![Sun Microsystems logo]

SunVTS™ 6.4 Patch Set 3 Documentation Supplement for x86 Platforms

Please
Recycle

Adobe PostScript™

Contents

Preface

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

This document is a supplement to the *SunVTS 6.4 Test Reference Manual for x86 Platforms* (820-1671) and the *SunVTS 6.4 User's Guide* (820-1669). Refer to these documents for details on the full collection of SunVTS 6.4 tests and software features that are not new or enhanced in this release.

Note – The Solaris™ Operating System (OS) release with which this version of SunVTS is delivered supports systems that use the SPARC® and some x86 processor architectures. This document cites any implementation differences between the platform types.
In this document, the term x86 refers to 64-bit and 32-bit systems. These systems are manufactured using processors compatible with the AMD64 or Intel Xeon/Pentium product families.

Before You Read This Book

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

- *SunVTS 6.4 User's Guide*
- *SunVTS 6.4 Test Reference Manual for x86 Platforms*
- *SunVTS Quick Reference Card*

How This Book Is Organized

This book is organized as follows:

Chapter 1 describes SunVTS requirements, test modes, user interfaces, the collection of tests, and how to run a test from the command line. The remaining chapters are actual SunVTS test reference manual chapters for tests with changes or enhancements for this release.

Using UNIX Commands

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

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

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

  http://docs.sun.com

Shell Prompts

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

Typographic Conventions

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

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

Related Documentation

This document covers SunVTS version 6.4 tests. It serves as a reference companion to the SunVTS documents listed below.

Application	Title	Part Number
Installation and Navigation	*SunVTS 6.4 User's Guide*	820-1669-10
Test Specific Reference	*SunVTS 6.4 Test Reference Manual for x86 Platforms*	820-1671-10
Quick Reference Card	*SunVTS Quick Reference Card*	820-1672-10

Documentation, Support, and Training

Sun Function	URL
Documentation	http://www.sun.com/documentation/
Support	http://www.sun.com/support/
Training	http://www.sun.com/training/

Third-Party Web Sites

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

Sun Welcomes Your Comments

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

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

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

SunVTS 6.4 Patch Set 3 Documentation Supplement for x86 Platforms,
part number 820-4704-10

SunVTS Overview

This chapter contains the following topics:

-
-
-
-

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

SunVTS contains 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.

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

Note – 64-bit tests are located in this directory: `/opt/SUNWvts/bin/64/`, or the relative path to which you installed SunVTS. If a test is not present in this directory, then it might be available as a 32-bit test only. For more information, see "32-Bit and 64-Bit Tests" on page 4.

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

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

SunVTS Version Information

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

x86 Platform Support

Starting with Solaris 10, the SunVTS infrastructure and core diagnostics are available for x86 Solaris platforms.

SunVTS is supported and tested on the following Sun x86 platforms:

- Sun Fire V20z
- Sun Fire V40z
- Sun Fire B100
- Sun Fire B200
- Sun Fire x4100
- Sun Fire x4100 M2
- Sun Fire x4200
- Sun Fire x4200 M2
- Sun Fire x4500
- Sun Fire x4600
- Sun Fire x4600 M2
- Sun Fire x6220
- Sun Blade x8400
- Sun Blade 6000 Chassis
- Sun Blade 6048 Chassis
- Sun Blade x6220 Server Module
- Netra CP3020

Note – If you perform SunVTS on an unsupported platform, a warning message appears and SunVTS stops.

You must install the x86 version of the SunVTS packages to perform SunVTS on x86 platforms. The software packages use the same names as in the SPARC environment. The SunVTS packages delivered separately for both x86 and SPARC Solaris platforms are as follows:

- `SUNWvts` – Contains the SunVTS core framework that includes the kernel and user interface.
- `SUNWvtsmn` – Contains the SunVTS online manual pages
- `SUNWvtsr` – Contains the SunVTS framework configuration files in the root partition (Superuser).

- `SUNWvtsts` – Contains the SunVTS test binaries.

The SunVTS components available for x86 Solaris platforms are as follows.

Infrastructure:

- sunvts
- vtsk
- vts_cmd
- vtstty
- vtsui
- vtsprobe

SunVTS Tests:

- BMC Environment Test (`bmcenvironment`)
- Optical Disk Drive Test (`cddvdtest`)
- CPU Test (`cputest`)
- Cryptographics Test (`cryptotest`)
- Disk and Floppy Drives Test (`disktest`)
- Data Translation Look-aside Buffer Test (`dtlbtest`)
- Emulex HBA Test (`emlxtest`)
- InfiniBand Host Channel Adapter Test (`ibhcatest`)
- Level 1 Data Cache Test (`l1dcachetest`)
- Level 2 SRAM Test (`l2sramtest`)
- Ethernet Loopback Test (`netlbtest`)
- Network Hardware Test (`nettest`)
- Physical Memory Test (`pmemtest`)
- Qlogic Host Bus Adapter Test (`qlctest`)
- RAM Test (`ramtest`)
- Serial Ports Test (`serialtest`)
- System Test (`systest`)
- Tape Test (`tapetest`)
- Virtual Memory Test (`vmemtest`)

Test Requirements

SunVTS Version 6.3 was first introduced and designed to run in the Solaris 10 6/06 (Solaris 10 Update 2) operating system (OS) and subsequent compatible releases. The operating system kernel must be configured to support all peripherals that are to be tested.

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

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.

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/64/`*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 – For 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 10, you can use the following command to identify the application support of your system.

```
# isainfo -v
```

SunVTS User Interfaces

You can run SunVTS tests from multiple interfaces: The 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 8). 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 graphical user interface 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.
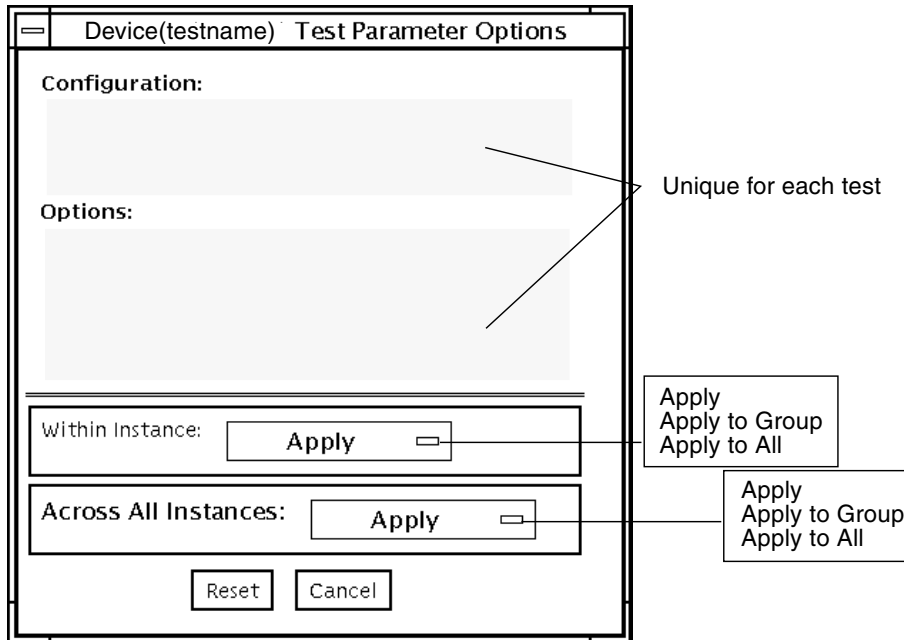
FIGURE 1-1   Test Parameter Options Dialog Box

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.
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*]

Standard Command-Line Arguments

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

TABLE 1-3 Standard Command-Line Arguments

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

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

Argument	Description
–p *number*	Defines the number of passes for scalable tests. The default is 1.
–i *number*	Defines the number of instances for scalable tests. The default is 1.
–w *number*	Defines to which instance the test is assigned; this option is for scalable tests. The default is 0.
-o	Indicates that the options and arguments that follow are test specific.

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

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

Test-Specific Arguments

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=***device-name***,count=***count-number***,test=***testlist***,size=***file-size***,retain=***mode* The test option format is specified by the man page getsubopt(3C).

CPU Test (`cputest`)

Starting from 6.3ps1, the `cputest` registers only one instance for the whole system. The parameter `dev`, which used to take option of the form `dev=cpu-unitx`, is enhanced to support testing multiple CPUs present on the system. The option `dev` now takes the following syntax:

`dev=all|All|cpu-unit0+cpu-unit1+.....+cpu-unitN`

The `dev` option also supports either "all" or "All" as a valid value in which case all the CPUs in the system can be tested.

Integer Unit Test (`iutest`)

Starting from 6.3ps1, the `iutest` registers only one instance for the whole system. The parameter `dev`, which used to take option of the form `dev=cpu-unitx`, is enhanced to support testing multiple CPUs present on the system. The option `dev` now takes the following syntax:
`dev=all|All|cpu-unit0+cpu-unit1+.....+cpu-unitN`

The `dev` option also supports either "all" or "All" as a valid value in which case all the CPUs in the system can be tested.

Ethernet Loopback Test (`netlbtest`)

This chapter describes the `netlbtest` external loopback test.

- "netlbtest Description" on page 15
- "netlbtest Test Requirements" on page 16
- "netlbtest Options" on page 17
- "netlbtest Test Modes" on page 19
- "netlbtest Command-Line Syntax" on page 20

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

`netlbtest` Description

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

The `netlbtest` uses DLPI RAW mode to talk to the device driver. For the purpose of this test, a packet is defined as an Ethernet header followed by the Ethernet data payload (refer to the IEEE 802.3z standard). The test generates and sends out the desired number of packets (a tunable parameter) and expects to receive the same number of packets through the loopback interface, external or internal. If an error

occurs (for example, packet mismatch or timeout), an error message indicating the type of error, its probable cause(s), and recommended action(s) is displayed on the SunVTS console.

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

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

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

`netlbtest` Test Requirements

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

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

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

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

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

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

`netlbtest` Options

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

ce1(netlbtest) Test Parameter Options

Configuration:
  Port Address: Unknown
  Host ID: 80d88a43
  Domain Name: nspg.sfbay.sun.com

Options:

Total_packets:	▲▼ 1000
Packet_Size:	▲▼ 1000
Time_Out(sec):	▲▼ 10
Loopback:	◉ External ○ Internal
Print_Warning:	○ Enable ◉ Disable
Debug_Timeout:	◉ Enable ○ Disable

Processor Affinity: **Bound to:**
  Sequential
  Processor 2
  Processor 3
  Processor 8

Within Instance: Apply

Across All Instances: Apply

Reset   Cancel

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

Refer to TABLE 4-1 for test parameter descriptions.

TABLE 4-1   `netlbtest` Options

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

`netlbtest` Test Modes

TABLE 4-2   `netlbtest` Supported Test Modes

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

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

`netlbtest` Command-Line Syntax

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

TABLE 4-3   `netlbtest` Command-Line Syntax

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