Sun Enterprise™ xx00 Problem Solving Manual

Error Recognition, Actions, and Reporting Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 U.S.A.
650-960-1300

Part No. 910-4188-11
February 2002, Revision A

Send comments about this document to: Martin.Canoy@Sun.COM

Please
Recycle

Adobe PostScript

Sun Proprietary/Confidential: Internal Use Only

Contents

CHAPTER 1

Sun Enterprise xx00 Problem Solving

1.1 Overview

Sun Microsystems offers the Sun Enterprise xx00 Server product line - a reliable, adaptable, maintainable, and performance-oriented product.

Sun Microsystems continually strives to produce robust reliable products. It is also Sun's objective to move quickly on any issues and to apply the necessary product expertise to resolve the issues in a timely manner. To this end, Sun Microsystems has a world-class support organization, Sun Enterprise Service, which is designed to facilitate the support of the entire Sun product line. If a problem with the Sun Enterprise Server occurs, the first step is to identify that a problem has occurred and then to notify Sun using the recommended service call reporting processes for your specific situation.

1.2 System Problem Categories

The root cause of a problem can vary. Examples include unforeseen hardware errors/failures, Solaris specific bugs, and third-party software problems.

The types of problems that one might encounter can be grouped into six main categories:

- **System Panics**: A system panic (crash) dumps specific areas (but not all) of active memory so that a system dump (vmcore) file can be created. This vmcore file is then analyzed to determine if the cause of the panic was software or hardware.
- **System Soft Hangs**: This type of hang is characterized by a non-responsive user-interface. The affected system should still respond to network activity, such as the ping command from a remote system.

  The affected system should also respond to a system abort sequence. Refer to Appendix C for detailed information on abort sequences.

  The system LEDs should still be in their normal flashing/cycling state.
- **System Hard Hangs**: This type of hang is also characterized by a non-responsive user-interface.

  Unlike the soft hang, all attempts to interrupt the system will fail. All network activity is non-responsive. The system LEDs may also be in a frozen state.
- **Hardware Fatal Resets** This is a result of an ‘‘illegal’’ hardware state being detected.

  A hardware fatal error can either be a transient error or a hard error. A transient error is one that intermittently fails. A hard error is one that consistently fails in the same way.
- **System Drop into OpenBoot (OBP)**: This type of interrupt is typically characterized by an unresponsive system, and when console access is obtained, the system is found at the OpenBoot or "ok" prompt.

  Certain failure modes, such as Solaris stack overflow errors, can cause exception conditions known as redstate conditions. This condition will cause Solaris to abort in such a way that OpenBoot will not automatically react and reboot the system.

  In some cases, a break signal may have been issued, by someone with console access or by a spurious signal from a serial terminal server. In these cases the system has not actually had an internal hardware or software problem, but was simply halted.

Many times a system will be said to have experienced an unexpected system reboot. Care must be taken to verify that this condition was not caused by a panic or Fatal Reset.

Although the system appears to unexpectedly reboot, in many cases it is due to a panic or Fatal Reset. Thus the need to be specific when determining why a system interrupted.

E-Cache parity errors that occur in user space (not kernel space) may cause Solaris to simply log the E-Cache failure data and reboot the system. The reason is that the error information that is logged is sufficient to troubleshoot and correct the problem. This reboot saves the time of taking a system panic dump. It also by-passes the need to run fsck(1M) on file systems as they are un-mounted during the reboot process.

This new reboot feature for user level E-Cache parity errors is a function of changes made in kernel update patches containing the kernel level scrubber for Solaris 2.6 and above. The KU patches are 105181-23 and higher for Solaris 2.6, 106541-13 and higher for Solaris 7 and 108528-04 and higher for Solaris 8.

CHAPTER 2

Failure Detection and Data Retrieval Tools

The Sun Enterprise xx00 Server product line ships with a number of software and firmware tools used for generating, gathering, and analyzing data. These tools include:

Firmware

- Power On Self Test (POST)
- OpenBoot (OBP)
- NVRAM Variables
- Standalone Solaris Debugger
- kadb
- Solaris Debuggers and Commands
- prtdiag
- prtconf
- adb
- crash

Solaris Messages

- Console Access/Output
- Solaris log files (/var/adm/messages)
- XIR, Key Switch, Solaris system logs, Sun Management Center

2.1 Power-on Self-Test (POST)

POST has two main objectives: to test the system hardware (excluding peripheral devices) to help diagnose a failing hardware component; and to configure the available hardware for use by OpenBoot (OBP) and then Solaris.

POST resides in the OpenBoot (OBP) PROM on each CPU/Memory board, I/O board, and disk board. POST controls the status LEDs on the system front panel and all boards and displays diagnostic and error messages on a console terminal (ttya serial port), if available.

POST is sometimes referred to as an “off-line” diagnostic tool which can be used to verify the integrity of the hardware on the system. “Off-line” is defined under the condition where Solaris is not running and POST is executing by itself.

Only POST can configure the system hardware, and only POST can enable hot-pluggable boards. If a new unit (board or modular power supply) is added to the card cage after the system has booted, the new unit will not work until the system is rebooted, at which time POST reconfigures the system, using the units that are found in the system at that time.

To capture the results of POST, an active console session is required. It is preferred to have a console connection via the serial ttya port (serial port A) to the system. This connection should be capable of logging the output via a large scroll buffer in the terminal window being used or preferably via logging the console session to a file using a tool such as "script".

POST can be invoked by one of the following methods:

- Issuing the "reset-por" command at the OBP prompt:

  Issuing the "reset-por" command at the OBP prompt causes POST to be entered and executed. A physical power-cycle of the system will also cause full POST to be run.

- Key switch in Diagnostic position

  There is a manual key switch interface to POST located on the front panel of the Sun Enterprise xx00 Server. If the key switch is turned to the "Diagnostic" position, each time the system is rebooted, POST will be executed at the maximum level.

- Setting the OBP value, diag-switch?, to TRUE

  The diag-switch? OBP variable is FALSE by default. Setting diag-switch? To TRUE will cause POST to be run at the level defined by the OBP variable diag-level during the next system reboot.

There are basically three POST levels that can be defined by the OBP diag-level variable. The levels are off, min, and max. The max level provides the most comprehensive set of POST tests.

The key switch position in Diagnostic mode, as described above, takes precedence over these OBP variables.

- System encounters a Fatal Reset (hardware exception)

  If the system detects an unrecoverable hardware exception, a Fatal Reset will result. The system stops executing Solaris immediately and does not log any error or failure information to the Solaris log files, i.e. /var/adm/messages. POST is invoked as part of the system recovery (reboot) process.

  Hardware components that fail POST will be marked as failed and will not be used during this boot sequence. These components will remain disabled across multiple system reboots. However, if the system is power cycled, the POST failures are cleared. If these components fail POST again, they are again disabled. If these components pass POST, they are then enabled and used in the system configuration.

The list of failed components will persist over time and as such new failures will accumulate after hardware errors. Failed component information will be cleared after either of the following events:

- The “reset-por” command from OBP (ok prompt) is issued.
- A physical power-cycle of the system is performed. This power cycle normally occurs during hardware replacement activities.

Note – IMPORTANT! Based on the information above, it is critical that POST output (console output) be logged so that it can be analyzed further as necessary. It is critical to use the `prtdiag -v` command PRIOR to any system power cycle in order to capture detailed failure information that is stored in EPROM.

2.2 OpenBoot (OBP)

The OpenBoot (OBP) firmware is stored in the boot PROM (programmable read-only memory) of a system so that it is executed immediately after you turn on your system. The primary task of the OpenBoot firmware is to boot the operating system from either a mass storage device or from a network. The firmware also provides extensive features for testing hardware and software interactively.

- OBP Diagnostic and Information Commands on Table 2-1 show commands that are available at the OBP (ok) prompt and allow the user the ability to perform some low level tests of installed hardware.

  These on-board tests allows the user to check devices such as the network controller, the floppy disk system, memory, installed SBus cards and SCSI devices, and the system clock. User-installed devices can be tested if these devices contain firmware which includes a self-test feature.

TABLE 2-1 OBP Diagnostic/Information Commands

OBP Command	Description
`probe-scsi`	Identify devices attached to the built-in SCSI bus.
`probe-scsi`	Perform probe-scsi on all SCSI buses installed in the system below the specified device tree node. If device-path is absent, the root node is used.
`test net`	Test the network connection
`test floppy`	Test the floppy drive, if installed
`test memory`	Test number of megabytes specified in the selftest-#megs NVRAM parameter; or test all of memory if diag-switch? is true
`test-all [device-specifier]`	
`watch-net`	Monitor the network connection.
`watch-clock`	Test the clock function.

2.3 OBP Debug Command

OBP Debug Commands on TABLE 2-2 show commands that are available at the OBP (ok) prompt. They allow the user the ability to perform some low level data gathering.

These commands are normally executed when a system was running Solaris and then hung and was forced (aborted) into OBP, or the system panic'd and the panic dump process failed.

TABLE 2-2 OBP Debug Commands

OBP Command	Description
`.registers`	Display values in %g0 through %g7, plus %pc, %npc, %psr, %y, %wim, %tbr.
`.trap-registers`	Display values in the trap related registers.
`.locals`	Display the values in the i, l and o registers.
`.psr`	Display the processor status register.
`ctrace`	Display the Solaris return stack showing C subroutines.
`sync`	Cause Solaris to attempt to save a system panic (vmcore).

2.4 kadb Command

kadb is a low-level kernel debug tool that is available from the OBP level. In normal operation, kadb is not enabled because it is not normally required. To enable kadb, Solaris must be booted with the kadb option. The following command is an interactive method to enable kadb for a single boot.

```
ok> boot kadb
```

If the system is rebooted or interrupts (panic, Fatal Reset, etc.), the system will not boot in kadb mode. You can set the boot-file OBP variable to enable kadb boots.

```
ok> setenv boot-file kadb
ok> boot kadb
```

Note – IMPORTANT! Booting the system in kadb mode should ONLY be set based on specific direction from Sun Microsystems support engineers. Booting kadb mode should NOT be default on all systems.

Once the system is booted in kadb mode, the user can then elect to enter into kadb at the time of the issue / problem by sending a break signal via the console connection.

However, if the nature of the error is preventing the system from correctly saving a system panic dump (vmcore file), booting kadb can be useful. Once the system panics, the system will drop into kadb and once inside kadb, the user can then attempt to analyze the systems behavior further.

An excellent reference book on panics and the use of kadb to analyze them is, *Panic, Unix System Crash Dump Analysis* - by C. Drake and K. Brown.

For a detailed explanation on the command, refer to the man pages for kadb(1M).

2.5 Solaris System Commands and Data Files

The operating system (Solaris 2.5.1 is the initial version for the Sun Enterprise xx00 Server product lines) contains several very useful data generating / gathering commands. Solaris also maintains data files containing useful information.

2.5.1 `prtdiag` Command

`/usr/platform/sun4u/sbin/prtdiag`–This command displays both the system configuration and hardware error information.

The `prtdiag`(1M) command provides information on POST failures as well as Fatal Reset information. When a Sun Enterprise Server takes a fatal reset, relevant information is copied to a region of non-volatile memory. The `prtdiag` command accesses this region and translates the information into a readable format. This command is only available on the Sun4u and Sun4d architectures. The recommended command syntax is:

```
# /usr/platform/sun4u/sbin/prtdiag -v
```

For a detailed explanation of the command, refer to the man page for prtdiag(1M).

2.5.2 prtconf Command

/usr/sbin/prtconf–The output of the prtconf command is a subset of the output from prtdiag.

The prtconf command can be useful for other detailed hardware configuration information. For a detailed explanation of the command, refer to the man page for prtconf(1M).

The recommended command sequence takes the form:

```
# /usr/sbin/prtconf -pv
```

2.5.3 adb Command

/usr/bin/adb–The adb command is an on-line debugging utility that can be used to analyze the running image of Solaris or a system panic dump (vmcore file).

The adb command to investigate a system panic dump (vmcore file) is:

```
# cd /var/crash/‘uname -n‘

# /usr/bin/adb -k unix.? vmcore.?
```

Where "?" is the integer number for the dump being analyzed.

Analyzing a system panic dump using adb must be done on the same system architecture as the system that originally generated the panic. An excellent reference book which describes panics and the use of adb to analyze crash dumps is, *Panic, Unix System Crash Dump Analysis* - by C. Drake and K. Brown. For a detailed explanation on the command, refer to the man pages for adb(1M)

The adb command used to investigate the running Solaris image (as root) is:

```
# /usr/bin/adb -k
```

2.5.4 mdb Command

usr/bin/mdb–The Solaris Modular Debugger (mdb) is another on-line debugging utility that can be used to analyze the running image of Solaris or a system panic dump (vmcore file). The mdb debugger is only available in Solaris 8.

The syntax for invoking mdb is basically the same as for adb described above. Refer to the mdb man page and other mdb documentation for further information.

2.5.5 `crash` Command

`usr/sbin/crash`–The crash command is used to examine the system memory image of a running or a crashed system by formatting and printing control structures, tables, and other information.

The command to investigate a system panic dump (vmcore file) is:

```
# cd /var/crash/‘uname -n‘
# /usr/sbin/crash -n unix.? -d vmcore.?
```

Where "?" is the integer number for the dump being analyzed.

Analyzing a system panic dump using crash must be done on the same system architecture as the system that originally generated the panic. For a detailed explanation on the command, refer to the man pages for crash(1M)

The crash command used to investigate the running Solaris image (as root) is:

```
# /usr/sbin/crash -n /dev/ksyms -d /dev/mem
```

2.5.6 Solaris Log Files

/var/adm/messages–This is an ASCII text data file that is the standard repository for all Solaris generated messages.

The messages file is archived over time and contains system warnings, errors, and notifications. Messages and errors from system memory and peripheral controllers and devices are what is primarily logged in these files.

The messages files also provides a time-stamped record of system events. It provides a timeline against which system behavior can be compared for corroborating or supporting evidence.

CHAPTER 3

Failure Scenarios

The following sections describes system failure scenarios and the information that is required to perform the most complete and timely diagnosis of the system interrupt, based on each type of failure scenario.

In many cases, the Sun Solution Center may ask that the system data gathering tool known as "Explorer" be run. This will collect some of the information below, as well as a number of other data components.

3.1 System Panic

For more information on system panics, refer to, *Panic!: Unix System Crash Dump Analysis* by C. Drake and K. Brown.

Commands and/or different command combinations can and should be tailored to the type of system exception being investigated. The output from the following list of commands will provide an initial picture of what happened on the Sun Enterprise Server xx00 in question.

IMPORTANT! It is also critical to inspect the `/var/adm/`messages file for relevant information.

The system reaction to a Solaris panic varies based on the following:

- If the system is booted normally, and the OBP variable auto-boot? is set to "true", the system should attempt to save a system dump (vmcore file) and reboot to multi-user mode automatically.

  Solaris commands that should be issued after the system has rebooted from the panic are:

  ```
  # /usr/platform/sun4u/sbin/prtdiag -v
  # /usr/sbin/prtconf -pv
  ```

```
# cd /var/crash/‘uname -n‘
# iscda unix.? vmcore.? > iscda.out
```

Where "?" is the integer number for the dump being analyzed.

Note – iscda is not part of the standard Solaris release, refer to Appendix B.

- If the system is booted normally, and the OBP variable auto-boot? is set to "false", the system should attempt to save a system dump (vmcore file). However, the system will reboot and stop at the OBP (ok) prompt.

  If the system saved a valid vmcore file, then the system should be booted immediately so that the system dump (vmcore file) and other system information can be saved and analyzed for root cause. The Solaris commands above should be run once the system is booted.

- If the system fails to save a system dump, and the system drops into OpenBoot (OBP), the following OBP commands would be run and the output logged/captured:

```
ok> printenv
ok> .registers
ok> .locals
ok> .psr
ok> .trap-registers
ok> ctrace
ok> sync
```

Note – IMPORTANT! If the system saved a valid dump or attempted to and failed, the sync command below may fail. The sync command is normally used to save a vmcore dump after the system has dropped from Solaris to OBP.

- If the system was last booted under kadb, a system panic should drop the system into kadb. If booted under kadb, execute the following kadb commands and record all the output.

```
<sp$<stacktrace
    $<threadlist
    $<thread.brief
    $<cpus
    $<msgbuf
```

3.2 System Soft Hang

A general definition of a system soft hang is when the usability of the system gradually or suddenly ceases. In some cases, soft hangs have been known to only affect new attempts to access the system. A gradual onset is usually characterized by progressively slower response until the system stops responding. A sudden onset is when all or some of system response suddenly ceases.

Some soft hangs will dissipate on their own while others will require that the system be interrupted, that information be gathered at the OpenBoot level, and the system be rebooted. A soft hang should respond to a break signal that is sent via the system console.

The possible causes of a system hang include memory leaks, software (most likely kernel, I/O related or device driver problems), network problems, and even possibly defective hardware.

In the case of a soft hang, try to determine the extent of the problem by doing the following:

1. **Record the state of the LEDs on the front of the system, if possible.**

   Note any amber LEDs and verify that the cycle LED (green flashing LED) is flashing on all System Boards, Clock Board and Main LED on the system front panel.

2. **Determine if any network activity is working (i.e. via ping, etc.) and if any existing logins from other users are active/responding.**

   If other active logins are responding, determine the state of the network interface and review the contents of /var/adm/messages for any indications of problems. Use commands such as `ifconfig`, `netstat`, `ping`, etc. to determine the state of the network and to attempt to make network contact with other systems.

3. **Determine if a console logging session to the system can be made through the console (ttya) connection.**

   If a working console connection can be established, then the problem may not be a true hang, but instead a network related problem. This network problem could be system hardware (network controller) or possibly a hardware problem in the network infrastructure (network hub/router/cable).

   For suspected network problems, attempt to ping, rlogin or telnet to the affected system. If it's not possible, attempt to ping, rlogin or telnet to another system that is on the same sub-network/hub/router that the affected system is on.

   If NFS services are served by the affected system, determine if NFS activity is sluggish/non-existent on other systems.

4. **If there are no responding login sessions, and a console login session is unable to be made, attempt to break the system so that it will fall into OpenBoot (OBP) mode.**

   If you manage to drop into OBP or kadb (depending upon how the system is booted) then the hang is classified as a soft hang. The output from the following list of commands will provide an initial picture of what happened on the Sun Enterprise xx00 server in question.

   For systems that are booted normally (without kadb), the following are useful commands:

   ok> **printenv**

   ok> **.registers**

   ok> **.locals**

   ok> **.psr**

   ok> **.trap-registers**

   ok> **ctrace**

   ok> **sync**

   For systems booted under kadb, the following are useful commands:

   <**sp**$<**stacktrace**

   $<**threadlist**

   $<**thread.brief**

   $<**cpus**

   $<**msgbuf**

   Some decisions can then be made on the contents of the stack trace. For example, the length of the stack trace and certain key routine names can indicate a rather normal stack or an abnormal stack.

   It is good practice to inspect the `/var/adm/messages` file for the following information:

   - A large gap in the time stamp of Solaris/application messages
   - Indications of last root logins to determine if any administrators can add any comments about the system state at the time of the hang
   - Warning messages about any hardware or software components

3.3 System Hard Hang

A system hard hang differs from a soft hang in one key area—a hard hang will not respond to a system break sequence. For this reason, it can be difficult to determine the root cause of a hard hang after a single occurrence.

Note – It is important to note the state of the system LEDs (i.e. are they cycling or frozen / on solid). The state of the LEDs should be determined prior to initiating any of the following system recovery operations.

In the case of a hard hang, try to determine the extent of the problem by doing the following:

1. **Record the state of the LEDs on the front of the system, if possible.**

   Note any amber LEDs and also verify that the cycle LED (green flashing LED) is flashing on all System Boards, Clock Board and Main LED on the system front panel.

2. **Determine if there are any active network connections that are still responding.**

   If no active connections respond, a new connection should be attempted using both the console and the network. This includes attempting to ping the effected system. If any login connection is successful, refer to the commands for the soft hang above to attempt to recover the system and / or force a system panic.

3. **If no active connections exist or can be made, attempt to break the system so that it will drop into OBP or kadb, depending on how the system was last booted.**

   If the system is able to drop into OBP or kadb, refer to the appropriate command sequences (in the soft hang situation) for each of these modes.

   If the system is still not responding to the above access methods, the choices are limited. Solaris may or may not be executing at this time. Due to the inability of aborting to OBP / kadb, the user is left with using the XIR button at the rear of the system. If that is unsuccessful in interrupting the system, attempt power-cycling the system via the front panel key (all the way to the left is power off).

4. **Attempt the XIR button. The XIR button should be tried first because it does not power cycle the system, where a transient error may be reset/masked.**

   The XIR button is equivalent to issuing the reset command while in OBP. Note that the XIR button may or may not cause the system to respond. This is dependent upon the severity of the hard hang. If XIR is recognized, some system information is saved.

   To display the XIR information, execute the following command:

   ```
   .xir-state-all
   ```

XIR failure data can then be sent to Sun and can be diagnosed at the following internal Sun web site:

```
http://cte-www.uk/cgi-bin/xir-cgi.tcl
```

There is a key-sequence that can be used to try and initiate an XIR state. This key sequence is two carriage returns, a tilde and the key sequence of the Control Key/Shift Key/ and the letter x. For example:

<CR> <CR> <~> <Control-Shift-x>

To execute an XIR state using the key-sequence, the following criteria must be met:

- The console must be connected to port A on the clock board.
- The key switch must be in the On or Diagnostic setting. If it is in the Secure or Off position, the remote key sequences and button resets are ignored.
- Security features (such as OpenBoot security-mode) must be disabled.
- The type speed must be no faster than 0.5 seconds and no slower than 5 seconds between characters.

5. **If the XIR button does not reset the system, power cycle the system to reset it and begin the reboot sequence.**

Note – Record the state of the system LEDs prior to powering the system off and on.

Use the key switch to power off the system. It is recommended to wait at least 30 seconds before powering the system on with the key switch.

It is good practice to inspect the /var/adm/messages file for the following information:

- relevant data such as a large gap in the time stamp of Solaris/application messages
- indications of last root logins to determine if any administrators can add any comments about the system state at the time of the hang
- warning messages about any hardware or software components

If all attempts to gain control of the system or to gain additional information fails, and the system must be power cycled to reset it, the Solaris deadman timer is one method that will prevent all but a hardware failure from leading to a system hard hang.

Note – The Solaris deadman timer should ONLY be enabled based on specific instructions from Sun Microsystems. This mode should NOT be default on systems.

The deadman timer, based on the TOD chip, will hard reset the system if the timer expires. To enable it, the appropriate /etc/system entry must be made and the system rebooted for this to take effect.

set watchdog_enable=1

set snooping=1

By enabling the deadman timer, the system should drop into OBP, if the timer expires. This is an indication of a system hang and the appropriate OBP commands for a soft hang should be executed:

ok> **printenv**

ok> **.registers**

ok> **.locals**

ok> **.trap-registers**

ok> **.psr**

ok> **ctrace**

ok> **sync**

3.4 Fatal Reset/Fatal Error

A Fatal Reset/Fatal Error condition is most commonly a hardware fault that is detected by the system. This condition is un-recoverable and continued operation of Solaris would jeopardize the system because system integrity has been lost. Thus Solaris is immediately terminated and as such no details of the Fatal Reset are logged in common locations such as /var/adm/messages, etc.

There are a few cases of software (typically device driver problems) causing Fatal Resets. These are rare and normally documented so that they can be identified easily. Some of these driver bugs manifest themselves as timeout problems (MTIMEOUT Fatal Resets). For past examples, refer to Sun Bug 4320047, 4306348 and 4230383. These bugs can be obtained via SunSolve Online (both internal or external versions of SunSolve) or by contacting Sun Microsystems.

When a Fatal Reset is detected, the system will reset and enter into POST at maximum diagnostic level (diag-level = max). Unfortunately, the most important data that specifically defines what caused the Fatal Reset is displayed only to the system console. If the system console is a frame buffer based monitor, the console output is typically lost. This is also true if the console connection is via the ttya serial port and the output is not being logged. The result is that important data from the Fatal Reset, including the results of POST testing during the system recovery process, is lost.

In the case of an intermittent error which caused the Fatal Reset, POST might not find the offending component. In other cases of hard failure components, POST will detect them, mark them as failed, and continue with the rest of the POST tests. The service (yellow) LEDs both on the individual system board(s) that had the error and the main service (yellow) LED on the front panel of the system should further define the problem area. Any subsequent system reboots will show that the system has ‘‘off-lined’’ the offending component, where possible.

CHAPTER 4

Fatal Reset Diagnosis

This section defines a variety of possible Fatal Resets and the recommended diagnosis for each type of error. Most of the following information is based on the data obtained from the console at the time of the Fatal Reset. Where possible, output from `prtdiag -v` is also shown.

4.1 Console Information

The single most valuable piece of information available from Fatal Resets is the system console output at the time of the error. Due to the processing of some Fatal Resets, the data in prtdiag may indicate components that have failed which are not actually the root cause of the Fatal Reset. These innocent components may be replaced, and the system may appear stable. This stability is not due to the hardware replacement. The problem that is still in the system is transient and there is only the illusion that the replacement hardware fixed the Fatal Reset.

IMPORTANT! Knowing the recent service history of systems which encounter Fatal Resets can be valuable information to have when making the diagnosis.

IMPORTANT! Use of the `prtdiag -v` command BEFORE any hardware is replaced and/or the system power cycled is a key data gathering process. Note that the -v or verbose option must be used so that all the critical hardware failure information is logged.

IMPORTANT! The FT_ARBERR and FT_SHERR error indicators are not always valid in identifying a failing FRU. See Special Fatal Reset Considerations below.

4.2 Special Fatal Reset Considerations

Each Address Controller (AC) in the Sun Enterprise xx00 servers contains groups of 16 control wires. Two of these groups are the Arbitration wires and Shared wires. An AC only drives one wire, the wire driven depends on the slot the board is in, for example the board in slot 0 drives FT_ARB[0], board in slot 15 drives FT_ARB[15]. Parity cannot be calculated on these wires, so the AC that drives the wire samples the value driven on the bus and compares it against the value that it drove.

During a Fatal Reset and a power cycle, the register in the AC that contains the slot number is cleared, then the slot number is loaded back into the register.

The resets are supposed to be synchronized, so that every AC sees reset at the same time. Unfortunately, a bug exists where the board that detects the fatal error gets reset two cycles ahead of the rest of the boards in the system. If the board that detects the fatal error is about to drive the arbitration wires, then it will drive FT_ARB[0], even if it is Board 2. Board 0 detects the value on the arbitration wires is not consistent with what it drove and sets the FT_ARBERR bit in the AC error register. The same is true for the shared wires. In this case, the FT_SHERR error bit is set in the AC error register.

The result of the bug mentioned above can be Fatal Resets that appear as follows:

```
Fatal Reset
0,0>FATAL ERROR
0,0>At time of error: System software was running.
0,0>Diagnosis: Board 0, centerplane pin, connector pin, AC
0,0>Diagnosis: Board 4, UPA PORT Device, AC
0,0>Log Date: Mar 23  2:58:10 GMT 2001
0,0>
0,0>RESET INFO for CPU/Memory  board in slot 0
0,0>AC ESR 00000400.00000000 FT_ARBERR
0,0>DC[0] 00
0,0>DC[1] 00
0,0>DC[2] 00
0,0>DC[3] 00
0,0>DC[4] 00
0,0>DC[5] 00
0,0>DC[6] 00
0,0>DC[7] 00
0,0>FHC  CSR 00050200 LOC_FATAL SYNC NOT_BRD_PRES
0,0>FHC RCSR 02000000  FATAL
0,0>
0,0>RESET INFO for CPU/Memory  board in slot 4
0,0>AC ESR 00000000.00600001 IPREP FERR UPA_A_ERR
0,0>DC[0] 00
0,0>DC[1] 00
0,0>DC[2] 00
0,0>DC[3] 00
0,0>DC[4] 00
0,0>DC[5] 00
0,0>DC[6] 00
0,0>DC[7] 00
0,0>FHC  CSR 00050030 LOC_FATAL SYNC BRD_LED_M BRD_LED_R
0,0>FHC RCSR 02000000  FATAL
0,0> Config policy change
0,0>
0,0>@(#) POST 3.9.28 2000/12/20 12:29
0,0>Copyright 2000 Sun Microsystems, Inc. All rights reserved.
0,0>
....
```

The problem is that the FT_ARBERR can be interpreted as being a failure and Board 0 would be replaced. This is not correct. The FT_ARBERR state is invalid. The root cause of this error is an Etag Parity Error on Board 4, CPU 0 (CPU 8).

As mentioned above, the FT_ARBERR can also be set to FT_SHERR. Either of these error modes should be discounted as not being a failing FRU. There should be other failure modes shown in the console output.

4.3 Diagnosis #1 - UPA_A_ERR (Etag Parity Error)

```
Fatal Reset
   0,0>   FATAL ERROR
   0,0>    At time of error: System software was running.
   0,0>    Diagnosis: Board 0, UPA PORT Device, AC
   0,0>   Log Date: May 22 12:02:03 GMT 2000
   0,0>   RESET INFO for CPU/Memory  board in slot 0
   0,0>    AC ESR 00000000.00600001 IPREP FERR UPA_A_ERR
   0,0>    DC[0] 00
   0,0>    DC[1] 00
   0,0>    DC[2] 00
   0,0>    DC[3] 00
   0,0>    DC[4] 00
   0,0>    DC[5] 00
   0,0>    DC[6] 00
   0,0>    DC[7] 00
   0,0>    FHC  CSR 00050200 LOC_FATAL SYNC NOT_BRD_PRES
   0,0>    FHC RCSR 02000000  FATAL
```

Comments:

Fatal Reset Errors that have the IPREP and FERR error bits set in the Address Controller Error Status Register indicate that an Ecache Tag parity Error may be the cause of the Fatal Reset. In the typical error message shown above, the failure can be diagnosed to be an Ecache Tag Parity Error on Board 0 CPU 0.

The Sun UltraSPARCSII Processor system CPUs will send a P_ERR P_REPLY to the AC for the following two conditions;

- Parity error on UPA address bus while AC is bus master & CPU is the slave.
- E-Cache tag parity error. This is not reported as a trap like an E-Cache data error because system coherence is lost for this condition and the system must be reset.

Improper torque may cause scenario (1) above but since the address bus is a bi-directional and the CPU is bus master, most of the time a mechanical connection problem would probably also produce fatal reset errors with the UPA_PERR bit set, i.e., the CPU is address bus master and the AC saw the parity error.

Absence of the UPA_PERR error bits (like the one shown above) indicates scenario (2) or a CPU Etag parity error is the most likely cause.

`prtdiag -v` output:

The following is sample output from a prtdiag -v session after a UPA_A_ERR Fatal Reset. This is prior to any hardware power cycling or replacement.

```
Analysis of most recent Fatal Hardware Watchdog:
================================================
Log Date: Sat Nov 11 06:47:43 2000
Analysis for Board 0
--------------------
AC: P_FERR error P_REPLY received from UPA Port
        The error could be caused by:
                CPU
                Address Controller
AC: Illegal P_REPLY received from UPA Port
        The error could be caused by:
                CPU
                Address Controller
```

Recommendation:

Note that UPA_A_ERR refers to CPU location 0 and UPA_B_ERR refers to CPU location 1.

CPU 0 on CPU/Memory Board 0 would be the FRU implicated in causing this failure and the CPU to replace.

4.4 Diagnosis # 2 - UPA_A_ERR (Etag Parity Error) Multiple Errors

```
Fatal Reset
7,0>FATAL ERROR
7,0> At time of error: System software was running.
7,0> Diagnosis: Board 9, UPA PORT Device, AC
7,0> Diagnosis: centerplane terminators
7,0>Log Date: Jan 19 13:54:45 GMT 2000
7,0>
7,0>RESET INFO for IO Type 4 board in slot 1
7,0> AC ESR 00002000.00000000 FTA_PERR
7,0> DC[0] 00
7,0> DC[1] 00
7,0> DC[2] 00
7,0> DC[3] 00
7,0> DC[4] 00
7,0> DC[5] 00
7,0> DC[6] 00
7,0> DC[7] 00
7,0> FHC CSR 00040000 LOC_FATAL
7,0> FHC RCSR 02000000 FATAL
7,0>RESET INFO for IO Type 4 board in slot 3
7,0> AC ESR 00002000.00000000 FTA_PERR
7,0> DC[0] 00
7,0> DC[1] 00
7,0> DC[2] 00
7,0> DC[3] 00
7,0> DC[4] 00
7,0> DC[5] 00
7,0> DC[6] 00
7,0> DC[7] 00
7,0> FHC CSR 00040000 LOC_FATAL
7,0> FHC RCSR 02000000 FATAL
Diagnosis # 2 - UPA_A_ERR (Etag Parity Error) Multiple Errors
(continued)
7,0>RESET INFO for CPU/Memory board in slot 7
7,0> AC ESR 00002000.00000000 FTA_PERR
```

```
7,0> DC[0] 00
7,0> DC[1] 00
7,0> DC[2] 00
7,0> DC[3] 00
7,0> DC[4] 00
7,0> DC[5] 00
7,0> DC[6] 00
7,0> DC[7] 00

7,0> FHC CSR 00050200 LOC_FATAL SYNC NOT_BRD_PRES
7,0> FHC RCSR 02000000 FATAL
7,0>RESET INFO for CPU/Memory board in slot 9
7,0> AC ESR 00000000.00600002 IPREP FERR UPA_B_ERR
7,0> DC[0] 00
7,0> DC[1] 00
7,0> DC[2] 00
7,0> DC[3] 00
7,0> DC[4] 00
7,0> DC[5] 00
7,0> DC[6] 00
7,0> DC[7] 00
7,0> FHC CSR 00050030 LOC_FATAL SYNC BRD_LED_M BRD_LED_R
7,0> FHC RCSR 02000000 FATAL
```

Comments:

The error on CPU/Memory Board 9 is either a CPU (location 1 / B port) Etag parity error or an incoming system address parity error on the UPA. Engineering experience says that it is most likely a CPU Etag (90%) as an ISAP would likely be accompanied by the bit UPA_PERR.

The FTA_PERRs reported by the other slots are likely artifacts.

Recommendation:

See Fatal Reset #1 above (UPA_A_ERR) for further information. The implicated FRU in this example is CPU 19 or CPU 1 on CPU/Memory Board 9.

4.5 DT_PERR (DTAG Parity Error)

```
Fatal Reset
0,0>FATAL ERROR
0,0>    At time of error: System software was running.
0,0>    Diagnosis: Board 6, Dtag A (UPA Port 0), AC
0,0>Log Date: Dec 17 22:20:15 GMT 2000
0,0>
0,0>RESET INFO for CPU/Memory  board in slot 6
0,0>    AC ESR 00000020.00000000 DT_PERRA
0,0>    DC[0] 00
0,0>    DC[1] 00
0,0>    DC[2] 00
0,0>    DC[3] 00
0,0>    DC[4] 00
0,0>    DC[5] 00
0,0>    DC[6] 00
0,0>    DC[7] 00
0,0>    FHC  CSR 00050030 LOC_FATAL SYNC BRD_LED_M BRD_LED_R
0,0>    FHC RCSR 02000000  FATAL
0,0> Config policy change
```

Comments:

The DT_PERRA in slot 6 indicates a Duplicate Tag SRAM (DTAG) parity error. These DTAG SRAMs reside on the CPU/Memory boards.

`prtdiag -v` output:

The following is sample output from a `prtdiag -v` session after a DT_PERRA or DT_PERRB Fatal Reset. This is prior to any hardware power cycling or replacement.

```
Analysis of most recent Fatal Hardware Watchdog:
================================================
Log Date: Sun Dec 17 14:20:15 2000
Analysis for Board 6
--------------------
AC: UPA Port A Dtag Parity Error
       The error could be caused by:
                Data Tags for UPA Port A
                Address Controller
```

Recommendation:

Note that DT_PERRA refers to DTAG SRAMs that refer to CPU location 0 and DT_PERRB refers to CPU location 1. Again, these DTAG SRAMs reside on the CPU/Memory Board, not on the CPU Module themselves.

In this example, the reporting CPU/Memory Board 6 should be replaced. The CPUs and memory on this CPU/Memory Board are good.

4.6 Diagnosis # 4 - DT_PERR (DTAG Parity Error) Multiple Errors

```
Fatal Reset
0,0>FATAL ERROR
0,0> At time of error: System software was running.
0,0> Diagnosis: Board 0, centerplane pin, connector pin, AC
0,0> Diagnosis: Board 14, Dtag A (UPA Port 0), AC
0,0> Diagnosis: centerplane terminators
0,0>Log Date: Jun 22 13:22:00 GMT 1999
0,0>RESET INFO for CPU/Memory board in slot 0
0,0> AC ESR 00000400.00000000 FT_ARBERR
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00050200 LOC_FATAL SYNC NOT_BRD_PRES
0,0> FHC RCSR 02000000 FATAL
0,0>RESET INFO for IO Type 4 board in slot 1
0,0> AC ESR 00002000.00000000 FTA_PERR
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00040000 LOC_FATAL
0,0> FHC RCSR 02000000 FATAL
0,0>RESET INFO for IO Type 4 board in slot 3
0,0> AC ESR 00002000.00000000 FTA_PERR
```

```
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00040000 LOC_FATAL
0,0> FHC RCSR 02000000 FATAL
0,0>RESET INFO for CPU/Memory board in slot 14
0,0> AC ESR 00000020.00000000 DT_PERRA
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00050030 LOC_FATAL SYNC BRD_LED_M BRD_LED_R
0,0> FHC RCSR 02000000 FATA
```

Comments:

Although this Fatal Reset output diagnoses several possible components, the real problem is a Dtag parity error on system board in slot 14.

The FT_ARBERR in slot 0 and the FTA_PERRs reported by the other slots are likely fatal reset artifacts.

Recommendation:

In this example, the reporting CPU/Memory Board 14 should be replaced. The CPUs and memory on this CPU/Memory Board are good.

4.7 Diagnosis #5 - FTA_PERR

```
0,0>FATAL ERROR
0,0> At time of error: POST was testing Board 0 Centerplane
0,0> Diagnosis: Board 0, backplane pins, board connector pins, AC
0,0>Log Date: Jun 19 7:10:00 GMT 1999
0,0>
0,0>RESET INFO for CPU/Memory board in slot 0
0,0> AC ESR 00002000.00000000 FTA_PERR
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00050200 LOC_FATAL SYNC NOT_BRD_PRES
0,0> FHC RCSR 02000000 FATAL
0,1>
```

Comments:

POST was testing "board 0 centerplane". However, there are cases where this type of Fatal Reset is not really caused by CPU / Memory Board 0, or the Centerplane. There may be other components in the system that is causing this Fatal Reset to occur.

Recommendation:

Although for this example, the reporting CPU / Memory board 0 can be replaced (existing CPUs and Memory on that board should be good), this error may return. If this is the case, it is recommended to try and test the system rigorously to determine if any components do fail. Replacement of the centerplane may not always correct this problem. This also points to some other FRU in the system not being called out in the Fatal Reset data.

4.8 Diagnosis #6 - FTC_PERR

```
# Fatal Reset
0,0>FATAL ERROR
0,0> At time of error: System software was running.
0,0> Diagnosis: Board 7, backplane pins, board connector pins, AC
0,0>Log Date: Aug 31 7:16:02 GMT 1997
0,0>
0,0>RESET INFO for IO Type 3 board in slot 7
0,0> AC ESR 00001000.00000000 FTC_PERR
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00040000 LOC_FATAL
0,0> FHC RCSR 02000000 FATAL
```

Comments:

Basically the same analysis as Fatal Reset case #5, except that we know system software was running and there are at least two boards in the system.

Recommendation:

In this example, the reporting I/O Board in slot 7 is the FRU to replace. Existing SBus controllers should be good.

4.9 Diagnosis #7 - MTIMEOUT

```
# Fatal Reset
0,0>FATAL ERROR
0,0> At time of error: System software was running.
0,0> Diagnosis: Board 7, any system board MTIMEOUT (target of
operation)
0,0>Log Date: Feb 12 6:35:31 GMT 199e
0,0>
0,0>RESET INFO for CPU/Memory board in slot 7
0,0> AC ESR 00000000.01000001 MTIMEOUT UPA_A_ERR
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00040000 LOC_FATAL
0,0> FHC RCSR 02000000 FATAL
```

Comments:

This error means that the CPU (location 0 / Port A) on CPU/Memory Board in slot 7, issued a request via the AC to a board which it knows is present, yet no response was received for 16M clock cycles (193ms @ 83Mhz). Note that this error cannot be caused by reading a non-existent location (that would cause a panic if it was issued by a CPU).

An example of this type of problem is where Psycho+ appears to not complete a register read to a PCI device in the allotted time.

In general this problem could be the fault of the reporting board, or the target of the board's operation. Unfortunately, we don't know the target. However, the target is many times an I/O board. As such, if there is only one or a few I/O boards in the system, you can narrow down your search.

NOTE: There are reported cases where a similar looking Fatal Reset will occur and the cause had been found to be a CPU Module. This has been seen during POST and the error signature appears as:

```
 0,0>        TESTCASE 00000000.01000002 AC ESR
```

`prtdiag -v` output:

The following is sample output from a `prtdiag -v` session after an MTIMEOUT Fatal Reset. This is prior to any hardware power cycling or replacement.

```
Analysis of most recent Fatal Hardware Watchdog:
================================================
Log Date: Mon May  8 14:33:19 2000
Analysis for Board 7
--------------------
AC: Timeout on a UPA Master Port
        The error could be caused by:
                Undetermined Address Controller in system
                Undetermined Board in system
```

Recommendation:

Unfortunately, the MTIMEOUT Fatal Reset gives little information regarding the true root cause of the error. Given that, it is recommended that the following actions be taken.

1. Review the system history to determine if any recent components have been recently replaced. Determine if any of these components could be likely suspects in the error.
2. If possible, shutdown the system and key-reset (power cycle) the system so that all components are active and run extended POST to determine if the failure is solid (not intermittent).
3. If the the failure is solid, use standard troubleshooting techniques to isolate the failing component.
4. If the failure is intermittent (meaning extended POST did not fail), try to run SunVTS. If SunVTS does not indicate a failing component, monitor the system for future failures. Especially on the CPU/Memory Board that was called out by the last MTIMEOUT error. This monitoring has proven beneficial in finding the correct FRU causing this type of Fatal Reset.

4.10 Diagnosis #8 - FTUPAOV

```
Fatal Reset
0,0>FATAL ERROR
0,0> At time of error: System software was running.
0,0> Diagnosis: Board 3, centerplane, board connector, AC, any
other AC
0,0>Log Date: Jul 29 20:14:41 GMT 1998
0,0>
0,0>RESET INFO for CPU/Memory board in slot 3
0,0> AC ESR 00000002.00000000 FTUPAOV
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00050020 LOC_FATAL SYNC BRD_LED_M
0,0> FHC RCSR 02000000 FATAL
```

Comments:

The Firetruck to UPA queue overflowed on CPU/Memory Board in slot 3. Most likely this is caused by a broken AC. Since CPU/Memory Board 3 is the only board reporting a problem, it is likely the reporting board. Otherwise, it could be some other board.

This is not likely a centerplane problem because you would usually see some sort of parity error in addition to the queue overflow.

Note – For some firetruck parity errors, all CPU/Memory and I/O boards will report errors. There may be one board that does not report an error. Based on this error mode, the board that did not report the error would be the suspect in sending of incorrect/bad data. Replacement of the non-reporting board would be a good first action.

Recommendation:

For this example, replace the reporting CPU/Memory Board in slot 3. Existing CPUs and Memory should be good.

4.11 Diagnosis #9 - Cacheable Write Error

```
Fatal Reset
0,0>FATAL ERROR
0,0> At time of error: System software was running.
0,0> Diagnosis: Board 2, software, any system board
0,0>Log Date: Mar 19 3:34:39 GMT 1998
0,0>
0,0>RESET INFO for CPU/Memory board in slot 2
0,0> AC ESR 00000000.00000041 ICWS UPA_A_ERR
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00050830 LOC_FATAL SYNC EPDB_OFF BRD_LED_M BRD_LED_R
0,0> FHC RCSR 02000000 FATAL
```

Comments:

This error is caused by a cacheable write being sent by the CPU (first CPU / Port A) on CPU/Memory Board 2 to an unmapped or non-accepting destination. Cacheable writes should only go to memory boards.

While this could be caused by software (e.g. system software). An example of a system software problem causing this error is during a Dynamic Reconfiguration operation where memory is being un-configured.

Thus the most likely problem in this example is either the reporting CPU/Memory Board 2 (or CPU), or the destination memory board. Unfortunately, the destination memory board is unknown.

`prtdiag -v output`:

The following is sample output from a `prtdiag -v` session after an ICWS Fatal Reset. This is prior to any hardware power cycling or replacement.

```
Analysis of most recent Fatal Hardware Watchdog:
================================================
Log Date: Fri Mar 17 01:08:48 2000

Analysis for Board 2
--------------------
AC: UPA Cacheable write to unmapped destination
        The error could be caused by:
               This Board
```

Recommendation:

Determine if any Dynamic Reconfiguration events were in process at the time of the failure. If not, for this example, the reporting CPU/Memory Board in slot 2 should be replaced. If available, the CPU Module in location 0 on CPU/Memory Board 2 should be replaced as well.

4.12 Diagnosis #10 - Non-Cacheable Write

```
Fatal Reset
0,0>FATAL ERROR
0,0> At time of error: System software was running
0,0> Diagnosis: Board 0, software, any system board
0,0>Log Date: Mar 20 2:05:33 GMT 1996
0,0>
0,0>RESET INFO for CPU/Memory board in slot 0
0,0> AC ESR 00000000.00000021 INCWS UPA_A_ERR
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00040000 LOC_FATAL
0,0> FHC RCSR 02000000 FATAL
```

Comments:

Same as previous, only a non-cacheable write. In this case the destination is likely to be an I/O board. Source (Master) was the CPU (location 0 / port A) on CPU/Memory Board 0.

`prtdiag -v` output:

```
Analysis of most recent Fatal Hardware Watchdog:
================================================
Log Date: Sat May 13 03:07:59 2000
Analysis for Board 0
--------------------
AC: UPA Non-cacheable write to unmapped destination
        The error could be caused by:
This Board
```

Recommendation:

Determine if any Dynamic Reconfiguration events were in process at the time of the failure. If not, for this example, the reporting CPU/Memory Board in slot 0 should be replaced. If available the CPU Module in location 0 on CPU/Memory Board 0 should be replaced as well.

4.13 Diagnosis #11 - Interrupt Error

```
0,0> At time of error: System software was running.
0,0> Diagnosis: Board 4, software, any system board
0,0>Log Date: Dec 4 19:07:57 GMT 1997
0,0>
0,0>RESET INFO for CPU/Memory board in slot 4
0,0> AC ESR 00000000.00000011 IIS UPA_A_ERR
0,0> DC[0] 00
0,0> DC[1] 00
0,0> DC[2] 00
0,0> DC[3] 00
0,0> DC[4] 00
0,0> DC[5] 00
0,0> DC[6] 00
0,0> DC[7] 00
0,0> FHC CSR 00040030 LOC_FATAL BRD_LED_M BRD_LED_R
0,0> FHC RCSR 02000000 FATAL
```

Comments:

Similar to the previous Fatal Reset, only an interrupt was sent instead of a write. The source was the CPU (location 0 / port A) on CPU/Memory Board in slot 4. The target of any interrupt would have been another CPU board (not I/O).

`prtdiag-v` output:

```
Analysis of most recent Fatal Hardware Watchdog:
================================================
Log Date: Wed Dec  1 10:10:00 1999
Analysis for Board 4
AC: UPA Interrupt to unmapped destination
        The error could be caused by:
                This Board
--------------------
```

Recommendation:

Determine if any Dynamic Reconfiguration events were in process at the time of the failure. If not, for this example, the reporting CPU/Memory Board in Slot 4 should be replaced. If available the CPU Module in location 0 on CPU/Memory Board 4 should be replaced as well.

CHAPTER 5

Enhanced Solaris Error Messaging

5.1 Improved Error Messages

With the introduction of the kernel level scrubber patches (Kernel Update or KU patches) for Solaris 2.6 and above, the CPU, Ecache, and memory error messages have been improved to be more accurate and complete. Text descriptions have been rewritten to emphasize the important parameters associated with each event. Also, the logic for reporting hardware errors has changed to ensure that error events are reported accurately, completely, and in the order they occurred. These new error messages will make it easier to determine the CPU that has encountered an error.

There are related patches to Sun Management Center so that it will recognize the improved error messages; without them, the management console will under-report the occurrence of corrected main memory errors. Refer to the Solaris KU patch readme files for further information.

Note – Solaris 2.5.1 and prior releases do not support improved error messaging..

5.2 Errors and Events

SunSPARC processors can detect errors that are reported in the following types of events (as detailed in the SunSPARC-I/II User's Manual, 802-7220-02):

- ETP – A parity error was detected by the CPU when reading from the Ecache Tag SRAM. This is a fatal error because system coherency has been lost. The system will reset (POR) and Starfire domains will arbstop (UPA Fatal error). No Solaris error message will be generated.
- EDP – A parity error was detected by the CPU when reading from the Ecache Data SRAM on a cache hit.
- LDP – A parity error was detected by the CPU while reading main memory through its Ultra Data Buffer (UDB) chip on an Ecache miss. Note that the Ecache itself is not involved. This can occur when the CPU is reading non-cacheable data (for example, a frame buffer or I/O device), or when filling a line of cache from main memory.
- WP – A parity error was detected by one of the UDB chips while data was being written back from the Ecache into main memory. The UDB chips convert the data with bad parity into data with bad ECC, so that a subsequent access to the same physical address will result in a UE. (See UE below.) (The conversion of a parity error to a latent UE does not occur on either UltraSPARC-IIi or -IIe, which is one of the reasons why improved error handling is not available on those processors.)
- CP – A parity error was detected during a copyout transaction; that is, a data transfer from one CPU's Ecache to another CPU. This error is detected by the UDB chips of the providing CPU, resulting in the CP event. The providing CPU's UDB chips convert the data with bad parity to data with bad ECC, so that the UDBs of the receiving CPU will report a UE event. (See UE below.)
- UE – An un-correctable memory error has occurred. This event refers to an error in the main system memory, reported by the system databus on a read access. The underlying source of this error could be main memory, another CPU module (see CP above), or another UPA device (for example, the I/O controller). The UDB chips detect this error.
- CE – A correctable error was detected when reading from main memory, or when reading from another CPU's UDB chips. The data read has been corrected and valid data is given to the CPU and the CPU's Ecache. This error is detected by the UDB chips.
- BERR – A bus error has occurred during an attempt to read from a memory address. Either there is no device at that address, or the device at that address has returned a bus error. Therefore, bus errors are caused by a programming error or by a corrupted or defective device.

- TO – A bus timeout was encountered during an attempt to read from a memory address. Too much time has elapsed waiting for a device at that address to respond.

5.3 Details on Improved Error Handling

Any of the above mentioned errors can occur in kernel instruction space, kernel data space, user instruction space, user data space, or when the kernel reads or writes user data (as in copyin). Depending on these different states, the operating system will react differently so as to maximize system availability.

On EDP, LDP, CP, UE, BERR, and TO events, the system will panic if the affected data is in kernel space or if the error occurs while the CPU is at a trap level greater than zero. Otherwise, the process that caused the error will be killed immediately (sent SIGKILL) and the system will be rebooted (as if a privileged user had entered "init 6").

Note – An active SC2.X cluster node will panic with a "Failfast timeout" (usually with "Device closed while Armed") when rebooted. It is therefore useful to check the system messages for EDP, LDP, CP, UE, BERR, and TO events while encountering "Failfast timeout" panics.

On WP events, an error is reported, and the memory scrubber is notified to scan all of system memory for the latent UE the hardware has written to memory (see below for the behavior of the memory scrubber on encountering UE events). If some CPU later attempts to read this location (other than on behalf of the memory scrubber), a UE event will occur. Hence, when a UE event is encountered, it is recommended that the log be checked for an earlier WP event that may have in fact caused the UE event.

If the memory scrubber detects a UE event the system will neither panic nor reboot but trigger a recovery mechanism instead. If the page containing the corrupted data is not in use, it will be retired and the error will be cleared. If it is in use, it will be marked for retirement and clearing if and when it is no longer in use.

Note – Due to hardware limitations there is no improved error handling for UltraSPARC-IIi and UltraSPARC-IIe based systems.

5.4 Details on Improved Error Messages

For each error that is detected, the kernel generates an individual report. This is a major change; previously, some errors would hide other errors, and some errors were combined into a single message. The report typically consists of several error messages. Each message [3] contains an AFT ("Asynchronous Fault Trap") tag that eases filtering, and an errID code that associates all of the messages emitted for the same event. The errID is a 64-bit code that corresponds to a specific set of error bits in the Asynchronous Fault Status Register (AFSR) at a specific instance in time; the value has no intrinsic meaning.

Each message may be longer than one physical line; long messages are folded using embedded newlines. Each folded line begins with four space characters.

Note – Because of the introduction of improved error messages, any tool using the affected error messages may have to be modified. Neither the format nor the content of kernel error messages are committed interfaces, and both may change without notice. Users (both internal and external) who rely on the exact format and/or content do so at their own risk.

5.5 Error Message Categories

The error messages can be grouped into four categories.

Category 1: Messages that identify the type and source of an error:

```
WARNING: [AFT1] EDP event on CPU1 Instruction access at TL=0, errID
0x0000ad88.6cd9989f
    AFSR 0x00000000.80408000<PRIV,EDP> AFAR 0x00000000.0f0c8080
    AFSR.PSYND 0x8000 (Score 95) AFSR.ETS 0x00 FAULT_PC 0x780b481c
    UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0000 UDBL.ESYND 0x00
```

Either the [AFT0] tag (for correctable errors) or the [AFT1] tag (for un-correctable errors) is present in the message. An "errID" field appears at the end of the first line of the message. Messages from this category are displayed on the console and collected in the log file.

Note – This is the default behavior. The /etc/system setting report_ce_console is no longer referenced and should therefore be removed.

To aid diagnosis of an Ecache-related error, especially if multiple components are involved, a heuristic algorithm has been included that automates analysis of the P_SYND bytes. Every component reporting a failure has its AFSR decoded and a score ranging from 5 to 95 is assigned ("Score 95" in the above example).

The Score indicates the likelihood that this component was the original source of the bad parity. The higher the value, the higher the likelihood that this component was the original source.

Category 2: Messages that supply a cache line or memory dump:

```
[AFT2] errID 0x0000ad88.6cd9989f PA 0x00000000.0f0c8080 E$tag
0x00000000.0bc001e1 E$State: Modified E$parity 0x05
    [AFT2] E$Data (0x00): 0xffffffff.beefface *Bad* PSYND=0x8000
    [AFT2] E$Data (0x08): 0x00000000.00000000
    [AFT2] E$Data (0x10): 0x6d656d6d.6f727920
    [AFT2] E$Data (0x18): 0x6572726f.7220696e
    [AFT2] E$Data (0x20): 0x6a656374.6f720000
    [AFT2] E$Data (0x28): 0x6d656d74.65737420
    [AFT2] E$Data (0x30): 0x6d757465.780059f8
    [AFT2] E$Data (0x38): 0x00000300.00c11000
    [AFT2] Event PA displayed in AFAR was derived from E$Tag
```

Messages from this category are targeted for Sun Microsystems support staff to be used in backline diagnosis and for statistics.

The [AFT2] tag is always present in these messages. The "errID" field appears at the beginning of the first line of the message. Messages from this category are by default only collected in the log file.

Category 3: Messages from the kernel error recovery code:

```
[AFT3] errID 0x00000058.0d0dc830 Above Error detected by protected
Kernel code that will try to clear error from system
```

Messages from this category supply analysis information from the kernel error recovery code, thereby indicating the actions the kernel took to contain the error.

The [AFT3] tag is always present in these messages. An "errID" field appears at the beginning of the first line of the message.

Messages from this category are by default only collected in the log file.

Category 4: Messages that indicate the disposition of an error:

```
    panic[CPU1]/thread=30000670800: [AFT1] errID
0x00000392.89cbfefc EDP Error(s)
         See previous message(s) for details
```

Messages from this category state the final handling (like panic or reboot) of a previously encountered error.

Either the [AFT0] tag (for correctable errors) or the [AFT1] tag (for un-correctable errors) is present in the message. The "errID" field appears at the beginning of the first line of the message. Messages from this category are displayed on the console and collected in the log file.

5.6 Error Messages Examples

The following compares previous messages with the new, improved error messages. Note that this is not an exhaustive list, but a sampling of possible messages for each event type. This also just shows what appears on the console; the log-only messages are not shown.

Error messages shown below do not necessarily appear exactly as they appear on the console. Due to message lengths, etc. the message lines shown below may wrap around, however the entire content of the message lines are shown below.

5.7 EDP Event - Ecache Data Parity Event

Previous Style Message - Kernel Mode (Panic):

```
panic[CPU1]/thread=3000225bcc0: CPU1 Ecache SRAM Data Parity
Error:
AFSR 0x00000000.80408000 AFAR 0x00000000.0bd83bd0
```

Improved Message - Kernel Mode (Panic):

```
    WARNING: [AFT1] EDP event on CPU1 Data access at TL=0, errID
0x00000093.6323e6f8AFSR 0x00000000.80408000<PRIV,EDP> AFAR
0x00000000.06901980

AFSR.PSYND 0x8000 (Score 95) AFSR.ETS 0x00 Fault_PC 0x78128a84
UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0000 UDBL.ESYND 0x00

panic[cpu1]/thread=30000ae5000: [AFT1] errID 0x00000093.6323e6f8
EDP Error(s)   See previous message(s) for details
```

Previous Style Message - User Mode (Panic):

```
panic[CPU3]/thread=30001f4fa00: CPU3 Ecache SRAM Data Parity
Error:
    AFSR 0x00000000.00400080 AFAR 0x00000000.01820000
```

Improved Message - User Mode (Reboot):

```
    Aug 16 16:47:20 thishost SUNW,UltraSPARC-II: WARNING: [AFT1]
EDP event on CPU3  Data access at TL=0, errID 0x00000057.d35eff81
    Aug 16 16:47:20 thishost    AFSR 0x00000000.00400080<EDP> AFAR
0x00000000.05e24418 AFSR.PSYND 0x0080 (Score 95) AFSR.ETS 0x00
Fault_PC 0x11ce8  UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0000
UDBL.ESYND 0x00
    Aug 16 16:47:20 thishost unix: NOTICE: Scheduling clearing of
error on page 0x00000000.05e24000
Aug 16 16:47:20 thishost unix: WARNING: [AFT1] initiating reboot
due to above error in pid 309 (mtst)
    Aug 16 16:47:23 thishost unix: NOTICE: Previously reported
error on page 0x00000000.05e24000 cleared
```

Improved Message - User Data (Reboot):

```
INIT: New run level: 6
The system is coming down.  Please wait.
System services are now being stopped.
Print services stopped.
Aug 16 16:47:27 thishost syslogd: going down on signal 15
The system is down.
syncing file systems... done
rebooting...
Resetting ...
```

5.8 Trap Level 1 Panic

Previous Solaris Message - Kernel Data at TL=1 (Panic):

```
panic[CPU3]/thread=30001cfabe0:  Async data error at tl1:  AFAR
0x00000000.0ab8f760 AFSR 0x00000000.80400080
```

Improved Message - Kernel Data at TL=1 (Panic):

```
WARNING: [AFT1] EDP event on CPU3 Data access at TL>0, errID
0x00000111.53a7b8dd AFSR 0x00000000.80408000<PRIV,EDP> AFAR
0x00000000.01f47dc0 AFSR.PSYND 0x8000 (Score 95) AFSR.ETS 0x00
Fault_PC 0x1002fe20 UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0000
UDBL.ESYND 0x00
panic[cpu3]/thread=30000a4e040: [AFT1] errID 0x00000111.53a7b8dd
EDP Error(s)
```

See previous message(s) for details

Previous Solaris Message - Kernel Instruction at TL=1 (Panic):

```
panic[CPU3]/thread=3000226a140:  Async instruction error at tl1:
AFAR 0x00000000.0dd55f70 AFSR 0x00000000.80408000
```

Improved Message - Kernel Instruction at TL=1 (Panic):

```
WARNING: [AFT1] EDP event on CPU3 Instruction access at TL>0, errID
0x00000043.24bfd349 AFSR 0x00000000.80400800<PRIV,EDP> AFAR
0x00000000.0605c790 AFSR.PSYND 0x0800 (Score 95) AFSR.ETS 0x00
Fault_PC 0x1002fe20 UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0000
UDBL.ESYND 0x00
panic[cpu3]/thread=30000ad05c0: [AFT1] errID 0x00000043.24bfd349
EDP Error(s)
```

See previous message(s) for details.

5.9 WP Event - Writeback Data Parity Error

Previous Solaris Message (Panic):

```
panic[CPU1]/thread=30001b26640:  CPU1 Ecache Writeback Data Parity
Error:        AFSR 0x00000000.00800080 AFAR 0x00000000.0d5010f0
```

Improved Message (Panic Deferred):

```
Aug 16 16:50:56 thishost SUNW,UltraSPARC-II: WARNING: [AFT1] WP
event on CPU1, errID 0x0000002b.3c7cd6d9

Aug 16 16:50:56 thishost   AFSR 0x00000000.00800080<WP> AFAR
0x000001c8.01802800

Aug 16 16:50:56 thishost    AFSR.PSYND 0x0080 (Score 95) AFSR.ETS
0x00 Fault_PC 0x11d7c  UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0000
UDBL.ESYND 0x00
Aug 16 16:50:56 thishost SUNW,UltraSPARC-II: WARNING: [AFT1]

Uncorrectable Memory Error on CPU3 Data access at TL=0, errID
0x0000002b.45daae92

Aug 16 16:50:56 thishost  AFSR 0x00000000.80200000<PRIV,UE> AFAR
0x00000000.03824418

Aug 16 16:50:56 thishost  AFSR.PSYND 0x0000(Score 05) AFSR.ETS 0x00
Fault_PC 0x10023414  UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0203<UE>
UDBL.ESYND 0x03 UDBL Syndrome 0x3 Memory Module 190x

Aug 16 16:50:56 thishost SUNW,UltraSPARC-II: WARNING: [AFT1] errID
0x0000002b.45daae92 Syndrome 0x3 indicates that this may not be a
memory module problem

Aug 16 16:50:56 thishost unix: NOTICE: Scheduling clearing of error
on page 0x00000000.03824000

Aug 16 16:50:58 thishost unix: NOTICE: Previously reported error
on page 0x00000000.03824000 cleared
```

Note – The last message (reporting clearing of the error) may appear much later, or may never appear, as the page may never drop out of use. Also, the message reporting scheduling of clearing may occur more than once, as the memory scrubber may encounter the particular UE more than once before it can be cleared.

5.10 CP Event - Copyout Data Parity Error

Previous Solaris Message (Panic):

```
panic[CPU3]/thread=2a100105d40:  CPU3 UE Error:  Ecache Copyout on
CPU1:      AFSR 0x00000000.01000080 AFAR 0x00000000.06c53090
```

Improved Message Kernel Mode (Panic):

```
WARNING: [AFT1] Uncorrectable Memory Error on CPU3 Data access at
TL=0, errID 0x0000003a.30aafcba AFSR 0x00000000.80200000<PRIV,UE>
AFAR 0x00000000.00347dc0 AFSR.PSYND 0x0000 (Score 05) AFSR.ETS
0x00 Fault_PC 0x78067b54 UDBH 0x0203<UE> UDBH.ESYND 0x03 UDBL
0x0000 UDBL.ESYND 0x00 UDBH Syndrome 0x3 Memory Module 190x

WARNING: [AFT1] errID 0x0000003a.30aafcba Syndrome 0x3 indicates
that this may not be a memory module problem

WARNING: [AFT1] CP event on CPU1 (caused Data access error on
CPU3), errID 0x0000003a.30aafcba AFSR 0x00000000.01008000<CP> AFAR
0x00000000.00347dc0 AFSR.PSYND 0x8000(Score 95) AFSR.ETS 0x00 UDBH
0x0000 UDBH.ESYND 0x00 UDBL 0x0000 UDBL.ESYND 0x00

panic[cpu3]/thread=2a100157d40: [AFT1] errID 0x0000003a.30aafcba
UE Error(s)
```

See previous message(s) for details.

Improved Message User Mode (Reboot):

```
Aug 16 17:06:44 thishost SUNW,UltraSPARC-II: WARNING: [AFT1]
Uncorrectable Memory Error on CPU3 Data access at TL=0, errID
0x0000002b.963a3d3c

Aug 16 17:06:44 thishost     AFSR 0x00000000.00200000<UE> AFAR
0x00000000.00224418 AFSR.PSYND 0x0000 (Score 05) AFSR.ETS 0x00
Fault_PC 0x12380  UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0203<UE>
UDBL.ESYND 0x03

Aug 16 17:06:44 thishost     UDBL Syndrome 0x3 Memory Module 190x

Aug 16 17:06:44 thishost SUNW,UltraSPARC-II: WARNING: [AFT1] errID
0x0000002b.963a3d3c Syndrome 0x3 indicates that this may not be a
memory module problem

Aug 16 17:06:44 thishost SUNW,UltraSPARC-II: WARNING: [AFT1] CP
event on CPU1 (caused Data access error on CPU3), errID
0x0000002b.963a3d3c                                          AFSR
0x00000000.01000080<CP> AFAR 0x00000000.00224418  AFSR.PSYND
0x0080 (Score 95) AFSR.ETS 0x00 UDBH 0x0000 UDBH.ESYND 0x00 UDBL
0x0000 UDBL.ESYND 0x00

Aug 16 17:06:44 thishost unix: NOTICE: Scheduling clearing of error
on page 0x00000000.00224000  WARNING: [AFT1] initiating reboot due
to above error in pid 304

Aug 16 17:06:46 thishost unix: NOTICE: Previously reported error
on page 0x00000000.00224000 cleared

INIT: New run level: 6
The system is coming down.  Please wait.
CP Event - Copyout Data Parity Error (continued)

Print services stopped.

syslogd: going down on signal 15

The system is down.

syncing file systems... done

rebooting...

Resetting ...
```

Note – Due to a coding error, early versions of some of the patches produce the string "CP Error" instead of "CP event"; programs that parse the messages must be prepared to deal with both

5.11 UE Event - Uncorrectable Memory Error

Previous Solaris Message - CPU Reference to Memory:

```
panic[CPU1]/thread=2a1000R7dd40: UE Error: AFSR
0x00000000.80200000 AFAR 0x00000000.089cd740 Id 0 Inst 0 MemMod
U0501 U0401
```

Improved Message - CPU Reference to Memory Kernel Mode (Panic):

```
WARNING: [AFT1] Uncorrectable Memory Error on CPU1 Instruction
access at TL=0, errID 0x0000004f.818d9280 AFSR
0x00000000.80200000<PRIV,UE> AFAR 0x00000000.0685c7a0 AFSR.PSYND
0x0000 (Score 05) AFSR.ETS 0x00 Fault_PC 0x7815c7a0 UDBH
0x0203<UE> UDBH.ESYND 0x03 UDBL 0x0000 UDBL.ESYND 0x00 UDBH
Syndrome 0x3 Memory Module 190x
WARNING: [AFT1] errID 0x0000004f.818d9280 Syndrome 0x3 indicates
that this may not be a memory module problem
panic[cpu1]/thread=30000ad6320: [AFT1] errID 0x0000004f.818d9280
UE Error(s)
```

See previous message(s) for details.

Improved Message - CPU Reference to Memory User Mode (Reboot):

```
Aug 16 17:03:04 thishost SUNW,UltraSPARC-II: WARNING: [AFT1]
Uncorrectable Memory Error on CPU1 Instruction access at TL=0,
errID 0x00000032.593d8229

Aug 16 17:03:04 thishost     AFSR 0x00000000.00200000<UE> AFAR
0x00000000.04921bf0 AFSR.PSYND 0x0000 (Score 05) AFSR.ETS 0x00
Fault_PC 0x11bf0

Aug 16 17:03:04 thishost    UDBH 0x0203<UE> UDBH.ESYND 0x03 UDBL
0x0000 UDBL.ESYND 0x00 UDBH Syndrome 0x3 Memory Module 190x

Aug 16 17:03:04 thishost SUNW,UltraSPARC-II: WARNING: [AFT1] errID
0x00000032.593d8229 Syndrome 0x3 indicates that this may not be a
memory module problem

Aug 16 17:03:04 thishost unix: NOTICE: Scheduling clearing of error
on page 0x00000000.04920000

Aug 16 17:03:07 thishost unix: NOTICE: Previously reported error
on page 0x00000000.04920000 cleared

Aug 16 17:03:07 thishost unix: WARNING: [AFT1] initiating reboot
due to above error in pid 304 (mtst)

INIT: New run level: 6

The system is coming down.  Please wait.

System services are now being stopped.

Print services stopped.

Aug 16 17:03:13 thishost syslogd: going down on signal 15

The system is down.

syncing file systems... done

rebooting...

Resetting ...
```

Previous Solaris Message - SBus I/O Reference to Memory:

```
panic[CPU1]/thread=2a10007dd40:  SBus0 UE Primary Error DMA read:
AFSR 0x40001be0.00000000 AFAR 0x00000000.02818000 MemMod U0501
U0401 Id 31
```

Improved Message - SBus I/O Reference to Memory:

```
WARNING: SBus0 UE Primary Error DMA read: AFSR 0x40001be0.00000000
AFAR 0x00000000.0d25c000 MemMod U0501 U0401 Id 31

panic[cpu0]/thread=2a10007dd40: Fatal Sbus0 UE Error
```

5.12 BERR Event - Bus Error

Previous Solaris Message:

```
panic[CPU1]/thread=30000d2c300:  CPU1 Privileged Bus Error:  AFSR
0x00000000.84000000 AFAR 0x00000000.03422000
```

Improved Message - Kernel Mode (Panic):

```
WARNING: [AFT1] Bus Error on System Bus in privileged mode from
CPU1 Data access at TL=0, errID 0x0000002c.52b3d2c8  AFSR
0x00000000.84000000<PRIV,BERR> AFAR 0x00000000.05224410

AFSR.PSYND 0x0000 (Score 05) AFSR.ETS 0x00 Fault_PC 0x780671a4
UDBH 0x0000 UDBH.ESYND 0x00 UDBL 0x0000 UDBL.ESYND 0x00

panic[cpu1]/thread=30000b06080: [AFT1] errID 0x0000002c.52b3d2c8
BERR Error(s)
```

See previous message(s) for details.

5.13 CE Event - Correctable Memory Error

Previous Solaris Messages:

```
May  8 14:35:30 thishost SUNW,UltraSPARC-II: CPU1 CE Error: AFSR
0x00000000.00100000 AFAR 0x00000000.8abb5a00 UDBH Syndrome 0x85
MemMod U0904
May  8 14:35:30 thishost SUNW,UltraSPARC-II:     ECC Data Bit 63
was corrected
May  8 14:35:30 thishost unix: Softerror: Intermittent ECC Memory
Error, U0904
```

Improved Message:

```
Aug 16 16:34:48 thishost SUNW,UltraSPARC-II: [AFT0] Corrected
Memory Error on CPU1, errID 0x00000036.629edc25  AFSR
0x00000000.00100000<CE> AFAR 0x00000000.00347dc0 AFSR.PSYND
0x0000 (Score 05) AFSR.ETS 0x00 Fault_PC 0x1002fe20

Aug 16 16:34:48 thishost    UDBH Syndrome 0x85 Memory Module 1904

Aug 16 16:34:48 thishost SUNW,UltraSPARC-II: [AFT0] errID
0x00000036.629edc25 Corrected Memory Error on 1904 is Intermittent

Aug 16 16:34:48 thishost SUNW,UltraSPARC-II: [AFT0] errID
0x00000036.629edc25 ECC Data Bit 63 was in error and corrected
```

CHAPTER 6

Configuration Steps

The following section describes the actions to take to properly configure a system to capture failure information, and the specific problem solving steps to take after one of the system problems scenarios previously discussed has been identified.

As mentioned previously, the Sun Solution Center engineers may ask for additional information or recommend additional steps. The following is a guide to configure systems and capture the most appropriate information for the majority of error situations.

Step 1 Initial System Configuration (before errors occur)

Step 2 Error Identification and Error Response

Step 3 Information Gathering and Follow-up Tasks

6.1 Step 1 - Initial System Configuration

Table 6-1 shows Sun Enterprise xx00 server preparation steps to properly recover and save necessary error information for further problem solving.

TABLE 6-1 System Configuration Task List

Configuration Item	Reason	Instructions or Process	Frequency
General Information Gathering Enable Console Logging	Some detailed failure information is only logged to the system console.	Refer to Appendix E for additional information.	Every Server
Update the Kernel Update (KU) patches on Solaris 2.5.1 and above.	Kernel Level E-Cache Scrubber and enhanced error messaging for 2.6 & above.	Refer to the Solaris KU patch README file for install instructions.	Every Server
Update system flashprom/EEPROM	Newer versions (27 & above) capture additional failure data.	Refer to firmware patch (103346) README file for instructions.	Every xx00 Server
Enable saving of a system panic dumps.	To save vmcore dumps for further analysis for panics.	Refer to Appendix A for information.	Every Server
Install iscda script	To produce initial vmcore dump analysis data.	Refer to Appendix B for information.	Every Server

6.2 Step 2 - Error Identification and Error Response

Table 6-2 describes the symptom, the possible reason for the problem, and the specific steps to take, once the problem has been identified.

TABLE 6-2 Identifying a System Error

Symptom	Identification Tasks	Data Collection
System Rebooted Determine if any of the following can be associated with the system reboot.	1. Was system reboot a user initiated event? f so, understand why this is considered unexpected reboot.	PossibleAdministrationissue. Consult with user who shutdown system unexpectedly to determine the reason.
	2. Did the system reboot due to E-Cache Parity Errors? Solaris 2.6 and higher KU patches have this capability.	Collect /var/adm/messages file for further review for indications of E-Cache parity errors.
	3. Does a new vmcore/unix system dump exist in /var/crash/‘uname -n‘ If yes, this was a system panic.	New vmcore/unix files indicates a system panic occurred. See System Panic in Step 3 below.
	4. Determine if a Fatal Reset/Fatal Error or other hardware problem was the cause of the system reboot. Console output can indicate this.	Collect console and prtdiag -v output. Determine if any new hardware problems were detected. See Fatal Reset in Step 3 below.
	5. Review console messages for indications of why the system may have rebooted.	Look for indications of Fatal Resets or Fatal Errors, system panics which failed to save a dump, etc.
System Panic Determine if any of the following can be associated with a system panic.	1. Does a new vmcore/unix dump exist in /var/crash/‘uname -n‘ If yes, then this was a system panic.	Execute the iscda script to obtain an initial analysis of the system panic. See System Panic in Step 3 below.
	2. Review console messages for indications of why the system may have or have not saved a vmcore file.	Look for indications as to why the system failed to save a dump or why the dump aborted. Dump device too small, second panic during the panic process, etc.

TABLE 6-2 Identifying a System Error

Symptom	Identification Tasks	Data Collection
Fatal Reset / Error Determine if any of the following can be associated with a Fatal Reset / Fatal Error.	1. Determine if a Fatal Reset / Fatal Error or other hardware problem was the cause of the reboot. Console output can indicate this.	Use console output and prtdiag -v to determine if any new hardware problems have been detected. See Fatal Reset in Step 3 below.
System Soft Hangs Determine if any of the following reasons can be associated with a system soft hang.	1. If possible, determine the state of the system LEDs to verify if they are in a normal operating condition.	Observe the system to determine the state of the LEDs. Prtdiag -v will provide some info if it can be used.
	2. Determine if there is any existing login sessions that are active. If so, use these sessions to collect additional information and if necessary force a system panic.	If an active login can be accessed, commands (ifconfig, netstat, etc.) should be issued to verify the state of the network. This active login can also be used to force a panic.
	3. Determine if a console login can be initiated using the serial console port.	Perform the same steps as shown above for an active network login.
	4. If no access is possible, attempt to send a system break signal via the system console.	If OBP can be entered, refer to Step 3 below regarding further OBP data collection commands.
	5. Locate the XIR button and prepare to push it. Document the system LEDs and console for any response.	The XIR button may or may not cause the system to respond, since this is a hard hang condition.
	6. Prepare to Power Cycle the system.	This will reset the system and reboot.
System Hard Hangs Determine if any of the following reasons can be associated with a system hard hang.	1. If possible, determine the state of the system LEDs to verify if they are in a normal operating condition.	Observe the system to determine the state of the LEDs. Prtdiag -v will provide some info if it can be used.

TABLE 6-2 Identifying a System Error

Symptom	Identification Tasks	Data Collection
System Hard Hangs (continued)	2. Determine if there is any existing login sessions that are active. If so, use these sessions to collect additional information and if necessary force a system panic.	If an active login can be accessed, commands (ifconfig, netstat, etc.) should be issued to verify the state of the network. This active login can also be used to force a panic.
	3. Determine if a console login can be initiated using the serial console port.	Perform the same steps as shown above for an active network login.
	4. If no access is possible, attempt to send a system break signal via the system console.	If OBP can be entered, refer to Step 3 below regarding further OBP data collection commands.
	5. Locate the XIR button and be prepared to push it. Note the system LEDs and console for any response.	The XIR button may or may not cause the system to respond, since this is a hard hang condition.
	6. Prepare to Power Cycle the system.	This will reset the system and reboot.

6.3 Step 3 - Information Gathering and Follow-up Tasks

Problem	Error Response	Follow-up Tasks / Reporting
System Rebooted	1. If reboot was caused by an E-Cache parity error, collect messages file.	Collect /var/adm/messages file to be given to Sun Support for analysis.
	2. Verify that all hardware is present and functional.	Use `prtdiag -v` to determine hardware status.
	3. Verify that all I/O (disks, filesystems, etc.) are present, still mirrored, etc.	Use appropriate Solstice DiskSuite and Veritas commands.
	4. Verify that all user and system processes are functional.	Use appropriate commands to get state of user and system processes.
	5. Report problem to Sun as an Unexplained System Reboot	Place service call to Sun Service using documented call process.
	6. Schedule maintenance window as necessary for any service actions.	As recommended by the Sun support Engineer.
System Panic	1. If valid vmcore/unix dump exists, execute iscda script for initial analysis.	Collect output from iscda. Key on panic string information.
	2. If no vmcore file, review console output for additional information.	Review console output for panic string and reason dump failed.
	3. Collect messages file as well and review for errors prior to the panic.	Collect /var/adm/messages file to be given to Sun Support for analysis.
	4. Verify that all hardware is present and functional.	Collect `prtdiag -v` output to determine hardware status.
	5. Verify that all I/O (disks, filesystems, etc.) are present, still mirrored, etc.	Use appropriate Solstice DiskSuite and/or Veritas commands.
	6. Verify that all user and system processes are functional.	Use appropriate commands to get state of user and system processes.
	7. Report problem to Sun as a System Panic	Place service call to Sun Service using documented call process.

Problem	Error Response	Follow-up Tasks / Reporting
	8. Schedule maintenance window as necessary for any service actions.	As recommended by the Sun support Engineer.
Fatal Reset / Error	1. If Fatal Reset / Fatal Error occurred, review console messages.	Collect console messages to determine root cause of Fatal Reset.
	2. **PRIOR** to power cycling the system, capture prtdiag -v output.	Collect prtdiag -v for additional Fatal Reset data & hardware status.
	3. Verify that all I/O (disks, filesystems, etc.) are present, still mirrored, etc.	Use appropriate Solstice DiskSuite and Veritas commands.
	4. Verify that all user and system processes are functional.	Use appropriate commands to get state of user and system processes.
	5. Report problem to Sun as an Fatal Reset / Fatal Error	Place service call to Sun Service using documented call process.
	6. Schedule maintenance window as necessary for any service actions.	As recommended by the Sun support Engineer.
	7. .If necessary, enable console logging if possible.	Request that customer enable console connection / logging via serial ttya port.
System Soft Hang	1. If possible, determine the state of the system LEDs to verify if they are in a normal operating condition.	Document LED state on all individual CPU / Memory Boards, Clock Boards and front panel LEDs.
	2. Determine if there is any existing login sessions that are active. Use these logins before proceeding any further.	Note if login sessions are available, and if so, use Solaris commands to get system status.
	3. Determine if a console login can be initiated using the serial console port (ttya port). Look for errors in /var/adm/messages, etc.	Note if login sessions are available, and if so, use Solaris commands to get system status.
	4. If no access is possible, attempt to send a system break signal via the system console.	Execute OBP commands to gain additional failure information. Refer to Appendix C & D.
	5. Press the XIR button. This button is located at the rear of the xx00 server.	Document the system LEDs and monitor the console for any output.

Problem	Error Response	Follow-up Tasks / Reporting
System Soft Hang (continued)	6. Power Cycle the system.	Power off the system using the key switch and turn key to the Diagnostic position to run maximum level POST. Document any errors.
	7. Verify the hardware state of the system.	ICollect prtdiag -v for hardware status & hardware configuration.
	8. Verify that all I/O (disks, filesystems, etc.) are present, still mirrored, etc.	Use appropriate Solstice DiskSuite and Veritas commands.
	9. Verify that all user and system processes are functional.	Use appropriate commands to get state of user and system processes.
	10.Report problem to Sun as an Soft System Hang.	Place service call to Sun Service using documented call process.
	11.Schedule maintenance window as necessary for any service actions.	As recommended by the Sun support Engineer.
System Hard Hang	1. If possible, determine the state of the system LEDs to verify if they are in a normal operating condition.	Document LED state on all individual CPU/Memory Boards, Clock Boards and front panel LEDs.
	2. Determine if there is any existing login sessions that are active. Use these logins before proceeding any further.	Note if login sessions are available, and if so, use Solaris commands to get system status.
	3. Determine if a console login can be initiated using the serial console port (ttya port). Look for errors in /var/adm/messages, etc.	Note that console login via the serial port is working, but network connections are not. Suspect a network problem.
	4. If no access is possible, attempt to send a system break signal via the system console.	Execute OBP commands to gain additional failure information. Refer to Appendix C & D.
	5. Press the XIR button. This button is located at the rear of the xx00 server.	Document the system LEDs and monitor the console for any output.
	6. Power Cycle the system	Power off the system using the key switch and turn key to the Diagnostic position to run maximum level POST. Document any errors.

Problem	Error Response	Follow-up Tasks / Reporting
System Hard Hang (continued)	7. Verify the hardware state of the system.	Collect prtdiag -v for hardware status & hardware configuration.
	8. Verify that all I/O (disks, filesystems, etc.) are present, still mirrored, etc.	Use appropriate Solstice DiskSuite and Veritas commands.
	9. Verify that all user and system processes are functional.	Use appropriate commands to get state of user and system processes.
	10.Report problem to Sun as a **Hard System Hang**.	Place service call to Sun Service using documented call process.
	11.Schedule maintenance window as necessary for any service actions.	As recommended by the Sun support Engineer.

CHAPTER **A**

Enabling Saving System Dump

The following section defines the four basic steps to enable the process of saving a system dump, known as a vmcore file.

For a complete description reference InfoDoc 12031.

A.1 Enabling savecore(1M) and Verifying Disk Space

Note – Frequency Recommendation: This process should be performed on every Sun server.

Step 1: Enable savecore(1M).

The savecore program must be enabled (turned on) in the Solaris scripts:

- Solaris 2.5.1 and 2.6: `/etc/rc2.d/S20sysetup` (Disabled by default)
- Solaris 7 and above: `/etc/rc2.d/S75savecore` (Enabled by default)

▼ Enable savecore in Solaris 2.5.1 or 2.6.

- **Manually un-comment the six lines in** `/etc/rc2.s/S20sysetup`

```
if [ ! -d /var/crash/‘uname -n‘]
  then mkdir -m 0700 -p /var/crash/‘uname -n‘
if
echo ’checking for crash dump...\c ’
```

```
savecore /var/crash/'uname -n'
echo ''
```

▼ Enabling savecore in Solaris 7 and 8

By default, savecore is ENABLED in Solaris 7 and 8.

- **Run the** `dumpadm` **command as root to verify this.**

  The following is example output.

```
# dumpadm
Dump content: kernel pages
Dump device: /dev/dsk/c0t0d0s1 (swap)
Savecore directory: /var/crash/machinename
Savecore enabled: yes
```

Step 2. Verify that there is sufficient swap space to dump memory.

Swap space is used to initially save the dump of system memory. By default Solaris uses the first swap device that is defined. This first swap device is known as the dump device.

For servers, this swap/dump device size should be at least 1Gbyte and preferably 2Gbyte.

To determine the initial swap device that is defined, use the `swap -l` command. The following is an example output.

```
# swap -l
swapfile                 dev   swaplo   blocks     free
/dev/dsk/c0t3d0s0   32,24     16        4097312    4062048
/dev/dsk/c0t1d0s0   32,8      16         4097312    4060576
/dev/dsk/c0t1d0s1   32,9      16         4097312    4065808
```

The amount of space available is taken from the "blocks" column and then multiplied by 512. Taking the blocks from the first entry, c0t3d0s0, we see a device size of ~2Gbyte.

4097312 * 512 = 2,097,823,744 or approximately 2Gbyte.

Step 3. Verify that there is sufficient file system space for vmcore files.

The scripts mentioned above (`/etc/rc2.d/S20sysetup or /etc/rc2.d/S75savecore`) define the directory where savecore will save the resulting vmcore files.

Note – It is recommended that there be a minimum of 1Gbyte of free space on the file system used to store vmcore files.

By default the file system and directory where savecore files will be saved is:

/var/crash/‘uname -n‘

i.e. for the "mysystem" server, the default directory is:

/var/crash/mysystem

The directory/file system specified must have space for the resulting vmcore.

This can be checked by looking at the free space of the file system. Use the `df -k` command to validate this.

```
# df -k /var/crash/‘uname -n‘
```

If there is no space in /var/crash (the default) then any other locally mounted (not NFS) file system can be used.

To change the default directory, the following commands can be used to remove the current savecore dump location, create a new directory in a larger file system, then create a symbolic link to point from the old location to the new location:

```
# cd /var/crash
# rmdir -f mysystem
# mkdir /some_large_filesystem/mysystem
# ln -s /some_large_filesystem/mysystem mysystemhost
```

Step 4. Verify the following savecore patches are applied.

If the swap/dump device shown above is configured over 2Gbyte, the following patches must be applied. Note that the actual system dump (vmcore file) may be much smaller than 2Gbyte, and the issue these patches correct is simply a problem with swap/dump device sizes greater than 2Gbyte.

For simplicity, it is recommended that these patches be applied no matter the size of the swap/dump device.

- Solaris 2.5.1: 108083-01

- Solaris 2.6: 107490-01

For additional detailed swap device configuration and sizing hints, start with these SunWorld Online articles:

- Swap Space part 1

  `http://www.sunworld.com/swol-12-1997/swol-12-insidesolaris.html`

- Swap Space part 2

  `http://www.sunworld.com/swol-01-1998/swol-01-insidesolaris.html`

- Clearing up questions

  `http://www.sunworld.com/sunworldonline/swol-07-1998/swol-07-perf.html`

- Adrian Cockcroft and Richard Pettit's new book.
- Sun Performance and Tuning, 2nd Edition.

A.2 Verify Core Dump Process

The following section describes how to validate that a system is capable of saving a valid system dump (vmcore).

For a complete description, reference Sun InfoDoc 12031 or contact Sun Microsystems for assistance.

Note – Frequency Recommendation: Test once on every machine where a system dump has not already been taken successfully due to an error.

This is normally a task that would be done just prior to placing a system into production.

If you are the system administrator or system owner, you must force your system to crash in order to test your savecore setup. The following describes this process for Solaris releases up to and including Solaris 7.

1. **Back up all of your data.**

   System crashes can result in non-recoverable and catastrophic loss of data.

2. **Gracefully halt your system using 'halt' or 'init 0'.**

3. **At the ok> boot prom prompt enter: sync**

   Your system should start to panic at this time. You should see "dumping" messages.

   Next, the system will attempt to reboot. During this process you should see some savecore messages.

4. **Once the system is rebooted, look in your savecore directory and see if you have system crash dump files there.**

   They will be named "unix.?" and "vmcore.?", where ? is the integer dump number. There should also be a "bounds" file. This contains the next crash number for savecore to use.

For Solaris 8, there is an option (-d) to the reboot(1M) command. This option to reboot causes a system dump to be taken, just prior to the system being rebooted.

1. **Back up all of your data.**

   System crashes can result in non-recoverable and catastrophic loss of data.

2. **Reboot the system using the -d option to reboot:** `reboot -d`

   The system will panic and then attempt to reboot. During this process you should see some savecore messages.

3. **Once the system is rebooted, look in your savecore directory and see if you have system crash dump files there.**

   They will be named "unix.?" and "vmcore.?", where ? is the integer dump number. There should also be a "bounds" file. This contains the next crash number for savecore to use.

CHAPTER **B**

ISCDA Script

The Initial System Crash Dump Analysis script (iscda script) is used to assist in the analysis of Solaris 2.X system crashes. It may also be used on live systems, but this is not normally the mode of execution for iscda

For a complete description, reference Sun InfoDoc 10214 or contact Sun Microsystems for assistance.

Note – Frequency Recommendation: Make script available to every machine. Run the iscda script only when a system panic dump (vmcore file) is produced.

The iscda script has been tested and runs on Solaris 2.3 through 2.6. Iscda also works on Solaris 7, with some "symbol not found" messages and the crash time will be incorrect if you are booted in 64 bit mode.

To prepare a system to run iscda, place the iscda script in the directory which savecore(1M) uses to save the vmcore and unix files. Execute `chmod` so that it is executable as root (you will need to run it as root).

To produce a text file summary analysis of a system dump (vmcore file), execute iscda as follows:

 # iscda unix.? vmcore.? > iscda.out

Where ? is the integer number extension on the vmcore and unix files.

A few examples of when this script may be helpful:

- Due to the size of the vmcore files, transferring them can take time.

  When placing a service call due to a system panic, inform the Sun engineer that you have iscda output. This iscda output can be easily e-mailed, etc. for an initial look into the reason for the system panic.

- Text items in the Solaris panic string and/or stack can be used to search SunSolve for related problems.

Note that some panics have similar appearances with regards to the panic string and stack trace, so any bugs believed to be related should be confirmed by the Sun engineer handling your call.

- If you wish to save a 'summary' of the core file for your own history rather than saving the whole kernel core file which is considerably larger.

Note – Iscda output doesn't take the place of analyzing the entire system dump/vmcore file, however, if the panic is a known issue, this output can sometimes help to reduce the overall troubleshooting time.

CHAPTER C

System Abort Sequences

This section outlines keyboard sequences and procedures. It will give exact syntax as well as excerpts from resultant information.

IMPORTANT! The abort sequence does NOT cause an automatic dump of system memory. If a vmcore file (dump) is the goal of the abort sequence, then make sure a sync command is issued while at the OBP ok prompt. If the sync command is not issued, and the system is rebooted, the state of the system is lost.

Note – Sometimes the local "Stop / a" key sequence or the remote send break sequences below do not work. If there is a local keyboard plugged into the keyboard connection at the rear of the system, attempt to disconnect / re-connect it. This sometimes forces the system to the OBP prompt.

C.1 L1 / a Stop / a Keyboard Abort Sequences (Local Terminal Connection)

IMPORTANT! The L1 / a or Stop / a keyboard sequence only works if you have a graphical based console and a terminal connected directly to the server.

For cases where the serial ttya port is being used as the console interface, refer to the next section regarding how to send a break signal via a terminal server or other device.

The abort sequence is the method whereby a user can immediately drop from running Solaris into kadb or OBP. This method takes the machine to a state where additional information can be gathered. The system can also be recovered or rebooted from this state.

The sequence is generated by holding down (depressing) the "Stop" key and then pressing the lower case "a" key. The "Stop" key is located on the upper left side of the Sun keyboard. It has the word Stop printed on the top and on some Sun keyboards, have L1 printed on the front face of the key as well.

On the console, the following output will be seen.

```
Type ‘go’ to resume
ok go
```

The system should resume executing Solaris.

Or to cause the system to save a dump, issue a sync command.

```
Type ’go’ to resume
ok sync
```

C.2 Abort Sequence via a Serial Terminal Server

This method is to be used when access to the Sun Enterprise xx00 console is via a network terminal server, using the telnet protocol.

```
mysystem% telnet annex_box
    Trying 129.xxx.xxx.xxx...
    Connected to annex_box
    Escape character is ‘^]’
```

```
Enter Annex port name or number 52
Annex username: user_name
Annex password:
Permission granted
Attached to port 52
```

At this point, you should try to login to the system. If the system is hung, etc. you may not be able to login. If so, skip down below to where you enter the telnet command mode.

```
mysystem console login
Unix(r) System V Release 4.0 (mysystem)
login: root
passwd:
Sun Microsystems Inc. SunOS 5.5.1 :02/28/96 May 1996
#
```

Issue the appropriate command sequence to enter the command mode of telnet. Normally this is done by pressing the "Control Key" and the "]" key simultaneously. The "]" key is often referred to as the right bracket key.

From the telnet command prompt, issue the `send brk` command to send a break signal from the terminal server.

```
telnet> send brk
Type ‘go’ to resume
{1b} ok
{1b} ok go
```

The system should resume executing Solaris.

Or to cause the system to save a dump, issue a `sync` command.

```
Type ’go’ to resume
{1b} ok sync
```

C.3 Abort Sequence via a Direct Connect ASCII Terminal

This method is used when access to the Sun Enterprise xx00 console is via an ASCII terminal directly connected to the system console (ttya) serial port.

Note – The key sequence to generate a break can vary, based on the type of ASCII terminal used. Some ASCII terminals have a break key that will generate the necessary sequence. Some ASCII terminals will send a break if the "Control Key" is held down and then a lower case "x" is pressed. For the appropriate key sequence to generate a break, consult the manual for the ASCII terminal being used.

You may or may not see a login prompt at the console screen.

On the console, the following output will be seen.

Type ‘go’ to resume

```
ok go
```

The system should resume executing Solaris.

Or to cause the system to save a dump, issue a sync command.

Type ’go’ to resume

```
ok sync
```

C.4 Remote System XIR and Remote Power Control Sequences

The following terminal key sequences can be executed ONLY with a console that is connected to ttya serial port of the Sun Enterprise xx00 server. In addition, the following criteria must be met:

- The key switch must be in either the On or Diagnostic setting. If it is in the Secure or Off position, the remote key sequences and button resets are ignored.

- Security features (such as OpenBoot security-mode) must be disabled.
- The type speed must be no faster than 0.5 seconds and no slower than 5 seconds between characters.

To force an XIR event, execute the following command. This key sequence is two carriage returns, a tilde, and the key sequence of the Control Key / Shift Key / and the letter x. For example:

<CR> <CR> <~> <Control-Shift-x>

Once the system enters the XIR state, the following command will display XIR data.

.xir-state-all

The XIR failure data from the above command can be diagnosed at:

`http://cte-www.uk/cgi-bin/xir-cgi.tcl`

To power cycle (toggle power off then back on) on a Sun Enterprise xx00 server, the following command sequence is used:

<CR> <CR> <~> <Control-Shift-p>

CHAPTER **D**

OBP & kadb Commands

D.1 OBP Debug Commands

TABLE D-1 show commands that are available at the OBP (ok) prompt. They allow the user the ability to perform some low level data gathering.

These commands are normally executed when a system was running Solaris and either hung and was forced (aborted) into OBP, or the system panic dump process failed and the system is left at the OBP prompt.

TABLE D-1 OBP Debug Commands

registers	Display values in %g0 through %g7, plus %pc, %npc, %psr, %y, %wim, %tbr.
.trap-registers	Display values in the trap related registers.
locals	Display the values in the i, l and o registers.
psr	Display the processor status register.
ctrace	Display Solaris return stack showing C subroutines.
sync	Cause Solaris to attempt to save a system panic (vmcore).

D.2 kadb Debug Commands

If booted under kadb, the system will enter kadb if the system panic'd or if the system was sent an abort sequence. TABLE D-2show commands that are available at the kadb prompt. They allow the user the ability to perform some low level kernel debugging.

Some of this output may be lengthy, so be prepared to log more than just a few lines of information.

TABLE D-2 kadb Debug Commands

<sp$<stacktrace	**Display Solaris kernel stack trace information.**
$<threadlist	Display a list of threads that were executing at the time.
$<thread.brief	Display more thread information.
$<cpus	Display CPU structures.
$<msgbuf	Display the Solaris message buffer.

CHAPTER **E**

System Console Logging

System console logging is the ability to collect and log Sun Enterprise server console output. This output is delivered through the serial port (ttya port) on each of the Sun Enterprise 6x00, 5x00, 4x00, 3x00 servers.

E.1 Purpose for Console Logging

The purpose of console logging is so that the console messages can be captured and used to improve the quality and timeliness of problem diagnosis. Fatal Reset details and POST output after a Fatal Reset is directed to the console.

This console data can result in fewer cases where a system interrupts and no data appears to be recorded. In many of these interrupts, this data is the only output to the console because in some failure modes, Solaris has already terminated and there is no software still running in the system capable of logging messages to traditional file system locations.

For this reason, console logging provides additional diagnostic information and reduces the number of "unexplained system reboots", as the important diagnostic/failure data is captured. This can also ensure that only the defective FRU is replaced.

The following sections outline the possible console logging options. Note that there may be other software and hardware vendors with equivalent products, however, the functionality of these other products should be similar to what is discussed below.

E.2 Configuring the Console on ttya

This functionality takes care of both normal system administration activities and the analysis of system resets. A standard null modem cable is used to connect a serial device (i.e. Network terminal server or ASCII terminal) to the ttya serial port. There are OBP variables that should be set so as to handle this configuration.

There are two ways to enable the console on ttya.

- using the `eeprom` command
- setting the variable directly while in OpenBoot (OBP)

▼ The `eeprom` command.

- The `eeprom` command displays/sets OBP variables.

Output has been deleted to save space.

```
mysystem# /usr/bin/eeprom
output-device=screen
input-device= keyboard
```

- The following `eeprom` commands set the output-device and input-device OBP variables to be ttya (serial console port A).

  mysystem# **`eeprom output-device=ttya`**

  mysystem# **`eeprom input-device=ttya`**

- The `eeprom` command displays OBP variables and their values.

Output has been deleted to save space.

```
mysystem# eeprom
output-device=ttya
input-device=ttya
```

- To revert back to a direct connect monitor and keyboard, execute the following commands.

```
mysystem# eeprom output-device=screen
mysystem# eeprom input-device=keyboard
```

You can verify that the values have been changed back via the `eeprom` command

▼ Setting the variable directly while in OpenBoot (OBP).

- The `printenv` OBP command will show OBP variables and their current setting. Output was deleted to save space.

```
ok> printenv
output-device screen
input-device keyboard
```

- The `setenv` OBP command will set OBP variables.

```
ok> setenv output-device ttya
ok> setenv input-device ttya
ok> printenv
```

```
output-device ttya
input-device keyboard
```

- To revert back to a direct connect monitor and keyboard, execute the following commands.

```
ok> setenv output-device=screen
ok> setenv input-device=keyboard
```

You can verify that the values have been changed back via the `eeprom` command

There are other OBP variables that may require setting, depending upon the type of connection that is being used. The default ttya port characteristics are denoted by the variables listed below. The speed of the ttya port is hard-coded to be 9600 BAUD.

ttya-mode=9600,8,n,1,-

ttya-ignore-cd=true

ttya-rts-dtr-off=false

E.3 Console Logging Options - Data Logging Terminal Servers

A replacement for traditional terminal servers which do not have console logging capability is a console server device from Lightwave, Inc. Lightwave console server is the equivalent to a traditional network based terminal server, however, the Lightwave device has memory added which is used as a "wrap around" message buffer.

As console messages are output from the Sun Enterprise 6x00, 5x00, 4x00, 3x00 servers and Sun SparcServer 1000, 1000E SparcCenter 2000, 2000E servers, they are stored in this memory. As the memory fills up, the oldest messages are overwritten. One can connect to this console server via the network, and then display the contents of the memory buffer for a specific system, thus retrieving the stored console messages.

More information on the Lightwave console sever can be found at:

```
http://www.lightwavecom.com/products/conserver.htm
http://www.lightwavecom.com/products/ConsoleServer800.htm
```

E.4 Console Logging Options - Centralized Console Control

A centralized console control solution is available from Aurora Technologies.

This is a solution that allows a single Sun workstation to serve as a console access and logging point. Hardware is installed in the Sun workstation which supports multiple serial ports and system consoles that are being controlled and monitored via these ports. The workstation can both grant console access as well as log all console activity on its local disk for review at anytime.

More information can be found at:

```
http://www.auratek.com/controltwr/controltwr.html
```

E.5 Console Logging Options - Tip line to ttya

This may be one of the least expensive console logging options, but can create challenges when attempting to monitor multiple systems. The system that is performing the monitoring function must be up and operational, or logging of the other systems console is lost.

To enable this console logging mode, take a standard serial cable and connect one end to the Sun Enterprise Server x000 ttya port on the clock board, then connect the other end of the cable to any serial port on any other local workstation.

Once the cable is connected, a user on this monitoring system can issue the tip command (subject to configurational issues mentioned below) and be connected to the other systems console. Note that prior to issuing the tip command, the user must enable some form of logging, (i.e. using the log to file option of an Xterm session, etc.).

CHAPTER **F**

LED Status Indicators

F.1 Overview

The following tables show the various LED indicators on the xx00 servers. There are LEDs on the servers' Front Panel and on the Clock Board assembly that show overall system or server status. These LEDs are shown in TABLE F-1 and TABLE F-2.

TABLE F-3 show LEDs on the CPU/Memory Boards, I/O Boards and Disk Board LEDs. The LEDs are either yellow or green in color. How they are illuminated determines the specific state or condition of that individual FRU assembly.

TABLE F-4 show LEDs on the power supplies.

TABLE F-1 Sun Enterprise Server Front Panel and Clock Board LED Status

Power LED	Service LED	Cycling LED	Condition
Off	Off	Off	No Power
Off	On	Off	Failure Mode
Off	Off	On	Failure Mode
Off	On	On	Failure Mode
On	Off	Off	Hung in POST/OBP
On	Off	On	Hung in OS
On	On	Off	Hung in POST/OBP Hung in OS/Failed Component
On	On	On	Hung in POST/OBP Hung in OS/Failed Component
On	Off	Flashing	OS Running Normally

TABLE F-1 Sun Enterprise Server Front Panel and Clock Board LED Status

Power LED	Service LED	Cycling LED	Condition
On	On	Flashing	OS Running with Failed Component
On	Flashing	Off	Service LED Flashing Slow Flash = Executing POST Fast Flash = Executing OBP
On	Flashing	On	Service LED Flashing OS or OBP Error

TABLE F-2 Notes for Sun Enterprise Server Front Panel and Clock Board LED Status

LED Name	Location	Note
Power LED	Left (Green)	Should always be on. If all three LEDs are off, suspect power problem. If this LED is in any other state than on and steady, it indicates a problem.
Service LED	Middle (Yellow)	This LED should be off in normal operation. If on, a component is in an error state and you should check check individual board LEDs. A lit service LED does not imply there is an OS-related problem.
Cycling LED	Right (Green)	This LED should be flashing -- this is the normal state.

TABLE F-3 Sun Enterprise CPU/Memory, I/O, and Disk Board LED Status

Power LED	Service LED	Cycling LED	Condition
Off	Off	Off	No Power
Off	On	Off	Failure Mode
Off	Off	On	Failure Mode
Off	On	On	Failure Mode
On	Off	Off	Hung in POST/OBP
On	Off	On	Hung in OS
On	On	Off	Hung in POST/OBP Hung in OS/Failed Component

TABLE F-3 Sun Enterprise CPU/Memory, I/O, and Disk Board LED Status

Power LED	Service LED	Cycling LED	Condition
On	On	On	Hung in POST/OBP Hung in OS/Failed Component
On	Off	Flashing	OS Running Normally
On	On	Flashing	OS Running with Failed
On	Flashing	Off	Service LED Flashing Slow Flash = Executing POST Fast Flash = Executing OBP
On	Flashing	On	Service LED Flashing OS or OBP Error

NOTES: Low Power Mode - If the status of the LEDs on the board is off-on-off, this means the board is in low power mode. This occurs when the board is disabled because it failed POST, or if the board was just inserted. Low power mode is the only state in which you may unplug the board while the system is running.

Disk Boards - The amber LED on disk boards installed in Sun Enterprise servers will remain on when the Sun Enterprise server is running Solaris 2.6 5/98 or above. This is normal, and it indicates the board is in low power mode (the board can be removed from the system provided the disks have been idled).

TABLE F-4 Power Supply LED Status

Green LED	Yellow LED	Note
Off	Off	No AC input or key switch is turned off
On	Off	Normal Operation
On	On	Fan failure or one or more voltages out of specification.
Off	On	One or more DC outputs failed, or voltages out of specification, or system in low power state

CHAPTER **G**

Decoding xx00 Device Paths

Understanding Sun Enterprise Server 6x00 - 3x00 device paths is critical in identifying messages logged, mainly by Solaris, regarding various I/O components in these systems.

G.1 Device Driver Acronyms

The following are the device driver abbreviations that are used in device path names to define various device drivers. These driver abbreviations will be found in the second part of the device path name, as shown below.

fas - driver for fast/wide SCSI controllers (on-board)

hme - driver for Fast Ethernet

isp - driver for differential SCSI controllers and the SunSwift card

glm - driver for UltraSCSI controllers

scsi - driver for Small Computer Serial Interface (SCSI) devices

sf - driver for soc+ or socal Fiber Channel Arbitrated Loop (FCAL)

socal - driver for SPARC Storage Array (SSA) controllers

G.2 Device Path Example

The following example uses a device path from an Sun Enterprise 3000. The path name is broken down into segments to identify the various locations slot, location on the I/O board, etc. For this example, the following device path is used.

/sbus@3,0/SUNW,fas@3,0/sd@0,0

Part 1 - sbus@3,0

The first portion of the device path indicates the I/O board slot location. In the first table below, we see that sbus@3,0 correlates to the I/O board in slot 1,located on the back of the machine (UE 3000's only use the rear, odd number slots because there are internal disks in the front).

Part 2 - SUNW,fas@3,0

The second portion of the device path indicates the type of onboardcontroller and the Sys I/O bus it uses. I/O boards are basically split in half, each half is controlled by a Psycho or Sys I/O chip (A or B). We see that SUNW,fas@3,0 correlates to the onboard (built-in) fast/wide SCSI controller at Sys I/O B on this board.

Part 3 - sd@0,0

The next portion of the device path (sd@0,0) correlates to the SCSI disk (sd) set to target id 0 (in this case an internal disk, since only internal disks should be controlled by the onboard SCSI controller of the I/O board in slot 1).

Another Example:

/sbus@6,0/SUNW,socal@d,0/sf@0,0/ssd@2200002136bcd49,0 (ssd27)

This device path correlates to an I/O board in slot 3 of a UE server (sbus@6), the onboard socal controller (socal@d), the GBIC port (sf@0). The ssd@2200002136bcd49,0 (ssd27) is a disk inside of an A5x00 array. The long number after the '@' sign is the world wide (unique) number of this particular disk.

Note – sf@0 is the GBIC port on the right and sf@1 is the GBIC port on the left when looking at an I/O with soc+ (socal) board. On an Enterprise 3000 & 3500, sf@0 is the GBIC port on the bottom and sf@1 is the GBIC port on the top.

G.3 Device Path Decoding Tables

TABLE G-1 On-Board Controller Definitions - (Part 1 segment)

FRONT		REAR	
Sysio A	Sysio B	Sysio A	Sysio B
soc@d	fas@3	soc@d	fas@3
sbus@1	hme@3	sbus@1	hme@3
sbus@2	sbus@0	sbus@2	sbus@0

In addition to the on-board interfaces, each I/O Board has two SBus slots. TABLE G-2 indicates how the SBus slots are numbered, depending on which I/O Board slot the SBus cards are installed.

TABLE G-2 I/O Board SBus Slot Assignments

I/O Board Slot	SBus #		I/O Board Slot	SBus #	
0	SBus@0	SBus@1	1	SBus@2	SBus@3
2	SBus@4	SBus@5	3	SBus@6	SBus@3
4	SBus@8	SBus@9	5	SBus@a	SBus@b
6	SBus@c	SBus@d	7	SBus@e	SBus@f
8	SBus@10	SBus@11	9	SBus@12	SBus@13
10	SBus@14	SBus@15	11	SBus@12	SBus@17
12	SBus@18	SBus@19	13	SBus@20	SBus@21
14	SBus@1c	SBus@1d	15	SBus@1e	SBus@1f

CHAPTER **H**

Use of prtdiag

The prtdiag(1M) command is used to display the system hardware configuration, details of certain system failures and Power On Self Test results.

The prtdiag command is available on sun4u (Ultra) and sun4d (SPARCcenter 1000 & 2000) system platforms.

- For a complete description reference InfoDoc 23479
- For a complete patch description reference Patch Description 104595
- For a complete patch description reference Patch Description 105642

The following patches are related to prtdiag, but are not critical.

- Solaris 2.5.1 - 104595-09
- Solaris 2.6 - 105642-08

Frequency Recommendation: Prtdiag should be run after every un-planned or unexpected system interrupt, i.e. System panic, etc.

H.1 `prtdiag` Command

The prtdiag command is available on sun4u (Ultra) and sun4d (SPARCserver 1000 & 2000) system platforms. The first part of the prtdiag output displays the overall active hardware configuration of the system (active CPUs and their speed, active Memory and its size, and active I/O controllers and their type).

Additional information, including firmware levels, diagnostic result information, etc. are displayed using the verbose or "-v" option.

The diagnostic portion of the prtdiag output displays any failed Field Replaceable Units (FRU's) in the system on most sun4u model servers.

H.2 Location of the `prtdiag` Command (sun4u and sun4d)

The `prtdiag` command, is available on the sun4u (Sun Enterprise 6x00 - 3x00) system platforms, as well as several other Sun server models.

To run the `prtdiag` command, one will need to know which system platform they are on because the directory path to the `prtdiag` command differs slightly, between a sun4u and a sun4d platform.

To find out the system platform of a particular system, issue the `uname -m` command.

If the system is a sun4u platform type, the `prtdiag` command resides in the following directory:

/usr/platform/sun4u/sbin

H.3 Use of the `prtdiag` Command (sun4u)

The following shows how to execute the prtdiag command, on a sun4u class server. Note that for the most information, use of the -v option is recommended.

```
# /usr/platform/sun4u/sbin/prtdiag -v
```

CHAPTER I

Using the Explorer Utility

The Explorer script can be obtained from Sun Enterprise Services. If there are specific questions regarding explorer, a Sun service call should be opened so that the question(s) can be answered, etc.

Frequency Recommendation: To be installed on every system and run once a month, or when any update is made to the system.

I.1 Installing Explorer

To install and run Explorer please perform the following steps:

Note – Explorer requires root privileges to run and root must have privileges to write to the directory in which you install the explorer package.

To install the Sun Explorer package, transfer the compressed tar file to a local file system on the server. Then uncompress, untar, and install the Explorer package. The following are example steps:

```
# zcat SUNWexplo.tar.Z | tar xf -
# pkgadd -d . SUNWexplo
```

During installation you will be asked several questions, including:

- Your Company Name
- Your contract ID
- This systems serial number, etc.

Sun asks that every effort be made to supply your ContractID, Serial number, and Company Name, as this enables us to track the explorer output more effectively.

I.2 Executing Explorer

To run explorer manually after the install process, perform the following steps as root:

```
# cd /opt/SUNWexplo
# ./explorer -mail(To automatically e-mail the results)
-or-
# ./explorer(To save a copy of the explorer output locally)
```

A copy of the explorer output is placed in /opt/SUNWexplo for your reference.

If you are asked to send explorer output to Sun with regards to an active problem, you will be given instructions on how to address the e-mail, which explorer file to send and where to send it to.

The Explorer output may also be sent to the Sun ftp site if it is too large to be e-mailed.

CHAPTER **J**

FTP Access to Sun

The following describe the procedures to transfer information (i.e. a vmcore file, etc.) to Sun Microsystems external ftp sever in the USA..

J.1 Instructions for USA

The following are instructions to ftp a file(s) to Sun Microsystems external ftp server known as sunsolve.sun.com.

Important: After ftp'ing any file it is necessary to inform the Sun Solution Center and/or Sun Engineer you are working with that the file has been sent. This is necessary because the files sent to Sun's anonymous ftp server are erased every two days. Thus files can be deleted before they are retrieved by Sun support engineers.

1. **What files to send.**

   Depending on the problem, you will be directed to gather certain files and prepare them to be sent to Sun via the anonymous ftp server, sunsolve.sun.com.

   Note that the problem category definition section of this document also has recommendations on the specific files that should be made available and be prepared to send based on the Sun Solution Center engineer or Sun field engineer request.

2. **How to prepare the files for sending:**

   The best way to do this is by creating a tar file named <SunCase#>.tar containing the required files and then compressing the file. Compressing the files is important, especially if you are sending system dumps (vmcore files) as these can be hundreds of MB's in size.

   The following is an example of how to create a tar file archive, assuming a Sun assigned problem case number of 62221234:

```
# tar -cvf 62221234.tar file_names_to_send
```

After the tar file is created, please compress the tar file. The preferred method is to use GNU Zip, which uses a tighter compression algorithm.

```
# gzip 62221234.tar
```

If you do not have gzip, use the standard Solaris compress command

```
# compress 62221234.tar
```

3. ftp the files to the Sun external ftp site:

Note that the standard ftp command may not work in all firewall situations or there may be ftp proxy servers in place. Simply use the ftp procedure specific to your location for external (outside the company) ftp transmissions.

```
# ftp sunsolve.sun.com
```

or

```
# ftp 192.18.99.148
```

Once connected, you can use "anonymous" login.

```
login:  anonymous
password:   your e-mail address
ftp> cd cores
ftp> binary
ftp> put Case#.tar.gz(for example:  62221234.tar.gz)
-or-
ftp> put Case#.tar.Z(for example:  62221234.tar.Z)
ftp> quit
```

CHAPTER **K**

SunVTS

SunVTS, Sun Validation Test Suite, tests and validates Sun hardware by verifying the connectivity and functionality of hardware devices, controllers and peripherals. SunVTS is used by OPs, OEMs, SunService/Field, Design Engineering, SQA and end users to ensure a "clean bill of health" of the overall system under test and its underlying hardware. The tool is used for both Hardware validation and repair verification. Several features are incorporated into SunVTS to enhance the ability of the tool to diagnose systems. SunVTS makes an effective diagnostics tool as it strives to stimulate, detect and identify hardware faults.

Frequency Recommendation: SunVTS should be loaded on every system and be used to certify a system prior to entering production. SunVTS may also be used as necessary when instructed by a Sun service engineer.

K.1 SunVTS Versions

When installing SunVTS it is important to realize this validation tool is OS version and revision specific. TABLE K-1 shows the current versions and patches necessary for each Solaris release.

TABLE K-1 SunVTS Version Matrix

Solaris Release	Sun VTS Version	SunVTS Patches
Solaris 8 1/01 (Update 3)	SunVTS 4.2	N/A
Solaris 8 10/00 (Update 2)	SunVTS 4.1	110353-xx
Solaris 8	SunVTS 4.0	110041-xx 110046-xx

TABLE K-1 SunVTS Version Matrix

Solaris Release	Sun VTS Version	SunVTS Patches
Solaris 7 11/99	SunVTS 3.4	109930-xx 110010-xx
Solaris 7 8/99	SunVTS 3.3	110039-xx
Solaris 7 5/99	SunVTS 3.2	110040-xx 110160-xx 107732-xx
Solaris 7 3/99	SunVTS 3.1	107890-xx
Solaris 7	SunVTS 3.0	107543-xx
Solaris 2.6 - 5/98	SunVTS 2.1.3	106387-xx 106810-xx 107135-xx 107542-xx
Solaris 2.6 - 3/98	SunVTS 2.1.2	106140-xx 106483-xx
Solaris 2.5.1 - 11/97	SunVTS 2.1.1	105799-xx 106073-xx 106247-xx 106528-xx 107237-xx
Solaris 2.6	SunVTS 2.1, REV=37.97.07.16	106114-xx
Solaris 2.5.1 - 8/97	SunVTS 2.1, REV=37.97.06.27	106250-xx
Solaris 2.5.1 - 4/97	SunVTS 2.0.1	105061-xx 105221-xx 105331-xx
Solaris 2.5.1	SunVTS 2.0	104774-xx 105673-xx
Solaris 2.5 SHWP (HW 1/96)	SunVTS 1.0	103097-xx 106703-xx
Solaris 2.5	SunVTS 1.0	103097-xx 106703-xx

CHAPTER L

Forth Debug

IMPORTANT! Enabling Forth debug mode should ONLY be set based on specific direction from Sun Microsystems support engineers. Setting Forth debug mode should NOT be default on all systems.

L.1 Forth Debug - Explanation and Setting

Forth Debug is a tool that is used while in OBP. Forth Debug is roughly analogous to adb/kadb.

For the system to have this tool enabled, there must be entries made in the /etc/system file in the user-settable variables section. The system to be analyzed must also be running a debug kernel.

set forthdebug=1

set obpdebug=1

A subsequent system reboot will show the following strings which verify that forth debug is enabled. The command, /usr/bin/dmesg, is one way to verify.

obpsym: symbolic debugging is available.

Read 72209 bytes from misc/forthdebug

With this feature enabled, additional analysis techniques are available to whomever is attempting system fault analysis on the Sun Enterprise Server. As in the case of adb, additional user-defined ‘‘forth debug macros’’ can be created and used.

CHAPTER **M**

Sun Enterprise xx00 Firmware Levels

The following information details how to determine the level of Sun firmware (OpenBoot and POST) that is installed on a server. Also included is a matrix of the latest released firmware versions based on server platform type.

Frequency Recommendation: Keep the flashprom up to date on all boards in every machine.

M.1 Purpose of Flash Prom Updates

Flashprom updates offer problem fixes, feature enhancements, and changes in the following areas:

- Ability to support new hardware (i.e. CPUs, Memory, etc.)
- Changes that affect prom level messaging.
- Fixes to areas such as OpenBoot (prom level code) and the Power On Self Test (POST).

As such, flashprom patches are considered critical patches and should be updated when new versions are available.

It is possible to determine the Open Boot Prom (OBP) revision of most SPARCstations from the UNIX prompt using the `prtconf` or `prtdiag -v` commands. This allows the firmware version to be checked without system downtime.

The following is an example of the flashprom section of the `prtdiag -v` output. In this example, the flashprom is at the 3.2.28 level.

```
# prtdiag -v
```

<extra output deleted for readability>

System Board PROM revisions:

```
Board 0:   OBP   3.2.28 2000/12/18 11:15   POST  3.9.28 2000/12/18
11:19
Board 1:   FCODE 1.8.28 2000/12/18 11:14   iPOST 3.4.28 2000/12/18
11:18
Board 2:   OBP   3.2.28 2000/12/18 11:15   POST  3.9.28 2000/12/18
11:19
Board 3:   FCODE 1.8.28 2000/12/18 11:14   iPOST 3.4.28 2000/12/18
11:18
Board 4:   OBP   3.2.28 2000/12/18 11:15   POST  3.9.28 2000/12/18
11:19
Board 5:   OBP   3.2.28 2000/12/18 11:15   POST  3.9.28 2000/12/18
11:19
Board 6:   OBP   3.2.28 2000/12/18 11:15   POST  3.9.28 2000/12/18
11:19
Board 7:   OBP   3.2.28 2000/12/18 11:15   POST  3.9.28 2000/12/18
11:19
```

For complete details regarding the problems that a flashprom upgrade resolves, reference the firmware patch 103346-xx README file.

For questions regarding Sun flashprom versions, please contact Sun Microsystems for additional information.

M.2 Flashprom Version Matrix

TABLE M-1 show the firmware level matrix for a variety of Sun systems. This table is current as of 12/11/00.

TABLE M-1 Sun Flashprom Version/Patch Matrix

Release Date	Machine Type	Patch ID
3/7/00	Ultra 1E	104288-xx
3/7/00	Ultra 1 (non-E)	104881-xx
3/7/00	Ultra 2	104169-xx
7/19/00	Ultra 5/10	106121-xx
3/7/00	Ultra 30	105930-xx
3/7/00	Ultra 60 E220R	106455-xx
3/7/00	Ultra 80 E420R	109082-xx
5/3/00	UE250	106503-xx
2/2/00	UE450	106122-xx
10/4/00	E3x00 - E6x00	103346-xx
8/7/00	Netra t1	108673-xx

CHAPTER **N**

ESD Handling & Tools

For more information on ESD and procedures, refer to Chapter Three of the *Sun Microsystems Data Center Site Planning Guide* (805-5863) or refer to `edist.corp` web site.

Frequency Recommendation: Proper ESD procedures should be followed whenever working on a any Sun system. ESD friendly environments should exist where all Sun systems are installed and/or service operations occur.

N.1 Electrostatic Discharge (ESD)

Electrostatic Discharge (ESD) can be extremely damaging to electronic components. By definition, static electricity is an electrical charge at rest. The discharge of this built-up energy can cause numerous problems.

N.2 ESD Damage

Today's electronic equipment has a much denser component geometry, and is composed of thinner, more easily damaged materials. Changes in the design, manufacturing process, and materials used has improved ESD sensitivity considerably.

While Sun equipment has been designed to be tolerant of some ESD events, it is important to take precautions in the design of the computer room to minimize exposure to discharges. This is particularly important when unprotected components are being handled during installations or upgrades.

Damage caused by ESD can take the form of catastrophic failures, but is more often low-grade damage that may not show up during initial installation or upgrades. However, damage caused by ESD can make the system more susceptible to a later failure. Cumulative degradation of the components can also occur as the result of repeated, low voltage exposures. These types of problems are very subtle and extremely difficult to detect.

N.3 ESD Control

A detailed site-specific evaluation should be conducted to determine the most appropriate ESD program for each controlled area. The following is a list of ways to control static generation and ESD:

- Use appropriate personal grounding equipment. Use of appropriate personal grounding equipment (wrist straps, heel grounders, etc.) by operators in contact with sensitive components can lessen the likelihood of human instigation of ESD.
- Close cabinet covers at all times. Covers should only be opened by trained personnel using proper grounding when inspections, repairs or reconfigurations are needed.
- Have a properly grounded access floor system with static dissipative tile surfaces. This will provide a proper path to ground.
- Use appropriate cleaning agents. The use of inappropriate cleaning agents or excessive build-up of contaminants on the floor grid can allow an insulating barrier to accumulate, thereby affecting the static dissipative capabilities of the floor.
- Use appropriate carts and furniture in the room. Use of appropriate carts and furniture in the room will significantly decrease the risk of ESD since the movement of inappropriate chairs or carts can easily generate static charges.
- Room ionizers may be appropriate to help neutralize static charges in manufacturing areas or print rooms. An ionizer emits negative and positive ions that are attracted to charged objects, and can neutralize them.
- Isolate activities or hardware that are likely to generate static charges. Isolation of activities or hardware that are likely to generate static charges can decrease the risks to more sensitive hardware. This is one reason why print equipment (copiers, printers, etc.) should be kept out of the computer room.
- Maintain appropriate relative humidity levels. Appropriate moisture levels will help ease the dissipation of charges, lessening the likelihood of catastrophic failures. The following chart illustrates the effect moisture levels can have on electrostatic charge generation.

N.4 Electrostatic Voltage At Workstations

Reference: Data Center Site Planning Guide on sun.com

TABLE N-1 Static Voltage

Means Of Static Generation	Relative Humidity 10-20%	Relative Humidity 65-90%
Walking Across Carpet	35000	1500
Walking over vinyl floor	12000	250
Worker at bench	6000	100
Vinyl envelopes for work instructions	7000	600
Common Poly bag picked up from bench	20000	1200
Work chair padded with urethane foam	18000	1500