Process Memory Management Commands

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

Administration EXEC

XR EXEC mode

Command History

Release	Modification
Release 7.0.12	This command was introduced.

Usage Guidelines

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
```

dumpcore

To manually generate a core dump, use the **dumpcore** command in XR 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

XR EXEC mode

Command History

Release	Modification
Release 7.0.12	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 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 [**choice** *preference*] [**compress** {**on** | **off**}] **filename** *filename lower-limit−higher-limit* **filepath** *filepath-name*
no exception [**choice** *preference*] [**compress** {**on** | **off**}] **filename** *filename lower-limit−higher-limit* **filepath** *filepath-name*

Syntax Description

choice *preference*	(Optional) Configures the order of preference for the destination of core dump files. Up to the three destinations can be defined. Valid values are 1 to 3.
compress {**on** | **off**}	(Optional) Specifies whether or not the core dump file should be sent compressed. By default, core dump files are sent compressed. If you specify the **compress** keyword, you must specify one of the following required keywords: • **on** —Compresses the core dump file before sending it. • **off** —Does not compress the core dump file before sending it.
filename *filename lower-limit-higher-limit*	(Optional) Specifies the filename to be appended to core dump files and the lower and higher limit range of core dump files to be sent to a specified destination before being recycled by the circular buffer. **filename** *filename lower-limit-higher-limit* See exception filepath, on page 6 for a description of the default core dump file naming convention. Vali **filename** *filename lower-limit-higher-limit* d 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. **Note** To uniquely identify each core dump file, a value is appended to each core dump file, beginning with the lower limit value configured for the *lower-limit* argument and continuing until the higher limit value configured for the *higher-limit* argument has been reached. After the higher limit value has been reached, the Cisco IOS XR software begins to recycle the values appended to core dump files, beginning with the lower limit value.
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.

Command Modes

Global configuration

XR Config

Command History

Release	Modification
Release 7.0.12	This command was introduced.

Usage Guidelines

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.

Note We recommend that you specify a location on the hard disk as the primary location.

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:

- harddisk:/dumper
- disk1:/dumper
- disk0:/dumper

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

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/module* notation, where the process that generated the core dump was running.
.*processor-type*	Type of processor (mips or ppc).
.*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 configure the core dump setting for the primary user-defined preferred location. In this example, core files are configured to be sent uncompressed; the filename of core dump files is set to “core” (that is, all core filenames will be named core); the range value is set from 0 to 5 (that is, the values 0 to 5 are appended to the filename for the first five generated core dump files, respectively, before being recycled); and the destination is set to a directory on the hard disk.

```
RP/0/RP0/CPU0:router(config)# exception choice 1 compress off
  filename core 0-5 filepath /harddisk:/corefile
```

follow

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

follow {**job** *job-id* | **process** *pid* | **location** *node-id*} [**all**] [**blocked**] [**debug** *level*] [**delay** *seconds*] [**dump** *address size*] [**iteration** *count*] [**priority** *level*] [**stackonly**] [**thread** *tid*] [**verbose**]

Syntax Description

job *job-id*	Follows a process by job ID.
process *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.
all	(Optional) Follows all threads.
blocked	(Optional) Follows the chain of thread IDs (TIDs) or PIDs that are blocking the target process.
debug *level*	(Optional) Sets the debug level for the following operation. Valid values for the level argument are 0 to 10.
delay *seconds*	(Optional) Sets the delay interval between each iteration. Valid values for the *seconds* argument are 0 to 255 seconds.
dump *address size*	(Optional) Dumps the memory segment starting with the specified memory address and size specified for the *address* and *size* arguments.
iteration *count*	(Optional) Specifies the number of times to display information. Valid values for the *count* argument are 0 to 255 iterations.
priority *level*	(Optional) Sets the priority level for the following operation. Valid values for the *level* argument are 1 to 63.
stackonly	(Optional) Displays only stack trace information.
thread *tid*	(Optional) Follows the TID of a process or job ID specified for the *tid* argument.
verbose	(Optional) Displays register and status information pertaining to the target process.

Command Default

Entering the **follow** command without any optional keywords or arguments performs the operation for five iterations from the local node with a delay of 5 seconds between each iteration. The output includes information about all live threads. This command uses the default scheduling priority from where the command is being run.

Command Modes

XR EXEC mode

Command History

Release	Modification
Release 7.0.12	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.
- Follow a process in a loop for a given number of iterations.
- Set a delay between two iterations while invoking the command.
- Set the priority at which this process should run while this command is being run.
- Dump memory from a given virtual memory location for a given size.
- 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

The following example shows how to use the **follow** command to debug the process associated with job ID 257 for one iteration:

```
RP/0/RP0/CPU0:router# follow job 257 iteration 1

Attaching to process pid = 28703 (pkg/bin/packet)
No tid specified, following all threads

DLL Loaded by this process
------------------------------

DLL path                 Text addr. Text size  Data addr. Data size  Version
/pkg/lib/libovl.dll      0xfc0c9000 0x0000c398 0xfc0c31f0 0x0000076c        0
/pkg/lib/libplatform.dll 0xfc0d6000 0x0000aa88 0xfc0e1000 0x00002000        0
/pkg/lib/libsysmgr.dll   0xfc0e3000 0x0000aeac 0xfc0c395c 0x00000388        0
/pkg/lib/libinfra.dll    0xfc0ee000 0x000332ec 0xfc122000 0x00000c70        0
/pkg/lib/libios.dll      0xfc123000 0x0002c4bc 0xfc150000 0x00002000        0
/pkg/lib/libc.dll        0xfc152000 0x00077ae0 0xfc1ca000 0x00002000        0
/pkg/lib/libsyslog.dll   0xfc1d4000 0x0000530c 0xfc122c70 0x00000308        0
/pkg/lib/libbackplane.dll 0xfc1da000 0x0000134c 0xfc0c3e6c 0x000000a8        0
/pkg/lib/libnodeid.dll   0xfc1e7000 0x000091fc 0xfc1e61a8 0x00000208        0
/pkg/lib/libdebug.dll    0xfc23e000 0x0000ef64 0xfc1e6680 0x00000550        0
/pkg/lib/lib_procfs_util.dll 0xfc24d000 0x00004e2c 0xfc1e6bd0 0x000002a8        0
/pkg/lib/libsysdb.dll    0xfc252000 0x00046224 0xfc299000 0x0000079c        0
```

```
/pkg/lib/libsysdbutils.dll 0xfc29a000 0x0000ae04 0xfc29979c 0x000003ec        0
/pkg/lib/libwd_evm.dll   0xfc2a9000 0x0000481c 0xfc299b88 0x00000188        0
/pkg/lib/lib_mutex_monitor.dll 0xfc35e000 0x00002414 0xfc340850 0x00000128        0
/pkg/lib/libchkpt.dll    0xfc477000 0x0002ee04 0xfc474388 0x00000950        0
/pkg/lib/libpacket_common.dll 0xfc617000 0x000130f0 0xfc6056a0 0x000007b0        0

Iteration 1 of 1
------------------------------

Current process = "pkg/bin/packet", PID = 28703 TID = 1

trace_back: #0 0xfc1106dc [MsgReceivev]
trace_back: #1 0xfc0fc840 [msg_receivev]
trace_back: #2 0xfc0fc64c [msg_receive]
trace_back: #3 0xfc0ffa70 [event_dispatch]
trace_back: #4 0xfc0ffc2c [event_block]
trace_back: #5 0x48204410 [<N/A>]

ENDOFSTACKTRACE

Current process = "pkg/bin/packet", PID = 28703 TID = 2

trace_back: #0 0xfc1106dc [MsgReceivev]
trace_back: #1 0xfc0fc840 [msg_receivev]
trace_back: #2 0xfc0fc64c [msg_receive]
trace_back: #3 0xfc0ffa70 [event_dispatch]
trace_back: #4 0xfc0ffc2c [event_block]
trace_back: #5 0xfc48d848 [chk_evm_thread]

ENDOFSTACKTRACE

Current process = "pkg/bin/packet", PID = 28703 TID = 3

trace_back: #0 0xfc17d54c [SignalWaitinfo]
trace_back: #1 0xfc161c64 [sigwaitinfo]
trace_back: #2 0xfc10302c [event_signal_thread]

ENDOFSTACKTRACE

Current process = "pkg/bin/packet", PID = 28703 TID = 4

trace_back: #0 0xfc1106c4 [MsgReceivePulse]
trace_back: #1 0xfc0fc604 [msg_receive_async]
trace_back: #2 0xfc0ffa70 [event_dispatch]
trace_back: #3 0xfc0ffc5c [event_block_async]
trace_back: #4 0xfc35e36c [receive_events]

ENDOFSTACKTRACE

Current process = "pkg/bin/packet", PID = 28703 TID = 5

trace_back: #0 0xfc17d564 [SignalWaitinfo_r]
trace_back: #1 0xfc161c28 [sigwait]
trace_back: #2 0x48203928 [<N/A>]

ENDOFSTACKTRACE
```

The following example shows how to use the **follow** command to debug TID 5 of the process associated with job ID 257 for one iteration:

```
RP/0/RP0/CPU0:router# follow job 257 iteration 1 thread 5
```

```
Attaching to process pid = 28703 (pkg/bin/packet)

DLL Loaded by this process
------------------------------

DLL path                  Text addr. Text size  Data addr. Data size  Version
/pkg/lib/libovl.dll       0xfc0c9000 0x0000c398 0xfc0c31f0 0x0000076c        0
/pkg/lib/libplatform.dll 0xfc0d6000 0x0000aa88 0xfc0e1000 0x00002000        0
/pkg/lib/libsysmgr.dll    0xfc0e3000 0x0000aeac 0xfc0c395c 0x00000388        0
/pkg/lib/libinfra.dll     0xfc0ee000 0x000332ec 0xfc122000 0x00000c70        0
/pkg/lib/libios.dll       0xfc123000 0x0002c4bc 0xfc150000 0x00002000        0
/pkg/lib/libc.dll         0xfc152000 0x00077ae0 0xfc1ca000 0x00002000        0
/pkg/lib/libsyslog.dll    0xfc1d4000 0x0000530c 0xfc122c70 0x00000308        0
/pkg/lib/libbackplane.dll 0xfc1da000 0x0000134c 0xfc0c3e6c 0x000000a8        0
/pkg/lib/libnodeid.dll    0xfc1e7000 0x000091fc 0xfc1e61a8 0x00000208        0
/pkg/lib/libdebug.dll     0xfc23e000 0x0000ef64 0xfc1e6680 0x00000550        0
/pkg/lib/lib_procfs_util.dll 0xfc24d000 0x00004e2c 0xfc1e6bd0 0x000002a8        0
/pkg/lib/libsysdb.dll     0xfc252000 0x00046224 0xfc299000 0x0000079c        0
/pkg/lib/libsysdbutils.dll 0xfc29a000 0x0000ae04 0xfc29979c 0x000003ec        0
/pkg/lib/libwd_evm.dll    0xfc2a9000 0x0000481c 0xfc299b88 0x00000188        0
/pkg/lib/lib_mutex_monitor.dll 0xfc35e000 0x00002414 0xfc340850 0x00000128        0
/pkg/lib/libchkpt.dll     0xfc477000 0x0002ee04 0xfc474388 0x00000950        0
/pkg/lib/libpacket_common.dll 0xfc617000 0x000130f0 0xfc6056a0 0x000007b0        0

Iteration 1 of 1
------------------------------

Current process = "pkg/bin/packet", PID = 28703 TID = 5

trace_back: #0 0xfc17d564 [SignalWaitinfo_r]
trace_back: #1 0xfc161c28 [sigwait]
trace_back: #2 0x48203928 [<N/A>]

ENDOFSTACKTRACE
```

The following example shows how to use the **follow** command to debug the chain of threads blocking thread 2 associated with the process assigned PID 139406:

```
RP/0/RP0/CPU0:router# follow process 139406 blocked iteration 1 thread 2

Attaching to process pid = 139406 (pkg/bin/lpts_fm)

DLL Loaded by this process
------------------------------

DLL path                  Text addr. Text size  Data addr. Data size  Version
/pkg/lib/libplatform.dll 0xfc0d6000 0x0000aa88 0xfc0e1000 0x00002000        0
/pkg/lib/libsysmgr.dll    0xfc0e3000 0x0000aeac 0xfc0c395c 0x00000388        0
/pkg/lib/libinfra.dll     0xfc0ee000 0x000332ec 0xfc122000 0x00000c70        0
/pkg/lib/libios.dll       0xfc123000 0x0002c4bc 0xfc150000 0x00002000        0
/pkg/lib/libc.dll         0xfc152000 0x00077ae0 0xfc1ca000 0x00002000        0
/pkg/lib/libltrace.dll    0xfc1cc000 0x00007f5c 0xfc0c3ce4 0x00000188        0
/pkg/lib/libsyslog.dll    0xfc1d4000 0x0000530c 0xfc122c70 0x00000308        0
/pkg/lib/libbackplane.dll 0xfc1da000 0x0000134c 0xfc0c3e6c 0x000000a8        0
/pkg/lib/libnodeid.dll    0xfc1e7000 0x000091fc 0xfc1e61a8 0x00000208        0
/pkg/lib/libdebug.dll     0xfc23e000 0x0000ef64 0xfc1e6680 0x00000550        0
/pkg/lib/lib_procfs_util.dll 0xfc24d000 0x00004e2c 0xfc1e6bd0 0x000002a8        0
/pkg/lib/libsysdb.dll     0xfc252000 0x00046224 0xfc299000 0x0000079c        0
/pkg/lib/libsysdbutils.dll 0xfc29a000 0x0000ae04 0xfc29979c 0x000003ec        0
/pkg/lib/libwd_evm.dll    0xfc2a9000 0x0000481c 0xfc299b88 0x00000188        0
/pkg/lib/libbag.dll       0xfc40c000 0x0000ee98 0xfc41b000 0x00000368        0
/pkg/lib/libwd_notif.dll 0xfc4f8000 0x00005000 0xfc4fd000 0x00001000        0
```

```
/pkg/lib/libifmgr.dll    0xfc665000 0x00029780 0xfc68f000 0x00003000        0
/pkg/lib/libnetio_client.dll 0xfca6a000 0x000065c8 0xfca2c4f8 0x000001b4        0
/pkg/lib/libpa_client.dll 0xfcec5000 0x00006e9c 0xfcecc000 0x00003000        0
/pkg/lib/libltimes.dll   0xfcecf000 0x00002964 0xfcdc4f20 0x000000a8        0

Iteration 1 of 1
-------------------------------

Current process = "pkg/bin/lpts_fm", PID = 139406 TID = 2

trace_back: #0 0xfc110744 [MsgSendv]
trace_back: #1 0xfc0fbf04 [msg_sendv]
trace_back: #2 0xfc0fbbd8 [msg_send]
trace_back: #3 0xfcec7580 [pa_fm_close]
trace_back: #4 0xfcec78b0 [pa_fm_process_0]

ENDOFSTACKTRACE

REPLY (node node0_RP1_CPU0, pid 57433)

No specific TID, following all threads of 57433 (pkg/bin/lpts_pa)
---------------------------------------------------------------------

DLL Loaded by this process
-------------------------------

DLL path                 Text addr. Text size  Data addr. Data size  Version
/pkg/lib/libplatform.dll 0xfc0d6000 0x0000aa88 0xfc0e1000 0x00002000        0
/pkg/lib/libsysmgr.dll   0xfc0e3000 0x0000aeac 0xfc0c395c 0x00000388        0
/pkg/lib/libinfra.dll    0xfc0ee000 0x000332ec 0xfc122000 0x00000c70        0
/pkg/lib/libios.dll      0xfc123000 0x0002c4bc 0xfc150000 0x00002000        0
/pkg/lib/libc.dll        0xfc152000 0x00077ae0 0xfc1ca000 0x00002000        0
/pkg/lib/libltrace.dll   0xfc1cc000 0x00007f5c 0xfc0c3ce4 0x00000188        0
/pkg/lib/libsyslog.dll   0xfc1d4000 0x0000530c 0xfc122c70 0x00000308        0
/pkg/lib/libbackplane.dll 0xfc1da000 0x0000134c 0xfc0c3e6c 0x000000a8        0
/pkg/lib/libnodeid.dll   0xfc1e7000 0x000091fc 0xfc1e61a8 0x00000208        0
/pkg/lib/libdebug.dll    0xfc23e000 0x0000ef64 0xfc1e6680 0x00000550        0
/pkg/lib/lib_procfs_util.dll 0xfc24d000 0x00004e2c 0xfc1e6bd0 0x000002a8        0
/pkg/lib/libsysdb.dll    0xfc252000 0x00046224 0xfc299000 0x0000079c        0
/pkg/lib/libsysdbutils.dll 0xfc29a000 0x0000ae04 0xfc29979c 0x000003ec        0
/pkg/lib/libwd_evm.dll   0xfc2a9000 0x0000481c 0xfc299b88 0x00000188        0
/pkg/lib/lrdlib.dll      0xfc2f6000 0x0000a900 0xfc2f551c 0x00000610        0
/pkg/lib/liblrfuncs.dll  0xfc30e000 0x00001998 0xfc2ebd80 0x000001ec        0
/pkg/lib/libdscapi.dll   0xfc310000 0x0000457c 0xfc2f5b2c 0x0000035c        0
/pkg/lib/liblrdshared.dll 0xfc315000 0x00005fec 0xfc31b000 0x00002000        0
/pkg/lib/libbag.dll      0xfc40c000 0x0000ee98 0xfc41b000 0x00000368        0
/pkg/lib/libchkpt.dll    0xfc477000 0x0002ee04 0xfc474388 0x00000950        0
/pkg/lib/libwd_notif.dll 0xfc4f8000 0x00005000 0xfc4fd000 0x00001000        0
/pkg/lib/libltrace_sdt.dll 0xfc65c000 0x000034fc 0xfc65b73c 0x00000568        0
/pkg/lib/libfabhandle.dll 0xfc6be000 0x00003354 0xfc65bca4 0x00000248        0
/pkg/lib/libfsdb_ltrace_util_rt.dll 0xfc6ea000 0x00001b74 0xfc605e50 0x00000108        0
/pkg/lib/libbcdl.dll     0xfc6fb000 0x0000f220 0xfc6fa6e8 0x0000045c        0
/pkg/lib/liblpts_pa_fgid.dll 0xfc8d7000 0x00006640 0xfc7acd5c 0x00000208        0
/pkg/lib/libfgid.dll     0xfc910000 0x0001529c 0xfc926000 0x00002000        0
/pkg/lib/libltimes.dll   0xfcecf000 0x00002964 0xfcdc4f20 0x000000a8        0

Current process = "pkg/bin/lpts_pa", PID = 57433 TID = 1

trace_back: #0 0xfc1106dc [MsgReceivev]
trace_back: #1 0xfc0fc840 [msg_receivev]
trace_back: #2 0xfc0fc64c [msg_receive]
trace_back: #3 0xfc0ffa70 [event_dispatch]
trace_back: #4 0xfc0ffc2c [event_block]
trace_back: #5 0x48201904 [<N/A>]
```

```
trace_back: #6 0x48201e3c [<N/A>]

ENDOFSTACKTRACE

Current process = "pkg/bin/lpts_pa", PID = 57433 TID = 2

trace_back: #0 0xfc1106dc [MsgReceivev]
trace_back: #1 0xfc0fc840 [msg_receivev]
trace_back: #2 0xfc0fc64c [msg_receive]
trace_back: #3 0xfc0ffa70 [event_dispatch]
trace_back: #4 0xfc0ffc2c [event_block]
trace_back: #5 0x4821e978 [<N/A>]

ENDOFSTACKTRACE

Current process = "pkg/bin/lpts_pa", PID = 57433 TID = 3

trace_back: #0 0xfc1106dc [MsgReceivev]
trace_back: #1 0xfc0fc840 [msg_receivev]
trace_back: #2 0xfc0fc64c [msg_receive]
trace_back: #3 0xfc0ffa70 [event_dispatch]
trace_back: #4 0xfc0ffc2c [event_block]
trace_back: #5 0x482064c4 [<N/A>]

ENDOFSTACKTRACE
```

The following example shows how to use the **follow** command to debug the chain of threads blocking thread 2 associated with the process assigned PID 139406:

```
RP/0/RP0/CPU0:router# follow process 139406 blocked iteration 1 stackonly thread 2

Attaching to process pid = 139406 (pkg/bin/lpts_fm)

Iteration 1 of 1
------------------------------

Current process = "pkg/bin/lpts_fm", PID = 139406 TID = 2

trace_back: #0 0xfc110744 [MsgSendv]
trace_back: #1 0xfc0fbf04 [msg_sendv]
trace_back: #2 0xfc0fbbd8 [msg_send]
trace_back: #3 0xfcec7580 [pa_fm_close]
trace_back: #4 0xfcec78b0 [pa_fm_process_0]

ENDOFSTACKTRACE

REPLY (node node0_RP1_CPU0, pid 57433)

No specific TID, following all threads of 57433 (pkg/bin/lpts_pa)
---------------------------------------------------------------

Current process = "pkg/bin/lpts_pa", PID = 57433 TID = 1

trace_back: #0 0xfc1106dc [MsgReceivev]
trace_back: #1 0xfc0fc840 [msg_receivev]
trace_back: #2 0xfc0fc64c [msg_receive]
trace_back: #3 0xfc0ffa70 [event_dispatch]
trace_back: #4 0xfc0ffc2c [event_block]
trace_back: #5 0x48201904 [<N/A>]
trace_back: #6 0x48201e3c [<N/A>]

ENDOFSTACKTRACE
```

```
Current process = "pkg/bin/lpts_pa", PID = 57433 TID = 2

trace_back: #0 0xfc1106dc [MsgReceivev]
trace_back: #1 0xfc0fc840 [msg_receivev]
trace_back: #2 0xfc0fc64c [msg_receive]
trace_back: #3 0xfc0ffa70 [event_dispatch]
trace_back: #4 0xfc0ffc2c [event_block]
trace_back: #5 0x4821e978 [<N/A>]

ENDOFSTACKTRACE

Current process = "pkg/bin/lpts_pa", PID = 57433 TID = 3

trace_back: #0 0xfc1106dc [MsgReceivev]
trace_back: #1 0xfc0fc840 [msg_receivev]
trace_back: #2 0xfc0fc64c [msg_receive]
trace_back: #3 0xfc0ffa70 [event_dispatch]
trace_back: #4 0xfc0ffc2c [event_block]
trace_back: #5 0x482064c4 [<N/A>]

ENDOFSTACKTRACE
```

process

To start, terminate, or restart a process, use the **process** command in admin EXEC mode.

process {**crash** | **restart** | **shutdown** | **start**} {*executable-namejob-id*} **location** {*node-id* | **all**}

Syntax Description

crash	Crashes a process.
restart	Restarts a process.
shutdown	Stops a process. The process is not restarted (even if considered "mandatory?).
start	Starts a process.
executable-name	Executable name of the process to be started, terminated, 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.
job-id	Job ID of the process instance to be started, terminated, or restarted. Supplying a job ID for the *job-id* argument performs the action for only the process instance associated with the job ID.
location { *node-id* | **all**}	Starts, terminates, or restarts a process on the designated node. The *node-id* argument is entered in the *rack/slot* notation. The **all** keyword specifies all nodes.

Command Default

None

Command Modes

Admin EXEC

Command History

Release	Modification
Release 7.0.12	This command was introduced.

Usage Guidelines

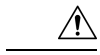
Under normal circumstances, processes are started and restarted automatically by the operating system as required. If a process crashes, it is automatically restarted.

Use this command to manually start, 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 shutdown

The **process shutdown** command shuts down (terminates) the specified process and copies associated with the specified process. The process is not restarted, even if considered "mandatory.? Use the **show processes** command to display a list of executable processes running on the system.



Caution Stopping a process can result in an RP switchover, system failure or both. This command is intended for use only under the direct supervision of a Cisco Technical Support representative.

process restart

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

process start

The **process start** command starts a process that is not currently running, such as a process that was terminated using the **process kill** command. If multiple copies are on the system, all instances of the process are started simultaneously.

Task ID

Task ID	Operations
root-lr	execute

The following example shows how to restart a process. In this example, the IS-IS process is restarted:

```
RP/0/RSP0/CPU0:router# process restart isis

RP/0/RSP0/CPU0:router#RP/0/RSP0/CPU0:Mar 30 15:24:41 : isis[343]: %ISIS-6-INFO_ST
RTUP_START : Cisco NSF controlled start beginning
RP/0/RSP0/CPU0:router#RP/0/RSP0/CPU0:Mar 30 15:24:52 : isis[352]: %ISIS-6-INFO_ST
RTUP_FINISH : Cold controlled start completed
```

The following example shows how to terminate a process. In this example, the IS-IS process is stopped:

```
RP/0/RP0/CPU0:router# process shutdown isis
RP/0/RP0/CPU0:router#
```

The following example shows how to start a process. In this example, the IS-IS process is started:

```
RP/0/RSP0/CPU0:router# process start isis

RP/0/RSP0/CPU0:router#RP/0/RSP0/CPU0:Mar 30 15:27:19 : isis[227]:
   %ISIS-6-INFO_STARTUP_START : Cold controlled start beginning
RP/0/RSP0/CPU0:Mar 30 15:27:31 : isis[352]: %ISIS-6-INFO_STARTUP_FINISH :
   Cold controlled start completed
```

process core

To modify the core dump options for a process, use the **process core** command in administration EXEC mode.

process {*executable-namejob-id*} **core** {**context** | **copy** | **fallback** | **iomem** | **mainmem** | **off** | **sharedmem** | **sparse** | **sync** | **text**} [**maxcore** *value*] **location** *node-id*

Syntax Description

executable-name	Executable name of the process for which you want to change core dump options. Specifying a value for the *executable-name* argument changes the core dump option for multiple instances of a running process.
job-id	Job ID associated with the process instance. Specifying a *job-id* value changes the core dump option for only a single instance of a running process.
context	Dumps only context information for a process.
copy	Copies a core dump locally before performing the core dump.
fallback	Sets the core dump options to use the fallback options (if needed).
iomem	Dumps the I/O memory of a process.
mainmem	Dumps the main memory of a process.
off	Indicates that a core dump is not taken on the termination of the specified process.
sharedmem	Dumps the shared memory of a process.
sparse	Enables sparse core dumps of a process.
sync	Enables only synchronous core dumping.
text	Dumps the text of a process.
maxcore *value*	(Optional) Specifies the maximum number of core dumps allowed for the specified process on its creation.
location *node-id*	Sets the core dump options for a process on a designated node. The *node-id* argument is entered in the *rack/slot* notation.

Command Default

By default, processes are configured to dump shared memory, text area, stack, data section, and heap information.

Command Modes

Administration EXEC

Command History

Release	Modification
Release 7.0.12	This command was introduced.

Usage Guidelines

The modular architecture of Cisco IOS XR software allows core dumps for individual processes. By default, processes are configured to dump shared memory, text area, stack, data section, and heap information.

Specifying an executable name for the *executable-name job-id* argument changes the core dump option for all instances of the process. Specifying a job ID for the value changes the core dump option for a single instance of a running process.

Task ID

Task ID	Operations
root-lr	execute

The following example shows how to enable the collection of shared memory of a process:

```
RP/0/RP0/CPU0:router# process ospf core sharedmem
```

The following example shows how to turn off core dumping for a process:

```
RP/0/RP0/CPU0:router# process media_ether_config_di core off
```

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 reboot
process mandatory reboot {**enable** | **disable**}

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.
reboot { **enable** | **disable**}	Enables or disables the reboot action when a mandatory process fails.
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

Administration EXEC

EXEC

Command History

Release	Modification
Release 7.0.12	This command was introduced.

Usage Guidelines

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
```

show context

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

administration EXEC mode or in 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

EXEC, Administration EXEC

Command History

Release	Modification
Release 7.0.12	This command was introduced.

Usage Guidelines

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
```

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

Table 2: 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.
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.

show memory

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

show memory [*jobid* | **summary** [**bytes** | **detail**]] **location** *node-id*

Syntax Description

job id	(Optional) Job ID associated with a process instance. Specifying a job ID for the *job-id* argument displays the memory available and memory usage information for only the process associated with the specified job ID. If the *job-id* argument is not specified, this command displays information for all running processes.
summary	(Optional) Displays a summary of the physical memory and memory usage information.
bytes	(Optional) Displays numbers in bytes for an exact count.
detail	(Optional) Displays numbers in the format "nnn.dddM" for more detail.
location *node-id*	Displays the available physical memory from the designated node. The *node-id* argument is entered in the *rack/slot* notation.

Command Default

None

Command Modes

Administration EXEC

EXEC

Command History

Release	Modification
Release 7.0.12	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 partial sample output from the **show memory** command entered without keywords or arguments. This command displays details for the entire router.

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

  Physical Memory:2048M total
   Application Memory :1802M (1636M available)
   Image:116M (bootram:116M)
   Reserved:128M, IOMem:0, flashfsys:0
   Total shared window:0
```

```
  kernel:jid 1
  Address         Bytes           What
  0008f000        12288           Program Stack
  000b2000        12288           Program Stack
  Total Allocated Memory:0
  Total Shared Memory:0

  sbin/devc-pty:jid 68
  Address         Bytes           What
  4817f000        4096            Program Stack (pages not allocated)
  48180000        516096          Program Stack (pages not allocated)
  481fe000        8192            Program Stack
  48200000        28672           Physical Mapped Memory
  48207000        4096            ANON FIXED ELF SYSRAM
  48208000        4096            ANON FIXED ELF SYSRAM
```

This example shows sample output from the **show memory** command entered with the job ID 7 to show the memory usage information for the process associated with this job identifier:

```
RP/0/RP0/CPU0:router# show memory 7

  Physical Memory: 256M total
   Application Memory : 249M (217M available)
   Image: 2M (bootram: 2M)
   Reserved: 4M, IOMem: 0, flashfsys: 0

  sbin/pipe: jid 7
  Address         Bytes           What
  07f7c000        126976          Program Stack (pages not allocated)
  07f9b000        4096            Program Stack
  07f9d000        126976          Program Stack (pages not allocated)
  07fbc000        4096            Program Stack
  07fbe000        126976          Program Stack (pages not allocated)
  07fdd000        4096            Program Stack
  07fdf000        126976          Program Stack (pages not allocated)
  07ffe000        4096            Program Stack
  08000000        122880          Program Stack (pages not allocated)
  0801e000        8192            Program Stack
  08020000        12288           Physical Mapped Memory
  08023000        4096            Program Text or Data
  08024000        4096            Program Text or Data
  08025000        16384           Allocated Memory
  08029000        16384           Allocated Memory
  7c001000        319488          DLL Text libc.dll
  7e000000        8192            DLL Data libc.dll
```

This example shows how to display a detailed summary of memory information for the router:

```
RP/0/RP0/CPU0:router# show memory summary detail

  Physical Memory: 256.000M total
   Application Memory : 140.178M (15.003M available)
   Image: 95.739M (bootram: 95.739M)
   Reserved: 20.000M, IOMem: 0, flashfsys: 0
   Shared window fibv6: 257.980K
   Shared window PFI_IFH: 207.925K
   Shared window aib: 8.972M
   Shared window infra_statsd: 3.980K
   Shared window ipv4_fib: 1.300M
   Shared window atc_cache: 35.937K
```

```
Shared window qad: 39.621K
Total shared window: 10.805M
Allocated Memory: 49.933M
Program Text: 6.578M
Program Data: 636.000K
Program Stack: 4.781M
```

Table 3: show memory summary Field Descriptions

Field	Description
Physical Memory	Available physical memory on the router.
Application Memory	Current memory usage of all the processes on the router.
Image	Memory that is currently used by the image and available memory.
Reserved	Total reserved memory.
IOMem	Available I/O memory.
flashfsys	Total flash memory.
Shared window fibv6	Internal shared window information.
Shared window PFI_IFH	Internal shared window information.
Shared window aib	Internal shared window information.
Shared window infra_statsd	Internal shared window information.
Shared window ipv4_fib	Internal shared window information.
Shared window atc_cache	Internal shared window information.
Shared window qad	Internal shared window information.
Total shared window	Internal shared window information.
Allocated Memory	Amount of memory allocated for the specified node.
Program Text	Internal program test information.
Program Data	Internal program data information.
Program Stack	Internal program stack information.

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 EXEC or administration EXEC System Admin 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

Administration EXEC

EXEC

XR EXEC mode

Command History

Release	Modification
Release 7.0.12	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.

Note The snapshot is similar to that resulting from entry of the show memory heap, on page 31 command with the optional **summary** keyword.

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:

```
Router# show memory compare report

  JID   name                 mem before   mem after    difference   mallocs  restarted
  ---   ----                 ----------   ---------    ----------   -------  ---------
  84    driver_infra_partner 577828       661492       83664        65
  279   gsp                  268092       335060       66968        396
  236   snap_transport       39816        80816        41000        5
  237   mpls_lsd_agent       36340        77340        41000        5
  268   fint_partner         24704        65704        41000        5
  90    null_caps_partner    25676        66676        41000        5
  208   aib                  55320        96320        41000        5
  209   ipv4_io              119724       160724       41000        5
  103   loopback_caps_partne 33000        74000        41000        5
  190   ipv4_arm             41432        82432        41000        5
  191   ipv6_arm             33452        74452        41000        5
  104   sysldr               152164       193164       41000        5
  85    nd_partner           37200        78200        41000        5
  221   clns                 61520        102520       41000        5
  196   parser_server        1295440      1336440      41000        5
  75    bundlemgr_distrib    57424        98424        41000        5
  200   arp                  83720        124720       41000        5
  201   cdp                  56524        97524        41000        5
  204   ether_caps_partner   39620        80620        41000        5
  206   qosmgr               55624        96624        41000        5
  240   imd_server           92880        104680       11800        28
  260   improxy              77508        88644        11136        10
  111   nrssvr               29152        37232        8080         60
  275   sysdb_svr_local      1575532      1579056      3524         30
  205   cfgmgr               31724        33548        1824         25
  99    sysdb_svr_shared     1131188      1132868      1680         14
  51    mbus-rp              26712        27864        1152         4
  66    wdsysmon             298068       299216       1148         15
  168   netio                1010912      1012060      1148         6
  283   itrace_manager       17408        17928        520          3
  59    devc-conaux          109868       110300       432          4
  67    syslogd_helper       289200       289416       216          2
  117   fctl                 41596        41656        60           2
  54    sysmgr               171772       171076       -696         -5
  269   ifmgr                539308       530652       -8656        -196        *
```

Table 4: show memory compare report Field Descriptions

Field	Description
JID	Process job ID.
name	Process name.

Field	Description
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.

show memory heap

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

show memory heap [**allocated**] [**dllname**] [**failure**] [**free**] {*jobid* | **all**}

Syntax Description

allocated	(Optional) Displays a list of all allocated heap blocks.
dllname	(Optional) Displays heaps with dynamic link library (DLL) names.
failure	(Optional) Displays a summary of heap failures.
free	(Optional) Displays a list of all free heap blocks.
summary	(Optional) Displays a summary of the information about the heap space.
job-id	Job ID associated with the process instance.
all	(Optional) Displays information about the heap space for all processes. The **all** keyword is only available when the **failure** or **summary** keywords are used.

Command Default

None

Command Modes

Administration EXEC

EXEC

Command History

Release	Modification
Release 7.0.12	This command was introduced.

Usage Guidelines

No specific guidelines impact the use of this command.

Task ID

Task ID	Operations
basic-services	read

This example shows sample output from the **show memory heap** command, specifying a job ID for the *job-id* argument:

```
RP/0/RP0/CPU0:router# show memory heap 111

  Malloc summary for pid 16433:
    Heapsize 16384: allocd 6328, free 8820, overhead 1236
    Calls: mallocs 144; reallocs 73; frees 5; [core-allocs 1; core-frees 0]
```

```
Block Allocated List
Total      Total      Block       Name/ID/Caller
Usize      Size       Count

0x000008c1 0x000008cc 0x00000001  0x7c018a10
0x000005ac 0x00000974 0x00000079  0x7c02b9e0
0x000004f0 0x000004f8 0x00000001  0x7c02b6fc
0x00000080 0x00000088 0x00000001  0x7c01936c
0x00000034 0x00000048 0x00000001  0x7c018954
0x00000024 0x00000030 0x00000001  0x7c019278
0x00000018 0x00000020 0x00000001  0x7c019b2c
0x00000008 0x00000010 0x00000001  0x7c017178
0x00000008 0x00000010 0x00000001  0x7c00fb54
0x00000008 0x00000010 0x00000001  0x7c00fb80
0x00000008 0x00000010 0x00000001  0x7c00fbb8
```

Table 5: show memory heap Field Descriptions

Field	Description
Malloc summary for pid	System-defined process ID (PID).
Heapsize	Size of the heap as allocated from the system by the malloc library.
allocd	Bytes allocated to the process.
free	Bytes available in the heap.
overhead	Malloc library overhead in bytes.
mallocs	Number of malloc calls.
reallocs	Number of realloc calls.
frees	Number of invocations to the caller interface provided in the malloc library for deallocating the memory.
[core-allocs 1; core-frees 0]	Number of core memory units, the memory units in the malloc library allocated by the system for the heap, allocated, and freed.

The following example shows sample output from the **show memory heap** command, specifying the **summary** *job-id* keyword and argument:

```
RP/0/RP0/CPU0:router# show memory heap summary 65

  Malloc summary for pid 20495 process pcmciad:
    Heapsize 65536: allocd 40332, free 16568, overhead 8636
    Calls: mallocs 883; reallocs 3; frees 671; [core-allocs 4; core-frees 0]
  Band size 16, element per block 48, nbuint 1
    Completely free blocks: 0
    Block alloced: 2, Block freed: 0
    allocs: 85, frees: 20
    allocmem: 1040, freemem: 496, overhead: 448
    blocks: 2, blknodes: 96
  Band size 24, element per block 34, nbuint 1
    Completely free blocks: 0
    Block alloced: 1, Block freed: 0
```

```
  allocs: 243, frees: 223
  allocmem: 480, freemem: 336, overhead: 168
  blocks: 1, blknodes: 34
Band size 32, element per block 26, nbuint 1
  Completely free blocks: 0
  Block alloced: 1, Block freed: 0
  allocs: 107, frees: 97
  allocmem: 320, freemem: 512, overhead: 136
  blocks: 1, blknodes: 26
Band size 40, element per block 22, nbuint 1
  Completely free blocks: 0
  Block alloced: 2, Block freed: 0
  allocs: 98, frees: 74
  allocmem: 960, freemem: 800, overhead: 240
  blocks: 2, blknodes: 44
Band size 48, element per block 18, nbuint 1
  Completely free blocks: 0
  Block alloced: 1, Block freed: 0
  allocs: 53, frees: 42
  allocmem: 528, freemem: 336, overhead: 104
  blocks: 1, blknodes: 18
Band size 56, element per block 16, nbuint 1
  Completely free blocks: 0
  Block alloced: 1, Block freed: 0
  allocs: 8, frees: 4
  allocmem: 224, freemem: 672, overhead: 96
  blocks: 1, blknodes: 16
Band size 64, element per block 14, nbuint 1
  Completely free blocks: 0
  Block alloced: 1, Block freed: 0
  allocs: 6, frees: 2
  allocmem: 256, freemem: 640, overhead: 88
  blocks: 1, blknodes: 14
Band size 72, element per block 12, nbuint 1
  Completely free blocks: 0
  Block alloced: 1, Block freed: 0
  allocs: 1, frees: 0
  allocmem: 72, freemem: 792, overhead: 80
  blocks: 1, blknodes: 12
```

Table 6: show memory heap summary Field Descriptions

Field	Description
Malloc summary for pid	System-defined process ID (pid).
Heapsize	Size of the heap as allocated from the system by the malloc library.
allocd	Bytes allocated to the process.
free	Bytes available in the heap.
overhead	Malloc library overhead in bytes.
mallocs	Number of malloc calls.
reallocs	Number of realloc calls.

Field	Description
frees	Number of invocations to the caller interface provided in the malloc library for deallocating the memory.
[core-allocs 1; core-frees 0]	Number of core memory units, the memory units in the malloc library allocated by the system for the heap, allocated and freed.
Band size	Small memory elements are arranged in bands. The band size specifies the size of elements within the band.
element per block	Number of elements per block in the band.
nbunit	Number of memory unit one block consists of. Any block in any band should be of a size that is an integer multiple of this basic unit.
Completely free blocks	Number of blocks in the band completely free (available for allocation).
Block alloced	Number of blocks currently allocated for the band.
allocs	Number of allocations currently performed from the band.
frees	Number of free calls that resulted in memory being returned to the band.
allocmem	Amount of memory currently allocated from the band.
overhead	Amount of memory in bytes as overhead for managing the band.
blocks	Number of blocks currently in the band.
blknodes	Number of nodes (elements) in all the blocks in the band.

show processes

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

show processes {*job-idprocess-name* | **aborts** | **all** | **blocked** | **boot** | **cpu** | **distribution** *process-name* | **dynamic** | **failover** | **family** | **files** | **location** *node-id* | **log** | **mandatory** | **memory** | **pidin** | **searchpath** | **signal** | **startup** | **threadname**} [**location** *node-id*] [**detail**] [**run**]

Syntax Description

job-id	Job identifier for which information for only the process instance associated with the *job-id* argument is displayed.
process-name	Process name for which all simultaneously running instances are displayed, if applicable.
aborts	Displays process abort information.
all	Displays summary process information for all processes.
blocked	Displays details about reply, send, and mutex blocked processes.
boot	Displays process boot information.
cpu	Displays CPU usage for each process.
distribution	Displays the distribution of processes.
dynamic	Displays process data for dynamically created processes.
failover	Displays process switchover information.
family	Displays the process session and family information.
files	Displays information about open files and open communication channels.
location *node-id*	Displays information about the active processes from a designated node. The *node-id* argument is entered in the *rack/slot* notation.
log	Displays process log.
mandatory	Displays process data for mandatory processes.
memory	Displays information about the text, data, and stack usage for processes.
pidin	Displays all processes using the QNX command.
searchpath	Displays the search path.
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.

detail	(Optional) Displays more detail. This option is available only with the *process-name* argument.
run	(Optional) Displays information for only running processes. This option is available only with the *process-name* argument.

Command Default

None

Command Modes

Administration EXEC

EXEC

Command History

Release	Modification
Release 7.0.12	This command was introduced.

Usage Guidelines

- Use the **show processes** command to display general information about the active processes. To display more detailed information for a process, specify a job ID or process for the *job-id* argument or *process-name* argument, respectively.
- You can also use the **monitor processes** command to determine the top processes and threads based on CPU usage.
- If you execute the **show processes blocked** <> command when multiple show techs are being collected, a transient and an intermittent error would occur for a few seconds. You can handle this issue in one of the following ways:
  - Ignore the error and retry the **show processes blocked** <> command.
  - Avoid executing the **show processes blocked** <> command when multiple **show tech-support** <> commands are running.

Task ID

Task ID	Operations
basic-services	read

The **show processes** command with the *process-name* argument displays detailed information about a process:

```
RP/0/RSP0/CPU0:router# show processes ospf

Tue Jul 28 09:23:17.212 DST
                  Job Id: 338
                     PID: 336152
         Executable path: /disk0/asr9k-rout-3.9.0.14I/bin/ospf
              Instance #: 1
              Version ID: 00.00.0000
                 Respawn: ON
           Respawn count: 1
  Max. spawns per minute: 12
            Last started: Tue Jul 14 15:26:26 2009
           Process state: Run
```

```
           Package state: Normal
       Started on config: cfg/gl/ipv4-ospf/proc/100/ord_z/config
                    core: MAINMEM
               Max. core: 0
               Placement: Placeable
            startup_path: /pkg/startup/ospf.startup
                   Ready: 1.312s
               Available: 1.334s
        Process cpu time: 93.382 user, 13.902 kernel, 107.284 total
JID   TID CPU Stack pri state        TimeInState    HR:MM:SS:MSEC   NAME
338    1    0  116K  10 Receive        0:00:00:0375    0:00:47:0139 ospf
338    2    0  116K  10 Receive        0:00:05:0734    0:00:00:0029 ospf
338    3    1  116K  10 Receive        0:00:06:0765    0:00:00:0056 ospf
338    4    1  116K  10 Receive        0:00:00:0096    0:00:00:0698 ospf
338    5    1  116K  10 Receive        0:49:33:0609    0:00:00:0129 ospf
338    6    1  116K  10 Sigwaitinfo  329:56:49:0531    0:00:00:0000 ospf
338    7    0  116K  10 Receive        0:00:00:0816    0:00:58:0676 ospf
338    8    1  116K  10 Receive        0:00:06:0765    0:00:00:0043 ospf
338    9    1  116K  10 Condvar       82:30:01:0311    0:00:00:0029 ospf
338    10   1  116K  10 Receive       82:30:05:0188    0:00:00:0478 ospf
338    11   0  116K  10 Receive      329:54:49:0318    0:00:00:0005 ospf
------------------------------------------------------------------------------
```

Table 7: show processes Field Descriptions

Field	Description
Job id	Job ID. This field remains constant over process restarts.
PID	Process ID. This field changes when process is restarted.
Executable path	Path for the process executable.
Instance	There may be more than one instance of a process running at a given time (each instance may have more than one thread).
Version ID	API version.
Respawn	ON or OFF. The field indicates if this process restarts automatically in case of failure.
Respawn count	Number of times this process has been started or restarted (that is, the first start makes this count 1).
Max. spawns per minute	Number of respawns not to be exceeded in 1 minute. If this number is exceeded, the process stops restarting.
Last started	Date and time the process was last started.
Process state	Current state of the process.
Started on config	Configuration command that started (or would start) this process.
core	Memory segments to include in core file.
Max. core	Number of times to dump a core file. 0 = infinity.

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

```
RP/0/RP0/CPU0:router# show processes memory

  JID    Text     Data     Stack    Dynamic  Process
  55     28672    4096     69632    17072128 eth_server
  317    167936   4096     45056    10526720 syslogd
  122    512000   4096     77824    9797632  bgp
  265    57344    4096     57344    5877760  parser_server
  254    40960    4096     143360   3084288  netio
  63     8192     4096     24576    2314240  nvram
  314    4096     4096     36864    1699840  sysdb_svr_local
  341    495616   4096     40960    1576960  wdsysmon
  259    53248    4096     28672    1490944  nvgen_server
  189    32768    4096     32768    1425408  hd_drv
  69     77824    4096     110592   1421312  qnet
  348    323584   4096     40960    1392640  ospf
  347    323584   4096     40960    1392640  ospf
  346    323584   4096     40960    1392640  ospf
  345    323584   4096     40960    1392640  ospf
  344    323584   4096     40960    1392640  ospf
  261    323584   4096     40960    1392640  ospf
   --More--
```

Table 8: show processes memory Field Descriptions

Field	Description
JID	Job ID.
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.

The **show processes** command with the **all** keyword displays summary information for all processes, as shown in the following example:

```
RP/0/RP0/CPU0:router# show processes all

  JID    LAST STARTED            STATE    RE-      PLACE-  MANDA-  MAINT- NAME(IID) ARGS
                                          START    MENT    TORY    MODE
  ------------------------------------------------------------------------------------
  82     03/16/2007 14:54:52.488 Run      1               M       Y      wd-mbi(1)
  58     03/16/2007 14:54:52.488 Run      1               M       Y      dllmgr(1)-r 60 -u
30
  74     03/16/2007 14:54:52.488 Run      1               M       Y      pkgfs(1)
  57     03/16/2007 14:54:52.488 Run      1                       Y      devc-conaux(1) -h
-d
                                                                          librs232.dll -m
                                                                          libconaux.dll -u
                                                                          libst16550.dll
```

```
  76     03/16/2007 14:54:52.488 Run      1                    Y      devc-pty(1) -n 32
  56     Not configured          None     0                    Y      clock_chip(1) -r
-b
   --More--
```

Table 9: show processes all Field Description

Field	Description
JID	Job ID.
Last Started	Date when the process was last started.
State	State of the process.
Restart	Number of times the process has restarted since the node was booted. If a node is reloaded, the restart count for all processes is reset. Normally, this value is 1, because usually processes do not restart. However, if you restart a process using the **process restart** command, the restart count for the process increases by one.
Placement	Indicates whether the process is a placeable process or not. Most processes are not placeable, so the value is blank. ISIS, OSPF, and BGP are examples of placeable processes.
Mandatory	M indicates that the process is mandatory. A mandatory process must be running. If a mandatory process cannot be started (for example, sysmgr starts it but it keeps crashing), after five attempts the sysmgr causes the node to reload in an attempt to correct the problem. A node cannot function properly if a mandatory process is not running.
Maint Mode	Indicates processes that should be running when a node is in maintenance mode. Maintenance mode is intended to run as few processes as possible to perform diagnostics on a card when a problem is suspected. However, even the diagnostics require some services running.
Name (IID)	Name of the process followed by the instance ID. A process can have multiple instances running, so the IID is the instance ID.
Args	Command-line arguments to the process.