Ultra Enterprise 10000 SSP 3.1 Reference Manual

Sun Microsystems Computer Company
A Sun Microsystems, Inc. Business
901 San Antonio Road
Palo Alto, CA 94303 USA
415 960-1300 fax 415 969-9131
U.S.A.

Part No: 805-3362-10
Revision A, December 1997

NAME

Intro – SSP files, etc.

DESCRIPTION

This section contains miscellaneous man pages for files, scripts, etc. that execute in the SSP environment.

blacklist(4)	list of system resources not to be booted
cb_config(4)	list of machines managed by the SSP
cb_port(4)	list of communication ports for cbe
domain_config(4)	list / description of configured domains
domain_history(4)	list / description of removed domains
edd.emc(4)	event monitor file
edd.erc(4)	event response file
fad_files(4)	file access daemon files
postrc(4)	**hpost**(1M) properties file
redlist(4)	list of system resources not to be touched
ssp_resource(4)	SSP processes resource file
ssp_to_domain_hosts(4)	hostname / domainname file

NAME

blacklist – list of system resources not to be booted

DESCRIPTION

$SSPVAR/etc/*platform_name***/blacklist** is an ASCII file that enables the system administrator (or root) to restrict, from the SSP, the configuration of the host system. It lists components POST cannot use at boot time. POST reads the **blacklist** file before preparing the system for booting, and passes along to OBP a list of only those components that have been successfully tested; those on the blacklist are excluded.

In the **blacklist** file:

- Keywords are not case-sensitive.
- Any part of a line that starts with **#** is a comment.
- Numbers are assumed decimal unless preceded by **0x**, which indicates hexadecimal. Exception: a board number entered as one of [a - f] or [A - F] is assumed hexadecimal.
- Each line has one and only one keyword.
- The same keyword can be used on more than one line.
- Each keyword has one or more arguments. Each argument is shown as an integer and multiple integers are separated by a period.

Keywords

All value ranges shown below are inclusive.

sysbd *board*

Do not test or configure the specified system board, where *board* is an integer, 0 to 15.

proc *board.pmod*

Do not test or configure the specified processor within the specified system board, where *board* is an integer, 0 to 15 and *pmod* is an integer, 0 to 3.

abus *abus*

Do not test or configure the specified address bus, where *abus* is an integer, 0 to 3. The meaning of this command is that the corresponding CIC ASIC on all system boards is marked black. See the keyword **cic**, below.

dbus *dbus*

Do not test or configure the specified 72-bit half of the 144-bit data router, where *dbus* is an integer, 0 or 1. The meaning of this command is that the corresponding half of the local data router on all system boards is marked black. See the keyword **ldpath**, below.

ioc *board.ioctl*

Do not test or configure the specified I/O controller on the specified system board, where *board* is an integer, 0 to 15 and *ioctl* is an integer, 0 or 1.

scard *board.ioctl.slot*

Do not test or configure the specified I/O adapter card within the specified I/O controller, which is within a system board. *board* is an integer, 0 to 15, *ioctl* is an integer, 0 or 1 and *slot* is an integer, 0 to 3.

mem *board*
Do not test or configure memory on the specified system board, where *board* = 0 to 15.

mgroup *board.group*
Do not test or configure the specified group of memory DIMMs within the specified system board, where *board* is an integer, 0 to 15 and *group* is an integer, 0 to 3.

mlimit *board.group.MBytes*
Restrict memory configuration of the specified group of DIMMs on the specified system board to the specified value (which is less than its actual value), where *board* is an integer, 0 to 15, *group* is an integer, 0 to 3 and MBytes is an integer, 64 or 256. This keyword may be useful in benchmarking where, for example, one might simulate memory with 8MB DIMMs (a 64MB group) on a machine that actually has 32MB DIMMs (a 256MB group). This level of control granularity is required because one board with 256MB memory groups behaves differently than four boards each with 64MB memory groups.

cplane *half_centerplane*
Do not test or configure the specified Enterprise 10000 half-centerplane, which contains two address buses and 72 bits of the global data router, where *half_centerplane* is an integer, 0 or 1. The meaning of this command is equivalent to the combination of the **abus** and **dbus** commands for the buses contained in this half-centerplane.

pc *board.pc*
Do not test or configure the specified port controller ASIC within the specified system board, where *board* is an integer, 0 to 15 and *pc* is an integer, 0 to 2.

xdb *board.xdb*
Do not test or configure the data buffer ASIC within the specified system board, where *board* is an integer, 0 to 15 and *xdb* is an integer, 0 to 3.

cic *board.cic*
Do not test or configure the specified coherent interface controller ASIC within the specified system board, where *board* is an integer, 0 to 15 and *cic* is an integer, 0 to 3. (*cic* corresponds to an address bus on that board).

ldpath *board.dbus*
Do not test or configure the specified 72-bit half of the 144-bit local data router within the specified system board, where *board* is an integer, 0 to 15 and *dbus* is an integer, 0 or 1.

EXAMPLE

```
# Sun Microsystems, Inc.
sysbd 3 5 0xA     # Disable system boards 3, 5 and 10.
sysbd 3 5 A       # Disable system boards 3, 5 and 10.
PROC 4.0 6.2      # Disable Processor 0 on System Board 4, and
                  # Processor 2 on System Board 6.
```

```
ScarD 3.0.1       # Disable I/O Adapter 1 on I/O Controller 0 on
                  # System Board 3.
mem 2             # Disable all memory on System Board 2.
mlimit 0xE.2.64   # Restrict use of Memory DIMM Group 2 on System
                  # Board 14 to 64MB.
cIc 1.2           # Disable CIC ASIC 2 on System Board 1.
```

SEE ALSO

hpost(1M), **redlist**(4)

NAME

cb_config – list of machines managed by the SSP

DESCRIPTION

Note: Do not edit this file manually. It is automatically maintained by domain-management tools and commands. To make a control board change, use **ssp_config**(1M) with its **cb** option.

The **$SSPVAR/.ssp_private/cb_config** file identifies the machines that are managed by the SSP and contains the names of both the primary and secondary control boards. Each one-line entry in this file represents one machine, and has the following colon-separated fields. (Spaces have been added to the following example for clarity. The file must NOT contain these spaces.)

platform_name : *platform_type* : **cb0**_*hostname* : *status0* : **cb1**_*hostname* : *status1*

where:

platform_name	The name of the Enterprise 10000 machine. This is a logical name only and does not represent a bootable domain.
platform_type	The platform type of the machine. In this release the *platform_type* is always **Ultra-Enterprise-10000**.
cb0_*hostname*	The hostname that has been assigned to the control board in Slot 0 of the machine.
status0	A value that indicates whether cb0 is the primary control board. If the value of *status0* is **P**, cb0 is the primary; if its value is anything else, it is not.
cb1_*hostname*	The hostname that has been assigned to the control board in Slot 1 of the machine.
status1	A value that indicates whether cb1 is the primary control board. If the value of *status0* is **P**, the *status1* field should be empty. If *status0* is anything other than **P**, *status1* should be **P**.

If one of the control board hostnames is missing, the corresponding status field should also be empty.

EXAMPLES

```
xf2:Ultra-Enterprise-10000:xf2-cb0:P:xf2-cb1:
```

In platform xf2, xf2-cb0 is the control board in slot 0, and it is configured as the primary control board. An alternate control board, xf2-cb1, is in slot 1.

FILES

$SSPVAR/.ssp_private/cb_config

SEE ALSO

cbs(1M), **ssp_config**(1M), **domain_config**(4)

NAME

cb_port – list of communication port for cbe

DESCRIPTION

The **$SSPVAR/.ssp_private/cb_port** file identifies the input / output communication port number for the control board executive (**cbe**). **cbe** is a program that runs on the control board, servicing JTAG and other requests. **cbs**(1M), the control board server, is the only SSP program that communicates with **cbe**.

This file contains the input / output port number on a single line.

FILES

$SSPVAR/.ssp_private/cb_port

SEE_ALSO

cbs(1M)

NAME

domain_config – list / description of configured domains

DESCRIPTION

Caution: Do not edit this file manually. It is automatically maintained by domain-management tools and commands. Editing it manually can cause catastrophic system failure.

The **domain_config** file contains a series of one-line entries, each of which represents a currently configured domain that was created via **domain_create**(1M). **cbs**(1M) uses this file to determine which platform it is to manage. The **domain_status**(1M) command displays the file's contents.

The number of domains that can be listed is unlimited Each line in the **domain_config** file appears in the following form:

domain_name : platform_type : platform_name : os_version : idn_info : sysbds

where:

domain_name	The name of a configured domain. This name must be unique across all physical platforms supported by the SSP where this domain configuration file exists.
platform_type	The type of the system on which the domain resides. Currently, the platform type is always **Ultra-Enterprise-10000**.
platform_name	The name of Enterprise 10000 system on which the domain resides.
os_version	The version of Solaris for the Enterprise 10000 that the domain is running (for example, 2.5.1).
idn_info	Identification information use by Inter-Domain Network software to determine which domains are members of the same IDN-based network. (IDN is available on Ultra Enterprise 10000 servers running Solaris 2.6 or later.)
sysbds	The system boards assigned to the domain, listed by board number and separated by spaces.

EXAMPLES

```
marvin : Ultra-Enterprise-10000 : production : 2.4.3.9 : 2 4 6 8 10

nebula : Ultra-Enterprise-10000 : test : 2.5.0.1 : 12 14 15
```

FILES

$SSPVAR/.ssp_private/domain_config

SEE ALSO

cbs(1M), **domain_create**(1M), **domain_remove**(1M), **domain_status**(1M) in the *Ultra Enterprise 10000 SSP Reference Manual*

drain(1M), **init_attach**(1M) in the *Dynamic Reconfiguration Reference Manual*

NAME

domain_history – list / description of removed domains

DESCRIPTION

Note: Do not edit this file manually. It is automatically maintained by domain-management tools and commands.

The **domain_history** file contains a series of one-line entries, each of which represents a domain that has been removed from the system via the **domain_remove**(1M) command. The **domain_create**(1M) command consults the **domain_history** file to determine whether the domain to be created has existed before.

For a list of currently configured domains, see **domain_config**(4).

Each line in the **domain_history** file appears in the following form:

domain_name : platform_type : platform_name : os_version : idn_info : sysbds

where:

domain_name	The name of the removed domain.
platform_type	The type of system on which the domain was created. Currently, the platform type is always **Ultra-Enterprise-10000**.
platform_name	The name of Enterprise 10000 system on which the domain was created.
os_version	The version of Solaris for the Enterprise 10000 that the domain was running (for example, 2.5.1).
idn_info	Identification information use by Inter-Domain Network software to determine which domains are members of the same IDN-based network. (IDN is available on Ultra Enterprise 10000 servers running Solaris 2.6 or later.)
sysbds	The system boards that were assigned to the domain, listed by board number and separated by spaces.

EXAMPLE

```
nebula : Ultra-Enterprise-10000 : igor : 2.5 : 4 6 8
```

FILES

$SSPVAR/.ssp_private/domain_history

SEE ALSO

domain_create(1M), **domain_remove**(1M), **domain_config**(4)

NAME

edd.emc – event monitor configuration file

DESCRIPTION

The **edd.emc** file is an ASCII file that specifies how the system monitors for certain events. Each system has only one EMC file.

The EMC file is generated from a template file upon invocation of the **ssp_config**(1M) command. The template file resides in **$SSPVAR/.ssp_private/templates/Ultra-Enterprise-10000**, and is named **edd.emc**.

The EMC file contains a series of lines in the following format:

event_type : load_event_monit

where:

event_type A mnemonic (name string) which corresponds to an event type. See **EXAMPLE**, below.

load_event_monit
A keyword, either **enabled**, which tells the system to load the event-monitoring script for the event type on the CBE, or **disabled**, which tells it not to.

The fields are separated by a single colon with or without a single space to its right and left. Words or characters that follow a pound sign (**#**) are treated as comments and are not parsed. The information in the **edd.emc** file is organized as follows:

System Board Temperature Events
System Board Voltage Events

Control Board Temperature Events
Control Board Voltage Events

Centerplane Temperature Events
Centerplane Voltage Events

Centerplane Support Board Temperature Events
Centerplane Support Board Voltage Events

Host Recovery Events

Other Events

System Board Temperature Events

```
sys_brd_temp_norm : enabled   #over-temp readings that go back to normal
sys_brd_temp_warn : enabled   #brd temp which crosses warning threshold
sys_brd_temp_max : enabled    #brd temp which crosses maximum threshold
sys_brd_temp_911 : enabled    #brd temp which crosses 911 threshold
sys_brd_temp_bad : enabled    #unable to obtain brd temperatures
sys_brd_temp_change : enabled #delta change in brd temperature
```

System Board Voltage Events

```
sys_brd_volt_norm : enabled   #max/min/bad voltage readings which go normal
sys_brd_volt_max : enabled    #brd voltage which crosses maximum threshold
sys_brd_volt_min : enabled    #brd voltage which crosses minimum threshold
sys_brd_volt_bad : enabled    #unable to obtain brd voltage values
sys_brd_volt_change : enabled   #delta change in brd voltage
```

Control Board Temperature Events

```
cb_temp_norm : enabled        #see system board temperature descriptions
cb_temp_warn : enabled
cb_temp_max : enabled
cb_temp_911 : enabled
cb_temp_bad : enabled
cb_temp_change : enabled
```

Control Board Voltage Events

```
cb_volt_norm : enabled        #see system board voltage descriptions
cb_volt_max : enabled
cb_volt_min : enabled
cb_volt_bad : enabled
cb_volt_change : enabled
```

Centerplane Temperature Events

```
centerplane_temp_norm : enabled      #see system board temperature descriptions
centerplane_temp_warn : enabled
centerplane_temp_max : enabled
centerplane_temp_911 : enabled
centerplane_temp_bad : enabled
centerplane_temp_change : enabled
```

Centerplane Voltage Events

```
centerplane_volt_norm : enabled      #see system board voltage descriptions
centerplane_volt_max : enabled
centerplane_volt_min : enabled
centerplane_volt_bad : enabled
centerplane_volt_change : enabled
```

Centerplane Support Board Temperature Events

```
supp_brd_temp_norm : enabled  #see system board temperature descriptions
supp_brd_temp_warn : enabled
supp_brd_temp_max : enabled
supp_brd_temp_911 : enabled
supp_brd_temp_bad : enabled
supp_brd_temp_change : enabled
```

Centerplane Support Board Voltage Events

```
supp_brd_volt_norm : enabled  #see system board voltage descriptions
supp_brd_volt_max : enabled
supp_brd_volt_min : enabled
supp_brd_volt_bad : enabled
supp_brd_volt_change : enabled
```

Event Types for System Boards, etc.

Certain event types are common among the different components listed above. The following list is organized by event type.

`temp_norm`
: When the temperature of a board goes from over-temperature back to normal.

`temp_bad`

This event simply reports that temperature readings were not obtainable.

`temp_change`
: When the temperature readings of critical components on the board have changed by a predefined delta (see Predefined Values, below).

`volt_norm`
: When the voltage reading of a board goes from a maximum or minimum voltage reading back to normal.

`volt_max`
: When the voltage reading of the board crosses a predefined maximum threshold value (see Predefined Values, below)

`volt_min`
: When the voltage reading of the board crosses a predefined minimum threshold value (see Predefined Values, below).

`volt_bad`
: When the voltage readings are not obtainable.

`volt_change`:
: When the voltage readings on the board have changed by a predefined delta (see Predefined Values, below).

IDN Auto-Link Support

Note that IDN is supported only on Ultra Enterprise 10000 systems running Solaris 2.6 or later.

```
idn_boot : enabled            #cpu signature states indicate idn_boot
idn_halt : enabled            #cpu signature states indicate idn_halt
cluster_arbstop : enabled     #idn cluster arbitration stop condition
cluster_recordstop : enabled  #idn cluster record stop condition
```

IDN Event Types

`idn_boot`
: When a domain supporting IDN has booted and the respective IDN driver is loaded. This event indicates to **edd**(1M) that automatic IDN linking of the respective domain with other members within the same IDN is necessary, provided those other member domains have also been booted.

`idn_halt`
: When an IDN driver that was previously loaded has been unloaded in the respective domain. This event works in conjunction with the `idn_boot` event to synchronize automatic IDN linking of domains.

`cluster_arbstop`
: When domains within an IDN experience a arbstop. In the standard arbstop event, only state from the boards within the given domain is saved. However, in a `cluster_arbstop`, the state of all boards from all domains within the IDN is saved.

`cluster_recordstop`
: When domains within an IDN experience a recordstop. In the standard recordstop event, only state from the boards within the given domain is saved.

However, in a `cluster_recordstop`, the state of all boards from all domains within the IDN is saved.

Host Recovery Recovery Events

```
arbstop : enabled             #arbstop condition on a sys brd in a domain
recordstop : enabled          #recordstop condition on a sys brd in a domain
watchdog : enabled            #watchdog condition on a sys brd in a domain
environment_shutdown : enabled
reboot : enabled              #cpu signature states indicate reboot condition
panic1 : enabled              #cpu signature states indicate panic1 condition
panic2 : enabled              #cpu signature states indicate panic2 condition
panic_reboot : enabled        #cpu sig states indicate panic reboot condition
obpbooting : enabled          #cpu sig states indicate obp/booting condition
heartbeat_failure : enabled   #cpu heartbeat bits indicate heartbeat failure
```

Host Recovery Event Types

`arbstop`
: When a system board that belongs to a particular domain experiences an arbitration stop condition.

`recordstop`
: When a system board that belongs to a particular domain experiences a record stop condition.

`watchdog`
: When a processor (or set of processors) that belongs to a particular domain experiences a watch-dog condition.

`environment_shutdown`
: When a processor (or set of processors) that belongs to a particular domain experiences an environmental shutdown condition.

`reboot`
: When a processor (or set of processors) that belongs to a particular domain experiences a reboot condition.

`panic1`
: When a processor (or set of processors) that belongs to a particular domain experiences a panic1 condition.

`panic2`
: When a processor (or set of processors) that belongs to a particular domain experiences a panic2 condition.

`panic_reboot`
: When a processor (or set of processors) that belongs to a particular domain experiences a panic reboot condition.

`obpbooting`
: When a processor (or set of processors) that belongs to a particular domain goes into OBP booting. This condition occurs when the domain is in the midst of booting.

`heartbeat_failure`
: When all the processors that belong to a particular domain experience a heartbeat failure condition.

Other Events

```
signature_change : enabled    #cpu signature has changed states
system_config_change : enabled#any machine module is inserted/removed
sys_brd_power_on : enabled    #a system board has been powered on
sys_brd_power_off : enabled   #a system board has been powered off
supp_brd_power_on : enabled   #a support board has been powered on
supp_brd_power_off : enabled  #a support board has been powered off
bulk_power_norm : enabled     #bulk p. supply has gone from Failed/OFF to ON
bulk_power_fail : enabled     #bulk p. supply has gone from ON to Failed/OFF
fan_norm : enabled            #fan has gone from Failed to ON or OFF
fan_fail : enabled            #fan has gone from OFF/ON to Failed
```

Event Types for Other Events

`signature_change`
: When a processor's signature block changes state.

`system_config_change`
: This event describes a situation where any system board, centerplane support board, control board, fan tray, and / or bulk power supply is removed or inserted into the system.

`sys_brd_power_on`
: When the power to a system board has switched from off to on.

`sys_brd_power_off`
: When the power to a system board has switched from on to off.

`supp_brd_power_on`
: When the power to a centerplane support board has switched from off to on.

`supp_brd_power_off`
: When the power to a centerplane support board has switched from on to off.

`bulk_power_norm`
: When a bulk power supply has gone from a off or failed state to on.

`bulk_power_fail`
: When a bulk power supply has gone from on to off or failed state.

`fan_norm:`
: When a fan has gone from a failed state to on or off state.

`fan_fail:`
: When a fan has gone from a on or off state to failed state.

Predefined Values

Predefined threshold values for over-temperature readings, delta values for temperature readings, and delta values for voltage readings are stored in the SSP's persistent store area; see **ssp_resource**(4).

EXAMPLE

```
#
# Event Monitor Configuration File
#
centerplane_temp_warn : enabled
centerplane_volt_max  : enabled
```

The above example tells the system to enable the two centerplane event types for monitoring.

FILES

$SSPVAR/etc/*platform_name***/edd.emc**

The path to an instantiated EMC file.

SEE ALSO

edd(1M), **ssp_config**(1M), **edd.erc**(4)

NAME

edd.erc – event response configuration file

DESCRIPTION

edd.erc files are ASCII files that specify how the system is to respond to certain events. Each domain has its own Event Response Configuration (ERC) file, and one exists for global (system-wide) events. Domain ERC files are responsible for events specific to a system board in that domain. Global ERC files handle events for non-system boards, system boards that are not part of a booted domain, and other non-domain-specific components.

Global ERC files are generated from a template file upon invocation of the **ssp_config**(1M) command. Domain ERC files are generated from a template file by **domain_create**(1M). The template files used by both commands reside in **$SSPVAR/.ssp_private/templates/Ultra-Enterprise-10000**. The global ERC template is named **edd.platform.erc** and the domain ERC template is named **edd.domain.erc**.

Each ERC file contains a series of lines in the following format:

event_type : invoke_action : throttle_timeout : throttle_counter: select_action

where:

event_type A mnemonic (name string) which corresponds to an event type. See **EXAMPLES**, below.

invoke_action A keyword, either **enabled**, which tells the system to invoke the Response Action Script for the event type; or **disabled**, which tells it not to. If this field is blank the system does not invoke the script.

throttle_timeout A time interval, specified in seconds, that indicates how often *throttle_counter select_action*(s) are able to run. (Additional *throttle_counter select_action*(s) are not permitted to run until *throttle_timeout* seconds have expired.)

throttle_counter A number that specifies the number of times *select_action* is permitted to run within each *throttle_timeout* interval. After *throttle_timeout* seconds has expired, *throttle_counter select_action*(s) is again permitted to run.

select_action The Response Action Script to be invoked. This field can contain the script name, with or without a full path, as necessary. The path may contain an environment variable, such as **$SSPOPT, $SSPVAR** or **$SUNW_HOSTNAME**. Optional arguments to the script name can be literals, such as **–v**, or event information: **%e** for event type, **%b** for the board with the event, **%t** for board type, and **%d** for the SNMP trap data. Valid board types (**%t**) are:

0: No type
1: System board
2: Control board
3: Centerplane
4: Centerplane support board

See **EXAMPLES**, below.

Fields are separated by a single colon with or without spaces to its right and left. Lines that begin with a pound sign (**#**) are considered comments and are not parsed.

The *throttle_timeout* and *throttle_counter* fields are used to control how often an action should be run. For example, with a *throttle_timeout* value of 600 seconds and a *throttle_counter* value of 3, an action (*select_action*) can run only three times every 10 minutes (600 seconds). This throttling of actions is helpful to reduce the number of repetitive log messages and dump files. Note that similar actions for different system boards, domains, etc. are throttled independently. For example, a `sys_brd_temp_max` event action would not throttle another `sys_brd_temp_max` event action for a different board. Similarly, an arbstop event action would not throttle another arbstop event action for a different domain.

The ERC file can specify more than one Response Action Script for a given event. To designate a secondary Response Action Script, use a second line with the same event-type mnemonic as that of the first line. Response Action Scripts are invoked sequentially (rather than in parallel) in the order they appear in the Event Response Configuration file. If multiple Response Action Scripts exist for an event, you can supply the name and exit status of the previous Response Action Script to the present Response Action Script through the arguments **%p** and **%s**, respectively.

The **edd.platform.erc** and **edd.domain.erc** files together contain the following information. For more information about event types, see **edd.emc**(4). The information in the ERC files is organized as follows:

System Board Events

Control Board Events

Centerplane Events

Centerplane Support Board Events

IDN Events

CBS/CBE Connection Events

System Configuration Change Events

Host Recovery Events

Other Events

System Board Events

```
sys_brd_temp_norm : enabled : 0 : 1 : TempNormact -b %b -e %e -d %d -t %t
sys_brd_temp_high : enabled : 300 : 1 : TempHighact -b %b -e %e -d %d -t %t
sys_brd_temp_warn : enabled : 300 : 1 : TempWarnact -b %b -e %e -d %d -t %t
sys_brd_temp_max : enabled : 300 : 1 : TempMaxact -b %b -e %e -d %d -t %t
sys_brd_temp_911 : enabled : 60 : 1 : Temp911act -b %b -e %e -d %d -t %t
sys_brd_temp_bad : enabled : 300 : 1 : TempBadact -b %b -e %e -d %d -t %t
sys_brd_volt_norm : enabled : 0 : 1 : VoltageNormalact -b %b -e %e -d %d -t %t
sys_brd_volt_max : enabled : 300 : 1 : Voltageact -b %b -e %e -d %d -t %t
sys_brd_volt_min : enabled : 300 : 1 : Voltageact -b %b -e %e -d %d -t %t
sys_brd_volt_bad : enabled : 300 : 1 : VoltageBadact -b %b -e %e -d %d -t %t
```

System Board Response Actions

`sys_brd_temp_norm`
: Log a message indicating that the board temperature has gone from an over-temperature condition to normal.

`sys_brd_temp_high` or `sys_bd_temp_warn`
: Log a message indicating that the board's temperature is high.

`sys_brd_temp_max`
: Execute the following steps, as necessary, to handle a maximum over-temperature event:

1. If the board is in a domain, shutdown the domain.
2. Power off the board.
3 If the board is in a domain and there are other boards with power in the domain, reboot the domain.

`sys_brd_temp_911`
: Power down the board regardless of whether belongs to a domain.

`sys_brd_temp_bad`
: Log a message indicating that the system was unable to obtain the temperature of the board.

`sys_brd_volt_norm`
: Log a message indicating that the board's voltage reading has returned to a normal condition.

`sys_brd_volt_max`
: Log a message indicating that the board's voltage reading has risen above the maximum threshold.

`sys_brd_volt_min`
: Log a message indicating that the board's voltage reading has dipped below the minimum threshold.

`sys_brd_volt_bad`
: Log a message indicating that system was unable to obtain the board's voltage reading.

Control Board Events

```
cb_temp_norm : enabled : 0 : 1 : TempNormact -b %b -e %e -d %d -t %t
cb_temp_high : enabled : 300 : 1 : TempHighact -b %b -e %e -d %d -t %t
cb_temp_warn : enabled : 300 : 1 : TempWarnact -b %b -e %e -d %d -t %t
cb_temp_max : enabled : 300 : 1 : TempMaxact -b %b -e %e -d %d -t %t
cb_temp_911 : enabled : 60 : 1 : Temp911act -b %b -e %e -d %d -t %t
cb_temp_bad : enabled : 300 : 1 : TempBadact -b %b -e %e -d %d -t %t
cb_volt_norm : enabled : 0 : 1 : VoltageNormalact -b %b -e %e -d %d -t %t
cb_volt_max : enabled : 300 : 1 : Voltageact -b %b -e %e -d %d -t %t
cb_volt_min : enabled : 300 : 1 : Voltageact -b %b -e %e -d %d -t %t
cb_volt_bad : enabled : 300 : 1 : VoltageBadact -b %b -e %e -d %d -t %t
```

Control Board Response Actions

`cb_temp_norm`
: Log a message indicating that the board temperature has gone from an over-temperature condition to normal.

`cb_temp_high or temp_warn`
: Log a message indicating that the board's temperature is high.

`cb_temp_max`
: If the system has fewer than two control boards configured, shutdown all domains in the system and power everything off. If the system has two control boards, power off that the control board that is reading maximum temperature.

`cb_temp_911`
: Shutdown the entire system.

`cb_temp_bad`
: Log a message indicating that system was unable to obtain the temperature of the board.

`cb_volt_norm`
: Log a message indicating that the board's voltage reading has returned to a normal condition.

`cb_volt_max`
: Log a message indicating that the board's voltage reading has risen above the maximum threshold.

`cb_volt_min`
: Log a message indicating that the board's voltage reading has dipped below the minimum threshold.

`cb_volt_bad`
: Log a message indicating that system was unable to obtain the board's voltage reading.

Centerplane Events

```
centerplane_temp_norm : enabled : 0 : 1 : TempNormact -b %b -e %e -d %d -t %t
centerplane_temp_high : enabled : 300 : 1 : TempHighact -b %b -e %e -d %d -t %t
centerplane_temp_warn : enabled : 300 : 1 : TempWarnact -b %b -e %e -d %d -t %t
centerplane_temp_max : enabled : 300 : 1 : TempMaxact -b %b -e %e -d %d -t %t
centerplane_temp_911 : enabled : 60 : 1 : Temp911act -b %b -e %e -d %d -t %t
centerplane_temp_bad : enabled : 300 : 1 : TempBadact -b %b -e %e -d %d -t %t
centerplane_volt_norm : enabled : 0 : 1 : VoltageNormalact -b %b -e %e -d %d -t %t
centerplane_volt_max : enabled : 300 : 1 : Voltageact -b %b -e %e -d %d -t %t
centerplane_volt_min : enabled : 300 : 1 : Voltageact -b %b -e %e -d %d -t %t
centerplane_volt_bad : enabled : 300 : 1 : VoltageBadact -b %b -e %e -d %d -t %t
```

Centerplane Response Actions

`centerplane_temp_norm`
: Log a message indicating that the board temperature has gone from an over-temperature condition to normal.

`centerplane_temp_high` or `centerplane_temp_warn`
: Log a message indicating that the board's temperature is high.

`centerplane_temp_max`
: Shutdown all remaining domains, then power off the system.

`centerplane_temp_911`
: Shutdown down the system.

`centerplane_temp_bad`
: Log a message indicating that system was unable to obtain the temperature of the centerplane.

`centerplane_volt_norm`
: Log a message indicating that the centerplane's voltage reading has returned to a normal condition.

`centerplane_volt_max`
: Log a message indicating that the centerplane's voltage reading has risen above the maximum threshold.

`centerplane_volt_min`
: Log a message indicating that the centerplane's voltage reading has dipped below the minimum threshold.

`centerplane_volt_bad`
: Log a message indicating that system was unable to obtain the centerplane voltage reading.

Centerplane Support Board Events

```
supp_brd_temp_norm : enabled : 0 : 1 : TempNormact -b %b -e %e -d %d -t %t
supp_brd_temp_high : enabled : 300 : 1 : TempHighact -b %b -e %e -d %d -t %t
supp_brd_temp_warn : enabled : 300 : 1 : TempWarnact -b %b -e %e -d %d -t %t
supp_brd_temp_max : enabled : 300 : 1 : TempMaxact -b %b -e %e -d %d -t %t
supp_brd_temp_911 : enabled : 60 : 1 : Temp911act -b %b -e %e -d %d -t %t
supp_brd_temp_bad : enabled : 300 : 1 : TempBadact -b %b -e %e -d %d -t %t
supp_brd_volt_norm : enabled : 0 : 1 : VoltageNormalact -b %b -e %e -d %d -t %t
```

```
supp_brd_volt_max : enabled : 300 : 1 : Voltageact -b %b -e %e -d %d -t %t
supp_brd_volt_min : enabled : 300 : 1 : Voltageact -b %b -e %e -d %d -t %t
supp_brd_volt_bad : enabled : 300 : 1 : VoltageBadact -b %b -e %e -d %d -t %t
```

Centerplane Support Board Response Actions

`supp_brd_temp_norm`
: Log a message indicating that the board temperature has gone from an over-temperature condition to normal.

`supp_brd_temp_high` or `supp_brd_temp_warn`
: Log a message indicating that the board's temperature is high.

`supp_brd_temp_max`
: Shutdown all running domains, then power off the system.

`supp_brd_temp_911`
: Shutdown down the system.

`supp_brd_temp_bad`
: Log a message indicating that system was unable to obtain the temperature of the centerplane support board.

`supp_brd_volt_norm`
: Log a message indicating that the centerplane support board's voltage reading has returned to a normal condition.

`supp_brd_volt_max`
: Log a message indicating that the centerplane support board's voltage reading has risen above the maximum threshold.

`supp_brd_volt_min`
: Log a message indicating that the centerplane support board's voltage reading has dipped below the minimum threshold.

`supp_brd_volt_bad`
: Log a message indicating that system was unable to obtain the centerplane support board's voltage reading.

IDN Events

```
idn_boot : enabled : 0 : 1 : IDNevent -e %e -d %d
idn_halt : enabled : 0 : 1 : IDNevent -e %e -d %d
cluster_arbstop : enabled : 900 : 3 : Arbstopact -d %d
cluster_recordstop : enabled : 900 : 3 : Recordstopact -d %d
```

IDN Event Response Actions

The Inter-Domain Network (IDN) feature is available on Starfire servers running Solaris 2.6 or later.

`idn_boot`
: If at least one other domain in the same IDN has also booted and loaded the IDN software, execute the **domain_link**(1M) command to link the subject domain into the IDN.

`idn_halt`
: Update internal IDN state information to enable the IDN event-handling routines to maintain accurate status of all IDN member domains.

`cluster_arbstop`
: Do an **hpost**(1M) dump of all IDN member domains, then do a complete bringup of all IDN member domains.

`cluster_recordstop`
: Do an **hpost**(1M) dump of all IDN member domains, then attempt to clear the record stop of all IDN member domains.

CBS/CBE Connection Events

```
cbe_connected : enabled : 0 : 1 : actionsysclock
cbe_connected : enabled : 0 : 1 : actioncb
cbe_connected : enabled : 0 : 1 : PowerFailRebootact
```

CBE/CBS Connection Response Actions

Set the system clock (if necessary) and fan speed, then re-establish control board heartbeat. All previously booted domains are checked for operating system and power status to determine if a domain must be rebooted due to a power failure condition. This event represents the condition where the SSP's CBS daemon and the Control Board Executive (CBE) lose connection.

System Configuration Change Event

```
system_config_change : enabled : 0 : 1 : SystemConfChangeact -d %d
```

System Configuration Change Event Response Action

Log a message that a system board, centerplane support board, control board, fan tray, and/or 48-volt power supply has been removed or inserted into the system.

Host Recovery Events

```
arbstop : enabled : 900 : 3 : Arbstopact -d %d
recordstop : enabled : 900 : 3 : Recordstopact -d %d
watchdog : enabled : 900 : 3 : WatchDogRebootact -d %d
environment_shutdown : enabled : 600 : 1 : Environmentact -d %d
reboot : enabled : 300 : 3 : Rebootact -d %d
panic1 : enabled : 900 : 3 : Panicact -t 300 -d %d -e %e
panic2 : enabled : 900 : 3 : Panicact -t 900 -d %d -e %e
panic_reboot : enabled : 900 : 3 : PanicRebootact -d %d
heartbeat_failure : disabled : 900 : 3 : PanicRebootact -d %d
```

Host Recovery Response Actions

`arbstop`
: Do an **hpost**(1M) dump and do a complete bringup on the domain.

`recordstop`
: Do an **hpost**(1M) dump and attempt to clear the record stop.

`watchdog`
: Execute the following steps:
1. Dump **resetinfo** of all processors in the domain.
2. Dump the signature block of all the processors in the domain.
3. Do an **hpost**(1M) dump on the domain.
4. Reboot the domain by doing a complete bringup.

`environmental_shutdown`
: Log a message indicating that the system detected an environmental shutdown on a specific domain.

`reboot`
: Carry out the user requested reboot by doing a quick bringup.

`panic1`
: Sleep for the time specified by the *–t* option of the `Panicact` field, then do a quick bringup if the domain is still in a panic1 state.

`panic2`
: Sleep for the time specified by the *–t* option of the `Panicact` field, then do a quick bringup if the domain is still in a panic2 state.

`panic_reboot`
: Reboot the system by doing a complete bringup.

`heartbeat_failure`
: Reboot the system by doing a complete bringup.

Other Events

```
sys_brd_power_on : enabled : 0 : 1 : PowerOnact -t %t -b %b
sys_brd_power_off : enabled : 0 : 1 : PowerOffact -t %t -b %b
supp_brd_power_on : enabled : 0 : 1 : PowerOnact  -t %t -b %b
supp_brd_power_off : enabled : 0 : 1 : PowerOffact -t %t -b %b
bulk_power_norm : enabled : 0 : 1 : BulkPowerNormact -d %d
bulk_power_fail : disabled : 300 : 1 : BulkPowerFailact -d %d
fan_norm : enabled : 0 : 1 : FanNormact -d %d
fan_fail : enabled : 180 : 1 : FanFailact -d %d
```

Other Event Response Actions

`sys_brd_power_on` and `sys_bd_power_off`
: Log a message indicating that the system board has been powered on or off.

`supp_brd_power_on` and `supp_brd_power_off`
: Log a message indicating that the centerplane support board has been powered on or off.

`bulk_power_norm`
: Log a message that the 48-volt power supply is on. (Note that 48-volt power is shown as bulk power in some system messages.)

`bulk_power_fail`
: Log a message indicating which 48-volt power supply has failed or is off, then determine if the system can continue operating with the current number of valid power supplies. If not, power down the entire system.

`fan_norm`
: Log a message that a fan has gone from a failed state to an on or off state.

`fan_fail`
: Log a message that a fan has gone from an on or off state to a failed state.

EXAMPLES

```
# Event Response Configuration File
#
centerplane_temp_warn : enabled : 300 : 1 : TempWarnact -b %b -e %e -d %d -t %t
centerplane_temp_warn : enabled : 300 : 1 : fans -hi
centerplane_temp_norm : enabled : 0 : 1 : fans -off
```

The first two lines in the above example of a global ERC file tell the system how it is to respond to an over-temperature event on the centerplane. The first line tells the system to pass the specified information – board number of the board experiencing the event (**%b**), event type (**%e**), SNMP trap data (**%d**), and board type (**%t**) – to an action script named TempWarnact, and to then execute TempWarnact. The second line tells it to turn on the fans at their high-speed setting.

The third line above tells the system to turn off the fans when it sees that the temperature of the centerplane is normal.

FILES

$SSPVAR/etc/*platform_name*/**edd.erc**
: The path to an instantiated global ERC file.

$SSPVAR/etc/*platform_name* / *domain_name*/**edd.erc**
: The path to an instantiated ERC file for domain-specific events.

SEE ALSO

edd(1M), **edd.emc**(4)

NAME

fad_files – file access daemon files

DESCRIPTION

Note: Do not edit this file manually. It is automatically maintained by domain-management tools and commands.

fad(1M) consults the **fad_files** file to determine which file access daemon files it should monitor, where those files are located, and what permissions they are assigned. **fad**(1M) also uses the information in **fad_files** to synchronize SSP file reads, writes, and notifications of modification. And other SSP daemons and clients across the net also access this file for file access daemon information.

The **fad_files** file is initially installed in **$SSPVAR/.ssp_private**. It normally contains one or more lines of the following format:

type permission path sub_path filename

where:

type File type, either **domain** or **nondomain**.

permission Access privilege, **r** (read-only), **w** (write only) or **r** (read and write). **fad**(1M) can read or write to a file only if that file has the proper permission.

path Valid Enterprise 10000 path. *path* can be any of the SSP environment variables such as SSPOPT, SSPVAR and SSPETC.

sub_path Absolute path under the Enterprise 10000 path. *sub_path* is normally **etc**, **data** or **.** (dot).

filename The actual **fad**(1M) file name.

Any line that has a pound sign (#) in the first character position is considered a comment line.

EXAMPLE

```
#
#     SSP-Enterprise 10000: fad_files example
#
nondomain   rw  SSPVAR   .ssp_private  domain_config
domain      rw  SSPVAR   etc           eeprom.image
nondomain   r   SSPETC   etc           ssp_startup.main
```

NOTES

This file is shipped with the SSP package and is installed with it.

SEE ALSO

fad(1M)

NAME

postrc – (.postrc) hpost properties file

DESCRIPTION

CAUTION: Many of the **.postrc** directives are appropriate only in an engineering or manufacturing environment. Others may be appropriate for field use, but only by a qualified service provider or properly trained system administrator. **Users without specific training in the use of these features should not use them.**

The **.postrc** file is an ASCII text file that contains the various configurable properties of the **hpost**(1M) program; it controls options in POST.

Some **hpost**(1M) functions can also be controlled from the command line. Such functions are clearly marked in the descriptions that follow. Command line arguments take precedence over commands in the **.postrc** file, which take precedence over built-in defaults. For a terse reminder of the **.postrc** options and syntax, enter **hpost –?postrc**.

Unless –**n** is the first argument on the **hpost**(1M) line, **hpost**(1M) reads the optional file **.postrc**, and executes the directives in that file before it begins operation with the host. If **hpost**(1M) does not find **.postrc** it proceeds without it. **hpost**(1M) first looks for **.postrc** in the current directory (.). If it does not find it there, **hpost**(1M) looks in **$SSPVAR/etc/***platform_name***/$SUNW_HOSTNAME**. If it does not find **.postrc** there, it looks in the user's home directory, **$HOME**. Exception: If the current directory is **$HOME**, the first element of the search path (.) is skipped. This exception allows a domain-specific **.postrc** file to take precedence over the user's default **$HOME/.postrc** when running from the user's home directory, while still allowing use of a local **./.postrc** to take precedence as long as the user is currently in any directory other than the user's home directory.

Some of the commands described below are accepted only when the local file **./.postrc** is used; such commands are clearly marked. These commands are never available when the current directory is **$HOME**, as the **./.postrc** search path is skipped in such cases.

The following rules apply to the **.postrc** file:

- Keywords are case-sensitive.
- Numeric arguments are decimal by default, taken as hex if preceded by **0x** or **x**.
- Keywords and optional arguments are delimited by whitespace.
- Any part of a line that begins with a pound sign (**#**) is considered a comment and ignored.

PROPERTIES

The following list of properties is provided in alphabetical order. Each of these commands can also be classified by the appropriate user level, as designated by the number in parentheses at the end of the title line. The levels are:

Level 1	System Administrator Commands. These commands control functionality either intended for system administrator control or safe because their effect is essentially benign (for example, **logfile**).
Level 2	Commands for use by Sun Microsystems and service providers only.
Level 3	Commands for use by Sun Microsystems POST developers (for debug) and Sun Microsystems engineers only.

allow_lab-only_components (Level 3)

SMI engineering labs sometimes need to operate systems containing hardware that is not acceptable for customer machines. These components are often early prototype parts with known bugs or speed problems, for example. To be sure such parts cannot inadvertently find their way into a production system, POST normally rejects and fails these parts immediately, based on their Component ID (CID) values. To support the engineering lab requirement, the **allow_lab-only_components** command instructs POST not to perform this CID-based rejection, and to accept certain specific values which are recognized but identified as lab-only. This command does not disable all CID checks, it just expands the list of acceptable CIDs.

allow_sys_errctl (Level 3)

When JTAG-configuring the PC, CIC, or MC ASICs, always enable system register write / clear access to error status, error mask, and force error registers. This property is normally enabled only when diagnostic code (such as POST) is run, and not in the final configuration.

arb_flip_ctr_limit *value* (Level 3)

All of the Enterprise 10000 arbitration ASICs reverse the priority order of their clients' requests after some number of grants to improve the fairness of the arbitration algorithm. The number of grants is programmable when the ASIC is configured. A default value is normally used. This command allows Sun Microsystems Engineering to experiment with this value.

asic_ignore *asic_type* (Level 3)

Do not test, configure or dump the named ASIC type. This command allows Sun Microsystems Engineering to exercise POST code in development systems in which certain ASICs do not exist, which would otherwise cause POST to fail immediately. To get a list of the available *asic_type* keywords, use **hpost -?postrc**.

assume_ldarb_rev_2 (Level 3)

Do not check the CID of LDARB ASICs to understand how to set their configuration; just assume they are not rev 1. This command supports a special mode of POST (see **scantool_simulate**) used to provide the JTAG configuration as input to Verilog ASIC simulation, in which no ASIC input is possible.

blacklist_file {*file* | **none**} (Level 2)

Command Line Equivalent: **–X**

Use the specified **blacklist**(4) file rather than the default, **$SSPVAR/etc/***platform_name***/blacklist**. *file* can be the path to the **blacklist**(4) file or **none**. If it is **none**, POST reads no blacklist file.

Be careful when using **blacklist_file** on a production system; other SSP software does not know when POST is using a nonstandard **blacklist**(4) file.

board_red_any_fail (Level 2)

If any failure is detected on a board, immediately fail and effectively redlist all the resources on the board. This line is intended to prevent further configuration of the board, so that a postmortem analysis of the stop can be performed. See also **system_red_any_fail** and **redlist**(4).

bus_shuffle_mode {**0 | 1 | 2**} (Level 3)

A facility in the PC ASIC allows the assignment of addresses to GABs to be varied from the default, which has it based on PA[8:6], with the exact assignment depending on how many and which GABs are configured. This default is shuffle mode 0. Shuffle mode 1 uses PA[8:6] XOR PA[18:16], shuffle mode 2 uses PA[8:6] XOR PA[20:18]. This command allows Sun Microsystems Engineering to experiment with the effect of this shuffle capability.

cplane_initial_config (Level 3)

Command Line Equivalent: **–C**

Start POST in the initial (rather than a subsequent) domain. Several independent domains may be running on an Enterprise 10000 system, but unless they are in **isolated_sysboard** mode, they all share the centerplane ASICs, which can be in only one functional bus configuration. By default, POST assumes it is not running in the first domain being brought up. So it probes the centerplane to determine the bus configuration, and constrains itself to use that configuration. Also, it does not do the functional configuration of the centerplane ASICs.

The usual way to start POST running in the initial domain is via the **–C** command line option. When POST is running in the initial domain, it must establish the bus configuration **and** do the initial centerplane ASIC configuration.

This command (and the **–C** command line option) starts POST in the initial domain, where it must establish the bus configuration **and** do the initial centerplane ASIC configuration.

cplane_initial_config is equivalent to the **–C** command line option for the convenience of Sun Microsystems Engineering, Manufacturing and Customer Service technicians who must repeatedly run POST on uninitialized systems.

Do **not** use this command on production systems that run, or may someday run, multiple domains, as it may cause POST running in the domain brought up second to crash anything running in the domain brought up first.

debug_mask *mask* (Level 2 / 3)

Use *mask* as the debug mask. This is a bitmask that controls experimental code or obscure behavior modes of POST. It is mostly for POST developer use. The *mask* is ORed into the internal mask, allowing one line / bit, so bits can be added / removed by the commenting out of a list in the **.postrc** file. Because these bits change over time users should execute **hpost –?postrc** for a list of current bits.

default_ecache_mbytes {**H | 1 | 2 | 4 | 8 | 16 | 32**} (Level 2)
Assume the specified ecache size for all processors if the ecache probe, done as part of the initial processor test phase, is skipped. If the probe is skipped and this value is incorrect, the configured system will not maintain correct cache coherency. The "default default" is 0.5MB – "H".

default_memgrp_mbytes {**64 | 256 | 1024**} (Level 2)
Use the specified dimm group size as the default. The default size is used for FOM calculations prior to the dimm probe phase, and also as the assumed value of all good dimm groups if the dimm probe phase is skipped by the **skip_phase** command. The "default default" is 64MB.

display_fom_calc {**off | on**} (Level 1)
If **on**, always make POST print the calculated Figures Of Merit (FOM) for all 45 bus configurations when it is making an FOM calculation to select a configuration. If **off**, never do so. If this command is not present, POST does the printing if the verbose level is above a certain value.

dom_transgress_err_enbl (Level 2)
Enable reporting of Domain Transgression Errors by the GDARB ASIC when configuring the centerplane, which is done when **hpost**(1M) is called with the –**C** option. (See **cplane_initial_config**). A transgression error occurs when a system board's data arbiter requests to send data to a board in a different domain cluster as defined by the GDARB's SMD mask configuration register.

By default, reporting of these errors is disabled to minimize the probability of GDARB asserting a global arbstop when a board fails and causes both a parity error and a transgression error. GDARB does inhibit transgressing requests; it just doesn't report them as errors. This **dom_transgress_err_enbl** command enables this error reporting, for cases where the additional fault detection is preferred over the additional domain fault isolation.

domain *boardmask* (Level 3)
Declare a set of system boards to be a domain. Domains are disjoint partitions of the system boards in an Enterprise 10000 system that are isolated from each other, except for shared memory (see **command shared_mem**).

This **.postrc** command is not how domains are managed in production systems; it is a static instruction to POST to configure the domain registers for Sun Microsystems Engineering debug of the domain hardware. This **domain** command is effective only when used in a **.postrc** file in the current directory. If the **.postrc** file used is not in the local directory, POST ignores this command and generates a warning message.

download_bbsram_verify {**off | on**} (Level 2)
If **on**, always verify that each JTAG POST code download to a processor BBSRAM actually occurred. If **off**, never do so. If it is not specified, verification is done only if the level is above a certain value. To do the verification, **download_bbsram_verify** uploads code from BBSRAM and compares it to the code file data. Unlike download, which is done with a JTAG broadcast, the

upload and compare must be done serially for each processor.

download_path *dirpath* (Level 2)
Use the download files for POST that are in the location *dirpath*. The default is **$SSPOPT/release/Ultra-Enterprise-10000/*/*/*/hostobjs**, where **/*/*/*/** indicates a release-level dependency.

dtag_test_all_procs (Level 2)
Normally, POST assigns each processor to test its own DTAG RAM for each CIC on the board. This process ensures that only resources that can be used are tested, as DTAG RAM for missing or failed processors is not usable for anything else. However, in some environments, such as manufacturing test, Sun Microsystems prefers that all the DTAG RAM be tested, in preparation for future installation of additional processors. This command causes each processor, during the DTAG RAM test phase, to test **all** the RAM on each CIC, not just its own. For the same reasons, this command also causes testing of all physically present DTAG SRAM locations, not just those that will be used to support the size of ecache on the currently installed processor modules.

dump_no_sneeprom (Level 3)
When creating a dump file, either with the –**D** command line option or automatically when certain errors are encountered while POST is running, do not include the contents of the serial number eeproms in the dump. This command can significantly speed up the dump process in some environments. It may be removed in the future.

dump_on_exit [*boardmask* [*path*]] (Level 2)
Make POST create a dump file of the system state just before it exits. The *boardmask* controls what boards are dumped, where bits [15:0] control system boards and [17:16] control half-centerplanes. The board is dumped if the corresponding bit is 1. The default is x3FFFF (dump all). This would normally be specified as a hex value. *path* can be used to override the default path for the dumpfile. A *path* can be specified only if a *boardmask* is specified. Both options are similar to those of the –**D** command line option.

dump_pathname *dirpath* (Level 2)
Create state dump files in the specified default directory rather than the program default, **$SSPVAR/adm/$SUNW_HOSTNAME**. If a path has been specified by either the command line option (–**D**) or the **dump_on_exit** command, that path overrides the one specified by **dump_pathname**.

fom_weight_abus *float-in_range* (Level 1)
Use the specified weight of the number of address buses in the Figure of Merit computation. The float-in range is 1.0 to 5.0; the default is 1.0.

fom_weight_dbus *float-in_range* (Level 1)
Use the specified weight of the number of data buses (72-bit datapaths) in the Figure of Merit computation. The float-in range is 1.0 to 5.0, inclusive; the default is 1.0.

fom_weight_mem *float-in_range* (Level 1)
Use the specified weight of memory in the Figure of Merit computation. The float-in range is 1.0 to 5.0, inclusive; the default is 1.0.

fom_weight_proc *float-in_range* (Level 1)
Use the specified weight of processors in the Figure of Merit computation. The float-in range is 1.0 to 5.0, inclusive; the default is 1.0.

fom_weight_scard *float-in_range* (Level 1)
Use the specified weight of SBus cards in the Figure of Merit computation. The float-in range is 1.0 to 5.0, inclusive; the default is 1.0.

force_enbl_cp_driver (Level 3)
Enterprise 10000 system ASICs that connect to the centerplane have both a loopback control and a separately controlled enable for the centerplane drivers. Normally, POST configures these ASICs so that the drivers are disabled when the chip is in loopback, and enabled when it is not. For probing the board signals, it is sometimes useful to drive the centerplane connector with the address and data buses even when in loopback. This command causes all ASICs to have their centerplane drivers enabled, even when configured in board loopback. See also **pc_loopback_lbus_enbl**.

gamux_seq_err_enbl (Level 2)
Early GAARB revisions, which should never be shipped in production machines, have a design bug that causes sequence errors that will be detected by the GAMUX ASICs when an arbstop occurs for some other reason in one domain. Since these sequence errors are considered global errors, this bug results in an arbstop in all other domains, as well.

By default, detection of these sequence errors is disabled when the GAARBs are rev 1 or 2, and enabled otherwise. The **gamux_seq_err_enbl** command allows explicit control, disabling detection of these errors when set to **0**, enabling it when set to **1** – regardless of GAARB revision.

interactive_use_postrc_skips (Level 2)
When running in interactive mode (–**i** command line option), the default behavior is to ignore any **skip_phase** or **skip_test** commands in the **.postrc** file, giving the user the option of running or skipping each interactively. In some cases that may not be the desired behavior; the user may not want to have to interactively skip dozens of phases and tests to get to the one of interest. This command causes POST to use any **skip_phase** or **skip_test** commands, and not query the user about whether to run them. See also the **no_skip_phase_covers_npb** command; the implied skips described there are considered to be in the **.postrc** file with respect to this command.

interconnect_MHz *float* (Level 2)
Use the specified system interconnect frequency, expressed in MegaHertz. This value is used for calculating system configuration parameters (most particularly, memory timing) in teststand environments lacking a real control board that can measure and report this frequency through the SNMP server.

In systems that do have a control board, where use of the SNMP frequency measurement is used, specifying this value causes POST to require that the measured value be within 0.5 percent of this value. If it is not specified, the measured value must be within -3.0 and +0.5 percent of the target interconnect frequency read from the SNMP server.

See also **proc_freq_ratio**.

iopc_rev_5_ok (Level 2)
The rev-5 PC asic has a bug that occurs when the asic is used in the I/O position, PC number 2 on any system board. The effects of the bug may be acceptable for certain applications, with or without certain software workarounds installed. POST normally fails these asics, but accepts them if their presence is explicitly acknowledged with this command.

jtag_lock_trace (Level 2)
Make POST print verbose debugging messages about its actions with respect to the locking and unlocking of JTAG boards and rings.

isolated_sysboard (Level 2)
Make POST ignore the fact that the centerplane ASICs appear missing or broken, which would normally cause it to fail immediately; and prevent POST from taking the board out of loopback. This function is used in development environments that do not have a centerplane.

level *level* (Level 1)
Command Line Equivalent: **–l**
Use the specified level to determine the amount of testing to be done and the thoroughness with which it is done. Higher levels result in more testing. Valid levels are in the range 7 to 127, inclusive. **hpost –?level** prints a brief summary.

This command is suitable for end user control, but do not set the level lower than normal. Doing so subverts the intent to boot only systems with a low likelihood of undetected hardware problems. Setting it higher than the default (see the **–l** option of **hpost**(1M)) may be appropriate for customers willing to trade longer boot times for more thorough diagnostic coverage.

logfile [*path*] (Level 1)
Command Line Equivalent: **–g**
Create a screen logfile. The default filename is **post***mmdd.hhmm***.log** and the default directory is **$SSPVAR/adm/$SUNW_HOSTNAME/post**. If the default path is used and the default directory does not exist, but **$SSPVAR/adm/$SUNW_HOSTNAME** does, POST creates the **post** subdirectory and places the log there.

If *path* is specified and it can be opened for append, POST creates the log there. Otherwise, if the default filename can be opened in *path* used as a directory, POST creates the log there. If a log is requested and none of the above is specified or successful, POST attempts to create the logfile in the current directory with the default name.

The logfile is always opened for append, and is timestamped both when opened and closed, so that a calling script or program can specify a single file to receive logs from a series of POST runs, and to prevent loss of information in the unlikely event of a name conflict.

This command is rated Level 1 because it is fairly benign, other than using up SSP disk space. However, note that in a production environment, POST output is usually diverted (by the **bringup**(1M) script using –**s**) to **syslog**, which may be a more appropriate way to save this information in production environments.

logger_level {**default | emerg | alert | crit | err | warn | notice | info | debug**} (Level 2)

POST captures all liblogger messages produced by the libraries it uses, so they can be part of its own displays and logs. Normally, it also overrides the default logger level to WARNING, so that the contents of its displays are not subject to external control. This command causes POST to either not override the current default logger level it inherits from the environment, allowing it to be controlled by the normal logger mechanisms, or to explicitly set the logger level to any desired value.

If the **.postrc** file used is not in the local directory, POST ignores this command and generates a warning message.

logical_memboard_swap *slot1 slot2* (Level 3)

Exchange the *logical memory board numbers* of the two named physical system boards. Physical memory addresses are assigned by POST starting at 8GB ∗ *logical board number*. By default, the assignment of logical memory board to physical system board is 1-1. Swaps may occur over time, for example as the result of Dynamic Reconfiguration board detach operations, or the sort done by POST as explained in the description of the command **no_memboard_sort**. In a production system SSP, this physical-to-logical mapping is kept in the SNMP MIB and read by POST each time it runs. This command allows Sun Microsystems Engineering to use non-default mappings in lab environments where SNMP is not running.

Any number of these commands may be present; the swaps occur in the order requested. The only constraints are that the two arguments are different valid system board numbers.

This command is effective only when used in a **.postrc** file in the current directory. If the **.postrc** file used is not in the local directory, this command is ignored and a warning message is generated.

Any swapping caused by this command is overwritten by the map information obtained from SNMP, and effectively ignored, unless that access is inhibited with the command **no_snmp_memmap** or **no_snmp_access**. Such swaps cannot be done in combination with the map obtained from SNMP; the two are mutually exclusive.

mc_gabfifo_hold_depth *6-bit value* (Level 3)
Override normal MC ASIC configuration of this field, which is the constants register bits [17:12].

mc_prescale_disbl (Level 2)
Override normal MC ASIC configuration which causes all timeouts to be prescaled by 2**10 system clocks. This field is the Timeout Select register bit 23.

mc_store_match_to *5-bit_value* (Level 2)
Override normal MC ASIC configuration of this field. This field is the Timeout Select register bits [9:5].

mc_timing_reg *reg value* (Level 3)
Override normal MC ASIC configuration of the specified memory timing control register [3:0], where register 0 is refresh timing, register 1 and 2 are memory timing 1 and 2, and register 3 is DIMM wire timing.

mc_xmit_req_to *5-bit_value* (Level 2)
Override normal MC ASIC configuration of this field, which is the Timeout Select register bits [4:0].

mc_xtra_data_to *5-bit_value* (Level 2)
Override normal MC ASIC configuration of this field, which is the Timeout Select register bits [14:10].

mem_board_interleave_ok (Level 1)
Permit POST to configure memory so that two boards with identical amounts of memory are interleaved in 256-byte blocks through a common address range, which is twice what either board would otherwise contain.

Interleaving can improve performance by distributing memory accesses of large blocks of contiguous data between two memory controllers. However, interleaving two boards can have the effect of restricting dynamic reconfiguration (DR); in particular, it may prevent an interleaved board from being DR detached.

The default POST behavior is to not interleave boards. This command enables interleaving of any two boards with the same amount of memory.

memboard_swap_threshold_mbytes *mbytes* (Level 2)
This command defines the minimum amount of memory that must be on the board with the lowest logical board number, to avoid swapping its physical addresses with those of the board in the domain with the most memory. See the command **no_memboard_sort** for an explanation of the sort and how it can be disabled.

The objective of this swap is to have enough memory in the first contiguous Physical Address chunk in the domain for the DR "caged kernel" to fit within this single board, increasing the likelihood that any board can be DR detached. However, if the first board has a required minimum amount of memory, the swap is not required even if another board has yet more memory, and Sun Engineering desires to avoid this unnecessary swap.

The default and minimum value of this no-swap threshold is 512MB. This

command is provided to allow field increases of this threshold if the caged kernel characteristics change. The maximum is 4096 (4GB), which is all the memory a board can contain with the largest supported memory DIMMs.

memchunk_page_truncate *#_of_8KB_pages_per_chunk* (Level 3)

Limit the amount of memory declared in the memory chunk list passed to OBP and cleared in the final configuration phase. If the number of pages in a contiguous chunk (usually the physical memory of one board or a pair of interleaved boards) is greater than this value, truncate it to the declared size. This command does not affect the configuration of the memory controller, only how much memory is declared to OBP and cleared. It is intended for controlled debug environments that can deal with a portion of configured physical memory not initialized, to save time, and is most useful in an emulation environment.

memtest_limit [*base*] *limit* (Level 2)

Limit the amount of memory tested in each dimm group by the POST memory tests. If only one argument is supplied, it is used as the limit and the base is left at 0.

Both values represent byte offsets within each group, as if it were configured at PA 0 without any interleaving. In general, this does not correspond to the physical addresses a given group is assigned in a fully configured machine, or those used during the memory tests themselves. In particular, memory testing is always done as if the banks were 4-way interleaved, even if not all 4 banks are present. Each processor tests a range of addresses 4 times the number of bytes it is testing, but it tests only 64 bytes out of each 256 within that range, all of which are in the same dimm group.

These limits are applied to every group tested. Since memory is tested only in 64-byte blocks, both arguments must be mod-64. The maximum for both is 0x40000000, or 1GB, which is the full size of a group of the largest dimms supported. The limit must be greater than the base. If the base is greater than a physical group's size, that group is (quietly) not tested.

new_cid_rev *component_id* (Level 2)

During its JTAG integrity test phase POST reads the component IDs of all the ASICs, as well as other chips that have a CID, and compares them against tables of acceptable CIDs for these chips. In general, discrepancies result in POST failing the chip. However, if a CID matches a value in the table except for the revision (the most significant hex digit of an 8-digit CID), and the actual revision is higher than the revision in the table, the chip is accepted and POST issues a warning about the *up-level* chip.

This process adds resilience to hardware upgrades without forcing distribution of a POST patch. It also serves notice that POST may not be aware of all features and requirements of all present hardware, so that if any hardware problems occur, the chip-version issue will be immediately obvious.

While the up-level message is only a warning, it may make some users uncomfortable. The purpose of this **new_cid_rev** command is to effectively add an

entry to the CID table to suppress the up-level warning. The **.postrc** file may contain multiple instances of this command. POST consults the list of values they create only when it is about to issue an up-level warning message. Thus, you cannot use **new_cid_rev** commands to declare new base CIDs, or to declare that a down-level CID revision is acceptable. Since the base part of the entered CID must match that of at least one of the base CIDs in the compiled table for some chip for this command to have any effect, it is not necessary to specify in this **new_cid_rev** command which chip type is having its table extended.

Service providers should use this facility to suppress the up-level warnings only when notified by Sun Microsystems Engineering that a specific up-level chip is really equivalent to the older revision compiled into POST.

Normally, *component_id* would be entered as an 8-digit (32-bit) hex value. It is checked only to be sure that it satisfies the requirement of any valid CID; that is, to ensure that its least significant digit is 1.

no_asic_config_check (Level 3)

Skip the normal readback and compare of ASIC CSRs written by JTAG when configuring the system. This command is useful in certain development environments during system simulation.

no_bbc_error_check (Level 2)

POST usually checks the BootBus error status register in the PC, at the end of major phases, for BootBus parity or access errors. Errors usually result in POST failing the PC. This command suppresses all checks of this register.

no_bbsram_clear (Level 2)

POST normally writes all of BBSRAM to prevent parity errors if some code accesses an uninitialized address. It does so on the first code download of a file declared to need a signature block, to avoid interfering with the special BootBus SRAM test file. To save download time, POST fills only those segments of BBSRAM that not loaded with code or data, and it does so only once.

This command suppresses this fill, so that areas of BBSRAM not explicitly loaded by the code are left untouched.

no_check_chain_length (Level 2)

During its JTAG integrity test phase, POST checks that all chips with component IDs have one that is the standard 32-bits long. This simple but useful test ensures that the chip's JTAG scan connection is reliable. This command makes POST skip the chain length test, but still do the probing operations of the JTAG integrity test phase.

no_dumpfile_on_stop (Level 2)

Make POST suppress its normal action of creating a dumpfile of "interesting" system state if and when it detects an arbstop or recordstop during testing.

If this command is not present, POST dumps the system state, edits it to skip boards that have no interesting state, then analyzes the dump. If POST finds no boards with interesting state, it does not create a dumpfile and just moves on. (For example, data ecc error recordstops that originate in memory may cause

POST to simply move on without creating a dumpfile.) See also **dump_pathname**.

If **no_dumpfile_on_stop** is not in **.postrc** but **no_stop_analyze** is, POST simply creates the dumpfile then moves on, neither editing nor analyzing it.

no_final_flush_reset (Level 2)

The normal action of POST is to pause and flush all non-redlisted processors' master queues in the PC ASIC, then reset each processor at the end of its final configuration. This leaves a state that facilitates download of OBP code to BBSRAM without any problems executing stale fetches from the POST spin-wait loop. If **no_final_flush_reset** is invoked, the processors are instead left spin-waiting in the POST final configuration code in BBSRAM, which may be useful for postmortems of final configuration processor state.

no_jtag_locks (Level 3)

SSP software that uses the JTAG communication facilities provided by the Enterprise 10000 control board is normally expected to *lock* the JTAG ring or rings with which it is currently communicating to prevent multiple SSP applications (e.g., POST and **obp_helper**(1M)) from interfering with each other. The locking is done through semaphore-based services provided by the SSP libraries.

This command causes POST to skip use of these locking services, which is useful in certain engineering development environments.

no_lockfile (Level 3)

Permit multiple POST processes to run simultaneously on the same Enterprise 10000 host domain. POST normally creates the lockfile **$SSPVAR/adm/$SUNW_HOSTNAME/hpost.lock** to prevent multiple instances of POST, since mutual interference is likely to occur, causing both POST processes to fail.

In a normal SSP environment, POST is usually run in daemon mode. It is not always obvious that this is happening and, in a development environment, it often interferes with user-initiated POST processes.

However, occasionally allowing two POST processes to run is considered less of a problem than POST refusing to run due to a stale lockfile. Therefore, if creation of the lockfile fails, POST attempts to validate that it was created by another active POST process. If this attempt fails, POST enters **–f** mode (described below) and proceeds to run, deleting any lockfile when it exits. This process provides automatic recovery from most cases in which a lockfile is left from an abnormally terminated POST process.

The effect of this **no_lockfile** command is to completely disable the lockfile function, both file creation and removal.

Note that this is significantly different from the **–f** command line argument. With **–f**, POST still tries to create a lockfile, but continues even if that fails; and it attempts to delete the lockfile at exit, even if the create failed. The purpose of **–f** is recovery from a stale lockfile that incorrectly appears to belong to an active POST process.

no_mc_hardreset (Level 2)

The Memory Controller (MC) ASIC has some functions, most notably memory refresh, that are not affected by normal system reset. These functions are cleared only by a *hard reset*, which can be caused via the JTAG TAP reset pin, which occurs during a power-on reset, or by a special hard reset control bit in a JTAG config register. It is possible, particularly when LBIST is run, to leave the MC in a state that requires a hard reset to make it usable.

POST normally asserts a hard reset to all MC ASICs during the initial reset phase. This command causes the hard reset to be skipped, which might be useful in some test environments. This hard reset is also skipped if POST is run in the special –**Z** reconfiguration mode.

no_memboard_sort (Level 2)

The default action of POST when running in a domain of two or more boards is to do a partial sort of the boards based on the amount of memory on each board. The object of this sort is to configure the domain such that the lowest physical memory addresses in the domain are on a board with the largest amount of memory, unless the amount on the lowest board is above a threshold amount. This optimizes the capabilities of some features of Dynamic Reconfiguration, as explained in the description of the command **memboard_swap_threshold_mbytes**.

Doing this sort may require a single physical-to-logical mapping swap of the lowest logical board in the domain with another board in the domain that has more memory. See the command **logical_memboard_swap** for more information about this mapping. Since the physical-to-logical mapping is maintained by the SNMP agent on the SSP, if a swap is required, POST must do a write operation to SNMP to inform it of the swap.

This command, **no_memboard_sort**, suppresses both the sorting and the SNMP write access. This suppression is implied if **no_snmp_memmap** or **no_snmp_access** is invoked.

no_memory_ok (Level 2)

Since the operating system cannot boot without memory, the amount of memory is one component of the configuration Figure of Merit (FOM) used to evaluate configurations. No memory results in a 0 FOM. For board bringup, however, Sun Microsystems sometimes wants POST to continue testing even with no memory installed. This command tells POST to pretend there's one *unit* (currently 64 MByte) of memory when calculating the FOM if there is actually none.

Other features in the processor modules and I/O require valid memory to be tested. Therefore, do not run board tests without memory as standard practice, as these other elements would not be tested. However, this capability is useful upon occasion.

no_non_proc_boards (Level 1)

Do not allow system boards without a processor on board. By default, Non-Processor Boards (NPBs) are permitted, so that their onboard memory and I/O is not lost because of processor failures. In general, such configurations are fully functional.

In certain situations such NPBs may be undesirable. This command causes POST to remove such boards from the configuration, even though they may have usable I/O and/or memory.

no_obp_handoff (Level 2)

During the final configuration phase do not create the post2obp structures in BootBus SRAM of the boot processor. This command causes an otherwise successful run of POST to exit with POST_EXIT_NOCONFIG instead of a boot processor number (0 to 63, inclusive).

no_poll_board_arbstop (Level 3)

Suppress POST's normal polling for arbstop and recordstop. POST does the polling by reading the Local Address Arbiter (LAARB) error register via JTAG. When this command is used, stops are detected in other ways, most often by timeouts. The processors, instead of the true cause of the error, are failed, and recovery capability is limited.

no_poll_timeouts (Level 3)

Disable all timeouts on the host processor RPC services. This is useful when running with breakpoints installed in the resident code of a host processor during Sun Microsystems Engineering's debug of POST. Compare to **poll_timeout_mult**.

no_procs_ok (Level 2)

Since the operating system cannot boot without processors, the number of processors is one component of the configuration Figure of Merit (FOM) POST uses to evaluate configurations. No processors results in a 0 FOM. For board bringup, however, Sun Microsystems Engineering sometimes wants POST to continue running even with no processors installed. This command tells POST to pretend that there is one processor in calculating the FOM if, actually, there are none. When no processors are present, POST can do only a very limited number of functions. But it can do a JTAG ASIC configuration, which is useful in certain bringup environments.

no_scards_ok (Level 2)

Since the operating system cannot boot without I/O, the number of SBus cards (or, more generally, the number of populated I/O slots) is one component of the configuration Figure of Merit (FOM) used to evaluate configurations. No SBus cards results in a 0 FOM. For board bringup, however, Sun Microsystems Engineering sometimes wants POST to continue testing even with no SBus cards installed. This flag tells POST to pretend that one SBUS card is present when it is calculating the FOM if, actually, there are none.

no_skip_phase_covers_npb (Level 2)

Where possible, POST tests the system resources in board loopback, using other onboard resources. This technique improves POST's ability to isolate failures and generally allows faster test execution, because more tests can run simultaneously on different system boards.

If a system board isn't fully populated or has experienced onboard failures, POST must use offboard resources. For example, a board with no good processors (a Non-Processor Board, or NPB) must use offboard processors to test its memory and I/O. A board with no good memory (A Non-Memory Board, or NMB) must use offboard memory to run tests of processor and I/O versus memory. A board with a single processor must run crosscall interrupt tests against processors on other boards.

When a base test phase, such as onboard memory tests, has been skipped due to use of the **skip_phase** command, most technicians would assume that all the memory tests would be skipped. That assumption would lead to confusion if the NPB memory tests then take over and test the on-board memory later in POST. Therefore, the default behavior of POST is that when such a base phase has been skipped, all phases that test the same resources offboard are also skipped automatically.

This is usually the desired behavior but, in some cases, it might not be. Forcing all tests to run out of loopback, while slower, can be an excellent stress test of the centerplane interconnect. Also, the technician or engineer may want to skip the base test, but run the NPB test while troubleshooting a particular problem. To support this, the **no_skip_phase_covers_npb** command suppresses the default implied skip of offboard phases when the base phase is skipped.

Note that while the command uses the acronym NPB, it actually affects all such offboard test phases, such as those for NMBs and xcall tests for single-processor boards.

Note also that the implied skip mechanism described operates only at the phase level. If all tests in a base phase have been skipped because of the **skip_test** command, the phase is not considered skipped.

no_snmp_access (Level 2)

Prevent POST from accessing the SSP's SNMP agent to obtain various information normally required for it to run. This command is provided to support various Sun Microsystems lab environments where the SNMP agent may not be running. It is effective only when used in a **.postrc** file in the current directory. The next component in the search path for **.postrc** requires the *platform_name*, which is obtained from the SNMP agent. If the **.postrc** file used is not in the local directory, POST ignores this command and generates a warning message.

Many of the items of information normally obtained from SNMP can be provided through the **.postrc** file, or through program defaults. See the following

commands: **interconnect_MHz**, **proc_freq_ratio**, **domain**, **logical_memboard_swap**, **platform_name**, and **no_snmp_freq_read**.

no_snmp_invalidate_bootproc (Level 2)
By default, when a successful invocation of POST returns a boot processor number in the range [0,63], POST invalidates the SNMP value of the bootproc for the domain in which it is running before it starts interaction with the host. (Exception: POST does not invalidate the SNMP value when run in the special –**H** Dynamic Reconfiguration mode.) By invalidating this value, POST causes other SSP monitoring software, particularly the Event Detector Daemon (EDD), to stop checking for arbstops and other events in this domain, possibly interfering with this **hpost**(1M) run.

The **no_snmp_invalidate_bootproc** command suppresses this invalidation. **no_snmp_invalidate_bootproc** is implied by **no_snmp_access**.

Note that POST never sets this SNMP element to a valid value, it only invalidates it. Setting it to a valid bootproc value is normally done by **bringup**(1M). Therefore, when POST is run other than by **bringup**(1M), the SNMP bootproc MIB element is invalidated and left that way.

no_snmp_memmap (Level 2)
Prevent POST from accessing the SSP's SNMP agent to obtain or modify the physical-to-logical memory board mapping. This prevention is implied if **no_snmp_access** is invoked. See the command **logical_memboard_swap** for more information about this mapping.

This command is effective only when used in a **.postrc** file in the current directory. If the **.postrc** file used is not in the local directory, this command is ignored and a warning message is generated.

no_snmp_freq_read (Level 2)
Prevent POST from accessing the SSP's SNMP agent to obtain the platform interconnect and processor frequency values. This prevention is implied if **no_snmp_access** is invoked. When either **no_snmp_freq_read** or **no_snmp_access** is used, POST takes these values from the **.postrc** commands **interconnect_MHz** and **proc_freq_ratio**, or from built-in default values.

no_stop_analyze (Level 2)
POST's normal action when detecting an arbstop or recordstop condition while running most tests is to read and analyze the status in the various ASICs, and to fail components when the fault can be determined. This command suppresses this function.

pc_color_freq *code* (Level 3)
The PC ASIC has a two-bit configuration parameter that controls the frequency with which PC changes the value of arbitration color, a property of the arbitration fairness algorithm. A default value is normally used. This command allows Sun Microsystems Engineering to experiment with this parameter. This field is bits [19:18] of the configuration register.

pc_errmask {**0 | 1 | io0 | io1**} *bit_number* (Level 3)
Allow individual bits in the error mask registers of PC ASICs to be forced to 0 during configuration, disabling the specified error from causing an arbstop. Multiple instances and versions of this command may be present; the bits to be masked are accumulated. If a number (0 or 1) follows this command, it affects only processor PCs; if the io form (io0 or io1) is used, it affects only I / O PCs. The 0 or 1 indicates which error mask register is specified. *bit_number* is an integer in the range 0-31, inclusive.

pc_grant_to *6-bit pse/value* (Level 2)
Override normal PC ASIC configuration of this field, which is the Timeout / Hold config register bits [10:6].

pc_grant_to_io *6-bit_pse/value* (Level 2)
Override normal PC ASIC configuration of this field, which is the Timeout / Hold config register bits [10:6] for I / O PCs. This command overrides any **pc_grant_to** command.

pc_loopback_always (Level 3)
Force PC loopback mode every time any PC is configured.

pc_loopback_lbus_enbl (Level 3)
During the initial processor module tests, POST uses a facility of the PC ASIC called PC loopback mode. In this mode the PC does not send transactions through the CICs, but instead loops them back internally. This process allows better fault isolation, as you know that any failures in these tests are not related to the CICs or the PC-CIC signals.

As with board loopback, the PC has separate controls for PC loopback and driver enables for the local bus to the CIC. POST normally configures this so the drivers are disabled in PC loopback mode. This command causes these drivers to be enabled in PC loopback mode, so that they may be probed during debug. Compare this command to **force_enbl_cp_driver**.

pc_master_pio_req_limit *2-bit_code* (Level 3)
Override normal PC ASIC configuration of this field for processor PCs. (It is always 0 for I / O PCs.) This is the Timeout / Hold config register bits [24:23].

pc_master_read_to *6-bit_pse/value* (Level 2)
Override normal PC ASIC configuration of this field, which is the Timeout / Hold config register bits [5:0]. Bit [5], the most significant bit of this field, is the prescale enable for all of the master read, interrupt, and slave response timeouts in the PC.

pc_master_read_to_io *6-bit_pse/value* (Level 2)
Override normal PC ASIC configuration of this field for I / O PCs. This field is the Timeout / Hold config register bits [5:0]. **pc_master_read_to_io** overrides any **pc_master_read_to** command.

pc_pc_hold *5-bit_value* (Level 3)
Override normal PC ASIC configuration of this field. This field is the Timeout / Hold config register bits [22:18].

pc_slave_response_to *5-bit value* (Level 2)
Override normal PC ASIC configuration of this field, using the specified 25-bit value for the Timeout / Hold config register bits [31:27]. Note that this timer uses the prescale enable for the master read timeout; see **pc_master_read_to**.

pc_slave_response_to_io *5-bit value* (Level 2)
Override normal PC ASIC configuration of this field for I / O PCs, using the specified 5-bit value for the Timeout / Hold config register bits [31:27]. This command overrides any **pc_slave_response_to** command.

pc_slave_wr_to *6-bit_pse/value* (Level 2)
Override normal PC ASIC configuration of this field. This field is the Timeout / Hold config register bits [16:12].

pc_slave_wr_to_io *6-bit_pse/value* (Level 2)
Override normal PC ASIC configuration of this field for I / O PCs. This field is the Timeout / Hold config register bits [16:12]. **pc_slave_wr_to_io** overrides any **pc_slave_wr_to** command.

pc2_iopc_cic_perr_disable (Level 2)
Cause POST to disable I / O PC ASICS from detecting parity errors on the CIC control buses. This is required for operation if any boards in the system have the "magic wire" ECO. This is generally true for all boards that have a PC below Rev 4 in the I / O position (PC 2 on that board), since such PCs do not work reliably without this ECO.

phase_time_report (Level 2)
Report the elapsed time for each POST phase.

platform_name *string* (Level 2)
See command **no_snmp_access**. If that command is invoked, this command can provide the platform name, normally obtained from the SNMP agent. If **no_snmp_access** is not invoked, POST ignores this command. If **no_snmp_access** is invoked and this command is not present, POST uses **$SUNW_HOSTNAME** as the platform name.

This command is effective only when used in a **.postrc** file in the current directory. If the **.postrc** file used is not in the local directory, POST ignores this command and generates a warning message.

poll_timeout_mult *mult_factor* (Level 2)
The timeout values used when tasks are assigned to host processors are multiplied by this integer value, which is in the range 1 to 100, inclusive. Sometimes, when very verbose messages have been enabled in a system with many processors, a healthy processor is so delayed by the poll and display overhead that it exceeds the normal time allowed, causing it to be marked as failed. This command effectively extends all such timeout allowances. It could also be used as a field workaround for inappropriately short timeout values. See also **servmgr_time_report** and **no_poll_timeouts**.

port_idle_use_pause (Level 3)
Use the less flexible PORT_PAUSE PC BootBus feature instead of the PORT_IDLE feature.

print_all_errors (Level 2)
POST normally prints only the first message or two from any particular test; the test then quits. This command causes it to print all error messages and to continue testing.

Turning this function on can cause so many failure messages that the test timeout is exceeded, and a perfectly good processor can be marked as failed. To compensate, use the **poll_timeout_mult** command, as well.

proc_freq_check_percent {*float_%* **| off**} (Level 2)
During the JTAG integrity test phase, POST reads the maximum rated speed of each processor, and compares it to the speed of the system-wide processor clock distributed from the control board. If the clock speed exceeds the rated speed by greater than the specified percent (*float_%*), POST fails the processor. The default is 0.5%. If the argument is **off**, POST does not do the check.

proc_freq_ratio {**2 | 2:1 | 3 | 3:1 | 3:2**} (Level 2)
Use the specified ratio of processor frequency to system interconnect frequency. This value is used for reporting processor frequency in the post2obp structure, and for checking against a processor's rated maximum speed, in teststand environments lacking a real control board which can measure and report this frequency.

In systems that do have a control board, specifying this value causes POST to require that the measured ratio be within 0.5 percent of this value. If it is not specified, the measured ratio must be within 0.5 percent of the target ratio read from the SNMP server. See also the command **interconnect_MHz**.

proc_timestamp *interval_secs* (Level 2)
When printing messages from the downloaded processor code of all host processors (those messages preceded by {*board.module*}), also print a timestamp of when the messages occurred; but print timestamps for any particular processor only when that processor prints a message and at least *interval_secs* has elapsed since the last timestamp for that processor. The form of the timestamp is {*board.module*} {*day/date/time*}, with 1-second resolution.

interval_secs is an integer for the number of seconds in the range of 0 to 86400 (24 hours; 60 times 60 times 24). Specifying 0 causes every message to get a timestamp.

qt_missrefresh_err_enbl (Level 3)
See the **quickturn** command. POST's default behavior with **quickturn** specified is to mask missed refresh errors in the MC ASIC. **qt_missrefresh_err_enbl** causes these refresh errors to be enabled to cause an arbstop. POST quietly ignores this command if the **quickturn** command is not invoked.

quickturn *KHz Mem_refresh_interval* (Level 3)

Declare that the system is running in a Quickturn (tm) emulation environment. *KHz* declares the system clock speed, and must be in the range 100 to 2000 (2 MHz), inclusive. *Mem_refresh_interval* is the refresh interval in system clock cycles for configuring the MC, and must be in the range 8 to 0xFFF = 4095. Both arguments must be specified.

Poll timeouts are set at a multiple of their normal value, based on the declared system frequency. The Memory Controller is set to use the specified refresh interval. Other configuration properties may be changed appropriately when this command is invoked.

recordstop_poll (Level 2)

When polling for arbstop, also poll for recordstop.

redlist_file {*filepath* | **none**} (Level 2)

Command Line Equivalent: **–R**

Use the values in the specified **redlist**(4) file rather than those in the default **redlist**(4) file, **$SSPVAR/etc/***platform_name***/redlist**. If **none** is specified, POST reads no redlist file. See **blacklist**(4) and **redlist**(4).

Be careful when using **redlist_file** on a production system; other SSP software does not know when POST is using a nonstandard **redlist**(4) file.

scantool_simulate (Level 3)

Intercept normal calls to the JTAG library functions and (crudely) simulate them internal to POST. This command is useful to Sun Microsystems Engineering during development to allow code debug when no real or hardware simulated system is present.

servmgr_time_report (Level 2)

When a host processor is assigned a task, such as running a test, POST's server manager sets a timeout by which it must complete or be considered failed. This flag causes the server manager to report how much time remained in the timeout when the processor reported back. This information is useful in determining the appropriate value of these timeouts.

shared_mem *brd boardmask* (Level 3)

Configure the specified system board to provide shared memory to the system boards in *boardmask*. *boardmask* must not include any boards in the specified board's domain, including itself. See the **domain** command for more information. This command does not configure the shared memory BAR/LAR in the CICs; host software normally does so. However, However, see the command **shmem_barlar**.

shmem_barlar *bar lar* (Level 3)

Configure the JTAG CIC shared memory base and limit address registers with the given *bar* and *lar* values. This command affects POST only for boards declared by the **shared_mem** command to have shared memory, and only a single set of *bar/lar* values is used for all such boards. The default, and the value for boards without shared memory, is 1 for *bar* and 0 for *lar* (no valid range). For

boards with shared memory, these two values are added to the board's base physical address (mod 2**41) and the result is that board's *bar/lar* configuration. These two registers count increments of 64 Kbytes, with implied [15:0] = 0. Any 25-bit value is acceptable for both *bar* and *lar*, and *bar* may be greater than *lar*.

skip_phase *phase_name* (Level 2)

Skip the named POST phase. Only one phase can be specified per command, but any number of **skip_phase** commands can appear in **.postrc.** To print the available phase names, execute **hpost -?postrc**. These names are also printed as progress messages when POST runs if the verbose level is sufficiently high. See also the **interactive_use_postrc_skips** and **no_skip_phase_covers_npb** commands.

skip_test *test_id* (Level 2)

Skip the designated test. Only one test can be specified per command, but any number of **skip_test** commands can appear in **.postrc.** *test_id* is a number in the range 0 to 0x1FF, used as index and identifier for a POST test. The Test ID is printed, along with the test name, before each test is run, if the verbose level is high enough. Getting the Test ID in this way is the most appropriate method for users to determine the ID they should use with this command. The Test ID is also printed when tests fail.

This command provides higher resolution control over execution than **skip_phase**. It might also be a field workaround for bugs or hardware changes that cause inappropriate failures of some test.

Some POST tests may be internally declared non-skippable. A check for such declarations occurs at test execution time, rather than when POST reads the **.postrc** command. If a **skip_test** command is present for such a test, POST prints a single message that it is ignoring the command. See also the **interactive_use_postrc_skips** command.

sysreset_ignore_redlist (Level 3)

Allow redlisting of some system resources that physically do not exist, without causing POST to skip issuing the system reset during the *init_reset* phase. This command is for use only in certain development environments. See **redlist**(4).

system_red_any_fail (Level 2)

Upon any POST failure, immediately fail and internally redlist the entire host system. This command is sometimes useful for doing postmortem analysis of failures in a debugging environment.

Since it obviously subverts any possibility of POST managing to configure around failed components, Sun Microsystems does not intend this command for normal use. You can think of it as a more extreme form of the command **board_red_any_fail**.

trace_print *board proc_module* (Level 2)

Enable the extremely verbose **tprintf** trace prints from the specified host processor. This command is off by default. To control trace prints for multiple processors, use more than one instance of this command.

verbose *level* (Level 1)

Command Line Equivalent: **–v**

Control the amount of progress information printed by POST. Higher verbose levels result in more verbose output. *level* is an integer from 0 to 255, inclusive. For more information about levels, execute **hpost –?verbose**.

The system administrator can use this command, but must be aware that high verbose levels slow POST execution. In some extreme cases, it slows it enough to cause timeouts.

verbose_print *board proc_module* (Level 2)

Control the verbose **vprintf** printing from an individual host processor. This command turns on control for one specific processor. To do so for multiple processors, use multiple instances of this command. This command is enabled for all processors if the verbose level is high enough.

SEE ALSO

bringup(1M), **hpost**(1M), **blacklist**(4), **redlist**(4).

NAME

redlist – list of system resources not to be touched

DESCRIPTION

Caution: Contact your service provider before using this file. Certain changes to this file may cause your system to become unusable.

$SSPVAR/etc/*platform_name***/redlist** is an ASCII file that enables the system administrator or root to restrict, from the SSP, the configuration of the host system. It lists components that POST cannot touch, and whose state POST cannot change. POST reads the **redlist** file before preparing the system for booting, and passes along to OBP a list of only those components that have been successfully tested; those on the redlist are excluded.

The **redlist** file is very restrictive, and used in a very limited way, mostly in specialized debug in the lab. For example, a Sun Microsystems engineer may use it while working with experimental hardware that is missing components with which POST expects to communicate, even if just to freeze them.

Redlisted components are also considered effectively blacklisted.

Redlisting components carries a price in capability and performance. If any component on a board is redlisted, POST cannot reset that board; since some failures require a board reset to clear, the entire board becomes unusable and, in some cases, the entire system can become unusable.

Caution: Never use redlisting if blacklisting will do.

In the **redlist** file:

- Keywords are not case-sensitive.
- Any part of a line that starts with **#** is a comment.
- Numbers are assumed decimal unless preceded by **0x**, which indicates hexadecimal. Exception: a board number entered as one of [a - f] or [A - F] is assumed hexadecimal.
- Each line has one and only one keyword.
- The same keyword can be used on more than one line.
- Each keyword has one or more arguments. Each argument is shown as an integer and multiple integers are separated by a period.

Keywords

All value ranges shown below are inclusive.

sysbd *board*

Do not test or configure the specified system board, where *board* = 0–15.

proc *board.pmod*

Do not test or configure the specified processor within the specified system board, where *board* = 0–15 and *pmod* = 0–3.

abus *abus*

Do not test or configure the specified address bus, where *abus* = 0–3. The meaning of this command is that the corresponding CIC ASIC on all system boards is marked red. See the keyword **cic**, below.

dbus *dbus*

Do not test or configure the specified 72-bit half of the 144-bit data router, where *dbus* = 0–1. The meaning of this command is that the corresponding half of the local data router on all system boards is marked red. See the keyword **ldpath**,

below.

ioc *board.ioctl*
Do not test or configure the specified I/O controller within the specified system board, where *board* = 0–15 and *ioctl* = 0–1.

scard *board.ioctl.slot*
Do not test or configure the specified I/O adapter card within the specified I/O controller, which is within the specified system board. *board* = 0–15, *ioctl* = 0–1 and *slot* = 0–3.

mem *board*
Do not test or configure memory on the specified system board, where *board* = 0–15

mgroup *board.group*
Do not test or configure the specified group of memory DIMMs within the specified system board, where *board* = 0–15 and *group* = 0–3.

cplane *half_centerplane*
Do not test or configure the specified Enterprise 10000 half-centerplane, which contains two buses and 72 bits of the global data router, where *half_centerplane* = 0–1. The meaning of this command is equivalent to the combination of the **abus** and **dbus** commands for the buses contained in this half-centerplane.

pc *board.pc*
Do not test or configure the specified port controller ASIC within the specified system board, where *board* = 0–15 and *pc* = 0–2.

xdb *board.xdb*
Do not test or configure the specified data buffer ASIC within the specified system board, where *board* = 0–15 and *xdb* = 0–3.

cic *board.cic*
Do not test or configure the specified coherent interface controller ASIC within the specified system board, where *board* = 0–15 and *cic* = 0–3 (*cic* corresponds to an address bus on that board).

ldpath *board.dbus*
Do not test or configure the specified 72-bit half of the 144-bit local data router within the specified system board, where *board* = 0–15 and *dbus* = 0–1.

EXAMPLE

```
# Sun Microsystems, Inc.
sysbd 3 5 0xA     # Disable system boards 3, 5 and 10.
sysbd 3 5 A       # Disable system boards 3, 5 and 10.
PROC 4.0 6.2      # Disable Processor 0 on System Board 4, and
                  # Processor 2 on System Board 6.
ScarD 3.0.1       # Disable I/O Adapter 1 on I/O Controller 0 on
                  # System Board 3.
mem 2             # Disable all memory on System Board 2.
mlimit 0xE 2.64   # Restrict use of Memory DIMM Group 2 on System
                  # Board 14 to 64MB.
cIc 1.2           # Disable CIC ASIC 2 on System Board 1.
```

SEE ALSO **hpost**(1M), **blacklist**(4)

NAME

ssp_resource – SSP processes resource file

DESCRIPTION

Note: Do not edit this file manually. Doing so may interfere with its use by SSP processes.

ssp_resource is a general-purpose resource file used by SSP processes. It contains the following information, with period-separated fields:

platform_type.resource_name:resource_value

platform_type.id_name.resource_name:resource_value

where:

platform_type	The platform type of the SSP. Currently, this value is always **Ultra-Enterprise-10000**.
resource_name	The name of the resource. This name must be unique within this file.
resource_value	The value of the resource.
id_name	The name of the SSP process or daemon that currently owns this resource.

EXAMPLE

```
Ultra-Enterprise-10000.confMemAddrMap:000102030405 ... OEOF
Ultra-Enterprise-10000.edd.scripts:sysbrdtemp.tcl,centerplanetemp.tcl
Ultra-Enterprise-10000.HIGHasic:60000
```

FILES

$SSPVAR/.ssp_private/ssp_resource

SEE ALSO

cbs(1M), **edd**(1M), **snmpd**(1M)

NAME

ssp_to_domain_hosts – hostname/domainname file

DESCRIPTION

Caution: Do not make any changes in this file. It is updated as necessary by **domain_create**(1M), **domain_remove**(1M) and **domain_rename**(1M). Altering it manually adversely affects the behavior of the SSP.

$SSPVAR/.ssp_private/ssp_to_domain_hosts maintains a list of domain names, each with the name of its assigned SSP hostname to its right, as shown in the example, below.

EXAMPLE

```
#
# Format: domain_name   ssp_hostname
#
xf2   xf2-ssp
ts4   ts4-ssp
```

SEE ALSO

domain_create(1M), **domain_remove**(1M), **domain_rename**(1M)