SunVTS 5.1 Patch Set 5 Documentation Supplement

Sun Microsystems, Inc.
www.sun.com

Part No. 817-4350-10
December 2003, 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™ 5.1 documentation and describes new features, tests, and test enhancements that are developed in the SunVTS 5.1 Patch Set releases. The new features, tests, and test enhancements included in this document are provided in the SunVTS 5.1 Patch Set 5 (PS5) software that is distributed on the Solaris Software Supplement CD.

For overall SunVTS features, test configuration modes, interfaces, and options refer to the *SunVTS 5.1 User's Guide*. Refer to the *SunVTS 5.1 Test Reference Manual* for detailed information on SunVTS test software and the full collection of tests released with SunVTS 5.1.

The new features of the SunVTS software introduced in SunVTS 5.1 Patch Set releases are described in Chapter 2.

The following new test is introduced in the SunVTS 5.1 PS5 release:

JNI 2GB FC HBA Test (`jnifctest`) – described in Chapter 3.

The following tests are enhanced in the SunVTS 5.1 PS5 release:

- Level 1 Data Cache Test (`l1dcachetest`), described in Chapter 4.
- System Service Processor Test (`ssptest`), described in Chapter 7.

The following tests were introduced in the SunVTS 5.1 PS4 release:

- Sun Netra™ 240 Alarm Card Test (`n240atest`), described in Chapter 8.
- RAM Test (`ramtest`), described in Chapter 9.

The following tests were enhanced in the SunVTS 5.1 PS4 release:

- Disk and Floppy Drives Test (`disktest`), described in Chapter 10.
- Sun Fire™ V880 FC-AL Disk Backplane Test (`dpmtest`), described in Chapter 11.
- Ethernet Loopback Test (`netlbtest`), described in Chapter 12.
- Multiprocessor Test (`mptest`), described in Chapter 13.

The following tests were introduced or enhanced in previous SunVTS 5.1 Patch Set releases:

- Chip Multi-Threading Test (`cmttest`), described in Chapter 14.
- Level 2 Cache Test (`l2sramtest`), described in Chapter 15.
- Alarm Card 2 Test (`alarm2test`), described in Chapter 16.
- Sun™ XVR-100 Graphics Accelerator Test (`pfbtest`), described in Chapter 17.
- Sun™ XVR-1200 Graphics Accelerator Test (`jfbtest`), described in Chapter 18.
- Sun™ XVR-4000 Graphics Accelerator Test (`zulutest`), described in Chapter 19.
- Blade Support Chip Test (`bsctest`), described in Chapter 20.
- Environmental Test (`env6test`), described in Chapter 21.
- I2C Inter-Integrated Circuit Test (`i2c2test`), described in Chapter 22.
- Physical Memory Test (`pmemtest`), described in Chapter 23.
- Virtual Memory Test (`vmemtest`), described in Chapter 24.
- Floating Point Unit Test (`fputest`), described in Chapter 25.
- System Test (`systest`), described in Chapter 26.
- Integer Unit Test (`iutest`), described in Chapter 27.

The System Service Processor test (`ssptest`) was previously titled the Remote System Control test (`rsctest`) in SunVTS 5.1. The reason for this change is that this test now supports Advanced Lights-Out Management hardware in addition to both Remote System Control 1.0 and 2.0 hardware.

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 User's Guide* for installation information.

Test Requirements

SunVTS Version 5.1 was first introduced and designed to run in the Solaris 8 2/02 and Solaris 9 operating environments. It is recommended that you run SunVTS 5.1 in either of these operating environments.

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.

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 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 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.

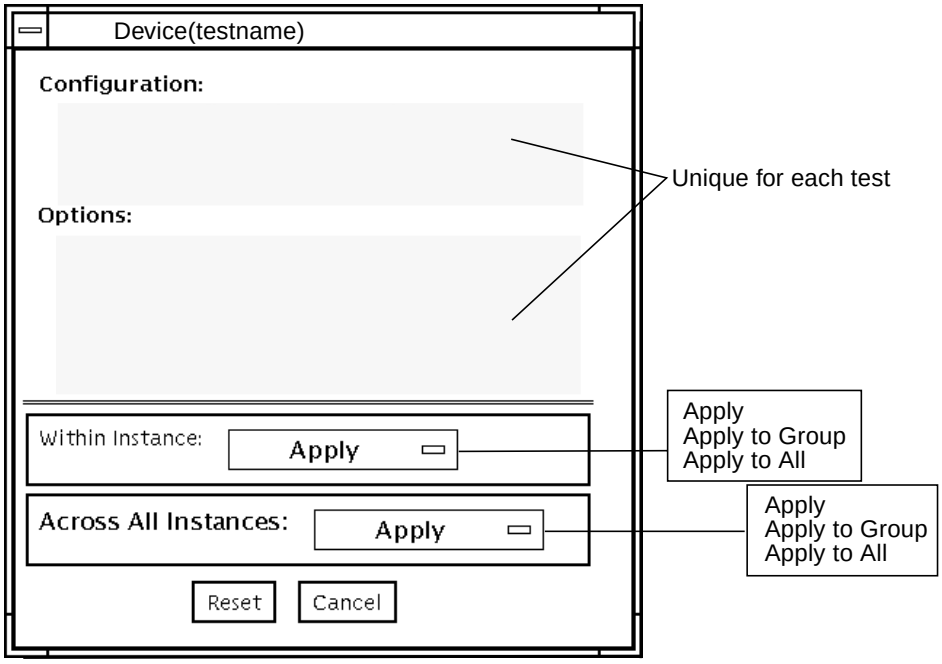

FIGURE 1-1 Test Parameter Options Dialog Box (CDE)

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 5.1 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).

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=cgthree0,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 Common Desktop Environment: Users's Guide* in the Solaris 9 User Collection. This document is available at: `docs.sun.com`.

Caution – If you are using the CDE 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). 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

New SunVTS Features

This chapter describes new features and user interface enhancements that are developed in Patch Set releases of SunVTS software.

Installing SunVTS Software

There are two ways to install SunVTS software from the Solaris Software Supplement CD:

- Solaris Web Start 2.0
- `pkgadd`

Before Installing SunVTS Software

If you installed SunVTS software from a previous release, you should remove all packages associated with that release before installing the new SunVTS software. Remove the old SunVTS packages with the same method you installed the packages.

If you installed SunVTS using Web Start 2.0, you must also remove the software using Web Start 2.0. If you installed SunVTS with the pkgadd command, you must also remove the software with the pkgrm command.

Installing SunVTS Software Using Solaris Web Start 2.0

You can use Solaris Web Start 2.0 to install SunVTS software after you have installed the Solaris operating environment.

▼ To Install SunVTS Software Using Solaris Web Start 2.0

1. **Insert the Solaris Software Supplement CD into your CD ROM drive.**

   If your system is running Sun Enterprise Volume Manager™, it should automatically mount the CD-ROM to the /cdrom/cdrom0 directory.

   If your system is not running Sun Enterprise Volume Manager, mount the CD-ROM as follows:

```
# mount -F hsfs -o ro /dev/dsk/c0t6d0s2 /cdrom
```

2. **In a terminal window, type:**

```
# cd /cdrom/cdrom0
# ./installer
```

3. **When the Solaris Web Start GUI is displayed, select Next.**
4. **Select Default Install or Custom Install from the Solaris Web Start GUI.**
   a. **If you want to install all of the default installed software, select Default Install, then select Next. SunVTS software is installed by default if you select Default Install. See the Solaris Sun Hardware Platform Guide for a list of what software is installed by default.**
   b. **If you only want to install some of the software products, select Custom Install, then select Next. Select the software you want to install, then select Next.**

To Remove SunVTS Software Using Solaris Web Start 2.0

Use the Product Registry utility (`/usr/bin/prodreg`) and select the product to be removed.

Installing and Removing SunVTS Software Using the `pkgadd` and `pkgrm` Commands

Refer to the *SunVTS 5.1 User's Guide* for details on using the `pkgadd` and `pkgrm` commands to install and remove SunVTS software.

Parallel Exclusive Mode

In SunVTS 5.1 Patch Set 3 and later releases, a certain number of instances of `fputest` may run in Exclusive mode in parallel. Note that no two disparate tests may run in Exclusive mode at the same time. However, the same test can run in parallel on different devices in Exclusive mode. This test based parallelism is currently available for only the `fputest`. The concurrency of `fputest` in Exclusive mode is dynamically set with the optimal value based on system resources and is not user configurable. For all other tests, it is not executed in parallel even if the device has multiple instances; instead, they are executed in sequence.

Schedule Manager

This section describes how to use the Schedule Manager, which is a new feature introduced in SunVTS 5.1 Patch Set 2 (PS2). The Schedule Manager is designed to be used with the SunVTS graphical user interface from the Common Desktop Environment (CDE). For details on how to start the SunVTS graphical interface, refer to the *SunVTS 5.1 User's Guide*.

The Schedule Manager allows you to create schedules to perform tests at a specific time, periodically, at intervals, or on kernel idle. You can configure schedules to perform tests with standard options or an option file. If errors occur when running a schedule, the Schedule Manager indicates the cause.

The Schedule Manager allows you to configure a schedule to switch from one test mode to another at a specific time. Additionally, you can specify the duration to run the schedule or specify unlimited; the default is one hour. The schedule will run until all tests are run as per configuration or the specified time, whichever is less.

Schedules can be performed in the standard SunVTS kernel state. If the SunVTS kernel is not running, the Schedule Manager can invoke a SunVTS kernel session at a scheduled time. With the Schedule Manager you can also force a start of a schedule and stop the currently running schedule at a specified time.

Configuring Schedules

The Schedule Manager allows you to create, edit, and delete schedules. Creating a new schedule requires selecting standard options or an option file. You must save schedules with a unique name. `None` and `Untitled` cannot be used as schedule

names. If you try to save a new schedule or modify an existing one with a name already in use, the Schedule Manager will prompt you before overwriting. You can also view the details of all of the schedules in the SunVTS graphical interface.

Configuring Schedules With the SunVTS CDE User Interface

From the main SunVTS Diagnostic window, select *Scheduler→Schedule Manager→Create Schedule* to bring up the Schedule Manager dialog box for creating, editing, or deleting schedules. Only one schedule can be run at a time. You can remove the currently running schedule and all schedules in the scheduler queue by selecting *Scheduler→Schedule Manager→Clean All* from the main window. The main SunVTS Diagnostic window indicates the currently running schedule.

To reach the Schedule Manager dialog box below, select *Scheduler→Schedule Manager→Create Schedule* from the main SunVTS Diagnostic window. A list of the existing schedules and the standard options are displayed.

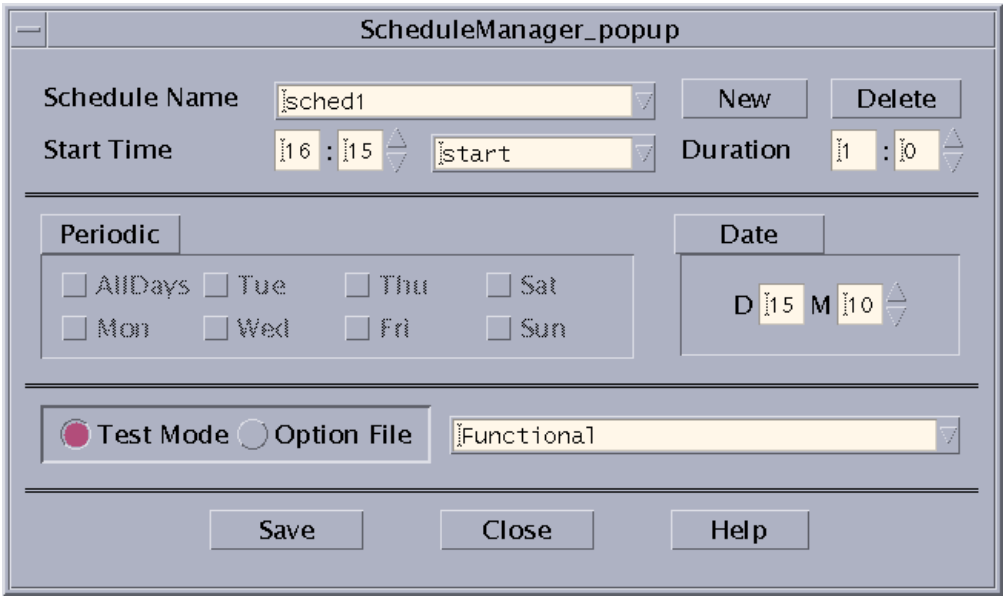

FIGURE 2-1 Schedule Manager Dialog Box

The following table describes the options listed in the Schedule Manager dialog box:

TABLE 2-1 Schedule Manager Options

CDE Inerface Options	Description
Schedule Name	Allows you to enter the name for which the current schedule will be saved. Also allows you to select and bring up existing schedules for editing from the drop-down menu.
New	Creates a new schedule with the default options.
Delete	Deletes the selected schedule that is registered with the kernel.
Start Time	Allows you to specify the time for a schedule to start.
Duration	Allows you to specify the duration to run the schedule. The default is one hour.
Periodic	Allows you to run the schedule periodically and select what days of the week you want to run the schedule. You can also select *All Days*.
Date	Allows you to specify a date that you want the schedule to start.
Test Mode or Option File	Selecting Test Mode allows you to specify which SunVTS test mode you want to use in the drop-down menu. Selecting Option File allows you to select an option file from the drop-down menu.
Save	Saves and registers the schedule to the kernel and `crontab`.
Clean All	This option can be selected from the main SunVTS Diagnostic window. This option removes the currently running schedule and all schedules in the scheduler queue.

To bring up an existing schedule for viewing or editing, select a schedule in the *Schedule Name* drop-down menu, or enter the name of a schedule in the field.

CHAPTER 3

JNI 2GB FC HBA Test (`jnifctest`)

The `jnifctest` tests the functionality of the JNI FC HBA. There are three tests: a self-test, an internal loopback test, and an external loopback test. The self-test tests the internal consistency of the board's internal computer. The loopback tests send out blocks of data to the HBA, receives blocks echoed back to the HBA, and compares the echoed packets to the original. If `jnifctest` detects problems in the self-test, problems sending or receiving the data, or any changes in the content of the data in the loopback tests, it sends out a descriptive error message to the SunVTS console and error log.

There are a small list of patterns that are most likely to detect problems on a FC network; these "critical" patterns are the default. There is also a longer list of patterns and a means for a user to input his own data pattern for testing.

Internal loopback tests require having a loopback plug or cable connected to the port. External loopback tests can be run on a port connected to storage, to a switch, or with a loopback plug or cable. The simplest way to get the greatest test coverage is to have all ports connected with a cable and run both the self-test and the external loopback test. These two tests are enabled by default.

The `jnifctest` runs in exclusive mode. Any storage behind a particular port will be inaccessable while the tests are running. Also, system console log messages reporting renegotiation of the link status may be generated for ports connected to a switch or storage when `jnifctest` is run.

`jnifctest` has three subtests available:

- Online selftest
- Internal loopback test
- External loopback test

jnifctest 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.

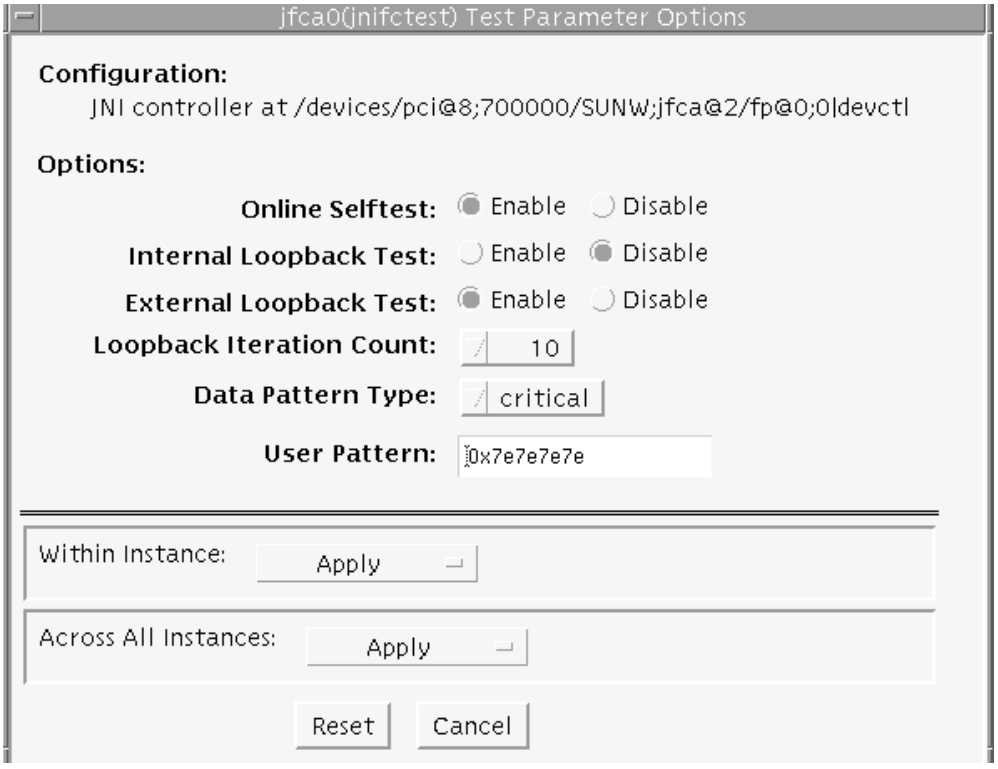

FIGURE 3-1 `qlctest` Test Parameter Options Dialog Box

TABLE 3-1 `jnifctest` Options

`jnifctest` **Options**	**Description**
Online Selftest	Enable or Disable the board self-test.
Internal Loopback Test	Enable or Disable the internal loopback test.
External Loopback Test	Enable or Disable the external loopback test.

TABLE 3-1 `jnifctest` Options *(Continued)*

`jnifctest` **Options**	**Description**
Loopback Iteration	Number of iterations to run the loopback test.
Data Pattern Type:	What type of data to send in the loopback tests. *Critical* patterns are a dozen patterns which are optimized to cause errors in marginal FC networks. With a small number of patterns, the test will run more rapidly. *All* patterns is a list of approximately 256 patterns, including the *Critical* patterns. The *User* pattern is a single pattern that you can specify to test with.
User Pattern	A 32-bit data pattern which is used if 'user' is specified in the Data Pattern Type option. The *User Pattern* should be input in the form 0x12345678

The default values are for the Online Selftest and External Loopback test to be enabled. The Internal Loopback test is disabled. The Loopback Iteration count defaults to 10 iterations. The default Data Pattern Type is *Critical* and the default *User Pattern* is 0x7e7e7e7e.

jnifctest Supported Test Modes

TABLE 3-2 jnifctest Supported Test Modes

Test Mode	Description
Exclusive	Runs full set of tests.

jnifctest Command-Line Syntax

/opt/SUNWvtshm/bin/jnifctest -vf -o dev=*jfca0*, **selftest=**{enable|disable}, **ilb=**{enable|disable}, **elb=**{enable|disable}, **iterations=**{1 – 1000000}, **selectpattern=**{critical|all|user}, **userpattern=**{*hex-value*}

TABLE 3-3 jnifctest Command-Line Syntax

Argument	Description
dev=device	Specifies device to be tested—for example, jfca0, jfca1, and so on.
selftest=*Enable* | *Disable*	Enables or disable the self-test.
ilb=*Enable* | *Disable*	Enables or disables the Internal loopback test.
elb=*Enable* | *Disable*	Enables or disables the External loopback test.
iterations= *1 – 1000000*	Specifies the number of iterations of the tests. The possible range of this parameter is 1 – 1,000,000. The most practical range is 10 – 5000.
selectpattern=*user* | *critical* | *all*	Specifies which data patterns are used for the loopback tests: the small list of *critical* hex-value data patterns, or the larger list of *all* hex-value data patterns. The *critical* hex-value data pattern list is 12 hex-value patterns. The *all* hex-value data pattern is a significantly larger list of hex-value data patterns.
userpattern=*hex-value*	If the selectpattern option is specified as *user*, this option specifies the data pattern that should be used for the loopback tests. The *hex-value* pattern is specified with 8 hex digits—for example, 0x12345678, 0x7e7e7e7e, or 0xcafebaba.

CHAPTER 4

Level 1 Data Cache Test (`l1dcachetest`)

`l1dcachetest` exercises the level1 Data cache in the CPU module of Sun systems. The test writes, reads, and verifies access of multiple virtual addresses. The virtual addresses are so chosen that they cause targeted hits and misses in the cache. The test dynamically determines the size and organization of the cache and tunes the test accordingly to be effective on the l1dcache.

The P-cache subtest of `l1dcachetest` provides diagnostic coverage for Prefetch cache present in UltraSPARC III+ and UltraSPARC IV processors. The Prefetch cache is a small (2 Kbytes) cache that is accessed in parallel with Data cache for floating-point loads. This subtest tests the Data SRAM of the prefetch cache, and uses stress testing by applying March SS and March SAM algorithms on the data SRAM. This subtest is performed in Exclusive test mode only.

`l1dcachetest` 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.

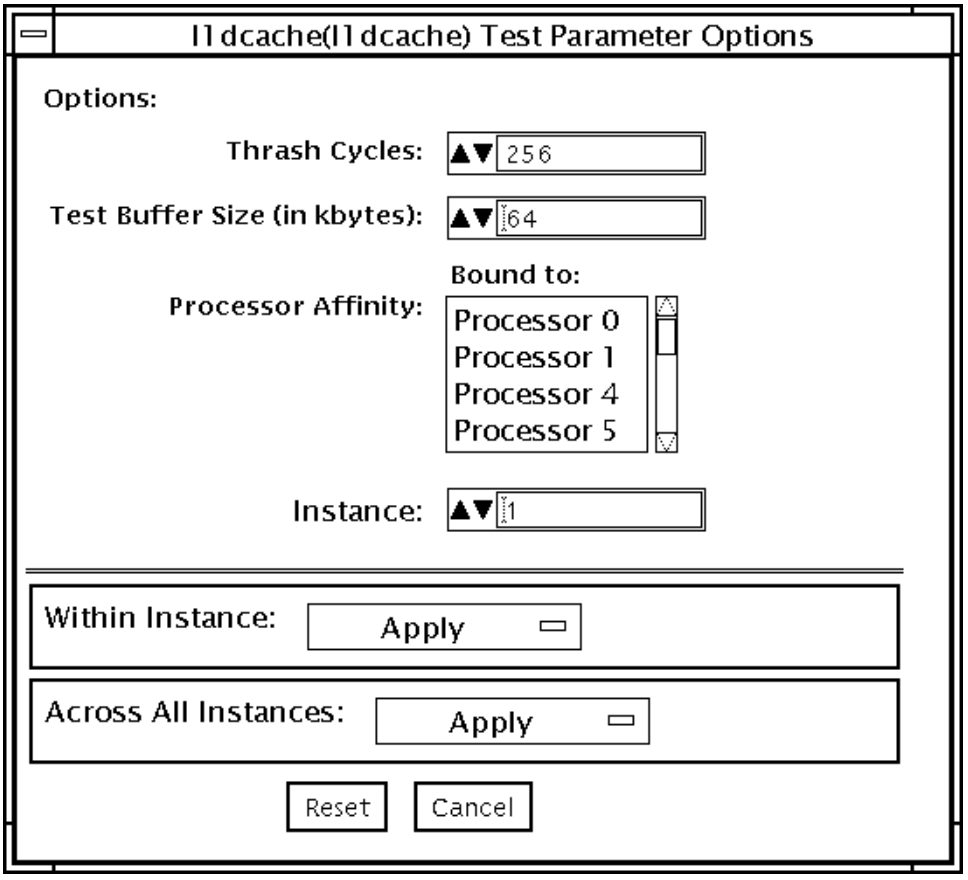

FIGURE 4-1 `l1dcachetest` Test Parameter Options Dialog Box

TABLE 4-1 `l1dcachetest` Options

Option	Description
Thrash Cycles	Specifies the number of thrashing cycles the test completes for the level1 cache on the system. Default value for Exclusive mode is 256.
Test Buffer Size	Sets the size of the buffer, in KBs, that the test allocates for testing. Default value is 64.

Note – The Test Buffer Size option will not be supported in a future release of SunVTS. The reason for this is that the test will dynamically determine the size of the cache and set the buffer size appropriately.

Note – The `l1dcachetest` is automatically bound to a processor. Users are advised to not use the Processor Affinity option for the `l1dcachetest`.

`l1dcachetest` Test Modes

TABLE 4-2 `l1dcachetest` Supported Test Modes

Test Mode	Description
Connection	Performs the Connection subtest.
Exclusive	Performs only the `l1dcachetest` (full test).

`l1dcachetest` Command-Line Syntax

`/opt/SUNWvts/bin/l1dcachetest` *standard_arguments* **`-o [`**
`[ dev=cpu-unit`*N*** `]`**
`[ count`=*number* **`]`**
`[ buffer`=*number* **`] ]`**

Note – The `l1dcachetest` is now per CPU, and *N* is the CPU number (0,1,2, etc.). Therefore, if the system has five CPUs (CPUs 1, 2, 5, 6, and 7), you can perform `l1dcachetest` on each CPU individually by specifying which CPU you want to verify when invoking the test. For example, if you want to perform `l1dcachetest` on CPU 7, you would enter the following:
`/opt/SUNWvts/bin/l1dcachetest` *-generic_options* **`-o dev=cpu-unit7`**

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 "32-Bit and 64-Bit Tests section of the *SunVTS Test Reference Manual*.

TABLE 4-3 `l1dcachetest` Command-Line Syntax

Argument	Description
`dev=cpu-unit`*N*	Specifies the name of the device.
`count=`*number*	Specifies the number of thrashing cycles the test completes for the level1 cache on the system. Default value for Offline mode is 256.
`buffer=`*number*	Sets the size of the buffer, in KBs, that the test allocates for testing. Default value is 64.

CHAPTER 5

Cache Consistency Test (`mpconstest`)

`mpconstest` verifies that cache coherency is maintained in a multi-processor environment by creating contention for one or more cache lines.

Note – The `mpconstest` subtests have been migrated to `mptest`. It is expected that mpconstest will be EOL in the future. You are advised to shift from `mpconstest` to `mptest`.

Only the following CPUs are supported:

- SuperSPARC [TI] (SS10/SS20/SS1000/SC2000)
- SuperSPARC II
- MicroSPARC II [TI] (50 MHz)
- MicroSPARC II (SS5)
- UltraSPARC I [TI] (143/167/200MHz)
- UltraSPARC II (250/333/336/360MHz)
- UltraSPARC III (500-600/750/900MHz)
- UltraSPARC III Cu
- UltraSPARC IIIi
- UltraSPARC IV

This test has several subtests, each designed to create a different kind of contention for cache lines. Each subtest uses different methods to test the shared memory buffer, the stride size, and any intermediate stores or loads.

When `mpconstest` starts, it creates a shared memory buffer. It then determines the number of CPUs on the system. For each CPU, the test takes the following steps:

1. Forks a thread and binds it to the CPU.
2. Runs the selected subtest in the thread.

3. Assigns each CPU an ID number from 1 to *n*. The CPU assigned `ID 1` is considered the master.

The above steps are repeated for each subtest. Only one subtest can be selected at a time.

This test is not scalable.

`mpconstest` Test Requirements

This test requires that the tested system has at least two CPUs. Otherwise, the test will not appear as an option.

`mpconstest` only runs on machines that support the v8plus standard of SPARC CPU hardware architecture. If the v8plus instructions are not supported, `mpconstest` will not appear on the Test Selection GUI. To determine whether a machine supports the v8plus standard, go to a command prompt on that machine and type:

```
% isalist
sparcv9+vis sparcv9 sparcv8plus+vis sparcv8plus sparcv8
```

Note – This set of tests is very sensitive to activity on the machine and must be run exclusive of all other tests.

`mpconstest` Subtests

TABLE 5-1 `mpconstest` subtests

Tests	Description
cons1	Each CPU writes to successive locations with a stride size of byte, half word, or full word. This subtest creates contention for a single cache line. No other loads or stores are performed between successive writes to shared memory.
cons2	Each CPU reads from a location that is *cachesize* bytes away from the last written location. Every read causes the previously written line to be written back. The test runs until the CPU has accessed all lines in the cache.
cons3	Similar to cons1 except that only one double word of each line is accessed. This creates simultaneous contention for multiple cache lines rather than a single line.
cons4	Similar to cons2, except that each CPU performs one store byte (`storeb`) and one load byte (`loadb`) operation between the detection of ID and the write of the next CPU ID. The target of the `storeb` and `loadb` is a unique byte in the line the CPU just read. This target is recognized as a different double word in the shared line *cachesize* bytes.
cons5	Similar to cons3 except that each CPU performs one `storeb` and one `loadb` operation between the detection of ID and the write of the next CPU ID. The target of the `storeb` is one unique byte of the next double word in the line that the CPU just read from the CPU ID. The `storeb` data is unique to each CPU and changes each time the address of the target line changes.
cons6	Similar to cons1 except that only one double word of each line is accessed. This creates simultaneous contention for multiple cache lines rather than a single line.
cons7	Similar to cons3 except that each CPU performs two `storeb` and one `loadh` operations between the detection of the CPU ID and the write of the next CPU ID. The targets of the `storebs` and `loadh` are two consecutive bytes of a double word in a shared line which is not a part of the shared memory buffer containing the IDs. The address of the target `storeb` and `loadh` instructions is held constant. The first `storeb` instruction gains ownership of the cache line, and the second `stroreb` is performed as a write hit. This occurs at the same time other CPUs are reading and writing the shared line containing the IDs.

TABLE 5-1 `mpconstest` subtests

Tests	Description
cons8	Similar to cons3 except that each CPU performs one `storeb` and one `loadb` operation between the detection of the CPU ID and the write of the next CPU ID. The target of the `storeb` and `loadb` is one unique byte of a double word of a private (unshared) line whose line number is identical to the line number containing the IDs. The `storeb` data is unique to each CPU and changes each time the address of the line containing the IDs changes.
cons9	Similar to cons8 except that the target of the `storeb` and `loadb` is one unique byte of a double word of a private line whose address does not change through the entire test.
cons10	Similar to cons9 except that two `storeb` and two `loadb` operations are performed to private (unshared) lines.The target of the second `storeb` is *cachesize* bytes away from the target of the first `storeb`. In a direct map cache, this results in a writeback of the unshared data written with the first `storeb`. The `loadb` operations are performed after the `storeb` in order to ensure that the writeback occurs correctly.
cons11	Similar to cons10 except that the target of the `storeb` and `loadb` operations is to a shared line rather than a private line.
cons12	Similar to cons7 except that two store double (`stored`) and load double (`loadd`) operations are used in place of the `storeb` and `loadb` operations. The target of the `stored` and `loadd` operations are two consecutive double words of a shared line. This test is designed to verify that the double word operations are performed correctly while the shared and owned state of the line containing the ID is changing.
cons13 through cons17	These tests are similar variations of intermediate operations, stride size etc, and do not involve any new interfaces.

`mpconstest` 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.

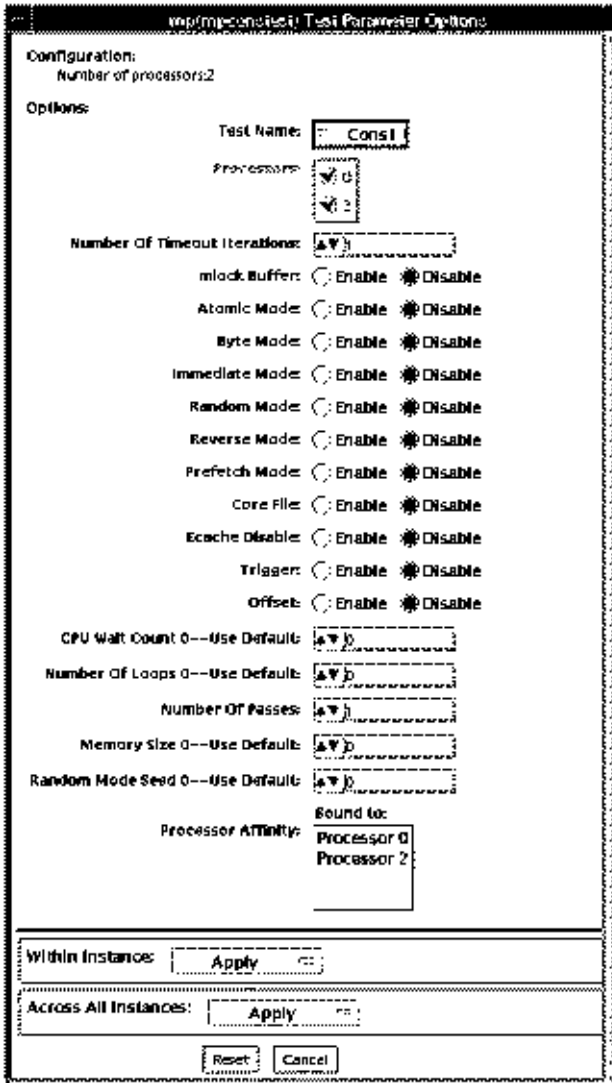

FIGURE 5-1 `mpconstest` Test Parameter Options Dialog Box

TABLE 5-2 `mpconstest` Options

Option	Description
Test Name	Selects the subtest to be run.
Number of Timeout Iterations	Sets the number of times the test is allowed to time out. Default is 1. Note that each timeout occurs after a greater amount of elapsed time than the previous one. That is, if the first timeout occurs after `T` units of time, the second occurs `2T` after `T`, and the third occurs `3T` after `2T`.
Lock Buffer	Locks Buffer in Memory. Default is not locked. Locking the buffer in memory will disable COMA (Cache Only Memory Architecture).
Atomic Mode	Uses the atomic instruction swap. Default is disabled.
Byte Mode	Uses byte instructions to load and store. Default is disabled.
Immediate Mode	Supports all subtests except cons1, cons2, cons3, cons15, cons16, and cons17.
Random Mode	Enables Random Mode.
Reverse Mode	Traverses the shared memory buffer in reverse. Default is disabled.
Prefetch Mode	Sets prefetch for read and write. Default is disabled.
CoreFile	Generates a core file. Exits in case of unexpected signals. Default is disabled.
Ecache Disable	Disables the external cache. Default is enabled.
Trigger	Sends an interrupt signal to all processors when one processor detects a failure. Default is disabled.
Offset	Specifies an offset of line size between successive writes. Default is disabled.
CPU Wait Count	Forces CPU 1 to write first if the number of CPUs is less than *cpucount*. Default is disabled. This option is not supported with subtests cons15, cons16, and cons17.
Number of Loops	Selects the number of test loops. Default is 5.
Number of Passes	Selects the number of passes. Increasing the number of passes increases system stress. Setting the number of passes to 0 will cause the test to run in an endless loop. Passes can only be set to 0 in command line mode, not from the GUI. Default is 1.
Memory Size	Selects the memory size, in Megabytes, for the shared buffer. Default is 128.
Random Mode Seed	Sets random number seed to a user specified value. Selects a random number seed by default.

mpconstest Test Modes

TABLE 5-3 mpconstest Supported Test Modes

Test Mode	Description
Functional (Offline)	Runs the full test.

mpconstest Command-Line Syntax

/opt/SUNWvts/bin/mpconstest *standard_arguments*
-o tst=*Cons1* | *Cons2*,**itm=***number*,**lck,a,b,c,e,h,loops=***number*,**memsize=***memsize*, **wait=***cpucount*,**passes=***passes*,**r,t,x,y,i,q,seed**=*number*

TABLE 5-4 mpconstest Command-Line Syntax

Argument	Description
tst= *Cons1* | *Cons2* | *Cons 3* | *Cons4* | *Cons5* | *Cons6* | *Cons7* | *Cons8* | *Cons9* | *Cons10* | *Cons11* | *Cons12* | *Cons12* | *Cons13* | *Cons14* | *Cons15* | *Cons16* | *Cons17*	Range of choices available between cons1 through cons17 subtests.
itm=*number*	Sets the number of times the test is allowed to time out. Default is 1. Note that each timeout occurs after a greater amount of elapsed time than the previous one. That is, if the first timeout occurs after T units of time, the second occurs 2T after T, and the third occurs 3T after 2T.
lck	Locks Buffer in Memory. Default is not locked. Locking the buffer in memory will disable COMA (Cache Only Memory Architecture).
a	Enables atomic mode. Uses the atomic instruction swap
b	Enables byte mode. Uses byte instructions to load and store.
c	Generates a core file. Exits in case of unexpected signals.
e	Disables the external cache.

TABLE 5-4 `mpconstest` Command-Line Syntax

Argument	Description
`h`	Prints usage message.
`loops=`*number*	Sets the number of loops for the iterations. Default is 5.
`memsize=`*memsize*	Selects the memory size, in Megabytes, for the shared buffer. Default is 128.
`wait=`*cpucount*	Forces CPU 1 to write first if the number of CPUs is less than *cpucount*.
`passes=`*passes*	Selects the number of passes. Increasing the number of passes increases system stress. Setting the number of passes to 0 will cause the test to run in an endless loop. Passes can only be set to 0 in command line mode, not from the GUI. Default is 1.
`r`	Enables Reverse mode. Traverses the shared memory buffer in reverse.
`t`	Enables Trigger. Sends an interrupt signal to all processors when one processor detects a failure.
`x`	Enables Prefetch. Sets prefetch for read and write.
`y`	Enables Offset. Specifies an offset of line size between successive writes.
`i`	Enables Immidiate Mode. Not suppored for subtests cons1, cons2, cons3, cons15, and cons 17.
`q`	Enables Random Mode.
`seed`	Sets a random number seed to the user specified value.

CHAPTER 6

SunPCi II Test (`sunpci2test`)

The `sunpci2test` tests the SunPCi™ II card, which is a PC processor embedded in an add-on card. This test consists of approximately 150 POST routines that perform diagnostic, hardware detection, and initialization functions. This test issues a reset, then launches POST in the SunPCi II BIOS to check the devices. Finally, the sunpci2test runs bridge and system diagnostics tests.

SunPCi-2 and SunPCi-3 cards are tested by the `sunpci2test` diagnostic. If the card under test is SunPCi-2 then the device name will be `sunpci2drvX`. If it is SunPCi-3 then the device name will be `sunpci3drvX`.

`sunpci2test` Test Requirements

Before running the test, the X-window for Microsoft Windows must be shut down. If this is not done, the test will not launch.

▼ To Shut Down Microsoft Windows and the SunPCi II Card:

1. **Click Start button in Microsoft Windows.**
2. **Click Shut Down.**

   The shutdown window appears. Wait for the "It is now safe to shut off your PC" message.
3. **Select "File" from the SunPCi window.**
4. **Select "Exit" from the file menu.**

5. **Click OK.**

`sunpci2test` 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.

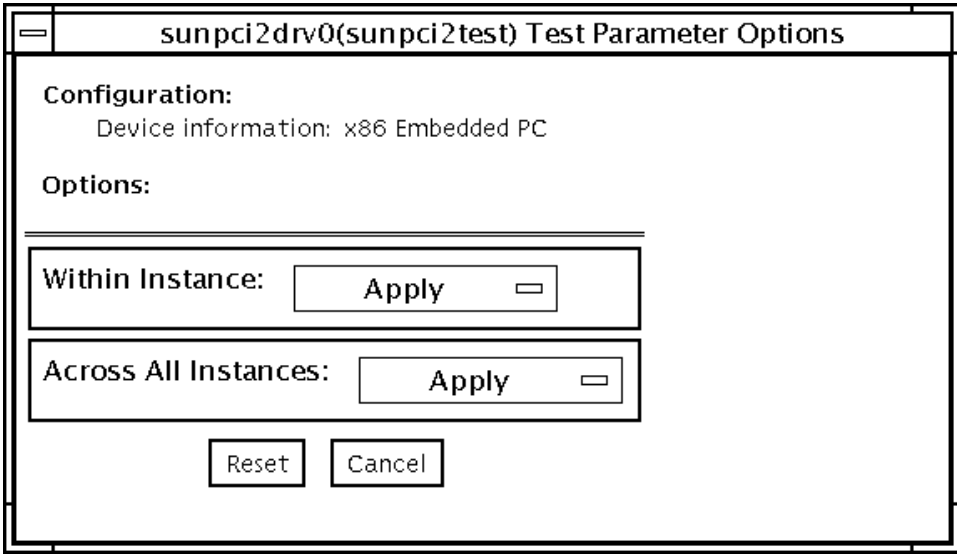

FIGURE 6-1 `sunpci2test` Test Parameter Options Dialog Box

`sunpci2test` only runs with the default parameters in place. Thus, this test does not allow any options to be configured specifically for an individual system. The number of instances is preset to 1 (the default value), as only one local copy of the test is supported.

sunpci2test Test Modes

TABLE 6-1 sunpci2test Supported Test Modes

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

sunpci2test Command-Line Syntax

/opt/SUNWvts/bin/sunpci2test *standard_arguments*

Note – There are no test-specific options for sunpci2test.

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 5.1 Test Reference Manual*.

CHAPTER 7

System Service Processor Test (ssptest)

The ssptest exercises the Remote System Control (RSC) feature, which is integrated on the Sun Enterprise 250 as well as the next-generation RSC 2.0 plug-in card introduced with the Sun Fire 280R line, and Advanced Lights-Out Management (ALOM) feature. The ALOM feature is integrated into the Sun Fire V210, Sun Fire V240, and Sun Fire V440 systems.

The RSC or ALOM provides secure remote access for system monitoring, firmware updates, and failure recovery. The RSC or ALOM communicates with the host through two internal serial lines, the I2C bus, and reset lines.

The RSC 1.0 hardware consists of the controller, flash, SEEPROM, 10MB Ethernet port, and an external console serial port.

The RSC 2.0 plug-in card hardware consists of the controller, flash, SEEPROM, 10MB Ethernet port, FRUSEEPROM, Time of Day (ToD) device, internal PCMCIA modem card, and battery backup.

The ALOM hardware consists of a Motorola MPC850 processor, flash, SEEPROM, 10MB/100MB Ethernet Port, FRUSEEPROM, Time of Day (ToD) device, Serial Transceiver, and battery backup.

ssptest is not scalable.

ssptest Subtests

The ssptest will present different subtests and options based on which type of hardware (RSC or ALOM) and which version of RSC hardware (1.0 or 2.0) it is testing.

The subtests common to RSC 1.0, RSC 2.0, and ALOM include:

TABLE 7-1 Subtests for Both RSC 1.0, RSC 2.0, and ALOM

Subtest	Description
Ethernet	Allows for internal loopback testing, on the Ethernet device with user specified data, size, and number of packets.
	Allows for external loopback testing with user-specified data, size, and number of packets. This requires a connection to a 10MB hub or switch for RSC 1.0, or a passive loopback connector for RSC 2.0, and ALOM.
	Allows for a ping to be sent to a specified host and checks the response.
Flash CRC	Performs a checksum test on the flash device.
SEEPROM CRC	Performs a checksum test on the SEEPROM device.
Serial	Allows internal loopback testing with user-specified data and size on the two internal serial ports.
	Allows for internal and/or external testing on the external `ttyu` port. The external test requires a passive loopback connector.

`ssptest` also presents the following subtests when running on the RSC 2.0 hardware:

TABLE 7-2 Subtests for RSC 2.0 Only

Subtest	Description
FRU SEEPROM CRC	Performs a checksum test on the SEEPROM device.
I2C	Tests the i2c bus connection between the host and the RSC.
ToD	Performs multiple reads to the ToD device and verifies that the time is incrementing.
Modem	Verifies that the modem is installed. Displays the manufacture information, in Verbose mode. Performs AT inquiry commands.

`ssptest` presents the following subtests when running on the ALOM hardware:

TABLE 7-3 Subtests for ALOM Only

Subtest	Description
I2C	Tests the i2c bus connection between the host and the ALOM.
ToD	Performs multiple reads to the ToD device and verifies that the time is incrementing.
FRU SEEPROM CRC	Performs a checksum test on the SEEPROM device. Note: The FRU SEEPROM CRC subtest is not enabled in `ssptest` on platforms using ALOM hardware with ALOM firmware. On these platforms, currently, `i2c2test` performs the checksum test on the SEEPROM device in ALOM hardware.

The subtests call test modlets that are written in the native Real Time Operating System (RTOS) that resides in the RSC firmware. The `ssptest` subtests execute the test modlets, pass parameters, and retrieve results from the RSC or ALOM using a test protocol on the host to RSC or ALOM internal serial lines.

`ssptest` 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.

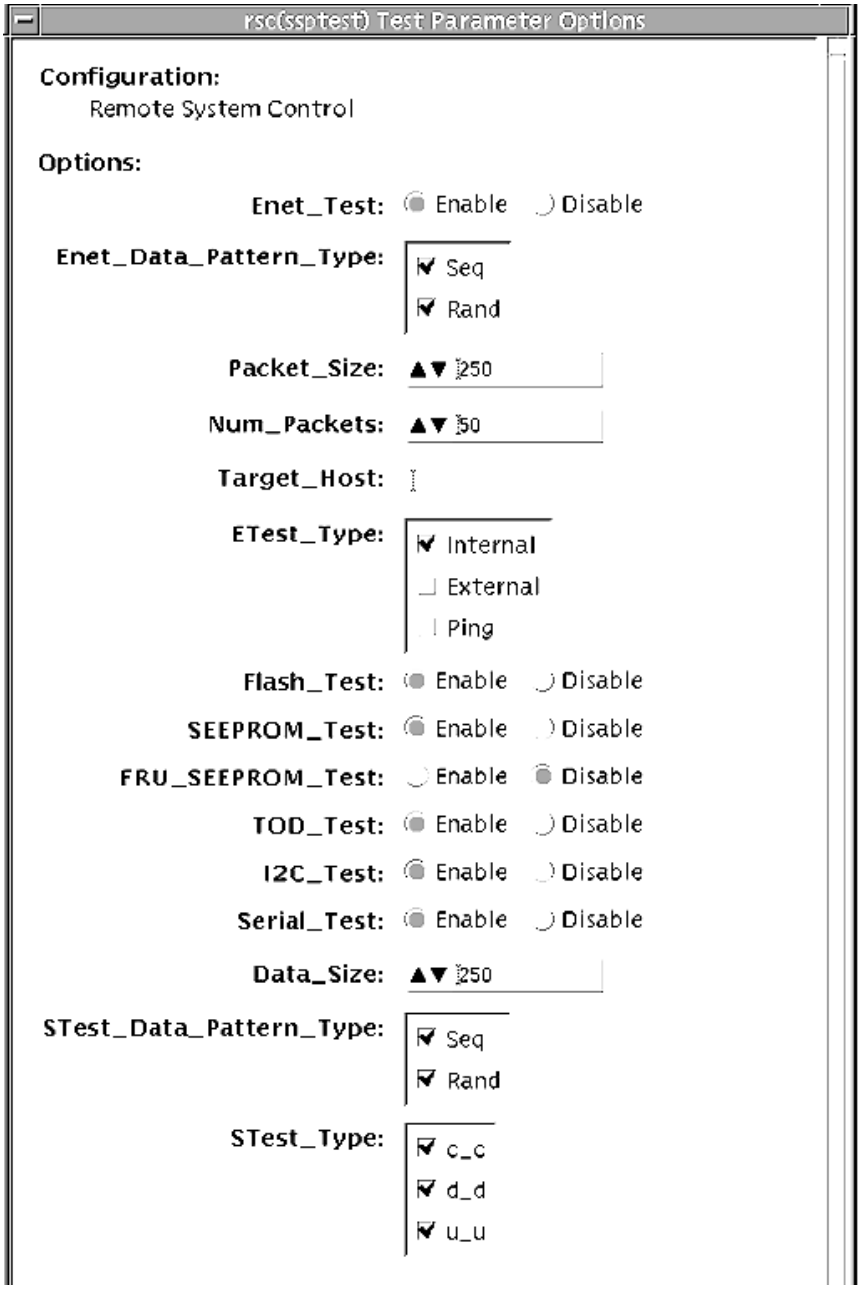

FIGURE 7-1 `ssptest` Test Parameter Options Dialog Box (Top Section)

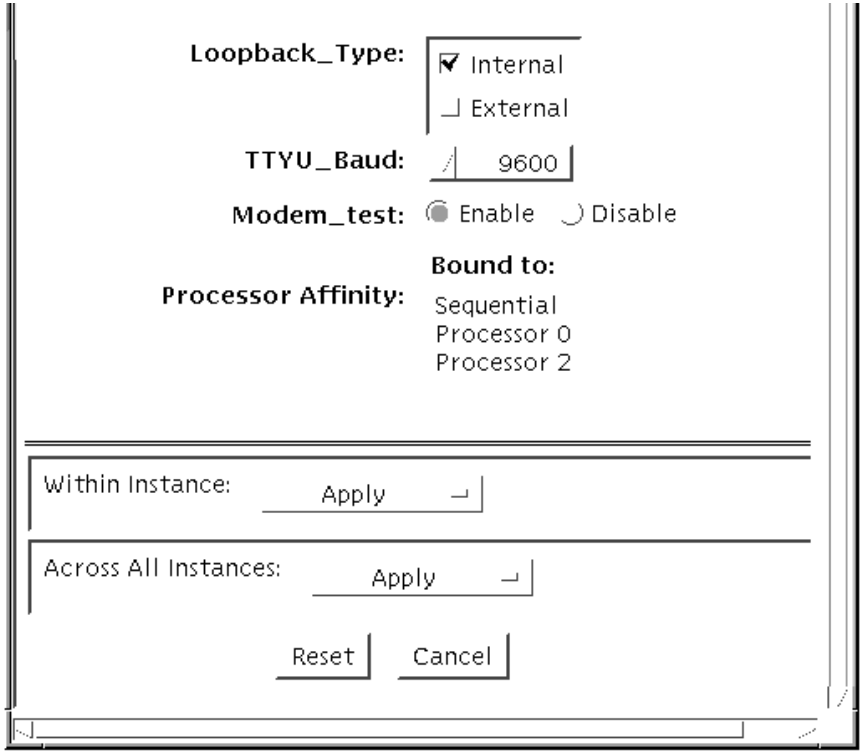

FIGURE 7-2 `ssptest` Test Parameter Options Dialog Box (Bottom Section)

Note – The Configuration field in the `ssptest` Test Parameter Options dialog box displays the which type of hardware (RSC or ALOM) is being tested. For RSC 1.0 and 2.0, *Remote System Control* is displayed. For ALOM, *Advanced Lights-Out Management* is displayed.

TABLE 7-4 `ssptest` Options

`ssptest` **Options**	**Description**
Enet test	Enables or disables RSC or ALOM Ethernet testing.
Data Pattern Type	Selects Sequential, Random, or both types of data patterns.
Packet Size	Defines the size of each data packet to be sent for all tests.
Num Packets	Specifies the number of data packets to send in one test loop.
Target Host	Specifies the IP address of a host to use for the ping test.
Enet Test Type	Selects any or all Internal, External, or ping tests.

TABLE 7-4 `ssptest` Options *(Continued)*

`ssptest` **Options**	**Description**
Flash test	Enables or disables the flash checksum test.
SEEPROM test	Enables or disables the SEEPROM checksum test.
FRU SEEPROM test	Enables or disables the FRU SEEPROM checksum test. (RSC 2.0 and ALOM only).
TOD test	Enables or disables the Time Of Day test.
I2C test	Enables or disables the I2C test (RSC 2.0 and ALOM only).
Serial test	Enables or disables the RSC or ALOM serial test.
Data Size	Defines the data size to be sent.
Loopback Type	Selects Internal, External, or both. External requires a loopback plug.
Data Pattern Type	Selects Sequential, Random, or both types of data patterns.
Serial Test Type	Selects serial ports to be tested, u to u, c to c, or d to d.
TTYU_Baud	Select a fixed baud rate or all baud rates for testing the ttyu port. The valid baud rates under TTYU_Baud are: ALL, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 76800, 115200. The default is 9600.
Modem Test	Used to Enable or Disable the RSC PCMCIA modem test (RSC 2.0 only).

`ssptest` Test Modes

`ssptest` supports the following tests as described in the table below.

TABLE 7-5 `ssptest` Supported Test Modes

Test Mode	**Description**
Connection	Reports the status of the RSC or ALOM.
Exclusive	Tests the RSC's and ALOM's Ethernet, flash, SEEPROM, ToD, I2C, and serial devices. All tests use the internal modes as defaults. The `ssptest` will not run the serial test on `ttyc` if the console has been redirected to the RSC. The `ttu` tests will not run if there is an open login on the `ttyu`.

ssptest Command-Line Syntax

RSC 1.0: **/opt/SUNWvts/bin/ssptest** *standard_arguments* **-o enet=***E/D*,**epattype=***seq+rand*,**esize=***packet_size*,**epkts=***number_packets*,**target=***IP_address*,**etest=***I+E+P*,**flash=***E/D*,**seeprom=***E/D*,**serial=***E/D*,**sdatsize=***data_size*,**slb=***I+E*,**spattype=***seq+rand*,**stest=***u_u+c_c+d_d*,**ttyubaud=***baud_rate*|*all*

RSC 2.0: **/opt/SUNWvts/bin/ssptest** *standard_arguments* **-o enet=***E/D*,**epattype=***seq+rand*,**esize=***packet_size*,**epkts=***number_packets*,**target=***IP_address*,**etest=***I+E+P*,**flash=***E/D*,**seeprom=***E/D*,**fruseeprom=***E/D*,**tod=***E/D*, **i2c=***E/D*,**serial=***E*|*D*,**sdatsize=***data_size*,**slb=***I+E*,**spattype=***seq+rand*, **stest=***u_u+c_c+d_d*,**ttyubaud=***baud_rate*|*all*,**rscmodem=***E/D*

ALOM: **/opt/SUNWvts/bin/ssptest** *standard_arguments* **-o enet=***E/D*,**epattype=***seq+rand*,**esize=***packet_size*,**epkts=***number_packets*,**target=***IP_address*,**etest=***I+E+P*, **flash=***E/D*,**seeprom=***E/D*,**tod=***E/D*,**i2c=***E/D*,**serial=***E/D*,**sdatsize=***data_size*,**slb=***I*, **spattype**=*seq+rand*, **stest=***d_d*

TABLE 7-6 ssptest Command-Line Syntax

Argument	Description
enet=*enable*|*disable*	Enables or disables RSC or ALOM Ethernet test.
epattype=*seq+rand*	Predefined pattern options used for Enet test.
esize=*packet_size*	Data size for each packet in the Enet test.
epkts=*number_packets*	Number of packets to send for Enet test.
target=*IP_address*	IP address of target system for Enet ping test.
etest=*Internal+External+Ping*	Selects any or all Internal, External, or ping tests.
flash=*enable*|*disable*	Enables or disables RSC or ALOM Flash checksum test.
seeprom=*enable*|*disable*	Enables or disables RSC or ALOM SEEPROM checksum test.
fruseeprom=*E/D (RSC 2.0 and ALOM ONLY)*	Enables or disables RSC FRU SEEPROM checksum test.
tod=*E/D (RSC 2.0 and ALOM ONLY)*	Enables or disables RSC or ALOM Time of Day test.
i2c=*E/D (RSC 2.0 and ALOM ONLY)*	Enables or disables RSC or ALOM i2c test.
serial=*enable*|*disable*	Enables or disables RSC or ALOM serial test.
sdatsize=*data_size*	Data size for the `rsc` or `alom` serial tests.
slb=*Internal+External*	Loopback type. External N/A on ports C and D.

TABLE 7-6 `ssptest` Command-Line Syntax *(Continued)*

Argument	Description
`spattype=`*seq+rand*	Predefined pattern options used for RSC or ALOM serial test.
`stest=`*u_u+c_c+d_d*	Defines port and configuration to use for RSC or ALOM serial test.
`ttyu_baud=`*ALL|specific_baud*	Defines baud rates to be used in testing the RSC's console port. The valid baud rates under `ttu_baud` are: ALL, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 76800, 115200. The default is 9600.
`rscmodem=`*E/D (RSC2.0 Only)*	Enables or disables the RSC PCMCIA modem test.

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 8

Sun Netra™ 240 Alarm Card Test (`n240atest`)

`n240atest` is designed to test the alarm card on Sun Netra 240 servers.

Caution – Solaris 8 2/02 operating environment or later is required to perform the `n240atest`.

`n240atest` 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.

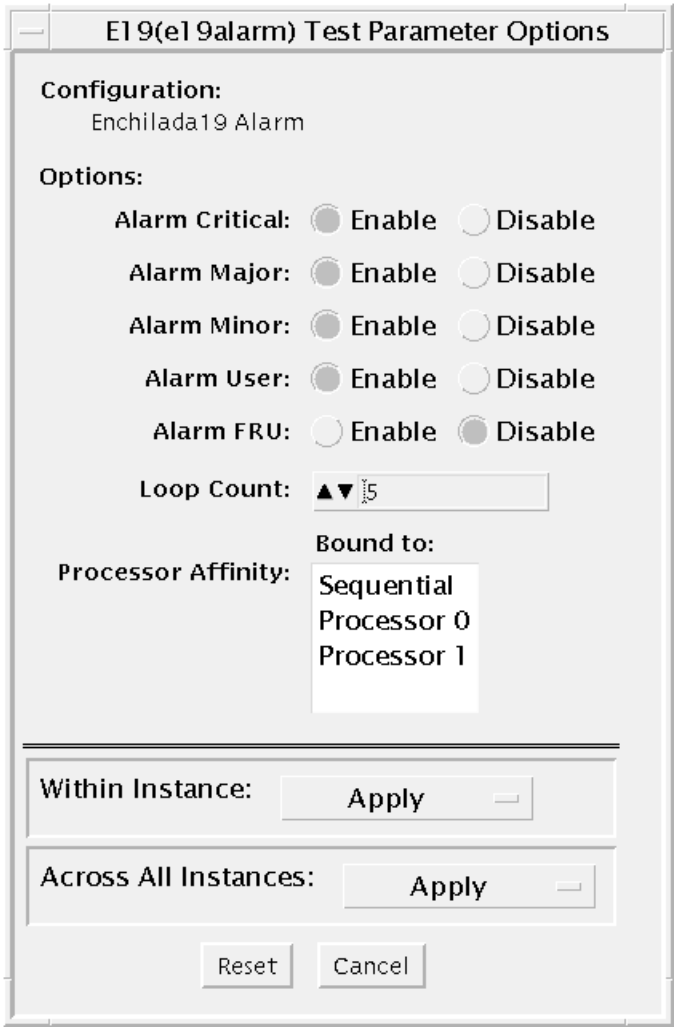

FIGURE 8-1 `n240atest` Test Parameter Options Dialog Box

The following table describes the `n240atest` options:

TABLE 8-1 `n240atest` Options

`n240atest` **Options**	**Description**
Alarm Critical	Toggles the Alarm critical LED.
Alarm Major	Toggle the Alarm critical Major.
Alarm Minor	Toggles the Alarm critical Minor.

TABLE 8-1 `n240atest` Options *(Continued)*

`n240atest` **Options**	**Description**
Alarm User	Toggles the Alarm critical User.
Alarm FRUID	Performs the FruID checksum test on the alarm card.
Loop Count	Sets up the loop count for toggling all four alarm LEDS. The count number is 1 to 10.

`n240atest` Test Modes

TABLE 8-2 `n240atest` Supported Test Modes

Test Mode	**Description**
Connection	The test determines if the devices are connected to the system you are testing and verifies that they are accessible. Device functionality is not verified; however, you can safely run connection mode tests while the system is offline.
Functional	Tests fully exercise all aspects of the device through the associated device drivers. These tests use a significant portion of the system resources and assume that the device is available for testing. For this reason, the system must be offline with no other users or application running. This mode is sometimes referred to as Offline mode.
Exclusive	Enables users to perform `alarmtest` exclusively with no other applications running at the same time.

`n240atest` Command-Line Syntax

`/opt/SUNWvts/bin/sparcv9/n240atest` *standard_arguments* **`[ -o`**

`[ dev=`<*device_name*>**`]`**
`[ cri=`<*E(nable)* | *D(isable)*>**`]`**
`[ maj=`<*E* | *D*>**`]`**
`[ min=`<*E* | *D*>**`]`**

[usr=<*E* | *D*>**]**
[fru=<*E* | *D*>**]**
[count=<*count_number*>**]]**

TABLE 8-3 `n240atest` Command-Line Syntax

Argument	Description
`dev`	Specifies the name of the raw device to test.
`cri`	Toggles the Alarm critical LED.
`maj`	Toggles the Alarm critical Major.
`min`	Toggles the Alarm critical Minor.
`usr`	Toggles the Alarm critical User.
`fru`	Performs FruID checksum test on the alarm card.
`count`	Sets up the loop count for toggling all four alarm LEDS *count_number* is 1 to 10.

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 "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

CHAPTER 9

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.

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.

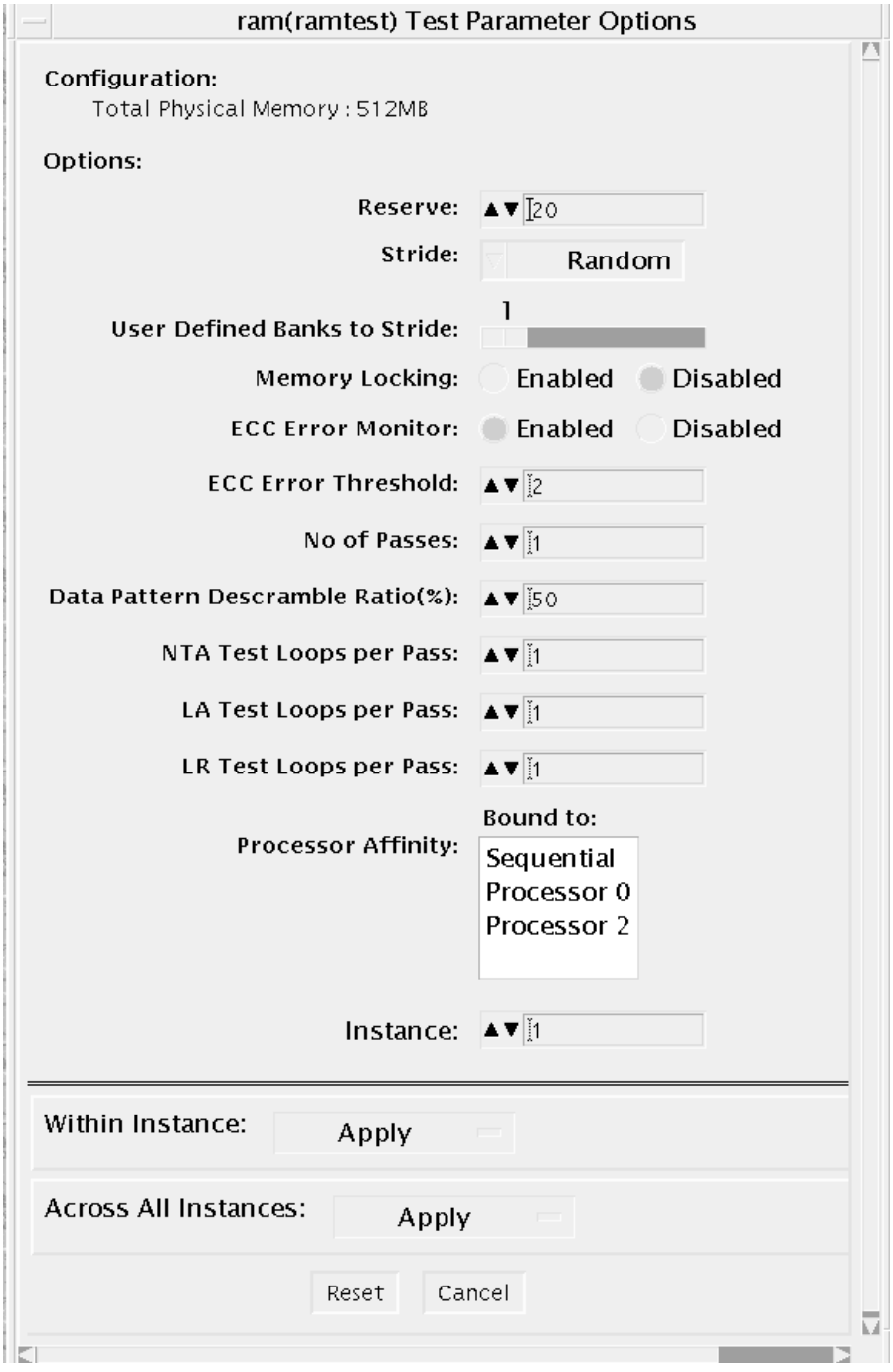

FIGURE 9-1 `ramtest` Test Parameter Options Dialog Box

The following table details the `ramtest` options:

TABLE 9-1 `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.
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).
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".
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.

TABLE 9-1 `ramtest` Options *(Continued)*

`ramtest` **Options**	**Description**
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.
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.

ramtest Test Modes

TABLE 9-2 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=<*Disabled* | *Enabled* | *Random* | *UserDefined*> **]**
[userstride=<*1 - 16*> **]**
[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*> **]**]

TABLE 9-3 `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.
`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).
`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

TABLE 9-3 `ramtest` Command-Line Syntax

Argument	Description
`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".
`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.
`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.
`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`	Descramble 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.

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 `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 "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

CHAPTER 10

Disk and Floppy Drives Test (disktest)

disktest verifies the functionality of hard drives and diskette drives using three subtests (see TABLE 10-1): Media, File System, and Asynchronous I/O. 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.

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 10-1 `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.

TABLE 10-1 `disktest` Subtests

Subtest	Description
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.
Self subtest	This test is run as part of the Media subtest. You can not enable or disable this subtest. It is performed in Functional test mode only. This subtest instructs the disk to run its internal diagnostics. A failure in the Self subtest indicates a hardware problem with the actual device under test.
Write/Read disk buffer subtest	This test is run as part of the Media subtest. You can not enable or disable this subtest. It is performed in Functional test mode only. This subtest verifies the Write/Read buffer for the disk. This subtest uses the pattern specified for the Media subtest or the default pattern to write a defined number of iterations to the Write/Read buffer. A failure in the Write/Read buffer subtest indicates a problem in the upstream component and not with the actual test disk.

`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.

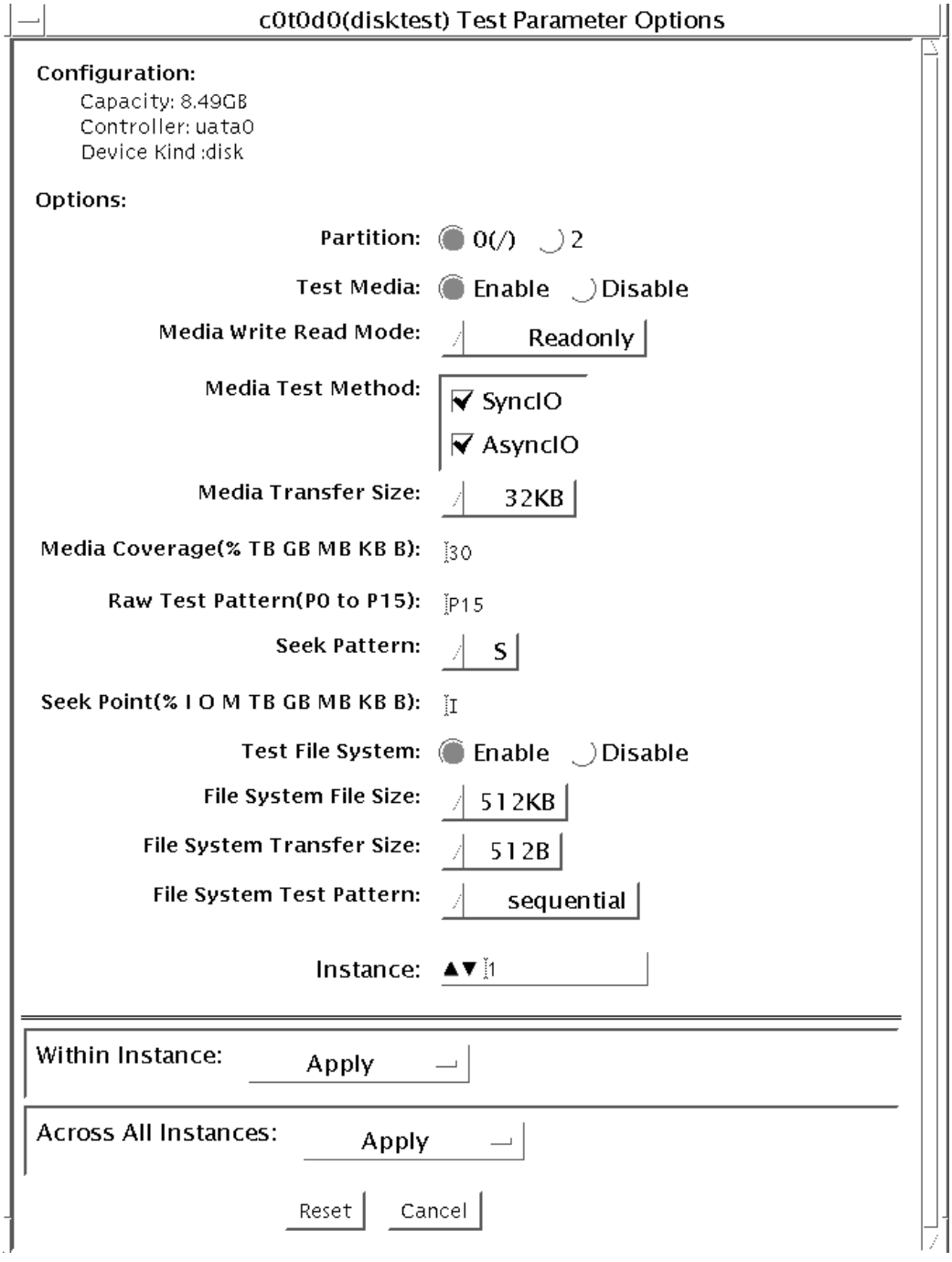

FIGURE 10-1 `disktest` Test Parameter Options Dialog Box

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

TABLE 10-2 `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 P15)	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
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 10-2 `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 10-2 `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 P15) • 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 P15) • 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 10-3 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 two additional subtests (Self subtest and Write/Read device buffer subtest) for enclosures. These two additional subtests help in isolating the errors and are completed before disktest continues with the Media subtest or File System subtest. In Functional test mode, disktest also monitors enclosures by checking for errors in the Read link status counters and issues a warning if any errors are detected.
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(<0-15>) | 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 10-4 `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* • *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 10-4 `disktest` Command-Line Syntax *(Continued)*

Argument	Description
`rawpattern=`*P(<0-15>)* | *0x<8 digit data pattern>*	`rawpattern` could be specified as a pre-defined pattern set, *P(0-15)*, 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 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 10-4 `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 `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 11

Sun Fire™ V880 FC-AL Disk Backplane Test (`dpmtest`)

`dpmtest` exercises and verifies the Fibre-Channel Mass Storage Subsystem in Sun Fire V880 product line platforms. `dpmtest` exercises various tests in the Fibre-Channel Backplane firmware for validating the mass storage system.

No special hardware is required to run the `dpmtest` test.

`dpmtest` 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.

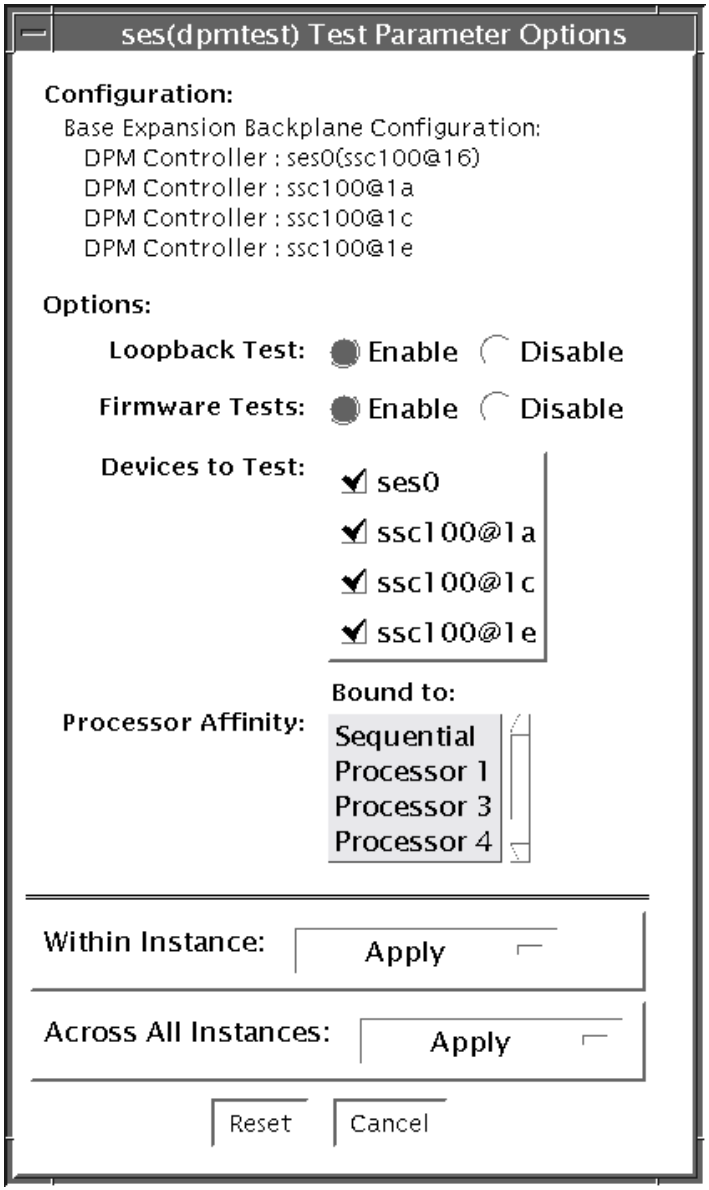

FIGURE 11-1 dpmtest Test Parameter Options Dialog Box

TABLE 11-1 `dpmtest` Test Options

Option	Description
Firmware Test	When enabled, the subtest runs the system friendly firmware tests on each of the selected SES/SSC100 devices. By default it is enabled.
Loopback Test	When Enabled, the subtest will cause the SES device to loop packets around the fiber loop with varying data patterns. The device reads the packet after the packet is received, and verifies that the data payload is correct. By default it is enabled. Note: This test will run only on SES/SSC100 devices which are in the base backplane.
Devices to Test	The SES/SSC100 devices being tested. Users have an option to select or deselect each device for being tested. By default all the devices are selected for testing. Note: At least one device has to be selected for testing. If the user tries to deselect all of the devices, then an error message will be popped up. Note: If the device has both fibre and i2c paths, only the fibre path is listed under 'Devices to Test'. When you perform the tests on this device, the tests are run on both fibre and i2c paths.

dpmtest Test Modes

TABLE 11-2 dpmtest Supported Test Modes

Test Mode	Description
Connection	The test opens each selected device, extracts information about the device (wwn/wwpn, firmware revision, drives installed, temperatures, etc.) and displays the information for the user. If the device has both fibre and i2c paths, then information will be extracted from both the paths. After the test is performed on all the selected devices, the test closes the devices and exists.
Functional	The test opens each selected device and runs the selected subtests against the device. When fully run, the test closes the device and reports the results. Note: When no subtests are selected and you try to perform the functional testing, then just a configuration check will be performed.

dpmtest Command-Line Syntax

/opt/SUNWvts/bin/dpmtest *standard_arguments* **-0 dev=***[device name],*
dpmdev=*[device1+device2+...]***, fwtest=***[Enable | Disable]***, lb=***[Enable | Disable]*

TABLE 11-3 `dpmtest` Command-Line Syntax

Argument	Description
`-o dev=`*[device name]*	*[device name]* is the path name of the device being tested. The default value is ses. Since the current SunVTS infrastructure doesn't allow specifying multiple devices under the `dev` suboption, this suboption is not used in `dpmtest`. A new suboption dpmdev has been introduced to satisy this requirement.
`dpmdev=`*[device1+device2...]*	*device1*, *device2*,... represent the SES/SSC100 devices being tested. The default value is all the SSC100s present in the system. Note: The values for the `dpmdev` suboption can be device names such as `ses0`, `ses1`, `ssc100@16`, `ssc100@1a`, etc. Multiple values can be specified with a '+' (plus sign) seperator. An absolute path through fibre paths to devices are allowed (for expample, `/dev/es/ses0`) as `dpmdev` suboption values. However, absolute paths through a i2c path to devices are not allowed because commas are not allowed as part of a suboption value. Commas delimit suboptions in the options string (for example, `/devices/pci@9`, `700000/ebus@1/i2c@1`, `30/controller@0`, `16:ssc100`). Note: The following devices may be specified for the `dpmdev` suboption values in the Sun Fire V880 product line platforms: Fibre Path: • `ses0` - fibre path to base backplane's SSC100 (`/dev/es/ses0`) device on loopA. • `ses1` - fibre path to base backplane's SSC100 (`/dev/es/ses1`) device on loopB. This is valid only when a PCI FC Network Adapter is connected to loopB. I2C Path: • `ssc100@16` - base backplane's SSC100 device on loopA through a `i2c` path. • `ssc1001a` - base backplane's SSC100 device on loopB through a `i2c` path. • `ssc1001e` - expansion backplane's SSC100 device on loopB through a `i2c` path. Note: The exact fibre path device node names (`ses0`, `ses1`, etc..) may vary depending on device nodes created in the system. The valid fibre path device nodes, that `dpmtest` found during probing, can be found under 'Devices to Test' in the `dpmtest` Test Parameter Options dialog box.
`lb=`*[Enable | Disable]*	*Enable* or *Disable* loopback test. The default value is *Enable*. Note: The loopback test will run only on SES/SSC100 devices that are in the base backplane.
`fwtest=`*[Enable | Disable]*	*Enable* or *Disable* firmware tests. The default value is *Enable*.

CHAPTER 12

Ethernet Loopback Test (netlbtest)

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

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

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

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

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

netlbtest Test Requirements

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

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

To run `netlbtest`, a loopback connector must be connected to the Ethernet interface. A loopback connector provides the network interface driver the necessary link for testing, while maintaining isolation from a live network. The loopback connector is required for both internal and external tests of the Ethernet device.

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

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

netlbtest Options

To reach the 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.

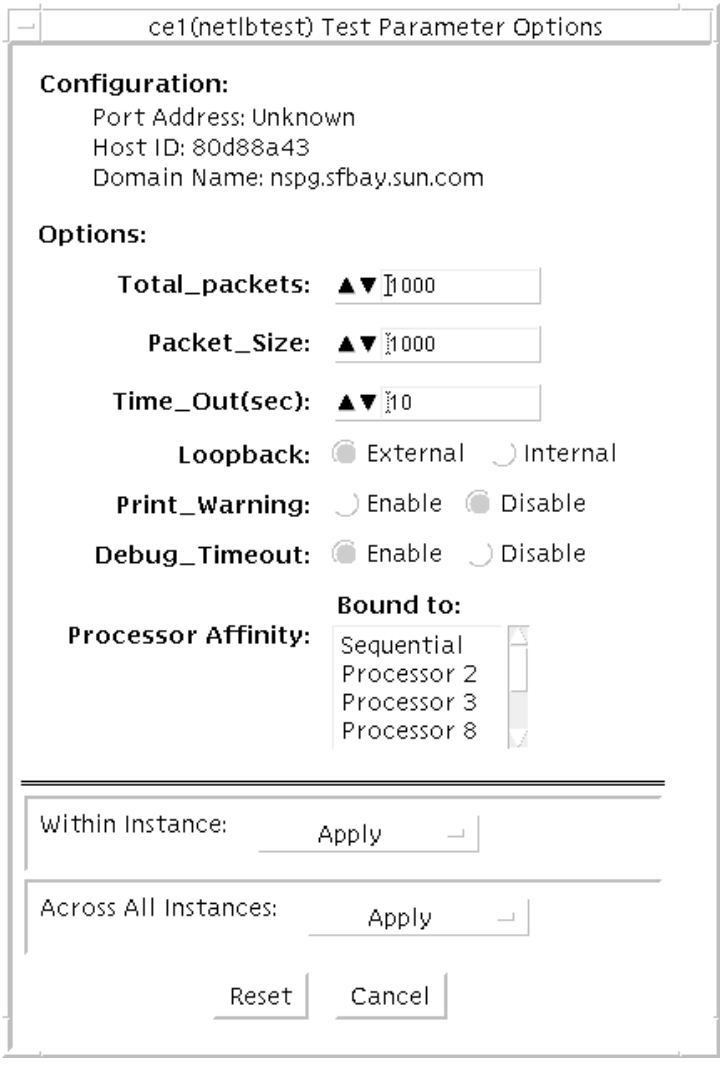

FIGURE 12-1 `netlbtest` Test Parameter Options Dialog Box

Refer to TABLE 12-1 for test parameter descriptions.

TABLE 12-1 `netlbtest` Options

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

`netlbtest` Test Modes

TABLE 12-2 `netlbtest` Supported Test Modes

Test Mode	**Description**
Functional (Offline)	Runs the full set of subtests. It is assumed that the host is not connected to the network through the intended test device(s).

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

`netlbtest` Command-Line Syntax

`/opt/SUNWvts/bin/netlbtest` *standard_arguments*
`-o dev=`*device*`,` **`tpkts=`***n*`,` **`pksz=`***pkt_size*`,` **`lb=`***Internal*`,` **`warn=`***Disable*`,`
`timeout=`*number_of_seconds*

TABLE 12-3 `netlbtest` Command-Line Syntax

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

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

Multiprocessor Test (`mptest`)

The `mptest` verifies the hardware functionality of multiprocessor hardware. The test provides diagnostic test coverage for different aspects of multiprocessor functionality like 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.

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.

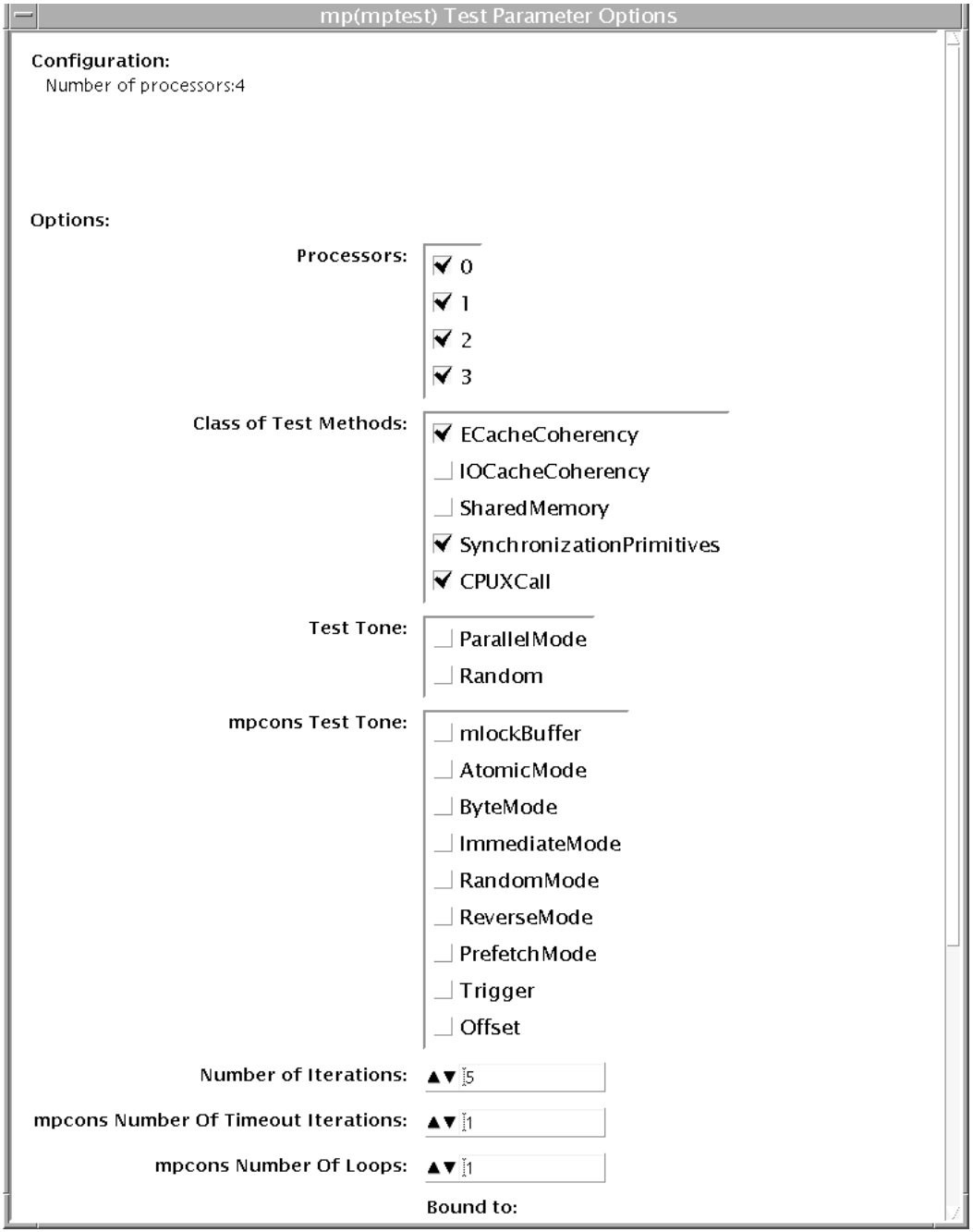

FIGURE 13-1 `mptest` Test Parameter Options Dialog Box

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 13-1 can be run alone or concurrently with other options.

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

`mptest` **Options**	**Description**
`mpcons` Test Tone	This option is used to select the tone for `mpconstest` cases in the `mptest`. These are options 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.

mptest Test Modes

TABLE 13-2 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*

TABLE 13-3 `mptest` Command-Line Syntax

`mptest` **Options**	**Description**
`M=`*4+5+6+7*	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. Example: If you want to select CPU IDs 4, 5, 6 and 7, specify: **`M=`***4+5+6+7*
`method=` *ECacheCoherency+IOCacheCoherency+SynchronizationPrimitives+SharedMemory+CPUXCall*	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. 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, SynchronizationPrimitives, and CPUXCall are selected by default.
`tone=`*ParallelMode+Random*	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 13-3 `mptest` Command-Line Syntax *(Continued)*

`mptest` **Options**	**Description**
`mpcons_tone`= mlockBuffer+AtomicMode+ByteMode +ImmediateMode+RandomMode+Rev erseMode+PrefetchMode+Trigger+Off set	This option is used to select the tone for `mpconstest` cases in the `mptest`. These are options 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]*	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.
`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.

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 14

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.

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.

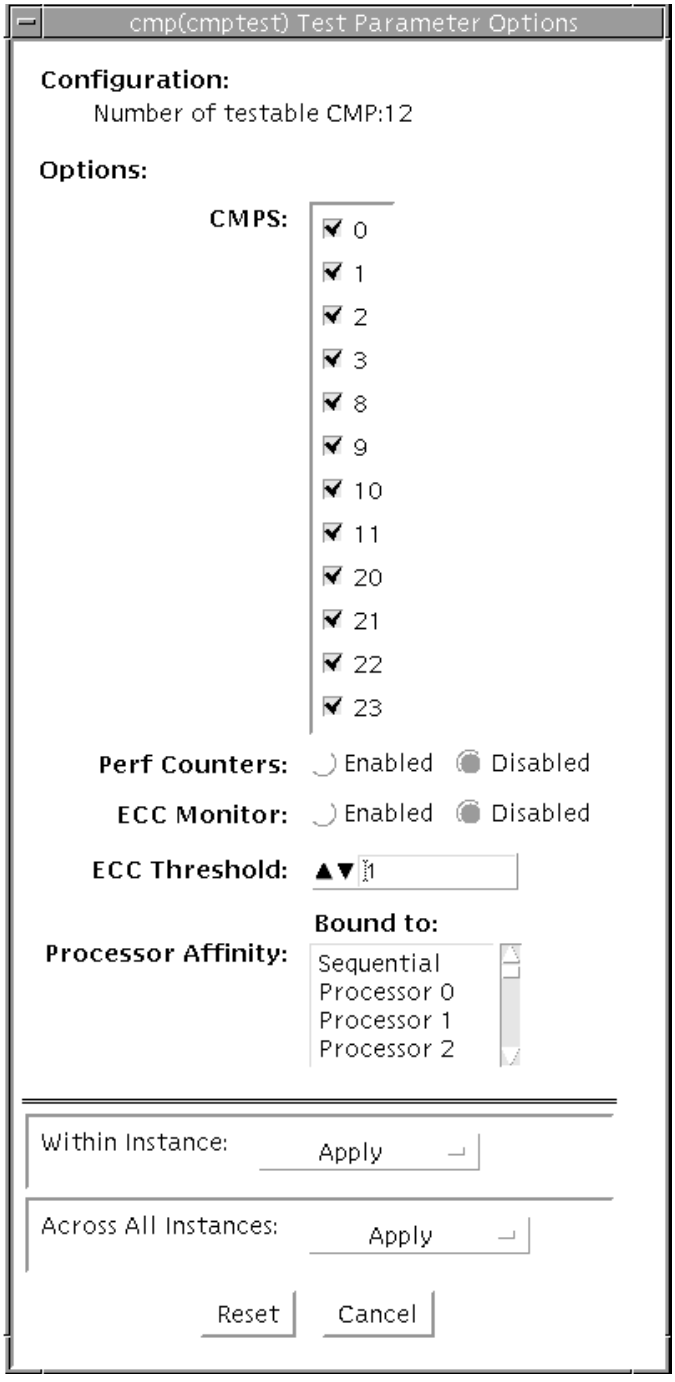

FIGURE 14-1 `cmttest` Test Parameter Options Dialog Box

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 14-1 `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.

cmttest Test Modes

TABLE 14-2 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*

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*

TABLE 14-3 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 GUI, CPU ID of Core 0 will be taken as the identifier for a CMT. For displaying the Error/INFO/LOG messages, the CPU ID of the core 0 is used.

TABLE 14-3 `cmttest` Command-Line Syntax

Arguments	Description
`em=`*Enabled* | *Disabled*	This option is used to Enable or Disable ECC error monitoring. The default value is Disabled.
`threshold=`*[0-255]*	The range is [0-255]. This determines how many correctable ECC errors can occur in the elapsed time before `cmttest` reports a test failure. 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.

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 15

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. 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.

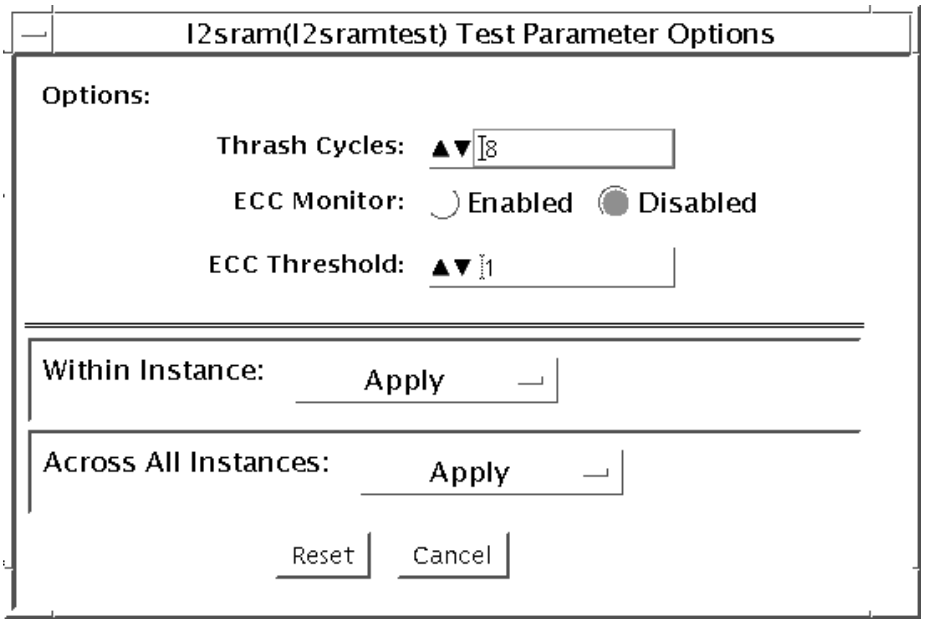

FIGURE 15-1 `l2sramtest` Test Parameter Options Dialog Box

TABLE 15-1 `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.

Note – The `l2sramtest` automatically handles processor binding. Users are advised to not use the Processor Affinity option for the `l2sramtest`.

l2sramtest Test Modes

TABLE 15-2 l2sramtest Supported Test Modes

Test Mode	Description
Connection	Performs the Connection subtest.
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]]*

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 15-3 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.

CHAPTER 16

Alarm Card Test for Netra™ CT Systems (`alarm2test`)

The `alarm2test` exercises the Alarm Card and System Controller Board on the Sun Netra™ CT 410 and CT 810 systems.

The Alarm Card is a hot-swappable add-on option for the Netra CT systems which provides secure remote access for system monitoring, failure recovery, and alarm notification. The Alarm Card can be used in both front and rear-access systems.

This test is not scalable.

Note – The Netra CT 410/810 system only runs the 64-bit OS (to take full advantage of UltraSPARCII). Although, `alarm2test` is available in 32-bit and 64-bit mode, only the 64-bit version of `alarm2test` can be performed on a Netra CT 410/810 system.

`alarm2test` Requirements

Solaris 9 4/03 operating environment or later is required to perform the `alarm2test`. Ethernet loopback and serial loopback connectors are also required to perform the `alarm2test`. In addition, you are required to select the Intervention mode due to the serial and Ethernet loopback connectors.

`alarm2test` Subtests

`alarm2test` consists of eight subtests which test and report on the following:

- Ethernet Internal, External, PHY loopback and PING test
- Internal/External loopback test on serial ports
- Checksum test on FLASH
- Alarm relay on/off test
- System status panel LED test
- Fan status test
- Power supply test
- FruID checksum test

`alarm2test` 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.

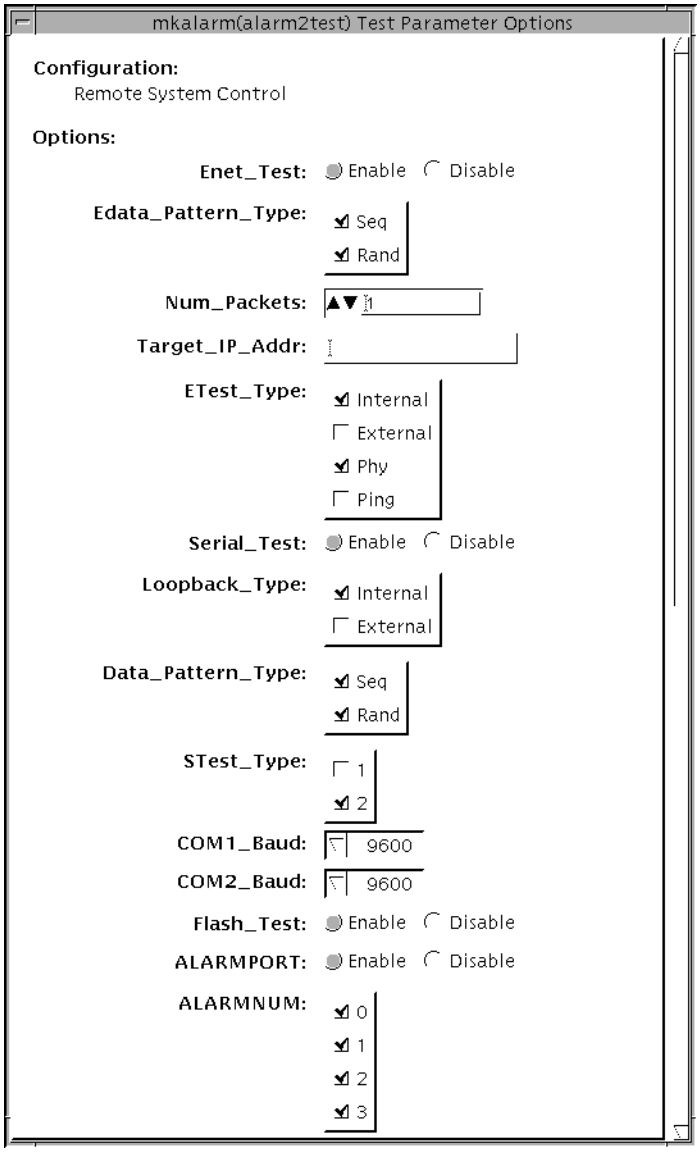

FIGURE 16-1 `alarm2test` Test Parameter Options Dialog Box With the Scroll Bar at the Top

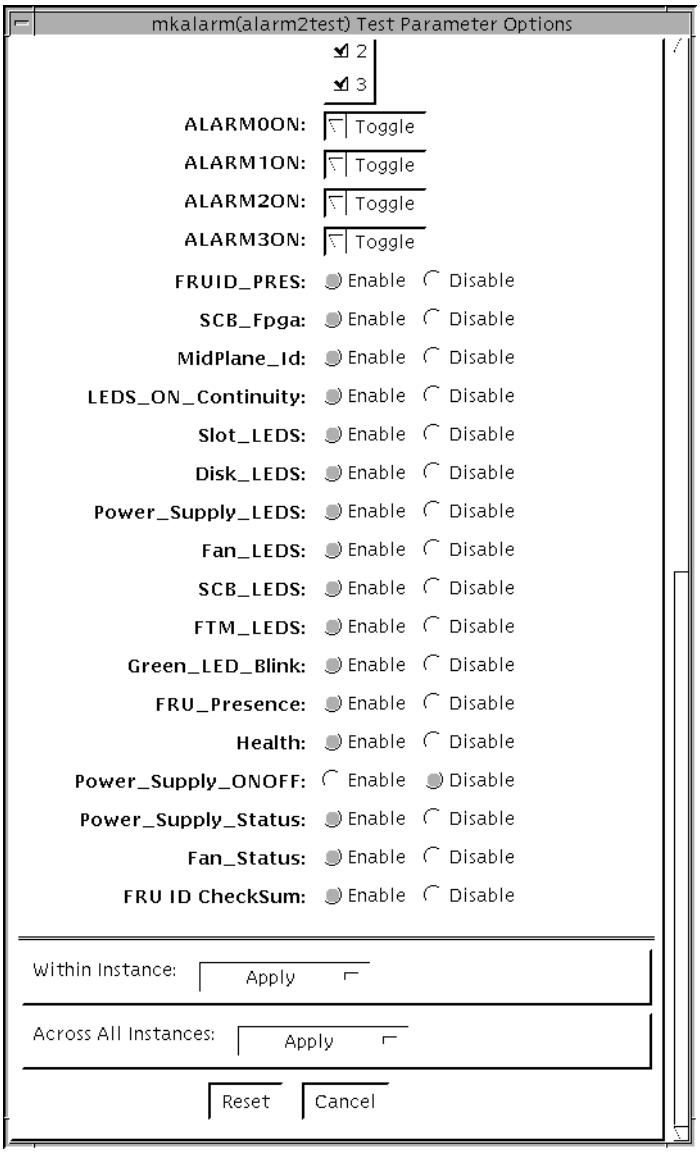

FIGURE 16-2 `alarm2test` Test Parameter Options Dialog Box With the Scroll Bar at the Bottom

TABLE 16-1 `alarm2test` Options

Option	Description
Enet_Test	Enables or disables Ethernet testing.
Edata_Pattern_Type	Selects the type of data pattern for `Enet_Test`: Sequential, Random, or both.
Num_Packets	Specifies the number of data packets to be sent in one test loop.
Target_IP_Addr	Specifies the IP address of a host to use for the ping test.
Etest_Type	Selects any or all internal, external, Phy (ethernet transceiver), or ping tests.
Serial_Test	Enables or disables `serial_test`.
Loopback_Type	Selects internal loopback, external loopback, or both.
Data_Pattern_Type	Selects the type of data pattern for `serial_test`: Sequential, Random, or both.
STest_Type	Selects ports to be tested: `c`, `d`, `u`, or `v`.
COM1_Baud	Selects the alarm card's COM1 port baud rate.
COM2_Baud	Selects the alarm card's COM2 port baud rate.
Flash_Test	Enables or disables the flash checksum test.
ALARMPORT	Enables or disables the alarmport test.
ALARMNUM	Selects any or all alarm ports to be tested: 0, 1, 2, 3.
ALARM0ON	Turns on, turns off, or toggles (on then off) alarm port 0.
ALARM1ON	Turns on, turns off, or toggles (on then off) alarm port 1.
ALARM2ON	Turns on, turns off, or toggles (on then off) alarm port 2.
ALARM3ON	Turns on, turns off, or toggles (on then off) alarm port 3.
FRUID_PRES	Enable/Disable FRU ID Presence test
SCB_Fpga	Enable/Disable scb_fpga register test
MidPlane_Id	Enable/Disable Midplane ID test
LEDS_ON_Continuity	Enable/Disable SCB LEDs test
Slot_LEDS	Enable/Disable Slot LEDs test
Disk_LEDS	Enable/Disable Disk LEDs test
Power_Supply_LEDS	Enable/Disable Power Supply LEDs test
Fan_LEDS	Enable/Disable Fan LED test
SCB_LEDS	Enable/Disable SCB Resgister LEDs test

TABLE 16-1 `alarm2test` Options

Option	Description
FTM_LEDS	Enable/Disable Front Tranission Module LEDs test
Green_LED_Blink	Enable/Disable Green LED Blink test
FRU_Presence	Enable/Disable FRU Presence test
Health	Enable/Disable Health test
Power_Supply_ONOFF	Enable/Disable Power Supply On/Off test
Power_Supply_Status	Enable/Disable Power Suppluy Status test
Fan_Status	Enable/Disable Fan Status test
FRU ID CheckSum	Enable/Disable FRU ID Checksum test for Midplane, SCB, Alarm, Fan1/Fan2, and Power Supply1/Supply2

`alarm2test` Loopbacks

The loopback tests use the following external loopbacks:

- Ethernet loopback test—standard RJ-45 connector. Connect pin 1 to pin 3, and pin 2 to pin 6.
- Serial loopback test—RJ-45. Connect pin 6 to pin 3, pin 1 to pin 8, and pin 2 to pin 7.

alarm2test Test Modes

TABLE 16-2 alarm2test Supported Test Modes

Test Mode	Description
Connection	Reports the status of the alarm card.
Functional	Runs the the full set of subtests.

alarm2test Command-Line Syntax

/opt/SUNWvts/bin/alarm2test *standard_arguments* **-o enet=** *E(nable)/D(isable)*, **epattype=***seq+rand*, **target=***IP_Address*, **etest=** *Internal+External+Ping+Phy*, **serial=***E(nable)* | *D(isable)*, **slb=***Internal+External*, **spattype=***Seq+Rand*, **com1baud=** *ALL* | *1200* | *2400* | *4800* | *9600* | *19200* | *38400* | *56000*, **com2baud=** *ALL* | *1200* | *2400* | *4800* | *9600* | *19200* | *38400* | *56000*, **flash=***E(nable)* | *D(isable)*, **aport=***E(nable)* | *D(isable)*, **anum=***0+1+2+3*, **a0on=***On* | *Off* | *Toggle*, **a1on=** *On* | *Off* | *Toggle*, **a2on=***On* | *Off* | *Toggle*, **a3on=***On* | *Off* | *Toggle*, **FruIdPres=** *E(nable)* | *D(isable)*, **FpgaId=***E(nable)* | *D(isable)*, **MidPlaneId=***E(nable)* | *D(isable)*, **Continuity=***E(nable)* | *D(isable)*, **SlotLeds=***E(nable)* | *D(isable)*, **DiskLeds=** *E(nable)* | *D(isable)*, **PsupplyLeds=***E(nable)* | *D(isable)*, **FanLeds=** *E(nable)* | *D(isable)*, **ScbLeds=***E(nable)* | *D(isable)*, **FtmLeds=***E(nable)* | *D(isable)*, **GreenLedsBlink=***E(nable)* | *D(isable)*, **FruPresence=***E(nable)* | *D(isable)*, **Health=***E(nable)* | *D(isable)*, **PowerSupply=***D(isable)* | *E(nable)*, **PsupplyStatus=** *E(nable)* | *D(isable)*, **FanStatus=***E(nable)* | *D(isable)*, **FruIdChkSum=** *E(nable)* | *D(isable)*

TABLE 16-3 alarm2test Command-Line Syntax

Argument	Explanation
enet=*E(nable)/D(isable)*	Enables or disables Ethernet testing.
epattype=*seq+rand*	Selects the type of data pattern for Enet_Test: Sequential, Random, or both.
target=*IP_Address*	Specifies the IP address of a host to use for the ping test.
etest= *Internal+External+Ping+Phy*	Selects any or all internal, external, Phy (ethernet transceiver), or ping tests.
serial=*E(nable)/D(isable)*	Enables or disables serial_test.

TABLE 16-3 `alarm2test` Command-Line Syntax

Argument	Explanation *(Continued)*
`slb=`*I+E*	Selects internal loopback, external loopback, or both.
`spattype=`*seq+rand*	Selects the type of data pattern for `serial_test`: Sequential, Random, or both.
`com1baud=`*ALL* | *specific_baud*	Defines baud rates to be used in testing the alarmcard's COM1 port.
`com2baud=`*ALL* | *specific_baud*	Defines baud rates to be used in testing the alarmcard's COM2 port.
`flash=`*E(nable)/D(isable)*	Enables or disables the flash checksum test.
`aport=`*[E]nable* | *[D]isable*	Enables or disables the alarmport test.
`anum=`*0+1+2+3*	Selects any or all alarm port to be tested: 0, 1, 2, 3
`a0on=`*On* | *Off*/ | *T]oggle*	Turns on, turns off, or toggles (on then off) alarm port 0.
`a1on=`*On* | *Off* | *[T]oggle*	Turns on, turns off, or toggles (on then off) alarm port 1.
`a2on=`*On* | *Off* | *[T]oggle*	Turns on, turns off, or toggles (on then off) alarm port 2.
`a3on=`*On* | *Off* | *[T]oggle*	Turns on, turns off, or toggles (on then off) alarm port 3.
`FruIdPres=`*E(nable)* | *D(isable)*	Enable/Disable FRU ID Presence test
`FpgaId=`*E(nable)* | *D(isable)*	Enable/Disable scb_fpga register test
`MidPlaneId=`*E(nable)* | *D(isable)*	Enable/Disable Midplane ID test
`Continuity=`*E(nable)* | *D(isable)*	Enable/Disable SCB LEDs test
`SlotLeds=`*E(nable)* | *D(isable)*	Enable/Disable Slot LEDs test
`DiskLeds=`*E(nable)* | *D(isable)*	Enable/Disable Disk LEDs test
`PsupplyLeds=`*E(nable)* | *D(isable)*	Enable/Disable Power Supply LEDs test
`FanLeds=`*E(nable)* | *D(isable)*	Enable/Disable Fan LED test
`ScbLeds=`*E(nable)* | *D(isable)*	Enable/Disable SCB Resgister LEDs test
`FtmLeds=`*E(nable)* | *D(isable)*	Enable/Disable Front Tranission Module LEDs test
`GreenLedsBlink=` *E(nable)* | *D(isable)*	Enable/Disable Green LED Blink test
`FruPresence=`*E(nable)* | *D(isable)*	Enable/Disable FRU Presence test
`Health=`*E(nable)* | *D(isable)*	Enable/Disable Health test
`PowerSupply=`*D(isable)* | *E(nable)*	Enable/Disable Power Supply On/Off test

TABLE 16-3 `alarm2test` Command-Line Syntax

Argument	Explanation *(Continued)*
`PsupplyStatus=` *E(nable) | D(isable)*	Enable/Disable Power Suppluy Status test
`FanStatus=`*E(nable) | D(isable)*	Enable/Disable Fan Status test
`FruIdChkSum=`*E(nable) | D(isable)*	Enable/Disable FRU ID Checksum test for Midplane, SCB, Alarm, Fan1/Fan2, and Power Supply1/Supply2

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 17

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.

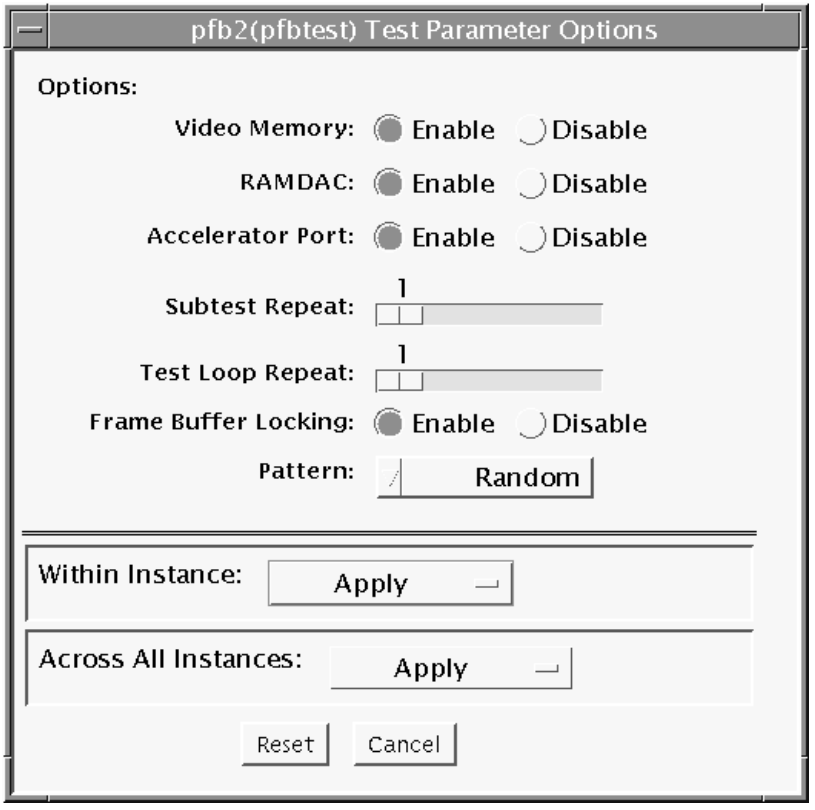

FIGURE 17-1 `pfbtest` Test Parameter Options Dialog Box

TABLE 17-1 `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.

TABLE 17-1 `pfbtest` Options *(Continued)*

`pfbtest` **Options**	**Description**
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.
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 17-2 `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 17-3 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 18

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.

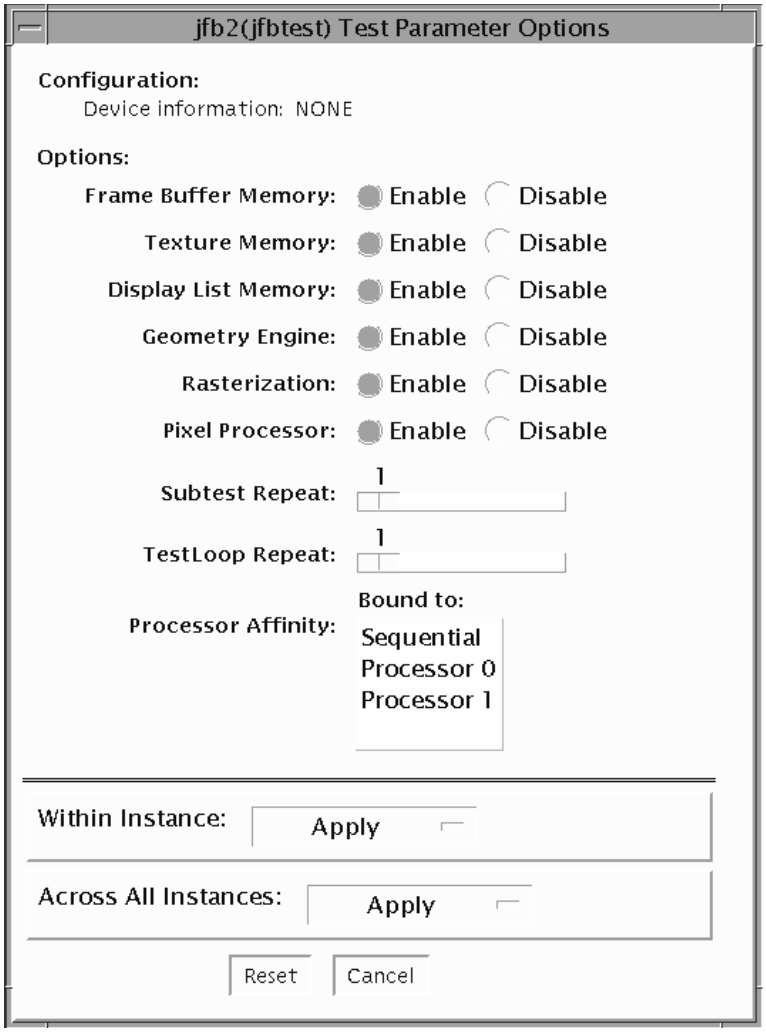

FIGURE 18-1 `jfbtest` Test Parameter Options Dialog Box

TABLE 18-1 `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 18-1 `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 18-1 `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 18-2 `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 18-3 `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 19

Sun™ XVR-4000 Graphics Accelerator Test (`zulutest`)

The `zulutest` does functional testing of the Sun™ XVR-4000 graphics accelerator device. zulutest detects and adapts to the video modes of Sun XVR-4000. All zulutest tests can be performed in several screen resolutions such as standard, Stereo, and high resolution. In Stereo mode, all tests write into the right and left eyes unless you specify otherwise. Use the fbconfig -dev device-name -prconf command to display the configuration of the frame buffer you want to test.

You can interrupt zulutest using Control-C. Turn off all other keyboard input if the CDE user interface is running on the unit being tested. Test accuracy is checked using a checksum algorithm. Possible locations of failing pixels are identified, in addition to the failing FRU.

zulutest is only available in 64-bit mode.

Caution – Do not run any 3D graphics applications screen lock or screen saver programs that uses the Sun XVR-4000 graphics accelerator port while running `zulutest`. This combination causes SunVTS to return incorrect errors.

`zulutest` Test Requirements

Disable all screen locks and screen savers before testing any graphics device. Type **`xset s off`** at a UNIX® prompt to disable the Solaris screen saver. Disable the Power Management software if it is running.

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

To start SunVTS with the `vtsui`, and without the `vtsk`, you must add the host name to `xhost` as follows: `xhost + host_name`

Using `zulutest` Without X-Windows

If you perform `zulutest` on a system that was powered on without running X-Windows, you must bring up X-Windows on the Sun XVR-4000 Graphics Accelerator device under test and kill the X-Windows process before performing `zulutest`. Otherwise, the Convolve subtest will fail, and other subtests may also fail.

Note – You must enable multisampling with the `fbconfig` command before performing the following workaround. To perform `zulutest` with X-Windows (CDE) the following workaround is not necessary.

Workaround

To bring up X-Windows on the Sun XVR-4000 Graphics Accelerator device under test, enter the following command:

```
/usr/openwin/bin/Xsun -dev /dev/fbs/device_name &
```

It takes 30 to 45 seconds before Xsun comes up. To kill the Xsun process, enter the following command:

```
pkill -KILL Xsun
```

Once the Xsun process is killed, the `zulutest` can be performed without the incorrect subtest errors.

The Sun XVR-4000 Graphics Accelerator cannot perform video read back in Interlaced and Stereo modes because the Convolve subtest cannot keep up.

For `zulutest` to be able to perform the Convolve subtest, multisampling must be enabled.

`zulutest` 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 `zulutest` options are enabled except for the Stereo test.

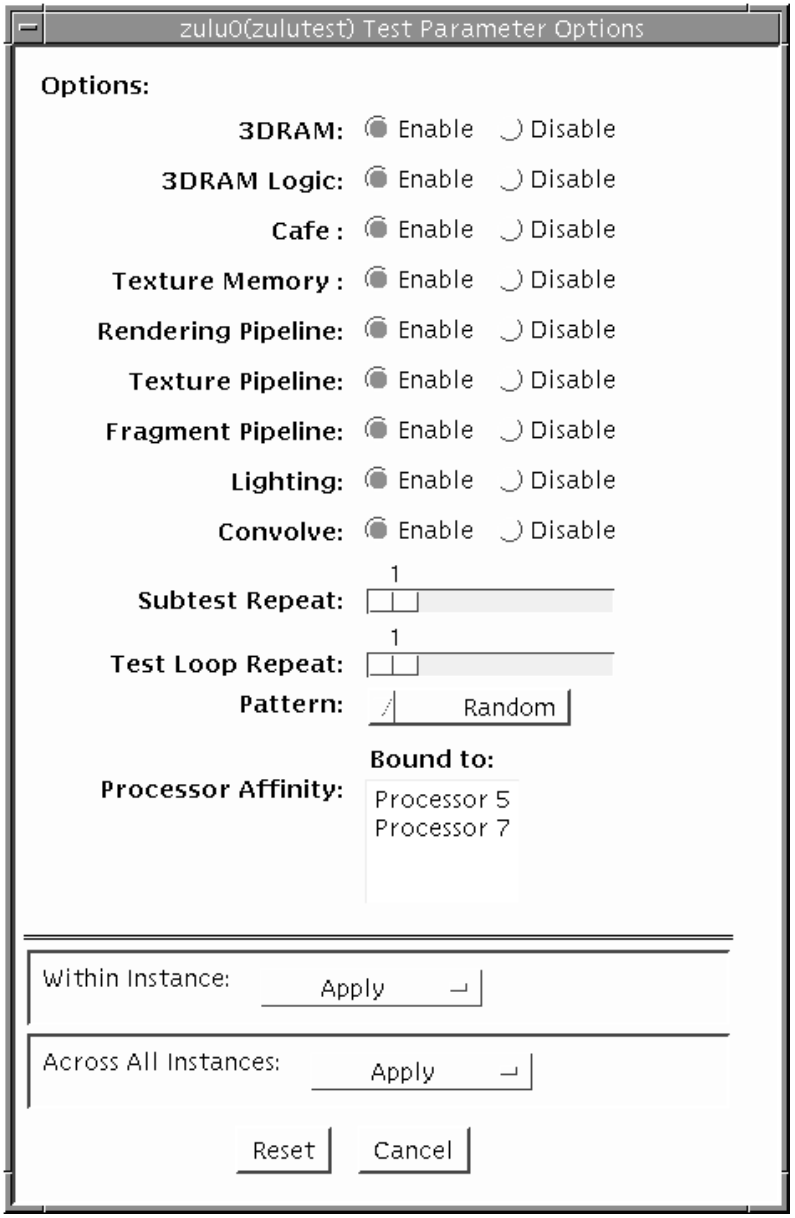

FIGURE 19-1 `zulutest` Test Parameter Options Dialog Box

TABLE 19-1 `zulutest` Options

`zulutest` **Options**	**Description**
3DRAM test	The 3DRAM test thoroughly tests the video memory in the Sun XVR-4000 graphics accelerator using 512-bit reads and writes. 3DRAM makes a full-screen pass, consisting of a write and a read to each pixel location, for each access mode on the list below. You can use either random data or specify data at the command line. A second pass is made with the one's complement of the data used in the first pass so that each memory location is tested with both a zero and a one. Errors in this subtest are attributed to the 3DRAM. A failing chip is indicated by (X, Y) locations and device-specific "U" numbers in the following access modes: • SFB Stencil 8 • SFB WID 16 • FB RGBAZ 64 - Buffer A • SFB RGBAZ 64 - Buffer B

TABLE 19-1 `zulutest` Options *(Continued)*

`zulutest` **Options**	**Description**
3DRAM Logic test	The 3DRAM Logic test provides logical functionality to the Sun XVR-4000 graphics accelerator. The following services are tested: • Compare Controls—Match AB • Compare Controls—Magnitude AB • Compare Controls—Match C • Compare Controls—Magnitude C • Match Mask—AB • Magnitude Mask—AB • Match Mask—C • Magnitude Mask—C • Raster Operations—RGB • Raster Operations—X • Raster Operations—YZ • Plane Mask—RGB Each function is tested separately with a series of SFB64 writes. A total of 16 writes are made for each different test case with Y coordinate values varying from 0 to 30 in increments of 2 pixels. This dotted column organization provides page thrashing and block flashing in all screen resolutions. For each operation, all possible combinations are tested. For example, in `ROP RGB` *new==old* there are three possible values: *new < old, new == old, and new > old*. Each of these cases are tested. Errors in this subtest are attributed to the 3DRAM.
Cafe test	This test will do non-destructive testing of the Cafe memory (RDRAM) and Cafe. Errors in this test are attributed to the Cafe and its memory.
Texture Memory test	Texture Memory test tests out all the of the Texture Memory by writing the data pattern selected (random, 0s, 1s, 5s, or 0xAs). By default, Random data is selected. The data is written using block writes and read back using block reads. Errors in this test are attributed to the Texture Memory and the Texture Memory subsystem.

TABLE 19-1 `zulutest` Options *(Continued)*

`zulutest` **Options**	**Description**
Rendering Pipeline test	Each primitive is tested thoroughly by exercising the following: • Simple Triangles • 2d primitives • 3d Primitives (such as Triangles, 3d lines etc.) • Vertex Processor Errors in this test are attributed to the pipelines of the Sun XVR-4000 graphics accelerator and/or 3DRAM.
Texture Pipeline test	This test renders textured primitives to test: • 2d texture Minification filtering • 2d texture Magnification filtering • 3d texture Minification filtering • 3d texture Magnification filtering • Texture environment • Filter4 and sharpen filters • Anisotropic filter Errors in this test are attributed to the pipelines of the Sun XVR-4000 graphics accelerator and/or 3DRAM.
Fragment Processor test	Fragment Processor test, a subtest, exercises the fragment pipe of each pipeline of the XVR-4000's. Auxiliary clipping (additive and subtractive): • Depth cueing • Alpha blend • Viewport clip (2D and 3D) • Area pattern (transparent and opaque) Errors in this test are attributed to the FBC3 and/or 3DRAM.
Lighting test	The Lighting test exercises Cafe and lighting microcode. This test lights an object with maximum number of lights that XVR-4000 can handle in hardware. A checksum is generated for the rendered image and compared with the checksum generated for the same image on a known good system. Errors in this test are attributed to the Cafe, Microcode, FBC3 and RD RAMs.

TABLE 19-1 `zulutest` Options *(Continued)*

`zulutest` **Options**	**Description**
Convolve test	Convolve test tests the Convolve chips functionality (convolution filters, Color look up tables and Gamma look up tables) along with the video read back functionality of convolves and master chip. This sub test renders an image which is made up of lines drawn radial. Then a block in the center of the image is super sampled and video read back is initiated. Once the video read back data is available to the zulutest, zulutest will generate checksum and compares with the checksum generated on known good system. Errors in this subtest can be attributed to FBC3, 3DRAM, Convolve, Master.
Stereo test	Currently, this sub test is not active. Stereo test displays an object in Stereo mode with different images for the right and left eye. You can verify proper operation by looking at the screen with stereo glasses and following the instructions displayed in the Parameter Options dialog box. This test temporarily switches the monitor into Stereo mode, renders a Stereo image and after displaying the image for five seconds, restores the monitor to its previous resolution.

`zulutest` 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, `zulutest` is only available in Functional test mode.

TABLE 19-2 `zulutest` Supported Test Modes

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

zulutest Command-Line Syntax

/opt/SUNWvts/bin/sparcv9/zulutest *standard_arguments* **-o dev=** *device_name*,**S=***subtest_number*,**F=#***_of_subtest_loops*,**B=#***_of_test_loops*,**P=***test_pattern*

TABLE 19-3 zulutest 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 zulu0.
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 together. For example, *n=0x3* runs both test 1 and test 2; *n=0x180* runs both test *0x080* and test *0x0100*. You do not need the leading zeros. • *n*—0x00001 Video Memory 3DRAM • *n*—0x00002 3DRAM Logic • *n*—0x00004 Cafe • *n*—0x00008 Texture Memory SDRAM • *n*—0x00010 Rendering Pipeline • *n*—0x00020 Texturing Pipeline • *n*—0x00040 Fragment Pipeline • *n*—0x00080 Lighting • *n*—0x00100 Convolve • *n*—0x00200 Stereo More than one subtest can be selected by ORing their subtest numbers. Example: n = 0x00011 indicates 3DRAM and Rendering Pipeline tests. A hex number must start with 0x, decimal numbers are also acceptable. [n = 0xff] If looping on a test, the verbose mode is disabled. F=n : Number of times to repeat each subtest [n = 1]. B=n : Number of times to repeat test loop before passing [n = 1]. P=*pattern* : test pattern - r for random, 0 for 0x00000000, 3 for 0x33333333, 5 for 0x55555555, or 9 for 0x9999999. [*pattern=r*]
F=#*_of_subtest_loops*	The number of times to repeat each subtest. The default is 1.
B=#*_of_test_loops*	The number of times to repeat a test loop before passing. The default is 1.
P=*test_pattern*	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 the 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 `zulutest` 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-4000 graphics accelerator.

CHAPTER 20

Blade Support Chip Test (`bsctest`)

The `bsctest` exercises the Blade Support Chip and supporting hardware used in Sun Fire™ B100 blade systems. This includes the Open Boot Prom (OBP) and Time of Day (ToD) Prom chips.

Caution – If the LED subtest is selected, please be aware that LEDs on the blade will change. They will return to their correct state when the test is completed.

`bsctest` 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.

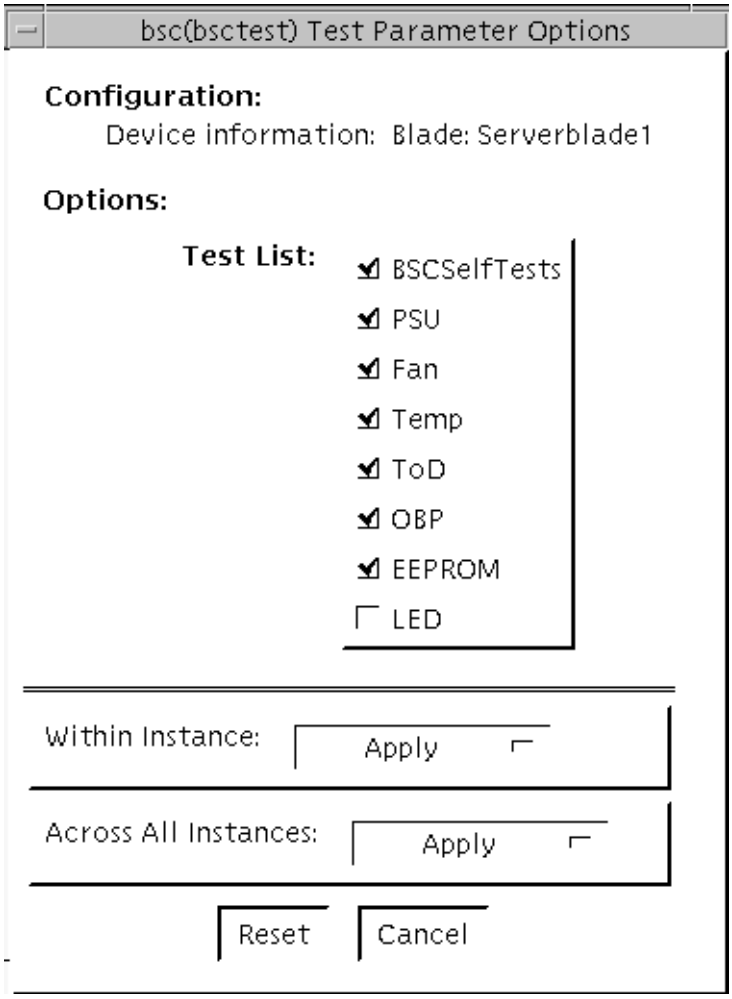

FIGURE 20-1 `bsctest` Test Parameter Options Dialog Box

TABLE 20-1 `bsctest` Options

`bsctest` **Options**	**Description**
BSCSelfTests	Calls on the BSC to execute its built-in self tests.
PSU	Performs read-only checks of Power Supply status.
Fan	Performs read-only checks of Fan status.
Temp	Performs read-only checks of Temperature Monitor status.
ToD	Performs read-only checks of Time of Day chip.

TABLE 20-1 `bsctest` Options *(Continued)*

`bsctest` **Options**	**Description**
OBP	Performs read-only checks of platform specific Open Boot properties.
EEPROM	Performs read-only check of EEPROM.
LED	Performs read-only check of Service Required LED status and performs a test in which all three LEDs (Power, Service Required, and Ready to Remove) are flashed simultaneously at 4Hz and then returned to their original state.

`bsctest` Test Modes

TABLE 20-2 `bsctest` Supported Test Modes

Test Mode	**Description**
Connection	Opens the BSC, OBP, and ToD devices.
Functional	Performs all tests with the LED testing off by default.
Online	Performs all tests except BSCSelfTests and LED *Flashing* test.

`bsctest` Command-Line Syntax

`/opt/SUNWvts/bin/bsctest` *standard_arguments* **`[-o dev`**=*device_name* **`test`**=*<test_list>***`]]`**

TABLE 20-3 `bsctest` Command-Line Syntax

Argument	**Description**
`dev=`*device_name*	*device_name* is the device to be tested, for example, `bsc`
`test`=*test_list*	*testlist* is the list of subtests, for example: `BSCSelfTests`, `PSU`, `Fan`, `Temp`, `ToD`, `OBP`, `EEPROM`, `LED`

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 the 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 21

Environmental Test (`env6test`)

`env6test` exercises and validates environmental subsytems. This test contains subtests to exercise a system's fans, keyswitch, LEDs, power supplies, and temperature sensors.

This test is not scalable.

`env6test` 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.

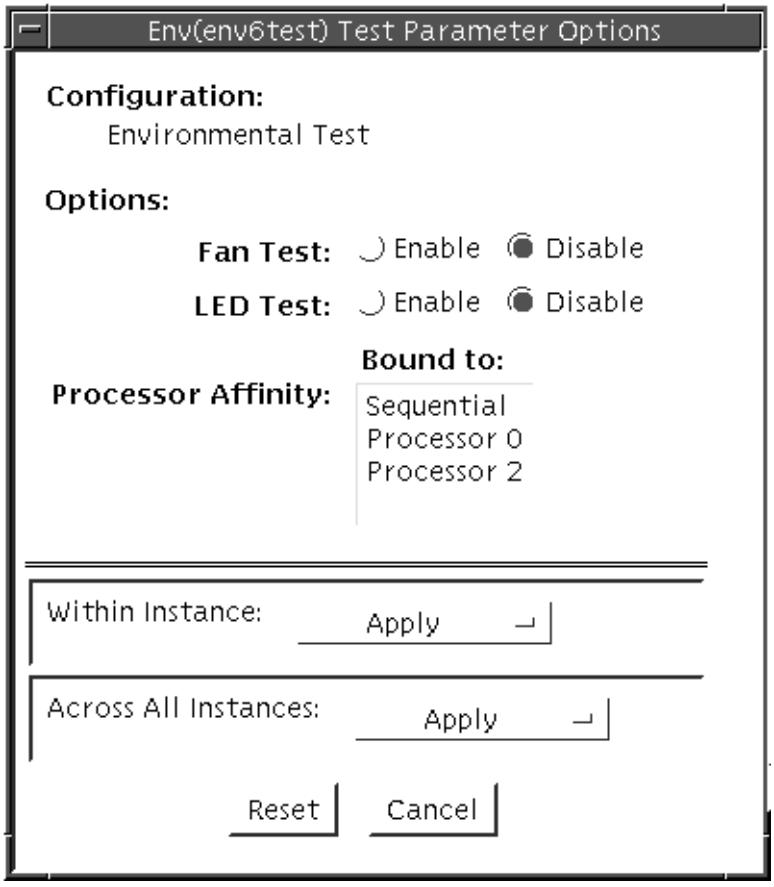

FIGURE 21-1 `env6test` Test Parameter Options Dialog Box

TABLE 21-1 `env6test` Options

`env6test` **Options**	**Description**
Fan Test	Checks the status, tolerance, and speed of the system's fans. Default is Disable.
LED Test	Checks overall status of system's LEDs by switching them ON and OFF. Default is Disable.

env6test Test Modes

TABLE 21-2 env6test Supported Test Modes

Test Mode	Description
Connection	Reports current state of devices.
Exclusive	Performs all tests including the Fan and LED subtests if they are enabled.

env6test Command-Line Syntax

/opt/SUNWvts/bin/env6test *standard_arguments*
-o dev=*raw_device_name*,**led=***Enable* | *Disable*,**fan=***Enable* | *Disable*

TABLE 21-3 env6test Command-Line Syntax

Argument	Description
dev=*raw_device_name*	Specifies the name of the raw device to test. Default is /dev/env
led=*Enable*	*Disable*	Enables or disables the LEDS subtest. Default is Disable.
fan=*Enable*	*Disable*	Enables or disables the Fans subtest. Default is Disable.

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 22

I2C Inter-Integrated Circuit Test (`i2c2test`)

The `i2c2test` is designed to verify the proper placement, operation, and data integrity on the various I2C devices.

This test is not scalable.

`i2c2test` 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.

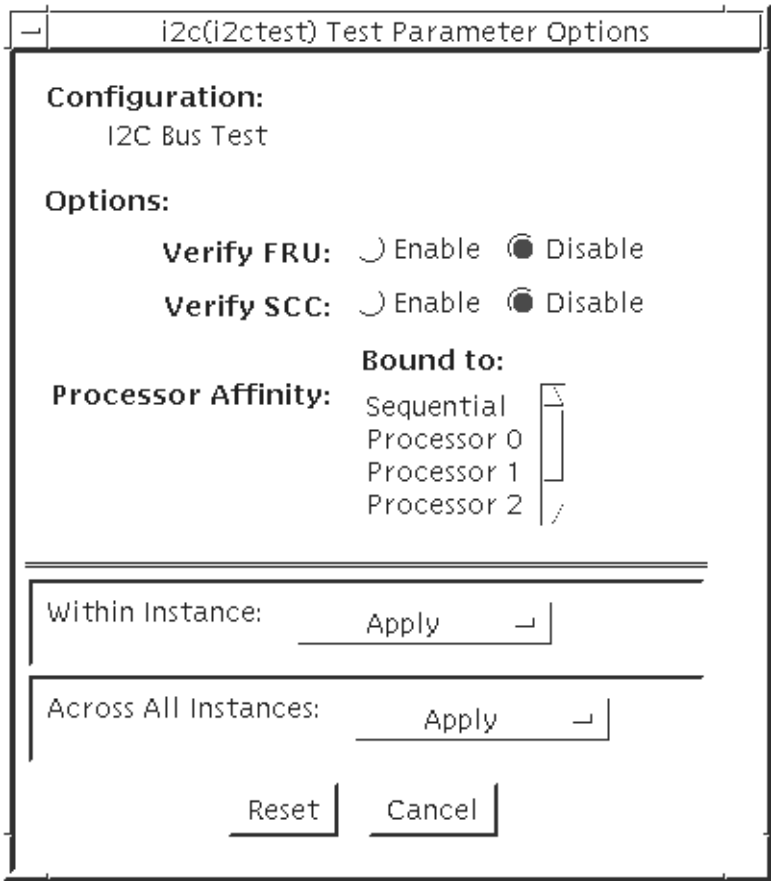

FIGURE 22-1 `i2c2test` Test Parameter Options Dialog Box

TABLE 22-1 `i2c2test` Options

`i2c2test` **Options**	**Description**
Verify FRU	Verifies the status of the FRU. Default is Disable.
Verify SCC	Verifies the status of the SCC. Default is Disable.
Processor Affinity	Specifies the processors to be tested in sequential order.

i2c2test Test Modes

TABLE 22-2 i2c2test Supported Test Modes

Test Mode	Description
Connection	Performs a test to verify connection to all I2C devices.
Exclusive	Performs a test to verify connection to all I2C devices, and also performs a test to verify that the `fru` and `scc` content is consistent with the user's selection.

i2c2test Command-Line Syntax

/opt/SUNWvts/bin/i2c2test *standard_arguments*
-o dev=*raw_device_name*,**chkfru**=*Enable* | *Disable*,**chkscc**=*Enable* | *Disable*

TABLE 22-3 i2c2test Command-Line Syntax

Argument	Description
dev=*raw_device_name*	Specifies the name of the raw device to test.
chkfru=*Enable*	*Disable*	Verifies the status of the FRU. Default is Disable.
chkscc=*Enable*	*Disable*	Verifies the status of the SCC. Default is Disable.

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 "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

CHAPTER 23

Physical Memory Test (pmemtest)

The pmemtest checks the physical memory of the system and reports hard and soft error correction code (ECC) errors, memory read errors, and addressing problems. The pseudo driver mem is used to read the physical memory.

This test reads through all the available physical memory. It does not write to any physical memory location.

pmemtest Options

pmemtest is supported both in physical mapping and logical mapping displays in the UI. In physical mapping, pmemtest provides support to test the memory on a per-board basis; users can select the pmemtest which is displayed under the physical memory board, which is to be tested and test only that board. In logical mapping, the pmemtest options apply to the complete memory across the boards.

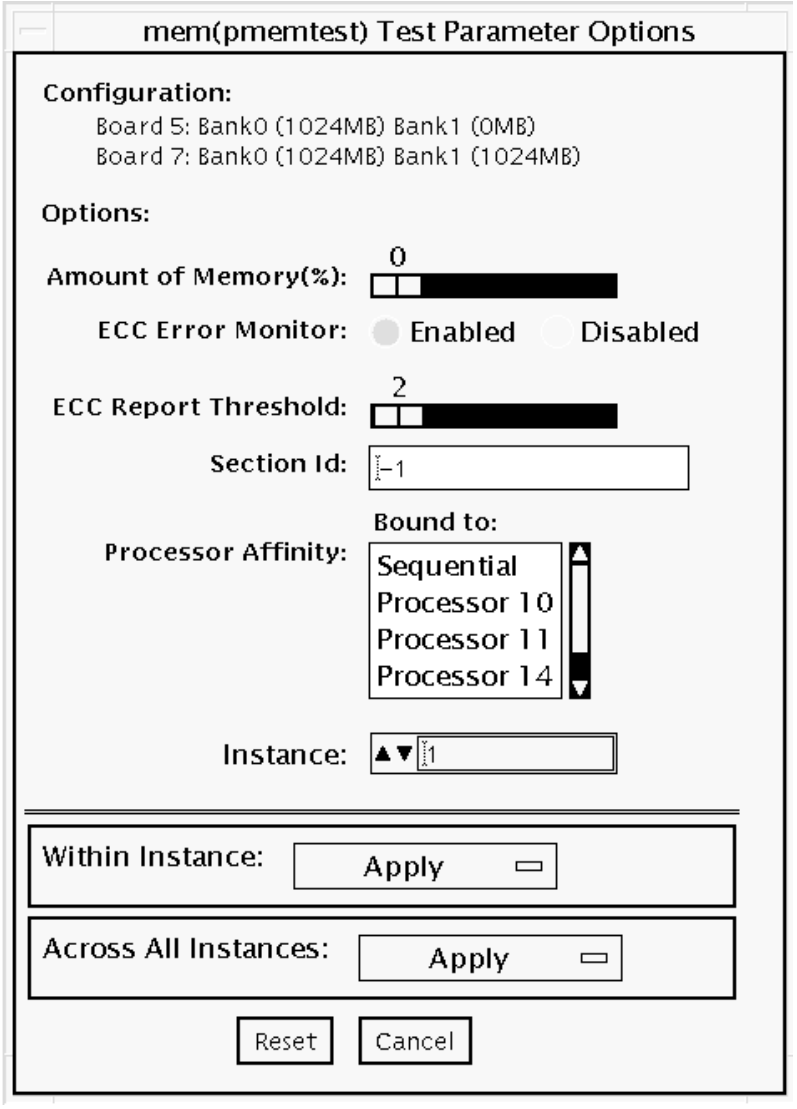

FIGURE 23-1 `pmemtest` Test Parameter Options Dialog Box

TABLE 23-1 `pmemtest` Options

`pmemtest` **Options**	**Description**
Configuration	Shows the total amount of physical memory, rounded up to the nearest megabyte, probed by the SunVTS kernel.
Amount of Memory	Specifies the percentage of the physical memory to be tested. The default 0% ensures dividing the total memory equally among instances which results in 100% coverage at the completion of every test pass. Note that one test pass includes one pass each by all instances.
ECC Error Monitor	This option is used to enable or disable ECC error monitoring.
ECC Report Threshold	Determines how many correctable ECC errors occurred in the elapsed time before pmemtest reports a test failure. A value of zero results in no report of any correctable ECC errors. The default is 2. This option is only available on UltraSPARC systems.
Section ID	When set to -1, pmemtest will test one memory section in each pass, automatically testing each subsequent memory section as testing progresses. When set to a number other than -1, only the section specified will be tested. A section is defined by the pass and instance number settings. This option is only available on UltraSPARC systems.
Instance	Instances are the number of copies of pmemtest to run simultaneously on the memory being tested.

Note – The amount of memory option is specified on a “per instance” basis. The real memory coverage for one test pass depends on the amount of memory option and the number of instances. For example, if there are four instances, and each instance specifies “50%” for the amount of memory option, then this will result in “200%” (4 times 50%) coverage on each test pass. For guaranteed 100% memory coverage for each test pass, choose default percentage size option as 0% for all instances.

pmemtest Test Modes

TABLE 23-2 pmemtest Supported Test Modes

Test Mode	Description
Connection Test	In this mode, one percent of the memory is read. pmemtest also informs the user how much physical memory is available. For sun4m, sun4u, and UltraSPARC servers, the test reports the ECC errors that have occurred since it was last invoked. The test reports ECC errors for a particular CPU or memory board when physical mapping is selected, otherwise it provides the SIMM number of the ECC memory error.
Functional (Offline)	In Functional test mode, the amount of memory to be read can vary. By default 100% of the memory is tested. Also for UltraSPARC servers, this test mode reports the ECC errors that have occurred since it was last invoked. The test reports ECC errors for a particular memory board when physical mapping is selected, otherwise it provides the SIMM number of the ECC memory error.
Online	In this mode too, the amount of memory to be read can vary. By default 100% of the memory is tested. Also for UltraSPARC servers, this test mode reports the ECC errors that have occurred since it was last invoked. The test reports ECC errors for a particular memory board when physical mapping is selected, otherwise it provides the SIMM number of the ECC memory error.

pmemtest Command-Line Syntax

For 32-bit configurations:

/opt/SUNWvts/bin/pmemtest *standard_arguments* **-o size=[0-100], dev=** *device_name,* **eccmon=***Enabled* | *Disabled*, **threshold=***report_threshold*, **bdinfo=** *number*, **section=***section_id*

For 64-bit configurations:

/opt/SUNWvts/bin/sparcv9/pmemtest *standard_arguments* **-o size=[0-100], dev=***device_name*, **eccmon=***Enabled* | *Disabled*, **threshold=** *report_threshold*, **bdinfo=***number*, **section=***section_id*

TABLE 23-3 `pmemtest` Command-Line Syntax

Argument	Description
`size=[0-100]`	Specifies the percentage of memory to be tested. The default is 0% (for 100% memory coverage).
`dev=`*device_name*	Specifies the device to test, for example, `mem`.
`eccmon=`*Enabled* | *Disabled*	ECC error monitoring is enabled or disabled.
`threshold=` *report_threshold*	Determines how many correctable ECC errors occur before they are reported as an error causing `pmemtest` to report a failure. A value of zero results in no report of any correctable ECC errors. The default is 2. This option is only available on UltraSPARC systems.
`bdinfo=`*number*	For UltraSPARC servers, this argument indicates board number information. For example, if board 0 and board 5 have memory and you want the test to read the memory on both boards, then this argument should read `bdinfo`=33 (2**5+2**0). The `bdinfo` value can be specified as 0 to test the memory present on all boards.
`section=`*section_id*	When set to `-1`, `pmemtest` will test one memory section in each pass, automatically testing each subsequent memory section as testing progresses. When set to a number other than `-1`, only the section specified will be tested. A section is defined by the pass and instance number settings. This option is only available on UltraSPARC systems.

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 24

Virtual Memory Test (vmemtest)

The vmemtest checks virtual memory; that is, it tests the combination of physical memory and the swap partitions of the disk(s).

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

This test uses the Solaris valloc (page aligned) system call to allocate, write, read, and compare virtual memory. These operations normally cause heavy paging activity on the system and simulate a stressful environment for the operating system. This test also detects ECC parity errors, memory read errors, and addressing problems, and displays the corresponding virtual memory addresses on failure.

Note – Do not run the vmemtest with fwcamtest at the same time on any Sun Blade™ system. This will cause the test to fail.

vmemtest Swap Space Requirements

Running this test places a significant burden on the operating system, since it uses the majority of swap space available for testing. You should use the vmemtest swap space reserve option when non-SunVTS test processes are started after SunVTS testing has started. See "Swap Space Requirements" in the *SunVTS User's Guide* for a complete discussion of swap space requirements.

`vmemtest` 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.

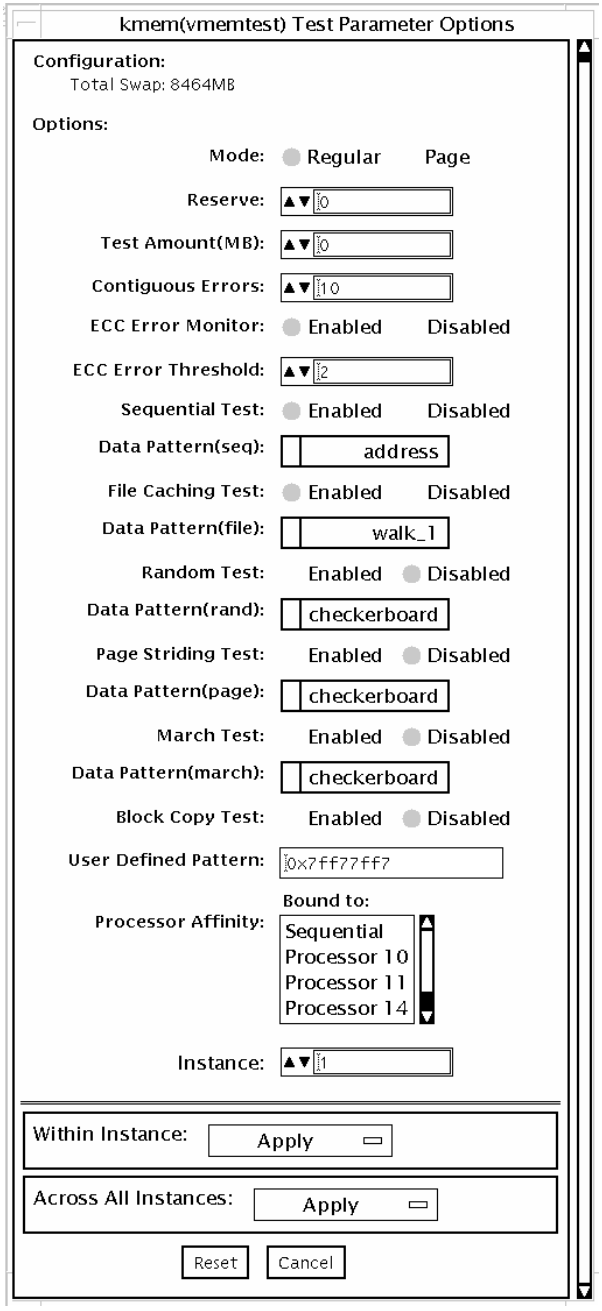

FIGURE 24-1 `vmemtest` Test Parameter Options Dialog Box

TABLE 24-1 `vmemtest` Options

vmemtest Options	Description
Mode	Two modes are available: • Regular mode tests the specified amount of memory as one chunk and passed as the size arugment to the different test algorithm functions (subtests). • Page mode tests assign virtual memory one page at a time. Each page is mapped to the temporary file /tmp/vmem.page and is then paged out to storage once test data is written. Next, the temporary page is paged back into memory for a read and compare. vmemtest runs in Regular mode as default setting
Reserve	The Reserve option specifies the amount of memory to reserve from being tested by vmemtest. The test ensures this amount of memory is left free on the system while evaluating the size of memory for testing. If specified value of reserve is Zero, test will use a default value evaluated based on the available free swap space for the instance. Trying to reserve more memory than available free memory by this instance will cause the test to fail.
Test Amount	An amount can be specified to test the virtual memory, instead of the default.The default value is 0, which means the default memory size is evaluated within the test. It is desirable to the user to know the memory configuration details on the target system while choosing Non default setting for "amount" option. If negetive values are specified, test will assume default setting while it runs. The actual size of memory tested by the instance is always evaluated with reference to the available free swap space on the system.
`vmemtest` Configuration	The amount of memory listed in the Configuration field is equivalent to the sum of the used and available swap space amounts returned by the swap -s command. It indicates the amount of virtual memory found, rounded up to the nearest Kbyte.
Contiguous Errors	Specifies the max. number of contiguous memory errors, which will be considered and counted as one non contiguous error. The default value is 10.
ECC Error monitoring	Enables or disables ECC error monitoring.
ECC Error threshold	Determines how many correctable ECC errors occurred in the elapsed time before vmemtest reports a test failure. The default threshold value is 2.

TABLE 24-1 `vmemtest` Options *(Continued)*

vmemtest Options	Description
Test Method	`vmemtest` runs the Sequential and File Caching subtests by default. • Sequential subtest – The whole memory is tested from the beginning address to the end address in a sequence. • Address Random subtest – Randomly selects memory addresses with in the specified range to test. • Page Striding subtest – Non-contiguous memory test, implemented sequentially and non-sequentially. • Sequential striding – Tests from the first page to the last page, withn a specified test range. Only one word is tested per page. • Non-sequential striding – Tests randomly from first to last page, within a specified memory range. Goes back and forth testing one word per page until all pages are tested. • Block Copy test – Writes and reads data between two memory blocks. Each memory block is half the memory to be tested. • File Caching test – Aimed at improving performance through the use of file caching in the Solaris kernel. This test is useful for large memory configurations. This test takes 30 to 70% less time than the Sequential test method.
Predefined Pattern	• Select one of the following patterns to use for the test: • Address--uses the virtual addresses of the tested memory locations. • walk_1--uses a pattern that starts with 0x80000000 through 0x11111111 • walk_0--uses a pattern that starts with 0x7fffffff through 0x00000000 • 0x00000000--uses all ones and zeros for testing • 0x5aa55aa5--uses 0x5aa55aa5 pattern • 0xdb6db6db--uses 0xdb6db6db pattern • Checkerboard--uses 0xaaaaaaaa patterns. • UserDefined--uses the pattern that is specified in the User Defined Pattern area (see below).
User Defined Pattern	Only used if the Predefined Pattern is set to UserDefined. The pattern specified should be in the form of an 8-digit, hexadecimal number such as 0x2a341234. Default setting is up= 0x7ff77ff7
Instance	Specifies how many copies of the vmemtest test to run.

vmemtest Test Modes

TABLE 24-2 vmemtest Supported Test Modes

Test Mode	Description
Functional	Runs the full set of tests.

In Functional test mode, vmemtest writes a pattern to an amount of virtual memory specified by the user. Then the data is read back and compared. If there is a miscompare, the data is read again and compared. Whenever there is a miscompare, the virtual address is reported.

vmemtest Command-Line Syntax

/opt/SUNWvts/bin/vmemtest *standard_arguments* **-o mode=***type*, **reserve=** *n*, **amount=***n*, **cerr=***n*, **eccmon=***Enabled* | *Disabled*, **eccthreshold=***n*, **type1=** *Enable* | *Disable*, **pp1=***pattern*, **type2=***Enable* | *Disable*, **pp2=***pattern*, **type3=***n*, **pp3=** *pattern*, **type4=***Enable* | *Disable*, **pp4=***pattern*, **type5=***Enable* | *Disable*, **pp5=***pattern*, **type6=***Enable* | *Disable*, **up=***hex_pattern*

TABLE 24-3 vmemtest Command-Line Syntax

Argument	Explanation
mode=*Page* | *Regular*	Specifies which mode of the vmemtest to run. Choose: • Page—tells the write/read memory test to proceed one system memory page at a time. • Regular—uses the valloc option to allocate the entire assigned memory, which is read and compared one long word at a time.
reserve=*n*	Specifies the amount of MB of virtual memory to reserve.
amount=*n*	Specifies the number of MB of memory to be tested instead of the default.
cerr=*n*	Specifies the maximum number of contiguous errors to be counted as one non contiguous error.
eccmon=*Enabled* | *Disabled*	Enables or disables the ECC error monitor.
eccthreshold=*n*	Specifies how many correctable ECC errors can occur in the elapsed time before vmemtest reports a test failure.

TABLE 24-3 `vmemtest` Command-Line Syntax

Argument	*(Continued)*Explanation
`type1=`*value* **`pp1=`***pattern*	`type1` is sequential test. The value is Enabled or Disabled; the default is Enabled. The default for the `pp1` pattern is `address`; select the `pp1` pattern from: **`address`**,**`walk_0`**,**`walk_1`**,**`Checkerboard`**, **`0x00000000`**,**`0xffffffff`**,**`0x5aa55aa5`**, **`0xdb6db6db`**,**`random`**,*UserDefined*
`type2=`*value* **`pp1=`***pattern*	`type2` is File cache test. The value is Enabled or Disabled; the default is Enabled. The default for the `pp1` pattern is `address`; select the `pp1` pattern from: **`address`**,**`walk_0`**,**`walk_1`**,**`Checkerboard`**, **`0x00000000`**,**`0xffffffff`**,**`0x5aa55aa5`**,**`0xdb6db6db`**, **`random`**,*UserDefined*
`type3=`*value* **`pp3=`***pattern*	`type3` is Random address test. The value is Enabled or Disabled; the default is Disabled. The default of the `pp3` pattern is `checkerboard`; select the `pp3` pattern from: **`Checkerboard`**,**`0x00000000`**,**`0xffffffff`**, **`0x5aa55aa5`**,**`0xdb6db6db`**,*UserDefined*
`type4=`*value* **`pp4=`***pattern*	`type4` is `page_striding` test. The value is Enabled or Disabled; the default is Disabled. The default of the `pp4` pattern is `checkerboard`; select the `pp4` pattern from: **`Checkerboard`**,**`0x00000000`**,**`0xffffffff`**, **`0x5aa55aa5`**,**`0xdb6db6db`**,*UserDefined*
`type5=`*value* **`pp5=`***pattern*	`type5` is `march_c` test. The value is Enabled or Disabled; the default is Disabled. The default for the `pp5` pattern is `checkerboard`; select the `pp5` pattern from: **`Checkerboard`**,**`0x00000000`**,**`0xffffffff`**, **`0x5aa55aa5`**,**`0xdb6db6db`**,*UserDefined*
`type6=`*value*	`type6` is `Block_Copy` test. The value is Enabled or Disabled; the default is Disabled. Note – The `Block_Copy` subtest uses its own set of the data patterns predefined in the test. It does not require any user specified data patterns for testing.
`up=`*hex_address*	Only used if the `pp` argument is set to `UserDefined`. The pattern specified should be in the form of a 8-digit, hexadecimal number such as `0x2a341234`.

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 25

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.

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:

- Linpack 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.

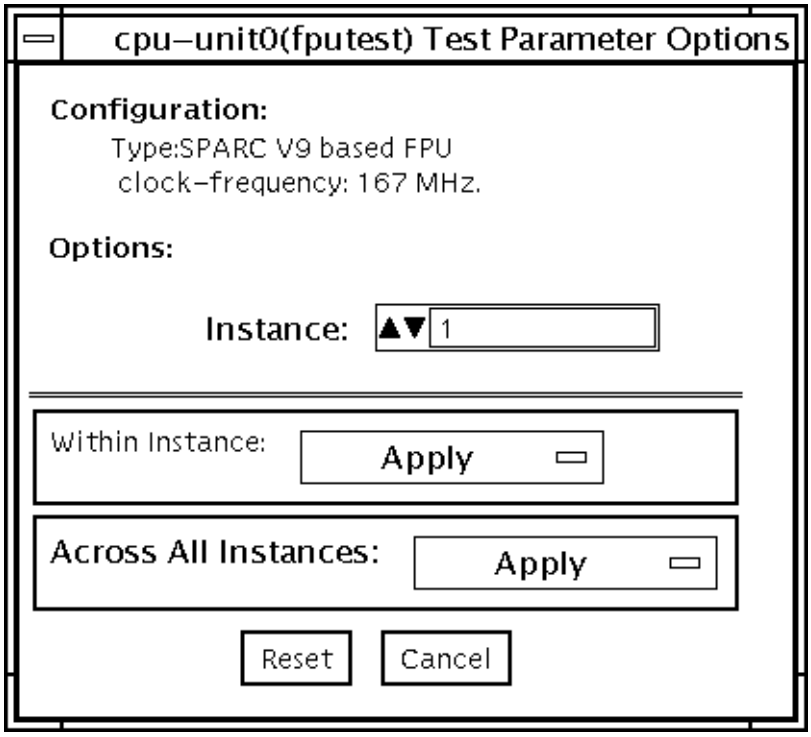

FIGURE 25-1 `fputest` Test Parameter Options Dialog Box

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 25-1 fputest Supported Test Modes

Test Mode	Description
Connection	Includes all the instruction tests.
Functional (Offline)	Performs all the instruction tests and all the benchmark tests.
Stress mode	Performs several fpu benchmark tests.

fputest Command-Line Syntax

/opt/SUNWvts/bin/fputest *standard_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 "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

CHAPTER 26

System Test (`systest`)

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

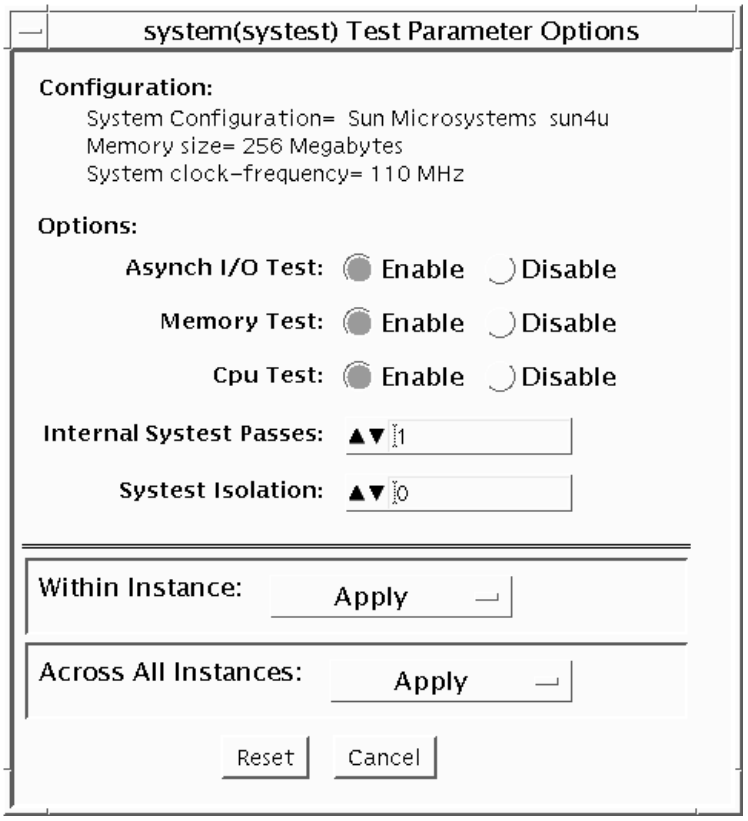

FIGURE 26-1 `systest` Test Parameter Options Dialog Box

Caution – Use discretion when defining the syspass parameter. One linpack 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 26-1 `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 linpack 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 26-2 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 26-3 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.
dev=*system*	Specifies the pseudo device name.
syspass=*1,2000*	Defines the number of internal linpack 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 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).

Recommended Option Selection

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

Command-Line Examples

The following examples assume the user wants 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
- One internal pass of linpack 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
- One internal pass of linpack 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
- Declares a set of boards free from residual errors after 15 internal passes of the linpack 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
- Declares a set of boards AND CPUs free from residual errors after 10 internal passes of the linpack algorithm
- If an error is found, `systest` will perform boards AND CPUs isolation

CHAPTER 27

Integer Unit Test (iutest)

The Integer Unit Test (iutest) tests the resident integer unit in Sun SPARC CPUs. It exercises all of the register windows present in the Integer Unit of the CPU. The successful completion of the test implies that all of the register windows are functioning properly and failure implies a faulty register.

iutest 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.

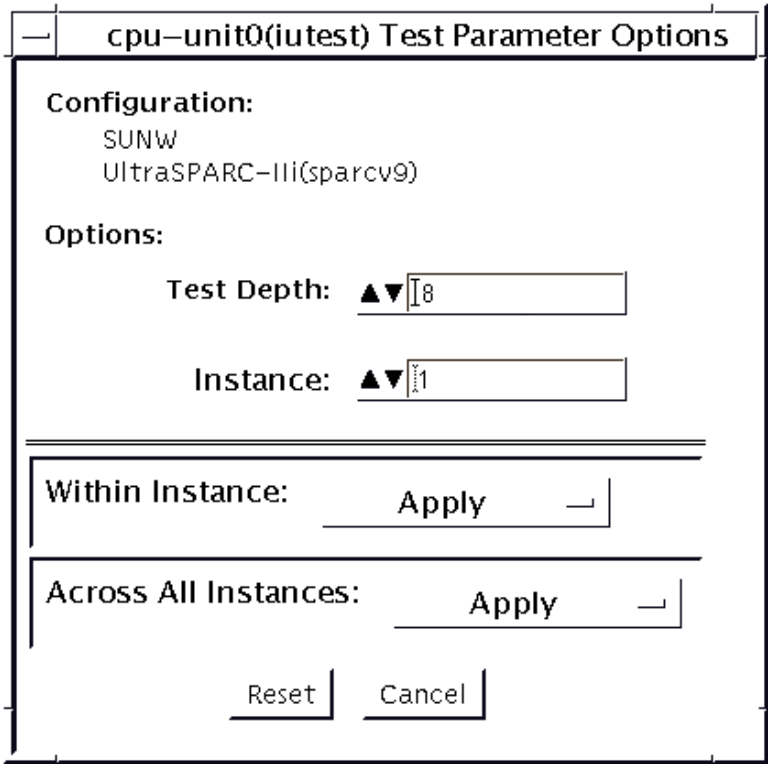

FIGURE 27-1 `iutest` Test Parameter Options Dialog Box

For the test options in the `iutest` Test Parameter Options dialog box, Test Depth is the only required option. Test Depth corresponds to the number of times that *all the register windows* are getting tested. The default, maximum and minimum values of the Test Depth are 8, 64, and 1 respectively.

iutest Test Modes

TABLE 27-1 iutest Supported Test Modes

Test Mode	Description
Connection	Displays the type of CPU implementation (for example, sparcv7 or sparcv9, etc.), the operating frequency, and CPU status (online, offline, etc.).
Functional (Offline)	Verifies all of the register windows and returns the appropriate error message if there is a faulty register. Otherwise, displays a successful test message.

iutest Command-Line Syntax

/opt/SUNWvts/bin/iutest *standard_arguments* **-o depth=***val*,**dev=cpu-unit***N*

In the iutest command-line syntax, *val* is the value of the Test_Depth parameter option as described in the preceding iutest options section. *N* is the CPU unit number (0,1,2, etc.). The test behavior is unpredictable if options other than those described in this section are entered.

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).
