SunVTS™ 6.0 Patch Set 2 Documentation Supplement for SPARC® Platforms

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2947-10
August 2005, Revision A

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

Contents

CHAPTER 1

Introduction

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

This manual is a supplement to the SunVTS™ 6.0 documentation and describes new features, tests, and test enhancements that are developed in the SunVTS 6.0 patch releases. The items described in this document are provided in the SunVTS 6.0 Patch Set 2 software.

For overall SunVTS features, test configuration modes, interfaces, and options refer to the *SunVTS 6.0 User's Guide*. Refer to the *SunVTS 6.0 Test Reference Manual* for detailed information on SunVTS test software and the full collection of tests released with SunVTS 6.0.

Refer to the latest version of the *SunVTS 6.0 Patch Set 2 Release Notes* (819-2949-10) for important release information and open issues—this document is available at: `http://www.sun.com/documentation`

New Features

The following tests are enhanced in this release:

- CD DVD Test (`cddvdtest`) – described in Chapter 2
- Floating Point Unit (FPU) Test (`fputest`) – described in Chapter 6
- System Test (`systest`) – described in Chapter 15
- Tape Drive Test (`tapetest`) – described in Chapter 16

SunVTS Overview

SunVTS is composed of many individual tests that support testing of a wide range of products and peripherals. Most of the tests are capable of testing devices in a 32-bit or 64-bit Solaris environment.

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

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

Such flexibility means that the proper test modes and options need to be selected to maximize its effectiveness.

Note – When an error occurs in VTS 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 manual.

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 6.0 User's Guide* for installation information.

Test Requirements

SunVTS 6.0 and 6.0 patch releases are supported in the Solaris 10 Operating System and future compatible Solaris releases.

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 book.

Getting SunVTS Version Information

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

Collection of SunVTS Tests

Many individual tests make up the collection of tests in the SunVTS application. 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.

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

32-Bit and 64-Bit Tests

Because each test is a separate program, you can run individual tests directly from the command line. When this is done, care must be taken to run the appropriate test (32-bit or 64-bit) that corresponds to the operating system that is running (32-bit or 64-bit). This is done by running tests from specific directories as follows:

- 32-bit tests—`/opt/SUNWvts/bin/`*testname*
- 64-bit tests—`/opt/SUNWvts/bin/sparcv9/`*testname*
  - The test is an actual 64-bit binary test if *testname* is a binary file.
  - The test is a 32-bit test capable of running in the 64-bit environment if *testname* is a symbolic link.

Note – The `SUNWvtsx` package must be installed for 64-bit SunVTS support. For more information on SunVTS packages and installation procedures refer to the *SunVTS 6.0 User's Guide*.

If you use the `sunvts` command to run SunVTS, SunVTS automatically allocates 32-bit or 64-bit tests based on the 32-bit or 64-bit Solaris operating environment that is running. Therefore, the only time that you need to be concerned with the 32-bit or 64-bit operation is when you run the SunVTS kernel or SunVTS tests from the command line.

If you are not sure which operating system is running, refer to the Solaris System Administration manuals. In Solaris 8 2/02 and Solaris 9, the following command can be used 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 various interfaces: The CDE graphical user interfaces, 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 7). 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 interfaces (GUIs)	Users can select tests and test options by pointing and clicking with a mouse button in the CDE interface.
TTY interface	Users can run SunVTS from a terminal or modem attached to a serial port. This feature requires that users use the keyboard instead of the mouse, and it displays one screen of information at a time.
Command-line execution	Users can run each of the SunVTS tests individually from a shell tool command line using the command-line syntax. Each test description in this book contains the corresponding command-line syntax.

Note – To increase or decrease a numeric value in a SunVTS CDE dialog box, you can use either the up or down arrows, or type a new value in the text box and press Return. Select Apply to apply all dialog box changes.

Running a Test From a User Interface

The common way to run SunVTS testing is through a SunVTS user interface—CDE 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 6.0 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 lower set of buttons are generic and are described below.

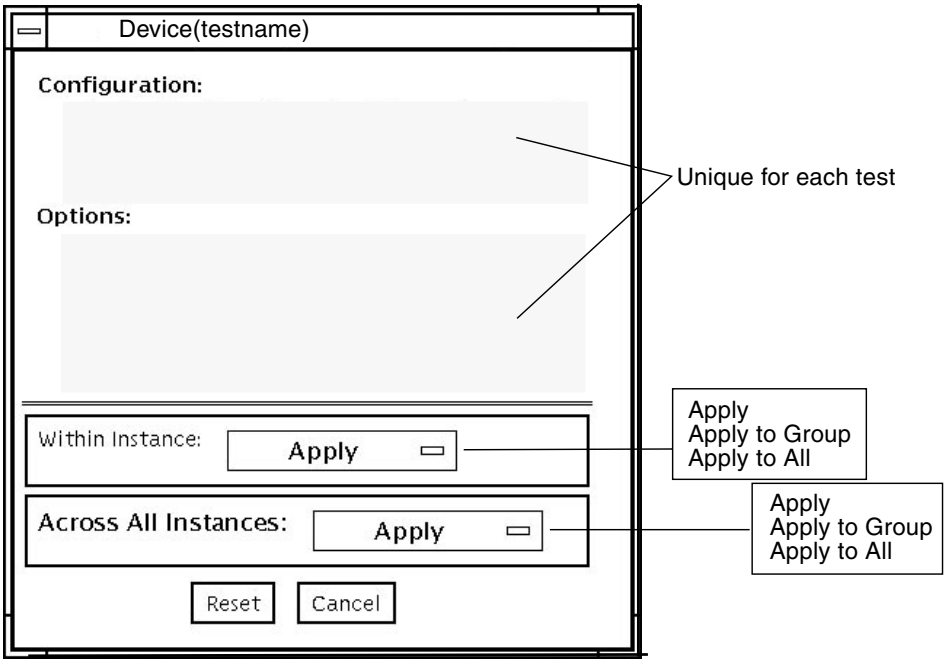

FIGURE 1-1 Test Parameter Options Dialog Box (CDE)

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, or • to all devices within this group with Apply to Group, or • 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, or • to all devices within this group with Apply to Group, or • 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 menu.
Cancel	Ignores any changes made to option values and closes the test parameter option menu.

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*. If a test is not present in this directory, then it may only be available as a 32-bit test. For more information refer to the *SunVTS 6.0 Test Reference Manual*.

Standard Command-Line Arguments

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

TABLE 1-3 Standard Command-Line Arguments

Argument	Description
`-x`	Runs the test in exclusive test mode. This mode assumes that the test has exclusive access to the device and the system. The testing done in exclusive mode is usually more stressful compared to functional mode. Also, running a test in exclusive mode usually assumes exclusive access to all resources and assumes no other SunVTS test is running at the same time.
`-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.
`-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 test Trace mode and displays messages that track function calls and sequences currently in use by the test code. The default is false.
`-e`	Runs the test in Stress mode by increasing the system load. 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.
`-n`	Runs the test in Connection mode. See the note below. The default is false.
`-f`	Runs the test in full Functional test mode. This mode assumes that the test has complete control of the device under test. See the note below. The default is false.

TABLE 1-3 Standard Command-Line Arguments *(Continued)*

Argument	Description
`-i` *number*	Defines the number of instances for scalable tests.
`-w` *number*	Defines to which instance the test is assigned; this option is for scalable tests.
`-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

There are test-specific arguments, as described in TABLE 1-4. Test-specific arguments follow the format specified in the `getsubopt(3c)` man page. For information about test-specific arguments refer to the specific test chapter in this book.

TABLE 1-4 SunVTS Test-Specific Arguments

Argument	Description
`-o`	Separate each test-specific argument by commas, with no space after the comma. For example: `#`**`./sample -v -o dev=/dev/audio,volume=78`** The test option format is specified by the man page `getsubopt`(3C).

SunPCi-3 Card Support

`sunpci2test` now supports the SunPCi-3 cards. Solaris 10 supports SunPCi-3 Version 3.2.2 with Patch 118591-01 only. Solaris 10 does not support the SunPCi-2 card.

Testing Frame Buffers

Before running a frame buffer test, determine whether the test requires frame buffer locking. Not all frame buffer tests have a locking option. Some tests set the lock automatically. Check the test chapter for each individual test to see if this step is needed. If locking is required, you can set the lock in one of two ways:

- If you are using the CDE SunVTS interface, go to the Option menu of the graphic test and select Enable for the frame buffer locking option.
- If you are working from the command line, you can enable frame buffer locking with the `lock=e/d` option. For example, to run the generic frame buffer test (`fbtest`) with a locked frame buffer, enter:

```
# ./fbtest -o dev=pfb0,lock=enable
```

(See the test command line argument descriptions in the individual test chapters.)

Caution – If frame buffer locking is disabled (unlocked) on frame buffers that are running `vtsui`, or if you move the mouse, you will receive false error messages. Even a slight mouse movement can cause a test to fail.

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 User Collection. This document is available at: `docs.sun.com`.

Caution – If you are using the GUI 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 CDE or GNOME). 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.

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.

CHAPTER 2

Optical Disk Drive Test (cddvdtest)

cddvdtest verifies the functionality of optical disk drives.The disktest probe detects the media type in the drive and shows the test options for the media found. If inserted media is not supported by the drive, the disktest probe shows an error and registers options for CD-ROM as default.

Note – cddvdtest is a newly consolidated test which is used to test the optical media drives such as the CD-ROM, DVD-ROM and CD-DVD-RW drives. cddvdtest supports the same set of options for both SPARC and x86.

Volume Management

cddvdtest tests the optical drive(s) even if the Volume Manager is not running. If the Volume Manager is running and no media is installed in the drive(s), SunVTS prompts you to install media in the drive before selecting the test. The test fails if you try to run it without media in the drive.

Note – When testing rewritable media, the media can be either blank or contain the SunVTS test data. When testing the write-once media, the media (such as CD-R) has to be blank at the start to run the write test. Such media could still run multiple passes of the test because after the first write test, the subsequent invocations will treat the media as Read Only and perform the test accordingly.

For CD-ROM and DVD-ROM drives, the test checks the unit by reading either the CD or DVD. For CD-ROM, each track is classified as follows:

- Mode 1 uses error detection/correction code (288 bytes).
- Mode 2 uses that space for auxiliary data or as an audio track.

For rewritable CD media, the test can write one or more tracks in one test pass. `cddvdtest` writes tracks on next available space on the media. If the media is full, `cddvdtest` automatically erases the whole media and starts the next test pass from the beginning of the media.

For rewritable DVD media, the test writes only one track in one test pass (because there is only one track in DVD format). `cddvdtest` blanks the media when starting the test, if the media is not already blank.

For rewritable media types, the test verifies write, read, and other supporting functions of CD and DVD RW drives. The supported media include the following:

- CD-R (must be blank)
- CD-RW (can either be blank or contain the SunVTS test data)
- DVD-R (must be blank)
- DVD+R (must be blank)
- DVD-RW
- DVD+RW

Note – `cddvdtest` is not a scalable test.

`cddvdtest` Hardware and Software Requirements

`cddvdtest` has different set of test requirements based on the media type as described below.

CD-ROM and DVD-ROM

The drive must have the appropriate CD-ROM or the DVD-ROM media before performing the test.

When a CD-ROM is loaded in the drive, `cddvdtest` uses CD-ROM specific options to test the drive. When a DVD-ROM is loaded, the test uses DVD-ROM specific options. Whenever you change the media in the drive you must perform a reprobe (refer to the *SunVTS User's Guide* for details) so the SunVTS kernel associates the correct test options based on the media that is loaded in the drive.

CD-RW and DVD-RW

When testing rewritable media, reprobing is required if the media is changed. To prevent accidentally erasing useful data on a media used for testing, cddvdtest accepts the test media only if it is blank or it contains SunVTS test data (data that is written by `cddvdtest` itself). The test checks these conditions at probe time and at the start of test.

In case of non-blank media OR the media with non-SunVTS test data, the media must be blanked first by using the `cdrw` utility. Because -R or +R media can be written only once, only a blank media should be used for write testing. Such media could still run multiple passes of the test because after the first write test, the subsequent invocations will treat the media as Read Only and perform the test accordingly.

To prevent media corruption, if stopped during a write, finalize, format, or erase, the test posts a Warning message and continues until the current operation is completed. Wait until the test completes before doing any operation on the drive.

Do not stop the test in the middle of a writing operation. Doing so may cause damage to the media in some cases. It is better to set a limited number of passes for `cddvdtest`, instead of setting Max Passes=0 (unlimited) and stop the test manually. If a media is damaged, you should blank the media with `cdrw` command.

Note – DVD+RW media cannot be blanked.

The default Delay between two passes for the Read Write media is three minutes. This setting is intended to make the test run less passes in long hours testing to preserve the media. It is also intended to give plenty of time to stop the test between passes.

`cddvdtest` Subtests

`cddvdtest` has different subtests based on the media type as described below.

CD-RW and DVD-RW

TABLE 2-1 `cddvdtest` Subtests for CD-RW and DVD-RW Media

Subtest	Description
Blank	Erases data on DVD-RW media and CD-RW media if not blank. This subtest does not apply to DVD+RW media because this media type can not be blanked.
Simulation	Performs a write track with Laser turned off. This mode tests writing function without data written to the media.
Write	Writes to media with predefined data patterns of 0..ff hex. For CD-RW media, track can be specified as data or audio track.
Read	Reads the written data.
Compare	Compares write/read data, reports failure if miscompared.
Read Disk/Track	Reads and shows Disk Table of Content (TOC).
Eject	Ejects media.

`cddvdtest` Options

To reach the dialog boxes below, 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

`cddvdtest` has different test options based on the media type as described below.

CD-ROM Test Options

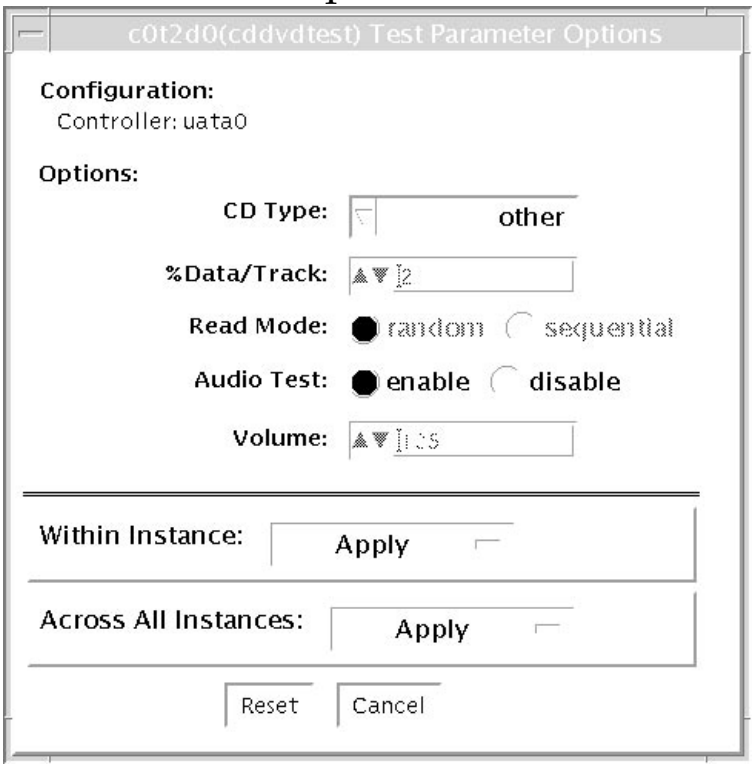

FIGURE 2-1 `cddvdtest` Test Parameter Options Dialog Box for CD-ROM

TABLE 2-2 `cddvdtest` Options for CD-ROM

Option	Description
CD Type	The types of compact discs that can be tested are listed in the CD Type menu. The choices are: pdo, multi-session, or other the default CD type is other). In the Connection test mode, this option has a default value of other. Note - Your choice must correspond with the disc ussed for testing.
% Data/Track	Tests a percentage of data on each track. Type a value between 0 and 100 in this field to indicate the percentage. In the online and connection tests this option has a canned value of 2%.
Read Mode	cddvdtest reads the CD either in Random or Sequential mode. In Random mode, data blocks are read from random track positions; in Sequential mode, data blocks are read in sequence. For both modes, the total number of blocks read is determined by the %_of_data option. In the online and Connection tests this option has a canned value of random.
Audio Test	Enables or disables the audio test. You must connect headphones or a speaker to the audio jack on the CD player to hear audio output. In the Connection test, this option has a default value of disable.
Volume	Adjusts the volume. Type a value between 0 and 255 in this field. In the online and connection tests this option has a default value of 125.

DVD-ROM Test Options

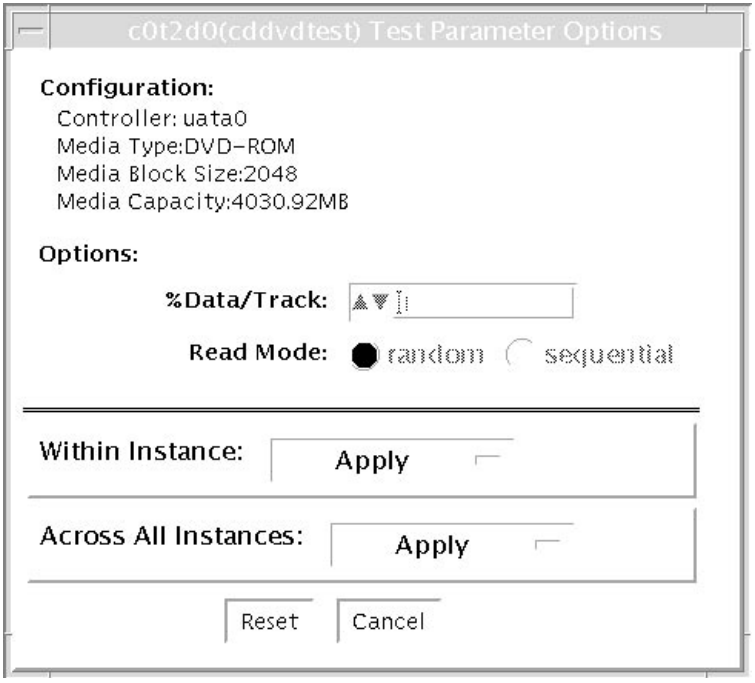

FIGURE 2-2 `cddvdtest` Test Parameter Dialog Box for DVD-ROM

TABLE 2-3 `cddvdrwtest` options for DVD-ROM

Option	Description
% Data/Track	Specifies a percentage of data to test for each track. Use a number between 0 and 100.
Read Mode	Specifies either Random or Sequential mode. Random mode reads data blocks from random track positions. Sequential mode reads data blocks in sequence. For both modes, the total number of blocks read is determined by the % Data/Track value.

CD-RW Test Options

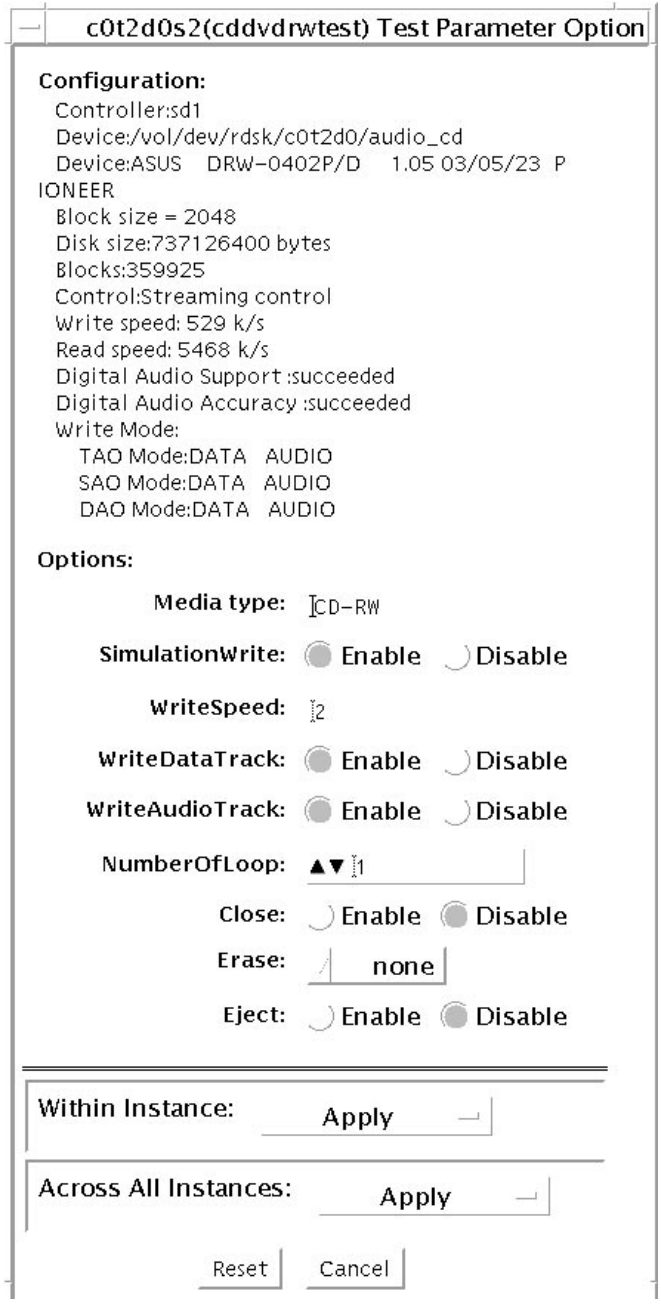

FIGURE 2-3 cddvdtest Test Parameter Options Dialog Box for CD-RW

TABLE 2-4 cddvdtest Options for CD-RW

Option	Description
SimulationWrite	Disable/Enable Simulation Write.
WriteSpeed	Speed entered in terms of *nX*. Speed will be set to closest approximation of *nX* allowed by device.
WriteDataTrack	Write a Data Track in one Test Loop.
WriteAudioTrack	Write an Audio Track in one Test Loop.
NumberOfLoop	Number of loop in one test pass comprising Data and Audio Track if enabled.
Close	Close after writing. Can not write more track on media.
Erase None	No erase All: Erase whole disk.
Eject	Eject the disk when test completed. Do not enable this option in a multiple pass test session.

DVD-RW Test Options

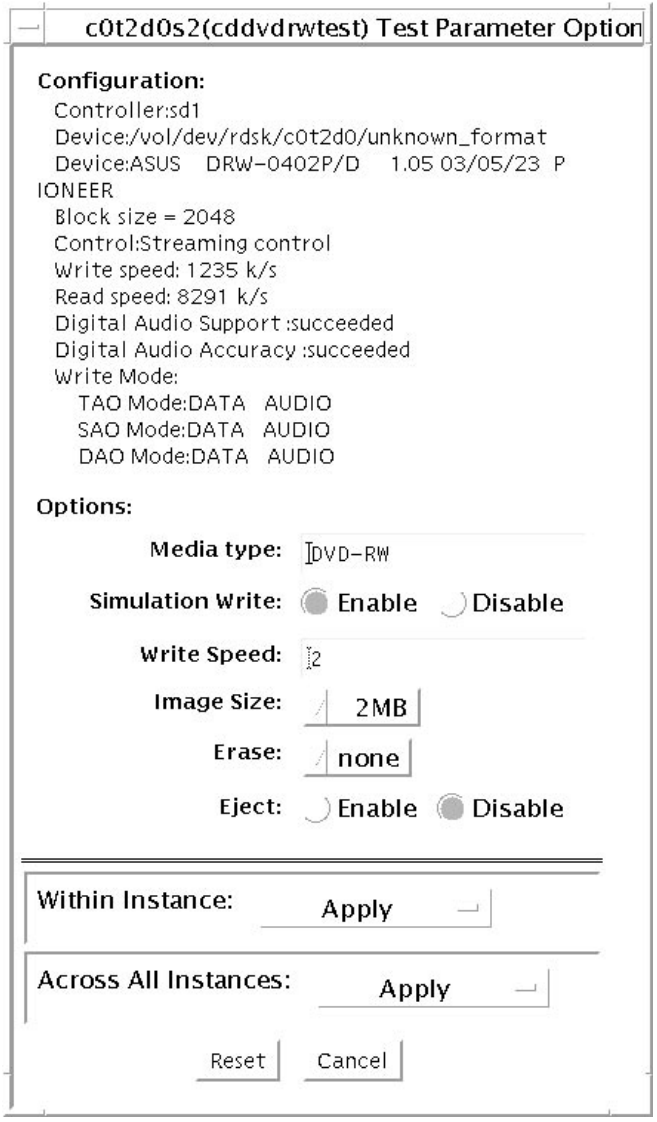

FIGURE 2-4 `cddvdtest` Test Parameter Options Dialog Box for DVD-RW

TABLE 2-5 `cddvdtest` Options for DVD-RW

Options	Description
SimulationWrite	Disable/Enable Simulation Write.
WriteSpeed	Speed entered in term of *nX*. Speed will be set to closest approximation of *nX* allowed by device.
ImageSize	Track Size, how much data is written: 2MB, 10MB, 2GB, 4GB.
Erase None	No erase.
Fast	Erase with "fast" option. Only Table of Content is erased.
All	Erase with "all" option. Erase whole disk surface.
Eject	Eject the disk when test completed. Do not enable this option in a multiple pass test session.

`cddvdtest` Supported Test Modes

This test supports Connection and Functional test modes for all media types. The test behavior in each mode for the different media types is explained below.

CD-ROM Test Modes

TABLE 2-6 Supportted Test Modes for CD-ROM

Test Mode	Description
Connection	In this mode, `cddvdtest` verifies that a CD-ROM drive is connected to and configured in the system.
Functional	In this mode, the test registers a failure if the device is found to be busy. This is because SunVTS tests make the assumption that all the resources will be available for testing in the Functional test and the unavailability of a device is interpreted as an indication of a fault condition.

DVD-ROM Test Modes

TABLE 2-7 Supported Test Modes for DVD-ROM

Test Mode	Description
Connection	Requests and displays information from the drive and reads two blocks of data from the media to confirm connectivity. An error is reported if no media is loaded in the drive.
Functional	Requests and displays information from the drive, then reads data from the media based on the options that are set in the Test Parameter Options Dialog Box. An error is reported if no media is loaded in the drive.

CD-RW and DVD-RW Test Modes

TABLE 2-8 Supported Test Modes for CD-RW and DVD-RW

Test Mode	Description
Connection	Shows basic drive information and supporting modes. Shows disk Table of Content (TOC).
Functional	The following subtests are done in sequence:Erase - Erases data if DVD-RW media is not blank, or CD-RW media if full. • Simulation (optional) - Performs a write track with Laser turned off. This mode tests writing function without data written to the media. • Write - Writes to media with predefined data patterns of 0..ff hex. For CD media track can be specified as data or audio track. • Read - Reads the written data. • Compare - Compares write/read data, reports failure. • if miscompared. • Blank media (optional). • Read Disk/Track - Reads and shows Disk Table of Content. • Eject (optional).

`cddvdtest` Command-Line Syntax

`cddvdtest` has different command line syntax based on the media type as described below.

CD-ROM Command-Line Syntax

/opt/SUNWvts/bin/sparcv9/cddvdtest *standard_arguments* **-o dev=** *raw_device_name*, mode=*mode*, read=random|sequential, data=*%_of_data*, vol=*volume*, audio=enable|disable, type=*CD_type*

TABLE 2-9 CD-ROM Command-Line Syntax

Argument	Description
dev=*raw_device_name*	Specifies the name of the raw device to be tested.
read=*random*|*sequential*	Indicates random or sequential read access.
data=*%_of_data*	Sets the percentage of data to be tested. You can specify 0 to 100 percent.
vol=*volume*	Controls the audio volume. You can specify 0 through 255; the default is 255.
audio=*enable*|*disable*	Enables or disables the audio test. You must connect headphones or a speaker to the audio jack on the CD player to hear audio output.
type=*CD_type*	Specifies the type of CD used for the test. The choices are: pdo, multi-session, sunos and other; the default is other.

DVD-ROM Command-Line Syntax

/opt/SUNWvts/bin/sparcv9/cddvdtest *standard_arguments* -o dev= *device_name*, read=random|sequential, data=*%_of_data*

TABLE 2-10 DVD-ROM Command-Line Syntax

Argument	Description
dev=*device_name*	Specifies the name of the device to test, for example /dev/rdsk/cntndn.
read=random|sequential	Indicates random or sequential read access.
data=%_of_data	Sets the percentage of data to test. Use a number from 0 to 100.

CD-RW Command-Line Syntax

`/opt/SUNWvts/bin/sparcv9/cddvdtest` *standard_arguments* `-o dev=`*cntndnsn*`, media=CD-RW, nosim, speed=`*n*`, nodata, noaudio, loop=`*n*`, close, erase={none, all}, eject`

TABLE 2-11 CD-RW Command-Line Syntax

Argument	Description
`dev=`*cntndnsn*	Specifies the device under test.
`media=CD-RW`	Specifies media.
`nosim`	Disables Simulation Write
`speed=`*n*	Specifies the speed entered in terms of *nX*.
`nodata`	Disables Data Track test.
noaudio	Disables Audio Track test.
`loop=`*n*	Specifies 1 to 40, the number of loops in one test pass.
close	Closes track after test after the test, no track can be added.
`erase={none, all}`	`none` - Do not erase media after test complete. `all` - Erase entire disk.
`eject`	Ejects disk after test completed.

DVD-RW Command-Line Syntax

`/opt/SUNWvts/bin/sparcv9/cddvdtest` *standard_arguments* `-o dev=`*cntndnsn*`, media={DVD-RW, DVD+RW}, nosim, speed=`*n*`, imagesize={2MB,10MB,2GB,4GB}, erase={none, fast, all}, eject`

TABLE 2-12 DVD-RW Command-Line Syntax

Argument	Description
`dev=`*cntndnsn*	Specifies the devise under test.
`media={DVD-RW, DVD+RW}`	Specifies media.
`nosim`	.Disables Simulation Write
`speed=`*n*	.Specifies the speed entered in terms of *nX*.

TABLE 2-12 DVD-RW Command-Line Syntax *(Continued)*

Argument	Description
imagesize={2MB,10MB,2GB,4GB}	.Specifies the image size used in write/read test
`erase={none, fast, all}`	.`none` - Do not erase media after test complete `fast` - Erase the last track added `all` - Erase entire disk
`eject`	Ejects the media.

CHAPTER 3

Chip Multi-Threading Test (cmttest)

cmttest verifies the proper functioning of the multiprocessor hardware with multiple cores in one CPU. cmttest tests the path between the cores on the same CPU in addition to performing CPU specific testing. cmttest uses the Cache Coherence, Shared Memory, and RAM subtests. The Cache Coherence subtest is used to test the coherence among all of the Cores in a CMT (Chip Multiprocessor). The Shared Memory subtest is used to test the shared memory among all the cores in a CMT. The RAM subtest is used to test the memory. The RAM subtest covers TLB, MMU, and bus balancing. The Interrupt subtest covers the intra-core and inter-core interrupt generation and receiving logic.

Only one cmttest is registered and cmttest is present under the logical name Processor(s). There is no physical name provided. The probe routine of cmttest probes all CMTs in which at least two cores are online.

Note – cmttest was named cmptest in previous SunVTS releases.

cmttest Options

To reach the dialog box below, 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

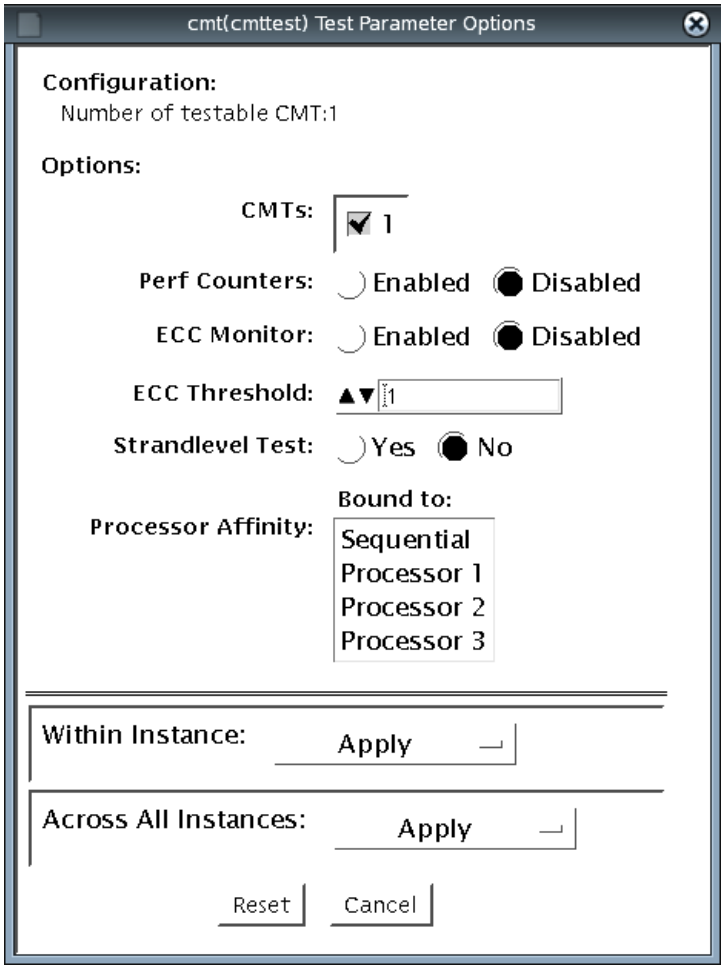

FIGURE 3-1 `cmttest` Test Parameter Options Dialog Box

The processors that can be tested are listed in the Configuration area of the menu. You can enable or disable the multiprocessing test for individual processors on this menu.

The options listed in the following table can be run alone or concurrently with other options.

TABLE 3-1 `cmttest` Options

`cmttest` **Options**	**Description**
CMTS	You can test specific CMTs by clicking Select on the check boxes to enable or disable each CMT. A check mark indicates the CMT is enabled for testing. The default setting is all CMTs enabled.
Perf Counters	By default performance monitoring is disabled. When performance monitoring is Enabled tests print memory bandwidth achieved while testing. Right now only ram subtest has the counters built in. Bandwidth calculations assume that all banks corresponding to all cpus are present and had same number of reads and writes. (Note: Perfcounter monitoring can be done on SUNW,UltraSPARC-IV processors, If user tries to enable perfCounter, and perfcounters are not supported ,on cpus the appropriate warning message is displayed, with disabling the perfcounter.)
ECC Monitor	This option is used to Enable or Disable ECC error monitoring. The default option is Disabled.
ECC Threshold	Range is [0-255].This determines how many correctable ECC errors occurred in the elapsed time before cmttest reports a test failure. The default threshold value is 1.
Strand Level Test	For processors like Niagara with multiple strands per core, the test selects one strand/core to perform testing. This behavior can be changed by setting this option to Yes. The default setting No does the most optimal testing and hence should be used in general.The interrupt subtest will not be run if this option is set to Yes. This option doesnt have any effect on non-multistranded processors such as the UltraSPARC IV+.

cmttest Test Modes

TABLE 3-2 cmttest Supported Test Modes

Test Mode	Description
Functional	The Functional test mode is supported.
Exclusive	Performs the full test.

cmttest Command-Line Syntax

For 32-bit configurations:

/opt/SUNWvts/bin/cmttest *standard_arguments*
-o cmts=*0+1+2...*, **em=***Enabled* | *Disabled*, **threshold=***[0-255]*, **perf=***Enabled* | *Disabled*, strandleveltest=*Yes* | *No*

For 64-bit configurations:

/opt/SUNWvts/bin/sparcv9/cmttest *standard_arguments*
-o cmts=*0+1+2...*, **em=***Enabled* | *Disabled*, **threshold=***[0-255]*, **perf=***Enabled* | *Disabled*, `strandleveltest=`*Yes* | *No*

TABLE 3-3 `cmttest` Command-Line Syntax

Arguments	Description
cmts=*0+1+2...*	*0, 1, 2,...* mentions the CPU ID of any one core of the CMTs to be tested. To display on the GUI, CPU ID of core 0 is taken as the identifier for a CMT. For displaying the Error/INFO/LOG messages, the CPU ID of the core 0 is used. For multistranded processors such as the Niagara, if more than one strand from a core is selected, only one of them is used for testing. To override this behavior,the option `strandleveltest` must be set to Yes.
em=*Enabled* | *Disabled*	Enable or Disable ECC error monitoring. The default value is Disabled.
threshold=*[0-255]*	Determines how many correctable ECC errors can occur in the elapsed time before `cmttest` reports a test failure. The range is [0-255]. The default value is 1.
perf=*Enabled* | *Disabled*	By default performance monitoring is *Disabled*. When performance monitoring is *Enabled* tests print memory bandwidth achieved while testing. Only the RAM subtest has the counters built in. Bandwidth calculations assume that all banks corresponding to all CPUs are present and have the same number of reads and writes. Note: Perfcounter monitoring can be done on SUNW, UltraSPARC IV processors. If you try to enable perfCounter, and the perfcounters are not supported on the CPUs, the appropriate warning message is displayed and the perfcounter is disabled.
strandleveltest=*Yes* | *No*	This option is used for strand level testing for multi-stranded processors such as the Niagara. Selecting Yes for this option, selects all strands for testing.

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 may only be available as a 32-bit test. For more information refer to the “32-Bit and 64-Bit Tests” section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

CHAPTER 4

Cryptographics Test (`cryptotest`)

The Encryption Framework in Solaris 10 provides a user level API for access to cryptographic accelerators. This API is based on the PKCS#11 standard. Cryptographic accelerators are referred to as PKCS#11 tokens, and each cryptographic algorithm the token accelerates is referred to as a mechanism.

`cryptotest` tests the mechanism supported by PKCS#11 tokens in Solaris 10.

`cryptotest` supports the Sun Cyrpto Accelerator 500, Sun Cyrpto Accelerator 1000, Sun Cyrpto Accelerator 4000, Niagara Crypto Provider, and all future cryptographic accelerators developed for Solaris 10 onward. PKCS documents and information are available at: `hppt://www.rsasecurity.com/rsalabs/PKCS`

TABLE 4-1 Definitions of the Mechanisms Tested by `cryptotest`

Algorithm	Description
DSA	Digital Signature Algorithm
DES	Data Encryption Standard as defined in FIPS PUB 46-3
MD5 RSA	Data Security MD5 message-digest algorithm.
RSA	Public key cryptosystem.
SHA1	The Secure Hash Algorithm.
RNG	Random Number Generator Algorithm.

cryptotest Subtests

TABLE 4-2 cryptotest Subtests

Subtest	Description
DES	Tests DES bulk encryption
3DES	Tests 3DES bulk encryption
RSA	Tests RSA public and private keys
DSA	Tests DSA signature verification
RNG	Tests random number generation

cryptotest Options

To reach the dialog box below, 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 framebuffer being tested. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the SunVTS User's Guide.

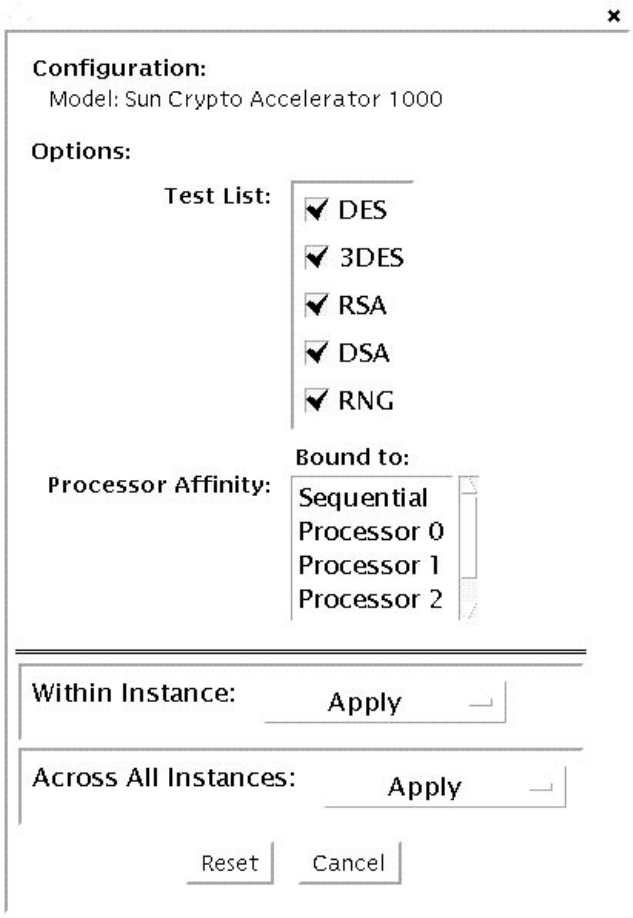

FIGURE 4-1 `dcatest` Test Parameter Options Dialog Box

TABLE 4-3 `dcatest` Options

Option	Description
DES	Tests DES bulk encryption
3DES	Tests 3DES bulk encryption
RSA	Tests RSA public and private keys
DSA	Tests DSA signature verification
RNG	Tests random number generation

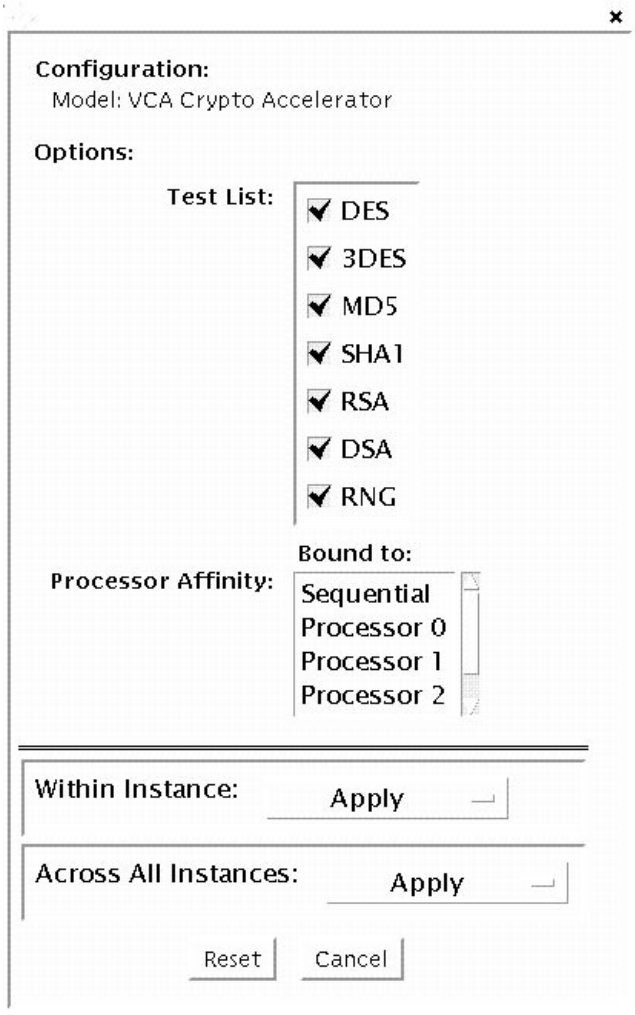

FIGURE 4-2 vcatest Test Parameter Options Dialog Box

TABLE 4-4 `vcatest` Options

Option	Description
DES	Tests DES bulk encryption
3DES	Tests 3DES bulk encryption
MD5	Data Security MD5 message-digest algorithm.

TABLE 4-4 `vcatest` Options *(Continued)*

Option	Description
SHA1	The Secure Hash Algorithm.
RSA	Tests RSA public and private keys
DSA	Tests DSA signature verification
RNG	Tests random number generation

`cryptotest` Test Modes

TABLE 4-5 `cryptotest` Supported Test Modes

Test Mode	Description
Functional	Runs the full set of tests.

`cryptotest` Command Line Syntax for `dcatest`

`/opt/SUNWvts/bin/sparcv9/cryptotest -f -o dev=vca2|dca2, tl=`*RSA+DSA*

TABLE 4-6 `cryptotest` Command Line Syntax for `dcatest`

Option	Description
`dev=dca`*N*	Specifies the instance of the device to test such as `dca0` or `dca2`. Defaults to `dca0` if not included. Note that *N* specifies the placement of the instance number of the device being tested.
`tl=`*testlist*	Specifies the list of subtests to be performed. The subtests for `tl` are separated by the + (plus) character. The supported subtests are DES, 3DES, DSA, RSA, and RNG, so `tl=DES+3DES+DSA+RSA+MD5+SHA1+RNG` enables all subtests. You can also insert `tl=all` which performs all tests. Defaults to `all` if no subtests are specified.

`cryptotest` Command Line Syntax for `vcatest`

`/opt/SUNWvts/bin/sparcv9/cryptotest -f -o dev=vca2, tl=`*RSA+DSA*

TABLE 4-7 `cryptotest` Command Line Syntax for `vcatest`

Option	Description
`dev=vca`*N*	Specifies the instance of the device to test such as `vca0` or `vca2`. Defaults to `vca0` if not included. Note that *N* specifies the placement of the instance number of the device being tested.
`tl=`*testlist*	Specifies the list of subtests to be performed. The subtests for `tl` are separated by the + (plus) character. The supported subtests are DES, 3DES, DSA, RSA, and RNG, so `tl=DES+3DES+DSA+RSA+MD5+SHA1+RNG` enables all subtests. You can also insert `tl=all` which performs all tests. Defaults to `all` if no subtests are specified.

Note – 64-bit tests are located in the `/bin/64` directory, or the relative path in 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-Bit and 64-Bit Tests" on page 3.

CHAPTER 5

Disk and Floppy Drives Test (disktest)

disktest verifies the functionality of hard drives and diskette drives using three subtests (see TABLE 5-1): Media, File System, and Asynchronous I/O.

Note – disktest does support x86 platforms on Solaris.

Most disk drives, such as SCSI disks, native or SCSI floppy disks, IPI, and so on, are supported. The type of drive being tested is displayed at the top of the Test Parameter Options dialog box.

Note – disktest is supported on x86 platforms that use the Solaris Operating System. For disks on x86 machines, the disk partitions could range from 0 to 15. disktest can be performed on any of these selected partitions.

The disktest Test Parameter Options dialog box shows all the partitions that are available for testing. The file System subtest can only be run if the selected partition is mounted (described below). The WriteRead option of the Media subtest is allowed only if a selected partition is *not* mounted.

disktest Test Requirements

By default, disktest does not mount any partitions. To have SunVTS pre-mount all mountable partitions, set the environment variable BYPASS_FS_PROBE to 0 (zero) before starting SunVTS. Pre-mounting can be disabled by unsetting BYPASS_FS_PROBE or changing it to a value other than 0 (zero).

The mount point used by `disktest` is the word *disktest* appended by the name of the disk partition. For example, if the disk partition name is `/dev/dsk/c0t3d0s0`, `disktest` mounts it as superuser under the name `/disktest_c0t3d0s0`.

Caution – If a power failure occurs OR if the disktest is terminated abruptly while the Media subtest is running in WriteRead mode, disk data may be corrupted.

Caution – Running the Media subtest on a disk partition in the WriteRead mode may cause data corruption if the same partition is being used by other applications. Please run SunVTS in the offline mode only when there are no other applications running.

`disktest` tests the floppy drive regardless of whether the Volume Management software is running or not. The following mount point names are used:

- If the Volume Management software *is* running, `disktest` tests the disk drive with the mount point name in the `/etc/mnttab` file.
- If the Volume Management software *is not* running, `disktest` tests the disk drive with the device name `dev=/dev/diskette`. Do not edit the `/etc/vold.conf` file to change the diskette drives. Currently, the SunVTS software is hard-coded to use these path names as the default logic names.

Loading an option file (refer to the *SunVTS User's Guide* for option file details) that was created when `BYPASS_FS_PROBE` was set to `0` (zero) might not work if the `BYPASS_FS_PROBE` environment variable is no longer set to `0`. Testing may fail with the following error:

```
SUNWvts.disktest.8088 07/24/98 15:47:22 disktest c0t0d0 FATAL:
"Couldn't get file system information on /disktest_s0t0d0s0,
statvfs() system call failure error: No such file or directory.
```

This error is caused when SunVTS expects to use the predefined mount point names that are created when `BYPASS_FS_PROBE` is set to `0` (zero), but these mount points do not exist while `BYPASS_FS_PROBE` is not set to `0`.

To use option files with `disktest`, create two separate option files for the two different states of the `BYPASS_FS_PROBE` environment variable.

When a large number of `disktest` instances are run in write/read mode, tests might fail with messages similar to the following.

```
03/22/03 03:33:40 ctech140 SunVTS5.1ps2: VTSID 8011 disktest.FATAL
c1t0d0: "Failed lock mtab semaphore. "semop" system call failure,
errmsg: Invalid argument." Probable_Cause(s): <disktest instances
exceeds system semaphore operation limitation (default system
limit for seminfo_semmnu = 30)><System software error>
Recommended_Action(s): <Add the line "set semsys:seminfo_semmnu=
0x100" to your /etc/system file and reboot the machine> <If the
problem persists, call your authorized Sun service provider.
```

To avoid this issue, add the following entry to the `/etc/system` file and reboot the system.

```
set semsys:seminfo_semmnu=0x100
```

disktest Subtests

The following table describes the disktest subtests:

TABLE 5-1 disktest Subtests

Subtest	Description
Media subtest	The Media subtest verifies the disk media by allowing users to run disktest in different modes such as ReadOnly, ReadCompare, and WriteRead. The Media subtest treats the disk partition as one large chunk of contiguous data. In the WriteRead mode, all instances of disktest communicate through a shared memory service to ensure that they do not overlay the same disk area at the same time. This avoids data corruption. Each of the above three modes could run two different methods of disk testings. These are Synchronous I/O and Asynchronous I/O. SyncIO: Test reads and writes data using Read/Write system calls in a sequential fashion until the specified percentage of media is covered. AsyncIO: Test reads and writes data using aio library calls such as aioread(), aiowrite() until the specified percentage of media is covered. aiowait() is used to synchronize aio operations.
File System subtest	The File system subtest is used to verify the disk file system integrity. It exercises mounted disk partitions carrying the file system. By default, the test only runs on system-mounted partitions, it does not pre-mount any additional partitions. If you want SunVTS to pre-mount all of the unmounted partitions which have a file system, you have to set the environment variable BYPASS_FS_PROBE to '0' (zero). The test creates two temporary files of the size specified by File System File Size, writes the data patterns and compares the two files against each other.

disktest Test Options

To reach the dialog box below, 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

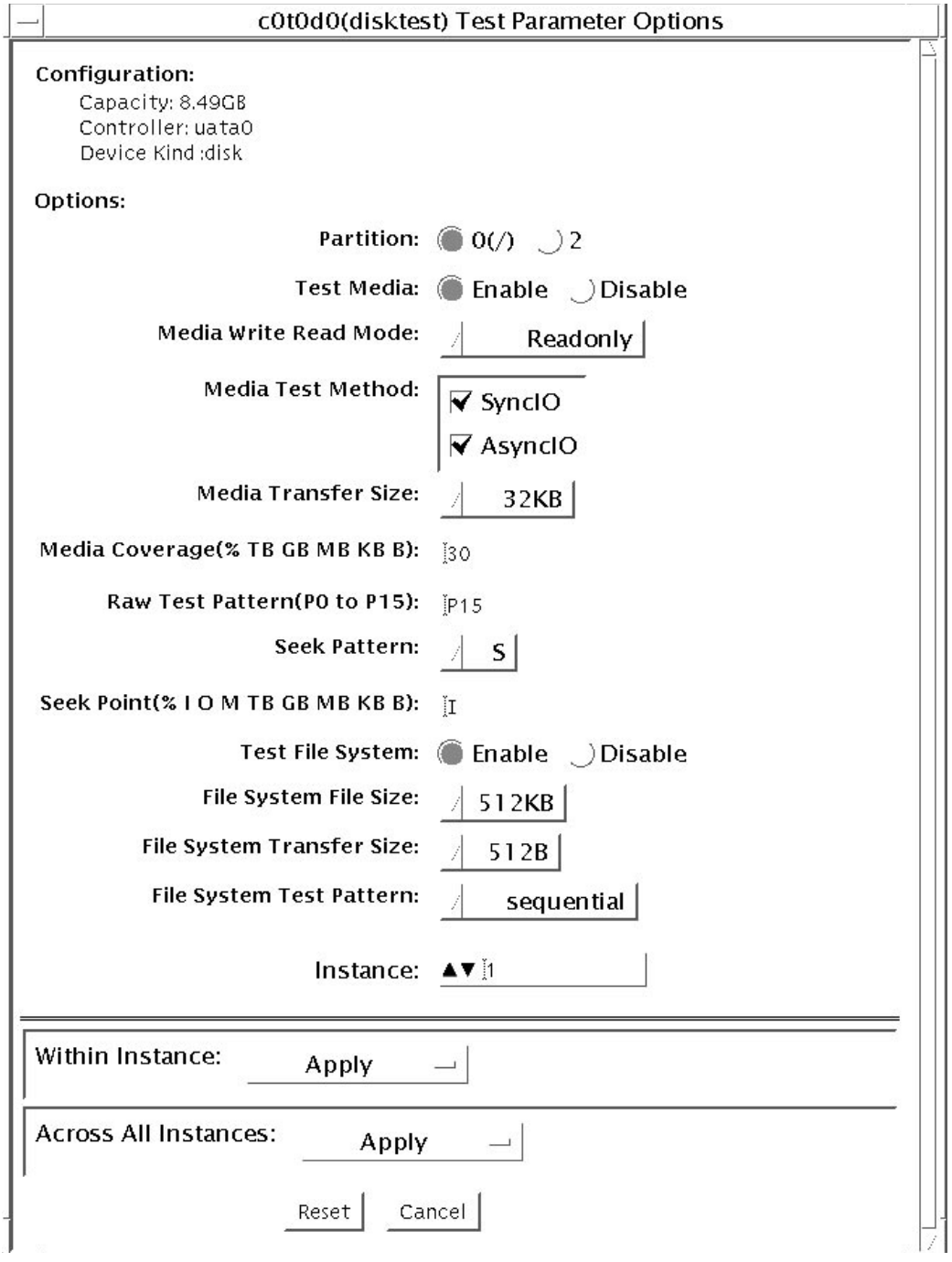

FIGURE 5-1 `disktest` Test Parameter Options Dialog Box

The following table describes the `disktest` option menu for different test modes.

TABLE 5-2 `disktest` Configurations and Options

`disktest` **Options**	**Description**
Partition	Displays the partition for the Media subtest. If a partition is mounted, its mount point is appended after the partition number, such as `1(/usr)`, where `1` is the partition number, and `"(/usr)"` is the mount point.
Test Media	Enable or Disable the media subtest.
Media Write Read Mode	Selects Read-Only or Compare after Read or Read after Write.
Media Test Method	Selects the Media Test Methods (SyncIO and AsyncIO).
Media Coverage (% TB, GB, MB, KB, B)	Enables users to test all or part of a partition (in percentage or in any of TB, GB, MB, KB, B units)
Raw Test Pattern (P0 to P16)	Enables user to specify the write, read pattern. • P0 – Low Frequency Pattern • P1 – Low Transition Density Pattern • P2 – High Transition Density Pattern • P3 – Compliant Jitter Pattern • P4 – Compliant Jitter: RPAT • P5 – Compliant Jitter: CRPAT • P6 – Compliant Jitter: JTPAT • P7 – Compliant Jitter: CJTPAT • P8 – Compliant Jitter: SPAT • P9 – Compliant Jitter: CSPAT • P10 – 8 Bit Cable Pattern • P11 – 16 Bit Cable Pattern • P12 – 8 Bit Xtalk Pattern • P13 – 16 Bit Xtalk Pattern • P14 – MFM Pattern • P15 – Generic Test Patterns • P16 – SATA Test Patterns
Seek Pattern	Enables specifying the pattern of the disk head movement. • S – Sequential • SR – Sequential Reverse • LS – Low Power Sequential • R – Random • LB – Low Power Butterfly • LR – Low Power Reverse Butterfly • AB – Actuator Butterfly • AR – Actuator Reverse Butterfly

TABLE 5-2 `disktest` Configurations and Options *(Continued)*

`disktest` **Options**	**Description**
Seek Point (%, I, O, M, TB, GB, MB, KB, B)	Enables specifying the seek point offset for the I/O. You can specify the offset in percentage or any of TB, GB, MB, KB, B or and I, M, O; that is, Initial, Middle), Outer.
Media Transfer Size	Displays the transfer size of the media subtest.
Test File System	Selects the File System subtest.
File System File Size	Specifies the size for each of the two temporary files for File System testing.
File System Transfer Size	Displays the transfer size of the File System subtest.
File System Test Pattern	Test pattern of File System subtest.
Connection Test for Hard Disk	• Option Menu for hard disk partition—0 - 7 [default] • Test Media—[Enable] (fixed to Enable) • Media Write Read Mode—[Read Only] (fixed to Read Only) • Media Test Method-[SyncIO] (fixed to SyncIO) • Media Coverage(%)—1 • Media Transfer Size—[2 KB] • Test File System—[Disable] (fixed to Disable)

TABLE 5-2 `disktest` Configurations and Options *(Continued)*

`disktest` **Options**	**Description**
Online Mode for Hard Disk	• Partition—0 - 7 [default] • Test Media—[Enable] [Disable] • Test Mode—[Read-only~] (fixed to Read-only) • Media Coverage (% TB GB MB KB B)—[10~] (fixed to 10%) • Media Transfer Size—[2KB~] (fixed to 2 KB) • Test File System—[Disable~] (fixed to Disable) • Media Test Method—[SyncIO] [AsyncIO] • Raw Test Pattern—[P15~] (fixed to P15) • Seek Pattern—[S~] (fixed to S) • Seek Point (% TB GB MB KB B)—[I~] (fixed to I)
Functional Test for Hard Disk	• Partition—0 - 7 [default] • Test Media—[Enable] [Disable] • Media Write Read Mode—[Readonly] [CompareRead] [WriteRead] • Media Test method—[SyncIO] [AsyncIO] • Media Coverage (% TB, GB, MB, KB, B) • Raw Test Pattern (P0 to P16) • Media Transfer Size—[2KB] [16KB] [32KB] [64KB] [128KB] [256KB] [512KB] • Test File System—[Enable] [Disable] • File System File Size—[512KB] [2MB] [8MB] [20MB] [100MB] [200MB] • File System Transfer Size—[512B] [1024B] [10KB] [40KB] [80KB] • File System Test Pattern—[sequential] [0x00000000] [0xffffffff] [0x5aa55aa5] [0xdb6db6db] [random] • Seek Pattern —[S~] (fixed to S) • Seek Point (% TB GB MB KB B)—[I~] (fixed to I)
Functional Test for Floppy Disk	• (under Other-Devices group)—partition: 0 - 7 [default] • Test Media—[Enable] [Disable] • Media Write Read Mode—[Read-only] [BackupWriteRead] • Media Test Method—[SyncIO] [AsyncIO] • Media Coverage (% TB, GB, MB, KB, B) • Raw Test Pattern (P0 to P16) • Media Transfer Size—[2KB] [10KB] [20KB] • Test File System—[Enable] [Disable] • Floppy File Size— [100KB] [200KB] • Floppy Transfer Size—[512B] [1024B] [10KB] • File System Test Pattern—[sequential] [0x00000000] [0xffffffff] [0x5aa55aa5] [0xdb6db6db] [random]

disktest Test Modes

TABLE 5-3 disktest Supported Test Modes

Test Mode	Description
Connection	Only one instance of disktest (which monitors UNIX error messages) is allowed for each disk device. disktest displays messages and reports errors. The test also opens the hard disk, checks the disk configuration, reads a few blocks, and then closes the hard disk. No File System subtest is run. No Write option is available in Connection test mode.
Functional	More than one instance of disktest is allowed for one disk device. The File System subtest, Media subtests, and floppy test can be run in Functional test mode. In Functional mode, disktest performs an additional subtest (Write/Read device buffer subtest) for enclosures.
Online	SunVTS disktest runs the Read Only rawtest with fixed transfer size and fixed rawtest pattern. Both SyncIO and AsyncIO test methods are available. The File system subtest is disabled in the Online test mode. Only one disktest instance could be run in the Online test mode.

disktest Command-Line Syntax

/opt/SUNWvts/bin/disktest *standard_arguments* **-o partition=**<*0-7*> [*"<(mount_point)>"*]**, rawsub=***E(nable)*|*D(isable)***, rawrw=** *Readonly*|*CompareRead*|*WriteRead***, rawiosize=**<*number*>{*...*|*KB*|*kb...*}|*random***, rawcover=**<*number*>|<*number*>{*TB*|*GB*|*MB*|*KB*|*B*|*tb*|*gb*|*mb*|*kb*|*b*} **rawpattern=***P(<0-16>)*|*0x<8 digit data pattern***>, seekpattern=** {*S*|*SR*|*LS*|*R*|*LB*|*LR*|*AB*|*AR*}**, seekpoint=**{*i*|*m*|*o*|<*number*>}**, method=**

AsyncIO+SyncIO, **fssub=***E(nable)*|*D(isable)*, **fssize=**<*number*>{*K*|*KB*|*M*|*MB*|*k*|*kb*|*m*|*mb*}, **fsiosize=**<*number*>{*K*|*KB*|*B*|*k*|*kb*|*b*}, **fspattern=**<*data_pattern*>, **dev=**<*device_name*>

TABLE 5-4 `disktest` Command-Line Syntax

Argument	Description
partition=<*0-7*> [“<*(mount_point)*>”]	Specifies the partition number as follows: • *n*—is the partition number (slice number), usually *0-7* for SPARC and *0-16* for x86 • *mount_point*—is the mount point for the mounted partition that you plan to test For example: **partition=6”(/export)”**
rawsub= *E(nable)*|*D(isable)*	Enables or disables the Media subtest. For example: **rawsub=** *Enable*
rawrw= *Readonly*|*CompareRead*|*WriteRead*	Specifies the Media subtest Read, Compare, and Write mode: • Read only • Read twice, Compare (works only with SyncIO method) • Write, Read, Compare, Restore For example: **rawrw=***ReadOnly*
rawiosize= <*number*>{...|*KB*|*kb*...}|*random*	Specifies the media size to transfer. The block size can be specified in kilobytes. For example: 2K,...512K. For example: **rawiosize=**9
rawcover= <*number*>|<*number*>{*TB*|*GB*|*MB*|*KB*|*B*|*tb*|*gb*|*mb*|*kb*|*b*}	Specifies media coverage from 0-100 (percentage) of the partition. Media Coverage can also be specified in units: TB, GB, MB, KB and B. For example: **rawcover=***40* OR **rawcover=***4GB*

TABLE 5-4 `disktest` Command-Line Syntax *(Continued)*

Argument	Description
`rawpattern=`*P(<0-16>)* | *0x<8 digit data pattern>*	`rawpattern` could be specified as a pre-defined pattern set, *P(0-16)*, or an 8 digit pattern could be specified as: *0xaa55aa55+0xff00ff00+0x*. The following is a description of the supported pre-defined patterns: • P0 – Low Frequency Pattern • P1 – Low Transition Density Pattern • P2 – High Transition Density Pattern • P3 – Compliant Jitter Pattern • P4 – Compliant Jitter: RPAT • P5 – Compliant Jitter: CRPAT • P6 – Compliant Jitter: JTPAT • P7 – Compliant Jitter: CJTPAT • P8 – Compliant Jitter: SPAT • P9 – Compliant Jitter: CSPAT • P10 – 8 Bit Cable Pattern • P11 – 16 Bit Cable Pattern • P12 – 8 Bit Xtalk Pattern • P13 – 16 Bit Xtalk Pattern • P14 – MFM Pattern • P15 – Generic Test Patterns • P16 - SATA Test Patterns For example: **`rawpattern=`***P1*
`seekpattern=` *{S | SR | LS | R | LB | LR | AB | AR}*	`seekpattern` could be specified to select the type of seek test to run on the disk drive. `disktest` supports the following pattern types: • `S` – Sequential • `SR` – Sequential Reverse • `LS` – Low Power Sequential • `R` – Random • `LB` – Low Power Butterfly • `LR` – Low Power Reverse Butterfly • `AB` – Actuator Butterfly • `AR` – Actuator Reverse Butterfly For exmaple: `seekpattern=S`

TABLE 5-4 `disktest` Command-Line Syntax *(Continued)*

Argument	Description
seekpoint=*{i | m | o | <number>}*	Specify the seek-point for the I/O. This could be specified either in terms of the range - inner, middle and outer. Or in terms of absolute seek location. The absolute location is specied by a number followed by any of the following units {TB | GB | MB | KB | B | tb | gb | mb | kb | b}. For example: a) `seekpoint=`*I*, start the I/O from block 1. b) `seekpoint=`*M*, start the I/O from middle offset of the partition.
method=*AsyncIO+SyncIO*	Specifies the Media access method. You can choose to use either or both methods . If you use both access methods together, you must insert a '+' between the two: AsyncIO: Runs the asynchronous I/O test, using the async read/write feature of the Solaris disk driver SyncIO: Runs the synchronous I/O test. For example: **method=***AsyncIO*
fssub=*E(nable) | D(isable)*	Enables or disables the File System subtest. File system subtest runs on a mounted partition with a file system.
fspattern=<*data_pattern*>	Specifies the file system data pattern as sequential or random or one of the patterns selected from the list. *{seq(uential) | 0x0(0000000) | 0xf(ffffffff) | 0xa (5a5a5a5) | 0x5(a5a5a5a) | ran(dom) | 0xd(b6db6db)}* For example: a) **fspattern=***0xa* a) **fspattern=***seq*
fssize= <*number*>*{K | KB | M | MB | k | kb | m | mb}*	Indicates the file system subtest size in Megabytes or Kilobytes: • `K | k | KB | kb` – kilobytes • `M | m | MB | mb` – megabytes `512KB | 2MB | 8MB | 20MB | 100MB | 200MB`
fsiosize= <*number*>*{K | KB | B | k | kb | b}*	Indicates the size of the file system subtest I/O transfer in bytes or Kilobytes: • B | b – bytes • `K | k | KB | kb` – Kilobytes `512B | 1024B | 10KB | 40KB | 80KB`
dev=*device_name*	Specifies the name of the disk to be tested. For example: c0t3d0.

The following example shows how to run `disktest` on a partition "0" (which is mounted under "/") for the disk device `c0t0d0`. The media subtest is enabled in ReadOnly mode using SyncIO method. The coverage specified is 30% with 512 KB transfer size. The File System subtest is disabled.

```
# /opt/SUNWvts/bin/disktest -f -o partition=0"(/)", rawsub=Enable,
rawrw=ReadOnly, method=SyncIO, rawcover=30, rawiosize=512KB,
fssub=Disable, dev=c0t0d0
```

Note – 64-bit tests are located in the `/bin/64` directory, or the relative path in 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-Bit and 64-Bit Tests" on page 3.

CHAPTER 6

Floating Point Unit Test (fputest)

The fputest checks the functionality of the floating point unit in a Sun SPARC based CPU. The test verifies the functionality by various arithmetic operations. In addition, the fputest stresses the CPU with the use of benchmarks. Both single and double precision numbers are used for the operations.

Note – fputest does support x86 platforms on Solaris.

Note – Three benchmarks of fputest—cparanoia, kcdiv, and kcsqrt—have been ported over for x86 systems. The x86 version of fputest only uses the Double Precision operations for rounding, chopping, sticky bit, and so on.

When fputest is chosen in Exclusive test mode from the SunVTS graphical user interface, it may run multiple instances in parallel on different CPUs. The number of such instances that may be running in parallel at the same time is dynamically determined depending on system resources.

fputest Subtests

Instruction tests:

- FSR Register test
- Registers test
- NACK test
- Move Registers test
- Positive to Negative test
- Negative to Positive test
- Absolute test
- Single-Precision Integer to Floating Point test

- Double-Precision Integer to Floating Point test
- Single-Precision Floating Point to Integer test
- Double-Precision Floating Point to Integer test
- Single-Precision Round Toward Zero test
- Double-Precision Round Toward Zero test
- Single to Double-Precision Format Conversion test
- Double to Single-Precision Format Conversion test
- Single and Double-Precision Addition, Subtraction, Multiplication, Square-root, Division, and Compare tests
- Single and Double-Precision Compare and Exception if Unordered tests
- Branching and No Branching on Condition Instructions tests
- Single and Double-Precision Chaining tests
- Weitek Status tests
- Lock test
- Single and Double-Precision Datapath tests
- Timing (load) test

Benchmark tests:

- Lapack test
- Cparanoia test
- Kcsqrt test
- Kcdiv test
- Clorenz test
- Cvector test

`fputest` Options

To reach the dialog box below, 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

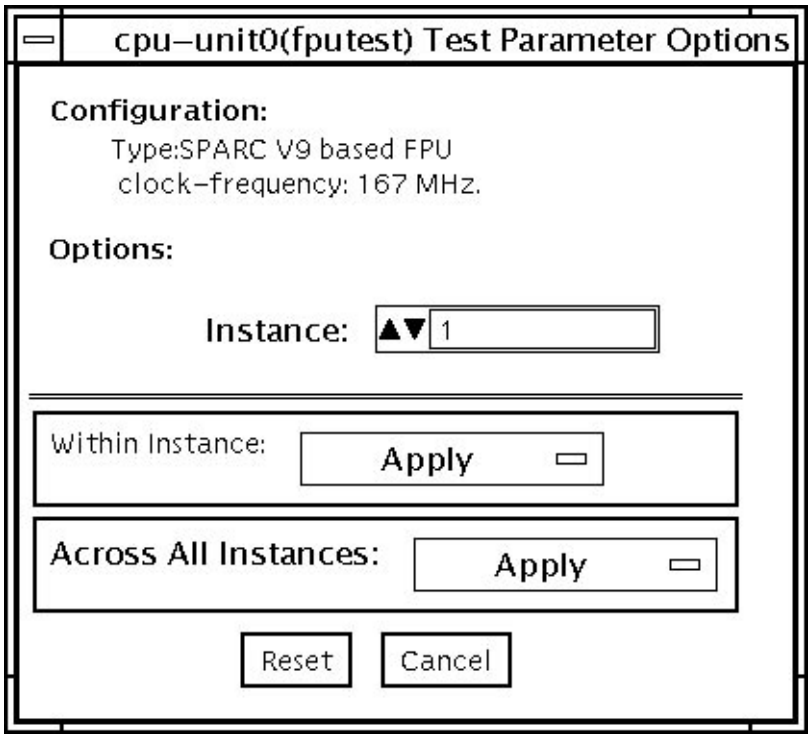

FIGURE 6-1 `fputest` Test Parameter Options Dialog Box

Note – It is not advisable to use the Processor Affinity option for this test. Doing so reduces the effectiveness of the test.

---

fputest Test Modes

TABLE 6-1 fputest Supported Test Modes

Test Mode	Description
Connection	Includes all the instruction tests.
Functional	Performs all the instruction tests and all the benchmark tests.
Exclusive	Performs several fpu benchmark tests.
Online	Supported

fputest Command-Line Syntax

/opt/SUNWvts/bin/fputest [**-scruvdtlxnf**] [**-p n**] [**-i n**] [**-w n**] [**-o dev=cpu-unit***N*]

Note – Options -s, -p, -i, and -w are not applicable when tests are invoked from the command line.

TABLE 6-2 fputest Command-Line Syntax

Argument	Description
-s	Enables SunVTS mode
-c	Enables core file
-r	Enables run on error
-u	Enables list usage
-v	Enables verbose mode
-d	Enables debug mode
-t	Enables test function trace mode
-l	Enables online mode, and forks tests automatically against all online CPU IDs in the system.
-x	Enables exlusive mode, and forks tests automatically against all online CPU IDs in the system.

TABLE 6-2 `fputest` Command-Line Syntax

Argument	Description
`-n`	Enables connectivity mode
`-f`	Enables offline mode, and forks tests automatically against all online CPU IDs in the system.
-p n	*n* specifies the number of passes. The default is 1.
`-i` *n*	*n* specifies the number of total instances for the test. The default is 1.
`-w` *n*	*n* specifies which instance this test is assignes. The default is 0.
`-o`	Enables test specific command arguments
`dev=cpu-unit`*N*	Specifies the CPU unit to be tested. *N* specifies the numeric ID of online CPU. This is not a required option.

Usage Examples:

Note – When using `fputest` specific arguments the `-o` AND `dev` options are required.

To execute exclusive `fputest` with verbose messages against CPU 72, use the following syntax:

```
# /opt/SUNWvts/bin/fputest -xvo dev=cpu-unit72
```

To execute functional `fputest` with verbose messages against CPU 3, use the following syntax:

```
# /opt/SUNWvts/bin/fputest -fvo dev=cpu-unit3
```

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-Bit and 64-Bit Tests" on page 4.

CHAPTER 7

Sun XVR-2200 and XVR-2400 Graphics Accelerator Test (`graphicstest`)

`graphicstest` verifies the functionality of XVR-2200 and XVR-2400 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 framebuffer's accelerator port while running graphicstest. This combination causes SunVTS to return incorrect errors.

Test Requirements

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
```

For full instructions on testing frame buffers, see "Testing Frame Buffers" on page 10.

Test Preparation

Perform the steps in this section prior to performing `graphicstest` to ensure the test runs as smoothly as possible.

1. You must run `graphicstest` through the SunVTS user interface. The window system (such as CDE or GNOME) must be running on the XVR-2200 or XVR-2400, and the window system must be setup to run in 24-bit mode (the default depth for XVR-2200 and XVR-2400).
2. Turn Power Management off, if it is enabled. See "Test Requirements" on page 61.
3. Verify that no other program is running that might modify the screen during the test.
4. Verify you have permission to lock the X server. `graphicstest` is designed to lock the X server during testing to prevent screen changes.
5. Verify that the CDE login window is not displayed during testing.
6. Verify that the window system is running on one frame buffer only.

`graphicstest` Options

To reach the dialog box below, 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 framebuffer being tested. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the SunVTS User's Guide.

By default, all `graphicstest` options are enabled.

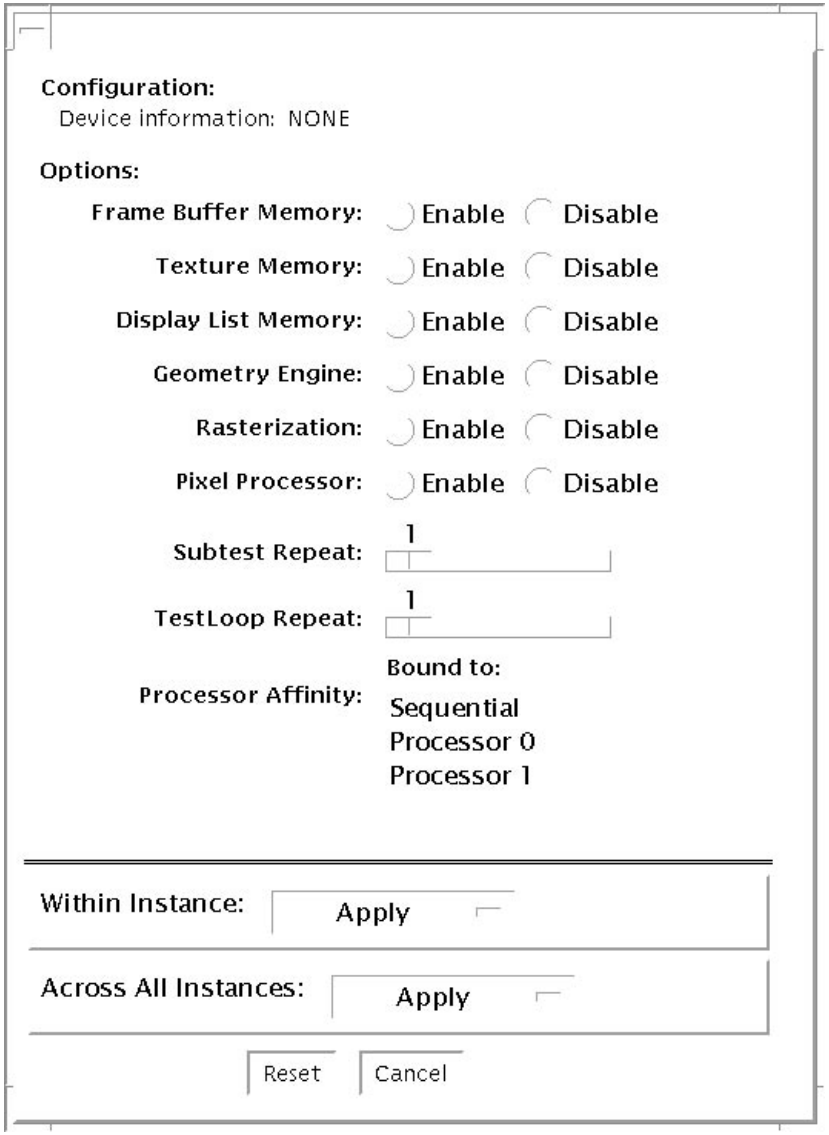

FIGURE 7-1 `graphicstest jfb0` Test Parameter Options Dialog Box

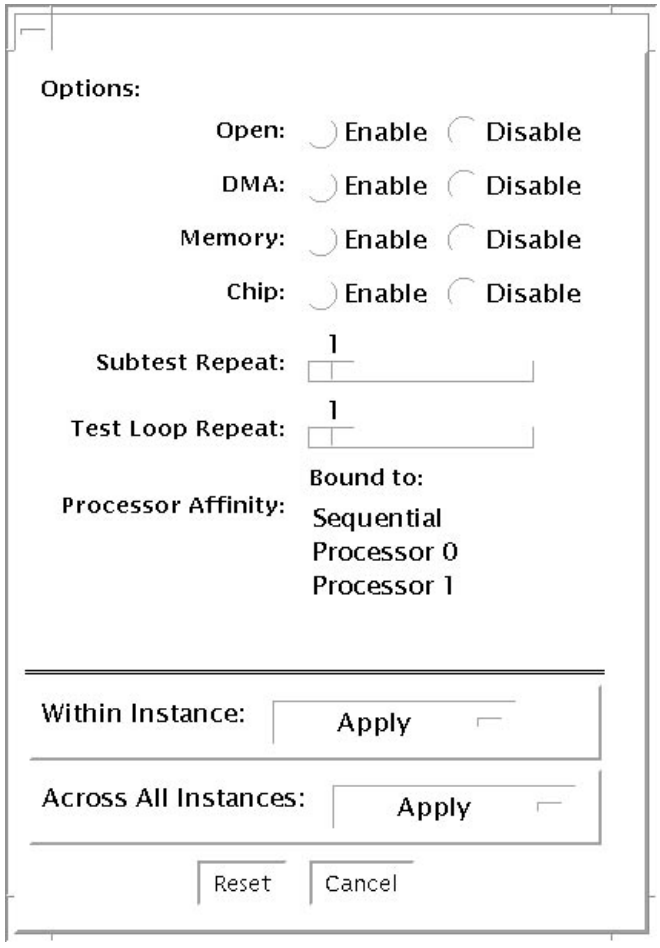

FIGURE 7-2 `graphicstest kfb0` Test Parameter Options Dialog Box

TABLE 7-1 `graphicstest` Options

Options	Description
Open test	The 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 framebuffer 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 framebuffer are written to and read back from to see if the the framebuffer was correctly mapped. This test takes little time and no progress is displayed.
Memory test	The memory tests 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 it 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	The 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 framebuffer and reads it back using the DMA engine and checks to see if the correct values were read back.
Chip Test	The 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

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

TABLE 7-2 `graphicstest` Supported Test Modes

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

graphicstest Command-Line Syntax

/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 7-3 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. Default is enable.
dma=*Enable/Disable*	Enables or disables the dma test. Default is enable.
mem=*Enable/Disable*	Enables or disables the memory test. Default is enable.
chip=*Enable/Disable*	Enables or disables the chip test. Default is enable.
B=*n*	Defines times to repeat each test loop Default is one.
F=*n*	Defines the number of times to repeat each subtest. 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

Note – 64-bit tests are located in the /bin/64 directory, or the relative path in 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-Bit and 64-Bit Tests" on page 3.

CHAPTER 8

Infiniband Host Channel Adapter Test (`ibhcatest`)

`ibhcatest` is comprised of multiple iRISC CPU cores, two 4x IB ports and integrated SerDes components. In addition, the `ibhcatest` external associated components include FLASH ROM and DDR memory. `ibhcatest` provides high speed interconnect through PCI interface to external IB fabric. Supported platforms include: two 1U and two 2U x86 AMD Opteron entry-level servers, Sun Fire V2xx, V4xx and E series high end servers.

Note – `ibhcatest` supports the same set of options for both SPARC and x86 platforms.

`ibhcatest` provides a mechanism to exercise and verify the proper operation of the Tavor chip and its associated components, such as DDR memory, Flash PROM and internal IB packet transmit/receive circuitry. The goal is to isolate single fault to the identifiable component(s).

`ibhcatest` supports three execution test modes in SunVTS: Connection, Exclusive and Functional. In Connection mode, the test will provide a basic sanity check. This basic sanity test is done by querying for the Tavor firmware / hardware revision and running internal loopback.

The internal loopback test is run at least once depending on the amount of time each pass takes. In Functional mode all subtests are executed according to the options selected. In Exclusive mode all subtests are executed in sequence.

Tavor supports an internal loopback mechanism which is very similar to the actual operation. The main difference is that data does not go through the integrated SerDes and the 4x IB port circuitry. Also on the receiving side, data does not get verified by the CRC algorithm. Otherwise, all other components of Tavor that involve in transmitting and receiving data packets are being exercised.

Tavor based HCA is designed to use a single, 256 MB DDR memory for data storage at run time. This data storage is intended to be used and shared by three interdependent clients: Tavor driver, firmware, and hardware. During driver initialization, predetermined data structures and data are laid out in the memory.

With no exclusive atomic access from the driver side, subsequent writes to any memory location that contain real data can cause undesirable results like a system crash. Furthermore, the data allocation size is fixed, writing to the remaining free memory does not add any value in terms of finding faults.

With these constraints in mind, the memory subtest is limited to read only operations to cover the entire DDR memory. There is no checking for data corruption and no mechanism for triggering single/double bit type of errors through writing to memory. The resulting benefit of read operations is a secondary effect that occurs in the generation of high volume PCI activities from the memory accesses. Thus the test becomes a good exerciser to bring out bus related problems.

ibhcatest Subtests

TABLE 8-1 ibhcatest Subtests

Subtest	Description
Internal Loopback Test	The HCA supports internal loopback for packets transmitted between QPs that are assigned to the same HCA port. When transmitting a packet, if it is destined to a DLID that is equivalent to the Port LID with the LMC bits masked out or the packet DLID is a multicast LID, the packet goes on the loopback path. In this latter case, the packet also is transmitted to the fabric. When a packet is looped back, it must pass the SL2VL mapping. If the mapping yields 15 or a nonoperational VL, the packet is discarded. In the inbound direction, the ICRC and VCRC checks are "blindly" passed for looped back packets. Note that internal loopback is supported only for packets that are transmitted and received on the same port. Packets that are transmitted on one port and received on another port are transmitted to the fabric. The fabric should direct them to the destination port. This subtest uses interfaces from the Tavor driver to perform loopback testing. It is a push button type of test. Information such as data pattern for data packets, port number, CQ polling, retries between iteration, and the number of iterations for each `ioctl` call are passed to the driver. Once finished, status regarding the number passes completed is returned. If the number of passes does not match the number of iterations, a failure has occurred. This might happen when the number of retries is exhausted, and the last failing buffer in the retry series is returned as result. SunVTS then determines exactly what failed in the buffer and reports the failure. The options for this subtest are as follows: • `lb`=*Enabled* | *Disabled*: Turn on/off the loopback test • `tlbport`=*1+2*: Loopback test on Port 1 and/or 2, default is 1+2 • `data`=*Pattern*: Specific data pattern (default patterns 0xa5a5a5a5) • `cq`=*Time*: Number of CQ polling time value (in microseconds per iteration); default: 55000; max: 1000000. • `loop`=*Number*: Number of loopback iterations for each pass; default: 200; max: 1000 • `warn`=*Enabled* | *Disabled*: When enabled, prints a warning message
DDR READ Test	This subtetst is comprised of two test modes, Sequential and Random . The start and end address offsets are determined dynamically by obtaining them from the firmware. In Sequential mode, length and starting offset are instructed by the test option `rdoffset` and `rdsz`. Then the test goes through and sequentially reads data from each memory address until all memory locations are covered or the end address is reached. Each read is accomplished by an `ioctl` call to the driver. The test returns pass or fail based on the completion status of each `ioctl` call. In the `random test` method, this subtest reads the number of `rdsz` times in a randomly generated address bound by the start and end address offset. The options for this subtest are as follows: • `ddr`=*Enabled* | *Disabled*: Turn off/off the DDR Memory test • `rdoffset`=*Offset*: Starting offset of DDR Memory read, default is 0x0, in hexadecimal • `rdsz`=*Size*: Read number of byte of DDR Memory from Offset to Max address location, default: 0x2000; max: 256 MB, in hexadecimal

Caution – In SunVTS environment `ibhcatest` and `nettest` are mutually exclusive, `nettest` has higher priority if the IB port interface is plumbed up when SunVTS is invoked. These two tests can not be run at the same time at the command line, if both of these tests are invoked at the command line, `ibhcatest` exits gracefully if the IB port interface is plumbed up. The commands to bring down the IB daemon (`ibd` [`IPoIB`]) are as follows:

```
# ifconfig ibdXX down
# ifconfig ibdXX unplumb
```

Where *XX* is instance number of the interface.

`ibhcatest` Options

To reach the dialog box below, 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 framebuffer

being tested. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the SunVTS User's Guide.

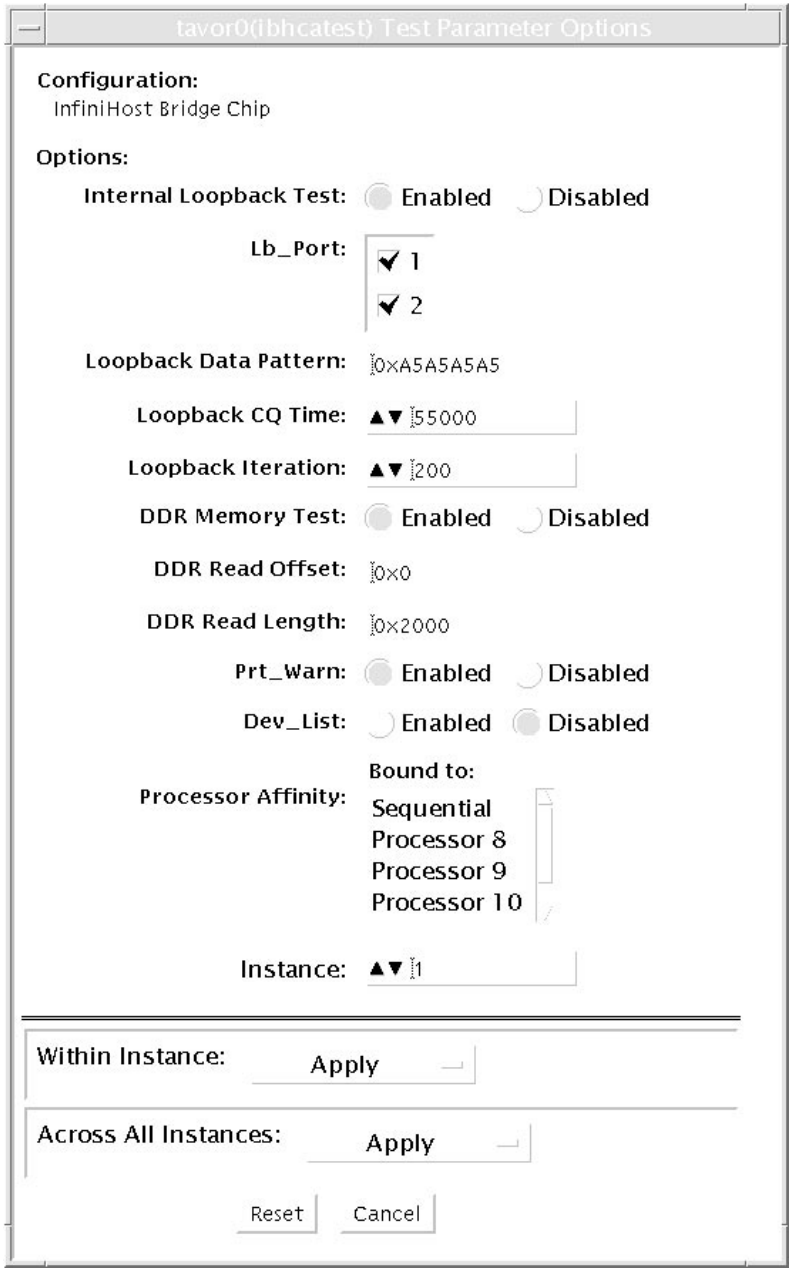

FIGURE 8-1 `ibhcatest` Test Parameter Options Dialog Box

TABLE 8-2 `ibhcatest` Options

Option	Description
`lb=`*Enabled* | *Disabled*	Turn on/off the loopback test
`tlbport=`*1*+*2*	Loopback test on Port 1 and/or 2, default is 1+2
`data=`*Pattern*	Specific data pattern (default patterns 0xa5a5a5a5)
`cq=`*Time*	Number of CQ polling time value (in microseconds per iteration), default: 55000; max: 1000000
`loop=`*Number*	Number of loopback iterations for each pass; default: 200; max: 1000
`ddr=`*Enabled* | *Disabled*	Turn on/off the DDR Memory test
`rdoffset=`*Offset*	Starting offset of DDR Memory read, default is 0x0, in hexadecimal
`rdsz=`*Size*	Read number of byte of DDR Memory from Offset to Max address location, default: 0x2000; max: 256 MB, in hexadecimal
`warn=`*Enabled* | *Disabled*	Print a warning message when enabled
`list`	Print device list, no testing when set

ibhcatest Test Modes

TABLE 8-3 ibhcatest Supported Test Modes

Test Mode	Description
Connection	Provides a basic sanity check by querying for the Tavor firmware / hardware revision and running internal loopback. The internal loopback test runs at least once depending on the amount of time each pass takes.
Exclusive	Executes all subtests sequentially.
Functional	Executes all subtests according to what is selected.

ibhcatest Command Line Syntax

ibhcatest [-scruvdtlxnf] [-p *n*] [-i *n*] [-w *n*] [-o [dev=*text*] [lb=*Enabled* | *Disabled*] [tlbport=*1+2*] [data=*Pattern*] [cq=*Time*] [loop=*Number*] [ddr=*Enabled* | *Disabled*] [rdoffset=*Offset*] [rdsz=*Len*] [warn=*Enabled* | *Disabled*] [list]]

Example:

```
# ibhcatest -p 0 -svf -o lb=Enabled, tlbport=1+2, data=0xA5A5A5A5,
cq=55000, loop=200, ddr=Enabled, rdoffset=0x0, rdsz=0x2000, warn=
Enabled, dev=tavor1
```

TABLE 8-4 ibhcatest Command Line Syntax

Option	Description
lb=*Enabled* | *Disabled*	Turn on/off the loopback test
tlbport=*1+2*	Loopback test on Port 1 and/or 2, default is 1+2
data=*Pattern*	Specific data pattern (default patterns 0xa5a5a5a5)
cq=*Time*	Number of CQ polling time value (in microseconds per iteration), default: 55000; max: 1000000.
loop=*Number*	Number of loopback iterations for each pass; default: 200; max: 1000
ddr=*Enabled* | *Disabled*	Turn on/off the DDR Memory test

TABLE 8-4 `ibhcatest` Command Line Syntax *(Continued)*

Option	Description
`rdoffset=`*Offset*	Starting offset of DDR Memory read, default is 0x0, in hexadecimal
`rdsz=`*Size*	Read number of byte of DDR Memory from Offset to Max address location, default: 0x2000; max: 256 MB, in hexadecimal
`warn=`*Enabled* | *Disabled*	Print a warning message when enabled
`list`	Print device list, no testing when set

Note – 64-bit tests are located in the `/bin/64` directory, or the relative path in 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-Bit and 64-Bit Tests" on page 3.

CHAPTER 9

Sun™ XVR-1200 Graphics Accelerator Test (`jfbtest`)

`jfbtest` verifies the proper functioning of the Sun™ XVR-1200 graphics accelerator.

`jfbtest` can detect and adapt to many video modes of the Sun XVR-1200 graphics accelerator. All tests can run at a resolution of 1024x768 or higher.

You can interrupt `jfbtest` using Control-C.

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 Sun XVR-1200 graphics accelerator port while running `jfbtest`. This combination causes SunVTS to return incorrect errors.

`jfbtest` Test Requirements

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

```
# xset -dpms
```

The display resolution must be 1024x768 or higher (the standard resolution). To change resolution, go to a UNIX prompt and type:

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

For full instructions on testing frame buffers, see the Testing Frame Buffers section of the *SunVTS 5.1 Test Reference Manual*.

Preparation for `jfbtest`

You should complete a few steps in advance to ensure that `jfbtest` runs as smoothly as possible.

If you are running `jfbtest` in a window system (such as CDE):

- Turn Power Management off, if it is enabled. The following is an alternate way to turn Power Management off. Change `allowFBPM=1` to `allowFBPM=0` in `/platform/sun4u/kernal/drv/jfb.conf` file.
- Make sure that no other program is running that might modify the screen during the test.
- Make sure you have permission to lock the X server. `jfbtest` is designed to lock the X server during testing to prevent screen changes.
- The CDE login window should not be displayed during testing.
- Check that the window system is only running on one Sun XVR-1200 graphics accelerator.

If you are not running `jfbtest` in a window system:

- Turn Power Management off, if it is enabled. The following is an alternate way to turn Power Management off. Change `allowFBPM=1` to `allowFBPM=0` in `/platform/sun4u/kernal/drv/jfb.conf` file.
- Make sure that no other program is running that might modify the screen during the test.
- Make sure the Sun XVR-1200 graphics accelerator being tested is not the console device. Console messages may modify the screen.

`jfbtest` Options

To reach the dialog box below, 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

By default, all `jfbtest` options are enabled.

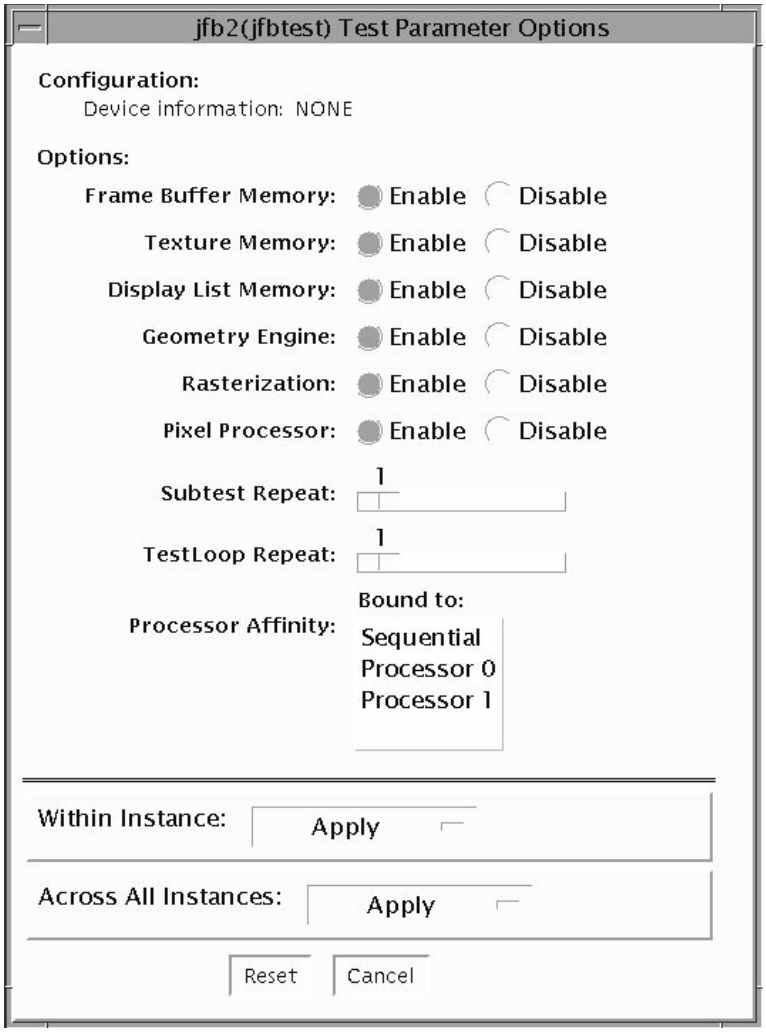

FIGURE 9-1 `jfbtest` Test Parameter Options Dialog Box

TABLE 9-1 `jfbtest` Options

`jfbtest` **Options**	**Description**
Frame Buffer Memory test	Thoroughly tests the Sun XVR-1200 video memory by using read and write requests. Tests for shorts or failed connections on the data bus by writing the following values to every address: • `0xFFFFFFFF` • `0xFFFF0000` • `0x0000FFFF` • `0xFF00FF00` • `0x00FF00FF` • `0xF0F0F0F0` • `0x0F0F0F0F` • `0xCCCCCCCC` • `0x33333333` • `0xAAAAAAAA` • `0x55555555` Tests for shorts or failed connections on the address bus by writing the offset of each memory location to each location and reading them back. This may also catch speed-related problems due to the volume of read/writes. Errors in the test are reported as an error in a particular address, not attributed to a specific chip. To help distinguish bit-related errors, the errors are summarized to list which bits had at least one error in the test. This test shows on the screen as random pixels.
Texture Memory test	This test is identical in process to the Frame Buffer Memory test (above). Since this test produces no visible effect, rectangles are drawn in rows across the screen to show progress.
Display List Memory test	This test is identical in process to the Frame Buffer Memory and Texture Memory tests (above), and is applied to direct burst memory. This test takes little time and no progress is displayed.
Geometry Engine test	Loads diagnostic microcode into the geometry engine and confirms that the processor operates correctly. This is a pass/fail test. This test takes little time and no progress is displayed.

TABLE 9-1 `jfbtest` Options

`jfbtest` **Options**	**Description**
Rasterization test	Renders many primitives with minimal fragment processing, to test the rasterization of the primitives. The primitives used are: • Dots • Anti-aliased dots • Lines using all for line-drawing primitives • Anti-aliased lines using all for line-drawing primitives • Triangles, Quads, and Polygons in point, line, and fill modes • Rectangles This tests for the following rasterization attributes: • Pixel coverage • Constant value registers for color, Z, and stencil • Interpolation of color, Z, and texture coordinates along lines and spans in polygons • Texture map sampling Resulting images are compared against stored images. Errors indicate which operation type and value was being tested, and the coordinate of the failed pixel.

TABLE 9-1 `jfbtest` Options

`jfbtest` **Options**	**Description**
Pixel Processor test	Tries the various pixel processing operators using a variety of fragment values. This tests the following fragment processing operations: • Depth Buffering • Blending • Alpha Test • Color Test • Color Clamp • Logic Operations • Color Matrix and Bias • Color Table • Control Planes • Fast Clear • Stencil • Scissor Clipping • Desktop Clipping • Mask Clipping • Write Masks • Window Origin • Fog • Pixel Texture • Accumulation Buffer • Pixel Buffers Resulting images are compared against stored images. Errors indicate which operation type and value was being tested and the coordinate of the failed pixel.

jfbtest Test Modes

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

TABLE 9-2 jfbtest Supported Test Modes

Test Mode	Description
Functional	Runs the full set of tests.
Connection	Runs the full set of tests.

jfbtest Command-Line Syntax

/opt/SUNWvts/bin/jfbtest *standard_arguments* **-o dev=***device_name***, fbmem=***E(nable)/D(isable)***,texmem=***E/D***,dlmem=***E/D***,geomeng=***E/D***, rasterization=***E/D***,pixelproc=***E/D***,subtest_repeat=***number***, test_repeat=***number*

TABLE 9-3 jfbtest 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.
fbmem=*E/D*	Enables or disables the Frame Buffer Memory test.
texmem=*E/D*	Enables or disables the Texture Memory test.
dlmem=*E/D*	Enables or disables the Display List Memory test.
geomeng=*E/D*	Enables or disables the Geometry Engine test.
rasterization=*E/D*	Enables or disables the Rasterization test.
pixelproc=*E/D*	Enables or disables the Pixel Processing test.
subtest_repeat=*number*	Defines the number of times to repeat each subtest. The default is 1.
test_repeat=*number*	Defines the number of times to repeat a test loop before passing. The default is 1.

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 may only be available as a 32-bit test. For more information refer to the "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

CHAPTER 10

Level 2 Cache Test (`l2sramtest`)

`l2sramtest` exercises the level2 cache in the CPU module of Sun systems. In most CPUs, the level2 cache is also the external cache, but in some cases the level2 cache is on the chip. This test writes, reads, and verifies access of multiple virtual addresses. This test contains multiple subtests that try to exercise the l2cache by causing hits/misses, performing marching patterns on the l2cache cells, and writing patterns that cause electrical stress.

`l2sramtest` is self scaling and adaptive. It scales with the size of the system. It will automatically retrieve the number of CPUs in the system and internally create that many threads of `l2sramtest` to give coverage to the whole system at a given time. For UltraSPARC-H20 and UltraSPARC-T1 systems, Level 2 cache is shared by all cores in the system thus the test retrieves the number of valid cores in the system and creates that many threads to give coverage. This test also dynamically determines the size and organization of the l2cache. The user does not have to input these values.

`l2sramtest` Options

To reach the dialog box below, 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

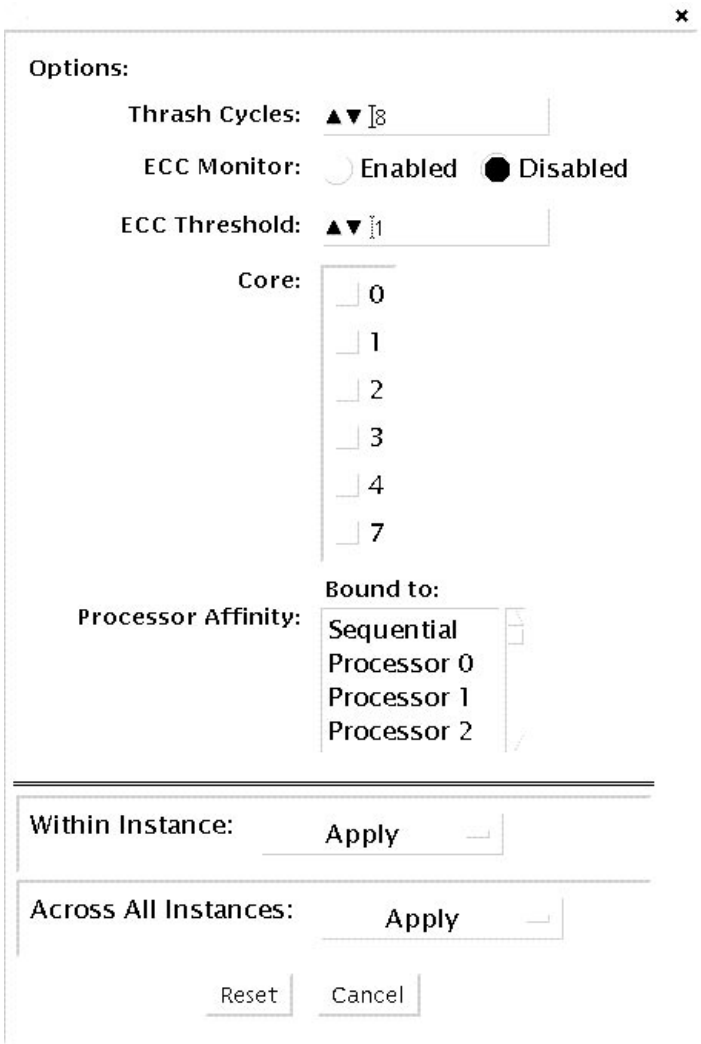

FIGURE 10-1 `l2sramtest` Test Parameter Options Dialog Box

TABLE 10-1 `l2sramtest` Options

Option	Description
Thrash Cycles	Specifies the number of thrashing cycles the test completes for the level2 cache on the system. The default value is 8.
ECC Error Monitor	Specifies whether the error monitoring should be on or off. The error monitor monitors the `/var/adm/messages` file for failure messages which could be caused due to the test. The default value is OFF.
Threshold	Specifies the threshold value of the number of errors after which the test would register an error. This argument is only applicable if the Error Monitor option is ON. The errors that come on the `/var/adm/messages` could be correctable error, that is why the threshold value is provided for the user to give a facility to ignore the errors if they are below the threshold value. The default value is 1.
Core	This option is displayed and supported on UltraSPARC-H20 and UltraSPARC-T1 based systems only. Use the core option to perform isolation testing on selected cores only. By default, isolation testing is not enabled and system-wide testing is performed with all available cores.

Note – The `l2sramtest` automatically handles processor binding. Do not use the Processor Affinity option for the `l2sramtest`.

`l2sramtest` Test Modes

TABLE 10-2 `l2sramtest` Supported Test Modes

Test Mode	Description
Exclusive	Performs only the `l2sramtest` (full test).

`l2sramtest` Command-Line Syntax

`/opt/SUNWvts/bin/sparcv9/l2sramtest` **`-`***standard_arguments* **`-o`** [**`dev=`***l2sram*, **`count=`***[1...1023]*, **`em=`***[Enabled,Disabled]*, **`threshold=`***[0..255]*, **`core`**=[*0+1+2+...*]]

Note – The core option is supported on UltraSPARC-H20 and UltraSPARC-T1 based systems only.

Note – The `l2sramtest` is not a per CPU test. There will be only one `l2sramtest` for the whole system (one image of Solaris). It will run on all the CPUs of the domain.

TABLE 10-3 `l2sramtest` Command-Line Syntax

Argument	Description
dev=*l2sram*	Specifies the device. The default value is `l2sram`.
count=*number*	Specifies the number of thrashing cycles that the test completes for the level2 cache on the system. Default value for Offline mode is 8.
em=*Enabled/Disabled*	Specifies the enabling or disabling of the ECC Error Monitor. The default value is Disabled.
threshold=*number*	Specifies the threshold value of how many correctable ECC errors can occur in the elapsed time before `l2sramtest` reports a test failure. The default value is 1.
core=[*0+1+2+...*]	This option is supported on UltraSPARC-H20 and UltraSPARC-T1 based systems only. Use the core option to perform isolation testing on selected cores only. By default, isolation testing is not enabled and system-wide testing is performed with all available cores.

CHAPTER 11

Level 3 Cache Test (`l3sramtest`)

`l3sramtest` exercises the level3 cache in the CPU module of Sun's Ultra-SPARC-IV+ systems. This is an external cache, with on-chip tags.

`l3sramtest` runs various subtests on the cache that try to exercise the cache by causing hits/misses, performing marching patterns on the level3 cache cells and writing patterns that cause electrical stress. This test also supports Cache Interconnect Stress test using SSO patterns that targets various interconnects between level 1, level 2 and level 3 caches.

`l3sramtest` is self scaling and adaptive. It scales with the size of the system. It will automatically retrieve the number of CPUs in the system and internally create that many threads to give coverage to the whole system at a given time. This test also dynamically determines the size and organization of the l3cache. The user does not have to input these values.

`l3sramtest` 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

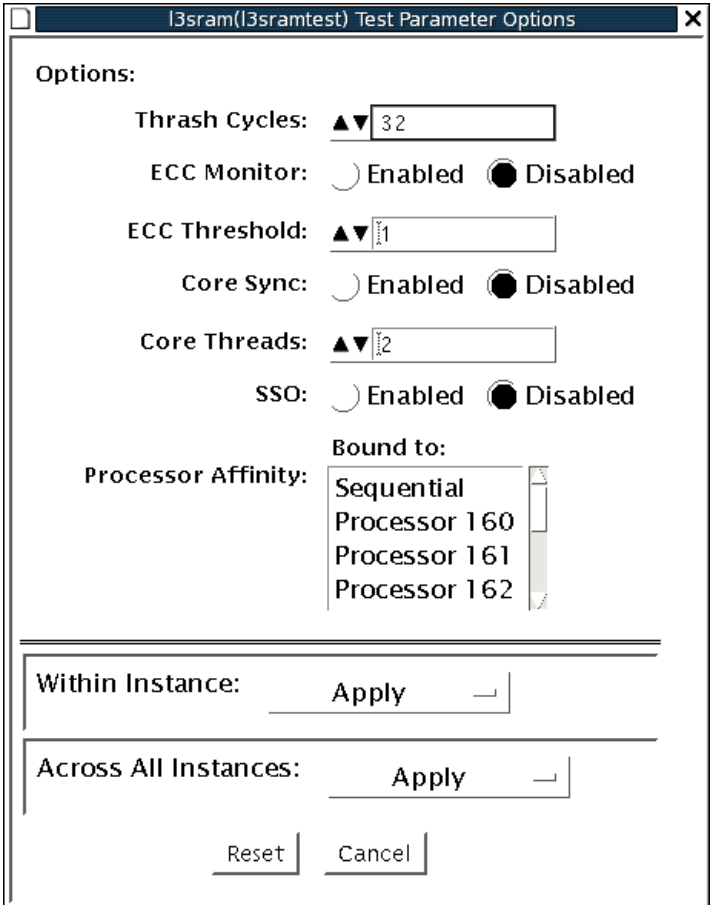

FIGURE 11-1 `l3sramtest` Test Parameter Options Dialog Box

TABLE 11-1 `l3sramtest` Options

Option	Description
Thrash Cycles	Specifies the number of thrashing cycles the test completes for the level3 cache on the system. The default value is 32.
ECC Error Monitor	Enables or disables the ECC Error Monitor. The error monitor monitors the `/var/adm/messages` file for failure messages that could be caused due to the test. The default value is disabled—do not enable this option.
ECC Threshold	Specifies the threshold value of the number of errors after which the test would register an error. This argument is only applicable if the Error Monitor option is enabled. The errors that come on the `/var/adm/messages` could be correctable, that is why the threshold value is provided to give a facility to ignore the errors if they are below the threshold value. The default value is 1.
Core Sync	Enables or disables the core syncronization mode. When this option is enabled, each CPU core runs with exclusive access to the cache during the entire cache test execution. The default value is disabled.
SSO	Enables or disables the Internal Cache Interconnect test. The default value is disabled.

Note – The `l3sramtest` automatically handles processor binding. Do not use the Processor Affinity option for the `l3sramtest`.

`l3sramtest` Test Modes

TABLE 11-2 `l3sramtest` Supported Test Modes

Test Mode	Description
Exclusive	Performs only the `l3sramtest` (full test).

`l3sramtest` Command-Line Syntax

`/opt/SUNWvts/bin/sparcv9/l3sramtest` [`-standard_arguments`] [`-o`

[dev=*l3sram*][,count=*1..1024*][,em=*Enabled* | *Disabled*][,threshold=*0..255*][,coresync=*Enabled* | *Disabled*][,corethreads=*1..4*][,sso=*Enabled* | *Disabled*]]

Note – The `l3sramtest` is not a per CPU test. There will be only one `l3sramtest` for the whole system (one image of Solaris). `l3sramtest` runs on all the CPUs of the domain.

TABLE 11-3 `l3sramtest` Command-Line Syntax

Argument	Description
dev=*l3sram*	Specifies the device. The default value is `l3sram`.
count=*number*	Specifies the number of thrashing cycles that the test completes for the level3 cache on the system. Default value for offline mode is 8.
em=*Enabled/Disabled*	Specifies the enabling or disabling of the ECC Error Monitor. The default value is disabled.
threshold=*number*	Specifies the threshold value of how many correctable ECC errors can occur in the elapsed time before `l3sramtest` reports a test failure. The default value is 1.
coresync= [*Enabled/Disabled*]	Specifies the enabling or disabling of the core syncronization mode. When this option is enabled, each CPU core runs with exclusive access to the cache during the entire cache test execution. The default value is disabled.
corethreads= *number*	Specifies the number of threads spawned per core. This is applicable in `coresync` and `sso` options only. The default value is 2.
sso= [*Enabled/Disabled]*	Specifies the enabling or disabling of the internal cache interconnect test. The default value is disabled.

CHAPTER 12

Multiprocessor Test (mptest)

The mptest verifies the hardware functionality of multiprocessor hardware. The test provides diagnostic test coverage for different aspects of multiprocessor functionality such as E-Cache Coherency, Synchronization Primitives, I/O Cache Coherency and Shared Memory, and Interprocessor Interrupts.

The mptest is adaptive to different cache size and line sizes. The test causes cache coherency operations for E-Cache and I/O Cache. It also tests the synchronization primitives provided by the sparcv8/sparcv9 architecture.

Note – mptest is not supported on sun4m platforms in SunVTS 5.1 PS6 onward.

Caution – mptest by default selects the CPUXCall class of test method. If CPUXCall is selected, and mptest is run, the machine might seem hung for a few minutes. The duration is dependent on the number of CPUs.

mptest Options

To reach the dialog box below, 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

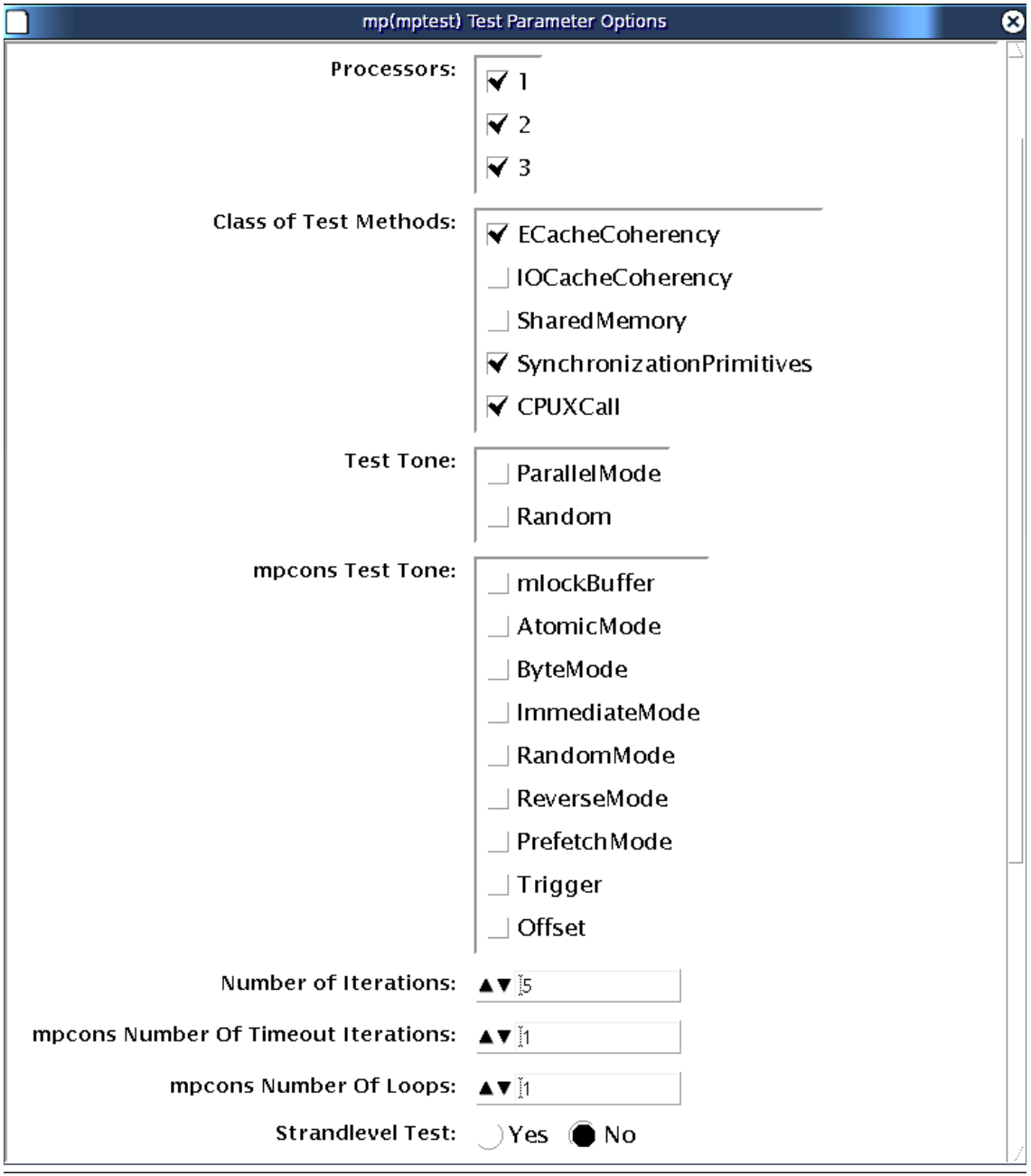

FIGURE 12-1 `mptest` Test Parameter Options Dialog Box

The processors that can be tested are listed in the Configuration area of the menu. You can enable or disable the multiprocessing test for individual processors on this menu.

The options listed in TABLE 12-1 can be run alone or concurrently with other options.

TABLE 12-1 `mptest` Options

`mptest` **Options**	**Description**
Processors	This option can be used to select the CPU IDs for which to run this test. The test will use all CPUs on the system by default. Hence, this parameter is optional. The CPU IDs currently present in the system can be retrieved with the `psrinfo`(1M) command. Specifying a CPU ID not present in the system or one which is currently offline induces an appropriate error message from the test.
Class of Test Methods	The Multiprocessor (MP) functionality consists of different components. A class of test method is used to specify the functionality of the MP system to be tested. Currently, the Class-of-Test methods supported by `mptest` are: E-CacheCoherency, IOCacheCoherency, SynchronizationPrimitives, and SharedMemory, and CPUXCall. This option can be used to selectively test one or more of the MP functions. If you do not specify the class of test methods, E-CacheConsistency and SynchronizationPrimitives are selected by default.
Test Tone	A test tone is a different way of executing the same test. Selecting a different test tone will exercise and test the MP functionality in a slightly different manner. The tone option can be used to select the test tone for the test. The supported test tones are: Random and Parallel mode. The "Random" test tone introduces some randomness in testing. The "ParallelTone" implies that the tests perform parallel operations (like write) on different CPUs at the same time. This tone option is optional. If you do not specify any option, then the test assumes a normal tone of testing.

TABLE 12-1 `mptest` Options *(Continued)*

`mptest` **Options**	**Description**
`mpcons` Test Tone	Selects the tone for `mpconstest` cases in the `mptest`. These options are directly from the `mpconstest`. The descriptions from the `mpconstest` options are as follows: • `mlockBuffer` – Lock the shared buffer in memory • `AtomicMode` – Enable Atomic mode • `ByteMode` – Enable Byte mode • `ImmediateMode` – Enable Immediate mode • `RandomMode` – Enable Random mode • `ReverseMode` – Reverse direction to decrement through memory • `PrefetchMode` – Enable use of V9 prefetch instructions • `Trigger` – Enable LA trigger on error • `Offset` – Enable use of linesize buffer offsets This option is not mandatory. By default, the `mpcons_tone` assumes a normal mode of operation.
Number of Iterations	Same option as in `mpcontest`. This option is used to select the number of iterations for running the test loops. The range for this option is 1 to 200 and the default is 5.
Number Of Timeout Iterations	Same option as in `mpcontest`. Selects number of timeout iterations.
CPU Wait Count 0	Same option as in `mpcontest`. Forces CPU 1 to write first if number of CPUs is less than *count*.
Number of Loops	Same option as in `mpcontest`. Sets loops to specified value.
Memory Size 0—Use Default	Same option as in `mpcontest`. Specifies memory size (MB). This should always be set to the default value.
Random Mode Seed 0	Same option as in `mpcontest`. Sets random number seed to specified value.
Strand Level Test	Performs strand level testing for high end multi-stranded processors. This option selects all strands for testing instead of selecting one strand/core. Thes option can be set to Yes or No.

mptest Test Modes

TABLE 12-2 mptest Supported Test Modes

Test Mode	Description
Exclusive	This test mode tests the user selected multiprocessor functionality.

mptest Command-Line Syntax

/opt/SUNWvts/bin/mptest *standard_arguments* M=*4+5+6+7*, method= ECacheCoherency+IOCacheCoherency+SynchronizationPrimitives+SharedMemory+ CPU Call, tone=*ParallelMode+Random*, mpcons_tone= *mlockBuffer+AtomicMode+ByteMode+ImmediateMode+RandomMode+ReverseMode+Prefet chMode+Trigger+Offset*, count=*[1-200]*, mpcons_numtmout=*[1-10]*, mpcons_wait=*0*, mpcons_loops=*[1-999]*, mpcons_memsize=*0*, mpcons_seed=*0*, strandleveltest=*Yes*|*No*

TABLE 12-3 `mptest` Command-Line Syntax

`mptest` **Options**	**Description**
`M=`*4+5+6+7*	Selects the CPU IDs for which to run this test. The test uses all CPUs on the system by default. Hence, this parameter is optional. The CPU IDs currently present in the system can be retrieved with the `psrinfo`(1M) command. Specifying a CPU ID not present in the system or one which is currently offline induces an appropriate error message from the test. Example: If you want to select CPU IDs 4, 5, 6 and 7, specify: `M=`*4+5+6+7*. The behavior is little different for high end multi-stranded processors. Each ID corresponds to a CPU ID of a strand in any core. If `strandleveltest` is set to No, only one CPU from a strand is chosen. To override this behavior, that is. to select more than one strand from a core, the `strandleveltest` option must be set to Yes.
`method=` *ECacheCoherency+IOCacheCoherency+SynchronizationPrimitives+SharedMemory+CPUXCall*	Selectively tests one or more of the MP functions. If you do not specify the class of test methods, E-CacheConsistency, SynchronizationPrimitives, and CPUXCall are selected by default. The Multiprocessor (MP) functionality consists of different components. A class of test method is used to specify the functionality of the MP system to be tested. Currently, the Class-of-Test methods supported by `mptest` are: E-CacheCoherency, IOCacheCoherency, SynchronizationPrimitives, SharedMemory, and CPUXCall.
`tone=`*ParallelMode+Random*	A test tone is a different way of executing the same test. Selecting a different test tone exercises and tests the MP functionality in a slightly different manner. The tone option can be used to select the test tone for the test. The supported test tones are: Random and Parallel mode. The Random test tone introduces some randomness in testing. The ParallelTone implies that the tests perform parallel operations (like write) on different CPUs at the same time. This tone option is optional. If you do not specify an option, the test assumes a normal tone of testing.

TABLE 12-3 `mptest` Command-Line Syntax *(Continued)*

`mptest` **Options**	**Description**
`mpcons_tone=` mlockBuffer+AtomicMode+ByteMode +ImmediateMode+RandomMode+Rev erseMode+PrefetchMode+Trigger+Off set	Selects the tone for `mpconstest` cases in the `mptest`. These options are directly from the `mpconstest`. The descriptions from thes `mpconstest` options are as follows: • `mlockBuffer` – Lock the shared buffer in • memory`AtomicMode` – Enable Atomic • mode`ByteMode` – Enable Byte • mode`ImmediateMode` – Enable Immediate • mode`RandomMode` – Enable Random Mode • `ReverseMode` – Reverse direction to decrement through memory • `PrefetchMode` – Enable use of V9 prefetch instructions • `Trigger` – Enable LA trigger on error • `Offset` – Enable use of linesize buffer offsets This option is not mandatory. By default, the `mpcons_tone` assumes a normal mode of operation.
`count=`*[1-200]*	Selects the number of iterations for running the test loops. The range for this option is 1 to 200 and the default is 5.
`mpcons_numtmout=`*[1-10]*	Same option as in `mpcontest`. Selects number of timeout iterations.
`mpcons_wait=`*0*	Same option as in `mpcontest`. Forces CPU 1 to write first if number of CPUs is less than *count*.
`mpcons_loops=`*[1-999]*	Same option as in `mpcontest`. Sets loops to specified value.
`mpcons_memsize=`*0*	Same option as in `mpcontest`. Specifies memory size (MB). This should always be set to the default value.
`mpcons_seed=`*0*	Same option as in `mpcontest`. Sets random number seed to specified value.
`dev=`*mp*	Specifies the device.
`strandleveltest=`*Yes*|*No*	Enables the strand level testing and selects all strands for testing. The default option No performs the most optimal testing and is the most generally used. This option doesnt have any effect on non-multistranded processors such as UltraSPARC IV+.

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 may only be available as a 32-bit test. For more information refer to the "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

CHAPTER 13

Sun™ XVR-100 Graphics Accelerator Test (`pfbtest`)

`pfbtest` tests the PCI-based Sun™ XVR-100 graphics accelerator by performing the following subtests:

- Video Memory test
- RAMDAC test
- Accelerator Port test

Caution – *Do not* run any other application or screen saver program that uses the Sun XVR-100 graphics accelerator port while running `pfbtest`. This program causes SunVTS to return incorrect errors.

Caution – If pfb0b is set to display, an error similar to the following occurs:
Accelerator: signature err or in test Ramdac.
Display should always be set to pfb0a when running SunVTS.

Note – Disable all screen savers before testing any graphics device. Type **`xset s off`** at a UNIX prompt to disable the Solaris screen saver. Type **`xset -dpms`** (to turn off power management) or type **`xset s noblank`** (to turn off screen saver). Disable Power Management software if it is running.

Note – To start SunVTS with `vtsui`, but without `vtsk`, you must add the host name to `xhost` as: **`xhost +`** *hostname.*

For full instructions on testing frame buffers, refer to the Testing Frame Buffers section of the *SunVTS 5.1 Test Reference Manual*.

`pfbtest` Options

To reach the dialog box below, 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

By default, all options are enabled except frame buffer locking.

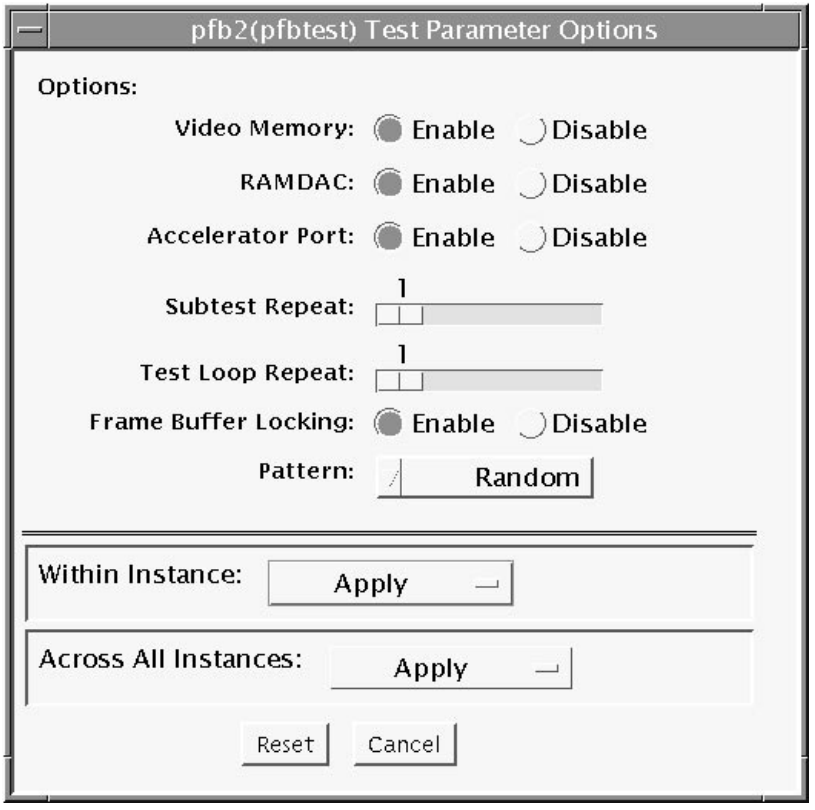

FIGURE 13-1 `pfbtest` Test Parameter Options Dialog Box

TABLE 13-1 `pfbtest` Options

`pfbtest` **Options**	**Description**
Video Memory test	Thoroughly tests the on-screen video memory (the memory part that is mapped on to the monitor) of the Sun XVR-100 graphics accelerator in 8-bit, 16-bit, 32-bit, 64-bit, and 64 byte (block) modes. Entire on-screen video memory is tested by testing 512 bit blocks at a time (8x8 pixel block). Each block is tested in two passes. Each pass consists of a data write and read. In the first pass, user specified data or 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 (electrical low state) and one (electrical high state).
RAMDAC test	Tests the RAMDAC in three phases. In the first phase the RAMDAC CLUT (Color LookUp Table) is tested using simple write/read patterns to determine if there are any bad bits in CLUT. The data patterns used are: • Random data • Complement of the random data (used as first data pattern) • The data pattern 0101 • The data pattern 10101 In the second phase, four different patterns are drawn on the screen. Each pattern stays on the screen for approximately 1/4 second. The four patterns are listed below. For each pattern, the signature is captured and compared with the signature obtained for the same pattern on a known good board. This test verifies that all the different data paths within the RAMDAC are functioning properly. Patterns drawn on screen: • Red ramp with cursor at top-left corner of the screen • Blue ramp with cursor at top-right corner of the screen • Green ramp with cursor at bottom-left of the screen • Grey ramp with cursor at bottom-right of the screen In the last (third) phase of the RAMDAC test the Vertical Retrace Interrupt is tested for approximately five seconds.
Accelerator Port test	Tests all of the following: • Data paths (sources: fixed color, host data, blit, fixed pattern) • Arithmetic and logic unit (ALU) • Primitives (destinations: line, rectangle) • Mono to color expansion logic Primitives are drawn using a combination of different data paths (allowed), ALU functions, and color comparator functions. A checksum is generated for each data combination and is compared with the checksum generated for the same data combination on a known good board.

TABLE 13-1 `pfbtest` Options *(Continued)*

`pfbtest` **Options**	**Description**
Frame Buffer Locking	This option is set to *disable* if the Sun XVR-100 graphics accelerator is not the console device. When the SunVTS GUI is brought up, Frame Buffer Locking is enabled by default if the Sun XVR-100 graphics accelerator is the console device. If the Sun XVR-100 graphics accelerator is not the console device, Frame Buffer Locking is disabled by default.

`pfbtest` Test Modes

Due to the nature of graphics tests, reading from or writing to the frame buffer during graphics tests will disturb user operation. This test is only available in the Functional test mode.

TABLE 13-2 `pfbtest` Supported Test Modes

Test Mode	**Description**
Functional	The `pfbtest` verifies the proper functioning of Sun XVR-100 graphics accelerator.

pfbtest Command-Line Syntax

/opt/SUNWvts/bin/pfbtest *standard_arguments* **-o dev=***device_name***, S=***subtest_number*,**F=**#*_of_subtest_loops*,**B=**#*_of_test_loops*,L=***disable*,**P=***test_pattern*

TABLE 13-3 pfbtest 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. The default is pfb0.
S=*subtest_number*	*subtest_number* is the test number of the subtest to be run. Select from the subtests below. You can run multiple subtests by adding the subtest numbers. For example, n=0x3 runs both test 1 and test 2; n=0x5 runs both test 1 and test 4. • n 0x1 VRAM • n 0x2 RAMDAC • n 0x4 Accelerator port test (Rendering Pipeline) More than one test can be selected by ORing subtest numbers. For example: n = 0x5 indicates VRAM and Rendering Pipeline tests. A hex number must be preceded by 0x, decimal numbers are also acceptable.
F=#*_of_subtest_loops*	Specifies the number of times to repeat each subtest. The default is 1.
B=#*_of_test_loops*	Specifies the number of times to repeat a test loop before passing; the default is 1.
L=*disable*	Disables the frame buffer lock. Disable the lock when the Sun XVR-100 graphics accelerator is not the console or when the server is not running on the Sun XVR-100 graphics accelerator under test.
P=*test_pattern*	Specifies the test pattern number. The default is r, for random patterns. You may also choose 0 for 0x0000000, 3 for 0x3333333, 5 for 0x5555555, or 9 for 0x9999999.

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 may only be available as a 32-bit test. For more information refer to the "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

Note – Errors returned by `pfbtest` are nonspecific. It is not possible to determine which component caused a failure. In all error conditions, the field replaceable unit (FRU) is the entire Sun XVR-100 graphics accelerator.

CHAPTER 14

RAM Test (ramtest)

ramtest is designed to stress the memory modules (RAM) instead of the whole memory subsystem. The test is optimized to achieve large memory bandwidth on UltraSPARC III (USIII) and UltraSPARC II (USII) class of CPUs. ramtest has an integrated ECC error monitor which reports the ECC errors found during the test run.

This test is being added only for the Exclusive mode testing because of the high stress it puts on the memory and the system interconnect. ramtest assumes that no other application is running at the same time.

Note – disktest is supported on x86 platforms that use the Solaris Operating System.

Caution – This is an Exclusive mode test. No other application should be running during this test.

ramtest Options

To reach the dialog box below, 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. Refer to the *SunVTS User's Guide* for more details.

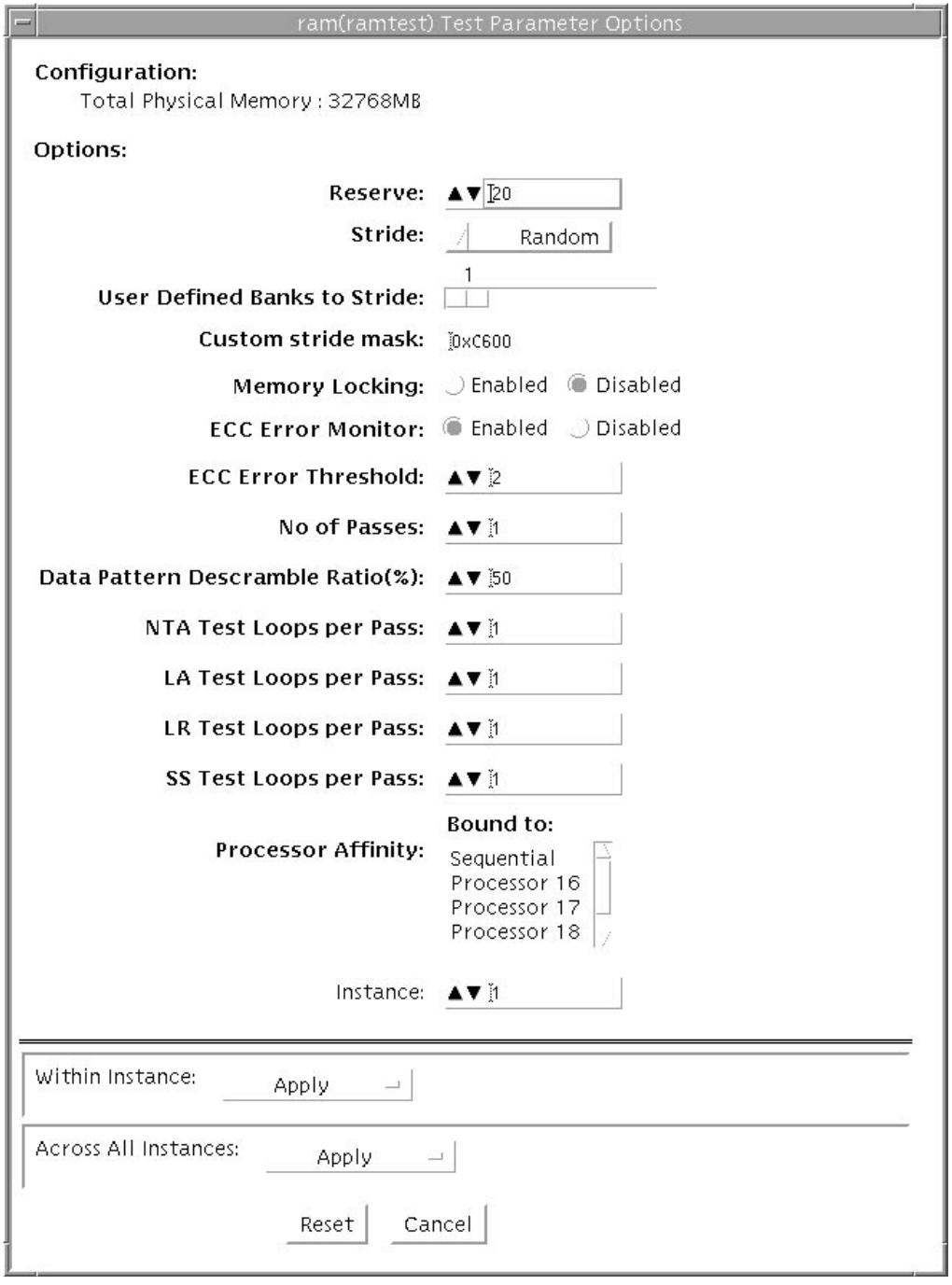

FIGURE 14-1 `ramtest` Test Parameter Options Dialog Box

The following table details the `ramtest` options:

TABLE 14-1 `ramtest` Options

`ramtest` **Options**	**Description**
Reserve	Reserve option represents the percentage of physical memory that is assumed to be in use by the OS or other processes. If you see excessive swapping while running ramtest, increase this percentage. The default is 20%; this means that ramtest allocates 80% of physical memory size for testing. Swapping decreases stress on memory and increases it on the system itself. For memory testing purposes, it is recommended to minimize swapping by tuning the reserve option. If for some reason the allocation or locking (in case Memory Locking is enabled) does not succeed, the amount of memory is reduced and the allocation process is repeated. Once the allocation succeeds, the amount of memory allocated is displayed in the messages.
Stride	By default this option is set to "Random". It can be set to "Column" or "Row" also. In case of random, either Row or Column are randomly selected for each pass. Value of stride defines the memory locations addressed consecutively in certain subtests, in a hardware dependent manner. All testable memory is still tested. Using different strides, checks coupling among different sets of memory cells; therefore random is the recommended value for this option unless both Column and Row are being explicitly used in different instances. For FA type of uses, stride may also be set to "UserDefined", in this case the test will stride the number of banks specified in the "userstride" option. Stride may be set to "Custom" in which case the stride values are randomly selected from the strides specified in the "stridemask" value. Stride may be set to “Custom” in which case the stride values are randomly selected from the strides specified in the “stridemask” value.
User-Defined Banks to Stride	Use this option to set the number of banks that the test should stride. One recommended choice is the interleave on the suspect bank, during FA. The value is currently limited to between 1 and 16. (This also means row striding is not possible while using this option).

TABLE 14-1 `ramtest` Options *(Continued)*

`ramtest` **Options**	**Description**
Stridemask	When stride=custom is selected, this value specifies the strides used. Each thread selects one of the stride values from stridemask by selecting one of the bits in the mask. The bits in the stridemask value represent the Least Significant Bit of the stride. Thus a value of 0x4000 calls for a stride of 16384 (using Bit 14 of the address). Mulitple bits can be set mixing row and column strides. Consult the Memory Controller section of the PRM for the CPU of the test system to discover how the memory reference address is divided between rows and columns in the DRAM. The value can be specified as a Decimal (NNN), Hexadecimal (0xNNN), or Octal (0NNN) value. The maximum value is 0x400000 (4194304). The default value is 0xC600 which represents strides using Bits 15, 14, 10, and 9.
Memory Locking	By default memory locking is "Disabled". To turn it on, set lock to "Enabled". This test uses ISM to lock the memory into the core, this gives 4 MB virtual pages and avoids swapping. Running without locking on the other hand, adds more randomness to the addressing sequence.
ECC Error Monitor	ECC Monitor is "Enabled" by default. The ECC error monitor runs as a separate thread in the test. When an ECC error is detected, the message is displayed on to the test output. The monitor can be turned off by setting this option to "Disabled". The ECC Monitor option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
ECC Error Threshold	This is the number of ECC errors after which the test will stop (if ECC monitor is running). When the threshold is reached, the test will exit with a non zero exit code. If set to zero, the test will still report all the errors but will not stop. The default of threshold is 2. The ECC Threshold option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
Number of Passes	This option specifies the number of passes, in the same instance. Increasing passes is recommended in case "lock" is enabled, this will save time spent on locking the memory every time a new process/instance is spawned by the VTS kernel. Note that this pass has no relation with the system passes in the VTS infrastructure, it will appear that `ramtest` is taking longer to complete system passes.

TABLE 14-1 `ramtest` Options *(Continued)*

`ramtest` **Options**	**Description**
NTA March Test	Specifies number of loops of NTA march(30N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. NTA march test attacks coupling and stuck at faults. NTA march is efficient at finding single, double, and some triple bit errors. Depending on the stride option, coupling faults between cells in adjacent columns, or rows that are targeted. Note that test time will be higher when row striding is selected because of greater page faults generated. For efficiency purposes, total memory is divided among available CPUs.
LA March Test	Specifies number of loops of LA march(22N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. LA march test attacks coupling and stuck-at-faults.
LR March Test	Specifies number of loops of LR march(14N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. LR march test attacks coupling and stuck-at-faults.
SS March Test	Specifies number of loops of SS march(22N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. SS march test attacks simple static faults. The SS March option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.

ramtest Test Modes

TABLE 14-2 ramtest Supported Test Modes

Test Mode	Description
Exclusive	Generates enormous amount of memory traffic.

ramtest Command-Line Syntax

/opt/SUNWvts/bin/sparcv9/ramtest *standard_arguments* **[-o**

[reserve=<*Integer between 0 and 90*> **]**
[stride=<*Row* | *Column* | *Random* | *UserDefined* | *Custom*>
[userstride=<*1 - 16*> **]**
[stridemask=<*0x40 - 0x400000*>]
[lock=<*Enabled* | *Disabled*> **]**
[dratio=<*Integer between 0 and 100*> **]**
[eccmonitor=<*Enabled* | *Disabled*> **]**
[threshold=<*Integer i; 0 <= i <= MAX_INT* >**]**
[pass=<*32 bit integer*> **]**
[ntaloops=<*32 bit integer*>**]**
[laloops=<*32 bit Integer*> **]**
[lrloops=<*32 bit Integer*> **]**
[ssloops=<*32 bit Integer*>]]

TABLE 14-3 `ramtest` Command-Line Syntax

Argument	Description
`reserve`	This is used to specify the amount of memory that will not be allocated for testing. Reserve represents a percentage of the total physical memory in the system. When the test starts, it probes the total memory present in the system, then tries to allocate (100 - reserve)% of memory. If the allocation or locking does not succeed the amount of memory is reduced before the retry. Before starting the test, the amount of memory allocated for testing is displayed. Default value for reserve option is 20. For US IIIi platforms, default value is tuned to 25. It should be noted that on low memory systems, the reserve value should be kept higher to avoid excessive swapping. For 32-bit booted systems, the reserve value represents the percentage of 4 GB rather than the percentage of total physical memory.
`stride`	By default stride is set to "Random". It can be set to "Column" or "Row" also. In case of random, either Row or Column are randomly selected for each pass. Value of stride defines the memory locations addressed consecutively in certain subtests, in a hardware dependent manner. All testable memory is still tested. Using different stride checks coupling among a different set of memory cells, therefore random is the recommended value for this option unless both column and row are being explicitly used in different instances. For FA type of uses, stride may also be set to "UserDefined", in this case the test will stride the number of banks specified in the "userstride" option. Stride may be set to "Custom" in which case the stride values are randomly selected from the strides specified in the "stridemask" value.
`userstride`	Use this option to set number of banks the test should stride. One of the good choices could be the interleave on the suspect bank, during FA. the value is limited between 1 and 16 right now. (This also means row striding is not possible while using this option).
`stridemask`	When stride=custom is selected, this value specifies the strides used. Each thread selects one of the stride values from stridemask by selecting one of the bits in the mask. The bits in the stridemask value represent the Least Significant Bit of the stride. Thus a value of 0x4000 calls for a stride of 16384 (using Bit 14 of the address). Mulitple bits can be set mixing row and column strides. Consult the Memory Controller section of the PRM for the CPU of the test system to discover how the memory reference address is divided between rows and columns in the DRAM. The value can be specified as a Decimal (NNN), Hexadecimal (0xNNN), or Octal (0NNN) value. The maximum value is 0x400000 (4194304). The default value is 0xC600 which represents strides using Bits 15, 14, 10, and 9.

TABLE 14-3 `ramtest` Command-Line Syntax

Argument	Description

`lock`	By default memory locking is "Disabled". To turn it on set lock to "Enabled". The test uses ISM to lock the memory into the core, this gives 4 MB virtual pages and avoids swapping. Running without locking on the other hand, adds more randomness to the addressing sequence.
	It should be noted that on low memory systems, this option can be "Enabled" to avoid excessive swapping.
	In case the test is unable to lock the memory, the user should put the following lines in `/etc/system` and reboot the machine.
	`set shmsys:shminfo_shmmax=0xFFFFFFFFFFFFFFFF`
	`set shmsys:shminfo_shmmin=1`
	`set shmsys:shminfo_shmmni=100`
	`set shmsys:shminfo_shmseg=10`

`eccmonitor`	ECC Monitor is "Enabled" by default. The ECC error monitor runs as a separate thread in the test. When an ECC error is detected, the message is displayed on to the test output. The monitor can be turned off by setting this option to "Disabled". The ECC Monitor option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
`threshold`	This is the number of ECC errors after which the test will stop (if ECC monitor is running). When the threshold is reached the test will exit with a non zero exit code. If set to zero, the test will still report all the errors but will not stop. The default threshold is 2. The ECC Threshold option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
`pass`	This option specifies number of passes, in the same instance. Increasing pass is recommended in case "lock" is enabled, this will save time spent on locking the memory every time a new process/instance is spawned by the VTS kernel. Note that this pass has no relation with the system passes in the VTS infrastructure, it will appear that `ramtest` is taking longer to complete system passes.
`ntaloops`	Specifies number of loops of NTA march(30N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. NTA march test attacks stuck-at-faults, two cell coupling faults, and some three cell coupling faults.
`laloops`	Specifies number of loops of LA march(22N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. LA march test attacks coupling and stuck-at-faults.

TABLE 14-3 `ramtest` Command-Line Syntax

Argument	Description
`ntaloops`	Specifies number of loops of NTA march test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. NTA march test attacks coupling and stuck at faults.
`lrloops`	Specifies number of loops of LR march(14N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. LR march test attacks coupling and stuck-at-faults.
`dratio`	Descrambles ratio can be used to tune the algorithm used to generate data patterns in `ramtest`. Descramble ratio of 100 means that all the data patterns generated will be descrambled. Where as if descramble ratio is 0, the test will generate the data patterns tuned towards bus noise. Default value is 50, which means that half the data patterns are descrambled.
`ssloops`	Specifies number of loops of SS march(22N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. The SS March test attacks simple static faults. The SS March option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
`custom`	When `stride=custom` is selected, this value specifies the strides used. Each thread selects one of the stride values from stridemask by selecting one of the bits in the mask. The bits in the stridemask value represent the Least Significant Bit of the stride. Thus a value of 0x4000 calls for a stride of 16384 (using Bit 14 of the address). Mulitple bits can be set mixing row and column strides. Consult the Memory Controller section of the PRM for the CPU of the test system to discover how the memory reference address is divided between rows and columns in the DRAM. The value can be specified as a Decimal (NNN), Hexadecimal (0xNNN), or Octal (0NNN) value. The maximum value is 0x400000 (4194304). The default value is 0xC600 which represents strides using Bits 15, 14, 10, and 9.

Note – 32-bit tests are located in the bin subdirectory, `/opt/SUNWvts/bin/`*testname*.

Note – ECC errors returned by `ramtest` are actually detected by the operating system and are logged in the `/var/adm/messages` file. Please review this file for more detailed information regarding errors.

Note – 64-bit tests are located in the `/bin/64` directory, or the relative path in which you installed SunVTS. If a test is not present in this directory, then it might be available as a 32-bit test only.

CHAPTER 15

System Test (systest)

The systest checks the overall functionality of a Sun system by exercising the CPU, I/O, and memory channels.

Note – systest does support x86 platforms on the Solaris OS.

Note – systest has been enhanced to enable the CPU test subtest on x86 systems. systest now supports the same set of options for both SPARC and x86.

The test ensures the concurrency of the different channels by the use of Solaris threads. The test aims at stimulating failures that might be caused due to the interaction of the various different hardware modules in the system. It is very stressful on the CPU, and stresses the parallel computational capability of a multiprocessor system.

systest 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

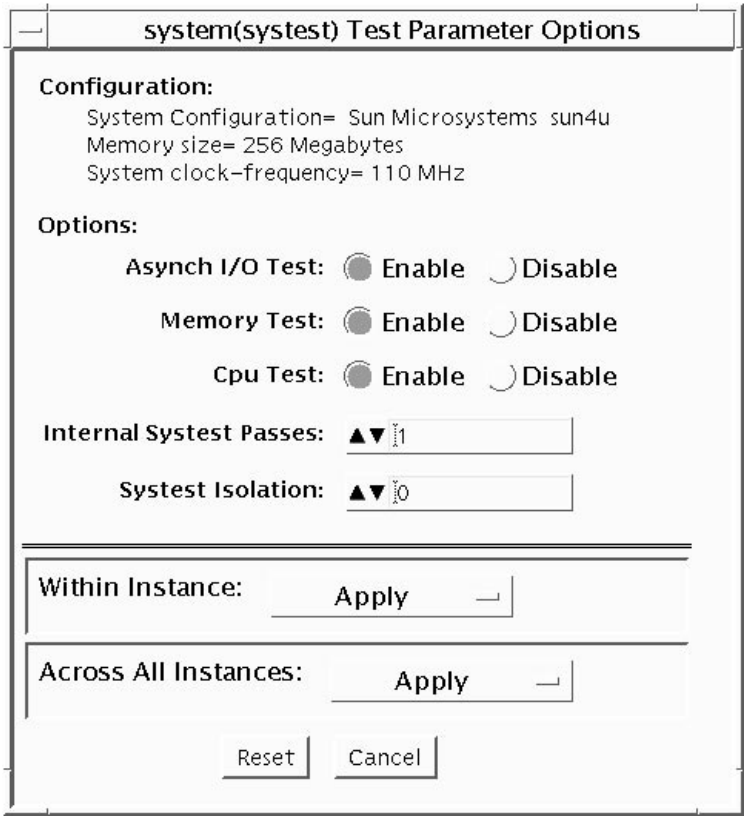

FIGURE 15-1 `systest` Test Parameter Options Dialog Box

Caution – Use discretion when defining the syspass parameter. One lapack pass (syspass=1) takes approximately 40 minutes on a server with 12 UltraSPARC® III processors. If the syspass value is set to a high value, it also increases the probability of detecting residual errors.

Caution – Use strong discretion when defining the System Isolation (sysiso) parameter. BE AWARE THAT sysiso MAY ONLINE / OFFLINE CPUs IN THE SYSTEM. DO NOT USE `sysiso` ON PRODUCTION SERVERS. If you choose CPUs (`sysiso`=2) Isolation, the run time may be much higher than for board(s) (`sysiso`=1) Isolation. The total run time for Isolation can not be precisely estimated. If a residual error is found in the initial evaluation phase, the Isolation functionality will online / offline CPUs in order to detect the defective boards and CPUs in the system.

Note – Users are advised to not use the Processor Affinity option for this test. Doing so reduces the effectiveness of the test.

TABLE 15-1 `systest` Options

`systest` **Options**	**Description**
Asynch I/O Test	Enables or disables the Asynch I/O subtest. The default is enable.
Memory Test	Enables or disables the Memory subtest. The default is enable.
Cpu Test	Enables or disables the CPU/FPU subtests. The default is enable.
Internal System Passes	Defines the number of internal lapack passes. A set of boards and CPUs will be declared GOOD after `syspass` number of passes. The default is 1.
System Isolation	Defines the type of Isolation that `systest` needs to perform if a residual error is found in the initial evaluation phase. 0 = No Isolation (default) 1 = Board(s) Isolation only 2 = Board(s) and CPUs Isolation

The default values are recommended for an initial evaluation of the system.

`systest` Test Modes

TABLE 15-2 `systest` Supported Test Modes

Test Mode	Description
Exclusive	Performs only the `systest` (full test).

`systest` Command-Line Syntax

`/opt/SUNWvts/bin/systest` *standard_arguments* **`-o -io=`***enable*`|`*disable* **`-mem=`***enable*`|`*disable*`,` **`-cpu=`***enable*`|`*disable*`,` **`-dev=`**system, **`-syspass=`***1,2000*`,` **`-sysiso=`***0*`|`*1*`|`*2

TABLE 15-3 `systest` Command-Line Syntax

Argument	Description
`io=`*enable*`|`*disable*	Enables or disables the Asynch I/O subtest.
`mem=`*enable*`|`*disable*	Enables or disables the Memory subtest.
`cpu=`*enable*`|`*disable*	Enables or disables the CPU/FPU subtests. The CPU Test is not supported on x86 platforms.
`dev=`*system*	Specifies the pseudo device name.
`syspass=`*1,2000*	Defines the number of internal lapack passes. A set of boards and CPUs will be declared "GOOD" after "`syspass`" number of passes. The default is 1.
`sysiso=`*0*`|`*1*`|`*2	Defines the type of Isolation that `systest` needs to perform if a residual error is found in the initial evaluation phase. 0 = No Isolation 1 = Board(s) Isolation only 2 = Board(s) and CPUs Isolation

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-Bit and 64-Bit Tests" on page 4.

Recommended Option Selection

The default values are recommended for an initial evaluation of the system.

Command-Line Examples

The following examples assume you want to execute `systest` from the command-line with verbose enabled.

Example 1:

```
# ./systest  -xv
```

The above example invokes the following:

- `systest` with default parameter values.
- I/O, MEM, and CPU subtests. The CPU test is not supported on x86 platforms.
- One internal pass of lapack and no Isolation.

Example 2:

```
# ./systest  -xv -o io=Disable,mem=Enable,cpu=Enable,dev=system
```

The above example invokes the following:

- `systest` without the I/O subtest
- MEM and CPU subtests. The CPU test is not supported on x86 platforms.
- One internal pass of lapack and no Isolation.

Caution – Do not perform the following `systest` examples (3 and 4) on production servers because `systest` may online / offline CPUs.

Example 3:

```
# ./systest  -xv -o syspass=15,sysiso=1
```

The above example invokes the following:

- I/O, MEM, and CPU subtests. The CPU test is not supported on x86 platforms.
- Declares a set of boards free from residual errors after 15 internal passes of the lapack algorithm.
- If an error is found, `systest` will perform boards isolation.

Example 4:

```
# ./systest  -xv -o syspass=10,sysiso=2
```

The above example invokes the following:

- I/O, MEM, and CPU subtests. The CPU Test is not supported on x86 platforms.
- Declares a set of boards AND CPUs free from residual errors after 10 internal passes of the lapack algorithm
- If an error is found, `systest` will perform boards AND CPUs isolation.

CHAPTER 16

Tape Drive Test (tapetest)

The tapetest synchronous I/O test writes a pattern to a specified number of blocks (or, for a SCSI tape, writes to the end of the tape). The tapetest then rewinds the tape and reads and compares the data just written. The tapetest asynchronous I/O test sends a series of up to five asynchronous read/write requests to the tape drive, writing to the tape and then reading and comparing the data. The terms asynchronous and synchronous referred to here, and in the method field are not related to the SCSI messaging terms of the same name.

The tapetest file test writes four files to the tape and then reads them back, comparing the data. For tape library testing, the pass count is incremented only after all tapes in the library have been tested. The read/write algorithms fare enhanced for DLT tape by using a random data pattern (1.5:1 compression) and a more robust read compare algorithm. Some default parameters have also changed.

tapetest Test Requirements

If you have a tape drive in your system, load a blank writable tape (scratch tape) before you start SunVTS. If you fail to do this, the tapetest option may display drive type:unknown on the option menu for the tapetest.

tapetest 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 may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

`tapetest` supports 4-mm, 8-mm, DLT, 1/4-inch cartridge, and 1/2-inch front-load tape drive testing. The options available for each of the tape devices differ slightly. An example of the Options dialog box for a device is shown in FIGURE 16-1.

The Async I/O subtest uses the asynchronous read and write feature of the Solaris tape driver to exercise tape drives. In read-only mode the test sends a maximum of four asynchronous read packets, each with a random size and a random offset, to the tape drive. The test then waits for all outstanding I/O activity to complete before issuing another round of packets. This process continues until the whole area being tested has been covered. In read-write mode, one write packet is issued for every four read packets to ensure a spot check of the write operation. The area of the tape to be tested is written to first in order for the test to work correctly. This test is only supported under the Solaris 8 and Solaris 9 OSs and future compatible releases.

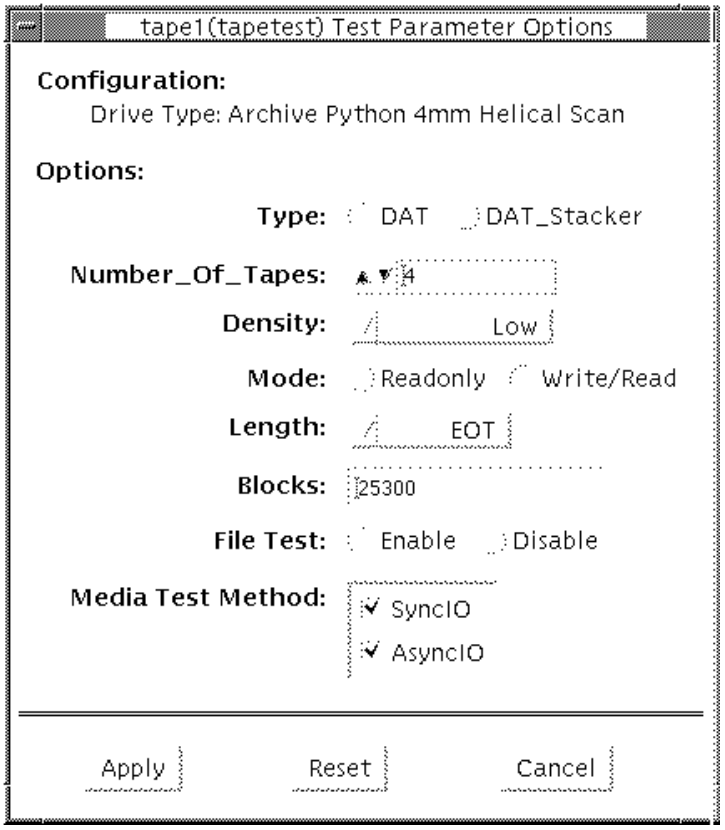

FIGURE 16-1 `tapetest` Test Parameter Options Dialog Box

Note – This test does not immediately stop after being disabled.

Note – Selecting nondefault options can significantly increase the run time for the test.

Note – The Options dialog box for the 1/4-inch, 1/2-inch, DLT, and 8-mm tape drives differ slightly from FIGURE 16-1.

TABLE 16-1 `tapetest` Options

`tapetest` **Options**	**Description**
Type	Normal tape drive or tape library (stacker).
Number of Tapes	The number of tapes in the tape library. `tapetest` registers a single tape library pass only after all tapes in the library pass.
Density	The following settings are available for most tape drives: • Low—Tests the `l` tape device. • Medium—Tests the `m` tape device. • Compression—Tests the `c` tape device. • All—Tests the `l`, `m`, and `c` tape devices. For half-inch tape drives, the available settings are 800, 1600, and 6250 BPI (blocks per inch). For certain QIC drives, select `QIC-11` (1-byte block ID) mode, `QIC-24` (4-byte block ID) mode, or `Both`. Note: On a DLT drive, the l and m settings both use no compression. tapetest does not support changing DLT capacity settings indicated on the front panel.
Mode	If you enable Write/Read mode, the test first writes to the tape and then reads it back to compare. If you enable Read_Only mode, the test assumes the tape has been properly written and merely reads and compares. This mode is useful to check proper head alignment. Note: If a read only pass is attempted and the tape was not previously written by tapetest, using the same test parameters currently set, a "Big Read Failure" will occur.
Length	The amount of the tape to be tested. The choices are: • EOT: The default; tests to the entire tape. • Long: The SCSI tape tests 70,000 blocks of the tape. • Short: Only the first 1000 blocks are tested. • Specified: You must type the number of blocks to be tested in the blocks field.

TABLE 16-1 `tapetest` Options

`tapetest` **Options**	**Description**
# of Blocks	If you select Specified under the Length option, you must type the number of blocks you want to test.
Blocksize	Block size specification. This option is only available for Tandberg QIC tape drives. There are two possible values. 512-bytes is for use with older tape media that have transfer size restrictions, while 64-kbytes is for use with current, high-capacity tape media. Note1: This option is only available in command line interface mode. Note2: With patches 110278-01 or 110211-01 applied, DLT writes either a 512 byte or 65536 byte block depending on how this parameter is set.
File Test	The tape file test sequence is as follows: 1. Writes three files. 2. Rewinds. 3. Reads part of the first file. 4. Forward spaces to the start of the second file. 5. Reads the second file. 6. Forward spaces to the start of the third file. 7. Tries to read to the end of that file for SCSI tapes only. The tape file test tries to backspace to the start of the second file and read it.
Retension	When enable is selected, the program retensions the tape.
Media Test Method	• Sync I/O—`tapetest` reads and or writes the number of blocks selected in Length. • Async I/O—`tapetest` makes four asynchronous read requests to the tape drive. If read and write testing is selected, one asynchronous write request is also sent. The test continues after completing the requests. Note – When testing Tandberg QIC drives, Async I/O testing is restricted to read-only due to asynchronous behavior differences with other tape drives. Note: This option is not associated with the synchronous data transfer request SCSI message. It is only synchronous or asynchronous in nature because the numbers of reads and writes are not synchronous to each other. The SDTR message is not invoked.

tapetest Test Modes

The tapetest supports the following test modes. It performs different test schemes on the tape device, according to the mode you select.

TABLE 16-2 tapetest Supported Test Modes

Test Mode	Description
Connection	tapetest verifies that the drive can be opened and that the drive type can be determined. If both checks are successful, or if the drive is currently busy, then the test passes. The tapetest fails if the open operation is unsuccessful for any reason other than the drive is busy.
Functional (Offline)	tapetest checks the status, rewinds the tape, erases and retensions it. If the device is a cartridge tape, tapetest writes a pattern to nblks or eot (default), rewinds the tape, and then reads and compares of the pattern. On the other hand, if the device is busy or if no tape cartridge can be found in the drive, the test cannot run and fails.

tapetest Command-Line Syntax

/opt/SUNWvts/bin/tapetest *standard_arguments* **-o dev=***device_name***,s=***block_count***,d=***density***,m=***mode***,l=***length***,method=***method***,ft=***enables* | *disables***,ret=***enables* | *disables***,dat=***dat_type***,8mm=***8mm_type***,num=***magazine_size***,blocksize=***block_size*

TABLE 16-3 tapetest Command-Line Syntax

Argument	Explanation
dev=*device_name*	Specifies the *device_name* of the tape drive (required).
s=*block_count*	Specifies the number of blocks to be tested.
d=*density*	Specifies the density of the tape to open.
m=*mode*	Enables either the Write_Read or Read_Only tests.
l=*length*	Specifies the length of the test (EOT, Specified, Long, or Short).
method=*method*	Specifies the media test method (SyncI/O and or AsyncI/O) used. Note: This option does not invoke the SCSI message synchronous data transfer request. It is only asynchronous or synchronous in nature.

TABLE 16-3 `tapetest` Command-Line Syntax *(Continued)*

Argument	Explanation *(Continued)*
`ft`=*enables* | *disables*	Enables or disables the File test.
`ret`=*enables* | *disables*	Enable or disables tape retension.
`dat=`*dat_type*	If you are testing a digital audio tape drive, specify whether it is a regular DAT drive or a DAT stacker. The choices are `DAT` and `DAT_Stacker`.
`8mm=`*8mm_type*	If you are testing an 8-mm tape drive, specify whether it is a regular 8-mm tape drive or a tape library. The command line choices are `8mm` and `8mm_Library`.
`num=`*magazine_size*	If you are testing a tape library, specify the magazine size.
`blocksize=` *block_size*	This option is only available on a Tandberg QIC drive and DLT drives. Specify whether to use a 64 kbyte block transfer or a 512 byte block transfer. Use 512 bytes when testing older media in the drive. DLT supports 512 byte and 65536 byte modes

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-Bit and 64-Bit Tests" on page 4.