proc
7      2    10   Receive     1         32768 (128K)   0.008   dumper.proc
7      3    10   Receive     1         32768 (128K)   0.000   dumper.proc
7      4    10   Receive     1         32768 (128K)   0.020   dumper.proc
7      5    10   Receive     1         32768 (128K)   0.008   dumper.proc
4104   2    10   Receive     1         12288 (128K)   0.000   pipe
4104   3    10   Receive     1         12288 (128K)   0.000   pipe
--More--
```

Table 30 describes the significant fields shown in the display.

Table 30   show processes (Software Modularity) Field Descriptions

Field	Description
PID	Process ID.
TID	Task ID.
Prio	Process priority.
STATE	Current state of the process.
Blocked	Thread (with given process ID) that is currently blocked by the process.
Stack	Size, in kilobytes, of the memory stack.
CPU	CPU time, in minutes and seconds, used by the process.
Name	Process name.

Cisco Catalyst 4500e Series Switches Running Cisco IOS XE Software

The following is sample output from the **show processes** command:

```
Switch# show processes

CPU utilization for five seconds: 1%; one minute: 4%;  five minutes: 3%
PID     TID     Runtime(ms) Invoked    uSecs  Stacks      Process
1               935         596        156971 84/8192     init
2               0           79         10405  0/8192      kthreadd
3               12          2206       5578   0/8192      migration/0
4               12          772        15601  0/8192      ksoftirqd/0
5               6           1089       6357   0/8192      migration/1
6               14          877        16484  0/8192      ksoftirqd/1
7               15          374        42475  0/8192      events/0
8               9           333        27531  0/8192      events/1
9               5           637        9070   0/8192      khelper
61              28          45         628533 0/8192      kblockd/0
62              80          175        461994 0/8192      kblockd/1
75              0           21         1238   0/8192      khubd
78              0           23         652    0/8192      kseriod
83              7           26         271115 0/8192      kmmcd
120             0           25         320    0/8192      pdflush
121             12          68         190382 0/8192      pdflush
122             0           29         172    0/8192      kswapd0
123             0           31         161    0/8192      aio/0
124             0           33         121    0/8192      aio/1
291             0           35         142    0/8192      kpsmoused
309             0           37         135    0/8192      rpciod/0
310             0           39         128    0/8192      rpciod/1
354             71          425        167583 84/8192     udevd
700             117         3257       35991  0/8192      loop1
716             0           55         1145   0/8192      loop2
732             115         2336       49574  0/8192      loop3
2203            86          627        138015 84/8192     dbus-daemon
2539            0           432        1974   84/8192     portmap
2545            0           434        2011   84/8192     portmap
2588            1           450        2384   84/8192     sshd
2602            2           444        6677   84/8192     xinetd
2606            1           444        3191   84/8192     xinetd
3757            0           71         70     84/8192     vsi work/0
--More--
```

Table 31 describes the significant fields shown in the display.

Table 31   *show processes (Software Modularity) Field Descriptions*

Field	Description
CPU utilization for five seconds	CPU utilization for the last 5 seconds. The "3%" indicates the percentage of CPU time spent at the interrupt level.
one minute	CPU utilization for the last minute.
five minutes	CPU utilization for the last 5 minutes.
PID	Process ID.
TID	Thread ID.
Runtime(ms)	CPU time that the process has used (in milliseconds).
Invoked	Number of times that the process has been invoked.
uSecs	Microseconds of CPU time for each process invocation.
Stacks	Size, in kilobytes, of the memory stack.
Process	Process name.

Related Commands

Command	Description
show processes cpu	Displays detailed CPU utilization statistics (CPU use per process) when a Software Modularity image is running.
show processes detailed	Displays detailed information about POSIX and Cisco IOS processes when a Software Modularity image is running.
show processes kernel	Displays information about System Manager kernel processes when a Software Modularity image is running.
show processes memory	Displays the amount of system memory used per system process.

show processes cpu

To display detailed CPU utilization statistics (CPU use per process) when Cisco IOS or Cisco IOS Software Modularity images are running, use the **show processes cpu** command in user EXEC or privileged EXEC mode.

Cisco IOS Software

> **show processes cpu** [**history** [**table**] | **sorted** [**1min** | **5min** | **5sec**]]

Cisco IOS Software Modularity

> **show processes cpu** [**detailed** [*process-id* | *process-name*] | **history**]

Syntax Description		
	history	(Optional) Displays CPU history in a graph format.
	table	(Optional) Displays CPU history in a table format.
	sorted	(Optional) For Cisco IOS images only. Displays CPU utilization sorted by percentage.
	1min	(Optional) Sorts CPU utilization based on 1 minute utilization.
	5min	(Optional) Sorts CPU utilization based on 5 minutes utilization.
	5sec	(Optional) Sorts CPU utilization based on 5 seconds utilization.
	detailed	(Optional) For Cisco IOS Software Modularity images only. Displays more detailed information about Cisco IOS processes (not for POSIX processes).
	process-id	(Optional) For Cisco IOS Software Modularity images only. Process identifier.
	process-name	(Optional) For Cisco IOS Software Modularity images only. Process name.

Command Modes   User EXEC (>)
Privileged EXEC (#)

Command History	Release	Modification
	12.0	This command was introduced.
	12.2(2)T	This command was modified. The **history** keyword was added.
	12.3(8)	This command was enhanced to display Address Resolution Protocol (ARP) output.
	12.3(14)T	This command was enhanced to display ARP output.
	12.2(18)SXF4	This command was enhanced to support Cisco IOS Software Modularity images.
	12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
	12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
	12.2(33)SCB3	This command was integrated into Cisco IOS Release 12.2(33)SCB3. Support was added for Cisco uBR10012 and uBR7200 routers.
	Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.

Usage Guidelines

Cisco IOS Software

If you use the optional **history** keyword, three graphs are displayed for Cisco IOS images:

- CPU utilization for the last 60 seconds
- CPU utilization for the last 60 minutes
- CPU utilization for the last 72 hours

Maximum usage is measured and recorded every second; average usage is calculated on periods of more than one second. Consistently high CPU utilization over an extended period indicates a problem. Use the **show processes cpu** command to troubleshoot. Also, you can use the output of this command in the Cisco Output Interpreter tool to display potential issues and fixes. Output Interpreter is available to registered users of Cisco.com who are logged in and have Java Script enabled.

For a list of system processes, go to
http://www.cisco.com/en/US/products/sw/iosswrel/ps1828/products_tech_note09186a00800a65d0.shtml.

Cisco IOS Software Modularity

Cisco IOS Software Modularity images display only one graph that shows the CPU utilization for the last 60 minutes. The horizontal axis shows times (for example, 0, 5, 10, 15 minutes), and the vertical axis shows total percentage of CPU utilization (0 to 100 percent).

Examples

Example output varies between Cisco IOS software images and Cisco IOS Software Modularity software images. The following sections show output examples for each image:

- Cisco IOS Software
- Cisco IOS Software Modularity

Cisco IOS Software

The following is sample output from the **show processes cpu** command without keywords:

```
Router# show processes cpu

CPU utilization for five seconds: 5%/2%; one minute: 3%; five minutes: 2%
 PID  Runtime (ms)    Invoked    uSecs   5Sec   1Min   5Min  TTY  Process
   1         1736         58    29931    0%     0%     0%   0   Check heaps
   2           68        585      116  1.00% 1.00%     0%   0   IP Input
   3            0        744        0    0%     0%     0%   0   TCP Timer
   4            0          2        0    0%     0%     0%   0   TCP Protocols
   5            0          1        0    0%     0%     0%   0   BOOTP Server
   6           16        130      123    0%     0%     0%   0   ARP Input
   7            0          1        0    0%     0%     0%   0   Probe Input
   8            0          7        0    0%     0%     0%   0   MOP Protocols
   9            0          2        0    0%     0%     0%   0   Timers
  10          692         64    10812    0%     0%     0%   0   Net Background
  11            0          5        0    0%     0%     0%   0   Logger
  12            0         38        0    0%     0%     0%   0   BGP Open
  13            0          1        0    0%     0%     0%   0   Net Input
  14          540       3466      155    0%     0%     0%   0   TTY Background
  15            0          1        0    0%     0%     0%   0   BGP I/O
  16         5100       1367     3730    0%     0%     0%   0   IGRP Router
  17           88       4232       20  0.20% 1.00%     0%   0   BGP Router
  18          152      14650       10    0%     0%     0%   0   BGP Scanner
  19          224         99     2262    0%     0%  1.00%   0   Exec
```

The following is sample output of the one-hour portion of the output. The Y-axis of the graph is the CPU utilization. The X-axis of the graph is the increment within the time period displayed in the graph. This example shows the individual minutes during the previous hour. The most recent measurement is on the left of the X-axis.

```
Router# show processes cpu history

!--- One minute output omitted

666577686575667667666666766767767676666676676776766656666 7
637801619899351370977199144373235868993274085826964392261 3
100
 90
 80        *   *                      *  *       *   *  *   *
 70   * * ***** *  ** ***** ***   **** ******   *  *******     * *
 60   #***##*##*#***#####*#*###*****#*###*#*#*##*#*##*#*##*****#
 50   #########################################################
 40   #########################################################
 30   #########################################################
 20   #########################################################
 10   #########################################################
     0....5....1....1....2....2....3....3....4....4....5....5....
              0    5    0    5    0    5    0    5    0    5
               CPU% per minute (last 60 minutes)
               * = maximum CPU% # = average CPU%

!--- 72-hour output omitted
```

The top two rows, read vertically, display the highest percentage of CPU utilization recorded during the time increment. In this example, the CPU utilization for the last minute recorded is 66 percent. The device may have reached 66 percent only once during that minute, or it may have reached 66 percent multiple times. The device records only the peak reached during the time increment and the average over the course of that increment.

The following is sample output from the **show processes cpu** command on a Cisco uBR10012 router:

```
Router# show processes cpu

CPU utilization for five seconds: 2%/0%; one minute: 2%; five minutes: 2%
 PID Runtime(ms)   Invoked     uSecs   5Sec   1Min   5Min TTY Process
   1         8        471        16   0.00%  0.00%  0.00%   0 Chunk Manager
   2         4        472         8   0.00%  0.00%  0.00%   0 Load Meter
   3         0          1         0   0.00%  0.00%  0.00%   0 IPC 0x50000 Vers
   4         0         10         0   0.00%  0.00%  0.00%   0 C10K Card Event
   5         0         65         0   0.00%  0.00%  0.00%   0 Retransmission o
   6         0          5         0   0.00%  0.00%  0.00%   0 IPC ISSU Dispatc
   7      5112        472     10830   0.63%  0.18%  0.18%   0 Check heaps
   8         0          1         0   0.00%  0.00%  0.00%   0 Pool Manager
   9         0          2         0   0.00%  0.00%  0.00%   0 Timers
  10         0          2         0   0.00%  0.00%  0.00%   0 Serial Backgroun
  11         0        786         0   0.00%  0.00%  0.00%   0 WBCMTS process
  12         0          1         0   0.00%  0.00%  0.00%   0 AAA_SERVER_DEADT
  13         0          1         0   0.00%  0.00%  0.00%   0 Policy Manager
  14         0          1         0   0.00%  0.00%  0.00%   0 Crash writer
  15         0          1         0   0.00%  0.00%  0.00%   0 RO Notify Timers
  16         0          1         0   0.00%  0.00%  0.00%   0 RMI RM Notify Wa
  17         0       2364         0   0.00%  0.00%  0.00%   0 Facility Alarm
  18         0         41         0   0.00%  0.00%  0.00%   0 IPC Dynamic Cach
```

The following is sample output from the **show processes cpu** command that shows an ARP probe process:

```
Router# show processes cpu | include ARP

17      38140    389690        97  0.00%  0.00%  0.00%   0 ARP Input
36          0         1         0  0.00%  0.00%  0.00%   0 IP ARP Probe
40          0         1         0  0.00%  0.00%  0.00%   0 ATM ARP INPUT
80          0         1         0  0.00%  0.00%  0.00%   0 RARP Input
114         0         1         0  0.00%  0.00%  0.00%   0 FR ARP
```

Table 32 describes the fields shown in the output.

Table 32   show processes cpu Field Descriptions

Field	Description
CPU utilization for five seconds	CPU utilization for the last 5 seconds. The second number indicates the percent of CPU time spent at the interrupt level.
1 minute	CPU utilization for the last minute.
5 minutes	CPU utilization for the last 5 minutes.
PID	Process ID.
Runtime (ms)	CPU time that the process has used (in milliseconds).
Invoked	Number of times that the process has been invoked.
uSecs	Microseconds of CPU time for each process invocation.
5Sec	CPU utilization by task in the last 5 seconds.
1Min	CPU utilization by task in the last minute.
5Min	CPU utilization by task in the last 5 minutes.
TTY	Terminal that controls the process.
Process	Name of the process.

Note Because platforms have a 4- to 8-millisecond clock resolution, run times are considered reliable only after several invocations or a reasonable, measured run time.

Cisco IOS Software Modularity

The following is sample output from the **show processes cpu** command when a Software Modularity image is running:

```
Router# show processes cpu

Total CPU utilization for 5 seconds: 99.6%; 1 minute: 98.5%; 5 minutes: 85.3%
PID     5Sec    1Min    5Min Process
1       0.0%    0.1%    0.8% kernel
3       0.0%    0.0%    0.0% qdelogger
4       0.0%    0.0%    0.0% devc-pty
6       0.7%    0.2%    0.1% devc-ser2681
7       0.0%    0.0%    0.0% dumper.proc
4104    0.0%    0.0%    0.0% pipe
8201    0.0%    0.0%    0.0% mqueue
8202    0.0%    0.0%    0.0% fsdev.proc
8203    0.0%    0.0%    0.0% flashfs_hes_slot1.proc
8204    0.0%    0.0%    0.0% flashfs_hes_slot0.proc
```

```
8205     0.0%    0.0%      0.0% flashfs_hes_bootflash.proc
8206     0.0%    0.0%      0.0% dfs_disk2.proc
8207     0.0%    0.0%      0.0% dfs_disk1.proc
8208     0.0%    0.0%      0.0% dfs_disk0.proc
8209     0.0%    0.0%      0.0% ldcache.proc
8210     0.0%    0.0%      0.0% watchdog.proc
8211     0.0%    0.0%      0.0% syslogd.proc
8212     0.0%    0.0%      0.0% name_svr.proc
8213     0.0%    0.1%      0.0% wdsysmon.proc
8214     0.0%    0.0%      0.0% sysmgr.proc
8215     0.0%    0.0%      0.0% kosh.proc
12290    0.0%    0.0%      0.0% chkptd.proc
12312    0.0%    0.0%      0.0% sysmgr.proc
12313    0.0%    0.0%      0.0% syslog_dev.proc
12314    0.0%    0.0%      0.0% itrace_exec.proc
12315    0.0%    0.0%      0.0% packet.proc
12316    0.0%    0.0%      0.0% installer.proc
12317   29.1%   28.5%     19.6% ios-base
12318    0.0%    0.0%      0.0% fh_fd_oir.proc
12319    0.0%    0.0%      0.1% fh_fd_cli.proc
12320    0.0%    0.0%      0.0% fh_metric_dir.proc
12321    0.0%    0.0%      0.0% fh_fd_snmp.proc
12322    0.0%    0.0%      0.0% fh_fd_none.proc
12323    0.0%    0.0%      0.0% fh_fd_intf.proc
12324   48.5%   48.5%     35.8% iprouting.iosproc
12325    0.0%    0.0%      0.0% fh_fd_timer.proc
12326    0.0%    0.0%      0.0% fh_fd_ioswd.proc
12327    0.0%    0.0%      0.0% fh_fd_counter.proc
12328    0.0%    0.0%      0.0% fh_fd_rf.proc
12329    0.0%    0.0%      0.0% fh_server.proc
12330    0.0%    0.0%      0.0% cdp2.iosproc
12331    0.0%    0.0%      0.0% fh_policy_dir.proc
12332    0.0%    0.0%      0.0% ipfs_daemon.proc
12333    0.0%    0.0%      0.0% raw_ip.proc
12334    0.0%    0.0%      0.0% inetd.proc
12335   19.1%   20.4%     12.6% tcp.proc
12336    0.0%    0.0%      0.0% udp.proc
```

Table 33 describes the significant fields shown in the display.

Table 33   ***show processes cpu (Software Modularity) Field Descriptions***

Field	Description
Total CPU utilization for five seconds	Total CPU utilization for the last 5 seconds. The second number indicates the percent of CPU time spent at the interrupt level.
1 minute	CPU utilization for the last minute.
5 minutes	CPU utilization for the last 5 minutes.
PID	Process ID.
5Sec	Percentage of CPU time spent at the interrupt level for this process during the last five seconds.
1Min	Percentage of CPU time spent at the interrupt level for this process during the last minute.
5Min	Percentage of CPU time spent at the interrupt level for this process during the last five minutes.
Process	Process name.

The following is partial sample output from the **show processes cpu** command with the **detailed** keyword when a Software Modularity image is running:

```
Router# show processes cpu detailed

Total CPU utilization for 5 seconds: 99.6%; 1 minute: 99.3%; 5 minutes: 88.6%
PID/TID   5Sec    1Min     5Min Process          Prio  STATE       CPU
1         0.0%    0.7%     0.7% kernel                              8.900
      1   0.4%    0.7%    11.4% [idle thread]        0  Ready       2m28s
      2   0.0%    0.0%     0.0%                     63  Receive     0.000
      3   0.0%    0.0%     0.0%                     10  Receive     0.000
      4   0.0%    0.0%     0.1%                     11  Receive     1.848
      5   0.0%    0.0%     0.0%                     63  Receive     0.000
.
.
.
PID/TID   5Sec    1Min     5Min Process          Prio  STATE       CPU
8214      0.0%    0.0%     0.0% sysmgr.proc                         0.216
      1   0.0%    0.0%     0.0%                     10  Receive     0.132
      2   0.0%    0.0%     0.0%                     10  Sigwaitin   0.000
      3   0.0%    0.0%     0.0%                     10  Receive     0.004
      4   0.0%    0.0%     0.0%                     10  Receive     0.000
      5   0.0%    0.0%     0.0%                     10  Receive     0.000
      6   0.0%    0.0%     0.0%                     10  Receive     0.004
      7   0.0%    0.0%     0.0%                     10  Receive     0.000
      8   0.0%    0.0%     0.0%                     10  Receive     0.000
      9   0.0%    0.0%     0.0%                     10  Receive     0.000
     10   0.0%    0.0%     0.0%                     10  Receive     0.000
     11   0.0%    0.0%     0.0%                     10  Receive     0.000
     12   0.0%    0.0%     0.0%                     10  Receive     0.000
     13   0.0%    0.0%     0.0%                     10  Receive     0.028
     14   0.0%    0.0%     0.0%                     10  Receive     0.040
     15   0.0%    0.0%     0.0%                     10  Receive     0.000
     16   0.0%    0.0%     0.0%                     10  Receive     0.000
     17   0.0%    0.0%     0.0%                     10  Receive     0.004
     18   0.0%    0.0%     0.0%                     10  Receive     0.000
     19   0.0%    0.0%     0.0%                     10  Receive     0.000
     20   0.0%    0.0%     0.0%                     10  Receive     0.000
     21   0.0%    0.0%     0.0%                     10  Receive     0.004
     22   0.0%    0.0%     0.0%                     10  Receive     0.000
PID/TID   5Sec    1Min     5Min Process          Prio  STATE       CPU
8215      0.0%    0.0%     0.0% kosh.proc                           0.044
      1   0.0%    0.0%     0.0%                     10  Reply       0.044
PID/TID   5Sec    1Min     5Min Process          Prio  STATE       CPU
12290     0.0%    0.0%     0.0% chkptd.proc                         0.080
      1   0.0%    0.0%     0.0%                     10  Receive     0.080
      2   0.0%    0.0%     0.0%                     10  Receive     0.000
PID/TID   5Sec    1Min     5Min Process          Prio  STATE       CPU
12312     0.0%    0.0%     0.0% sysmgr.proc                         0.112
      1   0.0%    0.0%     0.0%                     10  Receive     0.112
      2   0.0%    0.0%     0.0%                     10  Sigwaitin   0.000
PID/TID   5Sec    1Min     5Min Process          Prio  STATE       CPU
12316     0.0%    0.0%     0.0% installer.proc                      0.072
      1   0.0%    0.0%     0.0%                     10  Receive     0.000
      3   0.0%    0.0%     0.0%                     10  Nanosleep   0.000
      4   0.0%    0.0%     0.0%                     10  Sigwaitin   0.000
      6   0.0%    0.0%     0.0%                     10  Receive     0.000
Process sbin/ios-base, type IOS, PID = 12317
CPU utilization for five seconds: 12%/9%; one minute: 13%; five minutes: 10%
Task  Runtime(ms)  Invoked   uSecs   5Sec    1Min    5Min TTY Task Name
    1       219      1503      145   0.00%   0.00%   0.00%   0 Hot Service Task
    2     23680     42384      558   2.39%   6.72%   4.81%   0 Service Task
    3      6104     11902      512   3.51%   1.99%   1.23%   0 Service Task
    4      1720      5761      298   1.91%   0.90%   0.39%   0 Service Task
```

```
 5          0          5          0    0.00%  0.00%  0.00%    0 Chunk Manager
 6          0          1          0    0.00%  0.00%  0.00%    0 Connection Mgr
 7          4        106         37    0.00%  0.00%  0.00%    0 Load Meter
 8       6240       7376        845    0.23%  0.15%  0.55%    0 Exec
 9        379         62       6112    0.00%  0.07%  0.04%    0 Check heaps
10          0          1          0    0.00%  0.00%  0.00%    0 Pool Manager
11          3          2       1500    0.00%  0.00%  0.00%    0 Timers
12          0          1          0    0.00%  0.00%  0.00%    0 AAA_SERVER_DEADT
13          0          2          0    0.00%  0.00%  0.00%    0 AAA high-capacit
14        307        517        593    0.00%  0.05%  0.03%    0 EnvMon
15          0          1          0    0.00%  0.00%  0.00%    0 OIR Handler
16        283         58       4879    0.00%  0.04%  0.02%    0 ARP Input
17          0          2          0    0.00%  0.00%  0.00%    0 Serial Backgroun
18          0         81          0    0.00%  0.00%  0.00%    0 ALARM_TRIGGER_SC
19          0          2          0    0.00%  0.00%  0.00%    0 DDR Timers
20          0          2          0    0.00%  0.00%  0.00%    0 Dialer event
21          4          2       2000    0.00%  0.00%  0.00%    0 Entity MIB API
22          0         54          0    0.00%  0.00%  0.00%    0 Compute SRP rate
23          0          9          0    0.00%  0.00%  0.00%    0 IPC Dynamic Cach
24          0          1          0    0.00%  0.00%  0.00%    0 IPC Zone Manager
25          0          1          0    0.00%  0.00%  0.00%    0 IPC Punt Process
26          4        513          7    0.00%  0.00%  0.00%    0 IPC Periodic Tim
27         11        513         21    0.00%  0.00%  0.00%    0 IPC Deferred Por
28          0          1          0    0.00%  0.00%  0.00%    0 IPC Seat Manager
29         83       1464         56    0.00%  0.00%  0.00%    0 EEM ED Syslog
.
.
.
```

Table 34 describes the significant fields shown in the display.

Table 34   *show processes cpu detailed (Software Modularity) Field Descriptions*

Field	Description
Total CPU utilization for five seconds	Total CPU utilization for the last 5 seconds. The second number indicates the percent of CPU time spent at the interrupt level.
1 minute	CPU utilization for the last minute.
5 minutes	CPU utilization for the last 5 minutes.
PID/TID	Process ID or task ID.
5Sec	Percentage of CPU time spent at the interrupt level for this process during the last five seconds.
1Min	Percentage of CPU time spent at the interrupt level for this process during the last minute.
5Min	Percentage of CPU time spent at the interrupt level for this process during the last five minutes.
Process	Process name.
Prio	Priority level of the process.
STATE	Current state of the process.
CPU	CPU utilization of the process in minutes and seconds.
type	Type of process; can be either IOS or POSIX.
Task	Task sequence number.

Table 34 *show processes cpu detailed (Software Modularity) Field Descriptions (continued)*

Field	Description
Runtime(ms)	CPU time that the process has used (in milliseconds).
Invoked	Number of times that the process has been invoked.
uSecs	Microseconds of CPU time for each process invocation.
5Sec	CPU utilization by task in the last 5 seconds.
1Min	CPU utilization by task in the last minute.
5Min	CPU utilization by task in the last 5 minutes.
TTY	Terminal that controls the process.
Task Name	Task name.

Related Commands

Command	Description
show processes	Displays information about active processes.
show processes memory	Displays the amount of system memory used per system process.

show processes detailed

To display detailed information about POSIX and Cisco IOS processes when Cisco IOS Software Modularity images are running, use the **show processes detailed** command in user EXEC or privileged EXEC mode.

> **show processes detailed** [*process-id* | *process-name*]

Syntax Description

process-id	(Optional) Process identifier.
process-name	(Optional) Process name.

Command Default

If no process ID or process name is specified, detailed information is displayed about all processes.

Command Modes

User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines

Use the **show processes detailed** command to gather detailed information about the number of tasks running, the process state, and other information about a process that is not displayed by the **show processes** command.

Examples

The following is sample output from the **show processes detailed** command for the process named sysmgr.proc:

```
Router# show processes detailed sysmgr.proc

              Job Id: 67
                 PID: 8210
     Executable name: sysmgr.proc
     Executable path: sbin/sysmgr.proc
         Instance ID: 1
                Args: -p
             Respawn: ON
       Respawn count: 1
Max. spawns per minute: 30
        Last started: Mon Aug18  17:08:53 2003
       Process state: Run
                core: SHAREDMEM MAINMEM
           Max. core: 0
               Level: 39
PID    TID   Stack pri state      Blked  HR:MM:SS:MSEC FLAGS     NAME
8210   1      52K  10 Receive     1      0:00:00:0071 00000000  sysmgr.proc
8210   2      52K  10 Sigwaitinfo        0:00:00:0000 00000000  sysmgr.proc
8210   3      52K  10 Receive     8      0:00:00:0003 00000000  sysmgr.proc
8210   4      52K  10 Reply       1      0:00:00:0003 00000000  sysmgr.proc
```

```
8210    5     52K  10 Receive      1       0:00:00:0000 00000000 sysmgr.proc
8210    6     52K  10 Receive      1       0:00:00:0015 00000000 sysmgr.proc
8210    7     52K  10 Receive      1       0:00:00:0000 00000000 sysmgr.proc
8210    8     52K  10 Receive      1       0:00:00:0000 00000000 sysmgr.proc
----------------------------------------------------------------
                    Job Id: 78
                       PID: 12308
          Executable name: sysmgr.proc
          Executable path: sbin/sysmgr.proc
              Instance ID: 2
                     Args: -p
                  Respawn: ON
            Respawn count: 1
    Max. spawns per minute: 30
             Last started: Mon Aug18  17:08:54 2003
            Process state: Run
                     core: SHAREDMEM MAINMEM
                Max. core: 0
                    Level: 40
PID    TID  Stack pri state       Blked  HR:MM:SS:MSEC FLAGS     NAME
12308  1     16K  10 Receive      1       0:00:00:0039 00000000 sysmgr.proc
12308  2     16K  10 Sigwaitinfo           0:00:00:0000 00000000 sysmgr.proc
----------------------------------------------------------------
```

Table 35 describes the significant fields shown in the display.

Table 35   show processes detailed Field Descriptions

Field	Description
Job Id	Job identifier.
PID	Process ID.
Executable name	Process name.
Executable path	Path and filename of the process.
Instance ID	Instance number.
Args	Arguments sent to the process at startup.
Respawn	Ability to respawn process: on or off.
Respawn count	Number of respawns of this process since boot where boot equals one.
Max. spawns per minute	Maximum number of respawns per minute for this process.
Last started	Date and time the process was last started.
Process state	Current state of process.
Core	Core dump options specified for the process.
Max. core	Maximum number of dumps allowed for this process.
Level	Internal number that determines the startup order for the process.
TID	Task ID.
Stack	Size, in kilobytes, of the memory stack.
pri	Process priority.
state	Current state of process.
Blked	Thread (with given process ID) that is currently blocked by the process.

Table 35   *show processes detailed Field Descriptions (continued)*

Field	Description
HR:MM:SS:MSEC	Time (in hours, minutes, seconds, and milliseconds) used by the process.
FLAGS	Process flags (bitmask).
NAME	Process name.

Related Commands

Command	Description
show processes	Displays information about active processes.

show processes kernel

To display information about System Manager kernel processes when Cisco IOS Software Modularity images are running, use the **show processes kernel** command in user EXEC or privileged EXEC mode.

show processes kernel {**family** | **files** | **signal** | **startup**}

Syntax Description

family	Displays process family information.
files	Displays file and channel use per process.
signal	Displays signal use for processes.
startup	Displays process data for processes that are created at startup.

Command Modes

User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Examples

The following is sample output from the **show processes kernel** command with the **family** keyword:

```
Router# show processes kernel family

    PID Name                        Session  Pgroup    PPID Sibling   Child
      1 kernel                            1       1       0       0      67
  12290 dumper.proc                       1      87      67      56       0
      3 devc-pty                          1      66      67       0       0
      4 devc-ser2681                      1      54      67      66       0
      6 pipe                              1      69      67       5       0
   8199 mqueue                            1      68      67      69       0
   8200 fsdev.proc                        1      57      67      68       0
   8201 flashfs_hes_slot1.proc            1      58      67      57       0
   8202 flashfs_hes_bootflash.proc        1      51      67      58       0
   8203 flashfs_hes_slot0.proc            1      50      67      51       0
   8204 dfs_disk1.proc                    1      61      67      50       0
   8205 dfs_disk0.proc                    1      60      67      61       0
   8206 ldcache.proc                      1      64      67      60       0
   8207 syslogd.proc                      1      65      67      64       0
   8208 name_svr.proc                     1      63      67      65       0
   8209 wdsysmon.proc                     1      52      67      63       0
   8210 sysmgr.proc                       1      67       1       0      74
   8211 kosh.proc                        56      56      67      52       0
  12308 sysmgr.proc                       1      78      67      87       0
  12309 chkptd.proc                       1      70      67      78       0
  12310 syslog_dev.proc                   1      81      67      70       0
  12311 fh_metric_dir.proc                1      82      67      81       0
```

Table 36 describes the significant fields shown in the display.

Table 36   show processes kernel family Field Descriptions

Field	Description
PID	Process ID.
Name	Process name.
Session	Session number.
Pgroup	Process group.
PPID	Parent process ID.
Sibling	Sibling process ID.
Child	Process ID of the parent process. This process is the child of the identified process.

The following is sample output from the **show processes kernel** command with the **files** keyword:

```
Router# show processes kernel files

PID       Open Files  Open Channels Name
1         2           42            kernel
12290     11          8             dumper.proc
3         3           68            devc-pty
4         3           43            devc-ser2681
6         4           4             pipe
8199      4           11            mqueue
8200      10          15            fsdev.proc
8201      8           4             flashfs_hes_slot1.proc
8202      8           4             flashfs_hes_bootflash.proc
8203      9           4             flashfs_hes_slot0.proc
8204      10          4             dfs_disk1.proc
8205      10          4             dfs_disk0.proc
8206      10          7             ldcache.proc
8207      12          11            syslogd.proc
8208      9           37            name_svr.proc
8209      10          42            wdsysmon.proc
8210      21          35            sysmgr.proc
8211      6           1             kosh.proc
12308     11          10            sysmgr.proc
12309     12          22            chkptd.proc
12310     11          8             syslog_dev.proc
12311     14          8             fh_metric_dir.proc
```

Table 37 describes the significant fields shown in the display.

Table 37   show processes kernel files Field Descriptions

Field	Description
PID	Process ID.
Open Files	Number of files opened by this process.
Open Channels	Number of channels opened by this process.
Name	Process name.

The following is sample output from the **show processes kernel** command with the **signal** keyword:

```
Router# show processes kernel signal

PID    Name              Signals Pending  Signals Ignored  Signals Queued
8199   mqueue            0000000000000000 0000000006800000 0000000000000000
   1                     0000000000000000 0000000000020000
PID    Name              Signals Pending  Signals Ignored  Signals Queued
8200   fsdev.proc        0000000000000000 0000000006800000 0000000000000000
   1                     0000000000000000 0000000000204003
   2                     0000000000000000 0000000000204003
   3                     0000000000000000 0000000000204003
   4                     0000000000000000 0000000000204003
   5                     0000000000000000 0000000000204003
   6                     0000000000000000 0000000000204003
   7                     0000000000000000 0000000000204003
```

Table 38 describes the significant fields shown in the display.

Table 38   show processes kernel signal Field Descriptions

Field	Description
PID	Process ID.
Name	Process name.
Signals Pending	Signals in a pending state (waiting to be unblocked from a POSIX process or process thread) shown in hexadecimal format. A signal is an asynchronous notification of an event. Each POSIX process thread has a signal mask. Signals can be directed to a process or to a process thread.
Signals Ignored	Signals that are blocked from a POSIX process or process thread, shown in hexadecimal format.
Signals Queued	Signals waiting for the scheduler to run the signal handler, shown in hexadecimal format.

The following is sample output from the **show processes kernel** command with the **startup** keyword:

```
Router# show processes kernel startup

PID    Last Started     State    RCnt  Name:Instance_Id Args
3      08/18/2003 17:08 Run      1     devc-pty:1 -n 32
4      08/18/2003 17:08 Run      1     devc-ser2681:1 -e -2 -b9600,9600
0x1e840404^3,0x5
0      Not configured   None     0     ldcache_preload.proc:1 preload
6      08/18/2003 17:08 Run      1     pipe:1
0      Not configured   None     0     clock_chip.proc:1 -r
0      Not configured   None     0     c7200-p-blob:1 -b
8199   08/18/2003 17:08 Run      1     mqueue:1
8200   08/18/2003 17:08 Run      1     fsdev.proc:1 /dev/slot0: /dev/slot1:
/dev/disk0: /dev/disk1: /dev/bootflash:
8201   08/18/2003 17:08 Run      1     flashfs_hes_slot1.proc:1 -m /slot1: -d
/dev/slot1:
8202   08/18/2003 17:08 Run      1     flashfs_hes_bootflash.proc:1 -m
/bootflash: -d /dev/bootflash:
8203   08/18/2003 17:08 Run      1     flashfs_hes_slot0.proc:1 -m /slot0: -d
/dev/slot0:
8204   08/18/2003 17:08 Run      1     dfs_disk1.proc:1 -m /disk1: -d
/dev/disk1:
8205   08/18/2003 17:08 Run      1     dfs_disk0.proc:1 -m /disk0: -d
/dev/disk0:
```

```
8206   08/18/2003 17:08 Run       1   ldcache.proc:1
8207   08/18/2003 17:08 Run       1   syslogd.proc:1
8208   08/18/2003 17:08 Run       1   name_svr.proc:1 /chan/reg_svr
8209   08/18/2003 17:08 Run       1   wdsysmon.proc:1
```

Table 39 describes the significant fields shown in the display.

Table 39   ***show processes kernel startup Field Descriptions***

Field	Description
PID	Process ID.
Last Started	Date and time when process was last started.
State	Current state of process.
RCnt	Number of times this process has restarted.
Name:Instance_Id	Process name and instance ID.
Args	Arguments passed to this process when it was spawned.

Related Commands

Command	Description
show processes	Displays information about active processes.

show processes memory

To display the amount of memory used by each system process in Cisco IOS, Cisco IOS XE, or Cisco IOS Software Modularity images, use the **show processes memory** command in privileged EXEC mode.

Cisco IOS Software

> show processes memory [*process-id* | **sorted** [**allocated** | **getbufs** | **holding**]]

Cisco IOS Software Modularity

> show processes memory [**detailed** [*process-name*[**:***instance-id*] | *process-id* [**taskid** *task-id*]]] [**alloc-summary** | **sorted** {**start** | **size** | **caller**}]

Cisco Catalyst 4500e Series Switches Running Cisco IOS XE Software

> show processes memory [**detailed** [**process iosd** | **task** *task-id*] | **sorted** [**allocated** | **getbufs** | **holding**]]

Syntax Description	Cisco IOS Software Syntax	
	process-id	(Optional) Process ID (PID) of a specific process. When you specify a process ID, only details for the specified process will be shown.
	sorted	(Optional) Displays memory data sorted by the Allocated,Getbufs,or Holding column. If the **sorted** keyword is used by itself, data is sorted by the Holding column by default.
	allocated	(Optional) Displays memory data sorted by the Allocated column.
	getbufs	(Optional) Displays memory data sorted by the Getbufs (Get Buffers) column.
	holding	(Optional) Displays memory data sorted by the Holding column. This keyword is the default.
	Cisco IOS Software Modularity Syntax	
	detailed	(Optional) Displays detailed information about iosproc processes.
	process-name	(Optional) Process name.
	:*instance-id*	(Optional) Instance name of either the Cisco IOS task or POSIX process. The colon is required.
	process-id	(Optional) Process identifier.
	taskid *task-id*	(Optional) Displays detailed memory usage of a specified Cisco IOS task within a process.
	alloc-summary	(Optional) Displays summary POSIX process memory usage per allocator.
	sorted	(Optional) Displays POSIX process memory usage sorted by start address, size, or the PC that called the process.
	start	(Optional) Displays POSIX process memory usage sorted by the start address of the process.
	size	(Optional) Displays POSIX process memory usage sorted by the size of the process.
	caller	(Optional) Displays POSIX process memory usage sorted by the PC that called the process.

Command Default   **Cisco IOS Software**

The memory used by all types of system processes is displayed.

Cisco IOS XE Software and Software Modularity

The system memory followed by a one-line summary of memory information about each IOS XE or Software Modularity process is displayed.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
10.0	This command was introduced.
12.0(23)S	The **sorted**, **allocated**, **getbufs**, and **holding** keywords were added.
12.2(13)	The **sorted**, **allocated**, **getbufs**, and **holding** keywords were added.
12.2(13)S	The **sorted**, **allocated**, **getbufs**, and **holding** keywords were added.
12.2(13)T	The **sorted**, **allocated**, **getbufs**, and **holding** keywords were added.
12.0(28)S	The output of the header line was updated to support the Memory Thresholding feature.
12.2(22)S	The output of the header line was updated to support the Memory Thresholding feature.
12.3(7)T	The output of the header line was updated to support the Memory Thresholding feature.
12.0(30)S	The summary information (first lines of output) for this command was separated from the rest of the output and labeled by memory pool type (Total Process Memory, Total I/O Memory, and so on). This enhancement also corrected a total process memory mismatch error (mismatch between the **show processes memory** command, the **show processes memory sorted** command, and the **show memory** command and its variants).
12.2(28)S	The summary information (first lines of output) for this command was separated from the rest of the output and labeled by memory pool type (Total Process Memory, Total I/O Memory, and so on). This enhancement also corrected a total process memory mismatch error (mismatch between the **show processes memory** command, the **show processes memory sorted** command, and the **show memory** command and its variants).
12.3(11)T	The summary information (first lines of output) for this command was separated from the rest of the output and labeled by memory pool type (Total Process Memory, Total I/O Memory, and so on). This enhancement also corrected a total process memory mismatch error (mismatch between the **show processes memory** command, the **show processes memory sorted** command, and the **show memory** command and its variants).
12.2(18)SXF4	The syntax was modified to support Cisco IOS Software Modularity images.

Release	Modification
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
Cisco IOS XE Release 3.1.0.SG	This command was introduced on the Cisco Catalyst 4500e series switches.
Cisco IOS XE Release 3.3S	This command was introduced on the Cisco ASR 1000 Series Aggregation Services Routers.

Usage Guidelines   The **show processes memory** command and the **show processes memory sorted** command displays a summary of total, used, and free memory, followed by a list of processes and their memory impact.

If the standard **show processes memory** *process-id* command is used, processes are sorted by their PID. If the **show processes memory sorted** command is used, the default sorting is by the Holding value.

Output Prior to Releases 12.3(7)T, 12.2(22)S, and 12.0(28)S

The first line (header line) of the **show processes memory** [**sorted**] command listed Total memory, Used memory, and Free memory values.

Output in Releases 12.3(7)T, 12.3(8)T, and 12.2(22)S Through 12.2(27)S2, 12.0(28)S, and 12.0(29)S

In Releases 12.3(7)T, 12.2(22)S, and 12.0(28)S, the Memory Thresholding feature was introduced. This feature affected the header line and the Holding column of the **show processes memory** command as described in this section.

The value for Total in the **show processes memory** command and the values listed in the Holding column showed the total (cumulative) value for the processor memory pools and the alternate memory pool* (typically, the I/O memory pool). However, the **show processes memory sorted** version of this command, and other commands, such as the **show memory summary** command, did not include the alternate memory pool in the totals; that is, these commands showed the total value for the Processor memory pool only. This caused an observed mismatch of memory totals between commands.

If you are using these releases, use the output of the **show memory summary** command to determine the individual amounts of Total and Free memory for the Processor memory pool and the I/O memory pool.

Output in Releases 12.3(11)T, 12.2(28)S, 12.0(30)S, and Later Releases

Beginning in Releases 12.3(11)T, 12.2(28)S, and 12.0(30)S, the summary information (first output lines) for the **show processes memory** command is separated by memory pool. For example, there are now individual lines for Total Process Memory, Total I/O Memory, and Total PCI Memory. In these releases or later releases, your Total Process Memory should match the total process memory shown for other commands, such as the **show memory summary** command.

About Alternate Memory Pools

An "alternate memory pool" is a memory pool that can be used as an alternative to allocate memory when the target (main) memory pool has been filled. For example, many platforms have a memory type called "Fast" that is limited to a small size (because the memory media used for Fast memory is expensive). You can prevent memory allocations from failing once the available Fast memory has been used up by configuring the normal Processor memory as an alternative memory pool for the Fast memory pool.

Cisco IOS XE Software and Software Modularity

Use the **show processes memory** command without any arguments and keywords to display the system memory followed by a one-line summary of memory information about each modular Cisco IOS process. Use the **detailed** keyword with this command to display detailed memory information about all processes. Other arguments and keywords are used to display Cisco IOS Software Modularity process memory information for a specified process name or process ID.

On Cisco IOS XE software images only, the **detailed** keyword will also show Cisco IOS task memory details.

Examples

Example output varies between Cisco IOS software releases. To see the appropriate output, choose one of the following sections:

- show processes memory Command for Cisco IOS Releases Prior to 12.3(7)T, 12.2(22)S, and 12.0(28)S
- show processes memory Command for Cisco IOS Releases Prior to 12.3(11)T, 12.2(28)S, and 12.0(30)S
- show processes memory Command for Cisco IOS Software Modularity
- Cisco Catalyst 4500e Series Switches Running Cisco IOS XE Software

show processes memory Command for Cisco IOS Releases Prior to 12.3(7)T, 12.2(22)S, and 12.0(28)S

The following is sample output from the **show processes memory** command:

```
Router# show processes memory

Processor Pool Total:   25954228 Used:    8368640 Free:   17585588

  PID TTY  Allocated      Freed    Holding    Getbufs   Retbufs Process
    0   0    8629528     689900    6751716          0         0 *Init*
    0   0      24048      12928      24048          0         0 *Sched*
    0   0        260        328         68     350080         0 *Dead*
    1   0          0          0      12928          0         0 Chunk Manager
    2   0        192        192       6928          0         0 Load Meter
    3   0     214664        304     227288          0         0 Exec
    4   0          0          0      12928          0         0 Check heaps
    5   0          0          0      12928          0         0 Pool Manager
    6   0        192        192      12928          0         0 Timers
    7   0        192        192      12928          0         0 Serial Backgroun
    8   0        192        192      12928          0         0 AAA high-capacit
    9   0          0          0      24928          0         0 Policy Manager
   10   0          0          0      12928          0         0 ARP Input
   11   0        192        192      12928          0         0 DDR Timers
   12   0          0          0      12928          0         0 Entity MIB API
   13   0          0          0      12928          0         0 MPLS HC Counter
   14   0          0          0      12928          0         0 SERIAL A'detect
    .
    .
    .
   78   0          0          0      12992          0         0 DHCPD Timer
   79   0        160          0      13088          0         0 DHCPD Database
                                  8329440 Total
```

Table 40 describes the significant fields shown in the display.

Table 40   *show processes memory Field Descriptions*

Field	Description
Processor Pool Total	Total amount of memory, in kilobytes (KB), held for the Processor memory pool.
Used	Total amount of used memory, in KB, in the Processor memory pool.
Free	Total amount of free memory, in KB, in the Processor memory pool.
PID	Process ID.
TTY	Terminal that controls the process.
Allocated	Bytes of memory allocated by the process.
Freed	Bytes of memory freed by the process, regardless of who originally allocated it.
Holding	Amount of memory, in KB, currently allocated to the process.
Getbufs	Number of times the process has requested a packet buffer.
Retbufs	Number of times the process has relinquished a packet buffer.
Process	Process name.
Init	System initialization process.
Sched	The scheduler process.
Dead	Processes as a group that are now dead.
8329440 Total	Total amount of memory, in KB, held by all processes (sum of the "Holding" column).

The following is sample output from the **show processes memory** command when the **sorted** keyword is used. In this case, the output is sorted by the Holding column, from largest to smallest.

```
Router# show processes memory sorted

Processor Pool Total:   25954228 Used:    8371280 Free:   17582948

  PID TTY  Allocated      Freed    Holding    Getbufs    Retbufs Process
    0   0    8629528     689900    6751716          0          0 *Init*
    3   0     217304        304     229928          0          0 Exec
   53   0     109248        192      96064          0          0 DHCPD Receive
   56   0          0          0      32928          0          0 COPS
   19   0      39048          0      25192          0          0 Net Background
   42   0          0          0      24960          0          0 L2X Data Daemon
   58   0        192        192      24928          0          0 X.25 Background
   43   0        192        192      24928          0          0 PPP IP Route
   49   0          0          0      24928          0          0 TCP Protocols
   48   0          0          0      24928          0          0 TCP Timer
   17   0        192        192      24928          0          0 XML Proxy Client
    9   0          0          0      24928          0          0 Policy Manager
   40   0          0          0      24928          0          0 L2X SSS manager
   29   0          0          0      24928          0          0 IP Input
   44   0        192        192      24928          0          0 PPP IPCP
   32   0        192        192      24928          0          0 PPP Hooks
   34   0          0          0      24928          0          0 SSS Manager
   41   0        192        192      24928          0          0 L2TP mgmt daemon
   16   0        192        192      24928          0          0 Dialer event
   35   0          0          0      24928          0          0 SSS Test Client
  --More--
```

The following is sample output from the **show processes memory** command when a process ID (*process-id*) is specified:

```
Router# show processes memory 1

Process ID: 1
Process Name: Chunk Manager
Total Memory Held: 8428 bytes

Processor memory holding = 8428 bytes
pc = 0x60790654, size =      6044, count =    1
pc = 0x607A5084, size =      1544, count =    1
pc = 0x6076DBC4, size =       652, count =    1
pc = 0x6076FF18, size =       188, count =    1

I/O memory holding = 0 bytes

Router# show processes memory 2

Process ID: 2
Process Name: Load Meter
Total Memory Held: 3884 bytes

Processor memory holding = 3884 bytes
pc = 0x60790654, size =      3044, count =    1
pc = 0x6076DBC4, size =       652, count =    1
pc = 0x6076FF18, size =       188, count =    1

I/O memory holding = 0 bytes
```

show processes memory Command for Cisco IOS Releases Prior to 12.3(11)T, 12.2(28)S, and 12.0(30)S

The following example shows the output of the **show processes memory** command before the changes to the summary information were made. Note that the Total in the **show processes summary** command output indicates total memory for all memory pools; in this example, the **show processes memory** total of 35423840 can be obtained by adding the Processor and I/O totals shown in the output of the **show memory summary** command. Note also that the **show processes memory sorted** command lists the Total Processor Memory (matches the **show memory summary** Processor Total), but the **show processes memory** command (without the **sorted** keyword) lists the total for all memory pools (Processor plus I/O memory).

```
Router# show version | include IOS

Cisco IOS Software, 3600 Software (C3660-BIN-M), Version 12.3(9)

Router# show memory summary

              Head     Total(b)     Used(b)      Free(b)     Lowest(b)    Largest(b)
Processor   61E379A0   27035232     8089056     18946176     17964108     17963664
     I/O     3800000    8388608     2815088      5573520      5561520      5573472

.
.
.

Router# show processes memory

Total: 35423840, Used: 10904192, Free: 24519648
 PID TTY   Allocated       Freed      Holding     Getbufs      Retbufs Process
   0   0    14548868     3004980      9946092           0            0 *Init*
   0   0       12732      567448        12732           0            0 *Sched*
.
.
```

```
        .
Router# show processes memory sorted

Total: 27035232, Used: 8089188, Free: 18946044
 PID TTY   Allocated       Freed    Holding      Getbufs     Retbufs Process
   0   0    14548868     3004980    9946092            0           0 *Init*
  64   0       76436        3084      74768            0           0 CEF process
        .
        .
        .

Router# show version | include IOS

Cisco IOS Software, 3600 Software (c3660-p-mz), Version 12.0(29)S,

Router# show memory summary

                 Head     Total(b)     Used(b)      Free(b)   Lowest(b)   Largest(b)
Processor    126CB10   49,331,668     6454676     42876992    42642208    42490796

Router# show processes memory

Total: 50,994,868, Used: 6220092, Free: 44774776
 PID TTY   Allocated       Freed    Holding      Getbufs     Retbufs Process
   0   0     6796228      627336    5325956            0           0 *Init*
   0   0         200       29792        200            0           0 *Sched*
   0   0         192         744          0       349000           0 *Dead*
   1   0           0           0      12896            0           0 Chunk Manager
        .
        .
        .

Router# show processes memory sorted

Total: 50,994,868, Used: 6222644, Free: 44772224
 PID TTY   Allocated       Freed    Holding      Getbufs     Retbufs Process
   0   0     6796228      627336    5325956            0           0 *Init*
  13   0       39056           0      25264            0           0 Net Background
  48   0           0           0      24896            0           0 L2X SSS manager
  18   0           0           0      24896            0           0 IP Input
        .
        .
        .
```

show processes memory Command for Cisco IOS Software Modularity

In a Cisco IOS Software Modularity image IOS, each process maintains its own heap memory, which is taken from the system memory in blocks. The process reuses this memory as required. If all the memory that was requested in a block is no longer in use, then the process can return the memory block to the system.

The following is sample output from the **show processes memory** command when a Cisco IOS Software Modularity image is running:

```
Router# show processes memory

System Memory : 262144K total, 113672K used, 148472K free

PID       Text     Data    Stack   Dynamic     Total Process
1            0        0       12         0        12 kernel
12290       52        8       28       196       284 dumper.proc
3           12        8        8       144       172 devc-pty
4          132        8        8        32       180 devc-ser2681
```

```
6           16      12      24      48      100 pipe
8199        12      12       8      48       80 mqueue
8200        16      24      48     452      540 fsdev.proc
8201        52      20       8      96      176 flashfs_hes_slot1.proc
8202        52      20       8      80      160 flashfs_hes_bootflash.proc
8203        52      20       8     128      208 flashfs_hes_slot0.proc
8204        20      68      12     164      264 dfs_disk1.proc
8205        20      68      12     164      264 dfs_disk0.proc
8206        36       4       8     144      192 ldcache.proc
8207        32       8      20     164      224 syslogd.proc
8208        24       4      28     464      520 name_svr.proc
8209       124     104      28     344      600 wdsysmon.proc
8210       100     144      52     328      624 sysmgr.proc
8211        12       4      28      64      108 kosh.proc
12308      100     144      16     144      404 sysmgr.proc
12309       24       4      12     112      152 chkptd.proc
12310       12       4       8      96      120 syslog_dev.proc
12311       44       4      24     248      320 fh_metric_dir.proc
12312       36       4      24     216      280 fh_fd_snmp.proc
12313       36       4      24     216      280 fh_fd_intf.proc
12314       32       4      24     216      276 fh_fd_timer.proc
12315       40       4      24     216      284 fh_fd_ioswd.proc
12316       28       4      24     200      256 fh_fd_counter.proc
12317       80      20      44     368      512 fh_server.proc
12326      140      40      28     280      488 tcp.proc
12327       48       4      24     256      332 udp.proc
12328        4       4      28    4660     4696 iprouting.iosproc
12329        4       4      36     600      644 cdp2.iosproc
```

Table 41 describes the significant fields shown in the display.

Table 41   *show processes memory (Software Modularity) Field Descriptions*

Field	Description
total	Total amount of memory, in KB, on the device.
used	Amount of memory, in KB, used in the system.
free	Amount of free memory, in KB, available in the system.
PID	Process ID.
Text	Amount of memory, in KB, used by the text segment of the specified process.
Data	Amount of memory, in KB, used by the data segment of the specified process.
Stack	Amount of memory, in KB, used by the stack segment of the specified process.
Dynamic	Amount of memory, in KB, used by the dynamic segment of the specified process.
Total	Total amount of memory, in KB, used by the specified process.
Process	Process name.

The following example shows the output of the **show processes memory detailed** command wherein the process (ios-base) holds sufficient memory to process a request of the Cisco IOS tasks without having to request more memory from the system. So although the amount of memory of the Cisco IOS tasks increased, the ios-base process does not consume more system memory.

```
Router# show processes memory detailed 16424 sorted holding

System Memory : 2097152K total, 1097777K used, 999375K free, 0K kernel reserved
Lowest(b)    : 1017212928
```

```
Process sbin/ios-base, type IOS, PID = 16424
    248904K total, 0K text, 0K data, 168K stack, 248736K dynamic
    Heap : 385874960 total, 261213896 used, 124661064 free
Task TTY  Allocated       Freed    Holding    Getbufs    Retbufs TaskName
    0   0  156853816       11168  156365472         0          0 *Init*
   38   0   65671128     3320184   62248368         0          0 PF_Init Process
  661   0   73106800    38231816   33093704         0          0 PIM Process
  487   0 2656186248  3806507384   33039576         0          0 cmfib
  652   0   56256064    19166160   27087872         0          0 MFIB_mrib_read
    4   0   91088216    68828800   13093720         0          0 Service Task
  629   0    2059320      132840    1927392         0          0 Const2 IPv6 Pro
   49   0 2155730560  2153990528    1741536         0    9579588 DiagCard1/-1
    0   0 2510481432  1396998880    1463056   2804860      23260 *Dead*
  444   0    7333952     5940064    1410992         0          0 FM core
  411   0   12865536     7934952    1396544         0          0 CMET MGR
  310   0  113849160   121164584    1284240         0          0 Exec
```

The following is sample output from the **show processes memory** command with details about the memory of process 12322 and the task with the ID of 1:

```
Router# show processes memory detailed 12322 taskid 1

System Memory : 262144K total, 113456K used, 148688K free

Process sbin/c7200-p-blob, type IOS, PID = 12322
    16568K total, 16K text, 8K data, 64K stack, 16480K dynamic

Memory Summary for TaskID = 1
Holding = 10248

        PC     Size  Count
0x7322FC74     9192      1
0x73236538      640      1
0x73231E8C      256      1
0x74175060      160      1
```

Table 42 describes the significant fields shown in the display that are different from .

Table 42   show processes memory detailed process-id taskid Field Descriptions

Field	Description
type	Type of process: POSIX or IOS.
Memory Summary for TaskID	Task ID.
Holding	Amount of memory, in bytes, currently held by the task.
PC	Caller PC of the task.
Size	Amount of memory, in bytes, used by this task.
Count	Number of times that task has been called.

The following is sample output from the **show processes memory** command with details about the memory of POSIX process ID 234567 with summary process memory usage per allocator:

```
Router# show processes memory detailed 234567 alloc-summary

System Memory : 262144K total, 113672K used, 148472K free

Process sbin/sysmgr.proc, type POSIX, PID = 12308
    404K total, 100K text, 144K data, 16K stack, 144K dynamic
```

```
      81920 heapsize, 68620 allocated, 8896 free

Allocated Blocks
Address     Usize       Size        Caller
0x0806C358 0x00000478 0x000004D0 0x721C7290
0x0806D1E0 0x00000128 0x00000130 0x72B90248
0x0806D318 0x00003678 0x000036E0 0x72B9820C
0x0806D700 0x000002A0 0x000002C0 0x72B8EB58
0x0806D770 0x00000058 0x00000060 0x72BA5488
0x0806D7D8 0x000000A0 0x000000B0 0x72B8D228
0x0806D8A8 0x00000200 0x00000208 0x721A728C
0x0806FF78 0x00000068 0x00000070 0x72BA78EC
0x08071438 0x0000005C 0x00000068 0x72B908A8
0x08071508 0x0000010E 0x00000120 0x72BA7AFC
0x08072840 0x000000A8 0x000000C0 0x7270A060
0x08072910 0x0000010C 0x00000118 0x7273A898
0x08072A30 0x000000E4 0x000000F0 0x72749074
0x08072B28 0x000000B0 0x000000B8 0x7276E87C
0x08072BE8 0x0000006C 0x00000078 0x727367A4
0x08072C68 0x000000B8 0x000000C0 0x7271E2A4
0x08072D30 0x000000D0 0x000000D8 0x7273834C
0x08072E10 0x00000250 0x00000258 0x72718A70
0x08073070 0x000002F4 0x00000300 0x72726484
0x08073378 0x000006A8 0x000006B0 0x73EA4DC4
0x08073A30 0x00000060 0x00000068 0x7352A9F8
0x08073B38 0x00000068 0x00000070 0x72B92008
0x08073BB0 0x00000058 0x00000060 0x72B9201C
0x08073EB8 0x00002FB4 0x000031C0 0x08026FEC
0x08074028 0x000020B8 0x000020C0 0x72709C9C
0x08077400 0x000000A0 0x000000A8 0x721DED94
0x08078028 0x000022B8 0x000022C0 0x727446B8
0x0807C028 0x00002320 0x00002328 0x72B907C4

Free Blocks
Address     Size
0x0806FFF0 0x00000010
0x080714A8 0x00000058
0x08073E18 0x00000098
0x08073FE8 0x00000018
0x08076FA0 0x00000328
0x080774B0 0x00000B50
0x0807FFB8 0x00000048
0x08080028 0x00003FD8
```

Table 43 describes the significant fields shown in the display.

Table 43   *show processes memory detailed alloc-summary Field Descriptions*

Field	Description
heapsize	Size of the process heap, in KB.
allocated	Amount of memory, in KB, allocated from the heap.
free	Amount of free memory, in KB, in the heap for the specified process.
Address	Block address, in hexadecimal.
Usize	Block size, in hexadecimal, without the trailer header.
Size	Block size, in hexadecimal.
Caller	Caller PC of the allocator of this block.

Cisco Catalyst 4500e Series Switches Running Cisco IOS XE Software

The following is sample output from the **show processes memory** command:

```
Switch# show processes memory

System memory  : 1943928K total, 733702K used, 1210221K free, 153224K kernel reserved
Lowest(b)      : 642265088
PID     Text      Data      Stack     Dynamic   RSS       Total     Process
1       252       480       84        444       1648      3648      init
2       0         0         0         0         0         0         kthreadd
3       0         0         0         0         0         0         migration/0
4       0         0         0         0         0         0         ksoftirqd/0
5       0         0         0         0         0         0         migration/1
6       0         0         0         0         0         0         ksoftirqd/1
7       0         0         0         0         0         0         events/0
8       0         0         0         0         0         0         events/1
9       0         0         0         0         0         0         khelper
61      0         0         0         0         0         0         kblockd/0
62      0         0         0         0         0         0         kblockd/1
75      0         0         0         0         0         0         khubd
78      0         0         0         0         0         0         kseriod
83      0         0         0         0         0         0         kmmcd
120     0         0         0         0         0         0         pdflush
121     0         0         0         0         0         0         pdflush
122     0         0         0         0         0         0         kswapd0
123     0         0         0         0         0         0         aio/0
124     0         0         0         0         0         0         aio/1
291     0         0         0         0         0         0         kpsmoused
309     0         0         0         0         0         0         rpciod/0
310     0         0         0         0         0         0         rpciod/1
354     92        180       84        136       456       2188      udevd
700     0         0         0         0         0         0         loop1
716     0         0         0         0         0         0         loop2
732     0         0         0         0         0         0         loop3
2203    424       164       84        132       1172      3180      dbus-daemon
2539    76        160       84        132       532       1788      portmap
2545    76        160       84        132       532       1788      portmap
2588    232       396       84        132       992       4596      sshd
2602    196       320       84        132       752       2964      xinetd
2606    196       320       84        132       748       2964      xinetd
3757    76        160       84        132       532       1788      vsi work/0
3758    76        160       84        132       532       1788      vsi work/1
--More--
```

The following is sample output from the **show processes memory detailed** command:

```
Switch# show processes memory detailed

System memory  : 1943928K total, 734271K used, 1209657K free, 153224K kernel reserved
Lowest(b)      : 642265088
PID     Text      Data      Stack     Dynamic   RSS       Total     Process
1       252       480       84        444       1648      3648      init
354     92        180       84        136       456       2188      udevd
2203    424       164       84        132       1172      3180      dbus-daemon
2539    76        160       84        132       532       1788      portmap
2545    76        160       84        132       532       1788      portmap
2588    232       396       84        132       992       4596      sshd
2602    196       320       84        132       752       2964      xinetd
2606    196       320       84        132       748       2964      xinetd
3757    76        160       84        132       532       1788      vsi work/0
3758    76        160       84        132       532       1788      vsi work/1
3891    848       148       84        88        1432      2984      check_gdb_statu
3895    72        160       84        132       580       1676      watchdog
4453    848       276       84        216       1512      3112      app_printf.sh
```

```
4465    848           272      84        212      1508      3108      app_printf.sh
4596    148           43972    84        528      5176      56664     slproc

TaskID  TTY     Allocated Freed       Holding  Getbufs   Retbufs   Task
1       0       327920    1544        367952   0         0         Chunk Manager
2       0       184       184         37032    0         0         Load Meter
3       0       0         0           40032    0         0         Deferred Events
4       0       17840     3888        40032    0         0         SpanTree Helper
5       0       0         0           40032    0         0         Retransmission of I
6       0       0         0           40032    0         0         IPC ISSU Receive Pr
7       0       0         0           40032    0         0         Check heaps
8       0       179248    173976      45304    144568    140316    Pool Manager
9       0       184       184         40032    0         0         Timers
10      0       184       184         40032    0         0         Serial Background
--More--
```

The following is sample output from the **show processes memory detailed** command specifying the Iosd process:

```
Switch# show processes memory detailed process iosd

Processor Pool Total:  805306368 Used:  225960152 Free:  579346216
      I/O Pool Total:   16777216 Used:     216376 Free:   16560840

  PID TTY   Allocated      Freed    Holding    Getbufs    Retbufs Process
    0   0   226577984    4410320  211589320          0          0 *Init*
    0   0           0    1591600          0          0          0 *Sched*
    0   0     2568488    1960496     676992    5368513     362940 *Dead*
    1   0      327920       1544     367952          0          0 Chunk Manager
    2   0         184        184      37032          0          0 Load Meter
    3   0           0          0      40032          0          0 Deferred Events
    4   0       17840       3888      40032          0          0 SpanTree Helper
    5   0           0          0      40032          0          0 Retransmission o
    6   0           0          0      40032          0          0 IPC ISSU Receive
    7   0           0          0      40032          0          0 Check heaps
    8   0      210880     205608      45304     170080     165828 Pool Manager
    9   0         184        184      40032          0          0 Timers
   10   0         184        184      40032          0          0 Serial Backgroun
--More--
```

The following is sample output from the **show processes memory sorted** command:

```
Switch#show proc memory sorted
System memory : 1943928K total, 734279K used, 1209649K free, 153224K kernel reserved
Lowest(b)     : 642265088
PID     Text        Data    Stack   Dynamic   RSS     Total     Process
10319   67716       798420  84      252       954524  1012856   iosd
4888    1132        200108  84      4076      26772   275408    ffm
4884    620         690480  84      5328      18564   728076    eicored
7635    144         181696  84      7464      16660   202620    cli_agent
9374    1048        298308  84      1128      11488   328992    licensed
10335   1676        257544  84      1252      11044   293848    licenseagentd
4852    208         208996  84      1848      10812   237632    ha_mgr
7566    168         249336  84      1408      8560    273668    installer
7585    268         167656  84      1616      8432    185556    snmp_subagent
4880    308         135080  84      968       8200    153944    os_info_p
4894    100         232936  84      1144      8072    252748    plogd
7410    68          233708  84      1172      7928    253840    dtmgr
10329   160         142384  84      832       7144    228360    cpumemd
4968    104         158828  84      1052      7080    178184    iifd
5047    88          165604  84      700       6196    181184    pdsd
4870    80          157452  84      728       6088    172244    sysmgr
4856    200         132816  84      688       5872    147940    oscore_p
```

```
--More--
```

Table 44 describes the significant fields shown in the display.

Table 44   show processes memory Field Descriptions

Field	Description
Processor Pool Total	Total amount of memory, in KB, held for the Processor memory pool.
I/O Pool Total	Total amount of memory, in KB, held for the I/O memory pool.
Used	Total amount of used memory, in KB, in the Processor/I/O memory pool.
Free	Total amount of free memory, in KB, in the Processor/I/O memory pool.
PID	Process ID.
TTY	Terminal that controls the process.
Allocated	Bytes of memory allocated by the process.
Freed	Bytes of memory freed by the process, regardless of who originally allocated it.
Holding	Amount of memory, in KB, currently allocated to the process.
Getbufs	Number of times the process has requested a packet buffer.
Retbufs	Number of times the process has relinquished a packet buffer.
Process	Process name.
Init	System initialization process.
Sched	The scheduler process.
Dead	Processes as a group that are now dead.
8329440 Total	Total amount of memory, in KB, held by all processes (sum of the "Holding" column).

Related Commands

Command	Description
show memory	Displays statistics about memory, including memory-free pool statistics.
show processes	Displays information about the active processes.

show raw statistics

To display raw IP statistics when Cisco IOS Software Modularity software is running, use the **show raw statistics** command in user EXEC or privileged EXEC mode.

show raw statistics

Syntax Description   This command has no arguments or keywords.

Command Modes   User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   There are three transport protocols used in Software Modularity: Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and raw IP. The transport protocol statistics are generally counters, though some are averages and time stamps. Use the **show raw statistics** command to display the raw IP statistics, and use the **clear raw statistics** command to reset the raw IP statistics. Many of the statistics are relevant to all of the transport protocols. To view the other transport protocol statistics used in Software Modularity, see the **show tcp statistics** and **show udp statistics** commands.

Examples   The following is sample output from the **show raw statistics** command:

```
Router# show raw statistics

Current packet level is 0 (Clear)
Rcvd: 0 packets, 0 bytes
      0 packets dropped in total (0 %)
      0 packets dropped due to invalid length
      0 packets dropped due to no protocol listener
      0 packets dropped due to receive packet limits
      0 packets dropped due to receive byte limits
      0 bytes dropped due to receive limits
Sent: 11 packets, 0 bytes
26 Open sockets
0 Packets used by socket I/O
0 Packets recovered after starvation
0 Packet memory warnings
0 Packet memory alarms
0 Packet allocation errors
0 Transmission pulse errors
0 Packet punts from IP
9 Packet punts to IP
9 Packet punts from application
0 Packet punts to application
1 packets delivered to IP at a time
1 packets received from application at a time
```

```
3 read notification pulses
0 millisecond delay between notification and read
```

Table 45 describes the significant fields shown in the display.

Table 45   show raw statistics Field Descriptions

Field	Description
Current packet level	A packet level of 0 (Clear) shows that less than 67 percent of the packet supply is in use. A packet level of 1 (Warn) shows that at least 67 percent of the packet supply is in use, and a packet level of 2 (Alarm) shows that at least 90 percent of the packet supply is in use.
Rcvd:	Statistics in this section refer to packets received by the router.
packets, bytes	Total number and size, in bytes, of raw IP packets received.
packets dropped in total	Total number of packets dropped, with percentage.
packets dropped due to invalid length	Number of packets dropped with an invalid length.
packets dropped due to no protocol listener	Number of packets dropped by raw IP because of no registered protocol. Each dropped packet generates an ICMP protocol unreachable message.
packets dropped due to no port	Number of packets dropped with no port.
packets dropped due to receive packet limits	Number of packets dropped after the receive packet limit is exceeded.
packets dropped due to receive byte limits	Number of packets dropped after the receive byte limit is exceeded.
bytes dropped due to receive limits	Number of bytes dropped after the receive byte limit is exceeded.
Sent:	Statistics in this section refer to packets sent by the router.
packets, bytes	Total number and size, in bytes, of raw IP packets sent.
Open sockets	Number of open sockets.
Packets used by socket I/O	Number of packets enqueued on socket send buffers, receive buffers, or reassembly queues. In summary, the number of packets currently being held by the transport protocol.
Packets recovered after starvation	Number of packets released by the transport protocol due to memory warnings or memory alarms.
Packet memory warnings	Number of packets with memory warnings.
Packet memory alarms	Number of packets with memory alarms.
Packet allocation errors	Number of packets with allocation errors.
Transmission pulse errors	Number of transmission signaling mechanism errors.
Packet punts from IP, Packet punts to IP	Number of batches of packets moved from and to the IP layer.
Packet punts from application, Packet punts to application	Number of batches of packets moved from and to the application layers.

Table 45   *show raw statistics Field Descriptions (continued)*

Field	Description
packets delivered from IP at a time	Number of packets sent to the IP layer at one time.
packets received from application at a time	Number of packets received from the application layer at one time.
read notification pulses	Number of times that the transport protocol notified applications about input data.
millisecond delay between notification and read	Number of packets with a time delay of more than one millisecond between the time of notification and the time the packet was read.

Related Commands

Command	Description
clear raw statistics	Clears raw IP statistics.
show tcp statistics	Displays TCP statistics.
show udp statistics	Displays UDP statistics.

show registry

To display the function registry information when Cisco IOS or Cisco IOS Software Modularity images are running, use the **show registry** command in user EXEC or privileged EXEC mode.

Cisco IOS Software

> **show registry** [*registry-name* [*registry-number*]] [**brief** | **statistics**]

Cisco IOS Software Modularity

> **show registry** [**name** [*registry-name* [*registry-number*]]] [**brief** [**name** [*registry-name* [*registry-number*]]] | **preemptions** | **rpcp status** | **statistics** [**brief**] [**name** [*registry-name* [*registry-number*]]] [**remote**]] [**process** {*process-name* | *process-id*}]

Syntax Description	Cisco IOS Software Syntax	
	registry-name	(Optional) Name of the registry to display.
	registry-number	(Optional) Number of the registry to display.
	brief	(Optional) Displays limited functions and services information.
	statistics	(Optional) Displays function registry statistics.
	Cisco IOS Software Modularity Syntax	
	name	(Optional) Displays information about a specific registry.
	registry-name	(Optional) Name of the registry to examine.
	registry-number	(Optional) Number of the registry to examine.
	brief	(Optional) Displays limited functions and services information.
	preemptions	(Optional) Displays registry preemptions information.
	rpcp status	(Optional) Displays status of remote procedure call (RPC) proxy.
	statistics	(Optional) Displays function registry statistics.
	remote	(Optional) Displays name server interactions and call statistics.
	process	(Optional) Displays process-specific information.
	process-name	(Optional) Process name.
	process-id	(Optional) Process ID. Number in range from 1 to 4294967295.

Command Default	If no options are specified, registry information is displayed for all registries.

Command Modes	User EXEC (>)
	Privileged EXEC (#)

Command History	Release	Modification
	11.1	This command was introduced.
	12.2(18)SXF4	Keywords and arguments were added to support Software Modularity images and this command was integrated into Cisco IOS Release 12.2(18)SXF4.
	12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

Example output varies between Cisco IOS software images and Cisco IOS Software Modularity software images. To view the appropriate output, choose one of the following sections:

- Cisco IOS Software
- Cisco IOS Software Modularity

Cisco IOS Software

The following is sample output from the **show registry** command using the **brief** keyword:

```
Router# show registry atm 3/0/0 brief

Registry objects: 1799  bytes: 213412

--
Registry 23: ATM Registry
  Service 23/0:
  Service 23/1:
  Service 23/2:
  Service 23/3:
  Service 23/4:
  Service 23/5:
  Service 23/6:
  Service 23/7:
  Service 23/8:
  Service 23/9:
  Service 23/10:
  Service 23/11:
  Service 23/12:
  Service 23/13:
  Service 23/14:
.
.
.
Registry 25: ATM routing Registry
  Service 25/0:
```

Table 46 describes the significant fields shown in the display.

Table 46   show registry brief (Cisco IOS) Field Descriptions

Field	Description
Registry objects	Number of objects in the registry.
bytes	Registry size, in bytes.
Registry	Displays the specified registry service number and type of registry service.

Cisco IOS Software Modularity

The following is partial sample output from the **show registry** command when running a software Modularity image:

```
Router# show registry

Registry information for ios-base:1:
=====================================================

---------------------------
AAA_ACCOUNTING :   11 services
               /   1 :  List    list[000]
               /   2 :  List    list[000]
               /   3 :  Case    size[020] list[000] default=0x7267C5D0  returnd
               /   4 :  Case    size[020] list[000] default=0x7267C5D0  returnd
                     16  0x72779400
               /   5 :  Case    size[020] list[000] default=0x7267C5D0  returnd
               /   6 :  Case    size[020] list[000] default=0x7267C5D0  returnd
                     16  0x7277915C
               /   7 :  Retval  size[020] list[000] default=0x7267C5E4  returno
               /   8 :  Retval  size[020] list[000] default=0x7267C5E4  returno
               /   9 :  Retval  size[020] list[000] default=0x7267C5E4  returno
               /  10 :  Stub    0x7267C5E4  return_zero
               /  11 :  Stub    0x76545BA0
AAA_ACCOUNTING :   11 services,   140 global bytes,   160 heap bytes
.
.
.
```

Table 47 describes the significant fields shown in the display.

Table 47   show registry (Software Modularity) Field Descriptions

Field	Description
Registry information	Displays the registry information by process name.
services	Number of services displayed.
global bytes	Number of bytes for the service,
heap bytes	Size of the service heap, in bytes,

show tcp

To display the status of Transmission Control Protocol (TCP) connections when Cisco IOS or Cisco IOS Software Modularity images re running, use the **show tcp** command in user EXEC or privileged EXEC mode.

> **show tcp** [*line-number*] [**tcb** *address*]

Syntax Description

line-number	(Optional) Absolute line number of the line for which you want to display Telnet connection status.
tcb	(Optional) Specifies the transmission control block (TCB) of the ECN-enabled connection that you want to display.
address	(Optional) TCB hexadecimal address. The valid range is from 0x0 to 0xFFFFFFFF.

Command Modes

User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
10.0	This command was introduced.
12.3(7)T	The **tcb** keyword and *address* argument were added.
12.4(2)T	The output is enhanced to display status and option flags.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB. The display output was modified to include the SSO capability flag and to indicate the reason that the SSO property failed on a TCP connection.
12.2(18)SXF4	This command was integrated into Cisco IOS Release 12.2(18)SXF4 to support Software Modularity images.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.

Examples

Example output varies between Cisco IOS software images and Cisco IOS Software Modularity software images. To view the appropriate output, choose one of the following sections:

- Cisco IOS Software
- Cisco IOS Software Modularity

Cisco IOS Software

The following is sample output that displays the status and option flags:

```
Router# show tcp
.
.
.
Status Flags: passive open, active open, retransmission timeout, app closed
```

```
Option Flags: vrf id set

IP Precedence value: 6
.
.
.
SRTT: 273 ms, RTTO: 490 ms, RTV: 217 ms, KRTT: 0 ms
minRTT: 0 ms, maxRTT: 300 ms, ACK hold: 200 ms
 Status Flags: active open, retransmission timeout
 Option Flags: vrf id set
 IP Precedence value: 6
```

Table 48 contains the types of flags, all possible command output enhancements, and descriptions. See Table 49 through Table 53 for descriptions of the other fields in the sample output.

Table 48   Type of Flags, All Possible Output Enhancements, and Descriptions

Type of Flag	Output Enhancement	Description
Status		
	Passive open	Set if passive open was done.
	Active open	Set if active open was done.
	Retransmission timeout	Set if retransmission timeout aborts.
	Net output pending	Output to network is pending.
	Wait for FIN	Wait for FIN to be acknowledged.
	App closed	Application has closed the TCB.
	Sync listen	Listen and establish a handshake.
	Gen tcbs	TCBs are generated as passive listener.
	Path mtu discovery	Path maximum transmission unit (MTU) discovery is enabled.
	Half closed	TCB is half closed.
	Timestamp echo present	Echo segment is present.
	Stopped reading	Read half is shut down.
Option		
	VRF id set	Set if connection has a VRF table identifier.
	Idle user	Set if the connection is idle.
	Sending urgent data	Set if urgent data is being sent.
	Keepalive running	Set if keepalive timer is running, or if an Explicit Congestion Notification (ECN)-enabled connection, or a TCB address bind is in effect.
	Nagle	Set if performing the Nagle algorithm.
	Always push	All packets and full-sized segments (internal use) are pushed.
	Path mtu capable	Path MTU discovery is configured.
	MD5	Message digest 5 (MD) messages are generated.
	Urgent data removed	Urgent data is removed.

Table 48   Type of Flags, All Possible Output Enhancements, and Descriptions (continued)

Type of Flag	Output Enhancement	Description
	SACK option permitted	Peer permits a selective acknowledgment (SACK) option.
	Timestamp option used	Time-stamp option is in use.
	Reuse local address	Local address can be reused.
	Non-blocking reads	Nonblocking TCP is read.
	Non-blocking writes	Nonblocking TCP is written.
	No delayed ACK	No TCP delayed acknowledgment is sent.
	Win-scale	Peer permits window scaling.
	Linger option set	The linger-on close option is set.

The following is sample output from the **show tcp** command:

```
Router# show tcp

tty0, connection 1 to host cider
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 172.31.232.17, Local port: 11184
Foreign host: 172.31.1.137, Foreign port: 23

Enqueued packets for retransmit: 0, input: 0, saved: 0

Event Timers (current time is 67341276):
Timer:          Retrans    TimeWait      AckHold     SendWnd   KeepAlive
Starts:             30           0           32           0           0
Wakeups:             1           0           14           0           0
Next:                0           0            0           0           0

iss:   67317172  snduna:   67317228  sndnxt:   67317228    sndwnd:   4096
irs: 1064896000  rcvnxt: 1064897597  rcvwnd:        2144  delrcvwnd:      0

SRTT: 317 ms, RTTO: 900 ms, RTV: 133 ms, KRTT: 0 ms
minRTT: 4 ms, maxRTT: 300 ms, ACK hold: 300 ms
Flags: higher precedence, idle user, retransmission timeout
Datagrams (max data segment is 536 bytes):
Rcvd: 41 (out of order: 0), with data: 34, total data bytes: 1596
Sent: 57 (retransmit: 1), with data: 35, total data bytes: 55
```

Table 49 describes the first five lines of output shown in the above display.

Table 49   show tcp Field Descriptions—First Section of Output

Field	Description
tty	Identifying number of the line.
connection	Identifying number of the TCP connection.
to host	Name of the remote host to which the connection has been made.

Table 49   show tcp Field Descriptions—First Section of Output (continued)

Field	Description
Connection state is	A connection progresses through a series of states during its lifetime. The states that follow are shown in the order in which a connection progresses through them.
	• LISTEN—Waiting for a connection request from any remote TCP and port.
	• SYNSENT—Waiting for a matching connection request after having sent a connection request.
	• SYNRCVD—Waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
	• ESTAB—Indicates an open connection; data received can be delivered to the user. This is the normal state for the data transfer phase of the connection.
	• FINWAIT1—Waiting for a connection termination request from the remote TCP or an acknowledgment of the connection termination request previously sent.
	• FINWAIT2—Waiting for a connection termination request from the remote TCP host.
	• CLOSEWAIT—Waiting for a connection termination request from the local user.
	• CLOSING—Waiting for a connection termination request acknowledgment from the remote TCP host.
	• LASTACK—Waiting for an acknowledgment of the connection termination request previously sent to the remote TCP host.
	• TIMEWAIT—Waiting for enough time to pass to be sure that the remote TCP host has received the acknowledgment of its connection termination request.
	• CLOSED—Indicates no connection state at all.
	• For more information about TCBs, see RFC 793, *Transmission Control Protocol Functional Specification.*
I/O status	Number that describes the current internal status of the connection.
unread input bytes	Number of bytes that the lower-level TCP processes have read but that the higher-level TCP processes have not yet processed.
Local host	IP address of the network server.
Local port	Local port number, as derived from the following equation: *line-number* + (512 * *random-number*). (The line number uses the lower nine bits; the other bits are random.)
Foreign host	IP address of the remote host to which the TCP connection has been made.
Foreign port	Destination port for the remote host.

Table 49 **show tcp Field Descriptions—First Section of Output (continued)**

Field	Description
Enqueued packets for retransmit	Number of packets that are waiting on the retransmit queue. These are packets on this TCP connection that have been sent but that have not yet been acknowledged by the remote TCP host.
input	Number of packets that are waiting on the input queue to be read by the user.
saved	Number of received out-of-order packets that are waiting for all packets in the datagram to be received before they enter the input queue. For example, if packets 1, 2, 4, 5, and 6 have been received, packets 1 and 2 would enter the input queue, and packets 4, 5, and 6 would enter the saved queue.

Note Use the **show tcp brief** command to display information about the ECN-enabled connections.

The following line of output shows the current elapsed time according to the system clock of the local host. The time shown is the number of milliseconds since the system started.

```
Event Timers (current time is 67341276):
```

The following lines of output display the number of times that various local TCP timeout values were reached during this connection. In this example, the local host re-sent data 30 times because it received no response from the remote host, and it sent an acknowledgment many more times because there was no data.

```
Timer:        Retrans    TimeWait    AckHold    SendWnd    Keepalive    GiveUp    PmtuAger
Starts:          30          0         32          0          0           0          0
Wakeups:          1          0         14          0          0           0          0
Next:             0          0          0          0          0           0          0
```

Table 50 describes the fields in the above lines of output.

Table 50 **show tcp Field Descriptions—Second Section of Output**

Field	Description
Timer	Names of the timer types in the output.
Starts	Number of times that the timer has been triggered during this connection.
Wakeups	Number of keepalives sent without receiving any response. (This field is reset to zero when a response is received.)
Next	System clock setting that triggers a timer for the next time an event (for example, TimeWait, AckHold, SendWnd, etc.) occurs.
Retrans	Retransmission timer is used to time TCP packets that have not been acknowledged and that are waiting for retransmission.
TimeWait	A time-wait timer ensures that the remote system receives a request to disconnect a session.
AckHold	An acknowledgment timer delays the sending of acknowledgments to the remote TCP in an attempt to reduce network use.

Table 50 ***show tcp Field Descriptions—Second Section of Output (continued)***

Field	Description
SendWnd	A send-window timer ensures that there is no closed window due to a lost TCP acknowledgment.
KeepAlive	A keepalive timer controls the transmission of test messages to the remote device to ensure that the link has not been broken without the knowledge of the local device.
GiveUp	A give-up timer determines the amount of time a local host will wait for an acknowledgment (or other appropriate reply) of a transmitted message after the the maximum number of retransmissions has been reached. If the timer expires, the local host gives up retransmission attempts and declares the connection dead.
PmtuAger	A path MTU (PMTU) age timer is an interval that displays how often TCP estimates the PMTU with a larger maximum segment size (MSS). When the age timer is used, TCP path MTU becomes a dynamic process. If the MSS is smaller than what the peer connection can manage, a larger MSS is tried every time the age timer expires. The discovery process stops when the send MSS is as large as the peer negotiated or the timer has been manually disabled by being set to infinite.

The following lines of output display the sequence numbers that TCP uses to ensure sequenced, reliable transport of data. The local host and remote host each use these sequence numbers for flow control and to acknowledge receipt of datagrams.

```
iss:   67317172  snduna:   67317228  sndnxt:   67317228    sndwnd:    4096
irs: 1064896000  rcvnxt: 1064897597  rcvwnd:       2144  delrcvwnd:      0
```

Table 51 describes the fields shown in the display above.

Table 51 ***show tcp Field Descriptions—Sequence Numbers***

Field	Description
iss	Initial send sequence number.
snduna	Last send sequence number that the local host sent but for which it has not received an acknowledgment.
sndnxt	Sequence number that the local host will send next.
sndwnd	TCP window size of the remote host.
irs	Initial receive sequence number.
rcvnxt	Last receive sequence number that the local host has acknowledged.
rcvwnd	TCP window size of the local host.
delrcvwnd	Delayed receive window—data that the local host has read from the connection but has not yet subtracted from the receive window that the host has advertised to the remote host. The value in this field gradually increases until it is larger than a full-sized packet, at which point it is applied to the rcvwnd field.

The following lines of output display values that the local host uses to keep track of transmission times so that TCP can adjust to the network that it is using.

```
SRTT: 317 ms, RTTO: 900 ms, RTV: 133 ms, KRTT: 0 ms
minRTT: 4 ms, maxRTT: 300 ms, ACK hold: 300 ms
Flags: higher precedence, idle user, retransmission timeout
```

Table 52 describes the significant fields shown in the output above.

Table 52   show tcp Field Descriptions—Line Beginning with "SRTT"

Field	Description
SRTT	A calculated smoothed round-trip timeout.
RTTO	Round-trip timeout.
RTV	Variance of the round-trip time.
KRTT	New round-trip timeout (using the Karn algorithm). This field separately tracks the round-trip time of packets that have been re-sent.
minRTT	Smallest recorded round-trip timeout (hard-wire value used for calculation).
maxRTT	Largest recorded round-trip timeout.
ACK hold	Time for which the local host will delay an acknowledgment in order to add data to it.
Flags	Properties of the connection.

Note   For more information on the above fields, see *Round Trip Time Estimation*, P. Karn and C. Partridge, ACM SIGCOMM-87, August 1987.

The following lines of output display the number of datagrams that are transported with data.

```
Datagrams (max data segment is 536 bytes):
Rcvd: 41 (out of order: 0), with data: 34, total data bytes: 1596
Sent: 57 (retransmit: 1), with data: 35, total data bytes: 55
```

Table 53 describes the significant fields shown in the last lines of the **show tcp** command output.

Table 53   show tcp Field Descriptions—Last Section of Output

Field	Description
Rcvd	Number of datagrams that the local host has received during this connection (and the number of these datagrams that were out of order).
with data	Number of these datagrams that contained data.
total data bytes	Total number of bytes of data in these datagrams.
Sent	Number of datagrams that the local host sent during this connection (and the number of these datagrams that needed to be re-sent).
with data	Number of these datagrams that contained data.
total data bytes	Total number of bytes of data in these datagrams.

The following is sample output from the **show tcp tcb** command that displays detailed information by hexadecimal address about an ECN-enabled connection:

```
Router# show tcp tcb 0x62CD2BB8

Connection state is LISTEN, I/O status: 1, unread input bytes: 0
Connection is ECN enabled
Local host: 10.10.10.1, Local port: 179
Foreign host: 10.10.10.2, Foreign port: 12000

Enqueued packets for retransmit: 0, input: 0 mis-ordered: 0 (0 bytes)

Event Timers (current time is 0x4F31940):
Timer          Starts    Wakeups          Next
Retrans             0         0           0x0
TimeWait            0         0           0x0
AckHold             0         0           0x0
SendWnd             0         0           0x0
KeepAlive           0         0           0x0
GiveUp              0         0           0x0
PmtuAger            0         0           0x0
DeadWait            0         0           0x0

iss:          0 snduna:         0 sndnxt:          0    sndwnd:       0
irs:          0 rcvnxt:         0 rcvwnd:       4128 delrcvwnd:       0

SRTT: 0 ms, RTTO: 2000 ms, RTV: 2000 ms, KRTT: 0 ms
minRTT: 60000 ms, maxRTT: 0 ms, ACK hold: 200 ms
Flags: passive open, higher precedence, retransmission timeout

TCB is waiting for TCP Process (67)

Datagrams (max data segment is 516 bytes):
Rcvd: 6 (out of order: 0), with data: 0, total data bytes: 0
Sent: 0 (retransmit: 0, fastretransmit: 0), with data: 0, total data
bytes: 0
```

Cisco IOS Software Modularity

The following is sample output from the **show tcp tcb** command from a Software Modularity image:

```
Router# show tcp tcb 0x1059C10

Connection state is ESTAB, I/O status: 0, unread input bytes: 0
Local host: 10.4.2.32, Local port: 23
Foreign host: 10.4.2.39, Foreign port: 11000
VRF table id is: 0

Current send queue size: 0 (max 65536)
Current receive queue size: 0 (max 32768)  mis-ordered: 0 bytes

Event Timers (current time is 0xB9ACB9):
Timer          Starts    Wakeups          Next(msec)
Retrans             6         0                0
SendWnd             0         0                0
TimeWait            0         0                0
AckHold             8         4                0
KeepAlive          11         0          7199992
PmtuAger            0         0                0
GiveUp              0         0                0
Throttle            0         0                0

irs:   1633857851  rcvnxt: 1633857890  rcvadv: 1633890620  rcvwnd:   32730
iss:   4231531315  snduna: 4231531392  sndnxt: 4231531392  sndwnd:    4052
```

```
sndmax: 4231531392  sndcwnd:     10220


SRTT: 84 ms,  RTTO: 650 ms,  RTV: 69 ms,  KRTT: 0 ms
minRTT: 0 ms,  maxRTT: 200 ms, ACK hold: 200 ms

Keepalive time: 7200 sec, SYN wait time: 75 sec
Giveup time: 0 ms, Retransmission retries: 0, Retransmit forever: FALSE


State flags: none


Feature flags: Nagle


Request flags: none
Window scales: rcv 0, snd 0, request rcv 0, request snd 0
Timestamp option: recent 0, recent age 0, last ACK sent           0

Datagrams (in bytes): MSS 1460, peer MSS 1460, min MSS 1460, max MSS 1460
Rcvd: 14 (out of order: 0), with data: 10, total data bytes: 38
Sent: 10 (retransmit: 0, fastretransmit: 0), with data: 5, total data bytes: 76


Header prediction hit rate: 72 %


Socket states: SS_ISCONNECTED, SS_PRIV


Read buffer flags: SB_WAIT, SB_SEL, SB_DEL_WAKEUP
Read notifications: 4


Write buffer flags: SB_DEL_WAKEUP
Write notifications: 0
Socket status: 0
```

Related Commands	Command	Description
	show tcp brief	Displays a concise description of TCP connection endpoints.

show tcp statistics

To display TCP statistics, use the **show tcp statistics** command in user EXEC or privileged EXEC mode.

show tcp statistics

Syntax Description

This command has no arguments or keywords.

Command Modes

User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
11.3	This command was introduced.
12.2(18)SXF4	This command was integrated into Cisco IOS Release 12.2(18)SXF4, and the output was modified to display Software Modularity information.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Cisco IOS Software Modularity

There are three transport protocols used in Software Modularity: TCP, UDP, and raw IP. The transport protocol statistics are generally counters, though some are averages and time stamps. Use the **show tcp statistics** command to display the TCP statistics and use the **clear tcp statistics** command to reset the TCP statistics. Many of the statistics are relevant to all of the transport protocols. To view the other transport protocol statistics used in Software Modularity, see the **show raw statistics** and **show udp statistics** commands.

Examples

Example output varies between Cisco IOS software images and Cisco IOS Software Modularity software images. To view the appropriate output, choose one of the following sections:

- Cisco IOS Software

- Cisco IOS Software Modularity

Cisco IOS Software

The following is sample output from the **show tcp statistics** command:

```
Router# show tcp statistics

Rcvd: 210 Total, 0 no port
      0 checksum error, 0 bad offset, 0 too short
      132 packets (26640 bytes) in sequence
      5 dup packets (502 bytes)
      0 partially dup packets (0 bytes)
      0 out-of-order packets (0 bytes)
      0 packets (0 bytes) with data after window
      0 packets after close
      0 window probe packets, 0 window update packets
      0 dup ack packets, 0 ack packets with unsend data
```

```
       69 ack packets (3044 bytes)
Sent: 175 Total, 0 urgent packets
       16 control packets (including 1 retransmitted)
       69 data packets (3029 bytes)
       0 data packets (0 bytes) retransmitted
       73 ack only packets (49 delayed)
       0 window probe packets, 17 window update packets
7 Connections initiated, 1 connections accepted, 8 connections established
8 Connections closed (including 0 dropped, 0 embryonic dropped)
1 Total rxmt timeout, 0 connections dropped in rxmt timeout
0 Keepalive timeout, 0 keepalive probe, 0 Connections dropped in keepalive
```

Table 54 describes the significant fields shown in the display.

Table 54　*show tcp statistics Field Descriptions*

Field	Description
Rcvd:	Statistics in this section refer to packets received by the router.
Total	Total number of TCP packets received.
no port	Number of packets received with no port.
checksum error	Number of packets received with checksum error.
bad offset	Number of packets received with bad offset to data.
too short	Number of packets received that were too short.
packets in sequence	Number of data packets received in sequence.
dup packets	Number of duplicate packets received.
partially dup packets	Number of packets received with partially duplicated data.
out-of-order packets	Number of packets received out of order.
packets with data after window	Number of packets received with data that exceeded the window size of the receiver.
packets after close	Number of packets received after the connection was closed.
window probe packets	Number of window probe packets received.
window update packets	Number of window update packets received.
dup ack packets	Number of duplicate acknowledgment packets received.
ack packets with unsend data	Number of acknowledgment packets received with unsent data.
ack packets	Number of acknowledgment packets received.
Sent:	Statistics in this section refer to packets sent by the router.
Total	Total number of TCP packets sent.
urgent packets	Number of urgent packets sent.
control packets	Number of control packets (SYN, FIN, or RST) sent.
data packets	Number of data packets sent.
data packets retransmitted	Number of data packets re-sent.
ack only packets	Number of packets sent that are acknowledgments only.
window probe packets	Number of window probe packets sent.
window update packets	Number of window update packets sent.
Connections initiated	Number of connections initiated.

Table 54   show tcp statistics Field Descriptions (continued)

Field	Description
connections accepted	Number of connections accepted.
connections established	Number of connections established.
Connections closed	Number of connections closed.
Total rxmt timeout	Number of times that the router tried to resend, but timed out.
connections dropped in rxmit timeout	Number of connections dropped in the resend timeout.
Keepalive timeout	Number of keepalive packets in the timeout.
keepalive probe	Number of keepalive probes.
Connections dropped in keepalive	Number of connections dropped in the keepalive.

Cisco IOS Software Modularity

The following is sample output from the **show tcp statistics** command when a Software Modularity image is running under Cisco IOS Release 12.2(18)SXF4:

```
Router# show tcp statistics

Current packet level is 0 (Clear)
Rcvd: 0 Total, 0 no port
      0 checksum error, 0 bad offset, 0 too short
      0 packets (0 bytes) in sequence
      0 dup packets (0 bytes)
      0 partially dup packets (0 bytes)
      0 out-of-order packets (0 bytes)
      0 packets (0 bytes) with data after window
      0 packets after close
      0 window probe packets, 0 window update packets
      0 dup ack packets, 0 ack packets for unsent data
      0 ack packets (0 bytes)
      0 packets dropped due to PAWS
      0 packets dropped due to receive packet limits
      0 packets dropped due to receive byte limits
Sent: 0 Total, 0 urgent packets
      0 control packets (including 0 retransmitted)
      0 data packets (0 bytes)
      0 data packets (0 bytes) retransmitted
      0 data packets (0 bytes) fastretransmitted
      0 Sack retransmitted bytes, 0 Sack skipped bytes
      0 ack only packets (0 delayed)
      0 window probe packets, 0 window update packets
0 Connections initiated, 0 connections accepted, 0 connections established
0 Connections closed (including 0 dropped, 0 embryonic dropped)
0 Total rxmt timeout, 0 connections dropped in rxmt timeout
0 RTO, 0 KRTO (milliseconds)
0 VJ SRTT, 0 variance (milliseconds)
0 min RTT, 0 max RTT (milliseconds)
0 Keepalive timeout, 0 keepalive probe, 0 Connections dropped in keepalive
0 increase MSS, 0 decrease MSS
15 Open sockets
0 Timer interrupts
0 Packets used by socket I/O
0 Packets used by TCP reassembly
0 Packets recovered after starvation
```

```
0 Packet memory warnings
0 Packet memory alarms
0 Packet allocation errors
0 Packet to octet switches due to send flow control
0 Packet to octet switches due to partial ACKs
0 Packet to octet switches due to inadequate resources
0 Output function calls
0 Truncated write I/O vectors
0 Transmission pulse errors
0 Packet punts from IP 0 Packet punts to IP
0 Packet punts from application
0 Packet punts to application
```

Table 55 describes the significant fields shown in the display that are different from Table 45 on page 110.

Table 55   *show tcp statistics (Software Modularity) Field Descriptions*

Field	Description
Current packet level	A packet level of 0 (Clear) shows that less than 67 percent of the packet supply is in use. A packet level of 1 (Warn) shows that at least 67 percent of the packet supply is in use, and a packet level of 2 (Alarm) shows that at least 90 percent of the packet supply is in use.
packets dropped due to PAWS	Number of packets dropped because of sequence number wrap-around on high speed, low latency networks.
packets dropped due to receive packet limits	Number of packets dropped after the receive packet limit is exceeded.
packets dropped due to receive byte limits	Number of packets dropped after the receive byte limit is exceeded.
data packets fastretransmitted	Number of packets retransmitted before timer expiry because of excessive duplicate ACKs.
Sack retransmitted bytes, Sack skipped bytes	Number of retransmitted bytes due to selective acknowledgement.
RTO, KRTO	RTO is the current retransmission timeout, as calculated by Van Jacobson's algorithm. KRTO is the exponentially backed off retransmission timeout.
VJ SRTT, variance	Scaled mean and variance round trip times used by Van Jacobson's algorithm.
min RTT, max RTT	Minimum and maximum round-trip time (RTT), in milliseconds.
increase MSS, decrease MSS	Number of times that the maximum segment size (MSS) changed because of path MTU discovery.
Open sockets	Number of open sockets.
Timer interrupts	Number of packets received with timer interrupts.
Packets used by socket I/O	Number of packets enqueued on socket send buffers, receive buffers, or reassembly queues. In summary, the number of packets currently being held by the transport protocol.
Packets used by TCP reassembly	Number of out of order segments that cannot be passed to application because of missing holes in the data stream. These holes will be filled when the peer retransmits.

Table 55 *show tcp statistics (Software Modularity) Field Descriptions (continued)*

Field	Description
Packets recovered after starvation	Number of packets released by the transport protocol due to memory warnings or memory alarms.
Packet memory warnings	Number of packets with memory warnings.
Packet memory alarms	Number of packets with memory alarms.
Packet allocation errors	Number of packets with allocation errors.
Packet to octet switches due to send flow control	Number of times that TCP switched from packet I/O to octet buffer I/O because of inadequate send window.
Packet to octet switches due to partial ACKs	Number of times that TCP switched from packet I/O to octet buffer I/O because of partially acknowledged data.
Packet to octet switches due to inadequate resources	Number of times that TCP switched from packet I/O to octet buffer I/O because of inadequate packet resources.
Output function calls	Number of times that the TCP output engine was invoked.
Truncated write I/O vectors	Number of truncated segments due to inadequate write buffers.
Transmission pulse errors	Number of transmission signaling mechanism errors.
Packet punts from IP, Packet punts to IP	Number of batches of packets moved from and to the IP layer.
Packet punts from application, Packet punts to application	Number of batches of packets moved from and to the application layers.

Related Commands

Command	Description
clear tcp statistics	Clears TCP statistics.
show raw statistics	Displays raw IP transport protocol statistics.
show udp statistics	Displays UDP transport protocol statistics.

show udp statistics

To display User Datagram Protocol (UDP) statistics when Cisco IOS Software Modularity software is running, use the **show udp statistics** command in user EXEC or privileged EXEC mode.

> **show udp statistics**

Syntax Description   This command has no arguments or keywords.

Command Modes   User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   There are three transport protocols used in Software Modularity: Transmission Control Protocol (TCP), UDP, and raw IP. The transport protocol statistics are generally counters, though some are averages and time stamps. Use the **show udp statistics** command to display the UDP statistics, and use the **clear udp statistics** command to reset the UDP statistics. Many of the statistics are relevant to all of the transport protocols. To view the other transport protocol statistics used in Software Modularity, see the **show raw statistics** and **show tcp statistics** commands

Examples   The following is sample output from the **show udp statistics** command:

```
Router# show udp statistics

Current packet level is 0 (Clear)
Rcvd: 3291 packets, 0 bytes
      3291 packets dropped in total (100 %)
      0 packets dropped due to invalid length
      0 packets dropped due to invalid checksum
      3291 packets dropped due to no port
      0 packets dropped due to receive packet limits
      0 packets dropped due to receive byte limits
      0 bytes dropped due to receive limits
Sent: 0 packets, 0 bytes
5 Open sockets
0 Packets used by socket I/O
0 Packets recovered after starvation
0 Packet memory warnings
0 Packet memory alarms
0 Packet allocation errors
0 Transmission pulse errors
3291 Packet punts from IP
0 Packet punts to IP
0 Packet punts from application
0 Packet punts to application
1 packets received from IP at a time
```

Table 56 describes the significant fields shown in the display.

Table 56 **show udp statistics Field Descriptions**

Field	Description
Current packet level	A packet level of 0 (Clear) shows that less than 67 percent of the packet supply is in use. A packet level of 1 (Warn) shows that at least 67 percent of the packet supply is in use, and a packet level of 2 (Alarm) shows that at least 90 percent of the packet supply is in use.
Rcvd:	Statistics in this section refer to packets received by the router.
packets, bytes	Total number and size, in bytes, of UDP packets received.
packets dropped in total	Total number of packets dropped, with percentage.
packets dropped due to invalid length	Number of packets dropped with an invalid length.
packets dropped due to invalid checksum	Number of packets dropped with an invalid checksum.
packets dropped due to no port	Number of packets dropped with no port.
packets dropped due to receive packet limits	Number of packets dropped after the receive packet limit is exceeded.
packets dropped due to receive byte limits	Number of packets dropped after the receive byte limit is exceeded.
bytes dropped due to receive limits	Number of bytes dropped after the receive byte limit is exceeded.
Sent:	Statistics in this section refer to packets sent by the router.
packets, bytes	Total number and size, in bytes, of UDP packets sent.
Open sockets	Number of open sockets.
Packets used by socket I/O	Number of packets enqueued on socket send buffers, receive buffers, or reassembly queues. In summary, the number of packets currently being held by the transport protocol.
Packets recovered after starvation	Number of packets released by the transport protocol due to memory warnings or memory alarms.
Packet memory warnings	Number of packets with memory warnings.
Packet memory alarms	Number of packets with memory alarms.
Packet allocation errors	Number of packets with allocation errors.
Transmission pulse errors	Number of transmission signaling mechanism errors.
Packet punts from IP, Packet punts to IP	Number of batches of packets moved from and to the IP layer.
Packet punts from application Packet punts to application	Number of batches of packets moved from and to the application layers.
packets received from IP at a time	Number of packets received from the IP layer at one time.

Related Commands

Command	Description
clear udp statistics	Clears UDP statistics.
show raw statistics	Displays raw IP statistics.
show tcp statistics	Displays TCP statistics.

write checkpoint

To run the configuration checkpoint process when a Cisco IOS Software Modularity image is running, use the **write checkpoint** command in privileged EXEC mode.

write checkpoint

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines

If you have a large configuration file, the default implicit configuration checkpoint process may take some time to complete and prevent you from entering other command-line interface (CLI) commands to save or display the configuration. To disable the checkpoint process, enter the **no** form of the **service checkpoint-config** command. When you are ready to run the configuration checkpoint process, use the **write checkpoint** command to run the configuration checkpoint process.

Implicit configuration checkpointing means that configuration checkpointing occurs for all processes. A Software Modularity process can be restarted under an error condition or after upgrading. When the process is restarted and operational, the state of the process returns to the state the process was in prior to the restart. The software checkpoints the configuration information and when the process restarts, the configuration information is read from the checkpoint.

Configuration checkpoint information is implicitly generated as follows:

• Each time you exit from global configuration mode.

• Each time you enter the **write memory**, **copy running-config**, or **show run** command.

• When the action generated by the **write checkpoint** command has completed. The **write checkpoint** command is visible only after you enter the **no service checkpoint-config** command.

Examples

In the following example, the **no** form of the **service checkpoint-config** command is entered to disable the configuration checkpoint process, configuration commands are entered, and after exiting from the configuration mode the **write checkpoint** command is entered to run the configuration checkpoint process.

```
configure terminal
 no service checkpoint-config
!
! configuration commands are entered here
 end

write checkpoint
```

Related Commands

Command	Description
service checkpoint-config	Enables implicit configuration checkpointing when running a Cisco IOS Software Modularity image.

write core (Software Modularity)

To generate a core dump for a Cisco IOS Software Modularity process if the process crashes, use the **write core** command in privileged EXEC mode.

> **write core** *process-name* [**suspend**]

Syntax Description

process-name	Process name.
suspend	(Optional) Suspends the process while the core dump is performed.

Command Default   No core dumps are performed if a process crashes.

Command Modes   Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXF4	This command was introduced to support Software Modularity images.

Usage Guidelines   Use the **write core** (Software Modularity) command to dump the core of the process when the process crashes. The output generated in the dump can be used with the information generated by the **exception crashinfo file** command to verify the functionality of dumping the process core when the process crashes. Each Cisco IOS Software Modularity component has an associated .startup file that determines the core dump options (and other attributes) of that process. Use the **show processes detailed** command to display the core dump options for a process. Use the **exception core** command to override the default values set in the .startup file for the specific software component.

Examples   In the following example, a core dump is generated for the Cisco Discovery Protocol (CDP) process.

```
write core cdp.proc
```

In the following example, a core dump is generated for the CDP process and the CDP process is suspended while the core dump is performed.

```
write core cdp.proc suspend
```

Related Commands

Command	Description
exception core	Sets or changes the core dump options for a process.
exception crashinfo file	Enables the creation of a diagnostic file at the time of unexpected system shutdowns.
show processes detailed	Displays detailed process information.