IRIX® Based Field Diagnostics

SGI Confidential & Proprietary Information - For Internal Recipients Only

Document Number 108-0163-005

CONTRIBUTORS
Revised by Darrin Goss
Edited by Cindi Leiser
Production by Rhonda Kunsman
Engineering contributions by Nadia Pavlova

Contents

Figures

Tables

About This Guide

This document describes the diagnostic tests that you can use while the IRIX operating system is running in Silicon Graphics Crimson, Indy, Indigo, Indigo2, Power Indigo2, Octane, Onyx, and Power Onyx visual workstations and SGI Challenge and Power Challenge servers. These diagnostics run from the IRIX prompt and are called IRIX based field diagnostics.

This document is written for System Support Engineers (SSEs) and Field Engineers (FEs) to use as a reference document in training and on-site.

The information that this document contains is organized into the following chapters:

- Chapter 1, “Introduction to the IRIX Based Field Diagnostics,” introduces the IRIX based diagnostics, describes how to install them, and describes how to start the field_diag user interface.
- Chapter 2, “field_diag User Interface,” describes the user interface that you can use to run the IRIX based diagnostics.
- Chapter 3, “Diagnostic Tests,” describes the individual IRIX based diagnostic tests. It includes instructions on how to run each test from the field_diag user interface and information about the output that each test returns.

This document corresponds to the IRIX based field diagnostics in the *Internal Support Tools 2.4* CD-based diagnostic software release.

Typographical Conventions

This document uses the following typographical conventions:

- Information displayed on the screen is shown in `Courier` type.
- Text that you enter is shown in **`Courier bold`** type.
- Commands in text, filenames, pathnames, and variables are shown in *italic* type.

Chapter 1

Introduction to the IRIX Based Field Diagnostics

This chapter introduces the IRIX based field diagnostics and provides information about installing them.

1.1 About the IRIX Based Field Diagnostics

The IRIX based field diagnostics are diagnostic tests that you can run from an IRIX prompt while user operations continue on Silicon Graphics Crimson, Indy, Indigo, Indigo2, Power Indigo2, Octane, Onyx, and Power Onyx visual workstations and SGI Challenge and Power Challenge servers. There are two components to the IRIX based field diagnostics:

- The field_diag user interface
- The various diagnostic tests

Note: The System Verification Program (SVP) must be installed to run the field diagnostics because the field diagnostics package uses several of the diagnostics that are in SVP.

1.1.1 About the field_diag Program

You use the field_diag ASCII user interface to run any of the IRIX based field diagnostics that are available for the hardware in your system.

Refer to Chapter 2, “field_diag User Interface,” for more information about the user interface for the field_diag program.

Note: This interface is identical to the user interface for the field stress tool (FST). The similarity of the two interfaces provides a common “look and feel” for performing hardware maintenance.

Starting with IRIX operating system release version 6.5.5, the field_diag program logs events in the Embedded Support Partner framework. The field_diag program sends a START event to Embedded Support Partner when a test starts and an END event when a test ends. The Embedded Support Partner database includes information about which test was run, when it started, and the result of the test (pass or fail).

1.1.2 About the Diagnostic Tests

Diagnostic tests are available to test the following hardware components while IRIX is running on a system:

- CPU hardware
- Memory
- Graphics hardware
- Networking hardware
- SCSI hardware

Refer to Chapter 3, "Diagnostic Tests," for more information about the diagnostic tests that you can run from the field_diag program.

1.2 Installing the IRIX Based Field Diagnostics

The Internal Support Tools group releases the IRIX based field diagnostics in the *support* image on the *Internal Support Tools* CDs. Follow the installation instructions in the CD booklet to install the *support* image.

Note: The *Internal Support Tools* CDs also include a *field_diags_irix* image that contains online diagnostics for Origin and Onyx family systems. Refer to *Online Diagnostics*, publication number 108-0286-00x, for more information about these diagnostics.

Any future updates to the IRIX based field diagnostics will be available from the Internal Support Tools Web site (http://ist.csd.sgi.com/) or on future CD releases.

This document corresponds to the IRIX based field diagnostics in the *support* image in the *Internal Support Tools 2.4* CD-based diagnostic software release.

1.3 Summary of Chapter 1

This chapter introduced the IRIX based field diagnostics.

The IRIX based field diagnostics are implemented through two components:

- The field_diag user interface that you can use to load, run, and monitor the diagnostic tests
- The individual diagnostic tests

Chapter 2

field_diag User Interface

This chapter describes the field_diag user interface.

2.1 About the field_diag User Interface

The field_diag user interface provides a common ASCII interface that you can use to run the IRIX based field diagnostics. This interface enables you to perform the following actions:

- Select any available diagnostic test
- Modify any user-adjustable parameters for the selected diagnostic test
- Run the selected diagnostic to test the system
- View the output from the selected diagnostic test

This interface is identical to the user interface for the field stress tool (FST). The similarity of the two interfaces provides a common “look and feel” for diagnosing hardware failures.

2.2 Starting the field_diag User Interface

You must have root privilege to run the field_diag user interface.

Perform the following procedure to start the field_diag user interface:

1. Enter the following command to change to the */usr/diags/diags_interface/* directory:

   `cd /usr/diags/diags_interface`

2. Enter the following command to start the field_diag program:

   `./field_diag`

2.3 field_diag User Interface Components

Figure 2-1 shows the components of the main screen of the field_diag user interface. The text following the figure describes the components.

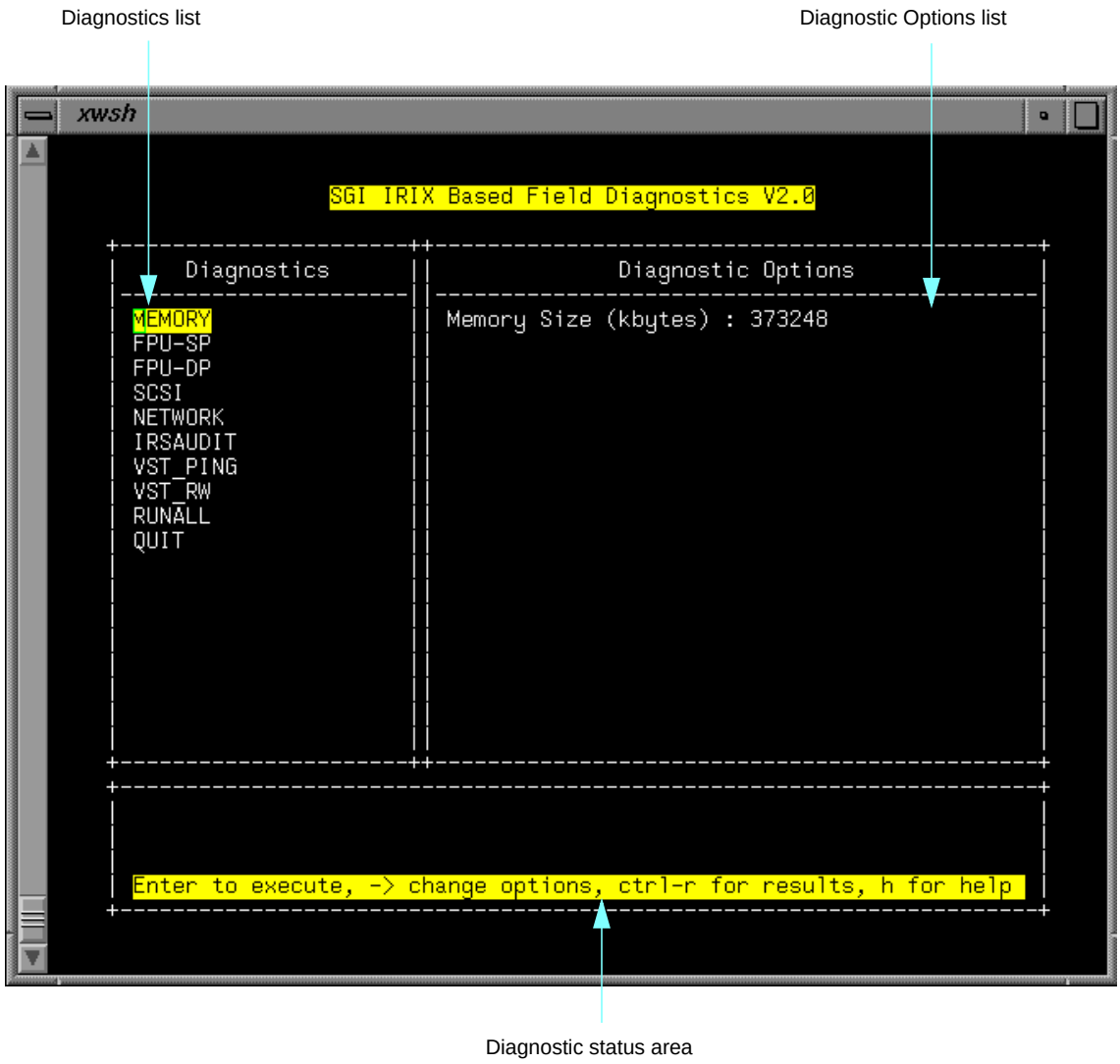

Figure 2-1 field_diag Interface (Main Screen Components)

The main screen includes the following components:

- Diagnostics list
- Diagnostic Options list
- Diagnostic status area
- Test Log Output window

When you type Ctrl+r on the main screen, the Test Log Output window appears. This window displays the output messages from the test that is currently running. Figure 2-2 shows the Test Log Output window.

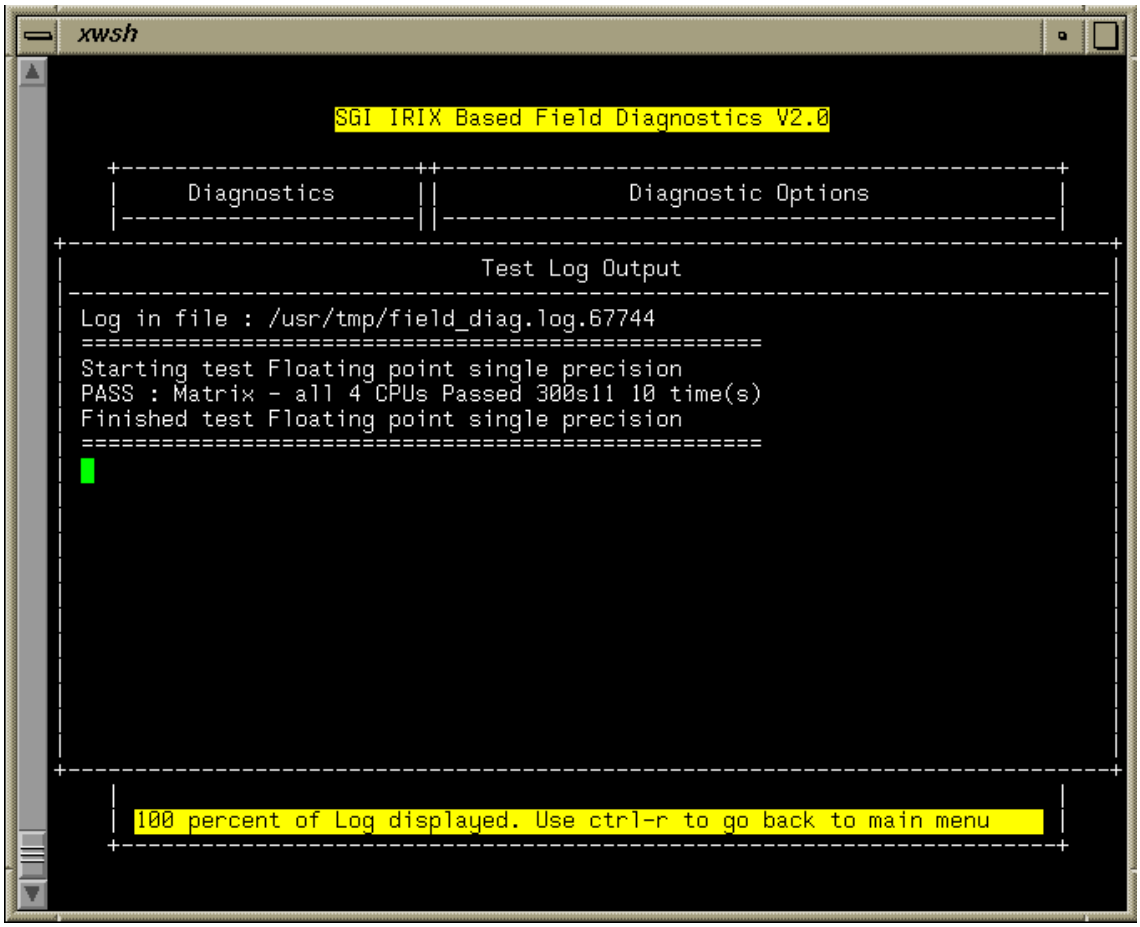

Figure 2-2 field_diag Interface (Test Log Output Window)

2.3.1 Diagnostics List

The Diagnostics list contains the names of all diagnostics that are available for the system configuration that you are using. The field_diag program automatically identifies the hardware in the system and adjusts this list to display only the diagnostics that can test the hardware in the system.

Table 2-1 describes all of the options that are available in the Diagnostics list.

Note: The interface that you use on-site displays only the options that correspond to the hardware in your system.

Table 2-1 Diagnostics List Options

Option	Description
MEMORY	Selects the memory test. Refer to Section 3.2, “Memory Test,” for more information.
FPU-SP	Selects the floating-point unit (single-precision) test. Refer to Section 3.3, “Floating-Point Unit (Single-Precision) Test,” for more information.
FPU-DP	Selects the floating-point unit (double-precision) test. Refer to Section 3.4, “Floating-Point Unit (Double-Precision) Test,” for more information.
SCSI	Selects the SCSI thrasher test. Refer to Section 3.5, “SCSI Thrasher Test,” for more information.
NETWORK	Selects the network thrasher test. Refer to Section 3.6, “Network Thrasher Test,” for more information.
IRSAUDIT	Selects the InfiniteReality graphics test. Refer to Section 3.7, “InfiniteReality Graphics Test,” for more information.
VHT_LOOPBACK	Selects the vicious HIPPI test (loopback test). Refer to Section 3.8, “Vicious HIPPI Tests,” for more information.
VHT_REMOTE	Selects the vicious HIPPI test (remote test). Refer to Section 3.8, “Vicious HIPPI Tests,” for more information.
VST_PING	Selects the vicious socket test (ping test). Refer to Section 3.9, “Vicious Socket Tests,” for more information.
VST_RW	Selects the vicious socket test (read and write test). Refer to Section 3.9, “Vicious Socket Tests,” for more information.
RUNALL	Runs all tests that have default parameter settings. By default, this option skips any tests that have a NEEDED option; however, if you specify values for the NEEDED options for a test before you select this option, this option will also run the test. field_diag stores the results from this option in */usr/tmp/runall_log.pid*, where the *pid* value indicates the process identification value for the process that runs the test.
QUIT	Quits the field_diag program.

Perform any of the following actions to navigate through the Diagnostics list:

- Use the up and down arrow keys to move through the diagnostic tests in the Diagnostics list.
- Use the right arrow key to move from the Diagnostics list into the Diagnostic Options list.

- Use the left arrow key to move from the Diagnostic Options list back into the Diagnostics list.
- Press the Enter key to run the selected test.
- Press the H key to display help messages for the selected test. (Figure 2-3 shows an example of a help message display.)

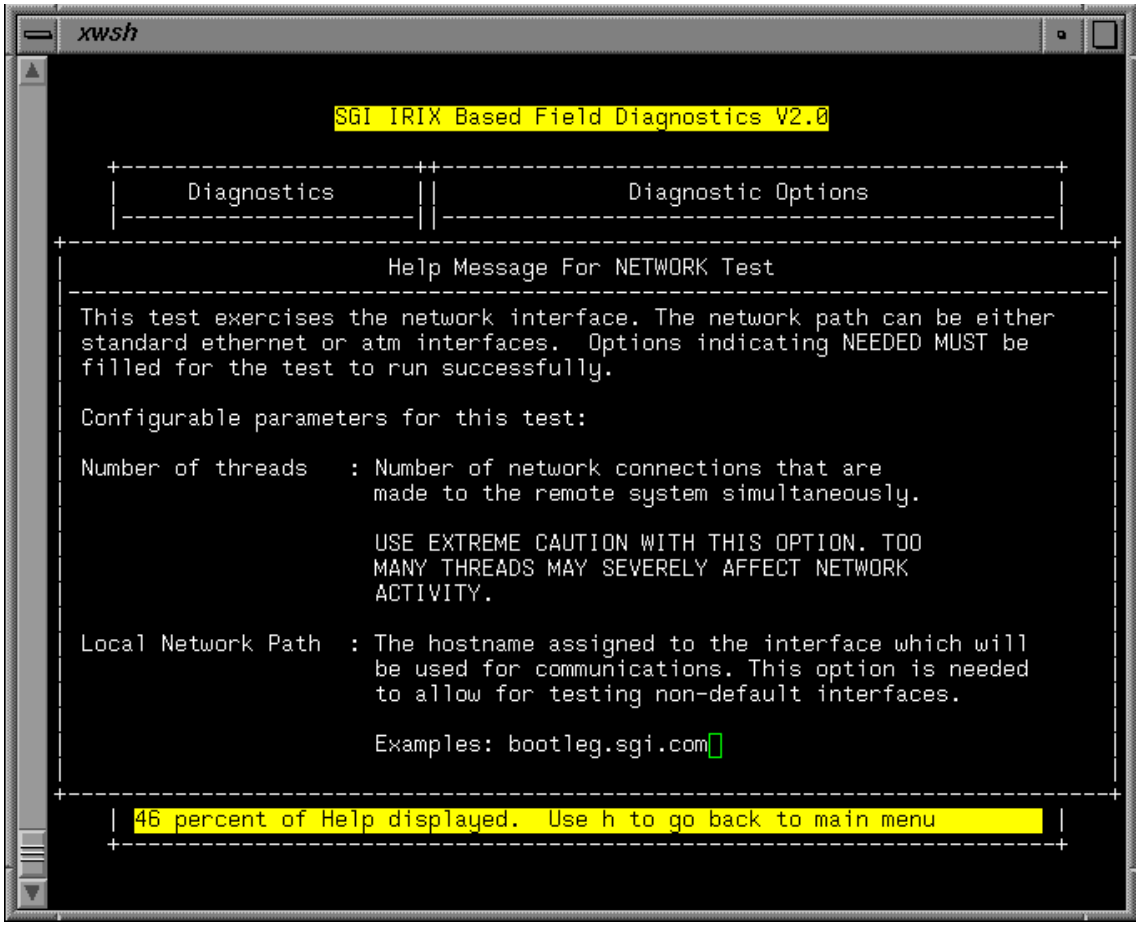

Figure 2-3 Example of Help Message Display

2.3.2 Diagnostic Options List

The Diagnostic Options list shows the parameters that you can adjust for each diagnostic test. If nothing appears in this list, the test does not have user-adjustable parameters.

For example, Figure 2-4 shows that the SCSI thrasher test has two parameters that you can adjust: the number of writes that the test performs and the size of the file that the test writes.

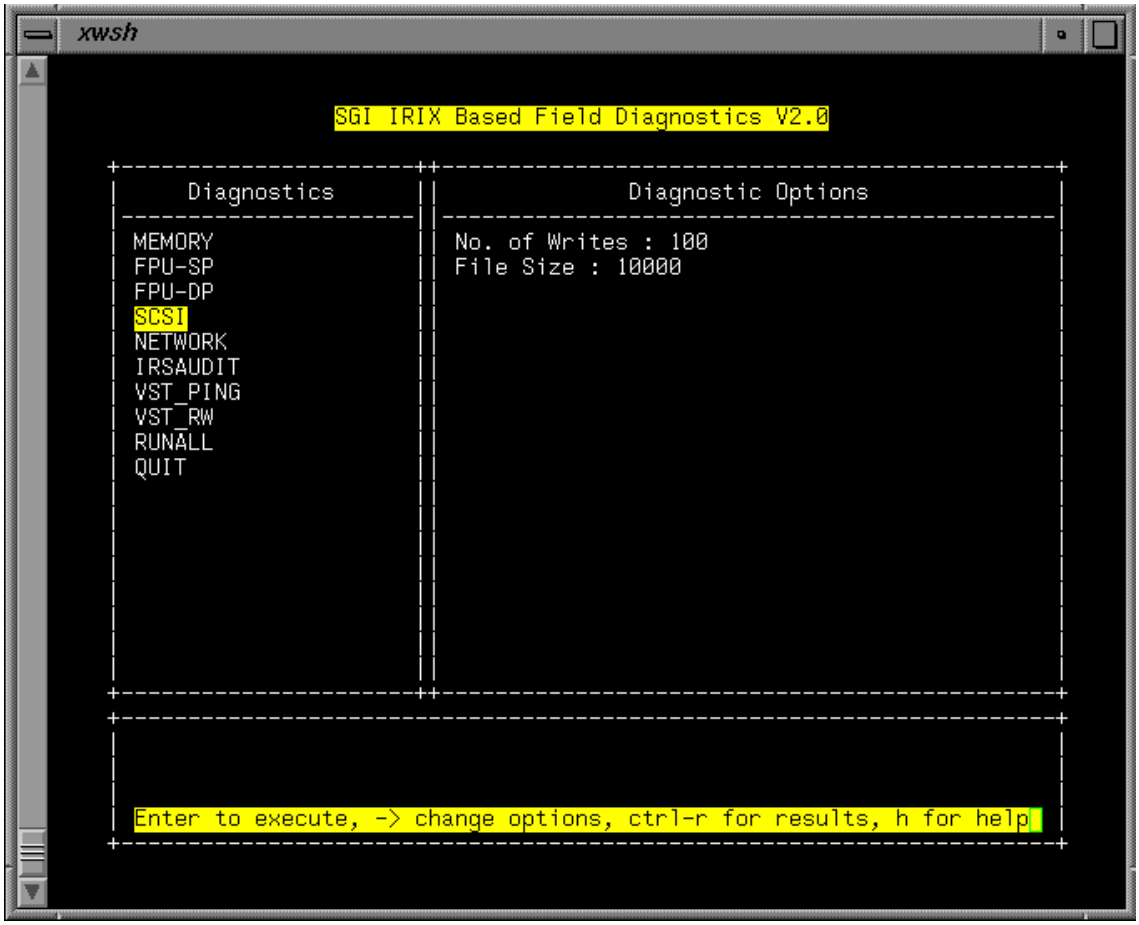

Figure 2-4 Example of Diagnostic Options List

Perform any of the following actions to navigate through the Diagnostic Options list:

- Use the up and down arrow keys to move through the parameters in the Diagnostic Options list.
- Press the Enter key to select the parameter that you want to change. (When the prompt `Enter New Value for the Option` appears in the diagnostic status area, type the new value and press the Enter key.)
- Use the left arrow key to move from the Diagnostic Options list into the Diagnostics list.

2.3.3 Diagnostic Status Area

The diagnostic status area provides information about the current status of the interface and diagnostic test that is running. It displays the following information:

- A prompt that indicates what you can do on the current screen (for example, `Hit enter to execute, Right Arrow to change options`)
- A status line that indicates what action the field_diag program is currently performing (for example, `Loading Diags ....` and `Executing : memory`)
- Status information about the state of the selected test (for example, `memory Test Passed`)

2.3.4 Test Log Output Window

The Test Log Output window displays output from the test that is currently running. This information includes any error messages that the test generates. If the output contains more than one screen of text, use the up and down arrow keys to scroll through the text. (The diagnostic status area on the main screen displays the percentage of the output that you have viewed.)

Enter Ctrl+r to toggle the display of this window to the main window. Refer again to Figure 2-2 for an example of a Test Log Output window.

2.4 Summary of Chapter 2

This chapter described the field_diag user interface.

The field_diag user interface provides a common ASCII interface from which you can run and monitor all IRIX based field diagnostics that are available for the hardware in your system. The interface has four main components:

- The Diagnostics list on the main screen
- The Diagnostic Options list on the main screen
- The diagnostic status area on the main screen
- The Test Log Output window

You can use the interface to perform the following actions:

- To select any available diagnostic test
- To modify any user-adjustable parameters for the selected diagnostic test
- To run the selected diagnostic to test the system
- To view the output from the diagnostic test

Table 2-2 summarizes the keyboard characters that you can use to navigate through the interface.

Table 2-2 field_diag Keyboard Commands

Interface Location	Keyboard Character	Navigation
Diagnostics list	Up arrow	Move up to the previous diagnostic test in the list
	Down arrow	Move down to the next diagnostic test in the list
	Right arrow	Move into the Diagnostic Options list
	Enter	Execute the selected diagnostic test
	H	Display the help information for the current test
Diagnostic Options list	Up arrow	Move up to the previous option in the list
	Down arrow	Move down to the next option in the list
	Left arrow	Move into the Diagnostics list
	Enter	Select the parameter to modify
Anywhere	Ctrl+r	Toggle the display of the Test Log Output window

Chapter 3

Diagnostic Tests

This chapter describes the following diagnostic tests, which you can run from the field_diag user interface:

- Memory test
- Floating-point unit (single-precision) test
- Floating-point unit (double-precision) test
- SCSI thrasher test
- Network thrasher test
- InfiniteReality graphics test
- Vicious socket tests
- Vicious HIPPI tests

Note: This document describes the tests in the order that they appear in the interface.

3.1 Test Availability

The diagnostic tests in the *support* image of the IST 2.4 CD-based diagnostic release are available for IRIX 6.2, IRIX 6.4, or IRIX 6.5 running on one of the systems shown in Table 3-1.

Table 3-1 Diagnostic Test Availability (by Processor Type)

Processor[a]	Memory Test	Floating-point Unit Tests	SCSI Thrasher Test	Network Thrasher Test	InfiniteReality Graphics Test	Vicious HIPPI Tests	Vicious Socket Tests
IP17 (Crimson)		x	x	x			x
IP19 (Challenge and Onyx)	x	x	x	x		x	x
IP20 (Indigo)		x	x	x			x
IP21 (Power Challenge and Power Onyx)	x	x	x	x	x	x	x
IP22 (Indy and Indigo2)		x	x	x			x
IP25 (Power Onyx)	x	x	x	x	x	x	x
IP26 (Power Indigo2)		x	x	x			x
IP28 (Power Indigo2)		x	x	x			x
IP30 (Octane)	x	x	x	x			x

a. The `field_diag` diagnostics are not available for IP32 processors.

To test systems with IP27, IP29, IP31, or IP35 processors; use the online diagnostics in the *field_diags_irix* image. (Refer to *Online Diagnostics*, publication number 108-0286-00x.)

3.2 Memory Test

The memory test checks the memory subsystem by performing sparse writes of data patterns, reading the data back, and verifying the results.

The amount of memory that it tests depends on the limit size that you specify with the `Memory size` parameter. If you set the `Memory size` parameter to `unlimited`, this test attempts to check the maximum memory size of the system. (This test attempts to use 85 percent of the memory that you specify.)

Caution: If you set the `Memory size` parameter to `unlimited` and the system contains a large amount of memory, this test will run for an undetermined length of time.

If this test fails, the failing FRU is a DIMM on the Node board that the test specifies, the Node board, or the midplane.

3.2.1 Prerequisites for Running the Memory Test

This test has the following prerequisites:

- The system that you want to test must have an IP19, IP21, IP25, or IP30 processor.
- The system that you want to test must have IRIX version 6.2, 6.4, or 6.5 booted and running.
- You must run this test from superuser mode.

3.2.2 Running the Memory Test

To run the memory test from the field_diag program, perform the following procedure:

1. Select the MEMORY option from the Diagnostics list.
2. If you want to modify the parameter `Memory Size (kbytes)` (which specifies the amount of memory to test), perform the following actions; otherwise, skip to Step 4.
    - Press the right arrow key to move into the Diagnostic Options list to select the `Memory Size (kbytes)` parameter.
    - Press the **`Enter`** key.
    - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
    - Press the **`Enter`** key.

3. Press the left arrow key to return to the MEMORY option in the Diagnostics list.
4. Press the **Enter** key to start running the memory test.

   The message `Executing : memory` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the memory test.

Note: The System Verification Program (SVP) also runs this test. For more information about SVP, refer to the *System Verification Program (SVP) Reference Guide*, SGI publication number 108-0165-00x.

3.2.3 Output from the Memory Test

This test returns output to the diagnostic status area of the user interface and the Test Log Output window.

3.2.3.1 Pass Output

If this test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `memory Test Passed`.

3.2.3.2 Failure Output

If this test detects a hardware failure or encounters an error that makes the test unable to continue, the diagnostic status area of the user interface displays the message `memory Test Failed`.

If this message appears, view the Test Log Output window for more information about the cause of the failure.

3.3 Floating-Point Unit (Single-Precision) Test

The floating-point unit (single-precision) test uses matrix manipulations of single-precision numbers to test the floating-point unit.

If this test fails, the failing FRU is the Node board that contains the floating-point unit that is tested or the midplane.

3.3.1 Prerequisites for Running the Floating-Point Unit (Single-Precision) Test

This test has the following prerequisites:

- The system that you want to test must have an IP17, IP19, IP20, IP21, IP22, IP25, IP26, IP28, or IP30 processor.
- The system that you want to test must have IRIX version 6.2, 6.4, or 6.5 booted and running.
- The FORTRAN libraries must be located in */usr/lib/libftn.so*.

3.3.2 Running the Floating-Point Unit (Single-Precision) Test

To run the floating-point unit (single-precision) test from the field_diag program, perform the following procedure:

1. Select the FPU-SP option from the Diagnostics list.
2. If you want to modify the parameter `Number of Iterations` (which specifies the number of matrix manipulations that the test should perform), perform the following actions; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list to select the `Number of Iterations` parameter.
   - Press the **`Enter`** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **`Enter`** key.
3. Press the left arrow key to return to the FPU-SP option in the Diagnostics list.
4. Press the **`Enter`** key to start running the floating-point unit (single-precision) test.

   The message `Executing : Floating point single precision` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the floating-point unit (single-precision) test.

Note: The System Verification Program (SVP) also runs this test. For more information about SVP, refer to the *System Verification Program (SVP) Reference Guide*, SGI publication number 108-0165-00x.

3.3.3 Output from the Floating-Point Unit (Single-Precision) Test

This test returns output to the diagnostic status area of the user interface and the Test Log Output window.

3.3.3.1 Pass Output

If this test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `Floating point single precision Test Passed`.

3.3.3.2 Failure Output

If this test detects a hardware failure or encounters an error that makes the test unable to continue, the diagnostic status area of the user interface displays the message `Floating point single precision Test Failed`.

If this message appears, view the Test Log Output window for more information about the cause of the failure.

3.4 Floating-Point Unit (Double-Precision) Test

The floating-point unit (double-precision) test uses matrix manipulations of double-precision numbers to test the floating-point unit.

If this test fails, the failing FRU is the Node board that contains the floating-point unit that is being tested or the midplane.

3.4.1 Prerequisites for Running the Floating-Point Unit (Double-Precision) Test

This test has the following prerequisites:

- The system that you want to test must have an IP17, IP19, IP20, IP21, IP22, IP25, IP26, IP28, or IP30 processor.
- The system that you want to test must have IRIX version 6.2, 6.4, or 6.5 booted and running.
- The FORTRAN libraries must be located in */usr/lib/libftn.so*.

3.4.2 Running the Floating-Point Unit (Double-Precision) Test

To run the floating-point unit (double-precision) test from the field_diag program, perform the following procedure:

1. Select the FPU-DP option from the Diagnostics list.
2. If you want to modify the parameter `Number of Iterations` (which specifies the number of matrix manipulations that the test should perform), perform the following actions; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list to select the parameter `Number of Iterations`.
   - Press the **`Enter`** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **`Enter`** key.
3. Press the left arrow key to return to the FPU-DP option in the Diagnostics list.
4. Press the **`Enter`** key to start running the floating-point unit (double-precision) test.

   The message `Executing : Floating point double precision` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the floating-point unit (double-precision) test.

Note: The System Verification Program (SVP) also runs this test. For more information about SVP, refer to the *System Verification Program (SVP) Reference Guide*, SGI publication number 108-0165-00x.

3.4.3 Output from the Floating-Point Unit (Double-Precision) Test

This test returns output to the diagnostic status area of the user interface and the Test Log Output window.

3.4.3.1 Pass Output

If this test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `Floating point double precision Test Passed`.

3.4.3.2 Failure Output

If this test detects a hardware failure or encounters an error that makes the test unable to continue, the diagnostic status area of the user interface displays the message `Floating point double precision Test Failed`.

If this message appears, view the Test Log Output window for more information about the cause of the failure.

3.5 SCSI Thrasher Test

The SCSI thrasher test stresses the SCSI subsystem. This diagnostic tests the SCSI hardware on MSCSI and BaseIO boards.

The test creates multiple processes that perform synchronized writes and reads. Each process writes a data pattern to a SCSI drive and then waits for the other processes to finish writing data patterns. When all processes have written their data patterns, the test reads the data back and compares that data with the data that was written. This test uses five different data patterns.

This test uses the following algorithm:

1. The tests mounts any SCSI drives and starts a semaphore program. (The semaphore program synchronizes all disk processes to ensure that no read operations are performed before all processes finish writing data.)
2. The test creates a */tmp* directory on the system that is tested if the directory does not already exist.
3. The test starts two copies of a *diskthrash* process for each mounted disk.
4. Each of the *diskthrash* processes writes data to two files in the */tmp* directory. The data files contain random numbers.
5. All processes wait for the semaphore program to signal that all processes have finished writing data to the */tmp* directory.
6. All processes read data back from the SCSI drives at the same time, which stresses the I/O channels with maximum traffic.
7. The test compares the data that was read back with the data that was written.
8. The test deletes all temporary files it created, deletes all temporary directories that it created, and unmounts any SCSI drives that it mounted.
9. The test generates a pass or failure message and stops.

Note: The *diskthrash* processes exit if they detect more than ten errors during the write process or read process.

If this test fails, the failing FRU is a BaseIO board, midplane, MSCSI board, or disk drive.

3.5.1 Prerequisites for Running the SCSI Thrasher Test

This test has the following prerequisites:

- The system that you want to test must have an IP17, IP19, IP20, IP21, IP22, IP25, IP26, IP28, or IP30 processor.
- The system that you want to test must have IRIX version 6.2, 6.4, or 6.5 booted and running.
- You must have disk space available on the system that you are testing to create the temporary files that this test uses.

3.5.2 Running the SCSI Thrasher Test

Caution: Run only one copy of this test at a time. If you run more than one copy at a time, the system may hang.

1. Select the SCSI option from the Diagnostics list.
2. If you want to modify any of the parameters, perform the following actions; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list.
   - Select the parameter that you want to modify. (Table 3-2 describes the parameters that you can modify.)
   - Press the **Enter** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **Enter** key.

Table 3-2 SCSI Thrasher Test User-Adjustable Parameters

Parameter	Description
`No. of Writes`	Specifies the number of writes that the test will perform.
`File Size`	Specifies the size of the files that the test will generate.

3. Press the left arrow key to return to the SCSI option in the Diagnostics list.
4. Press the **Enter** key to start running the SCSI thrasher test.

   The message `Executing : randomthrash` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the SCSI thrasher test.

Note: The System Verification Program (SVP) also runs this test. For more information about SVP, refer to the *System Verification Program (SVP) Reference Guide*, SGI publication number 108-0165-00x.

3.5.3 Output from the SCSI Thrasher Test

This test returns output to the diagnostic status area of the user interface and the Test Log Output window.

3.5.3.1 Pass Output

If this test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `randomthrash Passed`.

3.5.3.2 Failure Output

If this test detects a hardware failure or encounters an error that makes the test unable to continue, the diagnostic status area of the user interface displays the message `randomthrash Failed`.

If this message appears, view the Test Log Output window for more information about the cause of the failure. Possible messages include:

- `ERROR : Diskthrasher Thrasher Failed. Check error.log for error`

  The test failed. The *error.log* file contains more information about the failure. The failing FRU is the BaseIO board, MSCSI board, or disk drive.

- `ERROR : Data ERROR on device` *device_name*
  `Data mismatched at disk offset` *value* `Expected data` *expected* `Act.` *actual*

  The test failed. The data that was read back did not equal the data that was written. The failing FRU is the BaseIO board, MSCSI board, or disk drive.

3.6 Network Thrasher Test

The network thrasher test stresses the specified network hardware by sending large blocks of data to a remote system and then reading the data back.

The test uses the following client/server model:

- Multiple clients on the system that you are testing send data to a server on a remote system through a network connection.
- The remote server returns the data to a server on the system that you are testing.

The test compares the data that was returned with the data that was sent. (Figure 3-1 shows an example of this process with three clients.)

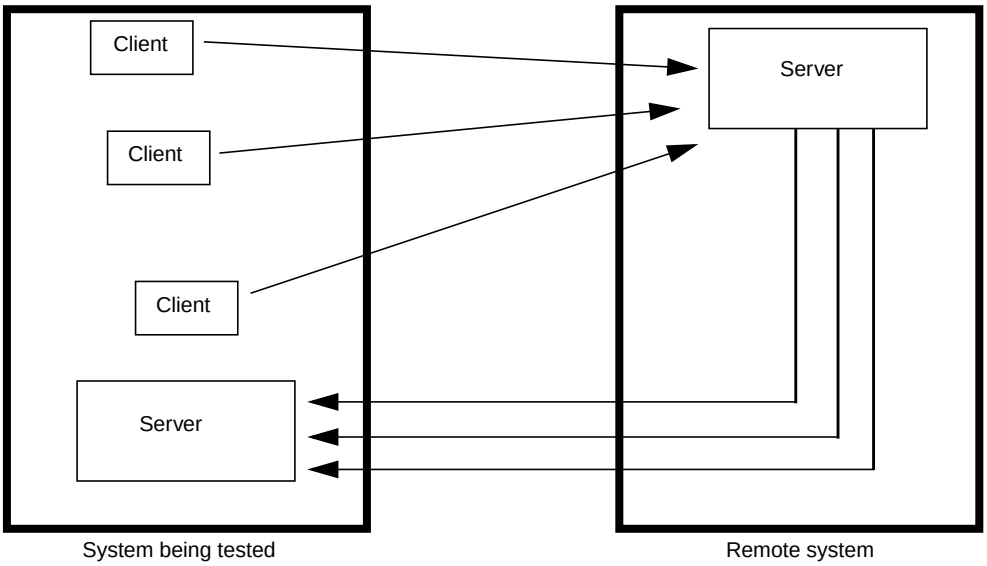

Figure 3-1 Network Thrasher Test Flow

The client processes run on the system that you are testing and simultaneously perform the following actions:

- Each client sends the address to which the server on the remote system should return the data.
- Each client transfers a large file (*/unix*) to the server on the remote system.

One server process runs on the remote system and forks a process for each connection that it receives from a client. Each forked process performs the following actions:

- The process reads the data that was sent from the client on the system that is tested.
- The process returns the data to a server process on the system that is tested.

The server process on the system that you are testing receives the data from the server on the remote system. The test compares the data that was received with the data that was sent.

If this test fails, the failing FRU is the BaseIO board or midplane.

3.6.1 Prerequisites for Running the Network Thrasher Test

This test has the following prerequisites:

- The system that you want to test must have an IP17, IP19, IP20, IP21, IP22, IP25, IP26, IP28, or IP30 processor.
- The system that you want to test must have IRIX version 6.2, 6.4, or 6.5 booted and running.
- You must have access to a remote system that is functional and running the IRIX operating system.
- You must specify values for all parameters in the Test Options list that are set to `NEEDED` by default.
- You must have disk space available on the system that you are testing to create the temporary files that this test uses. The amount of disk space that this test needs depends on the number of clients that you want it to fork.
- You must set the `remote host` parameter correctly for the system that you are testing.
- The */etc/hosts* file must have the proper information (IP address with fully qualified hostname) for the system that you want to test. For example:

  ```
  150.166.14.31 bootleg.csd.sgi.com bootleg
  ```

3.6.2 Running the Network Thrasher Test

To run the network thrasher test from the field_diag program, perform the following procedure:

1. Select the NETWORK option from the Diagnostics list.
2. If you want to modify any of the parameters, perform the following actions; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list.
   - Select the parameter that you want to modify. (Table 3-3 describes the parameters that you can modify.)
   - Press the **`Enter`** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **`Enter`** key.

Table 3-3 Network Thrasher Test User-Adjustable Parameters

Parameter	Description
`Number of Threads`	Specifies the number of network connections that the test will use. This test causes extreme stress on the network. Use caution when you adjust the number of threads that this test will use. Setting this parameter too high will adversely affect network performance and the performance of other systems on the network.
`Local Network Path`	Specifies the hostname that is assigned to the interface to be used for communications. (This option enables you to test nondefault interfaces.) Enter a hostname; for example, bootleg.sgi.com or atm-bootleg.sgi.com. You must set this parameter.
`Remote Host Name`	Specifies the remote host with which the local host will communicate. (Enter a hostname or IP address; for example, oddity.sgi.com or 150.166.237.10.) You must set this parameter.
`Login Name`	Specifies the username that the test will use to log into the remote system.
`Password`	Specifies the password that the test will use to access the remote system. If the username that you want to use does not require a password (for example, the guest account), leave this parameter blank.

3. Press the left arrow key to return to the NETWORK option in the Diagnostics list.
4. Press the **`Enter`** key to start running the network thrasher test.

   The message `Executing : network` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the network thrasher test.

Note: The System Verification Program (SVP) also runs this test. For more information about SVP, refer to the *System Verification Program (SVP) Reference Guide*, SGI publication number 108-0165-00x.

3.6.3 Output from the Network Thrasher Test

This test returns output to the diagnostic status area of the user interface and the Test Log Output window.

3.6.3.1 Pass Output

If this test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `network Test Passed`.

3.6.3.2 Failure Output

If this test detects a hardware failure or encounters an error that makes it unable to continue testing, the diagnostic status area of the user interface displays the message `network Test Failed`.

If this message appears, view the Test Log Output window for more information about the cause of the failure. Possible messages include:

- `ERROR: Network Thrasher Failed. Check error log for error`

  The test failed with an error. This error occurs when a data integrity error occurs after the data transfer is completed. The failing FRU is the board that contains the network interface that is tested.

- `ERROR: Remote System is not responding. Cannot Continue`

  The test failed with an error. This error occurs when the test checks to determine whether the remote system is accessible. The remote system may not be responding for many reasons. (For example, the network is not functioning, the remote system is down, or the test system is not configured correctly.)

- `ERROR: Could not fork off requested number of clients`

  The test failed with an error. This error occurs if the number of clients is greater than the number of processes that can be forked.

3.7 InfiniteReality Graphics Test

The InfiniteReality graphics test (irsaudit) checks the InfiniteReality graphics hardware.

If this test fails, the failing FRU is one of the graphics subsystems (RM, GE, or DG) or midplane. The test output indicates which subsystem is failing.

3.7.1 Prerequisites for Running the InfiniteReality Graphics Test

This test has the following prerequisites:

- The system that you want to test must have an IP21 or IP25 processor.
- The system that you want to test must have IRIX version 6.2, 6.4, or 6.5 booted and running.
- You must run this test from superuser mode.
- You must run this test from an alternate terminal; you cannot run this test from the graphics console.

3.7.2 Running the InfiniteReality Graphics Test

To run the InfiniteReality graphics test from the field_diag program, perform the following procedure:

1. Select the IRSAUDIT option from the Diagnostics list.
2. If you want to modify any of the parameters, perform the following actions; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list.
   - Select the parameter that you want to modify. (Table 3-4 describes the parameters that you can modify.)
   - Press the **`Enter`** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **`Enter`** key.

Table 3-4 InfiniteReality Graphics Test User-Adjustable Parameters

Parameter	Description
`Output TRCE Messages`	Specifies whether the test should output TRCE messages. Set this parameter to -notrace (to specify that the test not print these messages) or -trace (to specify that the test print these messages).
`Output CODE Messages`	Specifies whether the test should output CODE messages. Refer to Table 3-5 for more information about CODE messages. Set this parameter to -nocode (to specify that the test not print these messages) or -code (to specify that the test print these messages).
`Output INFO Messages`	Specifies whether the test should output INFO messages. Refer to Table 3-5 for more information about INFO messages. Set this parameter to -noinfo (to specify that this test not print these messages) or -info (to specify that the test print these messages).
`Continue on Error`	Specifies whether or not the test will continue when it detects an error. Set this parameter to -continue to continue testing when the test detects an error.

3. Press the left arrow key to return to the IRSAUDIT option in the Diagnostics list.
4. Press the **`Enter`** key to start running the InfiniteReality graphics test.

   The message `Executing : IRSAUDIT` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the InfiniteReality graphics test.

Note: Refer to the irsaudit reference page for information about running this test from the command line.

3.7.3 Output from the InfiniteReality Graphics Test

This test returns output to the diagnostic status area of the user interface and the Test Log Output window. The output includes pass messages, failure messages, and messages that are tagged with the four-character identifiers shown in Table 3-5.

Table 3-5 InfiniteReality Graphics Test Output Tags

Tag	Description
TEST	Test start message. This message, which is generated at the beginning of the test, includes the symbolic name of the test and a description of the test.
RSLT	Test result message. This message, which is generated at the end of a test, includes the symbolic name of the test and the test result. The result is either pass, fail, unresolved, or untested.
DIAG	Diagnosis message. This message precedes any “fail” or “unresolved“ message. The message indicates the components or wires that are possible causes of the failure.
INFO	Informational message. This message provides test progress reports, expected / received pairs, or other useful information. This message is useful to an advanced user of the test.
DBUG	Debugging informational message. This message is not intended for normal field use of this test but can be useful to diagnostic programmers. The test generates this type of message only when debugging output is turned on (not the default).
TIME	Time stamp. This message provides information at important time boundaries, such as the beginning and end of the test.
META	Summary message from multiple test runs. This message summarizes information from multiple test runs. The test generates a table that provides the pass and fail counts of each test run (and totals for all test runs) across multiple full-test loops.
ABRT	Abort message. This message reports an exceptional error condition that causes the test to abort. (Possible causes include a malloc failure, unexpected system call failure, or assertion failure.) This message should not appear under normal circumstances.
CODE	Encoded message. This message encodes board and ASIC failures for easy parsing.

Refer to the irsaudit online reference page for more information about the test output tags.

3.7.3.1 Pass Output

If this test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `IRSAUDIT Test Passed`.

3.7.3.2 Failure Output

If this test detects a hardware failure or encounters an error that makes the test unable to continue, the diagnostic status area of the user interface displays the message `IRSAUDIT Test Failed`.

If this message appears, view the Test Log Output window for more information about the cause of the failure.

3.8 Vicious HIPPI Tests

The vicious HIPPI tests are exercisers that test HIPPI channel pairs. There are two vicious HIPPI tests:

- The VHT_LOOPBACK test uses raw channel HIPPI data to exercise the configured HIPPI network device on the system that you are testing. It sends and receives data through a loopback cable on the HIPPI-S board. It then compares the actual data with expected values.
- The VHT_REMOTE test uses raw channel HIPPI data to exercise the configured HIPPI network device on the system that you are testing. It can run on one system with a loopback cable installed or between two connected systems.

  To exercise the HIPPI interface on one system, it sends and receives data through the loopback cable on the HIPPI-S board. It then compares the actual data with expected values. To run the test on one system, set the `Local Interface` and `Remote Interface` parameters to the same value. (Refer to Table 3-7.)

  To exercise the HIPPI interface using two systems, it writes data from the remote system to the local system. It then compares the actual data read with expected values.

If one of these tests fails, the failing hardware may be the HIPPI network interface that is tested or the midplane. The failing FRU may be the HIPPI-S board that contains the failing network interface or the midplane.

3.8.1 Prerequisites for Running the Vicious HIPPI Tests

These tests have the following prerequisites:

- The system that you want to test must have an IP19, IP21, or IP25 processor.
- The system that you want to test must have IRIX version 6.2, 6.4, or 6.5 booted and running.
- You must specify values for all parameters in the Test Options list that are set to `NEEDED` by default.
- You must use the HIPPI device down with the *ifconfig* command (for example, *ifconfig hip0 down*).
- To run the VHT_LOOPBACK test, you must connect a loopback cable on the HIPPI-S board that you want to test.
- To run the VHT_REMOTE test on one system, you must connect a loopback cable on the HIPPI-S board that you want to test.
- To run the VHT_REMOTE test on two systems, you must have access to a functional remote system that is running a version of IRIX that is equal to or newer than the version on the on the local system. (For example, if the local system is running IRIX version 6.2, then the remote system should be running IRIX version 6.2, 6.4, or 6.5. If the local system is running IRIX version 6.5, then the remote system must be running IRIX version 6.5.)

3.8.2 Running the Vicious HIPPI Tests

You can run the vicious HIPPI tests from the field_diag program.

3.8.2.1 Running the VHT_LOOPBACK Test

To run the VHT_LOOPBACK test from the field_diag program, perform the following procedure:

1. Select the VHT_LOOPBACK option from the Diagnostics list.
2. If you want to modify any of the parameters, perform the following actions; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list.
   - Select the parameter that you want to modify. (Table 3-6 describes the parameters that you can modify.)
   - Press the **Enter** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **Enter** key.

Table 3-6 VHT_LOOPBACK Test User-Adjustable Parameters

Parameter	Description
`num. packets`	Specifies the number of data packets that the test will send.
`Pattern`	Specifies the data pattern that the test will use. Enter a hexadecimal number or any of the following settings: `bits` specifies that each 64-bit data word will have a random sequence of consecutive 1 bits. `slide0` specifies that the first 64-bit data word will have all bits (except bit 0) set to 1; bit 0 will be set to 0. Subsequent data words will be circularly left-shifted by one bit position. `slide1` specifies that the first 64-bit data word will have all bits (except bit 0) set to 0; bit 0 will be set to 1. Subsequent data words will be circularly left-shifted by one bit position. `random` specifies that data words will be set to random values. `all` specifies that the test will use built-in data patterns, one per pass, in a circular pattern.

Table 3-6 (continued) VHT_LOOPBACK Test User-Adjustable Parameters

Parameter	Description
`Block size`	Specifies the block size to use for I/O operations (in bytes). Enter `pass` or one or more values in the following format: *max* [:*min*[:*step*]] When you use the *min* value, the test randomly generates a block size value in the range from *min* to *max*. If you specify a *step* value, the randomly selected value increases by the value of *step* for each pass of the test, until the test reaches the *max* value. When you use the `pass` option, the test sets the block size to the current pass count value.
`Interface`	Specifies the device name of the HIPPI interface that the test will use. (For example, /dev/hippi1)

3. Press the left arrow key to return to the VHT_LOOPBACK option in the Diagnostics list.
4. Press the **`Enter`** key to start running the VHT_LOOPBACK test.

   The message `Executing : VHT_LOOPBACK` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the VHT_LOOPBACK test.

3.8.2.2 Running the VHT_REMOTE Test

To run the VHT_REMOTE test from the field_diag program, perform the following procedure:

1. Select the VHT_REMOTE option from the Diagnostics list.
2. If you want to modify any of the parameters, perform the following actions; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list.
   - Select the parameter that you want to modify. (Table 3-7 describes the parameters that you can modify.)
   - Press the **`Enter`** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **`Enter`** key.

Table 3-7 VHT_REMOTE Test User-Adjustable Parameters

Parameter	Description
`num. packets`	Specifies the number of data packets that the test will send.
`remote host`	Specifies the remote system with which the test will communicate. Enter a hostname or IP address. You must set this parameter.
`Pattern`	Specifies the data pattern that the test will use. Specify a hexadecimal number or any of the following settings: `bits` specifies that each 64-bit data word will have a random sequence of consecutive 1 bits. `slide0` specifies that the first 64-bit data word will have all bits (except bit 0) set to 1; bit 0 will be set to 0. Subsequent data words will be circularly left-shifted by one bit position. `slide1` specifies that the first 64-bit data word will have all bits (except bit 0) set to 0; bit 0 will be set to 1. Subsequent data words will be circularly left-shifted by one bit position. `random` specifies that data words will be set to random values. `all` specifies that the test should use built-in data patterns, one per pass, in a circular pattern.
`Block size`	Specifies the block size to use for I/O operations (in bytes). Specify `pass` or one or more values in the following format: *max* [:*min*[:*step*]] When you use the *min* value, the test randomly generates a block size value in the range from *min* to *max*. If you specify a *step* value, the randomly selected value increases by the value of *step* for each pass of the test, until the test reaches the *max* value. When you use the `pass` option, the test sets the block size to the current pass count value.
`I-field`	Specifies the I-field value for the HIPPI interface on the local host. See the */usr/etc/hippi* file or use the *hipmap* command to determine the I-field value. (For example, 0x0300001d, 0x01000004, and 0x3c) You must set this parameter. If you are using one system in loopback mode or two systems that are directly connected (that is, no HIPPI switch is used), the value of this parameter is not important. Set this parameter to 0x00000000 in either of these situations.
`Local Interface`	Specifies the device name of the HIPPI interface on the local system. (This device will check for writes. For example, /dev/hippi0) You must set this parameter.
`Remote Interface`	Specifies the device name of the HIPPI interface on the remote system. (This device will send the writes. For example, /dev/hippi1) To run the test in loopback mode, set the `Local Interface` and `Remote Interface` parameters to the same value. You must set this parameter.

Table 3-7 (continued) VHT_REMOTE Test User-Adjustable Parameters

Parameter	Description
`Login Name`	Specifies the login (username) of the account to use on the remote host. (The default is root.)
`Password`	Specifies the password for the account on the remote host (if required).

3. Press the left arrow key to return to the VHT_REMOTE option in the Diagnostics list.
4. Press the **`Enter`** key to start running the VHT_REMOTE test.

   The message `Executing : VHT_REMOTE` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the VHT_REMOTE test.

3.8.3 Output from the Vicious HIPPI Tests

This test returns output to the diagnostic status area of the user interface and the Test Log Output window.

3.8.3.1 Pass Output

If the VHT_LOOPBACK test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `VHT_LOOPBACK Test Passed`.

If the VHT_REMOTE test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `VHT_REMOTE Test Passed`.

3.8.3.2 Failure Output

If the VHT_LOOPBACK test detects a hardware failure or encounters an error that makes it unable to continue testing, the diagnostic status area of the user interface displays the message `VHT_LOOPBACK Test Failed`.

If the VHT_REMOTE test detects a hardware failure or encounters an error that makes it unable to continue testing, the diagnostic status area of the user interface displays the message `VHT_REMOTE Test Failed`.

If one of these messages appears, view the Test Log Output window for more information about the cause of the failure. Possible messages include:

- ***** Data Compare Error *****
  Differences written to "*file*".
  Current pass = 1 Current data pattern = slide1

  The test failed because the actual data did not equal the expected data. The failing FRU may be the HIPPI-S board. Refer to the generated error file for more information.

  The following output shows a partial sample of the contents of an error file:

```
Pass : 1
A : expected
B : actual
x : logical difference

A( 0)+ 8000000000000198 0000000100800080 *...............*
B( 0)+ 8000000000000198 0000000000000000 *...............*
x( 0)+ 0000000000000000 0000000100800080 *...............*
A(10)+ 00000000007f0000 0000000000000006 *...............*
B(10)+ 0000000000000001 0000000000000002 *...............*
x(10)+ 00000000007f0001 0000000000000004 *...............*
A(20)+ 00000000001ffe00 00000fffffffe00 *...............*
B(20)+ 0000000000000004 0000000000000008 *...............*
x(20)+ 00000000001ffe04 00000fffffffe08 *...............*
```

- Executing Write Test
  connect() failed, errno = 152, text = "Connection refused".

  A write operation failed because there was no remote system to receive the data. This failure may have occurred because the remote system was not running or because the network interface failed. The failing FRU may be the HIPPI-S board.

3.9 Vicious Socket Tests

The vicious socket tests are socket-based exercisers that use TCP sockets to send and receive data across a network. There are two vicious socket tests:

- The VST_PING test uses ICMP to send data packets to a remote system, which then returns the packets to the vicious socket test process on the local system. The local system verifies that the data returned from the remote system is the same as the data sent to the remote system. You must run this test as root.
- The VST_RW test reads and writes data patterns between two systems. It compares the data that is sent from the remote system with the data that is received by the local system.

If one of these tests fails, the failing hardware is the network interface that is tested or the midplane. The failing FRU is the board that contains the failing network interface or the midplane.

3.9.1 Prerequisites for Running the Vicious Socket Tests

These tests have the following prerequisites:

- The system that you want to test must have an IP17, IP19, IP20, IP21, IP22, IP25, IP26, IP28, or IP30 processor.
- The system that you want to test must have IRIX version 6.2, 6.4, or 6.5 booted and running.
- You must have access to a remote system that is functional and is running the IRIX operating system if you want to run the vicious socket test between two systems. (The version of IRIX on the remote system must be equal to or newer than the version on the local system. [For example, if the local system is running IRIX version 6.2, then the remote system must be running IRIX version 6.2, 6.4, or 6.5. If the local system is running IRIX version 6.5, then the remote system must be running IRIX version 6.5.])
- You must specify values for all parameters in the Test Options list that are set to `NEEDED` by default.

3.9.2 Running the Vicious Socket Tests

You can run the vicious socket tests from the field_diag program.

3.9.2.1 Running the VST_PING Test

To run the VST_PING test from the field_diag program, perform the following procedure:

1. Select the VST_PING option from the Diagnostics list.
2. If you want to modify any of the parameters, perform the following actions for each parameter; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list.
   - Select the parameter that you want to modify. (Table 3-8 describes the parameters that you can modify.)
   - Press the **Enter** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **Enter** key.

Table 3-8 VST_PING Test User-Adjustable Parameters

Parameter	Description
`num. packets`	Specifies the number of data packets that the test will send to the remote system.
`remote host`	Specifies the host to which the test will send the data packets. You must enter this parameter.
`Pattern`	Specifies the data pattern that the test will use. Enter a hexadecimal number or any of the following settings: `bits` specifies that each 64-bit data word will have a random sequence of consecutive 1 bits. `slide0` specifies that the first 64-bit data word will have all bits (except bit 0) set to 1; bit 0 will be set to 0. Subsequent data words will be circularly left-shifted by one bit position. `slide1` specifies that the first 64-bit data word will have all bits (except bit 0) set to 0; bit 0 will be set to 1. Subsequent data words will be circularly left-shifted by one bit position. `random` specifies that data words will be set to random values. `all` specifies that the test will use built-in data patterns, one per pass, in a circular pattern.

3. Press the left arrow key to return to the VST_PING option in the Diagnostics list.
4. Press the **Enter** key to start running the VST_PING test.

   The message `Executing : VST_PING` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the VST_PING test.

3.9.2.2 Running the VST_RW Test

To run the VST_RW test from the field_diag program, perform the following procedure:

1. Select the VST_RW option from the Diagnostics list.
2. If you want to modify any of the parameters, perform the following actions for each parameter; otherwise, go to Step 4.
   - Press the right arrow key to move into the Diagnostic Options list.
   - Select the parameter that you want to modify. (Table 3-9 describes the parameters that you can modify.)
   - Press the **Enter** key.
   - At the prompt `Enter New Value for the Option` in the diagnostic status area, enter the new value.
   - Press the **Enter** key.

Table 3-9 VST_RW Test User-Adjustable Parameters

Parameter	Description
`num. packets`	Specifies the number of data packets that the test will send to the remote system.
`remote host`	Specifies the host to which the test will send the data packets. You must enter this parameter.
`Pattern`	Specifies the data pattern that the test will use. Enter a hexadecimal number or any of the following settings: `bits` specifies that each 64-bit data word will have a random sequence of consecutive 1 bits. `slide0` specifies that the first 64-bit data word will have all bits (except bit 0) set to 1; bit 0 will be set to 0. Subsequent data words will be circularly left-shifted by one bit position. `slide1` specifies that the first 64-bit data word will have all bits (except bit 0) set to 0; bit 0 will be set to 1. Subsequent data words will be circularly left-shifted by one bit position. `random` specifies that data words will be set to random values. `all` specifies that the test will use built-in data patterns, one per pass, in a circular pattern.
`Block size`	Specifies the block size to use for I/O operations. Specify a number of bytes or `pass`. `pass` indicates that the test will set the blocksize to the current pass count value.
`Login Name`	Specifies the login (username) of the account to use on the remote host. (The default is root.)
`Password`	Specifies the password for the account on the remote host (if required).

3. Press the left arrow key to return to the VST_RW option in the Diagnostics list.
4. Press the **Enter** key to start running the VST_RW test.

   The message `Executing : VST_RW` appears in the diagnostic status area, which indicates that the test is running.
5. View the diagnostic status area and Test Log Output window for messages from the VST_RW test.

3.9.3 Output from the Vicious Socket Tests

The tests return output to the diagnostic status area of the user interface and the Test Log Output window.

3.9.3.1 Pass Output

If the VST_PING test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `VST_PING Test Passed`.

If the VST_RW test completes without detecting any hardware failures, the diagnostic status area of the user interface displays the message `VST_RW Test Passed`.

3.9.3.2 Failure Output

If the VST_PING test detects a hardware failure or encounters an error that makes the test unable to continue, the diagnostic status area of the user interface displays the message `VST_PING Test Failed`.

If the VST_RW test detects a hardware failure or encounters an error that makes the test unable to continue, the diagnostic status area of the user interface displays the message `VST_RW Test Failed`.

If one of these messages appears, view the Test Log Output window for more information about the cause of the failure. Possible messages include:

- `Executing Ping Test`
  `Ping timed out`
  `Expected to write` *num* `bytes, sent 0 instead`
  `***** Error *****`
  `Current pass = 1`
  `Current data pattern =` *pattern*

  The VST_PING test failed because the remote system did not respond. The failing FRU may be the board that contains the network interface that connects to the remote system.
- `Executing Write Test`
  `connect() failed, errno = 152, text = “Connection refused”.`

  A write operation failed because there was no remote system to receive the data. This failure may have occurred because the remote system was not running or because the network interface failed. The failing FRU may be the HIPPI-S board.

3.10 Summary of Chapter 3

This chapter described the diagnostic tests that you can run from the field_diag user interface. Table 3-10 summarizes the pass and fail messages and the failing FRU for each test.

Table 3-10 Summary of Diagnostic Test Information

Diagnostic Test	Pass Message	Fail Message	Failing FRU
Floating-point unit (single-precision) test	`Floating point single precision Test Passed`	`Floating point single precision Test Failed`	Node board or midplane
Floating-point unit (double-precision) test	`Floating point double precision Test Passed`	`Floating point double precision test Failed`	Node board or midplane
InfiniteReality graphics test	`IRSAUDIT Passed`	`IRSAUDIT Failed`	Graphics subsystem (RM, GE, or DG) or midplane
Memory test	`memory Test Passed`	`memory Test Failed`	DIMM, Node board, or midplane
Network thrasher test	`network Test Passed`	`network Test Failed`	ATM, BaseIO, HIPPI-S, or MENET board or midplane
SCSI thrasher test	`randomthrash Test Passed`	`randomthrash Test Failed`	BaseIO board, MSCSI board, disk drive, or midplane
Vicious HIPPI tests	`VHT_LOOPBACK Test Passed` or `VHT_REMOTE Test Passed`	`VHT_LOOPBACK Test Failed` or `VHT_REMOTE Test Failed`	HIPPI-S board or midplane
Vicious socket tests	`VST_PING Test Passed` or `VST_RW Test Passed`	`VST_PING Test Failed` or `VST_RW Test Failed`	BaseIO board, MENET board, ATM board, HIPPI-S board, or midplane