Process and Memory Management Commands

This chapter describes the Cisco IOS XR software commands used to manage processes and memory.

For more information about using the process and memory management commands to perform troubleshooting tasks, see .

clear context

To clear core dump context information, use the **clear context** command in the appropriate mode.

clear context location {*node-id* | **all**}

Syntax Description

location {*node-id* | **all**}	(Optional) Clears core dump context information for a specified node. The *node-id* argument is expressed in the *rack/slot* notation. Use the **all** keyword to indicate all nodes.

Command Default

No default behavior or values

Command Modes

XR EXEC mode

Command History

Release	Modification
Release 5.0.0	This command was introduced.
Release 3.9.0	No modification.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Use the **clear context** command to clear core dump context information. If you do not specify a node with the **location** *node-id* keyword and argument, this command clears core dump context information for all nodes.

Use the **show context** command to display core dump context information.

Task ID

Task ID	Operations
diag	execute

The following example shows how to clear core dump context information:

```
RP/0/RP0/CPU0:router# clear context
```

Related Topics

show context, on page 27

dumpcore

To manually generate a core dump, use the **dumpcore** command in XR EXEC mode orSystem Admin EXEC mode.

dumpcore {**running** | **suspended**} *job-id* **location** *node-id*

Syntax Description

running	Generates a core dump for a running process.
suspended	Suspends a process, generates a core dump for the process, and resumes the process.
job-id	Process instance identifier.
location *node-id*	Generates a core dump for a process running on the specified node. The *node-id* argument is expressed in the *rack/slot* notation.

Command Default

No default behavior or values

Command Modes

System Admin EXEC mode

XR EXEC mode

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

When a process crashes on the Cisco IOS XR software, a core dump file of the event is written to a designated destination without bringing down the router. Upon receiving notification that a process has terminated abnormally, the Cisco IOS XR software then respawns the crashed process. Core dump files are used by Cisco Technical Support Center engineers and development engineers to debug the Cisco IOS XR software.

Core dumps can be generated manually for a process, even when a process has not crashed. Two modes exist to generate a core dump manually:

- **running** —Generates a core dump for a running process. This mode can be used to generate a core dump on a critical process (a process whose suspension could have a negative impact on the performance of the router) because the core dump file is generated independently, that is, the process continues to run as the core dump file is being generated.
- **suspended** —Suspends a process, generates a core dump for the process, and resumes the process. Whenever the process is suspended, this mode ensures data consistency in the core dump file.

Core dump files contain the following information about a crashed process:

- Register information
- Thread status information
- Process status information
- Selected memory segments

The following scenarios are applicable for creating full or sparse core dumps:

- Without the **exception sparse** configuration or exception sparse OFF, and default core size (4095 MB), a full core is created till the core size. Beyond this, only stack trace is collected.
- With non-default core size and without the **exception sparse** configuration, or exception sparse OFF , a full core is created until the core size limit is reached. Beyond the core size limit, only the stack trace is collected.
- With the exception sparse ON and default core size (4095 MB), a full core is created until the sparse size limit is reached, and a sparse core is created thereafter till the core size. Beyond this, only stack trace is collected.
- With non-default core size and with the exception sparse ON, a full core is created until the sparse size limit is reached. Beyond the sparse size limit, only the stack trace is collected.

Note By default, full core dumps are created irrespective of the **exception sparse** configuration. If there is not enough free shared memory available, then the core dump process fails.

Task ID

Task ID	Operations
diag	read, write

The following example shows how to generate a core dump in suspended mode for the process instance 52:

```
RP/0/RP0/CPU0:router# dumpcore suspended 52

RP/0/RP0/CPU0:Sep 22 01:40:26.982 : sysmgr[71]: process in stop/continue state 4104
RP/0/RP0/CPU0Sep 22 01:40:26.989 : dumper[54]: %DUMPER-4-CORE_INFO : Core for pid = 4104
 (pkg/bin/devc-conaux) requested by pkg/bin/dumper_gen@node0_RP0_CPU0
RP/0/RP0/CPU0Sep 22 01:40:26.993 : dumper[54]: %DUMPER-6-SPARSE_CORE_DUMP :
 Sparse core dump as configured dump sparse for all
RP/0/RP0/CPU0Sep 22 01:40:26.995 : dumper[54]: %DUMPER-7-DLL_INFO_HEAD : DLL path
 Text addr.  Text size   Data addr.  Data size   Version
RP/0/RP0/CPU0Sep 22 01:40:26.996 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libplatform.dll  0xfc0d5000  0x0000a914  0xfc0e0000  0x00002000        0
RP/0/RP0/CPU0Sep 22 01:40:26.996 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libsysmgr.dll  0xfc0e2000  0x0000ab48  0xfc0c295c  0x00000368        0
RP/0/RP0/CPU0Sep 22 01:40:26.997 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libinfra.dll  0xfc0ed000  0x00032de0  0xfc120000  0x00000c90        0
RP/0/RP0/CPU0Sep 22 01:40:26.997 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libios.dll  0xfc121000  0x0002c4bc  0xfc14e000  0x00002000        0
RP/0/RP0/CPU0Sep 22 01:40:26.997 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libc.dll  0xfc150000  0x00077ae0  0xfc1c8000  0x00002000        0
RP/0/RP0/CPU0Sep 22 01:40:26.998 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libsyslog.dll  0xfc1d2000  0x0000530c  0xfc120c90  0x00000308        0
RP/0/RP0/CPU0Sep 22 01:40:26.998 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libbackplane.dll  0xfc1d8000  0x0000134c  0xfc0c2e4c  0x000000a8        0
RP/0/RP0/CPU0Sep 22 01:40:26.999 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libnodeid.dll  0xfc1e5000  0x00009114  0xfc1e41a8  0x00000208        0
RP/0/RP0/CPU0Sep 22 01:40:26.999 : dumper[54]: %DUMPER-7-DLL_INFO :
```

```
 /pkg/lib/libttyserver.dll  0xfc1f1000  0x0003dfcc  0xfc22f000  0x00002000        0
RP/0/RP0/CPU0Sep 22 01:40:27.000 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libttytrace.dll  0xfc236000  0x00004024  0xfc1e44b8  0x000001c8        0
RP/0/RP0/CPU0Sep 22 01:40:27.000 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libdebug.dll  0xfc23b000  0x0000ef64  0xfc1e4680  0x00000550        0
RP/0/RP0/CPU0Sep 22 01:40:27.001 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/lib_procfs_util.dll  0xfc24a000  0x00004e2c  0xfc1e4bd0  0x000002a8        0
RP/0/RP0/CPU0Sep 22 01:40:27.001 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libsysdb.dll  0xfc24f000  0x000452e0  0xfc295000  0x00000758        0
RP/0/RP0/CPU0Sep 22 01:40:27.001 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libsysdbutils.dll  0xfc296000  0x0000ae08  0xfc295758  0x000003ec        0
RP/0/RP0/CPU0Sep 22 01:40:27.002 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/lib_tty_svr_error.dll  0xfc2a1000  0x0000172c  0xfc1e4e78  0x00000088        0
RP/0/RP0/CPU0Sep 22 01:40:27.002 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/lib_tty_error.dll  0xfc2a3000  0x00001610  0xfc1e4f00  0x00000088        0
RP/0/RP0/CPU0Sep 22 01:40:27.003 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libwd_evm.dll  0xfc2a5000  0x0000481c  0xfc295b44  0x00000188        0
RP/0/RP0/CPU0Sep 22 01:40:27.003 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libttydb.dll  0xfc2aa000  0x000051dc  0xfc295ccc  0x00000188        0
RP/0/RP0/CPU0Sep 22 01:40:27.004 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libttydb_error.dll  0xfc23a024  0x00000f0c  0xfc295e54  0x00000088        0
RP/0/RP0/CPU0Sep 22 01:40:27.004 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/librs232.dll  0xfc2b0000  0x00009c28  0xfc2ba000  0x00000470        0
RP/0/RP0/CPU0Sep 22 01:40:27.005 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/lib_rs232_error.dll  0xfc2bb000  0x00000f8c  0xfc295edc  0x00000088        0
RP/0/RP0/CPU0Sep 22 01:40:27.005 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libst16550.dll  0xfc2bc000  0x00008ed4  0xfc2ba470  0x00000430        0
RP/0/RP0/CPU0Sep 22 01:40:27.006 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libconaux.dll  0xfc2c5000  0x00001dc0  0xfc2ba8a0  0x000001a8        0
RP/0/RP0/CPU0Sep 22 01:40:27.006 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/lib_conaux_error.dll  0xfc1ee114  0x00000e78  0xfc295f64  0x00000088        0
RP/0/RP0/CPU0Sep 22 01:40:27.007 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libttyutil.dll  0xfc2c7000  0x00003078  0xfc2baa48  0x00000168        0
RP/0/RP0/CPU0Sep 22 01:40:27.007 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libbag.dll  0xfc431000  0x0000ee98  0xfc40cc94  0x00000368        0
RP/0/RP0/CPU0Sep 22 01:40:27.008 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libchkpt.dll  0xfc474000  0x0002ecf8  0xfc4a3000  0x00000950        0
RP/0/RP0/CPU0Sep 22 01:40:27.008 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libsysdbbackend.dll  0xfc8ed000  0x0000997c  0xfc8d3aa8  0x0000028c        0
RP/0/RP0/CPU0Sep 22 01:40:27.008 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libttymgmtconnection.dll  0xfce85000  0x00004208  0xfce8a000  0x00000468
0
RP/0/RP0/CPU0Sep 22 01:40:27.009 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libttymgmt.dll  0xfcea4000  0x0000e944  0xfce8abf0  0x000003c8        0
RP/0/RP0/CPU0Sep 22 01:40:27.009 : dumper[54]: %DUMPER-7-DLL_INFO :
 /pkg/lib/libttynmspc.dll  0xfcec7000  0x00004a70  0xfcec6644  0x000002c8        0
RP/0/RP0/CPU0Sep 22 01:40:28.396 : dumper[54]: %DUMPER-5-CORE_FILE_NAME :
 Core for process pkg/bin/devc-conaux at harddisk:/coredump/devc-conaux.by.
 dumper_gen.sparse.20040922-014027.node0_RP0_CPU0.ppc.Z
RP/0/RP0/CPU0Sep 22 01:40:32.309 : dumper[54]: %DUMPER-5-DUMP_SUCCESS : Core dump success
```

exception coresize

Halts the creation of the core file beyond the configured core file size limit.

exception coresize *size*
no exception coresize

Syntax Description

coresize	*size*	Defines the maximum limit of the core file size beyond which the core file creation is halted and only the stack trace output is printed on the screen. The core file size limit can range from 1 to 4095 MB.

Command Default

This command has no default behavior.

Command Modes

Global configuration

Command History

Release	Modification
Release 5.1.1	This command was introduced.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

The following scenarios are applicable for creating full or sparse core dumps:

- Without the **exception sparse** configuration or exception sparse OFF, and default core size (4095 MB), a full core is created till the core size. Beyond this, only stack trace is collected.
- With non-default core size and without the **exception sparse** configuration, or exception sparse OFF , a full core is created until the core size limit is reached. Beyond the core size limit, only the stack trace is collected.
- With the exception sparse ON and default core size (4095 MB), a full core is created until the sparse size limit is reached, and a sparse core is created thereafter till the core size. Beyond this, only stack trace is collected.
- With non-default core size and with the exception sparse ON, a full core is created until the sparse size limit is reached. Beyond the sparse size limit, only the stack trace is collected.

Note By default, full core dumps are created irrespective of the **exception sparse** configuration. If there is not enough free shared memory available, then the core dump process fails.

Task ID

Task ID	Operations
diag	read, write

The following example shows how you can disable the creation of core dump files by specifying the limit for core file size.

```
RP/0/RP0/CPU0:router(config)# exception coresize 1024
RP/0/RP0/CPU0:router(config)# commit
```

exception filepath

To modify core dump settings, use the **exception filepath** command in the appropriate configuration mode. To remove the configuration, use the **no** form of this command.

exception filepath*filepath-name*
noexception filepath*filepath-name*

Syntax Description

filepath-name	Local file system or network protocol, followed by the directory path. All local file systems are supported. The following network protocols are supported: TFTP and FTP.

Command Default

If you do not specify the order of preference for the destination of core dump files using the **choice** *preference* keyword and argument, the default preference is the primary location or 1.

Core dump files are sent compressed.

The default file naming convention used for core dump files is described in Table 1: Default Core Dump File Naming Convention Description, on page 9.

Command Modes

XR Config

Command History

Release	Modification
Release 5.0.0	This command was introduced.
Release 3.9.0	No modification.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Use the **exception filepath** command to modify core dump settings, such as the destination file path to store core dump files, file compression, and the filename appended to core dumps.

Up to three user-defined locations may be configured as the preferred destinations for core dump files:

- Primary location—The primary destination for core dump files. Enter the **choice** keyword and a value of **1** (that is, **choice 1**) for the *preference* argument to specify a destination as the primary location for core dump files.
- Secondary location—The secondary fallback choice for the destination for core dump files, if the primary location is unavailable (for example, if the hard disk is set as the primary location and the hard disk fails). Enter the **choice** keyword and a value of **2** (that is, **choice 2**) for the *preference* argument to specify a destination as the secondary location for core dump files.
- Tertiary location—The tertiary fallback choice as the destination for core dump files, if the primary and secondary locations fail. Enter the **choice** keyword and a value of 3 (that is, **choice 3**) for the *preference* argument to specify a destination as the tertiary location for core dump files.

When specifying a destination for a core dump file, you can specify an absolute file path on a local file system or on a network server. The following network protocols are supported: TFTP and FTP.

In addition to the three preferred destinations that can be configured, Cisco IOS XR software provides three default fallback destinations for core dump files in the event that user-defined locations are unavailable.

The default fallback destinations are:

Note If a default destination is a boot device, the core dump file is not sent to that destination.

We recommend that you configure at least one preferred destination for core dump files as a preventive measure if the default fallback paths are unavailable. Configuring at least one preferred destination also ensures that core dump files are archived because the default fallback destinations store only the first and last core dump files for a crashed process.

Note Cisco IOS XR software does not save a core file on a local storage device if the size of the core dump file creates a low-memory condition.

By default, Cisco IOS XR software assigns filenames to core dump files according to the following format:

process [.by. *requester* |.abort][.sparse]. *date-time* . *node* . *processor-type* [.Z]

For example:

```
packet.by.dumper_gen.20040921-024800.node0_RP0_CPU0.ppc.Z
```

Table 1: Default Core Dump File Naming Convention Description, on page 9 describes the default core dump file naming convention.

Table 1: Default Core Dump File Naming Convention Description

Field	Description
process	Name of the process that generated the core dump.
.by. *requester* | .abort	If the core dump was generated because of a request by a process (requester), the core filename contains the string ".by.*requester*" where the *requester* variable is the name or process ID (PID) of the process that requested the core dump. If the core dump was due to a self-generated abort call request, the core filename contains the string ".abort" instead of the name of the requester.
.sparse	If a sparse core dump was generated instead of a full core dump, "sparse" appears in the core dump filename.
.date-time	Date and time the dumper process was called by the process manager to generate the core dump. The *.date-time* time-stamp variable is expressed in the *yyyy.mm.dd-hh.mm.ss* format. Including the time stamp in the filename uniquely identifies the core dump filename.
. *node*	Node ID, expressed in the *rack/slot* notation, where the process that generated the core dump was running.
.processor-type	Type of processor (mips or ppc).

Field	Description
.Z	If the core dump was sent compressed, the filename contains the .Z suffix.

You can modify the default naming convention by specifying a filename to be appended to core dump files with the optional **filename** *filename* keyword and argument and by specifying a lower and higher limit ranges of values to be appended to core dump filenames with the *lower-limit* and *higher-limit* arguments, respectively. The filename that you specify for the *filename* argument is appended to the core dump file and the lower and higher limit ranges of core dump files to be sent to a specified destination before the filenames are recycled. Valid values for the *lower-limit* argument are 0 to 4. Valid values for the *higher-limit* argument are 5 to 64. A hyphen (-) must immediately follow the *lower-limit* argument. In addition, to uniquely identify each core dump file, a value is appended to each core dump file, beginning with the lower-limit value specified with the *lower-limit* argument and continuing until the higher-limit value specified with the *higher-limit* argument has been reached. When the configured higher-limit value has been reached, Cisco IOS XR software begins to recycle the values appended to core dump files, beginning with the lower-limit value.

Task ID

Task ID	Operations
diag	read, write

The following example shows how to use the command:

```
RP/0/RP1/CPU0:Linkwood(config)#exception filepath f1
```

Related Topics

exception pakmem, on page 11
exception sparse, on page 13
exception sprsize, on page 15
show exception, on page 33

exception pakmem

To configure the collection of packet memory information in core dump files, use the **exception pakmem** command in administration configuration mode or in global configuration mode. To remove the configuration, use the **no** form of this command.

exception pakmem {**on** | **off**}
no exception pakmem {**on** | **off**}

Syntax Description

on	Enables the collection of packet memory information in core dump files.
off	Disables the collection of packet memory information in core dump files.

Command Default

Packet memory information is not included in core dump files.

Command Modes

Administration configuration

Global configuration

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Use the **exception pakmem** command with the **on** keyword to configure the collection of packet memory information in core dump files. Cisco Technical Support Center engineers and development engineers use packet memory information to debug packet memory issues related to a process.

Caution Including packet memory information in core dump files significantly increases the amount of data generated in the core dump file, which may delay the restart time for the process.

Task ID

Task ID	Operations
diag	read, write

The following example shows how to configure core dumps to include packet memory information:

```
RP/0/RP0/CPU0:router(config)# exception pakmem on
```

Related Topics

exception filepath, on page 8
exception sparse, on page 13
exception sprsize, on page 15
show exception, on page 33

exception sparse

To enable or disable sparse core dumps, use the **exception sparse** command in administration configuration mode or in global configuration mode. To remove the configuration, use the **no** form of this command.

exception sparse {**on** | **off**}
no exception sparse

Syntax Description

on	Enables sparse core dumps.
off	Disables sparse core dumps

Command Default

Sparse core dumps are disabled.

Command Modes

Administration configuration

Global configuration

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Use the **exception sparse** command to reduce the amount of data generated in the core dump file. Sparse core dumps reduce the amount of time required to generate the core dump file because only referenced data is generated in the core file (at the cost of lost information in the core file). Reducing the time required to generate core dump files corresponds to faster process restart times.

Note Use the **exception sparse off** command in administration configuration mode to get a complete coredump of the transient processes on the RP.

Sparse core dumps contain the following information about crashed processes:

- Register information for all threads, and any memory pages referenced in these register values
- Stack information for all threads, and any memory pages referenced in these threads
- All memory pages referenced by a loaded dynamic loadable library (DLL) data section, if the final program counter falls in a DLL data section
- Any user-specified marker pages from the lib_dumper_marker DLL

The **exception sparse** command dumps memory pages based on trigger addresses found in the previously listed dump information, according to the following criteria:

- If the trigger address in the memory page is in the beginning 128 bytes of the memory page, the previous memory page in the continuous address region is dumped also.
- If the trigger address in the memory page is in the final 128 bytes of the memory page, the next memory page in the continuous address region is dumped also.
- In all other instances, only the memory page that includes the trigger address is dumped.

The following scenarios are applicable for creating full or sparse core dumps:

- Without the **exception sparse** configuration or exception sparse OFF, and default core size (4095 MB), a full core is created till the core size. Beyond this, only stack trace is collected.
- With non-default core size and without the **exception sparse** configuration, or exception sparse OFF , a full core is created until the core size limit is reached. Beyond the core size limit, only the stack trace is collected.
- With the exception sparse ON and default core size (4095 MB), a full core is created until the sparse size limit is reached, and a sparse core is created thereafter till the core size. Beyond this, only stack trace is collected.
- With non-default core size and with the exception sparse ON, a full core is created until the sparse size limit is reached. Beyond the sparse size limit, only the stack trace is collected.

Note By default, full core dumps are created irrespective of the **exception sparse** configuration. If there is not enough free shared memory available, then the core dump process fails.

Task ID

Task ID	Operations
diag	read, write

The following example shows how to enable sparse core dumps:

```
RP/0/RP0/CPU0:router(config)# exception sparse on
```

Related Topics

exception filepath, on page 8
exception pakmem, on page 11
exception sprsize, on page 15
show exception, on page 33

exception sprsize

To specify the maximum file size for core dumps, use the **exception sprsize** command in administration configuration mode or in global configuration mode. To remove the configuration, use the **no** form of this command.

exception sprsize *megabytes*
no exception sprsize

Syntax Description

megabytes	Size in megabytes (MB).

Command Default

megabytes : 192

Command Modes

Administration configuration

Global configuration

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Use the **exception sprsize** command to specify the maximum file size for core dumps. The maximum file size configured for the *megabytes* argument is used with the configuration set for the exception sparse, on page 13 command to determine whether or not to generate a sparse core dump file. If sparse core dumps are disabled and a core dump file is predicted to exceed the default value (192 MB) uncompressed or the value specified for the *megabytes* argument uncompressed, a sparse core dump file is generated. If sparse core dumps are enabled, a sparse core dump file is generated, regardless of the size of the core dump file.

The following scenarios are applicable for creating full or sparse core dumps:

- Without the **exception sparse** configuration or exception sparse OFF, and default core size (4095 MB), a full core is created till the core size. Beyond this, only stack trace is collected.
- With non-default core size and without the **exception sparse** configuration, or exception sparse OFF , a full core is created until the core size limit is reached. Beyond the core size limit, only the stack trace is collected.
- With the exception sparse ON and default core size (4095 MB), a full core is created until the sparse size limit is reached, and a sparse core is created thereafter till the core size. Beyond this, only stack trace is collected.
- With non-default core size and with the exception sparse ON, a full core is created until the sparse size limit is reached. Beyond the sparse size limit, only the stack trace is collected.

Note By default, full core dumps are created irrespective of the **exception sparse** configuration. If there is not enough free shared memory available, then the core dump process fails.

Task ID

Task ID	Operations
diag	read, write

The following example shows how to set the file size of sparse core dumps to 300 MB:

```
RP/0/RP0/CPU0:router(config)# exception sprsize 300
```

Related Topics

exception sparse, on page 13

follow

To unobtrusively debug a live process or a live thread in a process, use the **follow process** command in XR EXEC modeSystem Admin EXEC mode.

follow process [{*pid* | **location** *node-id*}]

Syntax Description

pid	Follows the process with the process ID (PID) specified for the *pid* argument.
location *node-id*	Follows the target process on the designated node. The *node-id* argument is expressed in the *rack*/*slot* notation.

Command Default

Entering the **follow process** command without any keyword displays the stack information of the live processes with all the threads, heap memory usage, and register values.

Command Modes

XR EXEC mode

System Admin EXEC mode

Command History

Release	Modification
Release 5.0.0	This command was introduced.

Usage Guidelines

Use this command to unintrusively debug a live process or a live thread in a process. This command is particularly useful for debugging deadlock and livelock conditions, for examining the contents of a memory location or a variable in a process to determine the cause of a corruption issue, or in investigating issues where a thread is stuck spinning in a loop. A livelock condition is one that occurs when two or more processes continually change their state in response to changes in the other processes.

The following actions can be specified with this command:

- Follow all live threads of a given process or a given thread of a process and print stack trace in a format similar to core dump output.
- Display register values and status information of the target process.

Take a snapshot of the execution path of a thread asynchronously to investigate performance-related issues by specifying a high number of iterations with a zero delay.

Task ID

Task ID	Operations
basic-services	read

This example shows how to use the **follow process** command:

```
sysadmin-vm:0_RP0# follow process 1 location 0/RP0

Location : 0/RP0
```

```
************************************************************

2013-09-20 01:57:30
Text address        Size        Library name
-----------------------------------------
00007f4b8a66c000 48 r-x--   libnss_files-2.12.so
00007f4b8a879000 1444 r-x--   libc-2.12.so
00007f4b8abec000 48 r-x--   libpci.so
00007f4b8adf9000 32 r-x--   librt-2.12.so
00007f4b8b002000 248 r-x--   libdbus-1.so.3.4.0
00007f4b8b241000 96 r-x--   libpthread-2.12.so
00007f4b8b45e000 128 r-x--   ld-2.12.so
-----------------------------------------
#0  0x00007f4b8a955c83 in select+0x13 from /lib64/libc-2.12.so
#1  0x000000000041f974 in ?? () from /sbin/init
#2  0x0000000000404b9d in ?? () from /sbin/init
#3  0x00007f4b8a897cce in __libc_start_main+0xfe from /lib64/libc-2.12.so
#4  0x0000000000404659 in ?? () from /sbin/init
```

Related Topics

monitor threads, on page 19
show processes, on page 41

monitor threads

To display auto-updating statistics on threads in a full-screen mode, use the **monitor threads** command in

XR EXEC

mode.

monitor threads [**dumbtty**] [**iteration** *number*] [**location** *node-id*]

Syntax Description

dumbtty	(Optional) Displays the output of the command as if on a dumb terminal (the screen is not refreshed).
iteration *number*	(Optional) Number of times the statistics display is to be updated, in the range from 0 to 4294967295.
location *node-id*	(Optional) Displays the output from the command from the designated node. The *node-id* argument is entered in the *rack/slot* notation.

Command Default

When all keywords are omitted, the **monitor threads** command displays the first ten threads for the local node, sorted in descending order by the time used. The display is cleared and updated every 5 seconds until you quit the command.

Command Modes

XR EXEC

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Use the **monitor threads** command to show the top ten threads based on CPU usage. The display refreshes every 10 seconds.

- To change the parameters displayed by the **monitor threads** command, enter one of the key commands described in Table 2: Interactive Display Commands for the monitor threads Command, on page 20.
- To terminate the display and return to the system prompt, enter the **q** key.
- To list the interactive commands, type ? during the display.

Table 2: Interactive Display Commands for the monitor threads Command, on page 20 describes the available interactive display commands.

Table 2: Interactive Display Commands for the monitor threads Command

Command	Description
?	Displays the available interactive commands.
d	Changes the delay interval between updates.
k	Kills a process.
l	Refreshes the screen.
n	Changes the number of threads to be displayed.
q	Quits the interactive display and returns the prompt to EXEC mode.

Task ID

Task ID	Operations
basic-services	execute

The following example shows sample output from the **monitor threads** command:

```
RP/0/RP0/CPU0:router# monitor threads

195 processes; 628 threads;
CPU states: 98.2% idle, 0.9% user, 0.7% kernel
Memory: 2048M total, 1576M avail, page size 4K

   JID   TID LAST_CPU PRI STATE  HH:MM:SS      CPU  COMMAND
     1    12   1       10 Rcv     0:00:09     0.42% procnto-600-smp-cisco-instr
     1    25   1       10 Run     0:00:30     0.36% procnto-600-smp-cisco-instr
   342     1   1       19 Rcv     0:00:07     0.20% wdsysmon
    52     5   0       21 Rcv     0:00:03     0.15% devc-conaux
    52     3   1       18 Rcv     0:00:02     0.07% devc-conaux
532670     1   0       10 Rply    0:00:00     0.07% top
   293     6   0       55 Rcv     0:00:06     0.03% shelfmgr
    55     8   0       10 Rcv     0:00:02     0.03% eth_server
   315     3   0       10 Rcv     0:00:11     0.03% sysdb_svr_local
    55     7   0       55 Rcv     0:00:11     0.02% eth_server
```

The following example shows sample output from the **monitor threads** command using the optional **location** keyword:

```
RP/0/RP0/CPU0:router# monitor threads location 0/RP0/CPU0

Computing times...195 processes; 628 threads;
CPU states: 95.1% idle, 2.7% user, 2.0% kernel
Memory: 2048M total, 1576M avail, page size 4K

   JID   TID LAST_CPU PRI STATE  HH:MM:SS      CPU  COMMAND
     1    25   0       10 Run     0:00:32     2.08% procnto-600-smp-cisco-instr
   265     5   0       10 SigW    0:00:09     0.89% packet
   279     1   1       10 Rcv     0:00:00     0.65% qsm
557246     1   0       10 Rply    0:00:00     0.51% top
   293     5   1       55 Rcv     0:00:01     0.07% shelfmgr
   180    13   1       10 Rcv     0:00:02     0.07% gsp
   315     3   0       10 Rcv     0:00:12     0.07% sysdb_svr_local
```

```
   55     7   1      55 Rcv     0:00:12     0.04% eth_server
  180     1   0      10 Rcv     0:00:01     0.04% gsp
  298     9   0      10 Rcv     0:00:01     0.04% snmpd
```

Table 3: monitor threads Field Descriptions, on page 21 describes the significant fields shown in the display.

Table 3: monitor threads Field Descriptions

Field	Description
JID	Job ID.
TIDS	Thread ID.
LAST_CPU	Number of open channels.
PRI	Priority level of the thread.
STATE	State of the thread.
HH:MM:SS	Run time of process since last restart.
CPU	Percentage of CPU used by process thread.
COMMAND	Process name.

Using Interactive Commands

When the **n** or **d** interactive command is used, the **monitor threads** command prompts for a number appropriate to the specific interactive command. The following example shows sample output from the **monitor threads** command using the interactive **n** command after the first display cycle to change the number of threads:

```
RP/0/RP0/CPU0:router# monitor threads

Computing times... 87 processes; 249 threads;
CPU states: 84.8% idle, 4.2% user, 10.9% kernel
Memory: 256M total, 175M avail, page size 4K

   JID   TID PRI STATE  HH:MM:SS      CPU  COMMAND
     1     6  10 Run     0:00:10    10.92% kernel
553049     1  10 Rply    0:00:00     4.20% top
    58     3  10 Rcv     0:00:24     0.00% sysdbsvr
     1     3  10 Rcv     0:00:21     0.00% kernel
    69     1  10 Rcv     0:00:20     0.00% wdsysmon
     1     5  10 Rcv     0:00:20     0.00% kernel
   159     2  10 Rcv     0:00:05     0.00% qnet
   160     1  10 Rcv     0:00:05     0.00% netio
   157     1  10 NSlp    0:00:04     0.00% envmon_periodic
   160     9  10 Intr    0:00:04     0.00% netio
n

Enter number of threads to display: 3
Please enter a number between 5 and 40
Enter number of threads to display: 8
```

```
87 processes; 249 threads;
CPU states: 95.3% idle, 2.9% user, 1.7% kernel
Memory: 256M total, 175M avail, page size 4K

   JID   TID PRI STATE  HH:MM:SS      CPU  COMMAND
     1     6  10 Run     0:00:11     1.76% kernel
    69     1  10 Rcv     0:00:20     1.11% wdsysmon
    58     3  10 Rcv     0:00:24     0.40% sysdbsvr
   157     1  10 NSlp    0:00:04     0.23% envmon_periodic
   159    19  10 Rcv     0:00:02     0.20% qnet
553049     1  10 Rply    0:00:00     0.20% top
   159    12  10 Rcv     0:00:03     0.13% qnet
   160     1  10 Rcv     0:00:05     0.10% netio
```

When a number outside the acceptable range is entered, the acceptable range is displayed:

```
Please enter a number between 5 and 40
Enter number of threads to display:
```

Related Topics

monitor processes

process

To terminate or restart a process, use the **process** command in the System Admin EXEC mode.

process {**crash** | **restart**} *executable-name* {*IID* **location** *node-id* | **location** *node-id*}

Syntax Description

crash	Ends a process. All active services hosted by the process that have high availability enabled are switched off and the process restarts.
restart	Restarts a process.
executable-name	Executable name of the process to be crashed or restarted. Supplying an executable name for the executable-name argument performs the action for all the simultaneously running instances of the process, if applicable.
IID	Process instance ID of the process to be crashed or restarted. Supplying a process ID for the *IID* argument performs the action for only the process instance associated with the process ID.
location *node-id*	Crashes or restarts a process on the designated node. The *node-id* argument is entered in the *rack/slot* notation.

Command Default

None

Command Modes

System Admin EXEC

Command History

Release	Modification
Release 5.0.0	This command was introduced.

Usage Guidelines

Under normal circumstances, processes are started and restarted automatically by the operating system as required. If a process crashes, it is automatically restarted.

Use this command to manually stop or restart individual processes.

Caution Manually stopping or restarting a process can seriously impact the operation of a router. Use these commands only under the direction of a Cisco Technical Support representative.

process restart

The **process restart** command restarts a process, such as a process that is not functioning optimally.

Task ID

Task ID	Operations
root-lr	execute

This example shows how to restart a process:

```
sysadmin-vm:0_RP0# process restart syslogd_helper location 0/3

proc-action-status User root (127.0.0.1) requested restart for process syslogd_helper(0)
at 0/3  'Sending signal 15 to process syslogd_helper(IID 0) pid=1801'
```

Related Topics

process mandatory, on page 25
show processes, on page 41

process mandatory

To set the mandatory reboot options for a process, use the **process mandatory** command in the appropriate mode.

process mandatory
process mandatory {**on** | **off**} {*executable-namejob-id*} **location** *node-id*

process mandatory toggle
process mandatory toggle {*executable-namejob-id*} **location** *node-id*

Syntax Description

on	Turns on mandatory process attribute.
off	Turns off the mandatory process attribute. The process is not considered mandatory.
toggle	Toggles a mandatory process attribute.
executable-name	Executable name of the process to be terminated. Specifying an executable name for the *executable-name* argument terminates the process and all the simultaneously running copies, if applicable.
job-id	Job ID associated with the process to be terminated. Terminates only the process associated with the job ID.
location *node-id*	Sets the mandatory settings for a process on a designated node. The node-id argument is expressed in the *rack/slot* notation.

Command Default

No default behavior or values

Command Modes

XR EXEC

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

If a process unexpectedly goes down, the following action occurs based on whether the process is considered mandatory.

- If the process is mandatory and the process cannot be restarted, the node automatically reboots.
- If the process is not mandatory and cannot be restarted, it stays down and the node does not reboot.

Task ID

Task ID	Operations
root-lr	execute

The following example shows how to turn on a mandatory attribute. In this example, the mandatory attribute is turned on for the media_ether_config_di process.

```
RP/0/RP0/CPU0:router# process mandatory on media_ether_config_di
```

The following example shows how to turn the reboot option on. In this example, the router is set to reboot the node if a mandatory process goes down and cannot be restarted.

```
RP/0/RP0/CPU0:router# process mandatory reboot enable

RP/0/0/CPU0:Mar 19 19:28:10 : sysmgr[71]: %SYSMGR-4-MANDATORY_REBOOT_ENABLE :
mandatory reboot option enabled by request
```

The following example shows how to turn off the reboot option. In this example, the router is set *not* to reboot the node if a mandatory process goes down and cannot be restarted. In this case, the mandatory process is restarted, but the node is not rebooted.

```
RP/0/RP0/CPU0:router# process mandatory reboot disable

RP/0/0/CPU0:Mar 19 19:31:20 : sysmgr[71]: %SYSMGR-4-MANDATORY_REBOOT_OVERRIDE
: mandatory reboot option overridden by request
```

Related Topics

show processes, on page 41

show context

To display core dump context information, use the **show context** command in

XR EXEC

mode.

show context [{*coredump-occurrence* | **clear**}] [**location** {*node-id* | **all**}]

Syntax Description

coredump-occurrence	(Optional) Core dump context information to be displayed based on the occurrence of the core dump. Valid values are 1 to 10.
clear	(Optional) Clears the current context information.
location { *node-id* | **all**}	Displays core dump information that occurred on the designated node. The *node-id* argument is expressed in the *rack/slot* notation. The **all** keyword specifies to display information for all nodes.

Command Default

If no *coredump-occurrence* value is specified, core dump context information for all core dumps is displayed.

Command Modes

XR EXEC

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Use the **show context** command to display core dump context information. This command displays context information for the last ten core dumps. Cisco Technical Support Center engineers and development engineers use this command for post-analysis in the debugging of processes.

Use the clear context, on page 2 command to clear core dump context information.

Task ID

Task ID	Operations
diag	read

The following example shows sample output from the **show context** command:

```
RP/0/RP0/CPU0:router# show context

Crashed pid = 20502 (pkg/bin/mbi-hello)
Crash time: Thu Mar 25, 2004: 19:34:14
```

```
Core for process at disk0:/mbi-hello.20040325-193414.node0_RP0_CPU0

    Stack Trace
#0 0xfc117c9c
#1 0xfc104348
#2 0xfc104154
#3 0xfc107578
#4 0xfc107734
#5 0x482009e4
                Registers info
            r0       r1       r2       r3
  R0   0000000e  481ffa80  4820c0b8  00000003
            r4       r5       r6       r7
  R4   481ffb18  00000001  481ffa88  48200434
            r8       r9      r10      r11
  R8   00000000  00000001  00000000  fc17ac58
           r12      r13      r14      r15
  R12  481ffb08  4820c080  481ffc10  00000001
           r16      r17      r18      r19
  R16  481ffc24  481ffc2c  481ffcb4  00000000
           r20      r21      r22      r23
  R20  00398020  00000000  481ffb6c  4820a484
           r24      r25      r26      r27
  R24  00000000  00000001  4820efe0  481ffb88
           r28      r29      r30      r31
  R28  00000001  481ffb18  4820ef08  00000001
           cnt       lr      msr       pc
  R32  fc168d58  fc104348  0000d932  fc117c9c
           cnd      xer
  R36  24000022  00000004

                      DLL Info
DLL path    Text addr.  Text size   Data addr. Data size   Version
/pkg/lib/libinfra.dll  0xfc0f6000  0x00032698  0xfc0f5268  0x00000cb4
```

The following example shows sample output from the **show context** command. The output displays information about a core dump from a process that has not crashed.

```
RP/0/RP0/CPU0:router# show context

node:      node0_RP0_CPU0
------------------------------------------------------------------

Crashed pid = 28703 (pkg/bin/packet)
Crash time: Tue Sep 21, 2004: 02:48:00
Core for process at harddisk:/packet.by.dumper_gen.20040921-024800.node0_RP0_CPU0.ppc.Z
```

Table 4: show context Field Descriptions, on page 28 describes the significant fields shown in the display.

Table 4: show context Field Descriptions

Field	Description
Crashed pid	Process ID (PID) of the crashed process followed by the executable path.
Crash time	Time and date the crash occurred.
Core for process at	File path to the core dump file.

Field	Description
Stack Trace	Stack trace information.
Registers Info	Register information related to crashed threads.
DLL Info	Dynamically loadable library (DLL) information used to decode the stack trace.

Related Topics

clear context, on page 2

show dll

To display dynamically loadable library (DLL) information, use the **show dll** command in

administration EXEC mode or in EXEC

XR EXEC

mode.

show dll [{**jobid** *job-id* [**virtual**] | [**symbol**]**address** *virtual-address* | **dllname** *dll-virtual-path* | **memory** | **virtual**}] [**location** *node-id*]

Syntax Description

jobid *job-id*	(Optional) Displays DLL information for the specified job identifier.
virtual	(Optional) Displays the virtual path of DLLs. The virtual path is expressed in the /pkg/lib/library-name.dll format where the library name is the name of the DLL followed by the .dll suffix.
symbol	(Optional) Displays the symbol at the virtual address specified for the *virtual-address* argument.
address *virtual-address*	(Optional) Displays the DLL that is mapped at the virtual address specified for the *virtual-address* argument.
dllname *dll-virtual-path*	(Optional) Displays the process IDs (PIDs) of the process that have downloaded the DLL specified for the *dll-virtual-path* argument.
memory	(Optional) Displays a summary of DLL memory usage.
location *node-id*	(Optional) Displays DLLs for the specified node. The *node-id* argument is expressed in the *rack/slot* notation.

Command Default

No default behavior or values

Command Modes

EXEC, Administration EXEC

XR EXEC

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Task ID

Task ID	Operations
basic-services	read

The following example shows sample output from the **show dll** command. In this example, the output displays all the DLLs loaded on the router.

```
RP/0/RP0/CPU0:router# show dll

DLL path                                Text VA    Text Sz    Data VA    Data Sz    Refcount
------------------------------------------------------------------------------------------
/lib/libui.dll                          0xfc000000 0x00007000 0xfc007000 0x00001000    1
/disk0/-base-0.48.0/lib/liblogin.dll 0xfc008000 0x00006000 0xfc00e000 0x00001000    1
/mbi/lib/libbanner.dll                  0xfc00f000 0x00003000 0xfc012000 0x00001000    1
/disk0/-base-0.48.0/lib/libaaav2.dll 0xfc013000 0x0000f000 0xfc022000 0x00001000    1
/disk0/-base-0.48.0/lib/libaaatty.dll 0xfc023000 0x00004000 0xfc027000 0x00001000    1
/mbi/lib/libtermcap.dll                 0xfc028000 0x00003000 0xfc02b000 0x00001000    1
/mbi/lib/lib_show_dll.dll               0xfc02c000 0x00004000 0xfc030000 0x00001000    1
/mbi/lib/libihplatform.dll              0xfc0bf2d4 0x00000c18 0xfc1e4f88 0x00000068    1
/lib/libovl.dll                         0xfc0c8000 0x0000c3b0 0xfc0c21f0 0x0000076c   23
/disk0/-admin-0.48.0/lib/libfqm_ltrace_util_common.dll 0xfc0d43b0 0x00000bfc 0xfc391f7c
0x00000068    1
/lib/libplatform.dll                    0xfc0d5000 0x0000aa88 0xfc0e0000 0x00002000  165
/lib/libsysmgr.dll                      0xfc0e2000 0x0000ab48 0xfc0c295c 0x00000368  166
/lib/libinfra.dll                       0xfc0ed000 0x0003284c 0xfc120000 0x00000c70  169
/lib/libios.dll                         0xfc121000 0x0002c4bc 0xfc14e000 0x00002000  166
/lib/libc.dll                           0xfc150000 0x00077ae0 0xfc1c8000 0x00002000  175
/mbi/lib/libltrace.dll                  0xfc1ca000 0x00007f5c 0xfc0c2cc4 0x00000188   96
/lib/libsyslog.dll                      0xfc1d2000 0x0000530c 0xfc120c70 0x00000308  129
/disk0/-base-0.48.0/lib/liblpts_ifib_platform.dll 0xfc1d730c 0x00000cc8 0xfcef4000 0x00000068
    1
/lib/libbackplane.dll                   0xfc1d8000 0x0000134c 0xfc0c2e4c 0x000000a8  163
/disk0/-base-0.48.0/lib/libipv6_platform_client.dll 0xfc1d934c 0x00000c48 0xfcef4f8c
0x00000068    1
/mbi/lib/libpkgfs_node.dll              0xfc1da000 0x000092d4 0xfc1e4000 0x000001a8    3
```

The following example shows sample output from the **show dll** command with the optional **jobid** *job-id* keyword and argument:

```
RP/0/RP0/CPU0:router# show dll jobid 186

DLLs mapped by PID 86111
DLL path                                Text VA    Text Sz    Data VA    Data Sz    Refcount
------------------------------------------------------------------------------------------
/lib/libovl.dll                         0xfc0c8000 0x0000c3b0 0xfc0c21f0 0x0000076c   23
/lib/libplatform.dll                    0xfc0d5000 0x0000aa88 0xfc0e0000 0x00002000  165
/lib/libsysmgr.dll                      0xfc0e2000 0x0000ab48 0xfc0c295c 0x00000368  167
/lib/libinfra.dll                       0xfc0ed000 0x0003284c 0xfc120000 0x00000c70  169
/lib/libios.dll                         0xfc121000 0x0002c4bc 0xfc14e000 0x00002000  166
/lib/libc.dll                           0xfc150000 0x00077ae0 0xfc1c8000 0x00002000  175
/mbi/lib/libltrace.dll                  0xfc1ca000 0x00007f5c 0xfc0c2cc4 0x00000188   96
/lib/libsyslog.dll                      0xfc1d2000 0x0000530c 0xfc120c70 0x00000308  129
/lib/libbackplane.dll                   0xfc1d8000 0x0000134c 0xfc0c2e4c 0x000000a8  163
/lib/libnodeid.dll                      0xfc1e5000 0x000091fc 0xfc1e41a8 0x00000208  163
/mbi/lib/libinst_mem.dll                0xfc232000 0x000044f8 0xfc1e43b0 0x00000108    4
/lib/libdebug.dll                       0xfc23c000 0x0000ef64 0xfc1e4680 0x00000550  159
```

Table 5: show dll Field Descriptions, on page 32 describes the significant fields shown in the display.

Table 5: show dll Field Descriptions

Field	Description
DLL path	Physical path of the DLL on the router.
Text VA	Virtual address of the text segment of the DLL.
Text Sz	Size of the text segment of the DLL.
Data VA	Virtual address of the data segment of the DLL.
Data Sz	Size of the data segment of the DLL.
Refcount	Number of clients using the DLL.

The following example shows sample output from the **show dll** command with the optional **dllname** *dll-virtual-path* keyword and optional argument:

```
RP/0/RP0/CPU0:router# show dll dllname /pkg/lib/libinst_mem.dll

PID:     4102  Refcount: 1
PID:     4105  Refcount: 1
PID:    24600  Refcount: 1
PID:    86111  Refcount: 1
```

Table 6: show dll dllname Field Descriptions, on page 32 describes the significant fields shown in the display.

Table 6: show dll dllname Field Descriptions

Field	Description
PID:	Process ID of the process.
Refcount	Number of references to the DLL by the process.

The following example shows sample **show dll** output from the command with the optional **memory** keyword:

```
RP/0/RP0/CPU0:router# show dll memory
------------------------------------------------------------------------------

Total DLL Text - 14778896 bytes  Total DLL Data - 12688500 bytes
Total DLL Memory - 27467396 bytes
```

show exception

To display the configured core dump settings, use the **show exception** command in

XR EXEC

mode.

show exception [**core-options** [**process** *process-name*] **location** *node-id*]

Syntax Description

core-options	(Optional) Displays process core option values.
process *process-name*	(Optional) Specifies the process for which to display the information.
location *node-id*	(Optional) Displays configured settings for a specified node. The *node-id* argument is expressed in the *rack/slot* notation.

Command Default

None

Command Modes

XR EXEC

Command History

Release	Modification
Release 5.0.0	This command was introduced.
Release 3.9.0	Support for the **core-options** keyword was added.

Usage Guidelines

To use this command, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Use the **show exception** command to display the configured core dump settings. The output from this command displays the core dump settings configured with the following commands:

- exception filepath, on page 8
- exception pakmem, on page 11
- exception sparse, on page 13
- exception sprsize, on page 15

Task ID

Task ID	Operations
diag	read

The following example shows sample output from the **show exception** command with the **location** keyword. All processes for the specified node are displayed.

```
RP/0/RP0/CPU0:router# show excep core-options location 0/rp0/cpu0

Mon Nov 30 01:31:31.391 PST
```

```
Process
        Options
attach_server:
        TEXT SHAREDMEM MAINMEM
attachd:
        TEXT SHAREDMEM MAINMEM
ksh-aux:
        TEXT SHAREDMEM MAINMEM
bcm_logger:
        TEXT SHAREDMEM MAINMEM
devf-scrp:
        TEXT SHAREDMEM MAINMEM
bfm_server:
        TEXT SHAREDMEM MAINMEM
ksh:
        TEXT SHAREDMEM MAINMEM
dllmgr:
        COPY
dumper:
        TEXT SHAREDMEM MAINMEM
eth_server:
        COPY SPARSE
inflator:
        TEXT SHAREDMEM MAINMEM
insthelper:
        TEXT SHAREDMEM MAINMEM
mbi-hello:
        TEXT SHAREDMEM MAINMEM
cat:
        TEXT SHAREDMEM MAINMEM
mq:
        COPY
mqueue:
        TEXT SHAREDMEM MAINMEM
nname:
        TEXT SHAREDMEM MAINMEM
nvram:
        TEXT SHAREDMEM MAINMEM
 --More--
```

The following example shows sample output from the **show exception** command for a specific process:

```
RP/0/RP0/CPU0:router# show excep core-options process upgrade_daemon location 0/6/cpu0

Mon Nov 30 01:32:20.207 PST
Process
        Options
upgrade_daemon:
        TEXT SHAREDMEM MAINMEM
```

Related Topics

exception filepath, on page 8
exception pakmem, on page 11
exception sparse, on page 13
exception sprsize, on page 15

show memory

To display the available physical memory and memory usage information of processes on the router, use the **show memory** command in System Admin EXEC and XR EXEC mode.

show memory [{**location** *node-id* | **pid** *pid* [**location** *node-id*] | **summary** [**location** *node-id*]}]

Syntax Description

location *node-id*	Displays the available physical memory from the designated node. The *node-id* argument is entered in the *rack/slot* notation.
pid *pid*	Displays memory usage of the specified process.
summary	Displays a summary of the physical memory and memory usage information.

Command Default

None

Command Modes

System Admin EXEC

XR EXEC

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

To display detailed memory information for the entire router, enter the **show memory** command without any parameters.

Task ID

Task ID	Operations
basic-services	read

This example shows how to display the output of the **show memory location** command:

```
sysadmin-vm:0_RP0#show memory location 0/RP0
Tue Aug  20 00:49:41.649 UTC
**********************************************************


Location : 0/RP0

**********************************************************

Tue Aug 20 00:49:41 UTC 2013
1:   /sbin/init
Address           Kbytes     RSS    Anon  Locked Mode   Mapping
0000000000400000     204       -       -       - r-x--  init
0000000000632000       4       -       -       - rw---  init
```

```
Address - Memory Address
Kbytes - Memory Size
RSS - Resident Set Size (portion of mem in RAM)
Anon - Non-shared Anonymous
Locked - locked memory
Mode - Read/Write/Executable mode
Mapping - process Mapping
```

Related Topics

show memory heap, on page 40
show processes, on page 41

show memory compare

To display details about heap memory usage for all processes on the router at different moments in time and compare the results, use the **show memory compare** command in System Admin EXEC and XR EXEC mode.

show memory compare {**start** | **end** | **report**}

Syntax Description

start	Takes the initial snapshot of heap memory usage for all processes on the router and sends the report to a temporary file named /tmp/memcmp_start.out.
end	Takes the second snapshot of heap memory usage for all processes on the router and sends the report to a temporary file named /tmp/memcmp_end.out. This snapshot is compared with the initial snapshot when displaying the heap memory usage comparison report.
report	Displays the heap memory comparison report, comparing heap memory usage between the two snapshots of heap memory usage.

Command Default

None

Command Modes

System Admin EXEC

XR EXEC

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

Use the **show memory compare** command to display details about the heap memory usage of all processes on the router at different moments in time and compare the results. This command is useful for detecting patterns of memory usage during events such as restarting processes or configuring interfaces.

Use the following steps to create and compare memory snapshots:

1. Enter the **show memory compare** command with the **start** keyword to take the initial snapshot of heap memory usage for all processes on the router.
2. Perform the test you want to analyze.
3. Enter the **show memory compare** command with the **end** keyword to take the snapshot of heap memory usage to be compared with the initial snapshot.
4. Enter the **show memory compare** command with the **report** keyword to display the heap memory usage comparison report.

Task ID

Task ID	Operations
basic-services	read

This example shows sample output from the **show memory compare** command with the **report** keyword:

```
sysadmin-vm:0_RP0# show memory compare start
Tue Aug  20 11:50:45.860 UTC
sysadmin-vm:0_RP0# show memory compare end
Tue Aug  20 11:50:57.311 UTC
sysadmin-vm:0_RP0# show memory compare report

PID       NAME           MEM BEFORE  MEM AFTER   DIFFERENCE  MALLOCS
-------------------------------------------------------------------------
21416  malloc_dump          34731       34731       0           0
21414  sh                   39652       39640       -12         0
21411  show_memory_common   984         984         0           0
8340   ntpd                 69033       69033       0           0
5172   inst_mgr             1800118     1800118     0           0
5166   fsdbagg              14907247    14907247    0           0
5175   fsdb_server          15475470    15475470    0           0
5177   led_mgr              3347339     3347339     0           0
5176   envmon_ui            889094      889094      0           0
5169   esdma                8954927     8954927     0           0
5164   fit_mgbl             952067      952067      0           0
5174   fab_fgid_service     9014924     9014924     0           0
5173   confd_helper         8018190     8018190     0           0
5171   debug_agent          8146830     8146830     0           0
5170   gaspp_mgbl           1285020     1285020     0           0
5168   ael_mgbl             787101      787101      0           0
5165   fpdserv              1149685     1149685     0           0
5167   ssh_key_server       661086      661086      0           0
2052   sfe_driver           35005323    35005323    0           0
2066   zen                  5083246     5083246     0           0
2017   ccc_driver           8872747     8882315     9568        1
2053   shelf_mgr            30666121    30666121    0           0
2031   esd                  6335087     6334783     -304        -2
2049   sdr_mgr              4366258     4366258     0           0
2025   dumper               616144      616144      0           0
2035   inst_agent           1820469     1820469     0           0
2062   syslogd_relay        657904      657904      0           0
2030   envmon               7853186     7853330     144         2
2041   ntp_helper           701348      701348      0           0
2539   ssh                  202441      202441      0           0
2015   bios_fpd             2950893     2950893     0           0
2042   obfl_mgr             2686006     2686006     0           0
2018   cm                   13755230    13755230    0           0
2047   obfl_show            686286      686286      0           0
2024   ds                   7826821     7826821     0           0
2060   syslogd_helper       912664      912664      0           0
2014   aaad                 804327      804327      0           0
2019   debug_client         577975      577975      0           0
2016   calv_alarm_mgr       2077250     2077250     0           0
2065   wdmon                3557984     3558056     72          1
2064   vm_manager           3149588     3149588     0           0
2037   mlap                 1520260     1520260     0           0
2056   ssh_key_client       612824      612824      0           0
2055   ship_server          778066      778066      0           0
2063   timezone_config      711110      711110      0           0
1744   pm                   7875584     7875584     0           0
```

Table 7: show memory compare report Field Descriptions

Field	Description
PID	Process ID.
name	Process name.
mem before	Heap memory usage at start (in bytes).
mem after	Heap memory usage at end (in bytes).
difference	Difference in heap memory usage (in bytes).
mallocs	Number of unfreed allocations made during the test period.
restarted	Indicates if the process was restarted during the test period.

Related Topics

show memory heap, on page 40
show processes, on page 41

show memory heap

To display information about the heap space for a process, use the **show memory heap** command in System Admin EXEC and XR EXEC mode.

show memory heap *pid*

Syntax Description

pid	Process ID

Command Default

None

Command Modes

System Admin EXEC

XR EXEC

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Task ID

Task ID	Operations
basic-services	read

This example shows the sample output from the **show memory heap** command:

```
sysadmin-vm:0_RP0#show memory heap 1933
Tue Aug  20 01:06:11.282 UTC

statistics (1933:vm_manager)

Global data:
 current usage:        3147787 bytes
 Wrapper uses:          109560 bytes(hash:32728)
 total high wm:        7342424 bytes
 current objs:            2401 entry
 malloc_db/malloc:       79946 times / 79946 times
 calloc_db/calloc:        1067 times / 1067 times
 realloc_db/realloc:     26342 times / 26342 times
 realloc_null:           25644 times
 realloc_db_miss :           0 times
 realloc_relocate:          39 times
 free_db/free:          104256 times / 104722 times
 free_null:                466 times
 free_db_miss:               0 times
 error:                      0 times
```

Related Topics

show memory, on page 35

show processes

To display information about active processes, use the **show processes** command in System Admin EXEC mode.

show processes {*process-name* {[{**detail** | **run**}] **location** *node-id* | **location** *node-id*} | **aborts location** *node-id* | **all location** *node-id* | **blocked** [{*PID* | **extended** | **location** *node-id*}] | **family** [{*PID* | **location** *node-id*}] | **files** [{*PID* | **details** | **location** *node-id*}] | **location** [{**all***node-id*}] | **mandatory location** *node-id* | **memory** [{*PID* | **location** *node-id*}] | **services** {*service-name* | **active** | **all** | **run** | **standby**} **location** *node-id* | **signal** [{*PID* | **location** *node-id*}] | **startup location** *node-id* | **threadname** [{*PID* | **location** *node-id*}]}

Syntax Description

process-name	Name of the executable.
detail	Displays detailed information of the process.
run	Displays information of running processes.
location *node-id*	Displays information about the active processes from a designated node. The *node-id* argument is entered in the *rack/slot* notation.
aborts	Displays process abort information.
all	Displays summary process information for all processes.
blocked	Displays details about reply, send, and mutex blocked processes.
PID	Displays process ID.
extended	Displays blocked processes in detail.
family	Displays the process session and family information.
files	Displays information about open files and open communication channels.
mandatory	Displays process data for mandatory processes.
memory	Displays information about the text, data, and stack usage for processes.
services *service name*	Displays service data for processes.
active	Displays active services data.
standby	Displays standby services data.
signal	Displays the signal options for blocked, pending, ignored, and queued signals.

startup	Displays process data for processes created at startup.
threadname	Displays thread names.

Command Default

None

Command Modes

System Admin EXEC

Command History

Release	Modification
Release 3.9.0	No modification.
Release 5.0.0	This command was introduced.

Usage Guidelines

Use the **show processes** command to display process level information across the system.

Task ID

Task ID	Operations
basic-services	read

The **show processes** command with the **memory** keyword displays details of memory usage for a given process as shown in the following example:

```
sysadmin-vm:0_RP0# show process memory


PID         Text       Data      Stack    Dynamic Process
=========================================================
1         204 KB     204 KB     136 KB   14932 KB init
12680      16 KB      48 KB     136 KB    3852 KB sleep
12747      32 KB    8432 KB     136 KB   24776 KB cmdptywrapper
12751      12 KB    8508 KB     136 KB   74040 KB show_processes_
12754     724 KB    8456 KB     136 KB   25832 KB sh
1299      724 KB     208 KB     136 KB   11280 KB oom.sh
1305      724 KB     208 KB     136 KB   11280 KB oom.sh
1443      476 KB     540 KB     136 KB   14984 KB dhclient
1486       28 KB     188 KB     136 KB    6104 KB syslogd
1490       20 KB    3056 KB     136 KB    6864 KB klogd
1545      224 KB     204 KB     136 KB   13172 KB lldpad
1557      308 KB     204 KB     136 KB   12844 KB dbus-daemon
1588      412 KB     444 KB     136 KB   23252 KB sshd
1593      412 KB     444 KB     136 KB   23252 KB sshd
1602      192 KB     372 KB     136 KB   11120 KB xinetd
1618       40 KB     692 KB     524 KB    7008 KB crond
1630      792 KB   49720 KB     136 KB   83164 KB libvirtd
1711      116 KB     636 KB     136 KB    4540 KB udevd
1712      116 KB     636 KB     136 KB    4540 KB udevd
1722      324 KB   16164 KB     136 KB  148164 KB pm
```

Table 8: show processes memory Field Descriptions

Field	Description
PID	Process ID.

Field	Description
Text	Size of text region (process executable).
Data	Size of data region (initialized and uninitialized variables).
Stack	Size of process stack.
Dynamic	Size of dynamically allocated memory.
Process	Process name.

Related Topics

monitor processes
monitor threads, on page 19