Solaris 2.5.1 Hardware: 4/97 Reference Manual for SMCC-Specific Software

Sun Microsystems Computer Company
2550 Garcia Avenue
Mountain View, CA 94043 USA
415 960-1300 fax 415 969-9131

Part No: 802-7674-10
Revision A, April 1997

NAME

symon – bring up the Solstice SyMON system monitor console

SYNOPSIS

symon [–**colorMap**] [–**cm**] [∗**colorMap**]
[–**dragthreshold** *pixels*] [∗**dragthreshold** *pixels*]
[–**flashDuration** *milliseconds*] [–**fd** *milliseconds*] [∗**flashDuration** *milliseconds*]
[–**flashInterval** *milliseconds*] [–**fi** *milliseconds*] [∗**flashInterval** *milliseconds*]
[–**heartbeatInterval** *intervals*] [–**hi** *intervals*] [∗**heartbeatInterval** *intervals*]
[–**interval** *intervals*] [–**i** *intervals*] [∗**interval** *intervals*]
[–**installDir** *path*] [–**I** *path*] [∗**installDir** *path*]
[–**minWait** *seconds*] [–**mw** *seconds*] [∗**minWait** *seconds*]
[–**pruneTime** *minutes*] [–**pt** *minutes*] [∗**pruneTime** *minutes*]
[–**session** *file*] [∗**session** *file*] [–**target** *machine*] [–**t** *machine*] [∗**target** *machine*]
[–**tempPruneTime** *minutes*] [–**tpt** *minutes*] [∗**tempPruneTime** *minutes*]
[–**vtsui** *file*] [∗**vtsui** *file*] [–**help**] [–**h**] [–**?**]

AVAILABILITY

SUNWsymon

DESCRIPTION

symon is the primary user interface to the Solstice SyMON system monitor. Invoking symon brings up the launcher window, from which the seven Solstice SyMON consoles are launched:

- Event Viewer
- Kernel Data Catalog
- Physical View
- Log Viewer
- Logical View
- Process Viewer
- On-line Diagnostics

For further details on the operation of **symon** please see the *Solstice SyMON User's Guide.*

OPTIONS

–**colorMap**	Use a private color map for the Launcher and Physical View windows to ensure correct colors in the images. May result in colormap flashing of images and of other applications, such as the Netscape browser (default is to use the default colormap).
–**cm**	Same as –**colorMap**
∗**colorMap**	Same as –**colorMap**
–**dragthreshold**	Sets the mouse drag threshold for Sysmeters (default is 10 pixels).
∗**dragthreshold**	Same as –**dragthreshold**
–**flashDuration**	Set time that flashes of the system indicator on the launcher console will last (default is 30 milliseconds).

–fd	Same as **–flashDuration**
***flashDuration**	Same as **–flashDuration**
–flashInterval	Set time interval between flashes of the system indicator on the launcher console (default is 2000 milliseconds).
–fi	Same as **–flashInterval**
***flashInterval**	Same as **–flashInterval**
–heartbeatInterval	Set the polling time for the heartbeat check for agents (default is 10 intervals).
–hi	Same as **–heartbeatInterval**
–installDir	Set the directory root to examine for tcl files, etc. (default is **/opt/SUNWsymon**).
–I	Same as **–installDir**
***installDir**	Same as **–installDir**
–interval	Set the polling interval for agents (default is 10 intervals).
–i	Same as **–interval**
–minWait	Set a minimum wait time between polls / updates (default is 1 second between the end of one poll and the start of the next).
–mw	Same as **–minWait**
–pruneTime	Time after which unchanged data (old processes) is pruned from the **sm_krd** (Kernel Reader) hierarchy (default is 120 minutes).
–pt	Same as **–pruneTime**
–session	Specifies a Tcl file, which defines the layout and contents of a Solstice SyMON instance. This file is read when Solstice SyMON starts up to restore a previously saved layout.
–tempPruneTime	Time after which unchanged Config Reader data (board temperature) will be pruned from **sm_configd** hierarchy (default is 1440 minutes).
–tpt	Same as **–tempPruneTime**
–target	System to be monitored.
–t	Same as **–target**
–vtsui	Name of SunVTS user interface binary (default is **vtsui**).
–help	Listing of arguments.
–h	Same as **–help**
–?	Same as **–help**

ENVIRONMENT

TCL_LIBRARY	Location of the Tcl library.
XFILESEARCHPATH	Location of the X Files.
DTAPPSEARCHPATH	Location of the CDE X Defaults files.
DTDATABASESEARCHPATH	Location of the CDE database files.
DTHELPSEARCHPATH	Location of the CDE help files.
XMICONSEARCHPATH	Location of the **symon** icons.

FILES

common.tcl	Common Tcl routines for the display.
cpu_utilization.tcl	Tcl routines to define the chart for CPU utilization.
disk_service_time.tcl	Tcl routines to define the chart for disk service time.
memory_usage.tcl	Tcl routines to define the chart for memory usage.
init.tcl	Tcl routines to initialize **symon.**
queue_lengths.tcl	Tcl routines to define the chart for queue lengths.
sysmeter.tcl	Tcl routines to define the chart for System Meters.

NOTES

Solstice SyMON uses ASCII-format Tcl files as a means of saving and restoring the state of the program's GUI. Currently, this feature only works for system meters, the process viewer, and the event viewer. Some Tcl files are provided with the Solstice SyMON product to serve as examples. Normally these Tcl files should be created by using the GUI to configure the desired windows, and then saved by invoking save in a system meter (to save the state of one system meter) or in the kernel data catalog window (to save the state of all system meters).

Symon examines or creates the directory **$HOME/.symon** and creates a directory structure there to contain Tcl files that the user has created and links to Tcl files in the official installation. The purpose is that both sets of files may be browsed easily at the same time in a single file selection dialog.

When a Solstice SyMON release is run for the first time by a user, it will create symbolic links in the user's directory (**$HOME/.symon/lib/tcl/C**) that point to any Tcl files in the installation directory (usually **/opt/SUNWsymon/lib/tcl/C**). Thus, any new Tcl files in a new release will be picked up. If the user has files or links in their directory that match the names of files in the official directory, then links will be removed and remade to the official files. User files matching official file names will result in a dialog box in Solstice SyMON that explains the options the user has at that point: Either to keep the local file, to remove it and have Solstice SyMON link to the official version, or to manually merge the two files.

SEE ALSO

sm_configd(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

arraymon – Array Monitor Daemon

SYNOPSIS

arraymon

DESCRIPTION

arraymon is the disk array daemon process. It performs these major functions:

1. Monitoring of the error information maintained by the disk array controllers.
2. Reporting of events that require operator attention in a manner selected by the user via the rmparams file and the rmscript file.
3. Launching of the **parityck** utility at the designated time, if the parity check option is enabled.

FILES

arraymon maintains logs of the messages currently outstanding on the system console and in the file **/etc/raid/rmlog.log.** In addition, all error information is written to the system error log **/var/adm/messages**).

EXIT STATUS

None - arraymon runs as a daemon process and never exits.

NOTES

arraymon is typically initiated when the system is booted and runs until the system is shut down.

SEE ALSO

lad(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

contrast – Adjust system screen contrast

SYNOPSIS

/usr/openwin/bin/contrast [–**k**] [–**ud** [**step**]]

AVAILABILITY

SUNWpmow

DESCRIPTION

contrast (1M) is a binary that can be used to adjust the system screen contrast level.

OPTIONS

–**u step**	Increase contrast by step.
–**d step**	Decrease contrast by step.
–**k**	This is the repeater mode for the binary. The binary expects the letter 'k' on its STDIN and increases or decreases the contrast by one until STDIN is closed by the other side. This is the mode in which the binary interacts with the **speckeysd (1M)** daemon.

NOTES

The contrast level is adjusted by communicating with the Power Management driver, **pm (7)** , through ioctls provided by the Power Management Framework.

SEE ALSO

speckeysd(1M), **pm**(7)

NAME

cvcd – virtual console daemon

DESCRIPTION

cvcd is a server that resides on an Enterprise 10000 host or domain. It accepts connections from **netcon_server**(1M) on an SSP to create a Network Console Window on that SSP. The Network Console Window is able to read data from, and possibly send data to, the host or domain. This process takes place via the SSP command **netcon**(1M). See **netcon_server**(1M) and **netcon**(1M)**in** *man Pages(1M): Ultra Enterprise 10000 SSP Administration Commands*.

When you execute **netcon**(1M) in an SSP Window, **netcon_server**(1M) connects with the **cvcd** daemon running on the host or domain specified in the SSP's SUNW_HOSTNAME environment variable, and the window becomes a Host Console Window.

The console session ends when you exit the session, **netcon_server** terminates, or a network failure occurs. If **cvcd** dies, **netcon** gets data from JTAG through the control board.

cvcd is normally started during boot. Only one **cvcd** process at a time can run on the host.

Caution: **cvcd** uses the file **ssphostname,** which resides on the host. If the SSP has been renamed, **ssphostname** must be edited to reflect that change.

SEE ALSO

Ultra Enterprise 10000 SSP 3.0 User's Guide

cvc(7), **cvcredir**(7), **netcon**(1M), **netcon_server**(1M), **services**(4)

NAME

dr_daemon – dynamic reconfiguration daemon

SYNOPSIS

dr_daemon [–**a**]

DESCRIPTION

The **dr_daemon** is an RPC program that provides the interface to the Dynamic Reconfiguration (DR) driver, **/dev/dr .** The Hostview and DR applications provide the user interface to DR. See **hostview**(1M) in *man Pages(1M): Ultra Enterprise 10000 SSP Administration Commands* and **dr**(1M) in *man Pages(1M): DR Administration Commands*.

OPTIONS

–**a** Disable communications with the Alternate Pathing daemon. See **ap_daemon**(1M) in *man Pages(1M): Alternate Pathing Administration Commands*.

Configuration Information

The **/usr/platform/sun4u1/sbin/dr_daemon** RPC program name is DRPROG, its RPC program number is 300326, and its underlying protocol is TCP. It is invoked as an inetd server using the TCP transport. The UID required for access to the daemon is ssp. This UID can be a non-login UID.

The entry for the daemon in the **/etc/inetd.conf** file is:

```
300326/4  tli rpc/tcp  wait root /usr/platform/sun4u1/sbin/dr_daemon dr_daemon
```

The daemon's only clients are Hostview and DR. Hostview provides a GUI interface; **dr1M)** is a command-line interface for non-windowing environments. The DR daemon uses **syslog**(3) to report status and error messages, which are logged with the LOG_DAEMON facility and the LOG_ERR and LOG_NOTICE priorities.

The **dr_daemon** communicates via RPC with the Alternate Pathing (AP) daemon (see **ap_daemon (1M)** in *man Pages(1M): Alternate Pathing Administration Commands*) to notify the AP software when controllers are attached to and detached from the system, or to gather information about the system configuration.

SEE ALSO

Dynamic Reconfiguration User's Guide
Alternate Pathing 2.0 User's Guide

dr(7) in this reference manual

ap(1M), **ap_daemon(1M)** in *man Pages(1M): Alternate Pathing Administration Commands*

dr(1M) in *man Pages(1M): DR Administration Commands*

hostview(1M), **hpost(1M)** in *man Pages(1M): Ultra Enterprise 10000 SSP Administration Commands*

add_drv(1M), **drvconfig**(1M), **devlinks**(1M), **disks**(1M), **inetd**(1M), **ports**(1M), **tapes**(1M), **prtconf**(1M), **syslog**(3) in *man Pages(1M): System Administration Commands*

NAME

dtpower – desk–top power manager, system and device power management tool

SYNOPSIS

dtpower [*generic–tool–arguments*] [–**sampleTime** *n* | –**st** *n*]
[–**warnTime1** *n* | –**wt1** *n*] [–**warnTime2** *n* | –**wt2** *n*]
[–**nobell**]

AVAILABILITY

SUNWpmow

DESCRIPTION

dtpower provides a graphical user interface (GUI) to the power management system (see **pm**(7)). It allows the user to configure certain power manageable devices to shutdown after a specified period of inactivity. Different hardware platforms support different devices. Most platforms allow power management of display(s). Some platforms allow power management of disk drives. The set of power configurations for all devices is called a power profile.

dtpower also displays the current autoshutdown settings (see **powerd**(1M)). If **dtpower** is run as root, these settings may be changed. These settings are not included in a power profile.

If a battery is present, **dtpower** monitors the battery level. If the system is running from the battery, **dtpower** displays low power warnings when the battery charge is running low. **dtpower** maintains two device power profiles – one for use on AC and one for use with the battery. This enables you to customize your device power settings, depending on your power source. **dtpower** switches profiles automatically when the machine's power supply changes. There may be a small delay (about 30 seconds) before **dtpower** notices a change in power source.

You must be console owner or root to run **dtpower.**

OPTIONS

generic–tool–arguments
dtpower accepts the generic tool arguments described in **xview**(7).

–**sampleTime n**
–**st n**
dtpower continually checks the battery capacity, if a battery is present. This option sets the period of this check. The default is 10 seconds.

–**warnTime1 n**
–**wt1**
–**warnTime2 n**
–**wt2**
dtpower displays two warnings of low battery power. These options set the time before battery exhaustion at which the warnings will occur. The default warning times are at 10 and 5 minutes. Note that **powerd**(1M) will shut the system down when the battery is exhausted.

–**nobell**
By default, whenever **dtpower** displays a warning dialog, it sounds a bell. This option disables the bell.

USAGE

dtpower operates via a set of pull-down menus, slider(s) and buttons in a control panel. From the control panel you may access one other panel, the autoshutdown panel.

The Control Panel

Menu Bar

File

Exit Exits the application. If you have pending changes, you will prompted to apply or discard them before exiting.

Help

Help

Displays an overview of the **dtpower** application.

Information

With Battery

The charge level of the battery is displayed. If the battery is the connected power source, then the estimated battery life is displayed below the charge gauge.

You can select which device power profile to edit using the toggle buttons above the slider(s). Note that the active profile is determined by your power source, not the toggle buttons.

Without battery

The power profile displayed is for an AC power supply. There is no access to the battery power profile.

Slider(s)

Screen This slider shows the amount of time the keyboard and mouse will be unused before the screen turns off. To change this time, move the slider and select apply. To turn the screen on, move the mouse or press a key.

Disk This slider shows the amount of time the disk will be idle before spinning down. This is not available on all platforms. The disk will automatically spin up the next time it is needed.

Buttons

Apply This applies any changes to your active power profile and saves all settings into *$HOME/.pmrc* so they are remembered the next time the application is started.

Reset to Standard

This resets the active power profile to its default values and saves these into *$HOME/.pmrc.*

Set Autoshutdown...

This brings up the autoshutdown panel.

The Autoshutdown Panel

As root

This panel allows you to view and edit the parameters governing

autoshutdown. The first box adjusts the amount of time the console keyboard and mouse must be unused before the system will auto-shutdown.

The toggle buttons beneath determine the times when auto-shutdown is in effect at all.

OK This applies any changes made and saves them to **power.conf**(4) as the default settings.

Cancel Dismisses the window and discards any changes

Help Displays a brief overview text.

As a regular user

This panel allows you to view the current settings. Changes are not permitted.

OK Disabled

Cancel Dismisses window

Help Displays a brief overview text.

FILES

$HOME/.pmrc Per user customized power profile
/etc/power.conf System-wide power configuration profile
/usr/openwin/lib/app-defaults/Dtpower
Text messages file

SEE ALSO

cpr(7), **pm**(7), **power.conf**(4), **pmconfig**(1M), **powerd**(1M)

NAME

ffbconfig – configure the FFB Graphics Accelerator

SYNOPSIS

/usr/sbin/ffbconfig [–**dev** *device-filename*]
[–**res** *video-mode* [**now** | **try**] [**noconfirm** | **nocheck**]]
[–**file machine | system**]
[–**deflinear true | false**]
[–**defoverlay true | false**]
[–**linearorder first | last**]
[–**overlayorder first | last**]
[–**expvis enable | disable**]
[–**sov enable | disable**]
[–**maxwids** *n*] [–**propt**] [–**prconf**] [–**defaults**]

/usr/sbin/ffbconfig [–**propt**] [–**prconf**]

/usr/sbin/ffbconfig [–**help**] [–**res** ?]

AVAILABILITY

SUNWffbcf

DESCRIPTION

ffbconfig configures the FFB Graphics Accelerator and some of the X11 window system defaults for FFB.

The first form of **ffbconfig** shown in the synopsis above stores the specified options in the OWconfig file. These options will be used to initialize the FFB device the next time the window system is run on that device. Updating options in the OWconfig file provides persistence of these options across window system sessions and system reboots.

The second and third forms which invoke only the –**prconf**, –**propt**, –**help**, and –**res** ? options do not update the OWconfig file. Additionally, for the third form all other options are ignored.

Options may be specified for only one FFB device at a time. Specifying options for multiple FFB devices requires multiple invocations of **ffbconfig**.

Only FFB-specific options can be specified through **ffbconfig**. The normal window system options for specifying default depth, default visual class and so forth are still specified as device modifiers on the openwin command line (see the Xsun(1) manual page in the Openwindows Reference Manual).

The user can also specify the OWconfig file that is to be updated. By default, the machine-specific file in the /etc/openwin directory tree is updated. The –**file** option can be used to specify an alternate file to use. For example, the system-global OWconfig file in the /usr/openwin directory tree can be updated instead.

Both of these standard OWconfig files can only be written by root. Consequently, the **ffbconfig** program, which is owned by the root user, always runs with setuid root permission.

OPTIONS

–**dev** *device-filename*
Specifies the FFB special file. The default is **/dev/fbs/ffb0**.

–file machine | system

Specifies which OWconfig file to update. If **machine**, the machine-specific OWconfig file in the /etc/openwin directory tree is used. If **system**, the global OWconfig file in the /usr/openwin directory tree is used. If the file does not exist, it is created.

–res *video-mode* [**now** | **try** [**noconfirm** | **nocheck**]]

Specifies the video mode used to drive the monitor connected to the specified FFB device.

The format of these built-in video modes is:

widthx**height**x**rate**

where **width** is the screen width in pixels, **height** is the screen height in pixels, and **rate** is the vertical frequency of the screen refresh. The **s** suffix of 960x680x112s and 960x680x108s means that these are stereo video modes. The **i** suffix of 640x480x60i and 768x575x50i designates interlaced video timing. If absent, non-interlaced timing will be used. As a convenience, **–res** also accepts formats with '@' (at sign) in front of the refresh rate instead of x. For example: 1280x1024@76. Note, some video-modes are supported only on certain revisions of FFB. Also, some video-modes, supported by FFB, may not be supported by the monitor. The list of video-modes supported by the FFB device and the monitor can be obtained by running **ffbconfig** with the **–res** ? option (the third form shown in the command synopsis above). A list of all possible video-modes supported on FFB is shown below.

1024x768x60	
1024x768x70	
1024x768x75	
1024x768x77	
1024x800x84	
1152x900x66	
1152x900x76	
1280x800x76	
1280x1024x60	
1280x1024x67	
1280x1024x76	
960x680x112s	(Stereo)
960x680x108s	(Stereo)
640x480x60	
640x480x60i	(Interlaced)
768x575x50i	(Interlaced)
1440x900x76	(hi-res)
1600x1000x66	(hi-res)
1600x1000x76	(hi-res)
1600x1280x76	(hi-res)
1920x1080x72	(hi-res)
1920x1200x70	(hi-res)

Symbolic names

For convenience, some of the above video modes have symbolic names defined for them. Instead of the form **width** x **height** x **rate**, one of these names may be supplied as the argument to –**res**. The meaning of the symbolic name **none** is that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

Name	Corresponding Video Mode
svga	1024x768x60
1152	1152x900x76
1280	1280x1024x76
stereo	960x680x112s
ntsc	640x480x60i
pal	768x575x50i
none	(see text above)

The –**res** option also accepts additional, optional arguments immediately following the video mode specification. Any or all of these may be present.

now If present, not only will the video mode be updated in the OWconfig file, but the FFB device will be immediately programmed to display this video mode. (This is useful for changing the video mode before starting the window system).

Note – It is inadvisable to use this suboption with **ffbconfig** while the configured device is being used (e.g. while running the window system); unpredictable results may occur. To run **ffbconfig** with the now suboption, first bring the window system down. If the now suboption is used within a window system session, the video mode will be changed immediately, but the width and height of the affected screen won't change until the window system is exited and reentered again. In addition, the system may not recognize changes in stereo mode. Consequently, this usage is strongly discouraged.

noconfirm Using the –**res** option, the user could potentially put the system into an usable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this, the default behavior of **ffbconfig** is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. The noconfirm option instructs **ffbconfig** to bypass this confirmation and to program the requested video mode anyway. This option is useful when **ffbconfig** is being run from a shell script.

nocheck If present, the normal error checking based on the monitor sense code (described above) will be suspended. The video mode specified by the user will be accepted regardless of whether it is appropriate for

the currently attached monitor. (This option is useful if a different monitor is to be connected to the FFB device). *Use of this option implies noconfirm well*.

try If present, the specified video mode will be programmed on a trial basis. The user will be asked to confirm the video mode by typing 'y' within 10 seconds. Or the user may terminate the trial before 10 seconds are up by typing any character. Any character other than 'y' or carriage return is considered a no and the previous video mode will be restored and **ffbconfig** will not change the video mode in the OWconfig file (other options specified will still take effect). If a carriage return is typed, the user is prompted for a yes or no answer on whether to keep the new video mode. This option implies the now suboption (see the warning note on the now suboption).

FFB possesses two types of visuals: linear and nonlinear. Linear visuals are gamma corrected and nonlinear visuals are not. There are two visuals that have both linear and nonlinear versions: 24-bit TrueColor and 8-bit StaticGray.
If true, the default visual is set to the linear visual that satisfies other specified default visual selection options (specifically, the Xsun(1) defdepth and defclass options described in the OpenWindows Reference Manual).
If false, or if there is no linear visual that satisfies the other default visual selection options, the non-linear visual specified by these other options will be chosen to be the default.
This option cannot be used when the –**defoverlay** option is present, because FFB doesn't possess a linear overlay visual.

–**defoverlay true** | **false**
The FFB provides an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the FFB visuals. This is called the overlay visual. Windows created in this visual will not damage windows created in other visuals. The converse, however, is not true. Windows created in other visuals will damage overlay windows. This visual has (256 – maxwids) number of opaque color values (refer to the –**maxwids** option).
If the value of this option is true, the overlay visual will be made the default visual.
If false, the nonoverlay visual that satisfies the other default visual selection options, such as defdepth and defclass, will be chosen as the default visual. See the Xsun(1) manual page in the OpenWindows Reference Manual.
Whenever –**defoverlay true** is used, the default depth and class chosen on the openwin command line must be 8-bit PseudoColor. If not, a warning message will be printed and the –**defoverlay** option will be treated as false.
This option cannot be used when the –**deflinear** option is present, because FFB doesn't possess a linear overlay visual.

–**linearorder first** | **last**
If true, linear visuals will come before their non-linear counterparts on the X11 screen visual list for the FFB screen. If false, the nonlinear visuals will come

before the linear ones.

–**overlayorder** first | last
If true, the depth 8 PseudoColor Overlay visual will come before the non-overlay visual on the X11 screen visual list for the FFB screen. If false, the non-overlay visual will come before the overlay one.

–**expvis** enable | disable
If enabled, OpenGL Visual Expansion will be activated. Multiple instances of selected visual groups (8-bit PseudoColor, 24-bit TrueColor ... etc) can be found in the screen visual list.

–**sov** enable | disable
If enabled, the root window's SERVER_OVERLAY_VISUALS property will be advertised. SOV visuals will be exported and their transparent types, values and layers can be retrieved through this property. If disabled, the SERVER_OVERLAY_VISUALS property will not be defined. SOV visuals will not be exported.

–**maxwids** *n*
Specifies the maximum number of FFB X channel pixel values that are reserved for use as window IDs (WIDs). The remainder of the pixel values in overlay colormaps are used for normal X11 opaque color pixels.
The reserved WIDs are allocated on a first-come first-serve basis by 3D graphics windows (such as XGL), MBX windows, and windows that have a non-default visual.
The X channel codes 0 to (255 – n) will be opaque color pixels. The X channel codes (255 – n + 1) to 255 will be reserved for use as WIDs. Legal values: 1, 2, 4, 8, 16, 32.

–**defaults**
Resets all option values to their default values.

–**propt** Prints the current values of all FFB options in the OWconfig file specified by the –**file** option for the device specified by the –**dev** option. Prints the values of options as they will be in the OWconfig file after the call to **ffbconfig** completes. This is a typical display:

--- OpenWindows Configuration for /dev/fbs/ffb0 ---
OWconfig: machine
Video Mode: NONE
Default Visual: Non-Linear Normal Visual
Visual Ordering: Linear Visuals are last
Overlay Visuals are last
OpenGL Visuals: disabled
SOV: disabled
Allocated WIDs: 32

–**prconf**
Prints the FFB hardware configuration. This is a typical display:

--- Hardware Configuration for /dev/fbs/ffb0 ---
Type: double-buffered FFB2 with Z-buffer
Board: rev x
PROM Information: @(#)ffb2.fth x.x xx/xx/xx
FBC: version x
DAC: Brooktree 9068, version x
3DRAM: Mitsubishi 1309, version x
EDID Data: Available - EDID version 1 revision x
Monitor Sense ID: 4 (Sun 37x29cm RGB color monitor)
Monitor possible resolutions: 1024x768x60, 1024x768x70, 1024x768x75, 1152x900x66, 1152x900x76, 1280x1024x67, 1280x1024x76, 960x680x112s, 640x480x60
Current resolution setting: 1280x1024x76

–help Prints a list of the **ffbconfig** command line options, along with a brief explanation of each.

DEFAULTS

For a given invocation of **ffbconfig** command line if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value.

When the window system is run, if an FFB option has never been specified via **ffbconfig**, a default value is used. The option defaults are as follows:

Option	Default
–dev	/dev/fbs/ffb0
–file	machine
–res	none
–deflinear	false
–defoverlay	false
–linearorder	last
–overlayorder	last
–expvis	disabled
–sov	disabled
–maxwids	32

The default for the **–res** option of none means that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

Note – This provides compatibility for users who are used to specifying the device resolution through the PROM. On some devices (e.g. GX) this is the only way of specifying the video mode. This means that the PROM ultimately determines the default FFB video mode.

EXAMPLES

The following example switches the monitor type to the resolution of 1280 × 1024 at 76 Hz:

example% /usr/sbin/ffbconfig –res 1280x1024x76

FILES **/dev/fbs/ffb0** device special file

SEE ALSO **mmap**(2), **fbio**(7I), **ffb**(7D)

NAME

lad – List Array Devices

SYNOPSIS

lad

DESCRIPTION

lad is a program that will list the names of all array devices connected to the system on stdout. Below is a sample illustrating the format of the **lad** output:

```
c0t1d0s0  1T000585693444821  LUNS:  0 2 4
c1t1d0s0  1T003456821988359  LUNS:  1 3 5
c0t3d0s0  1T043008976652398  LUNS:  0 3
c1t5d0s0  1T003984612112092  LUNS:  1 2 4 5 6 7
```

where the first field is the "system name" of a particular disk array controller and the second field is an "internal name" that uniquely identifies the controller. Following these two fields is the word "LUNS" followed by a list of logical units currently owned by the controller.

EXIT STATUS

Always zero.

SEE ALSO

arraymon(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

m64config – configure the M64 Graphics Accelerator

SYNOPSIS

/usr/sbin/m64config [**–dev** *device-filename*]
[**–res** *video-mode* [**now | try**] [**noconfirm | nocheck**]]
[**–file machine | system**] [**–propt**] [**–prconf**] [**–defaults**]

/usr/sbin/m64config [**–propt**] [**–prconf**]

/usr/sbin/m64config [**–help**] [**–res** ?]

DESCRIPTION

m64config configures the M64 Graphics Accelerator and some of the X11 window system defaults for M64.

The first form of **m64config** shown in the synopsis above stores the specified options in the OWconfig file. These options will be used to initialize the M64 device the next time the window system is run on that device. Updating options in the OWconfig file provides persistence of these options across window system sessions and system reboots.

The second and third forms which invoke only the **–prconf**, **–propt**, **–help**, and **–res** ? options do not update the OWconfig file. Additionally, for the third form all other options are ignored.

Options may be specified for only one M64 device at a time. Specifying options for multiple M64 devices requires multiple invocations of **m64config**.

Only M64-specific options can be specified through **m64config**. The normal window system options for specifying default depth, default visual class and so forth are still specified as device modifiers on the openwin command line (see **Xsun**(1)).

The user can also specify the OWconfig file that is to be updated. By default, the machine-specific file in the /etc/openwin directory tree is updated. The **–file** option can be used to specify an alternate file to use. For example, the system-global OWconfig file in the /usr/openwin directory tree can be updated instead.

Both of these standard OWconfig files can only be written by root. Consequently, the **m64config** program, which is owned by the root user, always runs with setuid root permission.

OPTIONS

–dev *device-filename*
Specifies the M64 special file. The default is **/dev/fbs/m640**.

–file machine | system
Specifies which OWconfig file to update. If machine, the machine-specific OWconfig file in the /etc/openwin directory tree is used. If **system**, the global OWconfig file in the /usr/openwin directory tree is used. If the file does not exist, it is created.

–res *video-mode* **[now | try [noconfirm | nocheck]]**
Specifies the video mode used to drive the monitor connected to the specified M64 device.

The format of these built-in video modes is:

widthx**height**x**rate**

where **width** is the screen width in pixels, **height** is the screen height in pixels, and **rate** is the vertical frequency of the screen refresh. The **i** suffix of 640x480x60i and 768x575x50i designates interlaced video timing. If absent, non-interlaced timing will be used. As a convenience, **–res** also accepts formats with @ in front of the refresh rate instead of x. For example: 1280x1024@76. The list of valid video-modes is shown below. This list can also be obtained by running **m64config** with the **–res** ? option (the third form shown in the command synopsis above). Note that not all resolutions are supported by both the video board and by the monitor. **m64config** will not permit you to set a resolution the board does not support, and will request confirmation before setting a resolution the monitor does not support.

720x400x70
720x400x88
640x480x60
640x480x67
640x480x72
640x480x75
800x600x56
800x600x60
800x600x72
800x600x75
832x624x75
1024x768x87
1024x768x60
1024x768x70
1024x768x75
1280x1024x75
1280x1024x76
1152x870x75
1280x1024x60
1152x900x66
1152x900x76
1280x1024x67
960x680x112S
960x680x108S
640x480x60i
768x575x50i
1600x1280x76
1920x1080x72
1280x800x76
1440x900x76
1600x1000x66
1600x1000x76
1920x1200x70

Symbolic names

For convenience, some of the above video modes have symbolic names defined for them. Instead of the form width**x**height**x**rate, one of these names may be supplied **as the argument to –res. The meaning of the symbolic name none is that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.**

Name	Corresponding Video Mode
svga	**1024x768x60**
1152	**1152x900x76**
1280	**1280x1024x76**
ntsc	**640x480x60i**
pal	**768x575x50i**
none	**(see text above)**

The –**res** option also accepts additional, optional arguments immediately following the video mode specification. Any or all of these may be present.

now If present, not only will the video mode be updated in the OWconfig file, but the M64 device will be immediately programmed to display this video mode. (This is useful for changing the video mode before starting the window system).

Note – It is inadvisable to use this suboption with **m64config** while the configured device is being used (e.g. while running the window system); unpredictable results may occur. To run **m64config** with the now suboption, first bring the window system down. If the now suboption is used within a window system session, the video mode will be changed immediately, but the width and height of the affected screen won't change until the window system is exited and reentered again. In addition, the system may not recognize changes in stereo mode. Consequently, this usage is strongly discouraged.

noconfirm

Using the –**res** option, the user could potentially put the system into an usable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this, the default behavior of **m64config** is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. The noconfirm option instructs **m64config** to bypass this confirmation and to program the requested video mode anyway. This option is useful when **m64config** is being run from a shell script.

nocheck If present, the normal error checking based on the monitor sense code (described above) will be suspended. The video mode specified by

the user will be accepted regardless of whether it is appropriate for the currently attached monitor. (This option is useful if a different monitor is to be connected to the M64 device). *Use of this option implies noconfirm well*.

try If present, the specified video mode will be programmed on a trial basis. The user will be asked to confirm the video mode by typing 'y' within 10 seconds. Or the user may terminate the trial before 10 seconds are up by typing any character. Any character other than 'y' or carriage return is considered a no and the previous video mode will be restored and **m64config** will not change the video mode in the OWconfig file (other options specified will still take effect). If a carriage return is typed, the user is prompted for a yes or no answer on whether to keep the new video mode. This option implies the now suboption (see the warning note on the now suboption).

–defaults

Resets all option values to their default values.

–propt Prints the current values of all M64 options in the OWconfig file specified by the **–file** option for the device specified by the **–dev** option. Prints the values of options as they will be in the OWconfig file after the call to **m64config** completes. This is a typical display:

```
--- OpenWindows Configuration for /dev/fbs/m640 ---
OWconfig: machine
Video Mode: not set
```

–prconf

Prints the M64 hardware configuration. This is a typical display:

```
--- Hardware Configuration for /dev/fbs/m640 ---
ASIC: version 0x41004754
DAC: version 0x0
PROM: version 0x0
Card possible resolutions:  640x480x60, 800x600x75,
        1024x768x60,  1024x768x70, 1024x768x75, 1280x1024x75,
        1280x1024x76, 1280x1024x60, 1152x900x66, 1152x900x76,
        1280x1024x67, 960x680x112S, 960x680x108S, 640x480x60i,
        768x575x50i, 1280x800x76, 1440x900x76, 1600x1000x66,
        1600x1000x76, vga, svga, 1152, 1280, stereo, ntsc, pal
Monitor possible resolutions:  720x400x70, 720x400x88,
        640x480x60, 640x480x67, 640x480x72, 640x480x75,
        800x600x56, 800x600x60, 800x600x72, 800x600x75,
        832x624x75, 1024x768x87, 1024x768x60, 1024x768x70,
```

```
                1024x768x75, 1280x1024x75, 1280x1024x76, 1152x900x66,
                1152x900x76, 1280x1024x67, 960x680x112S, vga, svga,
                1152, 1280, stereo
        Current resolution setting: 1280x1024x76
        Current depth: 8
```

–help Prints a list of the **m64config** command line options, along with a brief explanation of each.

DEFAULTS

For a given invocation of **m64config** command line if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value.

When the window system is run, if an M64 option has never been specified via **m64config**, a default value is used. The option defaults are as follows:

Option	Default
–dev	/dev/fbs/m640
–file	machine
–res	none

The default for the **–res** option of none means that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

Note – This provides compatibility for users who are used to specifying the device resolution through the PROM. On some devices (e.g. GX) this is the only way of specifying the video mode. This means that the PROM ultimately determines the default M64 video mode.

EXAMPLES

The following example switches the monitor type to the maximum resolution of 1280 × 1024 at 76 Hz:

```
example% /usr/sbin/m64config –res 1280x1024x76
```

FILES

/dev/fbs/m640 device special file

/usr/openwin/server/etc/OWconfig
System config file

/etc/openwin/server/etc/OWconfig
System config file

SEE ALSO

Xsun(1), **mmap**(2), **fbio**(7I), **m64**(7D)

NAME

nvutil – RAID controller NVSRAM validation and modification utility

SYNOPSIS

nvutil –**v** [–**q**] [–**f**] [–**s**] [*device*]
nvutil –**o** [–**q**] *offset_range1* [, *offset_range2* ...] [*device*]
nvutil –**o** [–**q**] *offset1=value1* [, *offset2=value2* ...] [*device*]

DESCRIPTION

nvutil is a utility for RAID controller NVSRAM management. It will:
(1) validate that controllers are set up in accordance with the requirements of the RAID management software,
(2) "fix" any NVSRAM settings that don't meet the requirements,
(3) display the value of an arbitrary NVSRAM offset, and
(4) modify the value at an arbitrary NVSRAM offset.

The optional *device* argument is the (system)name of a particular RAID controller to which the command applies; if ommitted, the command applies to all attached RAID devices.

COMMAND LINE PARAMETERS

–**f** Fix any NVSRAM settings that are not in accordance with the requirements; always done before any -o options are executed.

–**o** The argument can be in the form *offset_range* or *offset=value*. In the former case, **nvutil** displays all NVSRAM values in the range, which can be a single decimal number or two decimal numbers separated by a hyphen. The display output is one line per offset, per controller, formatted in this manner:
<device> <offset> <value>

Offsets for the same controller are listed together in ascending order, with a newline character separating different controllers.

The second form of the -o argument means that the device's NVSRAM should be modified at the decimal offset to contain the indicated value, also in decimal.

–**q** Operate in "quiet" mode, which generally suppresses output to stdout, without affecting notification to logs, console, etc.

–**s** Send a status change notice if invalid NVSRAM settings are found.

–**v** Validate that the controller(s) have the proper NVSRAM settings; always done before any -o options are executed.

EXIT STATUSES

0 - Error-free termination
1 - Error termination (event log will contain error)

NOTES

Invoking **nvutil** with no arguments or with an unrecognized function key letter will cause usage information to be displayed.

EXAMPLES

1. nvutil -v c0t1d0s0
   Validate NVSRAM settings of controller "c0t1".

2. nvutil -vfq c1t5d0s0
   Validate and fix if necessary NVSRAM settings of controller "c1t5". (operate in "quiet" mode)

3. nvutil -vf
   Validate and fix NVSRAM settings of all attached controllers.

4. nvutil -o 4,6,2,60-63 c2t2d0s0
   Display NVSRAM values at offset 2, 4, 6 and in the range 60-63 of controler "c2t2".

5. nvutil -o 2=3, 4=0, 45=2
   Set NVSRAM bytes at offset 2 to the value 3, at offset 4 to the value 0, and at offset 45 to the value 2 on all controllers.

NAME

parityck – Scan RAID devices for parity inconsistencies

SYNOPSIS

parityck [–**a**] [–**f**] [–**n**] [–**q**] [–**c** *device*]

DESCRIPTION

parityck scans RAID devices (RAID 1,3,5) for parity inconsistencies. All inconsistences are repaired as they are found. This ensures that the data and the parity information are consistent for the RAID device; it does not ensure the correctness of the data.

The device name is a system dependent name that specifies the LUN on which to run the scan. If a device name is specified, the –a option will be ignored.

parityck uses the standard RAID Manager notification script to provide event logging and remote notification. This script will be called once for each LUN on which parity inconsistencies are repaired.

Unless the –q option is used, **parityck** will display the following information:

Name of device being scanned: percentage complete | scan result

The following scan results are possible:

OK

Inconsistent parity repaired blocks *xxx-yyy* (*nn* blocks)

where *xxx* is the first block on which a parity inconsistency was found and *yyy* is the last block. *nn* is a count of the number of inconsistent blocks.

Scan aborted, RAID 0 LUN

Scan aborted, degraded LUN

Scan aborted, controller parity checking enabled

EXIT CODES

0	All LUNs scanned return OK.
1	Inconsistencies are found on any of the scanned LUNs or the LUN scan was aborted.
–1	Device name is not valid.

The following conditions will generate event notification:

Parity inconsistencies
Aborted scan

OPTIONS

–**a**	All. Scan all LUNs on all controllers. This option is ignored if a device name is given.
–**f**	Force. Forces a parity scan on a controller that has onboard parity scanning enabled.
–**n**	No notification. Events are NOT sent to the notification script.
–**q**	Quiet. No output is displayed.
–**c**	Device name. System name of device that the parity check is to be run on.

EXAMPLES

```
parityck -a

Scanning c1t0d0s0: OK
Scanning c1t0d1s0: Scan aborted, degraded LUN
Scanning c1t0d2s0: Inconsistent parity repaired blocks 0-20 (3 blocks)
Scanning c1t0d3s0: 55%
```

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

pmconfig – Configure the power management system

SYNOPSIS

/usr/sbin/pmconfig

/usr/sbin/pmconfig [–**r**]

AVAILABILITY

SUNWpmu

DESCRIPTION

pmconfig enables the current system autoshutdown information to be viewed and / or the power management configuration modified. **pmconfig** reads in the configuration file **power.conf**(4) and issues commands to make this power configuration active. This may involve commands to the power management pseudo driver (**pm**(7)) or a signal to the power daemon (**powerd**(1M)). If no daemon is present and autoshutdown information is present, a daemon will be started.

ERRORS

If the program cannot open either the pseudo driver or the configuration file it prints an error message to standard error. If the program encounters a syntax error in the configuration file, it prints an error message and the line number of the error in the configuration file. It then skips the rest of the information on that line and processes the next line. Any configuration information already processed on the line containing the error is *used.*

All error messages start with "pmconfig (line n): ", and may be followed by:

Can’t find device name :
The first field is not a device name.

Can’t find threshold value :
the field following the device name is not an integer.

Too many threshold values :
More idle times than the device supports were given.

Unrecognizable dependent name :
The dependent field is not a device name.

a standard error message
Returned from the pm driver.

OPTIONS

–**r** reset all power managed devices to unconfigured

FILES

/etc/power.conf
system power management configuration file

SEE ALSO

pm(7), **power.conf**(4), **powerd**(1M)

NAME

powerd – power manager daemon

SYNOPSIS

/usr/lib/power/powerd [–n]

AVAILABILITY

SUNWpmu

DESCRIPTION

This daemon manages two types of system shutdown. The two types of shutdown are automatic shutdown, set on a daily basis, and low power shutdown on systems which supports battery operation. If the system suspend module, **cpr**(7), is present, it will be used to shut the system down, otherwise the **poweroff**(1M) utility will be used. The auto-shutdown information is read from the file **/etc/power.conf** by the daemon. It is reread whenever the daemon receives a hangup signal, SIGHUP.

Automatic shutdown can occur when two conditions are met. The current time is between the start and finish times, and the system has been idle for at least the set time period. System idleness is determined by inactivity on the console keyboard and mouse.

The start and finish times are specified in the file in 24-hour time notation, measured since the start of the day (12:00 am). If the finish time is less than or equal to the start time, the active period of the daemon will span from midnight to the finish time and from the start time to the following midnight. Thus to specify continuous operation, the finish time may be set equal to the start time. Specifying a negative idle time, disables automatic shutdowns from occurring.

Low power shutdown will occur if the system is running from battery and the daemon monitors that the charge in the battery is too low to reliably continue operation.

Immediately prior to system shutdown, the daemon notifies **syslogd**(1M) of the shutdown, which broadcasts the notification.

OPTIONS

–n No broadcast mode. The daemon will shutdown the system silently without notifying **syslogd**(1M).

FILES

/etc/power.conf
used to obtain the current daemon autoshutdown settings

NOTES

The daemon uses shared memory IPC, which may increase the system image size if the shared memory module has not already been loaded.

The daemon ensures that only one daemon is running. If another daemon is running, then the new daemon will exit with an error. If the daemon dies unexpectedly (non-maskable signal) then residual shared memory state will remain. Starting a new daemon will remove this residual state.

SEE ALSO

cpr(7), **pm**(7), **pmconfig**(1M), **power.conf**(4), **poweroff**(1M), **syslogd**(1M)

NAME

prtdiag – print system diagnostic information

SYNOPSIS

/usr/platform/*platform-name***/sbin/prtdiag** [**–v**] [**–l**]

AVAILABILITY

SUNWkvm

DESCRIPTION

prtdiag displays system configuration and diagnostic information.

The diagnostic information lists any failed Field Replaceable Units (FRUs) in the system.

The interface, output, and location in the directory hierarchy for **prtdiag** are uncommitted and subject to change in future releases.

platform-name is the name of the platform implementation and can be found using the **–i** option of **uname**(1).

OPTIONS

The following options are supported:

–v Verbose mode.
Displays the time of the most recent AC Power failure, and the most recent hardware fatal error information, and (if applicable) environmental status.

The hardware fatal error information is useful to repair and manufacturing for detailed diagnostics of FRUs.

–l Log output.
If failures or errors exist in the system, output this information to **syslogd**(1M) only.

EXAMPLES

The example below displays sample output from an Ultra-Enterprise machine.

example% /usr/platform/‘uname –i‘/sbin/prtdiag –v

System Configuration: Sun Microsystems sun4u 8-slot Ultra Enterprise 4000/5000
System clock frequency: 83 MHz
Memory size: 256Mb

CPU Units: Frequency Cache-Size Version

	A: MHz	MB	Impl.	Mask	B:MHz	MB	Impl.	Mask
Board 0:	**167**	**0.5**	**10**	**2.2**	**167**	**0.5**	**10**	**2.2**
Board 2:	**167**	**0.5**	**10**	**2.2**	**167**	**0.5**	**10**	**2.2**

Memory Units: Size, Interleave Factor, Interleave With

	0: MB	Factor:	With:	1: MB	Factor:	With:
Board 0:	**64**	**4-way**	**A**	**64**	**4-way**	**A**
Board 2:	**64**	**4-way**	**A**	**64**	**4-way**	**A**

==========IO Cards==========
Board 1, SBus0:
Sbus clock frequency: 25 MHz
13:
SUNW, soc/SUNW, pln ’501-2069’
Board 1, SBus1:

```
                SBus clock frequency: 25 MHz
                0:
                    cgsix    'SUNW, 501-2325'
                3:
                    SUNW, hme
                    SUNW, fas/sd(block)
Board 3, SBus0:
                SBus clock frequency: 25 MHz
                13:
                    SUNW, soc      '501-2069'
Board 3, SBus1:
                SBus clock frequency: 25 MHz
                3:
                    SUNW, hme
                    SUNW, fas/sd(block)
Board 7, SBus0:
                Bus clock frequency: 25 MHz
                13:
                    SUNW, soc      '501-2069'
Board 7, SBus1:
                SBus clock frequency: 25 MHz
                3:
                    SUNW, hme
                    SUNW, fas/sd(block)

No failures found in System
===========================

No System Faults found
======================

=================Environmental Status=======================
Keyswitch position is in Normal Mode
System Power Status:  Redundant
System LED Status:      GREEN     YELLOW     GREEN
Normal                    ON        OFF     BLINKING
Fans:
-------
Unit      Status
Rack      OK
Key       OK
AC        OK
```

System Temperatures (Celsius):

	Temperature	Trend
Board 0:	**38**	**stable**
Board 1:	**36**	**stable**
Board 2:	**39**	**stable**
Board 3:	**39**	**stable**
Board 7:	**39**	**stable**
Control Board:	**32**	**stable**

Power Supplies:

Supply	Status
0	**OK**
1	**OK**
2	**OK**
3	**OK**
PPS	**OK**
System 3.3v	**OK**
System 5.0v	**OK**
Peripheral 5.0v precharge	**OK**
Peripheral 12v precharge	**OK**
System 3.3v precharge	**OK**
System 5.0v precharge	**OK**
AC Power	**OK**

ASIC Revisions:

	FHC	AC	SBus0	SBus1	PCI0	PCI1	FEPS
Board 0:	**1**	**4**					
Board 1:	**1**	**2**					**21**
Board 2:	**1**	**4**					
Board 3:	**1**	**2**					**21**
Board 6:							
Board 7:	**1**	**2**					**21**

System Board PROM revision

Board 0:	**OBP**	**3.1.0**	**1996/02/12**	**18:57**	**POST**	**2.5.1**	**1996/02/12**	**05:24**
Board 1:	**FCODE**	**1.6.0**	**1996/01/23**	**13:44**	**iPOST**	**1.1.4**	**1996/01/23**	**06:28**
Board 2:	**OBP**	**3.1.0**	**1996/02/12**	**18:57**	**POST**	**2.5.1**	**1996/02/12**	**05:24**
Board 3:	**FCODE**	**1.6.0**	**1996/01/23**	**13:44**	**iPOST**	**1.1.4**	**1996/01/23**	**06:28**
Board 7:	**FCODE**	**1.6.0**	**1996/01/23**	**13:44**	**iPOST**	**1.1.4**	**1996/01/23**	**06:28**

The example below displays sample verbose output from a SPARCcenter 2000.

```
example% /usr/platform/‘uname –i‘/sbin/prtdiag -v

System Configuration:  Sun Microsystems  sun4d SPARCcenter 2000
System clock frequency: 40 MHz
Memory size: 448Mb
Number of XDBuses: 2
          CPU Units: Frequency Cache-Size      Memory Units: Group Size
          A: MHz   MB    B: MHz  MB      0: MB  1: MB  2: MB  3: MB
Board1:     40     1.0     40    1.0       32     0      32     0
Board4:     40     1.0     40    1.0       32     32     32     32
Board7:     40     1.0     40    1.0       32     32     32     32
Board8:                                    32     32     32     32
======================SBus Cards============================
Board1:     0:   dma/esp(scsi)          ’SUNW,500-1902’
                 lebuffer/le(network)   ’SUNW,500-1902’
            1:   dma/esp(scsi)          ’SUNW,500-1902’
                 lebuffer/le(network)   ’SUNW,500-1902’
            2:   cgsix                  ’SUNW,501-1672’
            3:   dma/esp(scsi)          ’500-1869-01’
                 lebuffer/le(network)   ’500-1869-01’
Board4:     0:   <empty>
            1:   dma/esp(scsi)          ’500-1902-01’
                 lebuffer/le(network)   ’500-1902-01’
            2:   bf                     ’SUNW,501-1732’
            3:   bf                     ’SUNW,501-1732’
Board7:     0:   <empty>
            1:   <empty>
            2:   <empty>
            3:   <empty>
Board8:     0:   <empty>
            1:   <empty>
            2:   <empty>
            3:   <empty>

Failed Field Replaceable Units (FRU) in System:
==============================================

SBus Card unavailable on System Board #7
Failed Field Replaceable Unit is SBus card 0

cpu-unit unavailable on System Board #8
Failed Field Replaceable Unit is SuperSPARC Module A

Most recent AC Power Failure:
=============================
Sat May 22 14:21:18 1993
```

```
Analysis of most recent System Watchdog:
========================================
Log Date: Thu Feb 18 22:28:15 1993

Analysis for Board 7
----------------------------
MXCC
  Asynchronous Error
  Error Valid, CCOP=130 ERR= 2 PA=9.10081000
BW0 (CPU B)
  Client Device Error, Internal Error(s) = IOWSCE
```

EXIT STATUS

0 No failures or errors are detected in the system.

1 Failures or errors are detected in the system.

SEE ALSO

uname(1), **modinfo**(1M), **prtconf**(1M), **psrinfo**(1M), **sysdef**(1M), **syslogd**(1M), **openprom**(7D)

NAME

raidutil – RAID Manager Configuration Utility

SYNOPSIS

raidutil -c device [–s LUN size] [–D delete LUN]
[–l RAID level] [–z segment size]
[–r reconstruction rate]
[–n LUN number to create] [–g drive group]
[–q suppress print output during execution]
[–x segment 0 size]
[–p cache read ahead enable]
[–h create hot spare] [–H delete hot spare]
[–i display Inquiry information]
[–w cache enable] [–m cache mirroring enable]
[–v volatile caching allowed]
[–V view Page 08 (caching page)]
[–f multiplication factor]
[–t Disable pre-fetch transfer length]
[–k minimum pre-fetch]
[–b maximum pre-fetch ceiling]

DESCRIPTION

Use **raidutil** to perform initial configuration of the disk array when installing a disk array on a system. You can also use it to reconfigure the disk array whenever reconfiguration is required (e.g., addition of more disk drives, redefinition of LUNs, RAID levels, etc.).

OPTIONS

–c Specify the device name, the disk array controller to configure (eg: –c c1t0d0s0, -c drive1, etc).

–D Delete LUN
With no number (–D all) = delete all LUNs.
With number (–D 1,2) = delete specific LUNs.

–H Delete hot spare
With no number (–H all) = delete all hot spares.
With number (–H 50,51) = delete specific hot spares where the first number is the channel and the second number is the drive side SCSI ID of each drive. Use commas to separate drives.

–g Specify drive group (–g 11,21,31,41,51) where the first number is the channel and the second number is the drive side SCSI ID of each drive. Use commas to separate drives.

–h Create hot spare (–h 50,51) where the first number is the channel and the second number is the drive side SCSI ID of each drive. Use commas to separate drives.

–i Display inquiry information including the software and firware revision level and date.

–l Specify RAID level.

–m Cache mirror option
–m on 0,1 -m off 2,3 would mean cache mirroring is enabled for LUNs 0 and 1, and is disabled for LUNs 2 and 3 (CME – page 2e).

–n Specify the number of the LUN to be created.

–p Cache read ahead
–p enable 0,1 -p disable 2,3 would mean that cache read ahead is enabled for LUNs 0 and 1, and is disabled for LUNs 2 and 3 (RCD – page 08). Also see associated parameters -f, -t, -k and -b.

–q Suppress printing messages during raidutil execution. This is useful when running raidutil from a shell script. For this option to fully suppress all output, it must be the first parameter on the command line.

–r Specify reconstruction rate (combines delay interval and blocks per delay interval).

–r slow = slow
–r medslow = medium slow
–r med = medium
–r medfast = medium fast
–r fast = fast

–s Specify LUN size to be created in megabytes.
(eg: -s 1000 would create a 1 gigabyte LUN)

–v Volatile cache option
–v on 0,1 –v off 2,3 means volatile caching is enabled for LUNs 0 and 1 and is disabled for LUNs 2 and 3 (CWOB – page 2e).

–w Write cache option
–w on 0,1 –w off 2,3 means write cache is enabled for LUNs 0 and 1 and is disabled for LUNs 2 and 3
(WCE – page 08).

–x Specify segment 0 size in blocks.

–z Specify segment size in blocks.

–f Specify the multiplication factor (0 or 1) followed by the LUN numbers separated by commas.

–t Disable Pre-fetch Transfer Length value (0 to 0xffff) followed by the LUN numbers separated by commas.

–k Minimum Pre-fetch value (0 to 0xffff) followed by the LUN numbers separated by commas.

–b Maximum Pre-fetch Ceiling value (0 to 0xffff) followed by the LUN numbers separated by commas.

–V Print the values in the Caching mode page (08). –V 0 would display page 08 of LUN 0 for the device given by the –c parameter.

EXIT CODES

0 – Error-free termination
1 – Error termination (event log will contain error)

NOTES

In some cases, **raidutil** will require exclusive ownership of the hardware modules affected by the operations being performed, meaning, for example, that a logical unit of the array could not be accessed as a filesystem at the same time that its configuration was being fundamentally altered. **raidutil** does try to be as liberal as possible in terms of permitting those configuration actions that can be performed safely without exclusive ownership. If the program cannot acquire exclusive ownership when needed, it displays an appropriate notice and terminates.

Invoking **raidutil** with no arguments or with an unrecognized function key letter will cause usage information to be displayed.

EXAMPLE

The following command line entry

raidutil -c c1t3d0s0 -D all -n 0 -l 5 -s 1000 -g 11,21,31,41,51

would delete all existing LUN’s, then would create a 5 drive, RAID 5, 1 GB, LUN number 0 on the device c1t3d0s0.

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

rdacutil – redundant disk array controller management utility

SYNOPSIS

rdacutil –i <raid *module specifier*>
rdacutil –m <mode> [–b] <raid *module specifier*>
rdacutil –f <raid *controller specifier*>
rdacutil –u <raid *controller specifier*>
rdacutil –U <raid *controller specifier*>

DESCRIPTION

rdacutil is a very basic redundant controller management utility having the ability to retrieve information about the redundant controller configuration, set controller modes, balance LUNs between controllers, and fail a particular controller in a module.

OPTIONS

–b Valid only in conjunction with "-m 2"; instructs **rdacutil** to establish a "nominal" LUN load balancing.

–f Fail the indicated controller; the operand <raid *controller specifier*> may be either an RM6 module-relative controller name(e.g., "RAID Module 01.a") or an operating system device name(e.g., "c0t5d0s0") referring to a controller or one of the LUNs currently attached to the controller.

–i Display information about the redundant controller configuration for a RAID module, including controller mode (non-redundant, active / passive, active / failed, or dual-active) and LUN distribution. The operand <raid *module specifier*> may be either an RM6 module name(e.g., "RAID Module 01") or an operating system device name(e.g., "c0t5d2s0") referring to a controller or logical unit within the module.

Examples ilustration the format of the displayed infomation:

RAID Module 01: dual-active

active controller a (c1t5d0s0) units: 0 1 2 3

passive controller b (c2t5d0s0) units: none

RAID Module 02: single-controller

active controller a (c3t4d0s0) units: 0 1 2 3 4 5

RAID Module 03: active / passive

active controller a (c2t2d0s0) units: 0 1 2 3

failed controller b (c1t1d0s0) units: none

–m Set the mode of the controllers in the RAID module, where <mode> is 1 for active / passive and 2 for dual-active. The operand <raid *module specifier*> is the same as for the "-i" option. On a transition from mode 2 to mode 1, the first controller (controller "a") will become active. If the RAID module is already in mode 1 when "-m 1" is specified, the controller roles will be swapped.

–u Restore (unfail) a failed controller in a module; **rdacutil** unfails whichever controller is failed. The operand <raid *controller specifier*> is the same as for the "-f" option.

–U Restore (unfail) a failed controller in a module unconditionally; **rdacutil** unfails the alternate to the controller that the command is issued to. The operand <raid *controller specifier*> is the same as for the "-f" option. This option should be used when the RAID Module is connected to different hosts.

EXIT CODES

0 – Error-free termination
1 – Error termination (event log will contain error)

NOTES

When using the -u or -U options and you are using the cXtXdXs0 device argument form, the device you specify must be the operational controller in the module.

Invoking **rdacutil** with no arguments or with an unrecognized function key letter will cause usage information to be displayed.

You must enclose the entire *module specifier/controller specifier* in double quotation marks, if it contains spaces.

EXAMPLES

1. rdacutil -m 2 "RAID Module 01"
   Put "RAID Module 01" in mode 2.

2. rdacutil -m 2 -b "RAID Module 01"
   Put "RAID Module 01" in mode 2 and balance LUNs.

3. rdacutil -f "RAID Module 01.a"
   Fail controller "a"(the first controller) in "RAID Module 01".

4. rdacutil -u c1t5d0s0
   Restore the failed controller which is the alternate controller of c1t5d0s0.

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

rdaemon – Redundant Controller Resolution Daemon

SYNOPSIS

rdaemon *system-dependent_arg block_ major_number character_major_number*

DESCRIPTION

rdaemon is the redundant disk array controller (RDAC) resolution daemon. **rdaemon** runs continuously from system boot until system shutdown, and has the responsibility for receiving and reacting to RDAC-related events, the most significant of which are I/O path failures. In the event of an error, it is up to the daemon to determine and direct the appropriate course of action, which may include simply calling for an RDAC driver retry on the alternate path, or, in case of a hard path or controller failures, transferring all logical units over to the surviving controller, and resuming operations on the new path.

OPTIONS

system-dependent_arg	Not currently interpreted by the daemon, but must be present.
block_major_number	The block major number of the RDAC driver.
character_major_number	The character major number of the RDAC driver.

EXIT CODES

None – **rdaemon** runs as a daemon process and never exits.

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

rm6 – RAID Manager Overview

OVERVIEW

RAID Manager is a software package designed to facilitate disk array management and administration. The primary interface to these functions is through a set of graphical applications. In addition, certain tasks that are important to array management are carried out automatically by "daemon" processes that run from system startup to system shutdown. Also, some functions are available in the form of command line utilities that make it possible to embed customized and / or iterative operations in shell scripts.

COMPONENTS

Graphical User Interface

This release has four graphical applications which can be initiated from icons under the application "launcher". The applications are:

Configuration – This application is primarily for specifying how physical drives in the array are to be utilized in terms of allocation to logical units for data storage or "hot spare" assignment; various logical unit attributes may also be specified as part of LUN creation.

Status – This application permits an administrator to determine if an array has, or has had in the past, any abnormal or unusual status condition associated with it. Three kinds of status information are available under this application: (1) message log viewing, which permits browsing and detailed viewing of accumulated history information pertaining to array exception conditions; (2) "on-demand" health checking, which examines selected arrays for any fault conditions that need to be remedied; and (3) reconstruction status, which permits viewing of reconstruction progress for logical units that have had failed drives replaced.

Recovery Guru – This application assists an administrator in the process of carrying out recovery operations on degraded hardware. The recovery guru "knows" about certain failure modes and attempts to lead the user through the necessary recovery steps, ensuring that the user goes about replacing components in the right manner and taking combinations of multiple failed components into account, if necessary.

Maintenance and Tuning – This application provides control of certain array management tasks that arise from time to time in a RAID configuration, including controller software download, array parity validation, and controller cache tuning.

Command Line Programs

This release has several command Line utilities, each fully described in a separate man page:

lad (list array devices) – This program identifies what array controller and logical units are connected to the system.

nvutil (NVSRAM display / modification utility) – This program permits the viewing and changing of array controller non-volatile RAM settings,allowing for a level of "after-market" customization of controller behavior.

parityck (parity check / repair) – This program checks, and if necessary, repairs the parity information stored on the array. (While correct parity is vital to the operation of the array, the possibility of damage to parity is extremely unlikely).

raidutil (RAID configuration utility) – This program is the command line counterpart to the graphical configuration application. It permits RAID logical unit and hot spare creation and deletion to be controlled from a command line or shell script.

rdacutil (redundant disk array controller management utility) – This program permits certain redundant controller operations such as LUN load balancing and controller failover and restoration to be performed from a command line or shell script.

Daemon Programs and Driver Modules

The following daemon programs and driver modules are supplied with RAID Manager in support of certain automated functions for problem notification and error recovery:

arraymon (array monitor daemon) – The primary function of the array monitor daemon is to watch for the occurrence of exception conditions in the array subsystems and provide administrator notification when they happen.

rdaemon (redundant I / O path error resolution daemon) – The primary function of **rdaemon** is to receive and react to redundant controller exception events and to participate in the application-transparent recovery of those events through error analysis and, if necessary, controller failover.

rdriver (redundant I / O path routing driver) – The **rdriver** module works in cooperation with **rdaemon** in handling the transparent recovery of I / O path failures. Its primary responsibilities include routing I / Os down the proper path and communicating with the **rdaemon** about errors and their resolution.

Customizable Elements

RAID Manager has two primary means for end users to customize the program to their particular needs, the **rmparams** file and the **rmscript** file, each described below:

rmparams (RAID Manager parameter file) – **rmparams** is an ASCII file that has a number of parameter settings for such things as the array monitor poll interval, what time to perform the daily array parity check, etc. The RAID Manager programs read this file on startup or at select times during their execution and adapt their behavior accordingly. A subset of the parameters in **rmparams** are changeable under the graphical interface.

rmscript (RAID Manager notification script) – **rmscript** is a shell program that is called by the array daemon and other programs whenever an important event is reported. The file has certain standard actions, including posting the event to the RAID Manager message log, sending e-mail to the super user, and, in some case, sending an SNMP trap; however, **rmscript** may be edited as the user sees fit, so long as care is taken to not disturb any of the standard actions that the user wants to retain.

FILES

/etc/raid	"home" directory for the RAID Manager distribution files
/etc/raid/bin/rm6	shell script to launch the RAID Manager graphical interface
/etc/raid/rmparams	the RAID Manager parameter file
/etc/raid/rmscript	the RAID Manager notification shell script

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

rmscript – Array event notification script

SYNOPSIS

rmscript

DESCRIPTION

rmscript is the script called by the notification routines in the **arraymon** monitor and the **parityck** application. Those executables will create a file that contains an event record (see the RAID Manager Event Record Format), and call this script with the filename as the first argument to this script.

EXAMPLES

There are all sorts of possibilities for notification using this script, all of which consist of decoding the event file and generating a message to be sent to some other application. For example, to send the event to the RAID Manager proprietary log, the following line is used:

```
putplog $1
```

To add parity and AEN events to the system log (**/var/adm/messages**):

```
EVTYPE=‘cut -d\~ -f6 $1‘
(case $EVTYPE in
        #
        #  An AEN event
        #
        00) awk -F\~ ’NF > 0 { printf "AEN event Host=%s Ctrl=%s Dev=%s\n  ASC=%s
        ASCQ=%s FRU=%s LUN=%s LUN Stat=%s\n  Sense=%s", $1, $2, $3,
        substr($7,1,2), substr($7,3,2), $8, $9, $10, $11 }’ < $1 ;;

        #
        #  Mode Page events are ignored for now
        #
        10) ;;
        #
        #  Parity event
        #
        20) awk -F\~ ’NF > 0 { printf "Parity event Host=%s Ctrl=%s Dev=%s\n  Start
        Blk=%s End Blk=%s # Blks=%s LUN=%s ", $1, $2, $3, $7, $8, $9, $10 }’ < $1 ;;
        #
        #  Text events are ignored for now
        #
        90) ;;
esac) | logger -t rm5 -p user.err
```

The preceding example does not deal with mode page and text events. Note that a variety of tools can be used for decoding the event file, including **cut**(1), **awk**(1), and **sed**(1).

One final example, to mail the superuser that some event has occurred:

```
(awk -F\~ 'NF > 0 { printf "rm5 has detected an array event on Controller %s \n  Device
%s at Host %s - Time %s %s\n", $2, $3, $1, $4, $5 }' < $1;
 echo "\nCheck the event viewer for more details" ) | mailx -s "RAID Manager Event"
root
```

NOTES

rmscript should be placed in the directory specified by the System_RmHomeDirectory parameter found in the params file.

If an application is executed from the script without a full path specified, the script expects that application to be in one of the directories specified in the PATH environment variable, or in the directory specified by the System_RmHomeDirectory parameter found in the rmparams file.

It is the responsibility of rmscript to delete the event file once it has processed the file. The last statement in the script should be

```
rm $1
```

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmparams**(4)

NAME

sm_configd – Solstice SyMON configuration reader

SYNOPSIS

/opt/SUNWsymon/sbin/sm_configd [–**D** *debug-value*] [–**T** *file*] [–**i** *interval*]

AVAILABILITY

SUNWsymon

DESCRIPTION

Monitors the physical configuration of a machine and reports on the status of components. For further details, please see the *Solstice SyMON User's Guide.*

OPTIONS

–**D** Set a debug option for AIL.

–**T** Run the configuration from a file; for testing purposes.

–**i** Set the polling interval for the Config Reader.

FILES

cfg_sun4d.so.1

cfg_sun4u.so.1

cfg_sun4uI.so.1

SEE ALSO

symon(1), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

sm_confsymon – configures the agent host and event monitor host machines running Solstice SyMON software

SYNOPSIS

sm_confsymon –**s** *event_host* [–**v**] [–**k** *polling_time*] [–**c** *polling_time*] [–**p**] [–**i** *sampling_time*] [–**U** *user_name*]

sm_confsymon –**e** *server_host* [–**M** *max_events*] [–**i** *sampling_time*] [–**S** *SNMP_hostname*] [–**P** *platform_name*] [–**U** *user_name*]

sm_confsymon –**D**

AVAILABILITY

SUNWsymon

DESCRIPTION

sm_confsymon configures machines that are running Solstice SyMON software as an agent host (the server that is being monitored) and as the event monitor host (the machine that is monitoring the agent host).

This command is run on the respective machines used as agent host and event monitor host.

For further details on the operation of **sm_confsymon** please see the *Solstice SyMON User's Guide.*

OPTIONS

–**s** Configures the server being monitored so it will identify the machine that is being used as the event monitor host. The machine name of monitoring machine is specified as *event_host.*

–**v** Selects verbose mode, in which the system will echo all actions performed and will request permission to continue.

–**k** Sets polling interval time for **sm_krd** to the number of seconds given as *polling_time* (default is 10 seconds).

–**c** Sets polling interval time for **sm_configd** to the number of seconds given as *polling_time* (default is 10 seconds).

–**p** Modifies disk error message level in kernel and in /etc/system to log soft errors for PFA.

–**i** Sets sampling interval time to the number of seconds given as *sampling_time* (default is 10 seconds).

–**U** Sets the user ID used by **sm_logscand** (when included with the –s option) or sets the user ID used by **sm_egd** (when included with the –e option). The user ID is automatically generated when you provide the user name as the value of *user_name.*

–**e** Configures the machine doing the monitoring so it will identify the server that it is monitoring. The machine name of the monitored machine is specified as *server_host.*

–**M** Sets the maximum number of events, given as *max_errors,* before trimming

(default is 1000 events).

–S Causes SNMP traps to be sent to the machine given as *hostname.*

–P Specifies the type of platform that is being monitored. This value, *platform_name*, is the result of running the **uname -i** command on the server being monitored (such as **SUNW,SPARCserver-1000**).
If you do not specify this option, **sm_symonconfig** will prompt you to enter the number of a platform type from a list it displays. Configuration will not continue until you specify the platform type. You can enter the number 0 to exit at this point.

–D Completely removes the currently installed Solstice SyMON configuration.

SEE ALSO

symon(1), **sm_configd**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

sm_control – starts or stops Solstice SyMON software on the server subsystem host or on the event generator machine.

SYNOPSIS

sm_control [**start**] [**stop**]

AVAILABILITY

SUNWsymon

DESCRIPTION

sm_control starts Solstice SyMON software on the server subsystem host machine or the event generator machine without needing to reboot the machine. It also can shut down the program on the machine. In either case, **sm_control** must be run as superuser on that machine.

For further details on the operation of **sm_control** please see the *Solstice SyMON User's Guide.*

OPTIONS

start	Starts Solstice SyMON software on a machine that has been configured as the server being monitored or the machine doing the monitoring.
stop	Shuts down the Solstice SyMON software.

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

sm_egd – Solstice SyMON event generator

SYNOPSIS

/opt/SUNWsymon/sbin/sm_egd [–**i** *interval*] [–**d** *debug-level*]
[–**h** *log-file*] [–**H** *event-history-file*] [–**R** *rules-file*] [–**I** *init-file*]
[–**l** *shared-object* –**f** *shared-function*] [–**r** *export-root*]
[–**D** *AIL-debug-value*] [–**B** *event-directory*] [–**t** *target-machine*]
[–**S**] [**P**] [–**L** *Tcl-directory*] [–**U** *username*]
[–**n** *RPC-number*] [–**V** *run-directory*]

AVAILABILITY

SUNWsymon

DESCRIPTION

Monitors other **symon** agents and reports events based on Tcl rules defined in rules files.

OPTIONS

–**i** Specify the polling interval (in seconds) when data is collected and rules are run.

–**d** Specify a debug flag for the event generator. The following numbers can be added together to specify several debug options:
1=Provides debugging on the initialization.
2=Provides some basic Tcl debugging.
4=Provides debuggin information on basic calls to rules and AIL.
8=Provides data on the rules as understood by the event generator.
16=Provides debugging on AIL callbacks.
32=Provides debugging on building match lists for MULTI rules.
64=Provides debugging on agent births and deaths.

–**h** Specify the location of the event generator logfile.

–**H** Specify a file used by the event generator to track event numbers.

–**R** Specify a rules file. This file must contain the Rules variable in Tcl.

–**I** Specify a file to initialize Tcl procedures.

–**l** Specify a shared object to be loaded. This option must be used in conjunction with the –**f** option.

–**f** Specifies the function within a shared object that will be called when this object is loaded. This option must be used in conjunction with the –**l** option.

–**r** Specifies the name of the root for the outgoing hierarchy..

–**D** Specifies an AIL debugging flag. The following numbers can be added together to specify several AIL debug options:
1=Print AIP version.
2=List of hierarchy updates.
4=Trace requests and connections.
8=Tell if replacing an existing node.
16=Debug pruning.
32=Trace memory use.
64=Report **sm_symond** traffic.

128=Sleep 30 seconds before starting.
256=Fake server death if **/tmp/dead** exists.
512=Print out strings used.
1024=Print messages showing time for AIP transactions.

–B Specifies the directory for storing the event database.

–t Specifies the target machine to be polled.

–S Specifies that core dumps are allowed.

–P Specifes that process data should be polled.

–L Specifies the location of a Tcl library.

–U Specifies a user name under which to run the event generator program.

Specifies an RPC number for connecting to symond.

–V Specifies a directory for running the event generator. (This can override the location set by the **–t** option. However, the **–h, –H, or –B** flag can override the location specified in the **–V** flag.)

FILES

rules.tcl	Specifies the rules, in Tcl, for the event generator. Located in **/etc/opt/SUNWsymon.**
event_gen.tcl	The initialization file for the event generator. Located in **/etc/opt/SUNWsymon.**
event_log	The log file for events. Located in **/var/opt/SUNWsymon/***target***.**
EG_events	Stores the last event number. Located in **/var/opt/SUNWsymon/***target***.**
events/∗	Each event in the all events hierarchy. Located in **/var/opt/SUNWsymon/***target***.**

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

sm_krd – Solstice SyMON kernel reader

SYNOPSIS

/opt/SUNWsymon/sbin/sm_krd [–**d**] [–**D** *AIL-debug-flag*] [–**v**] [–**t**] [–**r**] [–**R**] [–**U** *kernel-file*] [–**M** *kmem-file*] [–**S** *swap-file*] [–**i** *interval*] [–**P** *count*] [–**T**] [*count*]

AVAILABILITY

SUNWsymon

DESCRIPTION

sm_krd monitors the kernel on an active machine, and reports data to clients. For more information, please see the *Solstice SyMON User's Guide.*

OPTIONS

–**d**	Activate Kernel Reader debugging.
–**D**	Specify an AIL debugging level (values can be added together for combinations of debug output): 1=print AIP version 2=list of hierarchy updates 4=trace requests and connections 8=tell if replacing an existing node 10=debug pruning 20=trace memory use 40=report **sm_symond** traffic 80=sleep 30 seconds before starting 100=fake server death if **/tmp/dead** exists
–**v**	Run the kernel reader in verbose mode.
–**t**	Set the timer flag.
–**r**	Set the resource information flag.
–**R**	Set the resource information summary flag.
–**U**	Specify the name of the kernel file.
–**M**	Specify the name for the kmem file.
–**S**	Specify the name of the swap file.
–**i**	Specify the polling interval.
–**P**	Run for the specified number of intervals, then quit.
–**T**	Build the tree for debugging.
count	Automatically report data for every *count* intervals.

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

sm_logscand – Solstice SyMON log file scanner

SYNOPSIS

/opt/SUNWsymon/sbin/sm_logscand [–**i** *interval*] [–**L** *TCL-library*] [–**U** *user-name*] *log-definition-file*

AVAILABILITY

SUNWsymon

DESCRIPTION

Scans the log files, as described in the log definition file.

OPTIONS

–**i** Set the polling interval to update log files.

–**L** Specify the location of the Tcl library.

–**U** Specify a user name for running the program.

FILES

log-definition-file Initialization file for the log scanner. Located in **/etc/opt/SUNWsymon.**

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

sm_symond – Solstice SyMON process controller

SYNOPSIS

/opt/SUNWsymon/sbin/sm_symond [–**n** *RPC-number*]
[–**d** *debug-level*] [–**D** *AIL-debug-level*] [–**p** *output-level*]
[–**P** *minutes*] [–**i** *intervals*] [–**A** *file*] [–**C** *file*]
[–**E** *directory*] [–**H** *directory*] [–**I** *directory*] [–**L** *file*]

AVAILABILITY

SUNWsymon

DESCRIPTION

sm_symond is a tool to manage Solstice SyMON processes. Its primary role is to start the program's agents, monitor those agents for crashes, and provide RPC information to clients that wish to access any of those agents.

The primary repository for agent data is the file **/etc/opt/SUNWsymon/sm_symond.conf** (see **sm_symond.conf**(4)).

When **sm_symond** is run, it first reads **/etc/opt/SUNWsymon/sm_symond.conf** to determine the local agents to be spawned. It then spawns those agents. If an entry indicates that an agent may exist on a remote system, sm_symond will poll that system looking for another symond to get information on that agent.

Symond serves a hierarchy of information via RPC to any requesting client. Each agent should produce a hierarchy that is readable.

sm_symond is also responsible for looking at the **auth_checker.tcl** and **auth_list.tcl** scripts to determine if a Solstice SyMON user has access to the symon data.

OPTIONS

–**n** Specify a custom RPC number for this program (the default is 100244). If you use this option to specify a different number for the monitored host, you must also supply it to any client programs, such as **symon** or **sm_egd.** This option does not dissociate process and child agents.

–**d** Debugging level for **sm_symond.** These values can be added together for combinations of debug output:
1=trace
2=callbacks
4=rpc
8=spawn info
16=debug access control
32=config file info

–**D** Debugging level for AIL for hierarchy transport.

–**p** Print hierarchy level:
1=nodes
5=nodes and prop
10=nodes, prop, and data

–**P** Turn on profiling to dump after specified number of minutes.

–**i** Sampling interval for checking if the agents are still alive.

–A Specifies alternative authorization checking file (default is **auth_checker.tcl**).

–C Specifies alternative configuration file (default is **sm_symond.conf**).

–E Specifies an alternative ‘‘etc’’ directory (default is **/etc/opt/SUNWsymon**).

–H Specifies an alternative ‘‘home’’ directory (default is **/var/opt/SUNWsymon**). **sm_symond** will run from inside a subdirectory called *hostname* under this directory. Any core file or debug file that is generated will reside there.

–I Specifies an alternative install directory (default is **/opt/SUNWsymon**). This contains a subdirectory called **etc** containing authorization files that are used if no authorization files are found in the directory specified by the **–E** option. This also contains a subdirectory called **lib/tcl** that contains the Tcl library.

–L Specifies an alternative authorization list file (default is **auth_list.tcl**).

FILES

/etc/opt/SUNWsymon/sm_symond.conf
list of agents for invocation.

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NOTES

sm_symond can only be run by root.

NAME

speckeysd – Detects special keys on Type 5 or Compact 1 keyboard

SYNOPSIS

/usr/openwin/bin/speckeysd

AVAILABILITY

SUNWpmow

DESCRIPTION

speckeysd(1M) is a daemon that is started at OpenWindows start time to pick up the Sun Special Key strokes from Type 5 and Compact 1 keyboards. The Sun Special Keys are the following:

Power Key
Shift-Power Key
RaiseVolume Key
RaiseBrightness Key
LowerVolume Key
LowerBrightness Key
Mute Key
Degauss Key

The daemon waits on the Sun Special Key strokes, which are sent to it by the X Windows server as XEvents. On receiving the keystrokes, the daemon will then fork off a service to handle the key.

If the Sun Special Key has been specified as a repeatable key, then a pipe is opened to the service's STDIN. Every subsequent keystroke that is received within a timeout is sent to the service through the pipe as the character 'k'.

The daemon reads speckeysd.map(4), a keys-to-service map file, to determine which of the Sun Special Keys to expect and what service to spawn off to handle the key stroke.

FILES

/usr/openwin/lib/speckeysd.map	keys-to-service map file
/tmp/speckeysd.lock	lock-file generated by the daemon

SEE ALSO

speckesyd.map(4)

NAME

ssp-config – set initial SSP configuration information on the host

DESCRIPTION

Caution: Never execute this command manually.

/usr/platform/sbin/bin/ssp-config is normally invoked by the **/etc/init.d/sspdefs** startup script during boot of the Enterprise 10000 host, but only if the file **./SSP_DEFAULTS** exists. **ssp-config** interactively prompts for information, including the SSP's hostname and IP address. It uses the information to set the initial configuration to allow communication between the server and the SSP.

Only super user can run **ssp-config**.

FILES

/.SSP_DEFAULTS
/etc/inet/hosts
/etc/ssphostname
/etc/syslog.conf

SEE ALSO

ssp-unconfig(1M)

NAME

ssp-unconfig – undo SSP and system information on the host

DESCRIPTION

Caution: Only super user can use this command. Exercise extreme caution in its use.

When executed on an Enterprise 10000 server, **/usr/platform/sun4u1/sbin/ssp-unconfig** removes configuration information established by the command **ssp-config**(1M), then invokes the SunOS command **sys-unconfig**(1M) to make the system ready to be configured again.

The **ssp-unconfig** command does the following:

- Removes SSP information from the **/etc/syslog.conf** and **/etc/inet/hosts** files.
- Removes the **/etc/ssphostname** file.

When finished, **ssp-unconfig**, invokes the SunOS command **sys-unconfig**(1M), which performs a system shutdown.

FILES

/.SSP_DEFAULTS
/etc/inet/hosts
/etc/ssphostname
/etc/syslog.conf

SEE ALSO

ssp-config(1M) in this reference manual

sys-unconfig(1M) in *man Pages(1M): System Administration Commands*

NAME

sunvts – Invokes the SunVTS kernel and its user interface

SYNOPSIS

sunvts [–**lepqstv**] [–**o** *option_file*] [–**f** *log_dir*] [–**h** *hostname*]

AVAILABILITY

SUNWvts

DESCRIPTION

The **sunvts** command is used to invoke the SunVTS user interface and kernel on the same system. It could be used to start the user interface on the local system and connect to the SunVTS kernel on the remote system. By default, it displays CDE Motif graphic interface for CDE environment, OpenLook graphic interface for OpenWindows environment, or TTY interface for non-windowing system.

OPTIONS

–**l** Displays SunVTS OpenLook graphic interface.

–**e** Disables the security checking feature.

–**f** *log_dir*
Specifies an alternative log_file directory. The default log_file directory is **/var/opt/SUNWvts/logs.**

–**h** *hostname*
Starts the SunVTS user interface on the local system, which connects to or invokes the SunVTS kernel on the specified host after security checking succeeds.

–**o** *option_file*
Starts the SunVTS kernel with the test options loaded from the specified *option_file*, which by default is located in **/var/opt/SUNWvts/options.**

–**p** Starts the SunVTS kernel **vtsk (1M)** such that it does not probe the test system's devices.

–**q** Automatically quits both the SunVTS kernel and the user interface when testing stops.

–**s** Automatically starts testing from a selected group of tests. The flag must be used with the –**o** *option_file* flag.

–**t** Starts **vtstty (1M),** a TTY based interface, instead of CDE or OpenLook interface.

–**v** Displays version information from **vtsui**(1M) and **vtsk**(1M).

NOTES

If **vtsk (1M)** is already running on the test system, the **sunvts** command ignores the –**e,** –**o,** –**f,** –**q,** –**p,** and –**s** options.

SEE ALSO

vtsk(1M), **vtstty**(1M), **vtsui**(1M), **vtsui.ol**(1M), **vtsprobe**(1M)

NAME

sys-suspend – Suspend the system and power off

SYNOPSIS

/usr/openwin/bin/sys-suspend [–**fnx**]

AVAILABILITY

SUNWpmow

DESCRIPTION

sys-suspend(1M) invokes the **uadmin**(1M) system call with the right options to suspend the whole system. A system can be suspended to conserve power or to prepare the system for transport. It should not be used in place of a shutdown when performing any hardware reconfiguration or replacement.

The current system state will be preserved until a resume operation is performed (the next power on).

On a resume from a manually initiated suspend in the windows environment, the system brings up **xlock**(1) to make certain that only the same person who suspended the system can have access to the system. In a non-windows environment, the user will be prompted for password. If the suspend was initiated by the **powerd**(1M), a. k. a. **AutoShutdown,** mechanism, no additional security measure is initated. It is the user's responsibility to secure his / her work session before **AutoShutdown** takes place.

It is possible that when devices or processes are performing critical or time sensitive operations (such as real time operations) the system may fail to suspend. When this occurs, the system will remain in its current running state. Messages reporting the failure will be displayed on the console. Once the system is successfully suspended the resume operation will always succeed barring external influences such as hardware reconfiguration or the like.

OPTIONS

–**f** Force suspend. This should be used with care. Using this option causes the system to force stops all processes that does not through the default mechnism. This option should be used only during unattended operations.

–**n** Disable confirmation. This flag disables the confirmation popup dialog at suspend time.

–**x** Disable lockscreen. This flag disables the execution of lockscreen at resume time.

FILES

/kernel/misc/cpr	loadable module for cpr
/cprboot	special bootstrapper for cpr
/.CPR	system state file
/.cpr_generic_info	sys-suspend control file
/.cpr_defaultboot_info	sys-suspend control file
/etc/default/sys-suspend	file for setting a default value for the PERM environment variable. PERM determines who are allowed to use this command. Allowed values are:
all	everybody can use this command (default)
-	nobody can use this command
<user1, user2, etc.>	a user in this user list can use this command

NOTES

xlock(1) on resume can be disabled by default. The following line needs to be added to the user's **.Xdefaults** or **.OWdefaults** file:

Syssuspend*xlock: False

The xlock mode defaults to *life*. This can be changed by adding the following line to the user's **.Xdefaults** or **.OWdefaults** file:

Syssuspend*xlockmode: <xlockmode>

SEE ALSO

uadmin(2), **cpr**(7)

NAME

verify_rules – Solstice SyMON syntax checker for event rules

SYNOPSIS

/opt/SUNWsymon/sbin/verify_rules [–**I** [*filename*]] [–**R** [*filename*]] [–**o**]

AVAILABILITY

SUNWsymon

DESCRIPTION

Run **verify_rules** to check if an event rules file in Tcl has the correct syntax. This command will respond with ‘‘GOOD RULES’’ if the syntax is correct and ‘‘BAD RULES’’ if the syntax is incorrect.

For more information about using **verify_rules** see the *Solstice SyMON User’s Guide.*

OPTIONS

–**I** Checks the file that contains supporting functions; the default file is **event_gen.tcl**

–**R** Checks the file that contains the EVENTS variable; the default file is **./rules.tcl**

–**o** Provides Tcl output to the standard error output for debugging.

NOTES

Running **verify_rules** only checks the correctness of syntax. It does not guarantee that a rule will work as expected.

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

vtsk – SunVTS diagnostic kernel

SYNOPSIS

vtsk [–**epqsv**] [–**o** *options_file*] [–**f** *logfile_directory*]

AVAILABILITY

SUNWvts

DESCRIPTION

The **vtsk** command starts up the SunVTS diagnostic kernel as a background process. There can only be one copy of **vtsk** running at a time. Only the superuser can execute this command.

Normally, **vtsk** is automatically started up by the **sunvts (1M)** command if it is not already running. **vtsk** will also be invoked by **inetd (1M)** when there is a connection request from vtsui or vtsui.ol. In that case, the security file, **.sunvts_sec,** will be checked for the permission before running vtsk on the target host specified by **vtsui**(1M) or **vtsui.ol**(1M).

OPTIONS

–**e** Enables the security checking for all connection requests.

–**p** Starts SunVTS diagnostic kernel, but does not probe system configuration.

–**q** Quits both the SunVTS diagnostic kernel and the attached User Interfaces when the testing is completed.

–**s** Runs enabled tests immediately after started.

–**v** Display SunVTS diagnostic kernel's version information only.

–**o** *options_file*
Starts the SunVTS diagnostic kernel and sets the test options according to the option file named *options_file.*

–**f** *logfile_directory*
Specifies an alternative logfile directory, other than the default.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

–**1** An error occurred.

FILES

/var/opt/SUNWvts/options default option file directory.
/var/opt/SUNWvts/logs default log file directory.

SEE ALSO

sunvts(1M), **vtsui**(1M), **vtsui.ol**(1M), **vtstty**(1M), **vtsprobe**(1M)

NAME

vtsprobe – prints the device probe information from the SunVTS kernel

SYNOPSIS

vtsprobe [–**m**] [–**h** *hostname*]

AVAILABILITY

SUNWvts

DESCRIPTION

vtsprobe is a utility that displays the device and configuration information contained in the SunVTS kernel. The output includes the SunVTS assigned group for the device, the device name, the device instance, the testname attached to this device, and the configuration information obtained from the device-specific test probe.

OPTIONS

–**m** Specifies manufacturing mode, which displays the probe information in a format that is easy to read using script files.

–**h** *hostname*
Specifies the *hostname* to connect to and get the device and configuration information. If not specified, the current host will be used.

USAGE

After the SunVTS kernel is up and running, you may type **vtsprobe** at the shell prompt to get the probe output. (See the **sunvts (1M)** man page for more information on how to start up SunVTS.

EXAMPLE

Running **vtsprobe** on a sun4m SPARCclassic produces the following output:

```
% vtsprobe

Processor(s)
    system(systest)
        System Configuration=sun4m SPARCclassic
        System clock frequency=50 MHz
        SBUS clock frequency=25 MHz
    fpu(fputest)
        Architecture=sparc
        Type=TI TMS390S10 or TMS390S15 microSPARC chip
Memory
    kmem(vmem)
        Total: 143120KB
    mem(pmem)
        Physical Memory size=24 Mb
SCSI-Devices(esp0)
    c0t2d0(rawtest)
        Capacity: 638.35MB
        Controller: esp0
        Vendor: MICROP
        SUN Id: 1588-15MBSUN0669
        Firmware Rev: SN0C
        Serial Number: 1588-15MB103
```

```
    c0t2d0(fstest)
        Controller: esp0
    c0t3d0(rawtest)
        Capacity: 404.65MB
        Controller: esp0
        Vendor: SEAGATE
        SUN Id: ST1480   SUN0424
        Firmware Rev: 8628
        Serial Number: 00836508
    c0t3d0(fstest)
        Capacity: 404.65MB
        Controller: esp0
        Vendor: SEAGATE
        SUN Id: ST1480   SUN0424
        Firmware Rev: 8628
        Serial Number: 00836508
    c0t3d0(fstest)
        Controller: esp0
    c0t6d0(cdtest)
        Controller: esp0
    tape1(tapetest)
        Drive Type: Exabyte EXB-8500 8mm Helical Scan
Network
    isdn0(isdntest)
        NT Port  TE Port
    le0(nettest)
        Host_Name: ctech84
        Host Address: 129.146.210.84
        Host ID: 8001784b
        Domain Name: scsict.Eng.Sun.COM
Comm.Ports
    zs0(sptest)
        Port a -- zs0  /dev/term/a : /devices/ ... a
        Port b -- zs1  /dev/term/b : /devices/ ... b
Graphics
    cgthree0(fbtest)

OtherDevices
    bpp0(bpptest)
        Logical name: bpp0
    sound0(audio)
        Audio Device Type: AMD79C30
    sound1(audio)
        Audio Device Type: DBRI Speakerbox
    spd0(spdtest)
```

Logical name: spd0

NOTES

The output of **vtsprobe** is highly dependent on the device being correctly configured into the system (so that a SunVTS probe for the device can be run successfully on it) and on the availability of a device-specific test probe.

If the device is improperly configured or if there is no probing function associated with this device, **vtsprobe** cannot print any information associated with it.

SEE ALSO

sunvts(1M), **vtsk**(1M), **vtsui**(1M), **vtsui.ol**(1M), **vtstty**(1M)

NAME

vtstty – TTY interface for SunVTS

SYNOPSIS

vtstty [–**qv**] [–**h** *hostname*]

AVAILABILITY

SUNWvts

DESCRIPTION

vtstty is the default interface for SunVTS in the absence of a windowing environment. It can be used in a non-windowing environment such as a terminal connected to the serial port of the system. However, its use is not restricted to this; **vtstty** can also be used from shelltool and commandtool.

OPTIONS

–**q** The "auto-quit" option automatically quits when the conditions for SunVTS to quit are met.

–**v** Prints the **vtstty** version. The interface is not started when you include this option.

–**h** *hostname*
Connects to the SunVTS kernel running on the host identified by *hostname.*

USAGE

The **vtstty** screen consists of four panels: main control, status, test groups, and console. The panels are used to display choices that the user can select to perform some function and/or to display information. A panel is said to be "in focus" or in a "selected" state when it is surrounded by asterisks and the current item is highlighted. In order to choose from the items in a panel, the focus should be shifted to that panel first.

The following are the different types of selection items that can be present in a panel:

Text string	Describes a choice that, when selected, either pops up another panel or performs a function. For example, "stop" will stop the SunVTS testing.
Data entry field	To enter or edit numeric or textual data.
Checkbox	Represented as "[ ]". Checkboxes are associated with items and indicate whether the associated item is selected or not. A checkbox can be in one of the following two states: Deselected [ ] or Selected [*].

The key assignments given below describe the keys for shifting focus, making a selection, and performing other functions:

TAB or <CTRL>W	Shift focus to another panel
RETURN	Select current item
Spacebar	Toggle checkbox
Up arrow or <CTRL>U	Move up one item
Down arrow or <CTRL>N	Move down one item

Left arrow or <CTRL>P	Move left one item
Right arrow or <CTRL>R	Move right one item
Backspace	Delete text in a data entry field
ESC	Dismiss a pop-up
<CTRL>F	Scroll forward in a scrollable panel
<CTRL>B	Scroll backward in a scrollable panel
<CTRL>X	Quit **vtstty** but leave the SunVTS kernel running
<CTRL>L	Refresh the **vtstty** screen

NOTES

1. To run **vtstty** from a telnet session, carry out the following steps:
   a. Before telnet-ing, determine the values for "rows and "columns". (See **stty**(1)).
   b. Set term to the appropriate type (for example, **set term=sun-cmd**
   c. Set the values of columns and rows to the value noted above.
2. Before running **vtstty** ensure that the environment variable describing the terminal type is set correctly.

SEE ALSO

sunvts(1M), **vtsk**(1M), **vtsui**(1M), **vtsui.ol**(1M), **vtsprobe**(1M)

NAME

vtsui – SunVTS Graphic User Interface (CDE)

SYNOPSIS

vtsui [–**qv**] [–**h** *hostname*]

AVAILABILITY

SUNWvts

DESCRIPTION

The **vtsui** command starts up the CDE Motif version of SunVTS graphic user interface. There can be multiple instances of **vtsui** running at the same time, all connected to one SunVTS diagnostic kernel, **vtsk**(1M). The name of the host machine running the diagnostic kernel, **vtsk**(1M), will be displayed in the title bar of the graphical user interface window.

vtsui is automatically started up by the **sunvts (1M)** command. **vtsui can** be also used to start **vtsk (1M)** if **inetd (1M)** is in operation. In that case, the security file, **sunvts_sec,** will be checked for the permission before running **vtsk** on the target host.

See the "SunVTS User's Guide" for a complete description on using the graphical user interface.

OPTIONS

–**q** Quits the SunVTS graphic user interface when testing has terminated.

–**v** Displays graphic user interface version information only.

–**h** *hostname*
Starts the SunVTS graphic user interface and connects to the SunVTS diagnostic kernel running on *hostname*, or invokes the kernel if not running, after security checking succeeds. If *hostname* not specified, the local host is assumed.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error occurred.

SEE ALSO

sunvts(1M), **vtsk**(1M), **vtsui.ol**(1M), **vtstty**(1M), **vtsprobe**(1M)

NAME

vtsui.ol – SunVTS Graphic User Interface (OpenLook)

SYNOPSIS

vtsui.ol [–**qv**] [–**h** *hostname*]

AVAILABILITY

SUNWvts

DESCRIPTION

The **vtsui.ol** command starts up the OpenLook version of **SunVTS** graphic user interface. There can be multiple instances of **vtsui.ol** running at the same time, all connected to one **SunVTS** diagnostic kernel, **vtsk**(1M). The name of the host machine running the diagnostic kernel, **vtsk**(1M), will be displayed in the title bar of the graphic user interface window.

vtsui.ol can be used to start **vtsk**(1M) if **inetd**(1M) is in operation. In that case, the security file, **.sunvts_sec,** will be checked for the permission before running **vtsk** on the target host. **vtsui.ol** is also automatically started up by the **sunvts**(1M) command.

See the "SunVTS User's Guide" for a complete description on using the graphic user interface.

OPTIONS

–**q** Quits the SunVTS graphic user interface when testing has terminated.

–**v** Displays graphic user interface version information only.

–**h** *hostname*
Starts the SunVTS graphic user interface and connects to the **SunVTS** diagnostic kernel running on *hostname*, or invokes the kernel if not running, after security checking succeeds. If *hostname* not specified, the local host is assumed.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error occurred.

SEE ALSO

sunvts(1M), **vtsk**(1M), **vtsui**(1M), **vtstty**(1M), **vtsprobe**(1M)

NAME

auth_checker.tcl – Parser for handling list of authorized Solstice SyMON users

SYNOPSIS

/opt/SUNWsymon/etc/auth_checker.tcl

DESCRIPTION

This Tcl file parses the list of authorized Solstice SyMON users contained in the **auth_list.tcl**(4) file.

For more information, see the *Solstice SyMON User's Guide*

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

auth_list.tcl – List of authorized Solstice SyMON users

SYNOPSIS

/opt/SUNWsymon/etc/auth_list.tcl

DESCRIPTION

This list identifies the users authorized to use the Solstice SyMON software on a system. Users, hosts, and groups can be defined as authorized, readonly, or unauthorized.

The data in **auth_list.tcl** is parsed by **auth_checker.tcl**(4).

For more information, see the *Solstice SyMON User's Guide*

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

event_gen.tcl – Defines procedures and variables used by rules in the Solstice SyMON program

SYNOPSIS

/opt/SUNWsymon/etc/event_gen.tcl

DESCRIPTION

When you run the **sm_confsymon** –**e** *servername* command, the **event_gen.tcl** file is copied to create a file called **event_gen.***servername***.tcl** that contains information specific to that machine within the Solstice SyMON program.

This information includes the host names of machines that will be sent snmp trap messages.

For more information, see the *Solstice SyMON User's Guide.*

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

logscan.tcl – Defines file that the Solstice SyMON program's Log Viewer will search

SYNOPSIS

/opt/SUNWsymon/etc/logscan.tcl

DESCRIPTION

This Tcl file contains a definition of the **/var/adm/messages** file that will be searched by the Log Viewer of the Solstice SyMON program.

For more information, see the *Solstice SyMON User's Guide*

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **rules.tcl**(4), **sm_symond.conf**(4)

NAME

power.conf – power management configuration information file

SYNOPSIS

/etc/power.conf

AVAILABILITY

SUNWpmr

DESCRIPTION

The **power.conf** file is used by the power management configuration program, **pmconfig**(1M), to initialize the settings for power management of the system.

There are two types of entries in the **power.conf** file, **device management** entries and **system management** entries. These two types of entries are described in the corresponding sections below.

DEVICE MANAGEMENT

Devices not appearing in this file will not be power managed without explicit configuration using the power management pseudo driver (see **pm**(7D)). It is recommended the power management framework be fully understood before modifying device management entries in this file. Although inappropriate settings will not cause system damage, severe performance reduction may result.

Device management entries consist of line by line listings of the devices to be configured. Each line is of the form:

device_name *threshold* . . . *dependents* . . .

Each line must contain a *device_name* field and a *threshold* field; it may also contain a *dependents* field. The fields must be in that order (*device_name*, *threshold*, *dependents*). Fields and sub-fields are separated by white space (tabs or spaces). A line may be more than 80 characters. If a newline character is preceded by a backslash ('\') it will be treated as white space. Comment lines must begin with a hash character ('#').

The *device_name* field specifies the device to be configured. *device_name* is either a pathname specifying the device special file or a "relative" pathname containing the name of the device special file. When using the latter format, instead of using the full pathname, it is possible to omit the portion of the pathname specifying the parent devices. This includes the leading '/'. Using this "relative" pathname format, the first device found with a full pathname containing *device_name* as its tail is matched. In either case, the leading **/devices** component of the pathname does not need to be specified.

For example, a SCSI disk target with the following full path name:

/iommu@f,e000/sbus@f,e001/espdma@f,4000/esp@f,8000/sd@1,0

may also be specified as:

sbus@f,e000/espdma@f,4000/esp@f,8000/sd@1,0

or

esp@f,8000/sd@1,0

or

sd@1,0

The *threshold* field is used to configure the power manageable components of a device. These components represent entities within a device which may be power managed separately. This field may contain as many integer values as the device has components. Each *threshold* time specifies the idle time in seconds before the respective component may be powered down. If there are fewer component *threshold* times than device components, the remaining components are not power managed. To explicitly disable power down for a component use a value of **–1**. At least one component *threshold* must be specified per device (in the file).

The *dependents* field may contain a list of *logical* dependents for this device. A *logical* dependent is a selected device that is not physically connected to the power managed device (e.g. the display and the keyboard). A dependent device is one which must be idle and powered down before the managed device may be powered down. The *dependents* field entries use the same formats allowed in the first field and are separated by white space. A device must previously have been configured before it may be used as a dependent.

SYSTEM MANAGEMENT

The system management entries control power management for the system as a whole. They are distinguished by the use of the special device names below.

Note that the following (**autoshutdown**) entry is not intended to be hand edited, but to be maintained by **dtpower**(1M).

If the *device_name* field contains the special device name "**autoshutdown**", the *threshold* value specifies the *system idle time* (measured as discussed below) before the system may be shut down by **powerd**(1M). The *threshold* value is followed by *start* and *finish* times (each in the format hh:mm) which specify the time period during which the system may be automatically shut down (see **powerd**(1M)). Following the *start* and *finish* times is the *behavior* field, consisting of one of the words **shutdown**, **noshutdown**, **autowakeup**, or **default.**

If the *behavior* field is **shutdown** then the system will be automatically shut down when it has been idle for the number of minutes specified in the *threshold* value and the time of day falls between the *start* and *finish* values.

If the *behavior* field is **noshutdown** then the system is never automatically shut down.

If the *behavior* field is **autowakeup** and the hardware has the capability to do autowakeup, then the system is shut down as if the value were **shutdown** and the system will be restarted automatically the next time that the time of day equals the *finish* time.

If the *behavior* field is **default** then the behavior of the system will depend upon which model it is. Desktop models which were first put into production after October 1, 1995 will behave as if the *behavior* field were set to **shutdown** and desktop models first put into production before this date and server models will act as if the *behavior* field were set to noshutdown. The determination of default behavior is made by looking for the existence of a root node property named energystar-v2.

If the *device_name* field contains the special device name "**ttychars**", the *threshold* field will be interpreted as the maximum number of tty characters which may pass through the ldterm module and the system still be considered to be idle. If no entry is provided this

value defaults to 0.

If the *device_name* field contains the special device name "**loadaverage**", the (floating point) *threshold* field will be interpreted as the maximum load average that may be seen and the system still be considered to be idle. If no entry is provided this value defaults to 0.04.

If the *device_name* field contains the special device name "**diskreads**", the *threshold* field will be interpreted as the maximum number of disk reads which may be done by the system and the system will still be considered to be idle. If no entry is provided this value defaults to 0.

If the *device_name* field contains the special device name "**nfsreqs**", the *threshold* field will be interpreted as the maximum number of NFS requests which may be sent or received by the system and it still be considered to be idle. Null requests, access requests and gettattr requests are excluded from this count. If no entry is provided this value defaults to 0.

The values for tty characters, disk reads and NFS requests are determined by periodic sampling of the kstat interface. The thresholds for these events apply to a period extending into the past for *system idle time* minutes as specified in the "**autoshutdown**" entry described above.

The value for load average is also determined by periodic sampling of the kstat interface. The threshold for this value is an instantaneous one. The system won't be considered idle with respect to load average until *system idle time* minutes have passed with the sampled load average value not exceeding the threshold.

If the *device_name* field contains the special device name "**idlecheck**", the *device_name* field must be followed by the pathname of a program to be executed to determine if the system is idle. If autoshutdown is enabled and the console keyboard, mouse, tty, CPU (as indicated by load average), network (as measured by NFS requests) and disk (as measured by read activity) have been idle for the amount of time specified in the *autoshutdown* entry specified above, and the time of day falls between the *start* and *finish* times, then this program will be executed to check for other idleness criteria. The value of the idle time specified in the above *autoshutdown* entry will be passed to the program in the environment variable PM_IDLETIME. The process must terminate with an exit code which represents the number of minutes that the process considers the system to have been idle.

There is no default idlecheck entry. The default behavior is to consider only mouse, keyboard, tty, load average, NFS requests and disk reads as indicators of non-idleness. To extend the definition of non-idleness a shell script can be created which must exit with the number of minutes it considers the system to have been idle by its criteria. The path to this new script can then be put in the idlecheck entry in **power.conf.**

EXAMPLES

The following is a sample **power.conf** file.

```
# This is a sample power management configuration file
# Fields must be separated by white space.
#
```

```
# Name          Threshold(s)   Logical Dependent(s)
/dev/kbd        1800
/dev/mouse      1800
/dev/fb         0 0            /dev/kbd /dev/mouse

#Example of a second display
/dev/fb1        0 0            /dev/kbd /dev/mouse

# This entry is maintained by dtpower(1M)
# This (default as of SunOS 2.5) entry causes the system  to be shut down
# after 30 minutes of idle time if it is a model first shipped after
# Oct 1, 1995.  Older models default to noshutdown.
#
#                                       autoshutdown in effect
# Auto-Shutdown       Idle(min)         Start/Finish(hh:mm)    Behavior
autoshutdown          30        9:00 9:00                      default

# Idlecheck  program is passed autoshutdown idle time entry in $PM_IDLETIME
# returns number of minutes the system has been idle in exit code
idlecheck    /home/critical/idlecheck
```

The following is a sample idlecheck script.

```
#!/bin/sh
# This is a sample idlecheck script which considers the system not idle
# if user critical is logged in

critical=‘who | grep -w critical‘
if [ "$critical" ]                  # if "$critical" is not null string
then
        exit 0                      # not idle because critical logged in
else
        exit $PM_IDLETIME           # idle long enough
fi
```

SEE ALSO

dtpower(1M), **pmconfig**(1M), **powerd**(1M), **pm**(7D)

Writing Device Drivers

NOTES

The default behavior for desktop models introduced after October 1, 1995 is to shut down after 30 minutes of idleness any time of day. **dtpower**(1M) can be used to change the default.

This behavior being the default as shipped is mandated by the US Government Environmental Protection Agency as a requirement for EnergyStar compliance. The user might want to use **dtpower**(1M) to set the autoshutdown start time to the end of the normal work day and to set the autoshutdown stop time to the start of the normal work day.

Remember that *physical* dependents are automatically included by the power manager and need not be specified.

The default **power.conf** file supports the standard hardware configuration. For each additional power manageable device (e.g. second display), a new entry must be manually added to the **power.conf** file and **pmconfig**(1M) executed to activate the new change.

Powering devices up and down frequently may reduce device reliability, especially for devices not designed for power management. Do not put additional devices under power management unless the hardware documentation permits it. At this time most SCSI hard disks are not power manageable.

NAME

rdriver – configuration file for Redundant Disk Array Controller driver

DESCRIPTION

The Redundant Disk Array Controller(RDAC) driver is a layered driver that is positioned logically above the SCSI disk driver and, in cooperation with certain daemon processes, is responsible for management of redundant paths to the disk array. This driver provides such capabilities as I/O path load balancing and transparent failover in the event of a path failure.

The configuration file for rdriver(rdriver.conf) is used to specify which of the possible paths to an array logical unit (or "LUN") is considered the "home" path for that unit. The significance of the home path is that it typically corresponds to the true physical path of the logical unit; however, in situations where the logical unit has moved to the alternate controller, it still constitutes the path name by which the OS and applications refer to the unit. Thus, in a controller pair (c1t1d0s0, c2t3d0s0), the home path for LUN 5 may be c1t1d5s0, but in exceptional situations, the true physical path of the LUN may be c2t3d5s0. In this case, the LUN would continue to be accessed by its home path name of c1t1d5s0. This behavior assures device name consistency in the face of LUN movement between controllers. (Note that RAID Manager will always indicate the true physical path for a logical unit.)

The rdriver.conf file is generated automatically when RAID Manager is installed; on every subsequent reconfiguration boot (boot -r) and only on reconfiguration boots, the file is regenerated so as to make the /dev device names line up with the true physical location of the LUNs. The automatic generation of rdriver.conf is desirable because it spares the user from the tedious task of understanding and entering the rdriver properties manually for each logical unit. Once a system administrator has settled on stable LUN-to-controller assignments for all of the system's RAID moddules, he will probably want to disable the auto-regeneration of rdriver.conf since any subsequent LUN movements should be short-term (as in just long enough to repair a failed I/O path component). This disabling is accomplished by inserting for example, a line into rdriver.conf containing the string "#!noauto_config" beginning in column one. With this string present, reconfiguration boots no longer affect rdriver.conf, and manual edits can be made without the concern that they will be overwritten.

PROPERTIES

Below are the properties that may be specified in the rdriver.conf file for each logical unit:

name The name of the RDAC driver; must be "rdriver"

module The number of the RAID module to which this LUN belongs (e.g., if the module name is "RAID Module 02", then this property would be specified as "module=2"

lun the number of the logical unit; this propety is just the standard "lun" property used by all drivers in the scsi class.

target the ID of the SCSI target which is the "home" controller for the LUN. Immediately after rdriver.conf auto-generation, the home controller will agree with the true physical location, but, as discussed above, detaching and re-attaching the LUN on the other controller will not cause a change in the LUN path name

- a "boot -r" is necessary for that to occur, and even then the name change does not occur if the string "#!noauto_config" is present in the .conf file.

parent This is just the standard "parent" property that may be specified for any loadable driver. This should not be changed from the value it receives when the rdriver.conf file is auto-generated.

nex This property is identical in setting and meaning to the "parent" property and is present because the parent property is not made available to the driver. (The driver needs this information for binding of newly-created LUNs to the proper path.)

unbound This property is not necessary for LUNs that actually exist. For non-existent LUNs, it should be present and set to 1. "Unbound" (non-existent) LUNs actually have properties for both possible paths to the LUN; then when the LUN is created, its binding is established so that its home path is the same as the true physical path.

USAGE GUIDELINES

Although essential to the correct operation of the RDAC driver, the need for end-user / administrator involvement in the maintenance of rdriver.conf is intended to be minimal. A typical scenario involving creation of LUNs and rdriver.conf setup would be as follows:

1. Create the logical units on your RAID modules as you want them.
2. Distribute the LUNs between controller pairs as you want them.
3. Once the LUNs are created and distributed, shut down the system and perform a reconfiguration boot (boot -r). This will result in a new rdriver.conf file being built in accordance with the current configuration.
4. Inspect the rdriver.conf file and the / dev path names that got created by the reconfiguration boot. If you are happy with them and want this rdriver.conf preserved through any subsequent reconfiguration boots, insert the line "#!noauto_config" into the .conf file.

Note that there is no provision for the rdriver.conf auto-generation process to leave some entries in the file alone while updating others; the file is either left unmodified, or completely rebuilt. If something like an incremental revision of the file is desired (e.g., in the case of adding a new RAID module), the user will need to either verify that all LUN distributions for all modules are as they should be before the reconfiguration boot, or save the current rdriver.conf, generate the new one, then manually merge the old and the new.

EXAMPLES

Here is a sample rdriver.conf file:

```
name="rdriver" module=1 lun=0 parent="/pseudo/rdnexus@1"; name="rdriver"
module=1 lun=1 parent="/pseudo/rdnexus@2"; name="rdriver" module=1 lun=2
parent="/pseudo/rdnexus@1"; name="rdriver" module=1 lun=3
```

```
parent="/pseudo/rdnexus@2"; name="rdriver" module=1 lun=4
parent="/pseudo/rdnexus@1"; name="rdriver" module=1 lun=5
parent="/pseudo/rdnexus@2"; name="rdriver" module=1 lun=6
parent="/pseudo/rdnexus@1"
   nex="pseudo/rdnesux@1" unbound=1;
name="rdriver" module=1 lun=6 parent="/pseudo/rdnexus@2"
   nex="pseudo/rdnesux@2" unbound=1;
name="rdriver" module=1 lun=7 parent="/pseudo/rdnexus@1"
   nex="pseudo/rdnesux@1" unbound=1;
name="rdriver" module=1 lun=7 parent="/pseudo/rdnexus@2"
   nex="pseudo/rdnesux@2" unbound=1;
```

This .conf file defines LUNs on the module "RAID Module 01". LUNs 0 through 5 are existing LUNs and are distributed between the two controllers on an odd/even basis. LUNs 6 and 7 are non-existent ("unbound") and have two entries each because they could potentially be created on either path.

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rm6**(1M), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

rmevent – RAID Manager Event File Format

SUMMARY

RAID Manager Event File Format

DESCRIPTION

This is the file format used by RAID Manager applications to dispatch an event to the **rmscript** notification script. It also happens to be the format for the proprietary log, used by the proprietary log viewer.

One event record consists of a single line of ASCII text, with each field separated by tildes "~". The line can be separated into a common portion and an event-specific portion. The common portion looks like:

{Host}~{Controller Serial #}~{Device Name}~{Date}~{Time}~{Event Type}~

Where

{Host}
: Is the name of the computer that generated the event.

{Controller Serial # or RAID Module Identification}
: Is the controller serial number or RAID module name that the disk array uses to identify itself. There may be a colon ":" followed by a number if the event occurred on a particular LUN.

{Device Name}
: Is a system specific name which the host uses to access the device, whether it is the controller or a LUN on the controller.

{Date} and {Time}
: The date and time when the event occurred.

{Event Type}
: Is a number which specifies the event type. The possible numbers are 00 for an AEN event, 10 for a mode page event, 20 for a parity event, 30 for an I/O error event, and 90 for a text event.

The event-specific fields for the AEN (00) and I/O error (30) events are:

{ASC/ASCQ Codes}~{FRU}~{LUN #}~{LUN Status}~{Sense Data}

Where

{ASC/ASCQ Codes}
: Are four ASCII-encoded hexadecimal digits, the first two specifying the Additional Sense Code, and the second two specifying the Additional Sense Code Qualifier.

{FRU}
: Is a two digit ASCII-encoded hexadecimal number representing the Field Replaceable Unit that is affected by the AEN. It is only valid if LUN status field is nonzero.

{LUN #}
: Is the LUN number (should be the same as any LUN specified in the {Controller Serial #} field) of the affected LUN. This field is only valid if

the LUN status field is nonzero.

{LUN Status}
Is an ASCII-encoded hexadecimal number which represents the LUN status.

{Sense Data}
Is a long string of ASCII-encoded hexadecimal digits which represent the sense data returned by the SCSI device. Every two digits specify one byte.

The event-specific fields for the Parity (20) event are:

{First Block}~{Last Block}~{Block count}~{LUN Number}~{Condition Code}

Where

{First Block}
Specifies the beginning of a region where parity errors were found.

{Last Block}
Specifies the end of a region where errors were found.

{Block Count}
Is the number of blocks in the region that have parity errors.

{LUN Number}
Is the LUN where the parity errors were found.

{Condition Code}
Is a special number used by RAID Manager's SNMP trap dispatcher.

The event-specific fields for the Mode Page (10) event are:

0~0~0~0~{Mode page data}

Where the {Mode page data} is a long string of ASCII-encoded hexadecimal digits which represent the sense data returned by the SCSI device. Every two digits specify one byte.

The event-specific fields for the Text (90) event are:

{Encoded string}~0~0~0~{Condition Code}

Where {Encoded string} is the message text, encoded so that newlines are represented by \n, tildes (~) are represented by \q, and backslashes (\) are represented by \\. The {Condition Code} is a special number used by RAID Manager's SNMP trap dispatcher.

Currently, the **arraymon** and **parityck** applications only generate AEN and parity events.

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmparams**(4), **rmscript**(1M)

NAME

rmparams – RAID Manager parameter file

DESCRIPTION

rmparams holds runtime parameters for the RAID Manager applications. The applications access these values via internal "get" and "put" interfaces. This scheme provides some control over the RAID Manager behavior without use of command line options or recompilation. The default location is **/etc/raid/rmparams.**

PARAMETER EXPLANATIONS

NOTE: This man page only documents parameters that are considered safe for users to modify. All other parameters should be left alone.

Monitor_PollInterval

This value is the time in minutes that **arraymon** daemon will sleep between successive scans of the attached arrays looking for abnormal events. The default value is 5 minutes. The range of acceptable values is 1 to 1440 (24 hours).

Monitor_ParityTime

This value is the time of day that the **arraymon** daemon will launch the parity check utility if enabled. The value is in the form hh:mm where hh is 0 to 23 and mm is 0 to 59. This utility should be run when system activity will be least impacted. The default time is 02:00 (2:00am).

Monitor_PchkEnabled

This value is tested by the **arraymon** daemon and if found 1 (enabled), the parity check utility is launched at the appropriate time. If the value is 0 (disabled), the parity check utility is not launched. The default value is 1 (enabled).

Parity_Amount

This value is the number of logical blocks that the parity check utility will scan before pausing for the amount of time specified by the Parity_Delay parameter. This parameter affects both automatic parity (if the Monitor_PchkEnabled parameter is 1 (enabled)) and manual parity. The default value is 10000 blocks. The range of acceptable values is 1000 to 37767 blocks.

Parity_Delay

This value is the amount of time in seconds that the parity check utility will pause for system I/O operations to take place. This parameter affects both automatic parity (if the Monitor_PchkEnabled parameter is 1 (enabled)) and manual parity. The default value is 1 second. The range of acceptable values is 0 to 10 seconds.

System_MaxControllers

This parameter is used by the identification module to dynamically allocate a memory structure to hold data describing each array controller found. The default value is 32. If more controllers are found than what the System_MaxControllers value is defined as, a message will be displayed on the system console noting that the utility will only keep

track of the first *x* number of controllers.

System_MaxHostAdapters
This parameter, along with System_MaxSCSIid and System_MaxLunsPerController, is used by the identification module to size a map for all existing device nodes. The max host adapters parameter itself specifies the largest c (controller) number found for all /dev/rdsk/cXtXdXsX nodes, plus one. The default value is 32.

System_MaxSCSIid
This parameter specifies the largest SCSI target ID, or largest number found for all /dev/rdsk/cXtXdXsX nodes. The default value is 31.

System_Max_LunsPerController
The disk array controller is capable of configuring up to 32 individual logical units (LUNs) per controller. The actual number of LUNs supported by an operating system varies. This parameter should be set to reflect the actual number of LUNs the OS supports. The default value is 8.

System_MaxLunsPerHostAdapter
This parameter is a decimal number that informs RAID Manager how many SCSI logical units may be accessed through a single host adapter. RAID Manager uses this information to determine in advance if it will be possible to create and register a new logical unit. Different operating systems or their releases will vary in what this parameter is allowed to be. The default value is 16.

System_CatalogLanguage
The core modules of the RAID Manager package have been internationalized by the use of a message catalog. The System_CatalogLanguage parameter corresponds to the set ID in the RAID Manager message catalog (**/etc/raid/messages.cat**). Supporting a new language involves adding a new set to the messages.txt file and building a new messages.cat file with the command 'gencat messages.cat messages.txt'. The new language is selected by setting the System_CatalogLanguage parameter to the new set ID. The currently defined sets are:

Set ID	Language
101	ENGLISH

The RAID Manager reserves sets 101 and above for its use. Sets 1-100 are available for user customization.

Notify_LogName
The name of the file where RAID Monitor notification places event entries.

Notify_LogEnabled
"1" if notification logging is enabled, "0" otherwise.

System_SafePassThroughLun
This parameter is a decimal number that specifies a particular logical unit to reserve for RAID Manager's use (in particular, this is the LUN that RAID Manager will use to send SCSI commands directly to drives in the array). Required if the controller software is earlier than 2.03. The specified LUN must exist and must not be used by any application other than RAID Manager.

Rdac_SupportDisabled
This parameter has two values, TRUE or FALSE. TRUE means the RDAC driver and daemons are not to be loaded when the system boots. FALSE (the default and assumed value if parameter is not present) means full RDAC support is enabled. When changing this parameter, perform a reconfiguration boot (boot -r) in order for the change to take effect.

SNMP_Target_IP_Address
This parameter specifies the IP address of the host that is to receive the SNMP traps sent by RAID Manager. This would normally be the IP address of the network management station. If this line is deleted(or commented out by placing a "#" character in the first position of the line), sending SNMP traps is disabled. The default value for this parameter is 127.0.0.1(loopback address).

SNMP_Community_Name
This parameter is the name of the SNMP community. The default value is "NONE". To enable SNMP traps, this parameter should be set to "public".

Viewer_LogSize
This parameter is a decimal number specifying the size in kilobytes that the RAID Manager message log file may grow to before "threshold reached" messages are displayed on application startup. The default value is 40.

SEE ALSO **arraymon**(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdac**(7), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmscript**(1M)

SEE ALSO arraymon(1M), lad(1M), nvutil(1M), parityck(1M) rm6(1M), raidutil(1M), rdac(7), rdacutil(1M) rdaemon(1M), rmevent(4), rmscript(1M), rdriver(4)

NAME

rules.tcl – The master set of event rules used by Tcl software in the Solstice SyMON program

SYNOPSIS

/opt/SUNWsymon/etc/rules.tcl

DESCRIPTION

This Tcl file contains a master list of event rules.

When you create a new rules file, add a **psource** command for the new rules file to the **rules.tcl** file so that the new rules file can be read.

For more information, see the *Solstice SyMON User's Guide*

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **sm_symond.conf**(4)

NAME

sm_symond.conf – list of agents for sm_symond to spawn and retrieve from other hosts

DESCRIPTION

The file **/etc/opt/SUNWsymon/sm_symond.conf** controls process spawning by **sm_symond**(1M). The processes most typically dispatched by **sm_symond** are symon agents.

The **sm_symond.conf** file is composed of entries that either list an agent and its arguments, or specify agents to run on remote machines.

Local agents are listed, one per line, with the normal command line arguments, and are invoked by sm_symond. Remote agent entries have the following format:

host:*agent-type*

Each entry is delimited by a newline. Comments may be inserted in the **sm_symond.conf** file by starting the line with a #.

The remote agent fields are:

host	The name of the remote host where the agent is to be run.
agent-type	The specific type of symon agent being run. Currently, the only agent type supported on remote machines is **EventGenerator.**

SEE ALSO

symon(1), **sm_configd**(1M), **sm_confsymon**(1M), **sm_control**(1M), **sm_egd**(1M), **sm_krd**(1M), **sm_logscand**(1M), **sm_symond**(1M), **verify_rules**(1M), **auth_checker.tcl**(4), **auth_list.tcl**(4), **event_gen.tcl**(4), **logscan.tcl**(4), **rules.tcl**(4)

NAME

speckeysd.map – Sun Special Keys to service map file for speckeysd

SYNOPSIS

/usr/openwin/lib/speckeysd.map

AVAILABILITY

SUNWpmow

DESCRIPTION

The **speckeysd.map** file is used by the **speckeysd**(1M) daemon to determine which Sun Special Keys to look for in the X Windows environment, and which service to spawn off to handle the keys.

The file is composed of entries for Sun Special Keys that are position-dependent and have the following format:

Sun Special Key Keysym Repeatable Service

Each entry is delimited by a newline. Each field is delimited by white-space (either a space or a tab). The whole entry must come before a newline, ie. you cannot extend lines by putting a backslash () preceding the newline.

The fields are:

Sun Special Key Keysym — Which Sun Special Key keysym should the **speckeysd**(1M) look for? Each key has a keysym associated with it in the X Windows environment. The Sun Special Keys and the Keysyms associated with them are:

Degauss Key	SunVideoDegauss
Mute Key	SunAudioMute
LowerVolume Key	SunAudioLowerVolume
LowerBrightness Key	SunVideoLowerBrightness
RaiseVolume Key	SunAudioRaiseVolume
RaiseBrightness Key	SunVideoRaiseBrightness
Power Key	SunPowerSwitch
Shift-Power Key	SunPowerSwitchShift

Repeatable — Is the Sun Special Key that **speckeysd (1m)** is supposed to look for repeatable? The valid options are:

r the key is repeatable
- the key is not repeatable

Service — Which service should be spawned off if one of the Sun Special Keys are pressed and what arguments should be passed to it? The service field is always considered to be everything after the Repeatable field and white-spaces following it to the newline character. To ensure that there are no PATH issues, specify the service with the complete path.

Comments are allowed in the file. However, the comments are full line entries, from an initial hash character (#) to the newline.

EXAMPLES

The following is a sample speckeysd.map file.

```
# This is the special keys service map file.
#
# This file will let speckeysd know what special keys (represented by X
# Windows Keysyms) to expect and what services to spawn off to handle the
# keys.
#
SunVideoRaiseBrightness         r       $OPENWINHOME/bin/contrast -k -u 1
SunVideoLowerBrightness         r       $OPENWINHOME/bin/contrast -k -d 1
SunPowerSwitch                  -       $OPENWINHOME/bin/sys-suspend
SunPowerSwitchShift     -       $OPENWINHOME/bin/sys-suspend -n
```

NOTES

If the file is changed and the system is already in X Windows, the **speckeysd (1M)** daemon must be restarted to pick up the changes.

SEE ALSO

speckeysd(1M)

NAME

cpr – Suspend and resume module

SYNOPSIS

/kernel/misc/cpr

AVAILABILITY

SUNWcpr

DESCRIPTION

cpr is a loadable module which is used to suspend and resume the whole system. You may wish to suspend a system to save power, or to temporarily power off for transport. It should not be used in place of a normal shutdown when performing any hardware reconfiguration or replacement. In order for resume to succeed, it is important that the hardware configuration remain the same. When the system is suspended, the entire system state is preserved in nonvolatile storage until a resume operation is conducted.

The principle way to suspend the system using this module is through the **sys-suspend**(1M) command. There are other utilities which may be installed on your system which will also access this module (such as **uadmin**(1M), **uadmin**(2), or the *Power* key and the *Shift+Power* key on a type 5 keyboard).

The module performs the following actions when suspending the system. The signal SIGFREEZE is first sent to all user threads and then the threads are stopped. The system is brought down to a uni–processor mode for multi–processor systems. Next dirty user pages are swapped out to their backing storage device and all file systems are synchronized. All devices are made quiescent and system interrupts are disabled. To complete the system suspend, the kernel memory pages and remaining user pages are written to the root file system in a compressed form.

When the system is powered on again, essentially the reverse of the suspend procedure occurs. The kernel image is restored from the root file system by the bootstrapper **/cprboot,** interrupts and devices are restored to their previous state. Finally the user threads are rescheduled and SIGTHAW is broadcast to notify any interested processes of system resumption. Additional processors, if available, are restored and brought online. The system is now back to exactly the state prior to suspension.

In some cases the **cpr** module may be unable to perform the suspend operation. If a system contains additional devices outside the standard shipped configuration, it is possible that these additional devices may not support **cpr.** In this case, the suspend will fail and an error message will be displayed to that effect. These devices must be removed or its device drivers unloaded for suspend to work. Contact the device manufacturer to obtain a new version of device driver that supports **cpr.** A suspend may also fail when devices or processes are performing critical or time sensitive operations (e. g. real time operations). In this case the system will remain in its current running state. Messages reporting the failure will be displayed on the console and status returned to the caller. Once the system is successfully suspended the resume operation will always succeed barring external influences such as hardware reconfiguration or the like.

Some network based applications may fail across a suspend and resume cycle. This largely depends on the underlying network protocol and the applications involved. In general, applications that retry and automatically reestablish connections will continue to

operate transparently on resume, those applications that do not, will likely fail.

The speed of suspend and resume can range from 15 seconds to a few minutes depending on the system speed, memory size and load. The typical time is around a minute.

FILES

/cprboot	special bootstrapper for cpr
/.CPR	system state file
/.cpr_generic_info	sys-suspend control file
/.cpr_defaultboot_info	sys-suspend control file

BUGS

The signals SIGFREEZE and SIGTHAW are not properly implemented for the Solaris 2.4 release, it will be available in a later release. This should only be a concern for specially customized applications that need to perform additional tasks at suspend or resume time, which none exists at the present time.

In extremely rare occasions the system may fail during the early stages of a resume. In this small window it is theoretically possible to be stuck in a loop that the system does not resume and it does not boot normally. If you are in such a loop, get to the prom ok prompt via *L1+A* and enter the following command.

<ok> *set-default boot-file*

This resets the system and on the next power on the system will boot normally.

NOTES

For suspend / resume to work on multi-processor platforms, it must be able to control all CPUs. It is recommended that no MP tests (such as sundiag CPU tests) are running when suspend is initiated because the suspend may be rejected, if it cannot shut off all CPUs.

Certain device operations such as tape, floppy disk activities are not resumable due to the nature removable media. These activities are detected at suspend time, and must be stopped before suspend will complete successfully.

SEE ALSO

sys-suspend(1M), **uadmin**(1M), **uadmin**(2)

NAME

cvc – virtual console driver

DESCRIPTION

cvc is a STREAMS-based pseudodriver that supports the network console, which is called cvc on the host side and netcon on the SSP. **cvc** interfaces with **console**(7).

Logically, the **cvc** driver sits below the **console**(7) driver. It intercepts console output, redirecting it to the **cvcredir**(7) driver.

cvc receives console input from **cvcredir**(7) and passes it to the process associated with **/dev/console**.

NOTES

The cvc facility supercedes the SunOS **wscons**(7) facility, which should **not** be used in conjunction with cvc. **wscons**(7) is useful for systems with directly attached consoles (frame buffers and keyboards), but is not useful with the Enterprise 10000 system, which has no local keyboard or frame buffer.

SEE ALSO

cvcd(1M), **cvc**(7), **cvcredir**(7) in this reference manual
netcon(1M), **netcon_server(1M)** in *UNKNOWN TITLE ABBREVIATION: UE10000REFEMAN1M*
console(7) in *man Pages(7): Device and Network Interfaces*

NAME

cvcredir – virtual console redirection driver

DESCRIPTION

cvcredir, the virtual console redirection driver, is a STREAMS-based pseudodriver that works in conjunction with the cvc driver, **cvc**(7), and the cvc daemon, **cvcd**(1M).

The **cvcredir** device is opened at start-of-day by the cvc daemon, **cvcd**(1M). **cvcredir** receives console output from **cvc**(7) and passes it to **cvcd**(1M). It receives console input from **cvcd**(1M) and passes it to **cvc**(7).

SEE ALSO

cvcd(1M), **cvc**(7) in this reference manual

netcon(1M),**netcon_server(1M)** in *man Pages(1M): Ultra Enterprise 10000 SSP Administration Commands*

console(7) in *man Pages(7): Device and Network Interfaces*

NAME

dr – dynamic reconfiguration driver, /dev/dr

SYNOPSIS

dr

DESCRIPTION

The DR driver provides a pseudo-driver interface to the kernel Dynamic Reconfiguration (DR) Attach and DR Detach features.

For DR Detach, the command **dr_daemon**(1M) executes SunOS **ioctl**(2) calls to:

- Detach selected devices from kernel usage
- Remove detached device nodes from the kernel's device tree
- Direct OBP to delete all detached nodes from its device tree

For DR Attach, **dr_daemon**(1M) executes **ioctl**(2) calls to:

- Direct OBP to probe the board and add nodes to its device tree
- Get the nodes from OBP and add proto nodes to the kernel's device tree
- Convert the proto nodes to CF1 nodes

The pathname of the device node is **/devices/pseudo/dr@0:0**.

SEE ALSO

Dynamic Reconfiguration User's Guide
Ultra Enterprise 10000 SSP 3.0 User's Guide
dr_daemon(1M) in this reference manual
hostview(1M), **hpost(1M)** in *man Pages(1M): Ultra Enterprise 10000 SSP Administration Commands*
dr(1M) in *man Pages(1M): DR Administration Commands*
add_drv(1M), **drvconfig**(1M), **devlinks**(1M), **disks**(1M), **ports**(1M), **tapes**(1M) in *man Pages(1M): System Administration Commands*

NAME

ecpp – IEEE 1284 ecp, nibble and centronics compatible parallel port driver

SYNOPSIS

```
#include <sys/types.h>
#include <fcntl.h>
#include <sys/ecppio.h>

fd = open("/dev/ecpp0", flags);
```

DESCRIPTION

The **ecpp** driver provides a bi-directional interface to IEEE 1284 compliant devices. The driver will operate in Centronics mode for non-IEEE 1284 compliant devices. An IEEE 1284 compliant peripheral device must operate at least in Compatibility mode and Nibble mode. The **ecpp** driver supports Compatibility, Nibble and ECP modes of operation as defined by IEEE 1284. Centronics and Compatibility modes of operation have identical physical characteristics. However, non-IEEE 1284 compliant devices will be logically defined as ECPP_CENTRONICS. IEEE 1284 devices that are in a similar mode will be logically defined as ECPP_COMPAT_MODE. ECPP_COMPAT_MODE operates in conjunction with ECPP_NIBBLE_MODE. The **ecpp** driver is an *exclusive-use* device. If the device has already been opened, subsequent opens fail with **EBUSY.**

Default Operation

Each time the ecpp device is opened, the device is marked as EBUSY and the configuration variables are set to their default values. The write_timeout period is set to 60 seconds. The driver sets the mode variable according to the following algorithm: The driver initially attempts to negotiate the device into ECP mode. If this should fail, the driver will attempt to negotiate into Nibble mode. If Nibble mode negotiation should fail, the driver will operate in Centronics mode. The application may attempt to negotiate the device into a specific mode or set the write_timeout values through the **ECPPIOC_SETPARMS ioctl**(2) call. In order for the negotiation to be successful, both the host workstation and the peripheral must support the requested mode.

The preferred mode of operation of an IEEE 1284 device is the bi-directional ECP mode. Nibble mode is a unidirectional backchannel mode. It utilizes a PIO method of transfer and consequently, is inefficient. For devices that primarily receive data from the workstation, such as printers, Nibble operation will have limited impact to system performance. Nibble mode should not be used for devices such as a scanner, that primarily send data to the workstation. Forward transfers under all modes are conducted through a DMA method of transfer.

Read/Write Operation

ecpp is a full duplex STREAMS device driver. While an application is writing to an IEEE 1284 compliant device, another thread may read from it. **write**(2) will return when all the data has been successfully transferred to the device.

Write Operation

write(2) returns the number of bytes successfully written to the stream head. If a failure occurs while a Centronics device is transferring data, the content of the status bits will be captured at the time of the error, and can be retrieved by the application program, using the **ECPPIOC_GETERR ioctl**(2) call. The captured status information will be overwritten each time an attempted transfer or a **ECPPIOC_TESTIO ioctl**(2) occurs.

Intelligent IEEE 1284 compliant devices, such as Postscript printers, return error information through a backchannel. This data may be retrieved with the **read**(2) call.

Read Operation

If a failure or error condition occurs during a **read**(2), the number of bytes successfully read is returned (short read). When attempting to read the port that has no data currently available, **read**(2) returns 0 if O_NDELAY is set. If O_NONBLOCK is set, **read**(2) returns -1 and sets errno to EAGAIN. If O_NDELAY and O_NONBLOCK are clear, **read**(2) blocks until data become available.

IOCTLS

The following **ioctl**(2) calls are supported:

ECPPIOC_GETPARMS

Get current transfer parameters.
The argument is a pointer to a **struct ecpp_transfer_parms**. See below for a description of the elements of this structure. If no parameters have been configured since the device was opened, the structure will be set to its default configuration. (see **Default Operation** above).

ECPPIOC_SETPARMS

Set transfer parameters.
The argument is a pointer to a **struct ecpp_transfer_parms**. If a parameter is out of range, **EINVAL** is returned. If the peripheral or host device can not support the requested mode, **EPROTONOSUPPORT** is returned. See below for a description of ecpp_transfer_parms and its valid parameters.

Transfer Parameters Structure

This structure is defined in **<sys/ecppio.h>.**

```
struct ecpp_transfer_parms {
  int    write_timeout;
  int    mode;
};
```

The **write_timeout** field is set to **ECPP_W_TIMEOUT_DEFAULT.** The **write_timeout** field specifies how long the driver will wait for the peripheral to respond to a transfer request. The value must be greater than 0 and less than ECPP_MAX_TIMEOUT. Any other values are out of range.

The **mode** field reflects the IEEE 1284 mode that the parallel port is currently configured to. The mode may be set to only one of the following bit values.

```
#define ECPP_CENTRONICS      0x1
#define ECPP_COMPAT_MODE     0x2
#define ECPP_NIBBLE_MODE     0x3
```

```
#define ECPP_ECP_MODE          0x4
#define ECPP_FAILURE_MODE      0x5
```

This command may set the mode value to ECPP_CENTRONICS, ECPP_COMPAT_MODE, ECPP_NIBBLE_MODE, or ECPP_ECP_MODE. All other values are not valid. If the requested mode is not supported, ECPPIOC_SETPARMS will return EPROTONOSUPPORT. Under this circumstance, ECPPIOC_GETPARMS will return to its original mode. If a non-recoverable IEEE 1284 error occurs, the driver will be set to ECPP_FAILURE_MODE. For instance, if the port is not capable of returning to its orignal mode, ECPPIOC_GETPARMS will return ECPP_FAILURE_MODE.

BPPIOC_TESTIO

Tests the transfer readiness of ECPP_CENTRONICS or ECPP_COMPAT_MODE devices.
If the current mode of the port is ECPP_CENTRONICS or ECPP_COMPAT_MODE, this command determines if write (2) would succeed. If it is not one of these modes, EINVAL is returned. BPPIOC_TESTIO determines if a **write**(2) would succeed by checking the open flag and status pins. If any of the status pins are set, a transfer would fail. If a transfer would succeed, zero is returned. If a transfer would fail, -1 is returned, and **errno** is set to **EIO** , and the state of the status pins is captured. The captured status can be retrieved using the **BPPIOC_GETERR ioctl (2)** call. Note that the **timeout_occurred** and **bus_error** fields will never be set by this **ioctl**(2)**. BPPIOC_TESTIO** and **BPPIOC_GETERR** are compatible to the ioctls specified in **bpp**(7). However, **bus_error** is not used in this interface.

BPPIOC_GETERR

Get last error status.
The argument is a pointer to a **struct bpp_error_status**. This structure is described below. This structure indicates the status of all the appropriate status bits at the time of the most recent error condition during a **write**(2) call, or the status of the bits at the most recent **BPPIOC_TESTIO ioctl**(2) call.

The **timeout_occurred** value is set when a timeout occurs during write (2). **bus_error** is not used in this interface.

pin_status indicates possible error conditions under ECPP_CENTRONICS or ECPP_COMPAT_MODE. Under these modes, the state of the status pins will indicate the state of the device. For instance, many Centronics printers lower the nErr signal when a paper jam occurs. The behavior of the status pins depends on the device. As defined in the IEEE 1284

Specification, status signals do not represent the error status of ECP devices. Error information is formatted by a printer specific protocol such as PostScript, and is returned through the backchannel.

Error Status Structure

struct bpp_error_status is defined in the include file **<sys/bpp_io.h>.** The valid bits for **pin_status** are presented below. A set bit indicates that the associated pin is asserted. For example, if BPP_ERR_ERR is set, nErr is asserted.

```
struct    bpp_error_status {
  char   timeout_occurred; /* 1=timeout */
  char   bus_error;       /* not used */
  u_char pin_status;    /*
                                * status of pins
                    * which could cause
                    * error.
                    */
};

/* pin_status values */
#define BPP_ERR_ERR      0x01  /* nErr=0 */
#define BPP_SLCT_ERR     0x02  /* Select=1 */
#define BPP_PE_ERR       0x04  /* PE =1 */
#define BPP_BUSY_ERR     0x40  /* Busy = 1  */
```

ERRORS

EBADF The device is opened for write-only access and a read is attempted, or the device is opened for read-only access and a write is attempted.

EBUSY The device has been opened and another open is attempted.
An attempt has been made to unload the driver while one of the units is open.

EINVAL A **ECPPIOC_SETPARMS ioctl( )** is attempted with an out of range value in the **ecpp_transfer_parms** structure.
A **ECPPIOC_SETREGS ioctl( )** is attempted with an invalid value in the ecpp_regs structure.
An **ioctl( )** is attempted with an invalid value in the command argument.
An invalid command argument is received from the vd driver (during **modload**(1M), **modunload**(1M).

EIO The driver encountered a bus error when attempting an access.

A read or write does not complete properly, due to a peripheral error or a transfer timeout.

ENXIO The driver has received an open request for a unit for which the attach failed. The driver has received a write request for a unit which has an active peripheral error.

FILES

/dev/ecpp0 1284 compatible and ecp mode parallel port device

SEE ALSO

ioctl(2), **read**(2), **write**(2), **streamio**(7)

NAME

m64 – 8-bit color memory frame buffer

SYNOPSIS

/dev/fbs/m64

DESCRIPTION

m64 is the Mach64 8-bit color frame buffer and graphics accelerator, with 8-bit colormap. It provides the standard frame buffer interface defined in **fbio**(7).

The m64 has registers and memory that may be mapped with **mmap**(2), using the offsets defined in **<sys/m64.h>**.

m64 accepts the following ioctls, defined in <sys/fbio.h> and <sys/visual_io.h>:

FBIOGATTR, FBIOGTYPE, FBIOPUTCMAP, FBIOGETCMAP, FBIOSATTR, FBIOSVIDEO, FBIOGVIDEO, FBIOVERTICAL, FBIOSCURSOR, FBIOGCURSOR, FBIOSCURPOS, FBIOGCURPOS, FBIOGCURMAX, FBIOGXINFO, FBIOMONINFO, FBIOVRTOFFSET, and VIS_GETIDENTIFIER are all implemented as described in fbio(7).

The value returned by VIS_GETIDENTIFIER is "SUNWm64".

FBIOPUTCMAP returns immediately, although the actual colormap update may be delayed until the next vertical retrace. If vertical retrace is currently in progress, the new colormap takes effect immediately.

FBIOGETCMAP returns immediately with the currently-loaded colormap, unless a colormap write is pending (see above), in which case it waits until the colormap is updated before returning. This may be used to synchronize software with colormap updates.

The size and linebytes values returned by FBIOGATTR, FBIOGTYPE and FBIOGXINFO are measured in bytes. The proper way to compute the size of a framebuffer mapping is *size=linebytes*height* or to use the **size** attribute in **FBIOGATTR, FBIOGTYPE**.

There is extra on-board memory which may be used for scratch-pad, double-buffering or off-screen rendering. The total amount of memory on the board may be found with the FBIOGATTR ioctl. Total mappable memory, including on-screen memory, is **attr.sattr.dev_specific[0]**.

The chip revision number is returned in **dev_specific[2]**.

The dac revision number is returned in **dev_specific[3]**.

The prom revision number is returned in **dev_specific[4]**.

The byte offset from the start of the framebuffer to the start of the visible part of the framebuffer is returned in **dev_specific[5]**

The Mach64 has a 2-color cursor. The color is determined by the mask and data planes, as written by the FBIOSETCURS ioctl. mask:data combinations are as follows: 0x=transparent, 10=color0, 11=color1.

Maximum cursor size is 64x64 pixels. The Mask and Image pointers in the fbcursor structure should point to data which is zero-padded to 32-bits per scanline and aligned on a 32-bit boundary.

Ioctl functions which nominally wait for vertical retrace (**FBIOVERTICAL, FBIOGETCMAP**) do not wait, but return immediately, if video is blanked or vertical retrace is not being generated. The vertical retrace counter page is not updated if vertical retrace is not being generated. Vertical retrace is not generated when the device is in energy-saving mode.

FILES

/dev/fbs/m64*n*	device special file
/dev/fb	default frame buffer
/usr/include/sys/m64.h	device-specific definitions

SEE ALSO

mmap(2), **fbio**(7)

NAME

mic – Multi-interface Chip driver

SYNOPSIS

```
#include <fcntl.h>
#include <sys/termios.h>
#include <sys/micio.h>

open("/dev/term/mic/a", mode);
open("/dev/term/mic/b", mode);
open("/dev/term/mic/ir", mode);
```

AVAILABILITY

SUNWmic

PLATFORM

SPARCstation Voyager

DESCRIPTION

The Multi-interface Chip (MIC) provides two asynchronous serial input/output channels. These channels provide high speed buffered serial I/O, with optional hardware flow control support. Baud rates from 110 to 115200 are supported.

The first channel can either be routed through an infra-red port or the "a" serial port. If the device is opened using the "ir" device, then the driver routes the first channel through the infra-red port. If the device is opened using the "a" device the first channel is routed through the "a" serial port. You cannot use both the "a" port and the "ir" port simultaneously. The second channel (the "b" serial port) has no infra-red capability and may be used independently of the first channel.

The **mic** module is a loadable STREAMS driver that provides basic support for the MIC hardware, together with basic asynchronous communication support. The driver supports those **termio**(7) device control functions specified by flags in the **c_cflag** word of the **termios** structure, excluding HUPCL, CLOCAL, CIBAUD, CRTSCTS and PAREXT. The driver does not support device control functions specified by flags in the **c_iflag** word of the **termios** structure. Specifically, the driver assumes that IGNBRK and IGNPAR are always set. All other **termio**(7) functions must be performed by STREAMS modules pushed atop the driver. When a device is opened, the **ldterm**(7) and **ttcompat**(7) STREAMS modules are automatically pushed on top of the stream, providing the standard **termio**(7) interface.

The infra-red port provides access to two different modes of modulation. The default mode is called pulse mode and is compatible with the Infra-red Data Association (IrDA) modulation and the Hewlett-Packard Serial Infra-red (SIR) modulation. The second modulation is called high frequency mode and is compatible with the Sharp Amplitude Shift Keying (ASK) modulation. The default modulation when using high frequency mode is 500 KHz.

The character-special devices **/dev/term/mic/a** and **/dev/term/mic/b** are used to access the two serial ports on the MIC chip.

The character-special device **/dev/term/mic/ir** is used to access the infra-red port of the chip.

IOCTLS

The standard set of **termio ioctl**( ) calls are supported by the **mic** driver.

Breaks can be generated by the **TCSBRK**, **TIOCSBRK**, and **TIOCCBRK ioctl**( ) calls.

The input and output line speeds may be set to any of the speeds supported by **termio**. The speeds cannot be set independently; when the output speed is set, the input speed is set to the same speed. To support higher speeds than defined in **termio** the two lowest speeds, B50 and B75, have been remapped to 96000 and 115200 baud respectively.

There are six **ioctl**( ) calls which are specific to the infra-red port and can only be used when the device has been opened in infra-red mode:

MIOCGETM_IR
Returns the current IR mode defined in micio.h

MIOCSETM_IR
Takes an additional argument of the desired IR mode (defined in micio.h) and sets the port to this mode.

MIOCGETD_IR
Returns the current IR carrier divisor. The carrier frequency can be calculated from the divisor and the formula:

carrier frequency = 19660 / (4 (divisor +1)) KHz

MIOCSETD_IR
Sets the current IR carrier divisor. The desired frequency can be set by using a divisor calculated by the following formula, where the frequency is specified in KHz:

divisor = 19660 / frequency / 4 – 1

MIOCSLPBK_IR
Set IR loopback mode. This enables the receiver during transmit, so that sent messages are also received through the IR port.

MIOCCLPBK_IR
Clears IR loopback mode.

There are two **mic** specific **ioctl( )** calls:

MIOCSLPBK
Set SCC loopback mode. This internally loops back transmitted messages within the channel.

MIOCCLPBK
Clear SCC loopback mode.

ERRORS

An **open**( ) will fail if:

ENXIO The unit being opened does not exist.

EBUSY The channel is in use by another serial protocol. Remember that both the "a"

and "ir" ports use the same channel.

FILES

/dev/term/mic/a	asynchronous serial line using port a
/dev/term/mic/b	asynchronous serial line using port b
/dev/term/mic/ir	asynchronous serial infra-red line using the infra-red port

DIAGNOSTICS

mic: Rx FIFO overflow
The mic's internal 64 character buffer overflowed before it could be serviced.

mic: Rx buffer full - draining
The driver's character input buffer overflowed before it could be serviced.

NOTES

Currently hardware flow control is not implemented. The state of DCD, CTS, RTS and DTR interface signals cannot be queried, nor can hardware flow control be enabled using the CRTSCTS flag in the **c_cflag** word of the **termios** structure.

SEE ALSO

tip(1), **ports**(1M), **ioctl**(2), **open**(2), **ldterm**(7), **termio**(7), **ttcompat**(7),

NAME

pm – Power Management Driver

SYNOPSIS

#include <sys/pm.h>
int ioctl(int fildes, int command, int arg);

AVAILABILITY

SUNWpmu

DESCRIPTION

The Power Management driver provides an interface for applications to configure the devices within the system for power management. The interface is provided through **ioctl**(2) commands. The *pm* driver may be accessed using **/dev/pm.**

fildes is an open file descriptor that refers to the **pm** driver. *command* determines the control function to be performed as described below. *arg* represents additional information that is needed by this command. The type of *arg* depends upon the *command,* but it is generally an integer or a pointer to a command-specific data structure.

COMMAND FUNCTIONS

Unless configured by using the commands below, **pm** does not power manage devices by default. Note, however, that the **pmconfig**(1M) program is typically run at boot time, and by reading the **power.conf**(4) file will use the commands below to configure **pm.** Any devices configured for power management by **pm** will have their drivers loaded (if not already) and locked into memory until that device is unmanaged. Some devices may be able to fully operate at non–full power levels. Using the command **PM_SET_POWER** on such a device allows this low power mode to become the normal (on) power level for that device. This mode of operation is distinct from the power managed mode of operation.

pm periodically searches the system for devices which it can power manage. A device will only be power managed when it is not in use (explained further below). When a power managed device is subsequently used, it will be automatically returned to normal power.

The **pm** model of power management is to view the system as a collection of devices. Each device is a collection of components, a component is the smallest power manageable unit. The devices, and the components within those devices, which are power manageable are dependent upon the implementation of their respective device drivers. A power manageable component has three states. It may be *busy* (in use), it may be *idle* (not in use but using normal power), or it may be *power managed* (not in use and not using normal power). The **pm** driver manages the component transition from the second to the third state. **pm** uses two factors to determine this transition: the component must have been idle for at least the threshold time; and the device to which the component belongs must satisfy any dependencies requirements. A dependency is when a device requires another device to be power managed before it can be power managed. A device is considered to be power managed when all of its components are power managed. Note that dependencies occur on a per device basis: when a dependency exists, no components of a device may be managed unless all the components it depends upon are first managed. For more information, see the **Guide to Writing Device Drivers manual, attach**(9E), **detach**(9E), **power**(9E).

Thus the configuration of a device for power management is the setting of the threshold for any component that is to be managed and defining any dependencies for that device.

For all commands excluding **PM_SCHEDULE, arg** points to a structure of type *pm_request* defined in **sys/pm.h:**

```
typedef struct {
        char    *who;           /* device to configure */
        int     select;         /* selects the component or
                                   dependent of the device */
        int     level;          /* power or threshold level */
        char    *dependent;     /* hold name of dependent */
        int     size;           /* size of dependent buffer */
} pm_request;
```

The fields should contain the following data. *who* is a pointer to the name of the device to be configured. The name must be in the format described in **power.conf**(4). *select* is a non–negative integer specifying the component or dependent being configured. The numbering starts at zero. *level* is non–negative integer giving the threshold level in seconds or the desired power level. *dependent* is a pointer to a buffer which contains or receives the name of a device on which this device has a dependency. It uses the same format as the first field. *size* is the size of the dependent buffer.

Not all fields are used in each command. Upon error the commands will return -1, and set *errno* to the error condition specified below. The following error codes are common to all commands.

EFAULT: Bad address passed in as argument.

ENODEV:
Device is not power manageable, or device is not configured (Use **PM_SET_THRESHOLD** command first).

ENXIO: Invalid instance number (device not attached).

EPERM: Permission denied. You must be root or console owner.

PM_SCHEDULE:
arg sets the period in seconds of **pm** device scans. A value of zero inhibits scans which stops any further components from being managed. A negative value is ignored. The ioctl returns the new (or current) period.

PM_GET_IDLE_TIME:
Using the fields *who* and *select*, this command returns the time in seconds since the component was last busy. Error codes:

EINVAL: Device component out of range.

PM_GET_NUM_CMPTS:
Using the field *who*, this command returns the number of components defined for this device.

PM_GET_THRESHOLD:
Using the fields *who* and *select, this command returns the* threshold level of the component. Error codes:

EINVAL: Device component out of range.

PM_SET_THRESHOLD:
Using the fields *who, select* and *level,* this command sets the threshold level of the component. It returns zero on success. Error codes:

EINVAL: Device component out of range, or threshold value < 0.

PM_GET_POWER:
Using the fields *who* and *select,* this command returns the current normal power level of the component.

EINVAL: Device component out of range.

EIO: Non−power manageable device (or properties are removed).

PM_SET_POWER:
Using the fields *who, select* and *level,* this command sets the current normal power level of the component to the given power level.

EINVAL: Device component out of range, or power level <= 0.

EIO: Failed to power device or its parent or its dependents.

PM_GET_CUR_PWR:
Using the fields *who* and *select,* this command returns the current power level of the component.

EINVAL: Device component out of range.

PM_GET_NUM_DEPS:
Using the field *who,* this command returns the number of dependents configured for this device.

PM_GET_DEP:
Using the fields *who, select, level* and *dependent,* this command writes the name of dependent into the buffer supplied by the *dependent* field.

EINVAL: Dependent component out of range, or user buffer is too small for dependent name

EFAULT: Bad buffer address was given.

PM_ADD_DEP:
Using the fields *who* and *dependent,* this command adds the dependent to the device.

ENODEV: Dependent is non−power manageable or is not configured.

PM_REM_DEP:
Using the fields *who* and *dependent,* this command removes the dependent from the device.

ENODEV: Dependent is non−power manageable or is not configured, or the device has no dependents

PM_REM_DEVICE:
Using the field *who*, this command unmanages the device and returns the device to normal power, if it is not already.

PM_REM_DEVICES:
This command unmanages all devices and returns them to normal power.

NOTES

To unload a power managed driver, the driver must first be unmanaged using **PM_REM_DEVICE(S).**

Currently it is NOT an error to remove a nonexistent dependent or add a repeated dependent. The pseudo driver will silently ignore the redundant command.

SEE ALSO

intro(2), **ioctl**(2), **pmconfig**(1M), **power.conf**(4), **attach**(9E), **detach**(9E), **power**(9E)

NAME

pmc – Platform Management Chip driver

SYNOPSIS

#include <sys/pmcio.h>
int ioctl(int fildes, int command, int arg);

AVAILABILITY

SUNWpmc

PLATFORM

SPARCstation Voyager

DESCRIPTION

The Platform Management Chip driver provides a number of miscellaneous platform specific functions. Principally these are to provide power control for devices which cannot manage their own power control (see **ddi_power (9F)**) and to provide information about the connection status of the machine. Not all functions are supported on all platforms.

The user interface is provided through **ioctl (2)** commands. The **pmc** driver may be accessed using **/dev/pmc.** The system interface (to power manage devices) is provided by registering its power function (using the "platform-pm" property of the root node).

fildes is an open file descriptor that refers to the **pmc** driver. *command* determines the control function to be performed as described below. *arg* is not used and may be any value.

COMMAND FUNCTIONS

These functions fall into three categories: connection status, power control and miscellaneous. Connection status can be used to find out whether the following devices are plugged in: keyboard, ethernet and ISDN.

The power control function controls the removal of the platform power. Miscellaneous functions enable the reading of the digital to analog converter.

PMC_GET_KBD:
This command returns the connection status of the keyboard. When the keyboard is connected it will return **PMC_KB_STAT,** and zero when it is not connected.

PMC_GET_ENET:
This command returns the connection status of the ethernet. When the ethernet is connected it will return **PMC_ENET_STAT,** and zero when it is not connected.

PMC_GET_ISDN:
This command returns the connection status of the isdn channels. The return value is a bit map of the connected channels: **PMC_ISDN_ST0** for **NT, PMC_ISDN_ST1** for **TE.**

PMC_GET_A2D:
This command returns the result of an eight bit analog to digital conversion. The meaning of the reading is platform specific.

PMC_POWER_OFF:
This command is only available to the super-user. It turns off all power to the

system. Note that critical data may be lost if proper preparation prior to power removal is not performed.

The **poll(2)** interface is supported. It may be used to poll for connection status changes. A process wishing to detect such connection changes should use the **POLLIN** event flag. When ANY connection status changes, the **poll (2)** mechanism will be notified. It is up to the user to verify whether the connection status change is of interest.

ERRORS

EPERM Must be privileged user to use **PMC_POWER_OFF.**

SEE ALSO

ddi_power(9F), **intro**(2), **ioctl**(2), **open**(2), **pm**(7), **poll**(2)

NAME

rdac – Redundant Disk Array Controller Support

OVERVIEW

Redundant disk array controller (RDAC) support in RAID Manager provides application-transparent protection for array controller and data path failures using a special-purpose daemon process and kernel-resident I/O driver. The RDAC software achieves this protection by positioning the kernel driver, which is primarily responsible for path routing, between the SCSI disk driver and the rest of the operating system. From this vantage point, the RDAC driver has full visibility of the I/O traffic to the disk array and is able to intervene and use the redundant path to recover controller and path failures.

The other essential part of this recovery action is provided by the "error resolution daemon", whose responsibility it is to interpret and analyze error conditions, determine the appropriate course of action and disposition of the failed I/O, and, if necessary, direct a surviving controller to take over the control of logical units from a failed controller.

In short, failed I/Os returning from the SCSI driver will result in the RDAC driver communicating the failure to the daemon, which will resolve the error as it sees fit, and communicate that resolution back to the RDAC driver for continuation of I/O processing, possibly now using the alternate path. All of this occurs without the OS or application realizing that there has been an error.

COMPONENTS

Major components of the RDAC support in RAID Manager consist of:

RDAC Kernel Driver

The RDAC driver is responsible for path routing and participating in the interaction with the daemon on error resolution cycles.

Resolution Daemon

The resolution daemon receives notice from the driver when an I/O failure occurs and attempts to decide if the error is a "routine" condition (such as a LUN transfer for purposes of load balancing), or if it signifies a controller or data path failure, in which case it initiates "failover" to the alternate.

rdacutil Command Line Utility

The RDAC software also includes a command line utility, **rdacutil,** which provides for scriptable RDAC management, including failover, controller mode transitions, and LUN load balancing.

FILES

The redundant path protection of RDAC is dependent on use of device special files that refer to the RDAC driver, which in effect constitute a "virtual," highly reliable path to the array. The general form of these special files (or device nodes) is:

/dev/[r]dsk/rdac/c<a>t<b>_c<c>t<d>.d<e>s<f>

where the optional [r] signifies character, rather than block device; <a> and <b> are the "controller-target" designators of one path, and <c> and <d> are the controller-target desingators of the other path; <e> is the disk (logical unit) number; and <f> is the slice number.

Examples:

/dev/dsk/rdac/c2t1_c1t2.d0s1 /dev/rdsk/rdac/c2t3_c1t4.d6s0

In addition, RDAC sets up soft links to the above nodes that are easier to work with and which provide a clear correlation to the RAID module names used in the RAID Manager graphical applications. These names have the form:

/dev/[r]<RAID module name>/<a>s<b>

where the optional [r] signifies character, rather than block device; <RAID module name> is the logical name of the RAID module used in RAID Manager; <a> is a logical unit number; and <b> is a slice number.

Examples:

/dev/RAID_Module_01/0s1
/dev/rRAID_Module_01/6s0

NOTES

1. Currently it is not possible for the RDAC driver to control the root or swap devices of the operating system.
2. For the greatest level of I/O path protection, each controller in a redundant controller pair should have its own SCSI connection (i.e., SCSI bus and host adapter) into the system.

SEE ALSO

arraymon(1M), **lad**(1M), **nvutil**(1M), **parityck**(1M), **raidutil**(1M), **rdacutil**(1M), **rdaemon**(1M), **rdriver**(4), **rm6**(1M), **rmevent**(4), **rmparams**(4), **rmscript**(1M)

NAME

fas – FAS SCSI Host Bus Adapter Driver

SYNOPSIS

fas@*sbus-slot*,0x8800000

AVAILABILITY

Limited to Sparc SBus-based systems with FAS366 based SCSI port, platforms and SBus SCSI Host Adapter options TBD.

DESCRIPTION

The **fas** Host Bus Adapter driver is a SCSA compliant nexus driver that supports the Qlogic FAS366 SCSI chip.

The **fas** driver supports the standard functions provided by the SCSA interface. The driver supports tagged and untagged queuing, wide and fast SCSI, almost unlimited transfer size (using a moving DVMA window approach), auto request sense but does not support linked commands.

Driver Configuration

The **fas** driver can be configured by defining properties in **fas.conf** which override the global SCSI settings. Supported properties are **scsi-options**, **target<***n***>-scsi-options**, **target<***n***>-sync-speed**, **target<***n***>-wide**, **target<***n***>-TQ**, **scsi-reset-delay**, **scsi-watchdog-tick**, **scsi-tag-age-limit**, **scsi-initiator-id**.

target<*n***>-scsi-options** overrides the **scsi-options** property value for **target<***n***>**. <*n*> can vary from 0 to f. The supported scsi-options are SCSI_OPTIONS_DR, SCSI_OPTIONS_SYNC, SCSI_OPTIONS_TAG, SCSI_OPTIONS_FAST, SCSI_OPTIONS_WIDE.

scsi-watchdog-tick is the periodic interval where the **fas** driver goes through all current and disconnected commands searching for timeouts.

scsi-tag-age-limit is the number of times that the **fas** driver attempts to allocate a particular tag ID that is currently in use after going through all tag IDs in a circular fashion. After finding the same tag ID in use **scsi-tag-age-limit** times, no more commands will be submitted to this target until all outstanding commands complete or timeout.

Refer to **scsi_hba_attach**(9F) for details.

EXAMPLES

Create a file **/kernel/drv/fas.conf** and add this line:

scsi-options=0x78;

This will disable tagged queuing, fast SCSI, and Wide mode for all **fas** instances. To disable an option for one specific **fas** (refer to **driver.conf**(4)):

```
name="fas" parent="/iommu@f,e0000000/sbus@f,e0001000"
    reg=3,0x8800000,0x10,3,0x8810000,0x40
    target1-scsi-options=0x58
    scsi-options=0x178 scsi-initiator-id=6;
```

Note that the default initiator ID in OBP is 7 and that the change to ID 6 will occur at attach time. It may be preferable to change the initiator ID in OBP.

The above would set scsi-options for target 1 to 0x58 and all other targets on this SCSI bus to 0x178.

The physical pathname of the parent can be determined using / devices tree or following the link of the logical device name:

```
# ls -l /dev/rdsk/c1t3d0s0
lrwxrwxrwx 1 root  other  78 Aug 28 16:05 /dev/rdsk/c1t3d0s0 ->
../../devices/iommu@f,e0000000/sbus@f,e0001000/SUNW,fas@3,8800000/sd@3,0:a,raw
```

The register property values can be determined from **prtconf**(1M) output (-v option):

```
SUNW,fas, instance #0
    ....
    Register Specifications:
        Bus Type=0x3, Address=0x8800000, Size=10
        Bus Type=0x3, Address=0x8810000, Size=40
```

Driver Capabilities

The target driver needs to set capabilities in the **fas** driver in order to enable some driver features. The target driver can query and modify these capabilities: **synchronous**, **tagged-qing**, **wide-xfer**, **auto-rqsense**, **qfull-retries**, **qfull-retry-interval.** All other capabilities can only be queried.

By default, **tagged-qing**, **auto-rqsense**, and **wide-xfer** capabilities are disabled, while **disconnect**, **synchronous**, **untagged-qing** are enabled. These capabilities can only have binary values (0 or 1). The default values for **qfull-retries** and **qfull-retry-interval** are both 10. The **qfull-retries** capability is a u_char (0 to 255) while **qfull-retry-interval** is a u_short (0 to 65535).

The target driver needs to enable **tagged-qing** and **wide-xfer** explicitly. The **untagged-qing** capability is always enabled and its value cannot be modified, because **fas** can queue commands even when **tagged-qing** is disabled.

Whenever there is a conflict between the value of scsi-options and a capability, the value set in scsi-options prevails. Only whom != 0 is supported in the **scsi_ifsetcap**(9F) call.

Refer to **scsi_ifsetcap**(9F) and **scsi_ifgetcap**(9F) for details.

FILES

/kernel/drv/fas	ELF Kernel Module
/kernel/drv/fas.conf	Optional configuration file

SEE ALSO

prtconf(1M), **driver.conf**(4), **scsi_abort**(9F), **scsi_hba_attach**(9F), **scsi_ifgetcap**(9F), **scsi_ifsetcap**(9F), **scsi_reset**(9F), **scsi_sync_pkt**(9F), **scsi_transport**(9F), **scsi_device**(9S), **scsi_extended_sense**(9S), **scsi_inquiry**(9S), **scsi_pkt**(9S)

Writing Device Drivers

ANSI Small Computer System Interface-2 (SCSI-2)

FAS366 Technical Manuals, QLogic Corp.

DIAGNOSTICS

The messages described below are some that may appear on the system console, as well as being logged.

This first five messages may be displayed while the **fas** driver is trying to attach. All of these messages mean that the **fas** driver was unable to attach. These messages are preceded by "fas%d", where "%d" is the instance number of the **fas** controller.

Device in slave-only slot, unused
The SBus device has been placed in a slave-only slot and will not be accessible; move to non-slave-only SBus slot.

Device is using a hilevel intr
The device was configured with a interrupt level that cannot be used with this **fas** driver. Check the SBus device.

Unable to map FAS366 registers
Driver was unable to map device registers; check for bad hardware. Driver did not attach to device, SCSI devices will be inaccessible.

Cannot map dma
Driver was unable to locate a dma controller. This is an auto-configuration error.

Cannot attach
The driver was unable to attach, usually follows another warning that indicates why attach failed.

Disabled TQ since disconnects are disabled
Tagged Queuing was disabled because disconnects were disabled in scsi-options.

Bad clock frequency
Check for bad hardware.

Sync of pkt (%x) failed.
Syncing a scsi packet failed. Refer to **scsi_sync_pkt**(9F).

All tags in use!
The driver could not allocate another tag number. The target devices do not properly support Tagged Queuing.

Cannot alloc tag queue
The driver could not allocate space for tag queue.

Gross error in FAS366 status.
The driver experienced severe SCSI bus problems. Check cables and terminator.

Spurious interrupt
The driver received an interrupt while the hardware was not interrupting.

Lost state in phasemanage
The driver is confused about the state of the SCSI bus.

Unrecoverable DMA error during selection
The DMA controller experienced host SBus problems. Check for bad hardware.

Bad sequence step (0x%x) in selection
The FAS366 hardware reported a bad sequence step. Check for bad hardware.

Undetermined selection failure
The selection of a target failed unexpectedly. Check for bad hardware.

Target <n>: failed reselection (bad reselect bytes)
A reconnect failed, target sent incorrect number of message bytes. Check for bad hardware.

Target <n>: failed reselection (bad identify message)
A reconnect failed, target didn't sent identify message or it got corrupted. Check for bad hardware.

Target <n>: failed reselection (not in msgin phase)
Incorrect scsi bus phase after reconnection. Check for bad hardware.

Target <n>: failed reselection (unexpected bus free)
Incorrect scsi bus phase after reconnection. Check for bad hardware.

Target <n>: failed reselection (timeout on receiving tag msg)
A reconnect failed, target failed to send tag bytes. Check for bad hardware.

Target <n>: failed reselection (botched tag)
A reconnect failed, target failed to send tag bytes. Check for bad hardware.

Target <n>: failed reselection (invalid tag)
A reconnect failed, target sent incorrect tag bytes. Check for bad hardware.

Target <n>: failed reselection (Parity error in reconnect msg's)
A reconnect failed, parity error detected. Check for bad hardware.

Target <n>: failed reselection (no command)
A reconnect failed, target accepted abort or reset, but still tries to reconnect. Check for bad hardware.

Unexpected bus free
Target disconnected from the bus without notice. Check for bad hardware.

Target <n> didn't disconnect after sending <message>
The target unexpectedly did not disconnect after sending <message>.

Illegal dma boundary?
An attempt was made to cross a boundary that the driver could not handle.

Unwanted data xfer direction for Target <n>
The target went into an unexpected phase.

Spurious <name> phase from target <n>
The target went into an unexpected phase.

SCSI bus DATA IN phase parity error
The driver detected parity errors on the SCSI bus.

SCSI bus MESSAGE IN phase parity error
The driver detected parity errors on the SCSI bus.

SCSI bus STATUS phase parity error
The driver detected parity errors on the SCSI bus.

Premature end of extended message
An extended SCSI bus message did not complete. Suspect a target f/w problem.

Premature end of input message
A multibyte input message was truncated. Suspect a target f/w problem.

Input message botch
The driver is confused about messages coming from the target.

Extended message <n> is too long
The extended message send by the target is longer than expected.

<name> message <n> from Target <m> garbled
Target <m> send message <name> of value <n> which the driver did not understand.

Target <n> rejects our message <name>
Target <n> rejected a message send by the driver.

Rejecting message <name> from Target <n>
The driver rejected a message received from target <n>

Cmd transmission error
The driver was unable to send out command bytes.

Target <n> refused message resend
The target did not accept a message resend.

Two byte message <name> <value> rejected
The driver does not accept this two byte message.

Unexpected Selection Attempt
An attempt was made to select this host adapter by another initiator.

Polled cmd failed (target busy)
A polled cmd failed because the target did not complete outstanding commands within a reasonable time.

Polled cmd failed
A polled command failed because of timeouts or bus errors.

Disconnected command timeout for Target <id>.<lun>
A timeout occurred while target/lun was disconnected. This is usually a target f/w problem. For tagged queuing targets, <n> commands were outstanding when the timeout was detected.

Disconnected tagged cmds (<n>) timeout for Target <id>.<lun>
A timeout occurred while target/lun was disconnected. This is usually a target f/w problem. For tagged queuing targets, <n> commands were outstanding when the timeout was detected.

Connected command timeout for Target <id>.<lun>.
This is usually a SCSI bus problem. Check cables and termination.

Target <id>.<lun> reverting to async. mode
A data transfer hang was detected. The driver attempts to eliminate this problem by reducing the data transfer rate.

Target <id>.<lun> reducing sync. transfer rate
A data transfer hang was detected. The driver attempts to eliminate this problem by reducing the data transfer rate.

Reverting to slow SCSI cable mode
A data transfer hang was detected. The driver attempts to eliminate this problem by reducing the data transfer rate.

Reset scsi bus failed
An attempt to reset the SCSI bus failed.

External SCSI bus reset
Another initiator reset the SCSI bus.

WARNINGS

The **fas** hardware (FAS366) supports both wide and fast SCSI mode. The maximum SCSI bandwidth is 20 MB / sec. Initiator mode block sequence (IBS) is not supported.

NOTES

The **fas** driver exports properties indicating per target the negotiated transfer speed **(target<n>-sync-speed)**, whether wide bus is supported **(target<n>-wide)**, scsi-options for that particular target **(target<n>-scsi-options)**, and whether tagged queuing has been enabled **(target<n>-TQ)**. The sync-speed property value is the data transfer rate in KB / sec. The **target<n>-TQ** and the **target<n>-wide** property have values 1 to indicate that the corresponding capability is enabled, or 0 to indicate that the capability is disabled for that target. Refer to **prtconf**(1M) (verbose option) for viewing the **fas** properties.

```
SUNW,fas, instance #1
  Driver software properties:
    name <target3-TQ> length <4>
      value <0x00000001>.
    name <target3-wide> length <4>
      value <0x00000000>.
    name <target3-sync-speed> length <4>
      value <0x00002710>.
    name <target3-scsi-options> length <4>
      value <0x000003f8>.
    name <target0-TQ> length <4>
      value <0x00000001>.
    name <pm_norm_pwr> length <4>
      value <0x00000001>.
    name <pm_timestamp> length <4>
      value <0x30040346>.
    name <scsi-options> length <4>
      value <0x000003f8>.
    name <scsi-watchdog-tick> length <4>
```

```
    value <0x0000000a>.
  name <scsi-tag-age-limit> length <4>
    value <0x00000002>.
  name <scsi-reset-delay> length <4>
    value <0x00000bb8>.
Register Specifications:
  Bus Type=0x3, Address=0x8800000, Size=10
  Bus Type=0x3, Address=0x8810000, Size=40
Interrupt Specifications:
  Interrupt Priority=0x35 (ipl 5)
```

NAME

ffb – 24-bit UPA color frame buffer and graphics accelerator

DESCRIPTION

ffb is a 24-bit UPA-based color frame buffer and graphics accelerator which comes in two configurations.

The single buffered frame buffer consists of 32 video memory planes of 1280 × 1024 pixels, including 24-bit single-buffering and 8-bit X planes.

The double buffered frame buffer consists 96 video memory planes of 1280 × 1024 pixels, including 24-bit double-buffering, 8-bit X planes, 28-bit Z-buffer planes and 4-bit Y planes. The driver supports the following frame buffer ioctls which are defined in **fbio**(7I).

FBIOPUTCMAP, FBIOGETCMAP, FBIOSVIDEO, FBIOGVIDEO,
FBIOVERTICAL, FBIOSCURSOR, FBIOGCURSOR, FBIOSCURPOS, FBIOGCURPOS, FBIOGCURMAX, FBIO_WID_PUT, FBIO_WID_GET

However, **ffb** does not support **FBIOGTYPE** which is part of **fbio**(7I).**The**replacement**is VIS_GETIDENTIFIER.**

FILES

/dev/fbs/ffb0 device special file

SEE ALSO

ffbconfig(1M), **mmap**(2), **fbio**(7I)

NAME

hme – SUNW,hme Fast-Ethernet device driver

SYNOPSIS

/dev/hme

DESCRIPTION

The SUNW,hme Fast-Ethernet driver is a multi-threaded, loadable, clonable, STREAMS hardware driver supporting the connectionless Data Link Provider Interface, **dlpi**(7P), over a SUNW,hme Fast-Ethernet controller. The motherboard and add-in SBus SUNW,hme controllers of several varieties are supported. Multiple SUNW,hme controllers installed within the system are supported by the driver. The **hme** driver provides basic support for the SUNW,hme hardware. It is used to handle the "SUNW,hme" device. Functions include chip initialization, frame transit and receive, multicast and promiscuous support, and error recovery and reporting.

SUNW,hme

The **SUNW,hme** device provides 100Base-TX networking interfaces using SUN's **FEPS ASIC** and an Internal Transceiver. The FEPS ASIC provides the Sbus interface and MAC functions and the Physical layer functions are provided by the Internal Transceiver which connects to a **RJ-45** connector. In addition to the RJ-45 connector, an **MII** (Media Independent Interface) connector is also provided on all SUNW,hme devices except the SunSwith SBus adapter board. The MII interface is used to connect to an External Transceiver which may use any physical media (copper or fiber) specified in the 100Base-TX standard. When an External Transceiver is connected to the MII, the driver selects the External Transceiver and disables the Internal Transceiver.

The 100Base-TX standard specifies an "auto-negotiation" protocol to automatically select the mode and speed of operation. The Internal transceiver is capable of doing "auto-negotiation" with the remote-end of the link (Link Partner) and receives the capabilities of the remote end. It selects the **Highest Common Denominator** mode of operation based on the priorities. It also supports **forced-mode** of operation where the driver can select the mode of operation.

APPLICATION PROGRAMMING INTERFACE

The cloning character-special device **/dev/hme** is used to access all SUNW,hme controllers installed within the system.

hme and DLPI

The **hme** driver is a "style 2" Data Link Service provider. All **M_PROTO** and **M_PCPROTO** type messages are interpreted as **DLPI** primitives. Valid **DLPI** primitives are defined in **<sys/dlpi.h>.** Refer to **dlpi**(7P) for more information. An explicit **DL_ATTACH_REQ** message by the user is required to associate the opened stream with a particular device (**ppa**). The **ppa** ID is interpreted as an **unsigned long** data type and indicates the corresponding device instance (unit) number. An error (**DL_ERROR_ACK**) is returned by the driver if the **ppa** field value does not correspond to a valid device instance number for this system. The device is initialized on first attach and de-initialized (stopped) at last detach.

The values returned by the driver in the **DL_INFO_ACK** primitive in response to the **DL_INFO_REQ** from the user are as follows:

- The maximum SDU is **1500** (**ETHERMTU** - defined in <sys/ethernet.h>).
- The minimum SDU is **0**.

- The **dlsap** address length is **8.**
- The MAC type is **DL_ETHER**.
- The **sap** length values is **–2** meaning the physical address component is followed immediately by a 2 byte **sap** component within the DLSAP address.
- The service mode is **DL_CLDLS**.
- No optional quality of service (QOS) support is included at present so the QOS fields are **0**.
- The provider style is **DL_STYLE2**.
- The version is **DL_VERSION_2**.
- The broadcast address value is Ethernet / IEEE broadcast address (**0xFFFFFF**).

Once in the **DL_ATTACHED** state, the user must send a **DL_BIND_REQ** to associate a particular SAP (Service Access Pointer) with the stream. The **hme** driver interprets the **sap** field within the **DL_BIND_REQ** as an Ethernet "type" therefore valid values for the **sap** field are in the [**0-0xFFFF**] range. Only one Ethernet type can be bound to the stream at any time.

If the user selects a **sap** with a value of **0**, the receiver will be in "802.3 mode". All frames received from the media having a "type" field in the range [**0-1500**] are assumed to be 802.3 frames and are routed up all open Streams which are bound to **sap** value **0**. If more than one Stream is in "802.3 mode" then the frame will be duplicated and routed up multiple Streams as **DL_UNITDATA_IND** messages.

In transmission, the driver checks the **sap** field of the **DL_BIND_REQ** if the **sap** value is **0**, and if the destination type field is in the range [**0-1500**]. If either is true, the driver computes the length of the message, not including initial **M_PROTO** mblk (message block), of all subsequent **DL_UNITDATA_REQ** messages and transmits 802.3 frames that have this value in the MAC frame header length field.

The **hme** driver **DLSAP** address format consists of the 6 byte physical (Ethernet) address component followed immediately by the 2 byte **sap** (type) component producing an 8 byte **DLSAP** address. Applications should *not* hardcode to this particular implementation-specific **DLSAP** address format but use information returned in the **DL_INFO_ACK** primitive to compose and decompose **DLSAP** addresses. The **sap** length, full **DLSAP** length, and **sap** / physical ordering are included within the **DL_INFO_ACK**. The physical address length can be computed by subtracting the **sap** length from the full **DLSAP** address length or by issuing the **DL_PHYS_ADDR_REQ** to obtain the current physical address associated with the stream.

Once in the **DL_BOUND** state, the user may transmit frames on the Ethernet by sending **DL_UNITDATA_REQ** messages to the **hme** driver. The **hme** driver will route received Ethernet frames up all those open and bound streams having a **sap** which matches the Ethernet type as **DL_UNITDATA_IND** messages. Received Ethernet frames are duplicated and routed up multiple open streams if necessary. The **DLSAP** address contained within the **DL_UNITDATA_REQ** and **DL_UNITDATA_IND** messages consists of both the **sap** (type) and physical (Ethernet) components.

In addition to the mandatory connectionless **DLPI** message set the driver additionally supports the following primitives.

hme Primitives

The **DL_ENABMULTI_REQ** and **DL_DISABMULTI_REQ** primitives enable / disable reception of individual multicast group addresses. A set of multicast addresses may be iteratively created and modified on a per-stream basis using these primitives. These primitives are accepted by the driver in any state following **DL_ATTACHED**.

The **DL_PROMISCON_REQ** and **DL_PROMISCOFF_REQ** primitives with the **DL_PROMISC_PHYS** flag set in the **dl_level** field enables / disables reception of all ("promiscuous mode") frames on the media including frames generated by the local host. When used with the **DL_PROMISC_SAP** flag set this enables / disables reception of all **sap** (Ethernet type) values. When used with the **DL_PROMISC_MULTI** flag set this enables / disables reception of all multicast group addresses. The effect of each is always on a per-stream basis and independent of the other **sap** and physical level configurations on this stream or other streams.

The **DL_PHYS_ADDR_REQ** primitive returns the 6 octet Ethernet address currently associated (attached) to the stream in the **DL_PHYS_ADDR_ACK** primitive. This primitive is valid only in states following a successful **DL_ATTACH_REQ**.

The **DL_SET_PHYS_ADDR_REQ** primitive changes the 6 octet Ethernet address currently associated (attached) to this stream. The credentials of the process which originally opened this stream must be superuser. Otherwise **EPERM** is returned in the **DL_ERROR_ACK**. This primitive is destructive in that it affects all other current and future streams attached to this device. An **M_ERROR** is sent up all other streams attached to this device when this primitive is successful on this stream. Once changed, all streams subsequently opened and attached to this device will obtain this new physical address. Once changed, the physical address will remain until this primitive is used to change the physical address again or the system is rebooted, whichever comes first.

hme DRIVER

By default, the hme driver performs "auto-negotiation" to select the **mode** and **speed** of the link, when the Internal Transceiver is used.

When an External Transceiver is connected to the **MII** interface, the driver selects the External Transceiver for networking operations. If the External Transceiver supports "auto-negotiation", the driver uses the auto-negotiation procedure to select the link speed and mode. If the External Transceiver does not support auto-negotiation, it will select the highest priority mode supported by the transceiver.

The link can be in one of the *4* following modes:

- 100 Mbps, full-duplex
- 100 Mbps, half-duplex
- 10 Mbps, full-duplex
- 10 Mbps, half-duplex

These speeds and modes are described in the 100Base-TX standard.

The *auto–negotiation* protocol automatically selects:

- Operation mode (half-duplex or full-duplex)
- Speed (100 Mbps or 10 Mbps)

The auto–negotiation protocol does the following:

- Gets all the modes of operation supported by the Link Partner
- Advertises its capabilities to the Link Partner
- Selects the highest common denominator mode of operation based on the priorities

The *internal transceiver* is capable of all of the operating speeds and modes listed above. When the internal transceiver is used, by *default*, auto-negotiation is used to select the speed and the mode of the link and the common mode of operation with the Link Partner.

When an *external transceiver* is connected to the **MII** interface, the driver selects the external transceiver for networking operations. If the external transceiver supports auto-negotiation:

- The driver uses the auto-negotiation procedure to select the link speed and mode.

If the external transceiver *does not* support auto-negotiation

- The driver selects the highest priority mode supported by the transceiver.

Sometimes, the user may want to select the speed and mode of the link. The **SUNW,hme** device supports programmable **"IPG"** (Inter-Packet Gap) parameters **ipg1** and **ipg2**. By default, the driver sets **ipg1** to 8 **byte-times** and **ipg2** to 4 **byte-times** (which are the standard values). Sometimes, the user may want to alter these values depending on whether the driver supports 10 Mbps or 100 Mpbs and accordingly, **IPG** will be set to 9.6 or 0.96 microseconds.

hme Parameter List

The hme driver provides for setting and getting various parameters for the **SUNW,hme** device. The parameter list includes **current transceiver status**, **current link status**, **inter-packet gap**, **local transceiver capabilities** and **link partner capabilities**.

The local transceiver has two set of capabilities: one set reflects the capabilities of the **hardware**, which are **read-only (RO)** parameters and the second set reflects the values chosen by the user and is used in **speed selection**. There are **read/write (RW)** capabilities. At boot time, these two sets of capabilities will be the same. The Link Partner capabilities are also read only parameters because the current default value of these parameters can only be read and cannot be modified.

FILES

/dev/hme	**hme** special character device.
/kernel/drv/hme	System wide default device driver properties

SEE ALSO

ndd(1M), **netstat**(1M), **driver.conf**(4), **dlpi**(7P), **ie**(7D), **le**(7D)

NAME

spcic – TI-1130 PC Card Interface Controller

DESCRIPTION

The TI-1130 PC Card Interface Controller provides one or more PCMCIA PC Card sockets. The **spcic** adapter driver provides an interface between the PCMCIA sockets and the PCMCIA nexus.

The driver supports the TI-1130 chip (only in R2 mode).

Direct access to the PCMCIA hardware is not supported. The driver exists solely to support the PCMCIA nexus.

FILES

/kernel/drv/spcic **spcic** driver

SEE ALSO

pcmcia(4)

Index