

# **HP-UX 11i Version 1.5 System Crash Dump White Paper**

**April 27, 2001**

© Copyright 2001 © Hewlett-Packard Company, All rights reserved..

# 1 **HP-UX 11i Version 1.5 System Crash Dump**

Beginning with the 11.0 release of HP-UX, system crash dump processing (configuration, generation, and use) has been totally revamped. The current functionality is described in this document, in the following sections.

- ❑ Changes for HP-UX 11i Version 1.5
- ❑ Overview of System Crash Dump
- ❑ Selective Dumps
- ❑ Crash Dump Configuration
- ❑ Dump-Time User Interface
- ❑ Post-Reboot Dump Processing
- ❑ Debugging of Crash Dumps

## Changes for HP-UX 11i Version 1.5

The major changes to system crash dump functionality in HP-UX 11i Version 1.5 are largely internal and do not affect the user-level interface:

- ❑ The boot loader passes user boot arguments through, unchanged, to the kernel, where they are parsed. Accordingly, the dump/crash path handles user boot arguments affecting dump device configuration, initializes, at boot-time, certain dump-related data structures, allows for user `config(1M)`-time and run-time configuration of dump devices and selection of memory to be dumped, and explicitly passes any information necessary for the taking of a crash dump to the dump application.
- ❑ Dump support is currently available for the Qlogic and LSI SCSI controllers only.
- ❑ There is currently no dump support for:
  - IDE (you can't dump to the CD read/write device or the LS-120 floppy device)
  - Fibre Channel
  - dump-to-tape

## Overview of System Crash Dump

The dump and reboot process is significantly faster than in previous releases of HP-UX, and can handle technologies such as more than 4GB of physical memory, dynamically loaded kernel modules (DLKM), non-contiguous physical memory, and the like.

System crash dumps do not contain the entire contents of physical memory by default. With memory sizes growing in leaps and bounds, it is critical that only those parts of physical memory which are considered useful in debugging a problem are dumped. For a discussion of which parts of memory are dumped, see “Selective Dumps” on page 12.

To simplify access of crash dumps by commands and utilities, HP-UX provides a crash dump access library, `libcrash`, which can be used to access any past-, current-, or future-format dump. For more information, see “Debugging of Crash Dumps” on page 18.

The system crash dump user interface affords an operator sitting at the console a limited degree of control of the dump process, plus a great deal more information about its progress while dumping than was available in the past. For more information, see “Dump-Time User Interface” on page 16.

Aside from speeding up the dump process itself, HP-UX speeds up the process of saving the dump into the file system after the system has rebooted. Proper use of the `savecrash(1M)` and `crashutil(1M)` utilities, combined with proper configuration of dump devices, can greatly reduce or nearly eliminate the time that used to be spent running `savecore` (now obsolete) in prior releases. For more information, see “Post-Reboot Dump Processing” on page 17.

It is not necessary to rebuild the kernel in order to configure dump devices: there are several ways of configuring dump devices. Also described here are the ways to override the defaults for which parts of memory get included in the dump. For more information, see “Crash Dump Configuration” on page 13.

## Selective Dumps

In order to dump only those parts of memory that are most useful for debugging a problem, HP-UX divides the pages of memory into several classes. The rightmost column in this table shows which classes of memory are included in a crash dump under most circumstances. However, these defaults can be overridden by the system administrator (see “Crash Dump Configuration” on page 13). Also, there are some rare circumstances in which only full dumps are possible.

**Table 1-1**

<b>Name</b>	<b>Description</b>	<b>Included?</b>
UNUSED	Pages not currently in use.	No
USERPG	User-space pages, except stacks.	No
USTACK	User process stacks.	Yes
KCODE	Kernel code.	No
KSDATA	Kernel static data.	Yes
KDDATA	Kernel dynamic data, except buffer cache data.	Yes
BCACHE	Buffer cache data, except buffers containing file system metadata.	No
FSDATA	Buffer cache buffers containing file system metadata (superblocks, indirect blocks, and cylinder groups).	Yes

## Crash Dump Configuration

In the HP-UX operating system prior to release 11.0, the only configuration option was the ability to specify which devices would be used to store a crash dump. This option was set during kernel build and only took effect after rebooting the system on the new kernel

Now, by default, the system dumps to the swap partition. If the root disk is VxFS, the system dumps to a swap partition on the root disk; for example, `/dev/dsk/cntndn`. During runtime the default location can be changed to dump to another (secondary) disk, but the following limitations currently apply if you do that:

- ❑ The default location cannot thereafter be restored during runtime. To restore the default location, the system must be rebooted.
- ❑ Any previous contents of the secondary disk will be lost because a dump will overwrite it from block 0.

In addition to the kernel and logical volume device crash dump configurations, there are two other methods and some additional options.

- ❑ Crash dump configuration can be done during system initialization (see `rc(1M)` and `rc.config.d(4)` man pages)
- ❑ Crash dump configuration can be done during run time directly through the `crashconf(1M)` command. A kernel reboot is not required.

Configuration of dump devices is modeled on configuration of swap devices Consider this comparison between them:

**Table 1-2**

Swap Devices	Dump Devices
A system call, <code>swapon(2)</code> configures swap devices.	A system call, <code>crashconf(2)</code> , configures dump devices.
A command, <code>swapon(1M)</code> , can configure a swap device named on its command line.	A command, <code>crashconf(1M)</code> , can configure one or more dump devices named on its command line.

**Table 1-2**

<b>Swap Devices</b>	<b>Dump Devices</b>
swapon(1M) can also read /etc/fstab and configure all devices marked as “swap”.	crashconf(1M) can also read /etc/fstab and configure all devices marked “dump”.
SAM can be used to configure swap devices.	SAM can be used to configure swap devices.
The primary swap device can also be configured into the kernel at kernel build time.	Dump devices can also be configured into the kernel at kernel build time, in the exact same manner as before.

An important feature listed in the table above is the ability to read /etc/fstab and configure all devices marked “dump”.

In addition to configuring dump devices, `crashconf` (both the system call and the command) can also be used to configure the classes of memory that will be included in, or excluded from, the dump. If desired, this information can also be configured at kernel build time, by setting the bitmask tunables `alwaydump` and `dontdump`. Refer to the `/usr/include/sys/crashconf.h` file or the output of the `crashconf(1M)` command for the current list of memory classes.

In most circumstances, it should not be necessary to modify the list of individual memory classes to be dumped. The default settings provide adequate information for debugging nearly all system panics while eliminating as much extraneous data from the dump as possible.

In actual practice, if crash dump is reconfigured at all, it is usually done to force a full dump, or to disable dumps entirely.

Full dumps are enabled by including all pages in the dump. This is done with “`crashconf -i all`” or by setting `CRASH_INCLUDED_PAGES="all"` in `/etc/rc.config.d/crashconf`.

Dumps are disabled by excluding all pages from the dump. Again, this is done with “`crashconf -e all`” or by setting `CRASH_EXCLUDED_PAGES="all"` in `/etc/rc.config.d/crashconf`.

Finally, `crashconf(1M)` uses the `pstat` interfaces `pstat_getcrashinfo(2)` and `pstat_getcrashdev(2)` to provide a report of which devices are configured, which memory classes will be included,

and how much space a selective dump will take. This information can be used to determine the amount of dump space that should normally be configured. Specifying “-v ” on the `crashconf(1M)` command line will display the current crash dump configuration to verify that the expected changes have taken effect.

## How Much Dump Space Do I Need?

For selective dumps, use `crashconf(1M)` to find out how much space would be needed for a selective dump of your machine. (Use the `-v` flag, and run it while your system is under its normal or higher than normal workload.) The space needed will vary depending on the workload of the machine, so add another 25% or so to be safe. The total dump space should meet or exceed this amount.

For machines that are relatively stable and don't expect to dump often, this is enough. If a full dump is needed from such a machine, additional space can be configured for it on the fly, anyway. This space can also be used for swap unless reboot times are critical.

For machines on which full dumps are required, the full size of physical memory, plus a little bit for dump headers and tables, should be configured as dump space. At least the amount needed for a selective dump should be configured on a device that is not used for swap activity.

For example, the size of a selective dump for a 1GB RAM system typically ranges from 100MB to 200MB (10% to 20%). Extrapolating this to a 64GB system, a dump could be as large as 12GB.

Whenever you have dump devices that are not also used for swap activity, make sure that they are configured last. This will cause them to be used first (dump goes from the end backward), which will minimize the chance of writing into an area shared by swap. Writing into swap space is undesirable because it will slow down your reboot processing; see “Post-Reboot Dump Processing” on page 17, for details.

## Dump-Time User Interface

As mentioned above, an operator sitting at a system's console when it dumps has some control over the process of dumping. The interface is somewhat similar to the boot-time user interface of most HP-UX systems: the dump path prints out what it expects to do, and gives the operator ten seconds to hit a key to indicate that a change from the defaults is desired.

The options allowed depend on the circumstances, but may include:

- Full dump
- Selective dump
- Partial dump (this is the same as a full dump, except that if there is insufficient space for the entire dump, priority is given to pages which are considered most useful)
- No dump at all

Once the dump has begun, in the fashion you choose, continuous status messages on the console track its progress. The dump can be aborted midstream by pressing **ESC**.

## Post-Reboot Dump Processing

Prior to HP-UX 11.0, one of the first user space tasks in the boot process was running `savecore` (now obsolete), which would check for a crash dump and, if found, copy it from the dump device(s) to the file system. This same program had many other features used by system administrators after the system was up.

In HP-UX release 11.0 and later releases, `savecrash(1M)` and `crashutil(1M)` handle boot-time tasks and post-boot tasks, respectively.

The `savecrash(1M)` command runs at boot time; it preserves any and all crash-related information which could be lost as system activity continues. It has many options, which may be set in `/etc/rc.config.d/savecrash`. One thing worth noting is that `savecrash` can be told to save only that portion of the crash dump endangered by swap activity, rather than all of it (the default). This is useful for debugging on the system that crashed: the debuggers can debug straight out of the dump devices. (See “Debugging of Crash Dumps” on page 18.) If care has been taken to configure dump devices that are not shared by swap activity (see “Crash Dump Configuration” on page 13), this means that virtually all of the time it would normally take to copy the dump onto the file system can be eliminated.

The `crashutil(1M)` command handles all of the post-boot crash dump manipulation. Currently it has two main functions. First, it completes saving those portions of the dump that reside on non-swap dump devices. The `crashutil` command copies the remaining portions of the dump onto file system so that it can be transported to another system for debugging. Second, it converts dumps from one dump format to another. For example, an 11.0 dump can be converted to an older format so that old debugging tools can access the dump.

Note: The concept of a special tape format for dumps has gone away. If dumps are needed on tape:

- run `crashutil(1M)` (to make sure that parts of it aren't still residing on dump devices)
- save the dump from the dump device to a directory on a file system (typically `/var/adm/crash`)
- use `tar` or any similar command to package and ship the crash dump directory

## Debugging of Crash Dumps

The `libcrash` library provides debugger owners and others with code that reads crash dumps. This library insulates the debuggers from the details of the crash dump format, allowing them to handle any type of crash dump with equal ease. The main kernel crash dump debuggers, `q4` and `adb(1)`, use this library.

As mentioned under “Post-Reboot Dump Processing” on page 17, it is no longer necessary to save the crash dump from the dump device into the file system before it can be debugged. `savecrash(1M)` must be run -- it has to create a directory for the dump and an INDEX file, at minimum -- but it can be told not to copy the dump data out of the dump devices.