Solaris™ on Sun™ Hardware Reference Manual Supplement

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

Part No.:806-3201-10
November 1999, Revision A

Send comments about this document to:
docfeedback@sun.com

Please
Recycle

Contents

Preface v

1. System Administration Commands (1M) 1M-1

`afbconfig` 1M-1

`GFXconfig` 1M-9

`hsi_init` 1M-13

`hsi_loop` 1M-16

`hsi_stat` 1M-19

`hsi_trace` 1M-22

`hsip_init` 1M-25

`hsip_loop` 1M-28

`hsip_stat` 1M-31

`nf_fddidaemon` 1M-34

`nf_install_agents` 1M-35

`nf_macid` 1M-36

`nf_smtmon` 1M-37

`nf_snmd` 1M-39

`nf_snmd_kill` 1M-41

`nf_stat` 1M-42

`nf_sync` 1M-45

`pf_fddidaemon` 1M-46

pf_install_agents 1M-47

pf_macid 1M-48

pf_smtmon 1M-49

pf_snmd 1M-51

pf_snmd_kill 1M-53

pf_stat 1M-54

rscadm 1M-57

sunvts 1M-61

vtsk 1M-62

vtsprobe 1M-63

vtstty 1M-66

vtsui 1M-68

vtsui.ol 1M-69

2. Device and Network Interfaces (7) 7-70

afb 7-70

ge 7-71

hsi 7-75

hsip 7-79

nf 7-83

pf 7-86

smt 7-91

Preface

The *Solaris on Sun Hardware Reference Manual Supplement* contains manual pages (man pages) for software provided to Sun hardware customers with the Solaris 7 product. These supplement the man pages provided in the general *Solaris 7 Reference Manual*.

Before you can access some of the information published in this book through the `man` command, you may need to install software from the Sun Microsystems Computer Systems Supplement CD for your Solaris release. In most cases, when you install a software cluster from the Sun Microsystems Computer Systems Supplement CD, man pages about the software in that cluster will be automatically installed. For information about installing the man page software, refer to the *Solaris 7 Sun Hardware Platform Guide.*

How This Book Is Organized

This manual contains man pages in alphabetical order within each man page category. Supplemental man pages are included for the following categories:

- System Administration Commands (1M)
- Device and Network Interfaces (7)

The man pages apply to the following products:

- AFB graphics accelerator: `afb`, `afbconfig`
- SunFDDI network adapter software: `nf`, `nf_fddidaemon`, `nf_install_agents`, `nf_macid`, `nf_smtmon`, `nf_snmd`, `nf_snmd_kill`, `nf_stat`, `nf_sync`, `pf`, `pf_fddidaemon`, `pf_install_agents`, `pf_macid`, `pf_smtmon`, `pf_snmd`, `pf_snmd_kill`, `pf_stat`, `smt`
- Sun HSI/P (PCI bus) network adapter software: `hsip`, `hsip_init`, `hsip_loop`, `hsip_stat`

- Sun HSI/S (Sbus) network adapter software: `hsi`, `hsi_init`, `hsi_loop`, `hsi_stat`, `hsi_trace`
- PGX32 frame buffer: `GFXconfig`
- Sun Remote System Control (RSC): `rscadm`
- SunVTS diagnostic software: `sunvts`, `vtsk`, `vtsprobe`, `vtstty`, `vtsui`, `vtsui.ol`

Sun Documentation on the Web

The `docs.sun.com` web site enables you to access Sun technical documentation on the World Wide Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at `http://docs.sun.com`.

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at `docfeedback@sun.com`. Please include the part number of your document in the subject line of your email.

NAME

afbconfig − configure the AFB Graphics Accelerator

SYNOPSIS

/usr/sbin/afbconfig [−**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*]
[−**extovl enable** | **disable**]
[−**g** *gamma-correction-value*]
[−**gfile** *gamma-correction-file*]
[−**propt**] [−**prconf**] [−**defaults**]

/usr/sbin/afbconfig [−**propt**] [−**prconf**]

/usr/sbin/afbconfig [−**help**] [−**res** ?]

AVAILABILITY

SUNWafbcf

DESCRIPTION

afbconfig configures the AFB Graphics Accelerator and some of the X11 window system defaults for AFB.

The first form of **afbconfig** shown in the synopsis above stores the specified options in the OWconfig file. These options will be used to initialize the AFB 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 AFB device at a time. Specifying options for multiple AFB devices requires multiple invocations of **afbconfig**.

Only AFB-specific options can be specified through **afbconfig**. 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 **afbconfig** program, which is owned by the root user, always runs with setuid root permission.

OPTIONS

−**dev** *device-filename*
Specifies the AFB special file. The default is **/dev/fbs/afb0**.

−**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 AFB 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, supported by AFB, may not be supported by the monitor. The list of video-modes supported by the AFB device and the monitor can be obtained by running **afbconfig** with the −**res** ? option (the third form shown in the command synopsis above). A list of all possible video-modes supported on AFB is shown below.

1024x768x60
1024x768x70
1024x768x75
1024x768x77
1024x800x84
1152x900x66
1152x900x76
1280x800x76
1280x1024x60
1280x1024x67
1280x1024x76
960x680x112s (Stereo)
960x680x108s (Stereo)
640x480x60
640x480x60i (Interlaced)
768x575x50i (Interlaced)

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 AFB 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 **afbconfig** while the configured device is being used (e.g. while running the window system); unpredictable results may occur. To run **afbconfig** 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 **afbconfig** 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 **afbconfig** to bypass this confirmation and to program the requested video mode anyway. This option is useful when **afbconfig** 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 AFB 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 **afbconfig** 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).

AFB 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 AFB doesn't possess a linear overlay visual.

−**defoverlay true | false**

The AFB provides an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the AFB 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. The number of colors available to the windows created using this visual depends on the settings for the extovl option. If the extovl is enabled, extended overlay with 256 opaque color values is available. (refer to the −**extovl** option). If extovl is disabled, extended overlay is not available and 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 AFB 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 AFB 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 AFB 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*
This option is available only if extovl is disabled. It specifies the maximum number of AFB 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, 64

−**extovl** enable | disable
If enabled, extended overlay is available. The overlay visuals will have 256 opaque colors. The SOV visuals will have 255 opaque colors and 1 transparent color. Also, this option enables hardware supported transparency, thus provides better performance for windows using the SOV visuals.

−**g** *gamma-correction value*
This option allows changing the gamma correction value. All linear visuals provide gamma correction. By default the gamma correction value is 2.22. Any value less than zero is illegal.
This option can be used while the window system is running. Changing the gamma correction value will affect all the windows being displayed using the

linear visuals.

−**gfile** *gamma-correction file*

This option loads gamma correction table from the specified file. This file should be formatted to provide the gamma correction values for R, G and B channels on each line. Each of these values should be in hexadecimal format and seperated from each other by atleast 1 space. Also this file should provide 256 such triplets. An example of this file is as follows.

0x00 0x00 0x00
0x01 0x01 0x01
0x02 0x02 0x02
...
...
0xff 0xff 0xff

Using this option, the gamma correction table can be loaded while the window system is running. The new gamma correction will affect all the windows being displayed using the linear visuals. Note, when gamma correction is being done using user specified table, the gamma correction value is undefined.

By default, the window system assumes a gamma correction value of 2.22 and loads the gamma table it creates correspoding to this value.

−**defaults**

Resets all option values to their default values.

−**propt** Prints the current values of all AFB 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 **afbconfig** completes. This is a typical display:

--- OpenWindows Configuration for /dev/fbs/afb0 ---
OWconfig: machine
Video Mode: 1280x1024x76
Default Visual: Non-Linear Normal Visual
Visual Ordering: Linear Visuals are last
Overlay Visuals are last
OpenGL Visual Expansion: enabled
Server Overlay Visuals: enabled
Extended Overlay: enabled
Underlay WIDs: 64 (not configurable)
Overlay WIDs: 4 (not configurable)
Gamma Correction Value: 2.220000
Gamma Correction Table: Available

−**prconf**

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

--- Hardware Configuration for /dev/fbs/afb0 ---

Type: double-buffered AFB with Z-buffer
Board: rev 0 (Horizontal)
Number of Floats: 6
PROM Information: @(#)afb.fth x.xx xx/xx/xx
AFB ID: 0x101df06d
DAC: Brooktree 9070, version 1 (Pac2)
3DRAM: Mitsubishi 130a, version x
EDID Data: Available - EDID version 1 revision x
Monitor Sense ID: 4 (Sun 37x29cm RGB color monitor)
Monitor possible resolutions: 1024x768x77, 1024x800x84, 1152x900x66, 1152x900x76, 1280x1024x67, 1280x1024x76, 960x680x112s, 960x680x108s
Current resolution setting: 1280x1024x76

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

DEFAULTS

For a given invocation of **afbconfig** 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 AFB option has never been specified via **afbconfig**, a default value is used. The option defaults are as follows:

Option	Default
−dev	/dev/fbs/afb0
−file	machine
−res	none
−deflinear	false
−defoverlay	false
−linearorder	last
−overlayorder	last
−expvis	enabled
−sov	enabled
−maxwids	32
−extovl	enabled
−g	2.22

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 AFB video mode.

EXAMPLES

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

```
example% /usr/sbin/afbconfig −res 1280x1024x76
```

FILES

/dev/fbs/afb0	device special file

SEE ALSO

mmap(2), **fbio**(7I), **afb**(7D)

NAME

GFXconfig − configure the Raptor GFX Graphics Accelerator

SYNOPSIS

/usr/sbin/GFXconfig [−**dev** *device-filename*]
[−**res** *video-mode* [**try** | **noconfirm** | **nocheck**]]
[−**file machine** | **system**]
[−**depth** 8|24]
[−**defaults**]

/usr/sbin/GFXconfig [−**propt**] [−**prconf**]

/usr/sbin/GFXconfig [−**help**] [−**res** ?]

/usr/sbin/GFXconfig [−**vpd**]

DESCRIPTION

GFXconfig configures the Raptor GFX Graphics Accelerator and some of the X11 window system defaults for Raptor GFX.

The first form of **GFXconfig** shown in the synopsis above stores the specified options in the OWconfig file. These options will be used to initialize the Raptor GFX 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 and fourth forms which invoke only the −**prconf**, −**propt**, −**help**, −**res**? and −**vpd** options do not update the OWconfig file. Additionally, for the third form all other options are ignored.

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

Only Raptor GFX-specific options can be specified through **GFXconfig**. 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 **GFXconfig** program, which is owned by the root user, always runs with **setuid** root permission.

OPTIONS

−**dev** *device-filename*
Specifies the Raptor GFX special file. The default is **/dev/fbs/gfxp0,** or **/dev/fbs/raptor0** if applicable.

−**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* [**try** | **noconfirm** | **nocheck**]

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

The format of these built-in video modes is:

widthxheightxrate

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. As a convenience, -res also accepts formats with @ in front of the refresh rate instead of x. For example: 1280x1024@76. The list can be obtained by running GFXconfig 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. GFXconfig 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.

Symbolic names

For convenience, some of the video modes have symbolic names defined for them. Instead of the form widthxheightxrate, 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**
vga	**640x480x60**
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.

noconfirm

Using the −**res** option,the user could potentially put the system into an unusable 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 GFXconfig 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 **GFXconfig** bypass this confirmation and to program the requested video mode anyway. This option is useful when **GFXconfig** 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 Raptor GFX device). Use of this option implies noconfirm as 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 pattern is displayed. If any character other than prompted for a yes or no answer on whether to keep the new video mode.

−defaults
Resets all option values to their default values.

−propt Prints the current values of all Raptor GFX 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 would be in the OWconfig file after the call to GFXconfig would have completed. This is a typical display:

--- OpenWindows Configuration for /dev/fbs/gfxp0 ---
OWconfig: machine
Video Mode: not set

−prconf
Prints the Raptor GFX hardware configuration. This is a typi- cal display:

--- Hardware Configuration for /dev/fbs/gfxp0 ---
DAC: version 0x0
Type:
Board:
PROM: version 0x0
PROM Information:
RAM:
EDID Data:
Monitor Sense ID:
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
Possible depths:
Current depth: 8

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

−**vpd** Prints a list of the card's Vital Product Data information including serial number and Prom Revision level.

DEFAULTS

For a given invocation of **GFXconfig,** if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value, except for **'−depth'** and **'−24only'.**

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

Option	Default
−**dev**	**/dev/fbs/gfxp0**
−**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 specify- ing the video mode. This means that the PROM ultimately determines the default Raptor GFX video mode.

EXAMPLES

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

example% /usr/sbin/GFXconfig -res 1280x1024x76

FILES

/dev/fbs/gfxp0	device special file
/usr/openwin/server/etc/OWconfig	System config file
/etc/openwin/server/etc/OWconfig	Machine config file

NAME

hsi_init − set high speed serial line interface operating parameters.

SYNOPSIS

/opt/SUNWconn/bin/hsi_init *device* **[[** *baud_rate* **] | [** *keyword=value***, ...] | [** *single-word option* **]]**

DESCRIPTION

The hsi_init utility allows the user to modify some of the hardware operating modes common to high speed synchronous serial lines. This may be useful in troubleshooting a link, or necessary to the operation of a communications package.

If run without options, hsi_init reports the options as presently set on the port. If options are specified, the new settings are reported after they have been made.

OPTIONS

Options to hsi_init normally take the form of a keyword, followed by an equal sign and a value. The exception is that a baud rate may be specified as a decimal integer by itself. Keywords must begin with the value shown in the options table, but may contain additional letters up to the equal sign. For example, "loop=" and "loopback=" are equivalent.

Recognized options are listed in the table below.

Keyword	Value	Effect
loopback	no	Disable internal loopback mode. If no other clocking options have been specified, perform the equivalent of **txc=txc** and **rxc=rxc**.
	yes	Set the port to operate in **internal loopback** mode. The receiver is electrically disconnected from the DCE receive data input and tied to the outgoing transmit data line. Transmit data is available to the DCE. If no other clocking options have been specified, perform the equivalent of **txc=baud** and **rxc=baud**.
nrzi	no	Set the port to operate with **NRZ** data encoding. NRZ encoding maintains a constant voltage level when data is present (1) and does not return to a zero voltage (0) until data is absent. The data is decoded as an absolute value based on the voltage level (0 or 1).
	yes	Set the port to operate with **NRZI** data encoding. **NRZI** encoding does a voltage transition when data is absent (0) and no voltage transition (no return to zero) when data is present (1). Hence, the name non-return to zero inverted. The data is decoded using relational decoding.
txc	txc	Transmit clock source will be the **TxCI** signal.
	-txc	Transmit clock source will be the inverted **TxCI** signal.
	rxc	Transmit clock source will be the **RxC** signal.
	baud	Transmit clock source will be the internal **baud rate generator**.

rxc	rxc	Receive clock source will be the **RxC** signal.
	-rxc	Receive clock source will be the inverted **RxC** signal.
	baud	Receive clock source will be the internal **baud rate generator**.
mode	fdx	HDLC Full Duplex mode (Default mode).
	ibm-fdx	IBM Full Duplex mode (SDLC).
	ibm-hdx	IBM Half Duplex mode (SDLC).
	ibm-mpt	IBM Multipoint mode (SDLC).
signal	yes	Notify application of modem signal (RTS and CTS) changes.
	no	Don't notify application of modem signal (RTS and CTS) changes.
speed	*integer*	Set the baud rate to *integer* bits per second. The speed can be set from 300 bps to 2048000 bps.
mtu		Set the Maximum Transmission Unit. This is the packet size that is transmitted. The maximum mtu is 1600 bytes.
mru		Set the Maximum Receive Unit. This is the packet size that is received. The maximum mru is 1600 bytes.
txd		This flags is used for inverting transmit data on serial lines. You can switch the polarity of a link by setting this flag to be negative, i.e. -txd.
rxd		This flags is used for inverting receive data on serial lines. You can switch the polarity of a link by setting this flag to be negative, i.e. -rxd.
reset		Resets the board. Terminates all incoming and outgoing traffic.

There are also several single-word options that set one or more paramaters at a time:

Keyword	**Equivalent to Options:**
external	txc=txc rxc=rxc loop=no
sender	txc=baud rxc=rxc loop=no
stop	speed=0

EXAMPLES

The following command sets the first CPU port to loop internally, use internal clocking and operate at 38400 bps:

example# hsi_init hih0 38400 loop=yes
port=hih0 speed=38309, mode=fdx, loopback=yes, nrzi=no, mtu=1600, mru=1600, txc=baud, rxc=baud, txd=txd, rxd=rxd, signal=no.

The following command sets the same port's clocking, local loopback and bit rate settings to their default values:

example# hsi_init hih0 1536000 loop=no
port=hih0 speed=1536000, mode=fdx, loopback=no, nrzi=no, mtu=1600, mru=1600, txc=txc, rxc=rxc, txd=txd, rxd=rxd, signal=no.

SEE ALSO

hsi_loop(1M), hsi_stat(1M), hsi_trace(1M), Intro(2), hsi(7D)

DIAGNOSTICS

device **missing minor device number**
The name *device* does not end in a decimal number that can be used as a minor device number.

bad speed: *arg*
The string *arg* that accompanied the "speed=" option could not be interpreted as a decimal integer.

Bad arg: *arg*
The string *arg* did not make sense as an option.

ioctl failure code = *errno*
An **ioctl(2)** system called failed. The meaning of the value of *errno* may be found in the **Intro(2)** manual page.

WARNINGS

hsi_init should not be used on an active serial link, unless needed to resolve an error condition. It should not be run casually, or if the user is unsure of the consequences of its use.

NAME

hsi_loop − high speed synchronous serial loopback test program for high speed serial interface.

SYNOPSIS

/opt/SUNWconn/bin/hsi_loop [−cdlsvt] *device*

DESCRIPTION

The hsi_loop command performs several loopback tests that are useful in exercising the various components of a serial communications link.

Before running a test, hsi_loop opens the designated port and configures it according to command line options and the specified test type. It announces the names of the devices being used to control the hardware channel, the channel number (ppa) corresponding to the *device* argument, and the parameters it has set for that channel. It then runs the loopback test in three phases.

The first phase is to listen on the port for any activity. If no activity is seen for at least four seconds, hsi_loop proceeds to the next phase. Otherwise, the user is informed that the line is active and that the test cannot proceed, and the program exits.

In the second phase, called the "first-packet" phase, hsi_loop attempts to send and receive one packet. The program will wait for up to four seconds for the returned packet. If no packets are seen after five attempts, the test fails with an error message. If a packet is returned, the result is compared with the original. If the length and content do not match exactly, the test fails.

The final phase, known as the "multiple-packet" phase, attempts to send many packets through the loop. Because the program has verified the integrity of the link in the first-packet phase, the test will not fail after a particular number of timeouts. If a packet is not seen after four seconds, a message is displayed. Otherwise, a count of the number of packets received is updated on the display once per second. If it becomes obvious that the test is not receiving packets during this phase, the user may wish to stop the program manually. The number and size of the packets sent during this phase is determined by default values, or by command line options. Each returned packet is compared with its original for length and content. If a mismatch is detected, the test fails. The test completes when the required number of packets have been sent, regardless of errors.

After the multiple-packet phase has completed, the program displays a summary of the hardware event statistics for the channel that was tested. The display takes the following form:

Port	**CRC errors**	**Aborts**	**Overruns**	**Underruns**	In <-Drops-> Out	
hih0	**0**	**0**	**0**	**0**	0	0

This is followed by an estimated line speed, which is an approximation of the bit rate of the line, based on the number of bytes sent and the actual time that it took to send them. This is a very rough approximation and should not be used in bechmarking, because elapsed time includes time to print to the display.

OPTIONS

The options for hsi_loop are described in the following table:

Option	Parameter	Default	Description
−c	*packet_count*	100	Specifies the number of packets to be sent in the multiple-packet phase.
−d	*hex_data_byte*	*random*	Specifies that each packet will be filled with bytes with the value of *hex_data_byte*.
−l	*packet_length*	100	Specifies the length of each packet in bytes.
−s	*line_speed*	9600	Bit rate in bits per second.
−v			Sets verbose mode. If data errors occur, the expected and received data is displayed.
−t	*test_type*	*none*	A number, from 1 to 4, that specifies which test to perform. The values for *test_type* are as follows:

1 Internal loopback test. Port loopback is on. Transmit and receive clock sources are internal (baud rate generator).

2 External loopback test. Port loopback is off. Transmit and receive clock sources are internal. Requires a loopback plug suitable to the port under test.

3 External loopback test. Port loopback is off. Transmit and receive clock sources are external (modem). Requires that one of the local modem, the remote modem, or the remote system (not a Sun) be set in a loopback configuration.

4 Test using predefined parameters. User defines hardware configuration and may select port parameters using the **hsi_init(1M)** command.

All numeric options except **−d** are entered as decimal numbers (for example, **−s 19200**). If you do not provide the **−t** *test_type* option, hsi_loop prompts for it.

EXAMPLES

The following command causes hsi_loop to use a packet length of 512 bytes over the first CPU port:

example# hsi_loop −l 512 hih0

In response to the above command, hsi_loop prompts you for the test option you want.

The following command performs an internal loopback test on the first CPU port, using 5000 packets and a bit rate of 56Kbps :

example# hsi_loop −t 1 −s 56000 −c 5000 hih0

SEE ALSO

hsi_init(1M), hsi_stat(1M), hsi_trace(1M), hsi(7d)

DIAGNOSTICS

device **missing minor device number**
The name *device* does not end in a decimal number that can be used as a minor device number.

invalid packet length: *nnn*
The packet length was specified to be less than zero or greater than 1600.

poll: nothing to read
poll: nothing to read or write.
The **poll(2)** system call indicates that there is no input pending and/or that output would be blocked if attempted.

len *xxx* **should be** *yyy*
The packet that was sent had a length of *yyy*, but was received with a length of *xxx*.

nnn **packets lost in outbound queueing**
nnn **packets lost in inbound queueing**
A discrepancy has been found between the number of packets sent by hsi_loop and the number of packets the driver counted as transmitted, or between the number counted as received and the number read by the program.

WARNINGS

To allow its tests to run properly, as well as prevent disturbance of normal operations, hsi_loop should only be run on a port that is not being used for any other purpose at that time.

NAME

hsi_stat − report driver statistics from a high speed synchronous serial link port.

SYNOPSIS

/opt/SUNWconn/bin/hsi_stat [-f] -a|*num_of_ports*
/opt/SUNWconn/bin/hsi_stat -c [-f] -a|*num_of_ports*
/opt/SUNWconn/bin/hsi_stat [-f] *device* **[***period***]**
/opt/SUNWconn/bin/hsi_stat -c [-f] *device*

DESCRIPTION

The hsi_stat command reports the event statistics maintained by a high speed synchronous serial device driver. The report may be a single snapshot of the accumulated totals, or a series of samples showing incremental changes.

Event statistics are maintained by a driver for each physical channel that it supports. They are initialized to zero at the time the driver module is loaded into the system when one of the driver's entry points is first called.

The **device** argument is the name of the high speed serial device as it appears in the /dev directory. For example, **hih0** specifies the first on-board high speed serial device.

As an alternative, you can display or clear the statistics for multiple physical channels using **num_of_ports** argument. The hsi_stat program will then display statistics accumulated from device **hih0** to **hih(num_of_ports - 1)**. Additionally, statistics for all ports can be displayed or cleared by the use of the -a option. In this case, the command will be issued for all the ports on the system. This option is not available for sampling purposes.

The following is a breakdown of hsi_stat output:

speed The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.

ipkts The total number of input packets.

opkts The total number of output packets.

undrun The number of transmitter underrun errors.

ovrrun The number of receiver overrun errors.

abort The number of aborted received frames.

crc The number of received frames with CRC errors.

isize The average size (in bytes) of input packets.

osize The average size (in bytes) of output packets.

iutil Reports the input line utilization expressed as a percentage.

outil Reports the output line utilization expressed as a percentage.

Additional fields for the 'f' flag are listed below.

ierror Reports the input error count. Errors can be incomplete frames, empty frames, or receive clock (RxC) problems.

inactiv Reports the number of input packets received when receive is inactive.

ishort Reports the number of short input packets. This is the number of input packets with lengths less than the number of CRC bytes.

ilong Reports the number of long input packets. This is the number of input packets with lengths larger than the MRU.

oerror Reports the output error count. Errors that can be lost are clear to send (CTS) signals or transmit clock (TxC) problems.

olong Reports the number of long output packets. This is the number of output packets with lengths with lengths larger than the MTU.

ohung Reports the number of times the transmitter hangs, which is usually due to a missing clock.

OPTIONS

−f Select full set of accumulated statistics for the device specified. This is useful while debugging the **hsi** driver.

−c Clear the accumulated statistics for the device specified. This may be useful when it is not desirable to unload a particular driver, or when the driver is not capable of being unloaded.

num_of_ports
Specify the number of devices that you want to dump the statistics.

−a Specify all of the ports in the system, regardless of the number of HSI boards.

interval Cause hsi_stat to sample the statistics every *interval* seconds and report incremental changes. The output reports line utilization for input and output in place of average packet sizes. These are the relationships between bytes transferred and the baud rate, expressed as percentages. The loop repeats indefinitely, with a column heading printed every twenty lines for convenience.

EXAMPLES

example# **hsi_stat hih0**

speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize
9600	15716	17121	0	0	1	3	98

example# **hsi_stat 5**

speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize
hih0	9600	15716	10100	0	0	1	3
hih1	9600	15234	20100	0	0	1	3
hih2	9600	15123	18254	0	0	1	3
hih3	9600	15378	18234	0	0	1	3

example# **hsi_stat -a**

speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize	osize

hih0	9600	15716	10100	0	0	1	3	98
hih1	9600	15234	20100	0	0	1	3	98
hih2	9600	15123	18254	0	0	1	3	98
hih3	9600	15378	18234	0	0	1	3	98
hih4	9600	13900	13000	0	0	1	3	98
hih5	9600	15218	13100	0	0	1	3	98
hih6	9600	15737	22100	0	0	1	3	98
hih7	9600	15143	11254	0	0	1	3	98

example# **hsi_stat -c hih0**

speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize	osize
9600	0	0	0	0	0	0	0	0

example# **hsi_stat hih0 5**

ipkts	opkts	undrun	ovrrun	abort	crc	iutil	outil
12	10	0	0	0	0	5%	4%
22	60	0	0	0	0	3%	90%
36	14	0	0	0	1	51%	2%

(In this final example a new line of output is generated every five seconds.)

SEE ALSO **hsi_init(1M), hsi_loop(1M), hsi_trace(1M), hsi(7D)**

DIAGNOSTICS

device **missing minor device number**
The name *device* does not end in a decimal number that can be used as a minor device number.

hsi_stat: Can't sample multiple ports simultaneously.
Sampling is only available with one specified port, i.e. hsi_stat hih0 10.

WARNINGS

Underrun, overrun, frame-abort and CRC errors have a variety of causes. Communication protocols are typically able to handle such errors and initiate recovery of the transmission in which the error occurred. Small numbers of such errors are not a significant problem for most protocols. However, because the overhead involved in recovering from a link error can be much greater than that of normal operation, high error rates can greatly degrade overall link throughput. High error rates are often caused by problems in the link hardware, such as cables, connectors, interface electronics or telephone lines. They may also be related to excessive load on the link or the supporting system.

The percentages for input and output line utilization reported when using the *interval* option may occasionally be reported as slightly greater than 100% because of inexact sampling times and differences in the accuracy between the system clock and the modem clock. If the percentage of use greatly exceeds 100%, or never exceeds 50%, then the baud rate set for the device probably does not reflect the speed of the modem.

NAME

hsi_trace − Dump and Parse the HSI/S driver trace buffer. This is a development/field support only diagnostic utility.

SYNOPSIS

/opt/SUNWconn/bin/hsi_trace

DESCRIPTION

hsi_trace utility id for support and field personnel only. This utility prints out the trace of the incoming and outgoing packets at the hsi driver level.

There are two levels of traces that can be captured. This is controlled by setting a variable in the driver in the /etc/system file.

set HSI:hsi_trace=1

The driver maintains an internal circular buffer to store 24K frames (both in and out).

Then run hsi_trace on the driver to collect the trace data.

hsi_trace > hsi_trace.log

This trace is useful when the problem occurs rarely (typically a week or so) and we do not have enough file system space.

This trace collects the last 24K of frame data.

Then there is another trace ‘strace‘ which can be used to collect all the data from the driver. This can be enabled by setting ‘hsi_trace‘ as

set HSI:hsi_trace=2

Then run

#strace 18515 all all > hsi_trace.log

This collects all the data from the driver. This trace is useful when we know that the problem occurs within a short time.

The trace output is as follows

In the first case (‘hsi_trace‘ utility)

13:26:38 0000004f hih9 len=0100 R: 31323334 35363738 fm: I-FR P/F=1 Nr=1 Ns=1

The fields are as follows

1 st field: Time stamp

2 nd field: time difference in microsecs between the last frame and current frame.

3 rd field: port

4 th field: length of the frame.

5 th field: R: received data T: transmitted data

6 th and 7 th field: First 8 bytes of the data transmitted or received.

7 th field: The frame type (SABM, TEST, XID, RR, RNR....)

Some of the frame types are described below.

Keyword	Value	Effect
RR	Receive Ready	This frame is used as a polling command by the primary station to solicit information frames from the secondary station.
RNR	Receive Not Ready	This frame is used as a flow control command or response to indicate that the station transmitting the Receive Not Ready frame is not able to accept any information frames at this time.
REJ	Reject	This frame is sent by a station to indicate that it has received a frame out of the normal sequence. This may indicate the loss of an information frame containing user data.
SABM	Set Async Balanced Mode	An LLC non-data frame requesting the establishment of a connection over which numbered information frames may be sent.
SNRM	Set Normal Response Mode	This command is sent from the primary station to a secondary station to place the secondary in the initialized normal SDLC operating mode.
SNRME	SNRM Extended	SNRM with two more bytes in the control field. Used in SDLC.
DISC	Disconnect	This command is sent from the primary station to the secondary station to place the secondary station in the off-line disconnected mode.
SIM	Set Initialization Mode	This command is sent from the primary station to the secondary station to being the initialization process.
UA	Unnumbered Ack	This response is sent from the secondary station to the primary station in response to an SNRM, DISC, or SIM command.
DM	Disconnect Mode	This response is sent from the secondary station to the primary station in response to any command other than SNRM or DISC.
RD	Request Disconnect	This response is sent from the secondary to the primary station to request that the secondary station be placed in the off-line or disconnect mode.
RIM	Req Init Mode	This response is sent from the secondary to the primary station to request initialization.

FRMR	Frame Reject	This response is sent from the secondary station to the primary station to indicate that an abnormal condition has been detected or that an invalid frame has been received. It contains bits which indicate the reason for the rejection of the frame.
XID	Exchange Identification	This frame may be either a command sent by the primary station or a response sent by the secondary station. It contains information that is used to identify the secondary station.
TEST	TEST	This command is sent from the primary station to the secondary station and may contain some form of a message that may be used to test the seconary's ability to receive data and transmit the data back to the primary station.
UI	Unnumbered Inforation	This command allows the primary station to send data to the secondary station and the unnumbered information response allows the secondary station to send data to the primary station.
INFO	Information	This frame contains the information and data relevant to the higher SNA architecture layers. INFO frames consist of several variable-length or optional fields, depending upon the implementation.
UP	unnumbered Poll frame	Used by a primary to poll a secondary.
BCN	Beacon	This is a beacon frame which is usually an indication of a problem.
CFGR	Configure	This is a configuration frame.

'strace' is the normal unix strace output.

020809 13:34:31 001c1330 0 ... 18515 0 hih8 len=0100 T: 31323334 35363738 fm: I-FR P/F=1 Nr=1 Ns=1

SEE ALSO

hsi_init(1M), hsi_stat(1M), hsi_loop(1M), hsi(7d)

DIAGNOSTICS

NAME

hsip_init − set high speed serial line interface operating parameters.

SYNOPSIS

/opt/SUNWconn/bin/hsip_init *device* **[[** *baud_rate* **] | [** *keyword=value***, ...] | [** *single-word option* **]]**

DESCRIPTION

The hsip_init utility allows the user to modify some of the hardware operating modes common to high speed synchronous serial lines. This may be useful in troubleshooting a link, or necessary to the operation of a communications package.

If run without options, hsip_init reports the options as presently set on the port. If options are specified, the new settings are reported after they have been made.

OPTIONS

Options to hsip_init normally take the form of a keyword, followed by an equal sign and a value. The exception is that a baud rate may be specified as a decimal integer by itself. Keywords must begin with the value shown in the options table, but may contain additional letters up to the equal sign. For example, "loop=" and "loopback=" are equivalent.

Recognized options are listed in the table below.

Keyword	Value	Effect
loopback	yes	Set the port to operate in **internal loopback** mode. The receiver is electrically disconnected from the DCE receive data input and tied to the outgoing transmit data line. Transmit data is available to the DCE. If no other clocking options have been specified, perform the equivalent of **txc=baud** and **rxc=baud**.
	no	Disable internal loopback mode. If no other clocking options have been specified, perform the equivalent of **txc=txc** and **rxc=rxc**.
	echo	Set the port to operate in **auto-echo** mode. The port will echo incoming receive data on the transmit data pin. When the loopback is set for echo and no clocking option is given the clocking is set txc=txc and rxc=rxc. Other clocking options can be used but line errors may occur due to the loopback=echo implementation.
nrzi	no	Set the port to operate with **NRZ** data encoding. **NRZ** encoding maintains a constant voltage level when data is present (1) and does not not return to a zero voltage (0) until data is absent. The data is decoded as an absolute value based on the voltage level (0 or 1).

	yes	Set the port to operate with **NRZI** data encoding. **NRZI** encoding does a voltage transition when data is absent (0) and no voltage transition (no return to zero) when data is present (1). Hence, the name non-return to zero inverted. The data is decoded using relational decoding.
txc	txc	Transmit clock source will be the **TxCI** signal.
	rxc	Transmit clock source will be the **RxC** signal.
	baud	Transmit clock source will be the internal **baud rate generator**.
	pll	Transmit clock source will be the output of the **DPLL** circuit. This can only be set with NRZI data encoding.
	-txc	Transmit clock source will be the inverted **TxCI** signal.
rxc	rxc	Receive clock source will be the **RxC** signal.
	txc	Receive clock source will be the **TxCI** signal. This can only be used with transmit clock option txc=txc.
	baud	Receive clock source will be the internal **baud rate generator**.
	pll	Receive clock source will be the output of the **DPLL** circuit. This can only be set with NRZI data encoding.
	-rxc	Receive clock source will be the inverted **RxC** signal.
txd	txd	Transmit data is not inverted.
	-txd	Transmit data is inverted.
rxd	rxd	Receive data is not inverted.
	-rxd	Receive data is inverted.
mode	fdx	HDLC Full Duplex mode (Default mode).
	ibm-fdx	IBM Full Duplex mode (SDLC).
	ibm-hdx	IBM Half Duplex mode (SDLC).
	ibm-mpt	IBM Multipoint mode (SDLC).
signal	yes	Notify application of modem signal (RTS and CTS) changes.
	no	Do not notify application of modem signal (RTS and CTS) changes.
mtu	*integer*	Set the maximum transmit unit to *integer* bytes with 2064 bytes maximum.
mru	*integer*	Set the maximum receive unit to *integer* bytes with 2064 bytes maximum.
speed	*integer*	Set the baud rate to *integer* bits per second with a minimum rate of 9600 bps and a maximum of 2048000 bps. Zero is also valid when txc is set to txc or -txc.

There are also several single-word options that set one or more paramaters at a time:

Keyword	Equivalent to Options:
external	txc=txc rxc=rxc loop=no
sender	txc=baud rxc=rxc loop=no
internal	txc=pll rxc=pll loop=no
stop	speed=0

EXAMPLES

The following command sets the first port to loop internally, use internal clocking and operate at 38400 baud:

example# hsip_init hihp0 38400 loop=yes
port=hihp0
speed=38400,
mode=fdx, signal=no, loopback=yes, nrzi=no, mtu=2064, mru=2064,
txc=baud, rxc=baud, txd=txd, rxd=rxd

The following command sets the same port's clocking, local loopback and baud rate settings to their default values:

example# hsip_init hihp0 speed=1536000 loopback=no txc=txc rxc=rxc
port=hihp0
speed=1536000,
mode=fdx, signal=no, loopback=no, nrzi=no, mtu=2064, mru=2064,
txc=txc, rxc=rxc, txd=txd, rxd=rxd

SEE ALSO

hsip_loop(1M), hsip_stat(1M), Intro(2), hsip(7D)

DIAGNOSTICS

device **missing minor device number**
The name *device* does not end in a decimal number that can be used as a minor device number.

bad speed: *arg*
The string *arg* that accompanied the "speed=" option could not be interpreted as a decimal integer.

Bad arg: *arg*
The string *arg* did not make sense as an option.

ioctl failure code = *errno*
An **ioctl(2)** system called failed. The meaning of the value of *errno* may be found in the **Intro(2)** manual page.

WARNINGS

hsip_init should not be used on an active serial link, unless needed to resolve an error condition. It should not be run casually, or if the user is unsure of the consequences of its use.

NAME

hsip_loop − high speed synchronous serial loopback test program for high speed serial interface.

SYNOPSIS

/opt/SUNWconn/bin/hsip_loop [−cdlsvt] *device*

DESCRIPTION

The hsip_loop command performs several loopback tests that are useful in exercising the various components of a serial communications link.

Before running a test, hsip_loop opens the designated port and configures it according to command line options and the specified test type. It announces the names of the devices being used to control the hardware channel, the channel number (ppa) corresponding to the *device* argument, and the parameters it has set for that channel. It then runs the loopback test in three phases.

The first phase is to listen on the port for any activity. If no activity is seen for at least four seconds, hsip_loop proceeds to the next phase. Otherwise, the user is informed that the line is active and that the test cannot proceed, and the program exits.

In the second phase, called the "first-packet" phase, hsip_loop attempts to send and receive one packet. The program will wait for up to four seconds for the returned packet. If no packets are seen after five attempts, the test fails with an error message. If a packet is returned, the result is compared with the original. If the length and content do not match exactly, the test fails.

The final phase, known as the "multiple-packet" phase, attempts to send many packets through the loop. Because the program has verified the integrity of the link in the first-packet phase, the test will not fail after a particular number of timeouts. If a packet is not seen after four seconds, a message is displayed. Otherwise, a count of the number of packets received is updated on the display once per second. If it becomes obvious that the test is not receiving packets during this phase, the user may wish to stop the program manually. The number and size of the packets sent during this phase is determined by default values, or by command line options. Each returned packet is compared with its original for length and content. If a mismatch is detected, the test fails. The test completes when the required number of packets have been sent, regardless of errors.

After the multiple-packet phase has completed, the program displays a summary of the hardware event statistics for the channel that was tested. The display takes the following form:

Port	**CRC errors**	**Aborts**	**Overruns**	**Underruns**	In	<-**Drops**->	**Out**
hihp0	**0**	**0**	**0**	**0**	**0**		**0**

This is followed by an estimated line speed, which is an approximation of the bit rate of the line, based on the number of bytes sent and the actual time that it took to send them. This is a very rough approximation and should not be used in bechmarking, because elapsed time includes time to print to the display.

OPTIONS

The options for hsip_loop are described in the following table:

Option	Parameter	Default	Description
−c	*packet_count*	100	Specifies the number of packets to be sent in the multiple-packet phase.
−d	*hex_data_byte*	*random*	Specifies that each packet will be filled with bytes with the value of *hex_data_byte*.
−l	*packet_length*	100	Specifies the length of each packet in bytes with a maximum of 2064 bytes.
−s	*line_speed*	9600	Bit rate in bits per second, minimum of 9600 bps and a maximum of 2048000 bps.
−v			Sets verbose mode. If data errors occur, the expected and received data is displayed.
−t	*test_type*	*none*	A number, from 1 to 4, that specifies which test to perform. The values for *test_type* are as follows:

1 Internal loopback test. Port loopback is on. Transmit and receive clock sources are internal (baud rate generator).

2 External loopback test. Port loopback is off. Transmit and receive clock sources are internal. Requires a loopback plug suitable to the port under test.

3 External loopback test. Port loopback is off. Transmit and receive clock sources are external (modem). Requires that one of the local modem or the remote modem be set in a loopback configuration.

4 Test using predefined parameters. User defines hardware configuration and may select port parameters using the **hsip_init(1M)** command.

All numeric options except **−d** are entered as decimal numbers (for example, **−s 19200**). If you do not provide the **−t** *test_type* option, hsip_loop prompts for it.

EXAMPLES

The following command causes hsip_loop to use a packet length of 512 bytes over the first CPU port:

example# hsip_loop −l 512 hihp0

In response to the above command, hsip_loop prompts you for the test option you want.

The following command performs an internal loopback test on the first CPU port, using 5000 packets and a bit rate of 56000 bps :

example# hsip_loop −t 1 −s 56000 −c 5000 hihp0

SEE ALSO

hsip_init(1M), hsip_stat(1M), hsip(7D)

DIAGNOSTICS

device **missing minor device number**
The name *device* does not end in a decimal number that can be used as a minor device number.

invalid packet length: *nnn*
The packet length was specified to be less than zero or greater than 2064.

poll: nothing to read
poll: nothing to read or write.
The **poll(2)** system call indicates that there is no input pending and/or that output would be blocked if attempted.

len *xxx* **should be** *yyy*
The packet that was sent had a length of *yyy*, but was received with a length of *xxx*.

nnn **packets lost in outbound queueing**
nnn **packets lost in inbound queueing**
A discrepancy has been found between the number of packets sent by hsip_loop and the number of packets the driver counted as transmitted, or between the number counted as received and the number read by the program.

WARNINGS

To allow its tests to run properly, as well as prevent disturbance of normal operations, hsip_loop should only be run on a port that is not being used for any other purpose at that time.

NAME

hsip_stat − report driver statistics from a high speed synchronous serial link port.

SYNOPSIS

/opt/SUNWconn/bin/hsip_stat [-f] -a|*num_of_ports*
/opt/SUNWconn/bin/hsip_stat [-f] *device* **[***period***]**
/opt/SUNWconn/bin/hsip_stat -c [-f] -a|*num_of_ports*
/opt/SUNWconn/bin/hsip_stat -c [-f] *device*

DESCRIPTION

The hsip_stat command reports the event statistics maintained by a high speed synchronous serial device driver. The report may be a single snapshot of the accumulated totals, or a series of samples showing incremental changes.

Event statistics are maintained by a driver for each physical channel that it supports. They are initialized to zero at the time the driver module is loaded into the system when one of the driver's entry points is first called.

The **device** argument is the name of the high speed serial device as it appears in the /dev directory. For example, **hihp0** specifies the first on-board high speed serial device.

As an alternative, you can display or clear the statistics for multiple physical channels using **num_of_ports** argument. The hsip_stat program will then display statistics accumulated for the first **n** number of ports, where **n** is **num_of_ports**.

The following is a breakdown of hsip_stat output:

speed	The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.
ipkts	The total number of input packets.
opkts	The total number of output packets.
undrun	The number of transmitter underrun errors.
ovrrun	The number of receiver overrun errors.
abort	The number of aborted received frames.
crc	The number of received frames with CRC errors.
isize	The average size (in bytes) of input packets.
osize	The average size (in bytes) of output packets.
ierror	Input error count (errors: Incomplete Frame, Empty frame, Glitch on RxC).
oerror	Output error count (errors: CTS lost, Glitch on TxC).
iutil	Input line utilization expressed as a percentage.
outil	Output line utilization expressed as a percentage.

OPTIONS

−f Select a complete set of accumulated statistics for the device specified. This is useful while debugging the **hsip** driver.

−a Select all devices.

−c Clear the accumulated statistics for the device specified. This may be useful when it is not desirable to unload a particular driver, or when the driver is not capable of being unloaded.

num_of_ports
Specify the number of devices that you want to dump the statistics.

period Cause hsip_stat to sample the statistics every *period* seconds and report incremental changes. The output reports line utilization for input and output in place of average packet sizes. These are the relationships between bytes transferred and the speed, expressed as percentages. The loop repeats indefinitely, with a column heading printed every twenty lines for convenience.

EXAMPLES

example# **hsip_stat hihp0**

speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize	osize
9600	15716	17121	0	0	1	3	98	89

example# **hsip_stat 5**

	speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize	osize
hihp0	9600	15716	10100	0	0	1	3	98	89
hihp1	9600	15234	20100	0	0	1	3	98	89
hihp2	9600	15123	18254	0	0	1	3	98	89
hihp3	9600	15378	18234	0	0	1	3	98	89
hihp4	9600	13900	13000	0	0	1	3	98	89

example# **hsip_stat -c hihp0**

speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize	osize
9600	0	0	0	0	0	0	0	0

example# **hsip_stat hihp0 5**

ipkts	opkts	undrun	ovrrun	abort	crc	iutil	outil
12	10	0	0	0	0	5%	4%
22	60	0	0	0	0	3%	90%
36	14	0	0	0	1	51%	2%

(In this final example a new line of output is generated every five seconds.)

SEE ALSO

hsip_init(1M), hsip_loop(1M), hsip(7D)

DIAGNOSTICS

bad interval: *arg*
The argument *arg* is expected to be an interval and could not be understood.

device **missing minor device number**

The name *device* does not end in a decimal number that can be used as a minor device number.

WARNINGS

Underrun, overrun, frame-abort and CRC errors have a variety of causes. Communication protocols are typically able to handle such errors and initiate recovery of the transmission in which the error occurred. Small numbers of such errors are not a significant problem for most protocols. However, because the overhead involved in recovering from a link error can be much greater than that of normal operation, high error rates can greatly degrade overall link throughput. High error rates are often caused by problems in the link hardware, such as cables, connectors, interface electronics or telephone lines. They may also be related to excessive load on the link or the supporting system.

The percentages for input and output line utilization reported when using the *interval* option may occasionally be reported as slightly greater than 100% because of inexact sampling times and differences in the accuracy between the system clock and the modem clock. If the percentage of use greatly exceeds 100%, or never exceeds 50%, then the baud rate set for the device probably does not reflect the speed of the modem.

NAME

nf_fddidaemon − start/stop the NF FDDI SMT/SNM daemon and its associated processes.

SYNOPSIS

nf_fddidaemon start | stop

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

The **nf_fddidaemon** script starts/stops the SNM daemon and its associated processes.

OPTIONS

start Starts the SNM daemon

stop Stops the SNM daemon

You must be root to run this command.

SEE ALSO

nf_snmd (1M)

NAME

nf_install_agents − install SunNet Manager agents for SunFDDI

SYNOPSIS

nf_install_agents

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

The **nf_install_agents** script copies the FDDI schema files to the directory in which the standard agents are installed and updates the configuration files for SunNet Manager

The **nf_install_agents** command takes no arguments.

You must be root to run this command.

SEE ALSO

nf_snmd (1M)

NAME

nf_macid − obtain MAC address from specified **nf** (SunFDDI) interface.

SYNOPSIS

nf_macid *interface*

AVAILABILITY

This command is available only with the *SunFDDI* product.

DESCRIPTION

This command queries the IDPROM on the SunFDDI SBus card associated with a **nf** interface to obtain the MAC address resident there. This address is a globally unique, 48-bit address that is drawn from the same pool from which Ethernet addresses are taken.

The **nf_macid** command does not allow you to set a MAC address, either on the SBus card or for an interface. Use **ifconfig** with the **ether** argument to assign the MAC address you obtain with **nf_macid** to an SunFDDI interface.

Normally, you use the host-resident MAC address for all network interfaces on a machine. You would only use the MAC address obtained with **nf_macid** under unusual circumstances.

You can be normal user (not root) to run this command.

OPTIONS

interface Specifies the FDDI interface (**nf**<*num*>). The default (which you can omit) is **nf0.**

EXAMPLE

Obtain the MAC address for **nf0:**

% nf_macid
8:0:20:3e:da:5

Set the **nf0** interface to have the MAC address in the SBus card IDPROM:

ifconfig nf0 ether ‘nf_macid‘

You would follow the preceding command with an **ifconfig** command to assign an IP address to **nf0** and bring up that interface. Normally, such **ifconfig** commands would be run from a startup file.

SEE ALSO

ifconfig (1M)

NAME

nf_smtmon − the SMT monitor.

SYNOPSIS

nf_smtmon [−**i** *interface*] [**-x**] [**-h**] [*frametype*]

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

nf_smtmon is used to display received SMT frames. You should run this command on the FDDI proxy system if the Console does not receive a response from a request for SMT MIB information.

You must be root to run this command.

OPTIONS

−**i** *interface* Specifies the FDDI interface (**nf***num* for SunFDDI). If this option is not specified, frames for all FDDI interfaces are displayed.

-x Displays the received frames in hex.

-h Displays the usage of this command.

frametype Specifies one or more types of SMT frames to be displayed. If this option is not specified, all types of frames are displayed. You can specify the following types of frames to be displayed:

ecf Echo Frame. Request and response frames are used for SMT-to-SMT loopback testing on an FDDI ring.

esf Extended Service Frame. Request, response, and announcement frames are used to extend new SMT services.

nif Neighborhood Information Frame. Request, response, and announcement frames are used to communicate station addresses and descriptions.

pmf_get Parameter Management Frame (PMF) Get Request. Request and response frames are used to retrieve SMT Management Information Base (MIB) attribute values.

rdf Request Denied Frame (response only). Sent in response to an unsupported or unknown request.

sifconfig Status Information Frame (SIF) Configuration. Request and response frames are used to retrieve configuration parameters for one or more stations on the ring.

sifoperation Status Information Frame (SIF) Operation. Request and response frames are used to retrieve operation information for one or more stations on the ring.

srf Status Report Frame. Announcement frame used to report Station Status. The current version of the SMT

daemon does not send out SRFs; however, any received SRFs are passed on to SNM as traps.

EXAMPLES

nf_smtmon -i nf0 nif sifconfig
displays the NIF and SIF configuration frames received in non-hex format on the **nf0** (SunFDDI) interface.

nf_smtmon -i nf1 -x ecf
displays, in hex, ECF frames received on the **nf1** (SunFDDI) interface.

SEE ALSO

smtd (1M)

NAME

nf_snmd − start the station management (SMT) to SunNet Manager daemon.

SYNOPSIS

nf_snmd [−**d**] [**-v5**]

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

Upon invocation, the SNM daemon starts up station management processes that allow the station to communicate with other stations using the SMT protocol, and collect and return FDDI statistics to a SunNet Manager (SNM) Console. The daemon also receives SMT requests and SMT responses. The daemon also sends out SMT requests to other stations on the ring on behalf of SNM. The SMT daemon also forwards received Status Report Frames (SRFs) to the SNM management station in the form of traps.

The processes started by the SNM daemon include two SNM agents: a local agent (fddi) and a proxy agent (fddismt). Like other SNM agents, the local agent and proxy agent communicate with the SNM management station using RPC. The local agent responds to SNM requests with FDDI statistics gathered on the local machine. These statistics are equivalent to those displayed with the **nf_stat** and **nf_stat -m** commands.

The proxy agent can return two types of SMT information to the SNM Console: actual SMT frames (ECF, ESF, NIF, SIF Configuration, or SIF Operation), and attribute values for selected SMT MIB groups. The proxy agent gathers information from target stations by issuing SMT request frames and receiving SMT response frames. The proxy uses PMF Get request and response frames to retrieve MIB attribute values from the target station.

If the target station does not support PMF Get frames, it returns an RDF response to the proxy system. If a Console request for MIB attributes values is not successful, run the SMT monitor on the proxy system to see if an RDF frame has been received from the target station. If PMF Get frames are not supported by the target station, you may be able to use NIF, SIF Configuration or SIF Operation frames to return the desired attribute values.

The SMT MIB attributes groups MAC, PATH, and PORT contain index parameters. If you send a Quick Dump request from the Console for attribute values from one of these groups, only the values associated with the first index are returned (from the Console's point of view, the key value associated with the request is 1). If you want to see attribute values associated with other indexes, you must send a Data Report request with the Key field in the request set to the desired index.

If you make any changes to the **/etc/opt/snm/snm.conf** file on the station (for example, you add an additional hostname to the **na.fddi.trap-rendez** entry), you must kill the SNM daemon with **nf_snmd_kill** and then restart it in order for the change(s) to take effect.

You must be root to run this command.

OPTIONS

-d (debug mode) Displays a one-line entry in the window where **nf_snmd** is started for each frame that the station sends or receives. If this option is not specified, you are returned to the system prompt and there is no display. Use of this option is not recommended if the **nf_snmd** command is included in **/etc/rc2.d/S98nf_fddidaemon .**

SEE ALSO

nf_snmd_kill (1M), nf_stat (1M)

NAME

nf_snmd_kill − kill the station management (SMT) to SunNet Manager daemon and its associated processes.

SYNOPSIS

nf_snmd_kill

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

The **nf_snmd_kill** script kills the SNM daemon and its associated processes. This command also kills the two SNM agents which are started by the SNM daemon: the local agent (fddi) and the proxy agent (fddismt). This command should not be used if the SNM daemon is not already running.

The **nf_snmd_kill** command takes no arguments.

You must be root to run this command.

SEE ALSO

nf_snmd (1M)

NAME

nf_stat − display SunFDDI interface statistics.

SYNOPSIS

nf_stat [−**m**][*interface*][*interval*][*count*]

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

The **nf_stat** utility displays statistics for the SunFDDI interface. Some statistics relate to the SunFDDI implementation of the ANSI FDDI Connection Management standard (CMT), while others contain packet throughput, or station neighbor information.

This utility can report, on a periodic basis, packet throughput statistics, reconfiguration events, and interface exceptions. It also reports the identity of neighboring stations, information on its PHYs, and some FORMAC error counters. Several of the counters and status variables are periodically passed to the host from the hardware during the heartbeat signal. These statistics are available when invoking the command without the *-m* option. Issuing the command without an *interval* value displays the accumulated statistics; issuing the command with an *interval* value displays any differences between values since the previous display.

OPTIONS

−**m** Dumps the current nearest neighbor information and FDDI/S timer settings (described below). The *interval* and *count* arguments have no effect when used with this option. Note that you must be root to invoke **nf_stat** with the **-m** option.

interface Specifies which SunFDDI interface, **nf***num.*

interval Specifies the interval in seconds at which to display the statistics.

count Specifies the number of times to display the statistics. If no count is provided, the utility runs forever. It can be terminated by typing ^C (Control-C).

USAGE

You invoke **nf_stat** *with* the **-m** option to display information about neighboring stations. It generates a columnar display containing the following categories of data:

PhyA On a machine running SunFDDI Dual, shows the PHY type of the neighboring station that is connected to PHYA. Values are **A**, **B**, **S**, **M**, and **None** (if no connection). This column does not appear on a machine running SunFDDI SAS - Single Attached Station. (See Chapter 9 of the document ANSI/FDDI Station Management (SMT) Rev7.2 (25 June 1992)).

PhyB On a machine running SunFDDI Dual, shows the PHY type of the neighboring station that is connected to PHYB. Values are **A**, **B**, **S**, **M**, and **None** (if no connection). This column does not appear on a machine running SunFDDI SAS. (See Chapter 7 of the document ANSI/FDDI Station Management (SMT) Rev7.2 (25 June 1992)).

PhyS On a machine running SunFDDI SAS, shows the PHY type of the neighboring station that is connected to PHYS. Values are **A**, **B**, **S**, **M**, and **None** (if no connection). If connected to a concentrator, this will be **M.** This column

does not appear on a machine running SunFDDI Dual.

Frame FDDI MAC standard counter, frames received.

Error FDDI MAC standard counter, frame with the E bit first detected at this station.

Lost Frames whose reception is aborted.

SA MAC address; the unique 48-bit address of the SunFDDI interface. Where an IP hostname exists, it is displayed; otherwise, the 48-bit MAC address is used.

UNA The address of this station's upstream neighbor, using the SMT NIF protocol.

DNA The address of this station's downstream neighbor, using the SMT NIF protocol.

Display status information : You invoke **nf_stat** without the *-m* option, or with values for *interface* or *interval*, to display status information. Issuing the command without an *interval* value displays the accumulated statistics; issuing the command with an *interval* value displays any differences between values since the previous display.

One use of **nf_stat** without the **-m** option is to monitor the **Ring_OP** (Ring Operational) column; if it indicates more than one ring_op per second, there are media problems that must be fixed.

When invoked without the **-m** option, **nf_stat** generates a columnar display containing the following categories of data:

Ring Indicates whether the ring is up or down (that is, the Claim has succeeded).

Note: The following five fields use terms described in the SMT document, Chapter 9.

ECM (**ec_state**). Shows the current state of the ECM state machine. Valid values are: **Out**, **In**, **Trace**, **Leave**, **Path_Test**, **Insert**, **Check**, and **Deinsert**.

RMT (**rmt_state**). Shows the current state of the RMT state machine. Valid values are: **Isolated**, **Non_Op**, **Ring_Op**, **Detect**, **Non_Op_Dup**, **Ring_Op_Dup**, **Directed**, and **Rm_Trace**.

PCMA/PCMB (for SunFDDI Dual) **PCMS** (for SunFDDI SAS)
(**pc_state**). Is a variable from PCM to other management entities containing the current state of the PCM state machine. Current valid values are: **Off (O)**, **Break (B)**, **Reject (R)**, **Connect (C)**, **Next (N)**, **Signal (S)**, **Join (J)**, **Verify (V)**, **Active (A)**, and **Maint (M)**.

Ring_OP (**Ring Operational**). Indicates the number of times the ring has come up (and therefore implies the number of times the ring has gone down).

XmitP The number of packets transmitted.

RecvP The number of packets received.

SEE ALSO **netstat (1M)**

NAME

nf_sync − configure SunFDDI interface to operate in synchronous mode.

SYNOPSIS

nf_sync *nf<inst>* [*tsync sap*]

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

The **nf_sync** utility is used to configure SunFDDI interfaces to operate in synchronous mode. By default, the SunFDDI interface configure to carry asynchronous traffic only.

OPTIONS

nf<inst> Specifies the FDDI interface,

tsync Specifies synchronous timer in nanoseconds, 400000 nanoseconds minimum,

sap Specifies the service access point (SAP) for synchronous operation.

USAGE

Running nf_sync without specifying values for *tsync* and *sap* returns current configuration of the interface.

To reconfigure SAP for asynchronous operations, specify tsync=0

EXAMPLES

nf_sync nf0

displays current configuration on the **nf0** (SunFDDI) interface.

nf_sync nf0 1000000 800

configures SAP 800 for synchronous operation with a clock rate 1000000 nanoseconds (1ms)

NAME

pf_fddidaemon − start / stop the PF FDDI SMT / SNM daemon and its associated processes.

SYNOPSIS

pf_fddidaemon start | stop

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

The **pf_fddidaemon** script starts / stops the SNM daemon and its associated processes.

OPTIONS

start Starts the SNM daemon

stop Stops the SNM daemon

You must be root to run this command.

SEE ALSO

pf_snmd (1M)

NAME

pf_install_agents − install SunNet Manager agents for SunFDDI

SYNOPSIS

pf_install_agents

AVAILABILITY

This command is available with the *SunFDDI* product.

DESCRIPTION

The **pf_install_agents** script copies the FDDI schema files to the directory in which the standard agents are installed and updates the configuration files for SunNet Manager

The **pf_install_agents** command takes no arguments.

You must be root to run this command.

SEE ALSO

pf_snmd (1M)

NAME

pf_macid − obtain MAC address from specified **pf** (SunFDDI/P) interface.

SYNOPSIS

pf_macid *interface*

AVAILABILITY

This command is available only with the *SunFDDI* product.

DESCRIPTION

This command queries the IDPROM on the SunFDDI card associated with a **pf** interface to obtain the MAC address resident there. This address is a globally unique, 48-bit address that is drawn from the same pool from which Ethernet addresses are taken.

The **pf_macid** command does not allow you to set a MAC address, either on the PCI card or for an interface. Use **ifconfig** with the **ether** argument to assign the MAC address you obtain with **pf_macid** to an SunFDDI interface.

Normally, you use the host-resident MAC address for all network interfaces on a machine. You would only use the MAC address obtained with **pf_macid** under unusual circumstances.

You can be normal user (not root) to run this command.

OPTIONS

interface Specifies the FDDI interface (**pf**<*num*>). The default (which you can omit) is **pf0.**

EXAMPLE

Obtain the MAC address for **pf0:**

% pf_macid
8:0:20:3e:da:5

Set the **pf0** interface to have the MAC address in the PCI card IDPROM:

ifconfig pf0 ether ‘pf_macid‘

You would follow the preceding command with an **ifconfig** command to assign an IP address to **pf0** and bring up that interface. Normally, such **ifconfig** commands would be run from a startup file.

SEE ALSO

ifconfig (1M)

NAME

pf_smtmon − the SMT monitor.

SYNOPSIS

pf_smtmon [−**i** *interface*] [**-x**] [**-h**] [*frametype*]

AVAILABILITY

This command is available with the *SunFDDI/P* product.

DESCRIPTION

pf_smtmon is used to display received SMT frames. You should run this command on the FDDI proxy system if the Console does not receive a response from a request for SMT MIB information.

You must be root to run this command.

OPTIONS

−**i** *interface* Specifies the FDDI interface (**pf***num* for SunFDDI/P). If this option is not specified, frames for all FDDI interfaces are displayed.

-x Displays the received frames in hex.

-h Displays the usage of this command.

frametype Specifies one or more types of SMT frames to be displayed. If this option is not specified, all types of frames are displayed. You can specify the following types of frames to be displayed:

ecf Echo Frame. Request and response frames are used for SMT-to-SMT loopback testing on an FDDI ring.

esf Extended Service Frame. Request, response, and announcement frames are used to extend new SMT services.

nif Neighborhood Information Frame. Request, response, and announcement frames are used to communicate station addresses and descriptions.

pmf_get Parameter Management Frame (PMF) Get Request. Request and response frames are used to retrieve SMT Management Information Base (MIB) attribute values.

rdf Request Denied Frame (response only). Sent in response to an unsupported or unknown request.

sifconfig Status Information Frame (SIF) Configuration. Request and response frames are used to retrieve configuration parameters for one or more stations on the ring.

sifoperation Status Information Frame (SIF) Operation. Request and response frames are used to retrieve operation information for one or more stations on the ring.

srf Status Report Frame. Announcement frame used to report Station Status. The current version of the SMT

daemon does not send out SRFs; however, any received SRFs are passed on to SNM as traps.

EXAMPLES

pf_smtmon -i pf0 nif sifconfig
displays the NIF and SIF configuration frames received in non-hex format on the **pf0** (SunFDDI/P) interface.

pf_smtmon -i pf1 -x ecf
displays, in hex, ECF frames received on the **pf1** (SunFDDI/P) interface.

SEE ALSO

smtd (1M)

NAME

pf_snmd − start the station management (SMT) to SunNet Manager daemon.

SYNOPSIS

pf_snmd [−**d**] [**-v5**]

AVAILABILITY

This command is available with the *SunFDDI/P* product.

DESCRIPTION

Upon invocation, the SNM daemon starts up station management processes that allow the station to communicate with other stations using the SMT protocol, and collect and return FDDI statistics to a SunNet Manager (SNM) Console. The daemon also receives SMT requests and SMT responses. The daemon also sends out SMT requests to other stations on the ring on behalf of SNM. The SMT daemon also forwards received Status Report Frames (SRFs) to the SNM management station in the form of traps.

The processes started by the SNM daemon include two SNM agents: a local agent (fddi) and a proxy agent (fddismt). Like other SNM agents, the local agent and proxy agent communicate with the SNM management station using RPC. The local agent responds to SNM requests with FDDI statistics gathered on the local machine. These statistics are equivalent to those displayed with the **pf_stat** and **pf_stat -m** commands.

The proxy agent can return two types of SMT information to the SNM Console: actual SMT frames (ECF, ESF, NIF, SIF Configuration, or SIF Operation), and attribute values for selected SMT MIB groups. The proxy agent gathers information from target stations by issuing SMT request frames and receiving SMT response frames. The proxy uses PMF Get request and response frames to retrieve MIB attribute values from the target station.

If the target station does not support PMF Get frames, it returns an RDF response to the proxy system. If a Console request for MIB attributes values is not successful, run the SMT monitor on the proxy system to see if an RDF frame has been received from the target station. If PMF Get frames are not supported by the target station, you may be able to use NIF, SIF Configuration or SIF Operation frames to return the desired attribute values.

The SMT MIB attributes groups MAC, PATH, and PORT contain index parameters. If you send a Quick Dump request from the Console for attribute values from one of these groups, only the values associated with the first index are returned (from the Console's point of view, the key value associated with the request is 1). If you want to see attribute values associated with other indexes, you must send a Data Report request with the Key field in the request set to the desired index.

If you make any changes to the **/etc/opt/snm/snm.conf** file on the station (for example, you add an additional hostname to the **na.fddi.trap-rendez** entry), you must kill the SNM daemon with **pf_snmd_kill** and then restart it in order for the change(s) to take effect.

You must be root to run this command.

OPTIONS

-d (debug mode) Displays a one-line entry in the window where **pf_snmd** is started for each frame that the station sends or receives. If this option is not specified, you are returned to the system prompt and there is no display. Use of this option is not recommended if the **pf_snmd** command is included in **/etc/rc2.d/S98pf_fddidaemon .**

SEE ALSO

pf_snmd_kill (1M), pf_stat (1M)

NAME

pf_snmd_kill − kill the station management (SMT) to SunNet Manager daemon and its associated processes.

SYNOPSIS

pf_snmd_kill

AVAILABILITY

This command is available with the *SunFDDI/P* product.

DESCRIPTION

The **pf_snmd_kill** script kills the SNM daemon and its associated processes. This command also kills the two SNM agents which are started by the SNM daemon: the local agent (fddi) and the proxy agent (fddismt). This command should not be used if the SNM daemon is not already running.

The **pf_snmd_kill** command takes no arguments.

You must be root to run this command.

SEE ALSO

pf_snmd (1M)

NAME

pf_stat − display SunFDDI/P interface statistics.

SYNOPSIS

pf_stat [−**m**][*interface*][*interval*][*count*]

AVAILABILITY

This command is available with the *SunFDDI/P* product.

DESCRIPTION

The **pf_stat** utility displays statistics for the SunFDDI/P interface. Some statistics relate to the SunFDDI/P implementation of the ANSI FDDI Connection Management standard (CMT), while others contain packet throughput, or station neighbor information.

This utility can report, on a periodic basis, packet throughput statistics, reconfiguration events, and interface exceptions. It also reports the identity of neighboring stations, information on its PHYs, and some FORMAC error counters. Several of the counters and status variables are periodically passed to the host from the hardware during the heartbeat signal. These statistics are available when invoking the command without the *-m* option. Issuing the command without an *interval* value displays the accumulated statistics; issuing the command with an *interval* value displays any differences between values since the previous display.

OPTIONS

−**m** Dumps the current nearest neighbor information and FDDI/S timer settings (described below). The *interval* and *count* arguments have no effect when used with this option. Note that you must be root to invoke **pf_stat** with the **-m** option.

interface Specifies which SunFDDI/P interface, **pf***num.*

interval Specifies the interval in seconds at which to display the statistics.

count Specifies the number of times to display the statistics. If no count is provided, the utility runs forever. It can be terminated by typing ^C (Control-C).

USAGE

You invoke **pf_stat** *with* the **-m** option to display information about neighboring stations. It generates a columnar display containing the following categories of data:

PhyA On a machine running SunFDDI/P Dual, shows the PHY type of the neighboring station that is connected to PHYA. Values are **A**, **B**, **S**, **M**, and **None** (if no connection). This column does not appear on a machine running SunFDDI/P SAS - Single Attached Station. (See Chapter 9 of the document ANSI/FDDI Station Management (SMT) Rev7.2 (25 June 1992)).

PhyB On a machine running SunFDDI/P Dual, shows the PHY type of the neighboring station that is connected to PHYB. Values are **A**, **B**, **S**, **M**, and **None** (if no connection). This column does not appear on a machine running SunFDDI/P SAS. (See Chapter 7 of the document ANSI/FDDI Station Management (SMT) Rev7.2 (25 June 1992)).

PhyS On a machine running SunFDDI/P SAS, shows the PHY type of the neighboring station that is connected to PHYS. Values are **A**, **B**, **S**, **M**, and **None**

(if no connection). If connected to a concentrator, this will be **M.** This column does not appear on a machine running SunFDDI/P Dual.

Frame FDDI MAC standard counter, frames received.

Error FDDI MAC standard counter, frame with the E bit first detected at this station.

Lost Frames whose reception is aborted.

SA MAC address; the unique 48-bit address of the SunFDDI/P interface. Where an IP hostname exists, it is displayed; otherwise, the 48-bit MAC address is used.

UNA The address of this station's upstream neighbor, using the SMT NIF protocol.

DNA The address of this station's downstream neighbor, using the SMT NIF protocol.

Display status information : You invoke **pf_stat** without the *-m* option, or with values for *interface* or *interval*, to display status information. Issuing the command without an *interval* value displays the accumulated statistics; issuing the command with an *interval* value displays any differences between values since the previous display.

One use of **pf_stat** without the **-m** option is to monitor the **Ring_OP** (Ring Operational) column; if it indicates more than one ring_op per second, there are media problems that must be fixed.

When invoked without the **-m** option, **pf_stat** generates a columnar display containing the following categories of data:

Ring Indicates whether the ring is up or down (that is, the Claim has succeeded).

Note: The following five fields use terms described in the SMT document, Chapter 9.

ECM (**ec_state**). Shows the current state of the ECM state machine. Valid values are: **Out**, **In**, **Trace**, **Leave**, **Path_Test**, **Insert**, **Check**, and **Deinsert**.

RMT (**rmt_state**). Shows the current state of the RMT state machine. Valid values are: **Isolated**, **Non_Op**, **Ring_Op**, **Detect**, **Non_Op_Dup**, **Ring_Op_Dup**, **Directed**, and **Rm_Trace**.

PCMA/PCMB (for SunFDDI/P Dual) **PCMS** (for SunFDDI/P SAS)
(**pc_state**). Is a variable from PCM to other management entities containing the current state of the PCM state machine. Current valid values are: **Off (O)**, **Break (B)**, **Reject (R)**, **Connect (C)**, **Next (N)**, **Signal (S)**, **Join (J)**, **Verify (V)**, **Active (A)**, and **Maint (M)**.

Ring_OP (**Ring Operational**). Indicates the number of times the ring has come up (and therefore implies the number of times the ring has gone down).

XmitP The number of packets transmitted.

RecvP The number of packets received.

SEE ALSO **netstat (1M)**

NAME

rscadm − administer SUN(tm) Remote System Control (RSC)

SYNOPSIS

```
rscadm help
rscadm resetrsc [-s]
rscadm set variable value
rscadm download [boot] file
rscadm show [variable]
rscadm date [-s] | [[mmdd]HHMM | mmddHHMM[cc]yy][.SS]
rscadm send_event [-c] message
rscadm modem_setup
rscadm useradd username
rscadm userdel username
rscadm usershow [username]
rscadm userpassword username
rscadm userperm username [cuar]
```

DESCRIPTION

rscadm administers the SUN(tm) Remote System Console (RSC). It allows the host server to interact with the RSC. The following operations are supported:

rscadm help
: Displays a usage screen.

rscadm resetrsc
: Reset the RSC. There are two types of reset allowed, a "hard" reset and a "soft" reset. The hard reset is done by default. The soft reset can be selected by using the -s option.

rscadm set
: Set RSC configuration variables. Examples of RSC configuration variables include RSC IP address and RSC hostname. See the RSC documentation for a complete list of RSC configuration variables.

rscadm download
: Program the RSC's firmware. There are two parts to the firmware, the boot monitor and the main image. By default, rscadm download programs the main firmware image. The boot option selects programming of the boot monitor.

rscadm show
: View the current RSC configuration variable settings. If no variable is specified, rscadm shows all variable settings.

rscadm date
: Show or set RSC's time and date. The -s options can be used to set RSC's time and date to the hosts time and date.

rscadm send_event
: Send a text based event to RSC. RSC may forward the event based on its event configuration.

rscadm modem_setup
: Direct connection to the RSC modem. This allows the user to enter AT commands to configure the modem. "~." returns to prompt.

rscadm useradd
: Add user account to RSC. RSC can support up to four separate users.

rscadm userdel
: Delete a user account from RSC.

rscadm usershow
: Show details on the specified user account. If a username is not specified, all user accounts will be shown.

rscadm userpassword
: Set a password for the user account specified. This password overrides any existing password currently set. There is no verification of the old password before setting the new password. See the RSC documentation on valid password formats.

rscadm userperm
: Set the authorization profile for the user. See the userperm options section in this man page for more detail.

OPTIONS

The following options are supported for rscadm:

rscadm resetrsc

[-s] Perform a "soft" reset instead of a "hard" reset. A hard reset physically resets the RSC hardware. The RSC software jumps to the boot firmware, simulating a reset, for a soft reset.

rscadm download

[boot] Program the boot monitor portion of the flash. The main portion of the flash is usually programmed.

rscadm show

[variable] Show the value of that particular variable.

rscadm date

[-s] Set the date to the hosts time and date.

[[mmdd]HHMM | mmddHHMM[cc]yy][.SS]
: the date.

mm - month
dd - day
HH - hour
MM - minute
cc - the first two digits of the four digit year
yy - last 2 digits of the year number
SS - seconds

rscadm send_event

[-c] Send a critical event. Without the -c, send_event sends a warnings. Warnings are only logged in the RSC event log and not forwarded further.

rscadm usershow

[username] RSC account name to display info on. If no username is given, all accounts will be displayed.

rscadm userperm

[cuar] Set permissions for RSC account. If no permissions are specified, all four permissions will be disabled. The options are to; allow user to connect to (c)onsole, allow user to use the (u)ser commands to modify RSC accounts, allow user to (a)dminister/change the RSC configuration variables, allow the user to (r)eset RSC and to power on/off the host.

OPERANDS

The following operands are supported for rscadm:

rscadm set

variable RSC configuration variable to set. See the RSC documentation for a list of configuration variables.

value Value to set RSC configuration variable to. See the RSC documentation for a list of valid values.

rscadm download

file Firmware file to download. The file should contain the RSC boot monitor image or RSC main image.

rscadm send_event

message Text message to describe event. Should be enclosed in quotes.

rscadm useradd

username Username for new RSC account.

rscadm userdel

username RSC account to be removed.

rscadm userpassword

username RSC account to have password set.

rscadm userperm

username RSC account to have permissions changed.

EXIT STATUS

= 0 on success
!= 0 on failure (with status message)

EXAMPLES

```
# rscadm date
# rscadm date -s
# rscadm date 050113101998
```

```
# rscadm set hostname rsc15
# rscadm show
# rscadm show hostname
# rscadm send_event -c "The UPS signaled a loss in power!"
# rscadm send_event "The disk is close to full capacity"
# rscadm useradd rscroot
# rscadm userdel olduser
# rscadm usershow
# rscadm usershow rscroot
# rscadm userperm rscroot cuar
# rscadm userperm newuser c
# rscadm userperm newuser
```

NOTES

rscadm modem_setup - "~." will only work after a new line.

rscadm MUST be run as root.

BUGS

None known.

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

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 shell window.

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 after telnet-ing(for example, **set term=vt100**

   c. Set the values of columns and rows to the value noted above. (See **stty**(1)).

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)

Index

A

afb (7D), 7-70
afbconfig (1M), 1M-1

D

device and network interfaces, 7-70 to 7-93

G

GFXconfig (1M), 1M-9

H

hsi (7D), 7-75
hsi_init (1M), 1M-13
hsi_loop (1M), 1M-16
hsi_stat (1M), 1M-19
hsi_trace (1M), 1M-22
hsip (7D), 7-79
hsip_init (1M), 1M-25
hsip_loop (1M), 1M-28
hsip_stat (1M), 1M-31

M

maintenance commands, 1M-1 to 1M-69

N

nf (7), 7-83
nf_fddidaemon (1M), 1M-34
nf_install_agents (1M), 1M-35
nf_macid (1M), 1M-36
nf_smtmon (1M), 1M-37
nf_snmd (1M), 1M-39
nf_snmd_kill (1M), 1M-41
nf_stat (1M), 1M-42
nf_sync (1M), 1M-45

P

pf (7), 7-86
pf_fddidaemon (1M), 1M-46
pf_install_agents (1M), 1M-47
pf_macid (1M), 1M-48
pf_smtmon (1M), 1M-49
pf_snmd (1M), 1M-51
pf_snmd_kill (1M), 1M-53
pf_stat (1M), 1M-54

R

rscadm (1M), 1M-57

S

smt (7), 1M-91
sunvts (1M), 1M-61

V

vtsk (1M), 1M-62
vtsprobe (1M), 1M-63
vtstty (1M), 1M-66
vtsui (1M), 1M-68
vtsui.ol (1M), 1M-69