SunVTS™ 6.2 Patch Set 1 Documentation Supplement for SPARC® Platforms

Sun Microsystems, Inc.
www.sun.com

Part No. 819-7446-10
August 2006, Revision A

Submit comments about this document at: http://www.sun.com/hwdocs/feedback

Please
Recycle

Adobe PostScript

Contents

Preface

SunVTS™ is the Sun Microsystems™ Inc. Validation Test Suite. SunVTS is a comprehensive software diagnostic package that tests and validates Sun SPARC® and x86 hardware by verifying the configuration and functionality of controllers, devices, and platforms.

This document is a supplement to the *SunVTS 6.2 Test Reference Manual for SPARC Platforms* (819-6455) and the *SunVTS 6.2 User's Guide* (819-6454). Refer to these documents for details on the full collection of SunVTS 6.2 tests and software features that are not new or enhanced in this release.

Note – The Solaris™ release with which this version of SunVTS is delivered supports systems that use the SPARC and x86 families of processor architectures UltraSPARC®, SPARC 64, AMD64, Pentium, and Xeon EM64T. The supported systems appear in the *Solaris 10 Hardware Compatibility List* at `http://www.sun.com/bigadmin/hcl`. This document cites any implementation differences between the platform types.

Before You Read This Book

To fully use the information in this document, you must have thorough knowledge of the topics described in these documents:

- *SunVTS 6.2 User's Guide*
- *SunVTS 6.2 Test Reference Manual for SPARC Platforms*
- *SunVTS Quick Reference Card*

How This Document Is Organized

Chapter 1 provides a brief overview of SunVTS and describes new or enhanced features for this release..

The remaining chapters are actual test reference chapters of SunVTS tests that incurred changes for this release. These chapters are extracted from the SunVTS 6.2 Test Reference Manual.

Using UNIX Commands

This document might not contain information on basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices.

Refer to one or more of the following for this information:

- *Solaris Handbook for Sun Peripherals*
- Software documentation that you received with your system
- Solaris operating system documentation, which is at:

  `http://docs.sun.com`

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Typographic Conventions

Typeface*	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this. To delete a file, type `rm` *filename*.

* The settings on your browser might differ from these settings.

Related Documentation

This document covers SunVTS version 6.2 tests. It serves as a reference companion to the SunVTS documents listed below.

Application	Title	Part Number
Installation and Navigation	*SunVTS 6.2 User's Guide*	819-6454
Test Specific Reference	*SunVTS 6.2 Test Reference Manual for SPARC Platforms*	819-6455
Quick Reference Card	*SunVTS Quick Reference Card*	819-6458

Accessing Sun Documentation

You can view, print, or purchase a broad selection of Sun documentation, including localized versions, at:

http://www.sun.com/documentation

Third-Party Web Sites

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in this document, go to:

http://www.sun.com/service/contacting

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

http://www.sun.com/hwdocs/feedback

Please include the title and part number of your document with your feedback:

SunVTS 6.2 Patch Set 1 Documentation Supplement for SPARC Platforms, part number 819-7446-10

CHAPTER 1

SunVTS Overview

This chapter contains the following topics:

The Sun Validation and Test Suite (SunVTS) software performs multiple diagnostic hardware tests from a single user interface. SunVTS verifies the connectivity, functionality, and reliability of controllers and devices.

Use SunVTS to test one device or multiple devices. Some of the major test categories are as follows:

- Audio tests
- Communication (serial and parallel) tests
- Graphic or video tests
- Memory tests
- Network tests
- Peripherals (disks, tape, CD-ROM, DVD-ROM, printer, diskette) tests
- Processor tests
- Storage tests

SunVTS comprises of many individual tests that support testing of a wide range of products and peripherals. Most of the tests can test devices in a 32-bit or 64-bit Solaris operating system (OS).

Such flexibility requires that the proper test modes and options need to be selected to maximize its effectiveness. This document covers the individual test options, modes, and requirements. For overall test configuration modes and options refer to the *SunVTS User's Guide*.

Note – When an error occurs in SunVTS testing, the test message window displays the error number, the error description, the probable cause of the error, and the recommended actions. Because this information is displayed at the time of the error, error messages are not included in this document.

The default installation directory for SunVTS is `/opt/SUNWvts`. However, when you are installing SunVTS, you can specify a different directory. Refer to the *SunVTS User's Guide* for installation information.

Note – SunVTS does not support processor sets. If processor sets are defined, you must first delete the processor sets before running SunVTS.

SunVTS Version Information

The standard command line argument, `-V`, displays the SunVTS version and release date of the test, if available.

SunPCi-3 Card Support

`sunpci2test` now supports the SunPCi™ III cards. Solaris 10 OS supports SunPCi-III Version 3.2.2 with Patch 118591-03 only. Solaris 10 does not support the SunPCi-2 card.

Test Requirements

SunVTS 6.2 was first introduced and designed to run in the Solaris 10 1/06 (Solaris 10 Update 1) OS and subsequent compatible releases. SunVTS 6.2 is not supported on OS releases prior to Solaris 10 3/05 (Solaris 10).

The operating system kernel must be configured to support all peripherals that are to be tested.

Some SunVTS tests have special requirements such as the connection of loopback connectors, installation of test media, or the availability of disk space. These requirements are listed for each test in the corresponding chapter in this document.

Collection of SunVTS Tests

Many individual tests make up the SunVTS collection of tests. Each test is a separate process from the SunVTS kernel. Each test can be run individually from the command line or from the SunVTS user interface.

When SunVTS is started, the SunVTS kernel automatically probes the system kernel to determine the hardware devices. The devices are then displayed on the SunVTS control panel with the appropriate tests and test options. This provides a quick check of your hardware configuration, and no time is wasted trying to run tests that are not applicable to your configuration.

During testing, the hardware tests send the test status and messages to the SunVTS kernel through interprocess communication (IPC) protocols. The kernel passes the status to the user interface and logs the messages.

SunVTS has a shared object library that contains test-specific probing routines. At runtime, the SunVTS kernel dynamically links in and calls these probing routines to initialize its data structure with test-specific information. You can add new tests into the SunVTS environment without recompiling the SunVTS source code.

Beginning with SunVTS 3.0, the SunVTS kernel and most tests support 32-bit and 64-bit operating systems. When the `sunvts` command is used to start SunVTS, the appropriate tests (32-bit or 64-bit versions) are presented.

32- and 64-Bit Tests

In Solaris 10 or later OSs, only 64-bit compatible tests are supported. Because each test is a separate program, you can run individual tests directly from the command line. Run tests from specific directories as follows:

- 32-bit tests – `/opt/SUNWvts/bin/`*testname*
- 64-bit tests – `/opt/SUNWvts/bin/sparcv9/`*testname*

If you are not sure which OS is running, refer to the Solaris System Administration manuals. In Solaris 10 OS, you can use the following command to identify the application support of your system.

```
# isainfo -v
```

Note – The `isainfo` command is not available in Solaris 2.6 or earlier releases.

SunVTS User Interfaces

You can run SunVTS tests from the JDS graphical user interface or the TTY interface. SunVTS tests can also be run individually from a shell tool command line, using the command-line syntax for each test (refer to "Running a Test From the Command Line" on page 6). TABLE 1-1 describes the various SunVTS user interfaces. Refer to the *SunVTS User's Guide* for more information on these interfaces.

TABLE 1-1 SunVTS System Interfaces

SunVTS System Interfaces	Description
Graphical user interface (GUI)	Select tests and test options with a mouse in the Solaris Java Desktop System (JDS) interface.
TTY interface	Run SunVTS from a terminal or modem attached to a serial port. You use the keyboard instead of the mouse. The interface displays one screen of information at a time.
Command-line execution	Run each of the SunVTS tests individually from a shell tool using the command-line syntax. Each test description in this document contains the corresponding command-line syntax.

Running a Test From a User Interface

The common way to run SunVTS testing is through a SunVTS user interface—JDS or the TTY interface.

Test configuration, control, and results are easily accessed through buttons and dialog boxes. These buttons and dialog boxes are covered in the *SunVTS User's Guide*. However, the Test Parameter Options dialog box is unique for each test, and is therefore covered in this manual.

Test Parameter Options Dialog Box

The options displayed in this menu differ for each test, but the Apply menu, and the Reset and Cancel buttons are generic. TABLE 1-2 describes all the items.

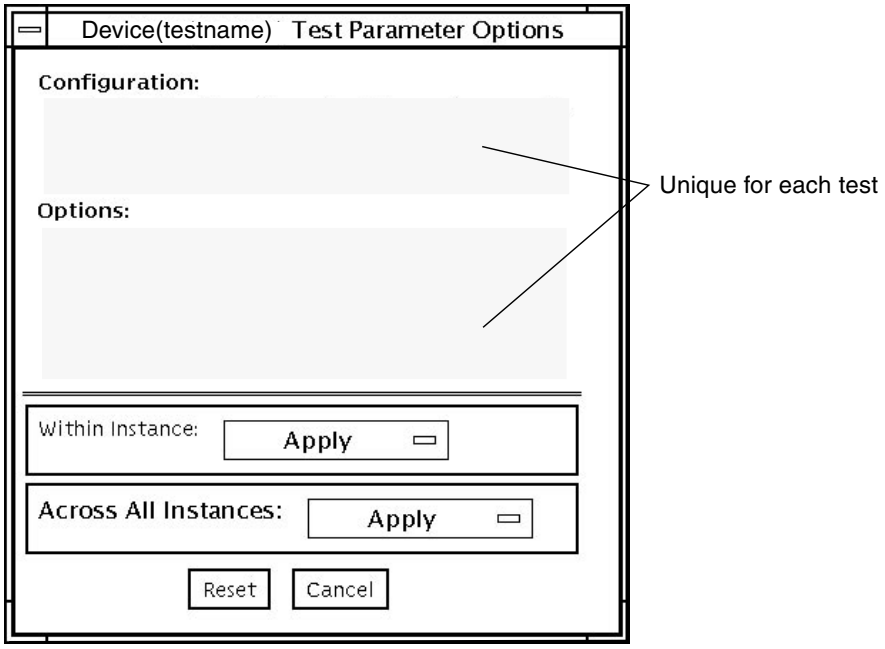

FIGURE 1-1 Test Parameter Options Dialog Box

TABLE 1-2 Test Parameter Options Dialog Box Items

Menu Item	Description
Configuration	Information such as device type, capacity, revision, and serial numbers for the selected device. This information cannot be changed.
Options	A list of test options that are used to customize the testing of the selectable device, group, or all devices. The options are specific for each test and are covered in the test specific-chapters in this manual.
Within Instance	Provides the means to apply the settings: • To this device only with Apply • To all devices within this group with Apply to Group • To all devices (of the *same device type* for *all controllers*) with Apply to All The option settings are only applied to one instance of the test.

TABLE 1-2 Test Parameter Options Dialog Box Items *(Continued)*

Menu Item	Description
Across All Instances	Provides the means to apply the settings globally: • To this device only with Apply • To all devices within this group with Apply to Group • To all devices (of the *same device type* for *all controllers*) with Apply to All The option settings are applied to all instances.
Reset	Returns the option values to their default settings and closes the Test Parameter Option dialog box.
Cancel	Ignores any changes made to option values and closes the Test Parameter Option dialog box.

Note – The Test Parameter Options dialog box descriptions also apply to the Test Parameter Options menu in the TTY interface.

Running a Test From the Command Line

In some cases it may be more convenient to run a single SunVTS test from the command line rather than through a SunVTS user interface. The following information describes how to do this.

Unless specified, the test runs without the SunVTS kernel (`vtsk`). All events and errors are sent to `stdout` or `stderr` and are not logged in the log files.

When you run a test in this way, you must specify all test options in the form of command-line arguments. There are two types of command-line arguments:

- Standard arguments—common to all tests. Refer to TABLE 1-3 for details.
- Test specific arguments—unique to a specific test. Refer to the test-specific chapters in this book for details.

The standard syntax for all SunVTS tests is:

`testname` `[`**`-scruvdtelnf`**`]` `[`**`-i`** *number*`]` `[`**`-w`** *number*`]`[**`-o`** *test-specific-arguments*`]`

Note – 64-bit tests are located in the `sparcv9` subdirectory: `/opt/SUNWvts/bin/sparcv9/`*testname*, or the relative path to which you installed SunVTS. If a test is not present in this directory, then it might be available as a 32-bit test only. For more information, see "32- and 64-Bit Tests" on page 3.

Standard Command-Line Arguments

The following table defines the standard SunVTS command-line arguments:

TABLE 1-3 Standard Command-Line Arguments

Argument	Description
`-s`	Runs a test as though it were invoked from the SunVTS kernel (`vtsk`). The default is to send the output to `stdout` or `stderr`.
`-c`	Enables a core image of the test process to be created in the current working directory upon receipt of certain signals, otherwise those signals are caught and handled to prevent a core from being generated. The default is to disable the creation of a core image.
`-r`	Enables run on error so that when an error occurs, the test continues with the next test sequence instead of exiting. The default is false.
`-v`	Runs the test in Verbose mode and displays messages with more detailed information about the testing process. The default is false.
`-V`	Displays the SunVTS version and release date of the test.
`-d`	Runs the test in Debug mode and displays messages to help programmers debug their test code. The default is false.
`-t`	Runs the test in Trace mode and displays messages that track function calls and sequences currently in use by the test code. The default is false.
`-l`	Runs the test in Online Functional mode. This is the same mode that tests run in when executed with the `vtsui.online` command. It is a non-intrusive version that will not significantly affect other applications. See the note below. The default is true.
`-x`	Runs the test in Exclusive mode.
`-n`	Runs the test in Connection mode. See the note below. The default is false.
`-f`	Runs the test in full Functional mode. This mode assumes that the test has complete control of the device under test. See the note below. The default is false.
`-p` *number*	Defines the number of passes for scalable tests. The default is 1.
`-i` *number*	Defines the number of instances for scalable tests. The default is 1.
`-w` *number*	Defines to which instance the test is assigned; this option is for scalable tests. The default is 0.
`-o`	Indicates that the options and arguments that follow are test specific.

Note – Separate each test-specific argument by commas, with no space after each comma.

Note – If you choose to specify a test mode with the `l`, `n`, or `f` option, specify only one option at a time because only one test mode can be selected at a time.

Test-Specific Arguments

SunVTS includes test-specific arguments that follow the format specified in the `getsubopt(3c)` man page. Separate each test-specific argument by commas, with no space after the comma. For example: `#`**`./sample -v -o dev=/dev/audio,volume=78`**. For information about test-specific arguments refer to the specific test chapter in this document.

Frame Buffer Tests

SunVTS includes a number of tests that exercise frame buffers:

- `cryptotest`
- `graphicstest`
- `ibhcatest`
- `ifbtest` (Expert 3D)
- `jfbtest` (XVR-1200)
- `m64test`
- `pfbtest` (XVR-1000)

If you are testing more than one frame buffer, follow these guidelines and instructions.

Caution – Disable the Power Management screen saver option and the Save/Resume option before you run any of the SunVTS frame buffer tests. For information on disabling these Power Management features, refer to the Power Management chapter in the *Solaris Common Desktop Environment: Users's Guide* in the Solaris 9 User Collection. This document is available at: `docs.sun.com`.

Caution – If you are using the JDS interface for SunVTS, do not conduct frame buffer tests through the `dtlogin` window. Log in as `root` and disable the auto-logout option.

Caution – Do not run TTY mode and frame buffer tests concurrently on the console monitor. The frame buffer test may fail.

Testing Multiple Frame Buffers

The following rules apply when you test multiple frame buffers (displays) simultaneously:

- Only the console monitor can run the window environment (such as JDS). The console monitor is the monitor connected to the frame buffer appointed by `/dev/fb`. SunVTS enables frame buffer locking on the console monitor by default.
- The frame buffer that is running the window environment must have window locking enabled to avoid false test failures. All other frame buffers must have window locking disabled.
- The group concurrency for graphics must be set to 1.

Remote Testing of Frame Buffers

If you start `sunvts` or `vtsk` from a screen other than the console monitor, frame buffer locking is not available. In this case:

- Disable the window locking option on the remote screen by setting it to `d`.
- Enable frame buffer locking for the console monitor, as shown in the example above. The SunVTS user interface cannot display on a monitor if locking is disabled.

Do not run any graphic programs (including `vtsui`) on the remote frame buffer during graphic testing.

Locking Frame Buffers

If you are testing multiple frame buffers or remote frame buffers, you might need to enable or disable frame buffer locking.

▼ To Enable Frame Buffer Locking

- **Take one of the following actions:**
  - In the JDS SunVTS interface, go to the Option menu of the graphics test and select Enable for the Frame Buffer Locking option.
  - At the command line, use the `lock=e` option.

  For example:

```
# ./fbtest -o dev=cgthree0, lock=enable
```

▼ To Disable Frame Buffer Locking

- **Take one of the following actions:**
  - In the JDS SunVTS interface, go to the Option menu of the graphics test and select Disable for the Frame Buffer Locking option.
  - At the command line, use the `lock=d` option. For example:

```
# ./fbtest -o dev=cgthree0, lock=disable
```

CHAPTER 2

Sun XVR-2500 and XVR-300 Graphics Accelerator Test (`graphicstest`)

This chapter describes how to use graphicsttest for both the XVR-300 and XVR-2500. The following topics are included:

`graphicstest` Description

`graphicstest` verifies the functionality of XVR-2500 and XVR-300 frame buffers. Test accuracy is checked using direct image comparison against compressed images. Failed pixel locations are printed as error messages.

Caution – Do not run any other application or screen saver program that uses the frame buffer's accelerator port while running `graphicstest`. This combination causes SunVTS to return incorrect errors.

graphicstest Test Requirements for XVR-2500

To run graphicstest as super user, you must use the console device or start the desktop with the -ac option.

Note – The CDE desktop is an X client that prevents the X server from restarting after each graphicstest finishes testing the kfb device.

Disable all screen savers before testing any graphics device. To disable the Solaris screen saver, type the following at a UNIX prompt:

```
# xset s off
```

To turn Display Power Management off, type the following at a UNIX prompt:

```
# xset -dpms
```

The display resolution must be 1280x1024. To change resolution, go to a UNIX prompt and type:

```
# fbconfig -res 1280x1024x76
```

graphicstest Test Preparation for XVR-2500

You must run graphicstest through the SunVTS user interface. The desktop must be running on the XVR-2500, and the desktop must be setup to run in 24-bit mode (the default depth for XVR-2500). Perform on of the following procedures prior to performing graphicstest to ensure that the test runs as smoothly as possible.

▼ To Run `graphicstest` in a Desktop GUI

1. **Turn Power Management off, if it is enabled. See** "`graphicstest` Test Requirements for XVR-2500" on page 12**.**
2. **Verify that no other program is running that might modify the screen during the test.**
3. **Verify that you have permission to lock the X server.** `graphicstest` **is designed to lock the X server during testing to prevent screen changes.**
4. **Verify that the JDS login window is not displayed during testing.**
5. **For frame buffers other than kfb, verify that the desktop is running on one frame buffer only.**

   If a kfb device is in the platform, the desktop must not be run on any other non-kfb graphics devices in the system when performing SunVTS.

Note – To perform `graphicstest` on kfb frame buffers, the desktop must be running on each kfb device in the system.

`graphicstest` Options for XVR-2500

To reach the following dialog box, right-click on the test name in the System Map and select Test Parameter Options. Because graphics test can test multiple types of frame buffers, the test name that is displayed will correspond to the particular frame buffer being tested. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system might not include the device appropriate to this test. Refer to the *SunVTS User's Guide*.

By default, all `graphicstest` options are enabled.

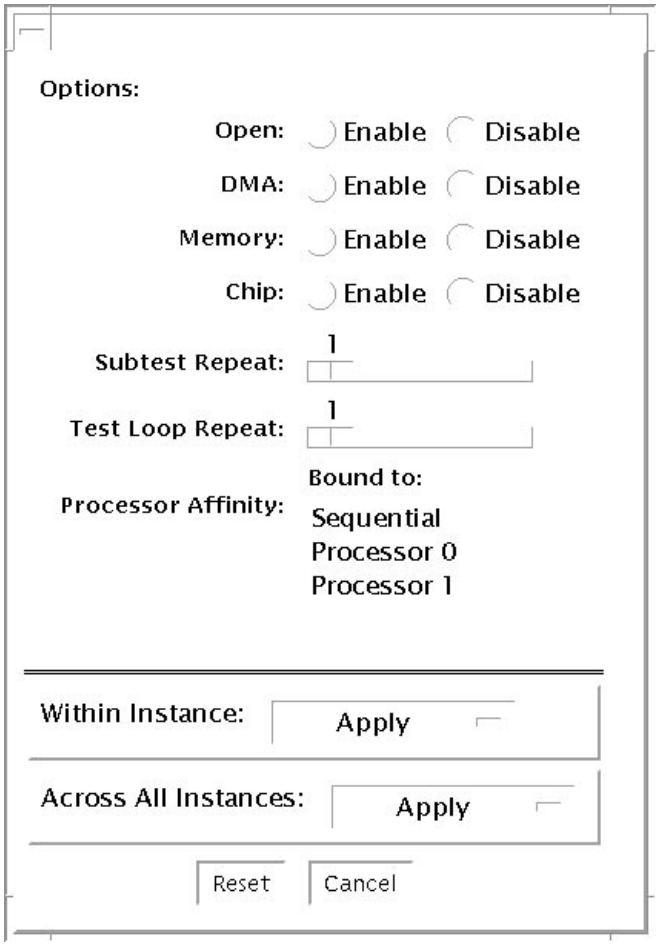

FIGURE 2-1 `graphicstest kfb0` Test Parameter Options Dialog Box

TABLE 2-1 `graphicstest kfb0` Options

Option	Description
Open test	Verifies that the graphics device can be correctly opened. First the device is opened with an open system call. Next the register and frame buffer regions are mapped into the virtual address space of the test process. Next a subset of the read/write resisters are written to and read back from to verify that the register region was correctly mapped. Next selected areas of the frame buffer are written to and read back from to see if the the frame buffer was correctly mapped. This test takes very little time and no progress is displayed.
Memory test	Verifies that all the memory on the graphics card is working. The test opens the device, allocates all the device memory, and then writes to memory using DMA or programmed I/O. The test then reads back the values from memory and checks to see if the correct values were read back.
DMA test	Verifies that the DMA engine on the device is functioning. The test writes data to an area of the frame buffer using DMA, then reads back the same area using programmed I/O and checks if the values are correct. The second part of the test writes data into the frame buffer and reads it back using the DMA engine, checking to see if the correct values were read back.
Chip Test	Exercises the graphics chip on the board to see if it is functioning correctly. The chip test exercises the main data paths through the graphics chip. Commands are sent to the graphics board to initiate a graphics program. Once the program has run the results are read from the frame buffer to verify if they are correct.

`graphicstest` Test Modes for XVR-2500

Due to the nature of graphic tests, reading data from, or writing data to the frame buffer during graphic tests disturbs user operation. For this reason, `graphicstest` is only available in Functional test mode.

TABLE 2-2 `graphicstest` Supported Test Modes

Test Mode	Description
Functional	Performs the entire set of tests offline.

`graphicstest` Command-Line Syntax for XVR-2500

`/opt/SUNWvts/bin/sparcv9/graphicstest` *standard_arguments* **`-o dev=`***device_name* **`open=`***Enable/Disable* **`dma=`***Enable/Disable* **`mem=`***Enable/Disable* **`chip=`***Enable/Disable* **`B=`***n* **`F=`***n* **`S=`***value*.

TABLE 2-3 `graphicstest` Command-Line Syntax

Argument	Description
`dev=`*device-name*	*device_name* is the relative path name of the device being tested with respect to `/dev/fbs`. There is no default.
`open=`*Enable/Disable*	Enables or disables the open test. The default value is Enable.
`dma=`*Enable/Disable*	Enables or disables the dma test. The default value is Enable.
`mem=`*Enable/Disable*	Enables or disables the memory test. The default value is Enable.
`chip=`*Enable/Disable*	Enables or disables the chip test. The default is Enable.
`B=`*n*	Defines the number of times to repeat each test loop. The default value is 1.
`F=`*n*	Defines the number of times to repeat each subtest. The default is one.
`S=`*value*	Sets a bit mask to select which subtests will be used. • Bit 1 – test open • Bit 2 – test dma • Bit 3 – test mem • Bit 4 – test chip For example `-o S=3` would run the open and dma tests only. The default is to run all tests.

`graphicstest` Test Requirements for XVR-300

Disable all screen savers before testing any graphics device. To disable the Solaris screen saver, type the following at a UNIX prompt:

```
# xset s off
```

To turn Display Power Management off, type the following at a UNIX prompt:

```
# xset -dpms
```

For full instructions on testing frame buffers, see "Frame Buffer Tests" on page 8.

`graphicstest` Test Preparation for XVR-300

Perform all of the following procedures prior to performing `graphicstest` to ensure that the test runs as smoothly as possible.

▼ To Run `graphicstest` for XVR-300 With the Window System

1. **Turn Power Management off, if it is enabled. See graphicstest Test Requirements.**
2. **Verify that no other program is running that might modify the screen during the test.**
3. **Verify that you have permission to lock the X server. graphicstest is designed to lock the X server during testing to prevent screen changes.**
4. **Verify that the JDS login window is not displayed during testing.**
5. **If you are running the window system and testing multiple framebuffers, make sure you set Group Concurrency of Graphics to 1.**

▼ To Run `graphicstest` for XVR-300 From the Command Line

1. **Turn Power Management off, if it is enabled. See graphicstest Test Requirements.**
2. **Ensure that no other program is running that might modify the screen during the test.**
3. **Ensure that the frame buffer being tested is not the console device. Console messages might modify the screen.**

`graphicstest` Options for XVR-300

To reach the following dialog box, right-click on the test name in the System Map and select Test Parameter Options. Because graphics test can test multiple types of frame buffers, the test name that is displayed will correspond to the particular frame buffer being tested. If you do not see this test in the System Map, you might need to

expand the collapsed groups, or your system might not include the device appropriate to this test. Refer to the *SunVTS User's Guide*. By default, all graphicstest options are enabled.

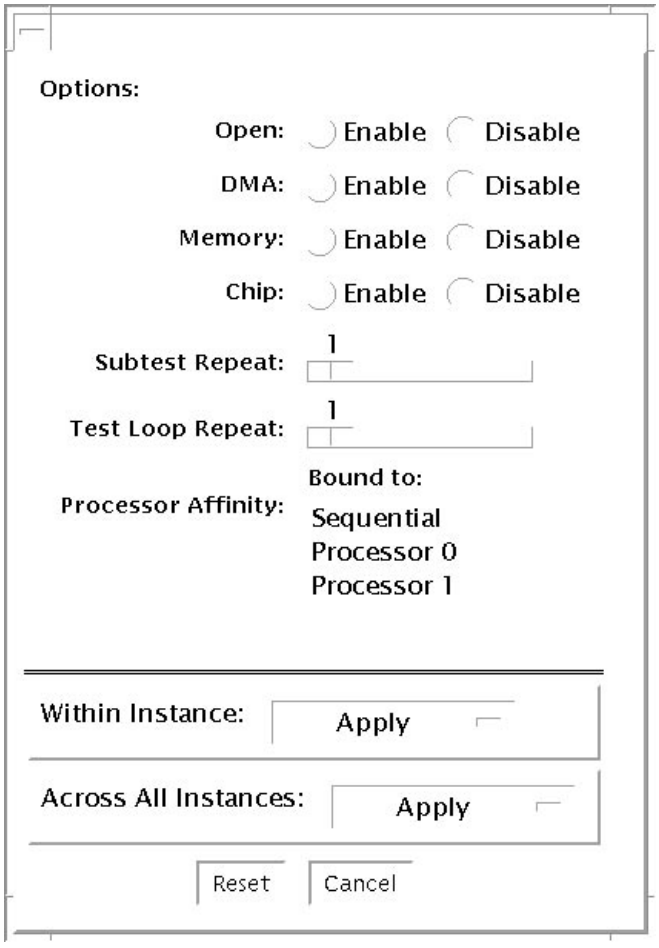

FIGURE 2-2 `graphicstest nfb0` Test Parameter Options Dialog Box

`graphicstest nfb0` Options for XVR-300

TABLE 2-4

Option	Description
Open test	Verifies that the graphics device can be correctly opened. First the device is opened with an open system call. Next the register and frame buffer regions are mapped into the virtual address space of the test process. Next a subset of the read/write registers are read to see if the the frame buffer was correctly mapped. This test takes very little time and no progress is displayed.
Memory test	Verifies that all the memory on the graphics card is working. The test opens the device, allocates the device memory, and then writes to memory using programmed I/O. The video memory is tested by testing 512 bit blocks at a time (8x8 pixel block). Each block is tested in two passes. In the first pass, random data is used, and in the second pass, one's complement of the data used in the first pass is used so that each on-screen video memory location (bit) is tested with a zero and one.
DMA test	DMA is not implemented on this device and hence is not being tested.
Chip Test	Exercises the graphics chip on the board to see if it is functioning correctly. The chip test exercises the main data paths through the graphics chip by sending commands to the graphics board to generate multiple patterns using the line and rectangle primitives.

`graphicstest` Test Modes for XVR-300

Due to the nature of graphic tests, reading data from, or writing data to the frame buffer during graphic tests disturbs user operation. For this reason, `graphicstest` is only available in Functional test mode.

TABLE 2-4 lists the `graphicstest` Supported Test Modes for XVR-300.

TABLE 2-5 Supported Test Modes for XVR-300

Test Mode	Description
Functional	Performs the entire set of tests offline.

`graphicstest` Command-Line Syntax for XVR-300

`/opt/SUNWvts/bin/sparcv9/graphicstest` *standard_arguments* `-o dev=` *device_name* `open=`*Enable/Disable* `dma=`*Enable/Disable* `mem=`*Enable/Disable* `chip=` *Enable/Disable* `B=`*n* `F=`*n* `S=`*value*

TABLE 2-6 `graphicstest` Command-Line Syntax

Argument	Description
`dev=`*device-name*	device_name is the relative path name of the device being tested with respect to `/dev/fbs`. There is no default.
`open=`*Enable/Disable*	Enables or disables the open test. The default value is Enable.
`dma=`*Enable/Disable*	Enables or disables the dma test. The default value is Enable.
`mem=`*Enable/Disable*	Enables or disables the memory test. The default value is Enable.
`chip=`*Enable/Disable*	Enables or disables the chip test. The default is Enable.
`B=`*n*	Defines the number of times to repeat each test loop. The default value is 1.
`F=`*n*	Defines the number of times to repeat each subtest. The default is one.
`S=`*value*	Sets a bit mask to select which subtests will be used. • Bit 1 - test `open` • Bit 2 - test `dma` • Bit 3 - test `mem` • Bit 4 - test `chip` The default is to run all tests.

CHAPTER 3

Ethernet Loopback Test (`netlbtest`)

Note – The `netlbtest` external loopback test is not supported in 1000Mbps mode for the Sun PCI-Express Dual Gigabit Ethernet MMF/UTP Adapter and the `e1000g` driver.

`netlbtest` Description

The `netlbtest` replaces the `gemtest` previously included in SunVTS. It provides functional test coverage of the devices which have device drivers that support the Ethernet loopback test. These devices include `eri` (the Ethernet device in the RIO chip) and `ge` (Gigabit Ethernet), `ce` (GigaSwift Ethernet), `dmfe` (10/100 Mbps Ethernet), and `vca` (Sun Crypto Accelerator 4000). The `netlbtest` runs in loopback (external/internal) mode.

The `netlbtest` uses DLPI RAW mode to talk to the device driver. For the purpose of this test, a packet is defined as an Ethernet header followed by the Ethernet data payload (refer to the IEEE 802.3z standard). The test generates and sends out the desired number of packets (a tunable parameter) and expects to receive the same number of packets through the loopback interface, external or internal. If an error occurs (for example, packet mismatch or timeout), an error message indicating the type of error, its probable cause(s), and recommended action(s) is displayed on the SunVTS console.

The data sent out is generated by a random number generator, and put into a data buffer. Each time a packet is sent, it is selected from a different starting point of the data buffer, so that any two consecutively transmitted packets are not the same.

Note – Do not run `nettest` and `netlbtest` at the same time or the tests might fail.

A new debugging capability has been added in `netlbtest`. After one packet is not received, four more packets are transmitted. If all of the packets are not received within the timeout time, the test will stop with the error message, `timed out for receiving ...`. If up to four packets are missing, the test will stop with an error message, `Missed %d packet(s)...`. If a packet is received late and the current transmitted packet is not received, the test will stop with a warning message, `Packet delay...`. If the packets arrived late but within five times the timeout value and no packet is missing, the test passes.

`netlbtest` Test Requirements

You must have the Ethernet card and the device driver installed, a loopback connector in place (external loopback only), and Intervention mode enabled before running `netlbtest`. `netlbtest` cannot run and does not appear in the GUI if the network interface is connected to a live network. `netlbtest` also requires that the Ethernet device be configured offline before running the test. Use the `ifconfig(1M)` command to bring the Ethernet device down before running `netlbtest`. Enter the following commands to bring the interface down:

```
# ifconfig interface down
# ifconfig interface unplumb
```

To run `netlbtest`, a loopback connector must be connected to the Ethernet interface when performing the external loobpback test only. A loopback connector provides the network interface driver the necessary link for testing, while maintaining isolation from a live network.

Note – The loopback connector is *not* required for both internal and external tests of the Ethernet device. The loopback connector is required for the external loopback test only.

The loopback cable for `ge` and Sun GigaSwift Ethernet MMF adapter (`ce` fiber) is based on the following specifications— multimode, duplex, 62.5/125 micron, sc connector, 850nm. The cable can be made by splitting a standard fiber optic cable in two. The two ends of the cable should be connected to the TX and RX ports of the adapter (the order does not matter), thus forming a loop.

The loopback connector for the `eri` device is a standard RJ-45 connector. See Appendix A in the *SunVTS User's Guide* for the diagram. The loopback connector for a Sun GigaSwift Ethernet UTP adapter (`ce` copper) is a standard RJ-45 with all 8 pins connected. See Appendix A of the *SunVTS User's Guide* for the diagram.

`netlbtest` Options

To reach the following dialog box, right-click on the test name in the System Map and select Test Parameter Options. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system might not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

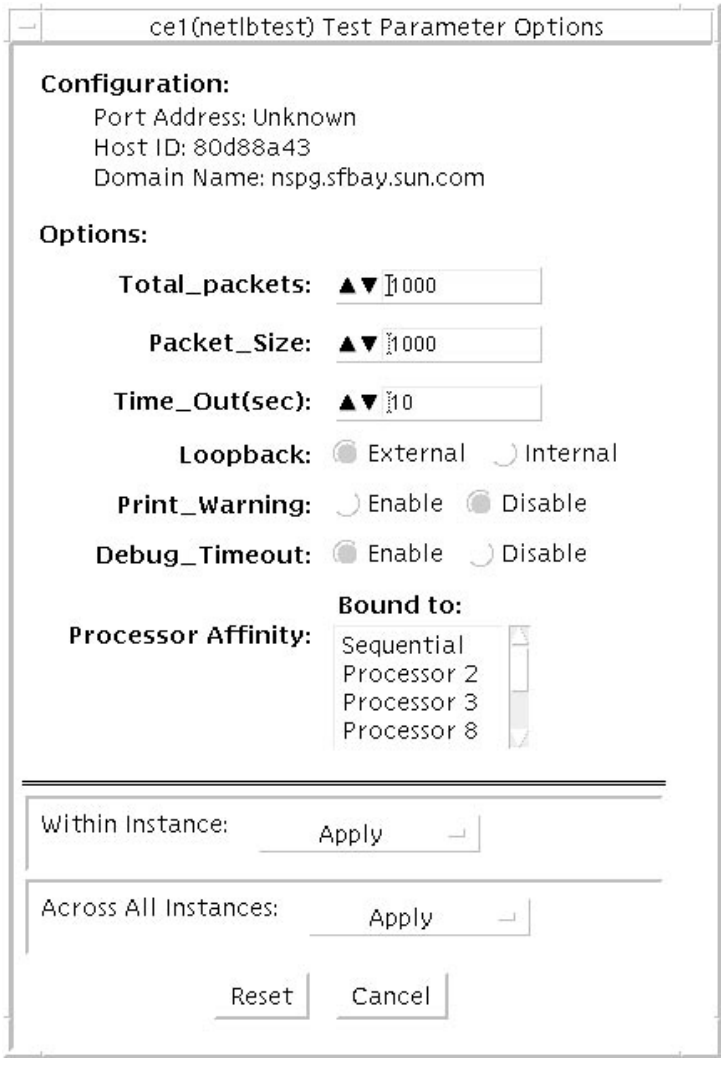

FIGURE 3-1 `netlbtest` Test Parameter Options Dialog Box

Refer to TABLE 3-1 for test parameter descriptions.

TABLE 3-1 `netlbtest` Options

`netlbtest` **Options**	**Description**
Configuration	Specifies the port address, host ID, and domain name of the system under test.
Total Packets	Specifies the total number of the packets to send. The default number of packets is 1,000. The maximum number of packets is 100,000,000.
Packet size	Determines the size (in bytes) of the packets to be transmitted. 60 <= packet size <= 1514. The default packet size is 1000 bytes.
Time_Out(sec)	Determines the amount of time (in seconds) that `netlbtest` can wait to receive packets. If no packets are received within this time frame, `netlbtest` reports an error message. The range for timeout is from 1 to 1,000 seconds.
Loopback	Determines the external and internal loopback mode. The default setting is internal loopback mode.
Print_Warning	Enables or disables the printing of warning messages. The default setting is Disable.
Processor Affinity	Binds the test to a specific processor. If no processor is specified, the test migrates between processors. This option is only available on multiprocessor systems.
Debug_Timeout	Enables or disables the debugging feature of `netlbtest`. The default setting is Disable. The range for timeout is from 1 to 1,000 seconds.

`netlbtest` Test Modes

TABLE 3-2 `netlbtest` Supported Test Modes

Test Mode	**Description**
Functional	Runs the full set of subtests. The host must not be connected to the network through the intended test device.

Since `netlbtest` requires a loopback connector for the external loopback test, it can only be selected when Intervention mode is enabled.

`netlbtest` Command-Line Syntax

`/opt/SUNWvts/bin/netlbtest` *standard-arguments*
`-o dev=`*device*`,` **`tpkts=`***n*`,` **`pksz=`***pkt-size*`,` **`lb=`***Internal*`,` **`warn=`***Disable*`,`
`timeout=`*number-of-seconds*

TABLE 3-3 `netlbtest` Command-Line Syntax

Argument	Description
`dev=`*device-name*	Specifies the device to test such as `ge0` or `eri0`.
`tpkts=`*n*	[1...100000], count of packets to loopback. Packets number can be 1 to 100,000. The maximum number of packets is 100,000,000.
`pksz=`*pkt-size*	Determines the size (in bytes) of the packets to be transmitted. 60 <= packet size <= 1514. The default packet size is 1000 bytes.
`lb=`*Internal*	Selects internal (or external) loopback mode.
`warn=`*Disable*	Enables or disables printing of warning messages.
`timeout=`*number-of-seconds*	Determines the amount of time (in seconds) that `netlbtest` can wait to receive packets. If no packets are received within this time frame, `netlbtest` reports an error message. The range for timeout is from 1 to 1,000 seconds.