System Verification Program (SVP) Reference Guide

SGI Confidential & Proprietary Information - For Internal Recipients Only

Document Number 108-0165-007

CONTRIBUTORS
Written by Stacey Delcore
Edited by Cindi Leiser, Sara Horwath
Engineering contributions by Nadia Pavlova

System Verification Program (SVP) Reference Guide
Document Number 108-0165-007

SGI
Mountain View, California

Contents

Figures

Tables

Introduction

This document contains information about the SGI service tool SVP, the System Verification Program. This information includes a general program overview; instructions on installation, configuration, and operation of this program; and descriptions of status and error reports.

The following typographic conventions are used throughout this document:

Convention	Meaning
`TYPEWRITER FONT`	Denotes literal items such as command names, file names, routines, directory names, path names, signals, messages, and programming language structures.
italic font	Denotes variable entries and words or concepts being defined.
`bold typewriter font`	In screen drawings of interactive sessions, denotes literal items entered by the user. Output is shown in nonbold typewriter font.
[]	Indicates an optional item.
<>	Indicates a required variable within an optional item.

Within this document, reference is made to the online man pages that are available under the IRIX operating system through the `man` command. A *man page* is a discussion of a particular element of the IRIX operating system or a compatible product.

Each man page includes a general description of one or more commands, routines, or other topics and provides details of their usage (command syntax, routine parameters, system call arguments, and so on). If more than one topic appears on a page, the entry will appear in the printed manual alphabetized only under its major name. You can access a man page named `ls` online by typing **`man ls`**.

Man pages are grouped into sections that are numbered from 1 to 8. Each section contains entries of a particular type. Types of entries include user commands (1), administrator commands (8), system calls (2), library routines (3), file formats (5), and device descriptions (4).

Section numbers appear in parentheses after man page names. Man pages are referenced in text by entry name and section number.

Chapter 1

Introduction to the System Verification Program

This chapter provides an overview of the System Verification Program (SVP), explains its purpose, and lists the benefits and limitations of this service tool.

1.1 Product Overview

The System Verification Program (SVP) is an IRIX based software tool that contains utilities and test programs, inventories the system hardware and software components, loads and runs a suite of test programs, and generates a set of files that report the results of this activity.

SVP helps a field Systems Support Engineer (SSE) or a Channel Support Engineer (CSE), hereinafter collectively referred to as the *support engineer,* verify the configuration and functionality of a newly installed system. SVP is a system acceptance test tool, and the test result is part of the system installation sign-off requirement for the support engineer.

Starting with IRIX operating system release version 6.5.5, SVP logs events in the Embedded Support Partner framework.

1.2 Rationale for SVP

The SVP process is a uniform approach that is used to verify the functionality of a newly delivered system. This process will be used for all high-end systems.

The SVP process strengthens the link between Manufacturing and the field. The testing of systems before and after shipment is a joint venture of Manufacturing and Service personnel.

The SVP process ultimately reduces support costs. SVP guides the service provider to a predictable and rational conclusion of the installation process. Problems, once resolved, need not be readdressed at future installations.

SVP builds customer confidence. The SVP process assures customers that the complete shipment of hardware and software that they have ordered has been delivered and that the assembled system is functional.

1.3 SVP Process

As the last step in the Origin system final integration-testing process, Manufacturing runs SVP to measure and document the functionality and performance of the system. This test immediately precedes the shutdown, disassembly, and packaging of the system for shipment to the customer.

SVP results are automatically logged to the Embedded Support Partner (ESP) for IRIX 6.5.5 and later versions. Refer to the *Embedded Support Partner User Guide*, publication number 007-4065-004, for more information on ESP. Refer to Chapter 4, "SVP Configuration Information," for information on SVP results for earlier IRIX versions.

After delivery, installation, and initial power-on diagnostic testing at the customer site, the support engineer retests the system with SVP, without the overpack items installed. (Perform *Out of Box* tests first.) The results of this test session are compared to the results of the final Manufacturing test. Ideally, the results of both tests will match. Then the support engineer should install the overpack items as the final setup of the customer configuration and run SVP again.

Subsequently, if the system installation plan requires that hardware or software modifications be made to the system, the support engineer must ensure that the initial SVP session has completed successfully before proceeding with these modifications. Following system modifications, the support engineer should again run SVP to establish new SVP results for the machine and to log them into ESP. Running SVP is always the final step of any hardware or software modification activity.

1.4 Current Test Coverage

Current test coverage for SVP includes:

- Peripheral confidence tests

  The peripheral confidence tests check the system peripherals such as the monitor, keyboard, mouse, spaceball, CD-ROM, tape drive, tablet, video, and floppy drive. The peripheral confidence tests require that the system contain a graphics head.

  Peripheral confidence tests are interactive. For each test, the support engineer must intervene manually to ensure the functionality of the device. SVP tests only connected devices. Unconnected devices can be set up with the system manager from the toolchest.

- Subsystem-level diagnostics under the IRIX operating system

  The board-level diagnostic set exercises the CPU, floating-point unit (FPU), memory, and I/O subsystems.

  CPU and FPU functionality tests use matrix manipulations for both single-precision and double-precision floating-point numbers. Matrix tests require the FORTRAN libraries located at `/usr/lib/libftn.so`.

  The memory diagnostic allocates a chunk of memory and performs bit-pattern testing. The amount of allocated memory depends on the value of `memoryuse`, which is returned by the `limit` command that is built into `sh(1)`.

Caution: Changing this limit parameter requires rebuilding the kernel and should be done only if you seriously doubt the integrity of memory. After you have finished testing the memory, reset the per-process limit to the IRIX default value.

If you are using IRIX 6.2 or IRIX 6.5 on a system with an IP21 or an IP25 processor, you can call cachetest by turning on field mode (the `-f` option). Cachetest tests the second-level cache in each CPU, and it is capable of detecting all stuck-at faults and some pattern sensitivity problems (that can be modeled as coupling between rows and columns of the memory array).

I/O subsystem diagnostics test the BASEIO, MIO, HIPPI, ATM, MSCSI, and MENET. All PCI boards will have some amount of coverage with USRPCI diagnostics. The current release limits the USRPCI diagnostics to ATM, HIPPI, and MIO. The USRPCI diagnostics perform low-level diagnosis with PIO reads/writes to check functionality of the subsystem. Some of the tests may require external cable interfaces to cover 100% functionality. Otherwise, the tests cover about 50% to 70% functionality of the board.

The Ethernet subsystem contains a test that is automatically called by SVP. This test is called the Ethernet Thrasher and is a client/server structured socket-level program that can communicate with another host system. The host system must be running the IRIX operating system.

I/O tests exercise only mounted disks. SVP detects mounted disks and creates temporary directories from which `dd` and `tar` commands are executed. SVP then compares the test data against a source directory. Another test, the SCSI Thrasher, stresses the SCSI subsystem. This test detects all mounted disks and, using system calls, it performs parallel reads/writes. Semaphores synchronize all reads/writes on all disks.

The processor element random instruction (`olperi`) test verifies all integer, floating-point, branch, and memory load and store operations. It uses all available virtual address space to test the address translation logic of a selected processor. The processor test runs with IRIX 6.5 and later IRIX versions on Origin 3000, Onyx 3, and Origin 300 systems.

Configuration collection for the fibre channel RAID subsystem does not read and write data on the fibre channel disks. However, the configuration collection process does issue pass-through commands to the disk drives and, in this way, tests the functionality of the SCSI bus for both data and commands. The configuration information that is collected is stored with the other configuration data for the system.

Note: For additional information on all subsystem level diagnostics under IRIX and support matrix for IP types, refer to the document titled *Online Diagnostics*, publication number 108-0286-002.

- IRIX command tests

  The IRIX command tests check basic IRIX commands, compiler commands, network-related commands, graphics-related commands for all graphics, and commands that test OpenGL libraries. The IRIX command tests demonstrate graphics capabilities and also conduct audio or audio-related tests. All of these tests depend on the availability of supporting software. If the required software is not present, SVP skips the tests and displays messages to indicate that the software is missing.

- InfiniteReality graphics diagnostic test

A prerequisite to running the InfiniteReality graphics diagnostic test is that `irsaudit` diagnostics, bundled either in `diag.sw.IR` or in `eoe.sw.gfx`, be loaded on the disk. SVP does not execute `irsaudit` diagnostics by default. To run `irsaudit`, you must add the `-I` option when you enter the `svp` command.

Caution: If InfiniteReality graphics exist, you must run SVP from another terminal attached to the system because `irsaudit` will reset the terminal and you will lose all the results of the previous SVP tests. All demos and results of the `irsaudit` and IRIX applications tests will be displayed only on the graphics terminal.

- Digital Video Diagnostic test (DIVO)

  A prerequisite to running the DIVO test is that divotest and loopback diagnostics bundled in `divo.sw.diags` be loaded on the disk. SVP recognizes installed DIVO boards and automatically runs the Digital Video Diagnostic test. This diagnostic is similar to `irsaudit`. It performs both board tests and system tests to verify the functionality of the DIVO board.

- Torpedo test

  The `torpedo` test is a floating-point unit (FPU) stress test that uses several standard floating-point algorithms to test the FPU in each CPU. It compares the results from all CPUs and reports any discrepancies. The `torpedo` test verifies basic FPU functionality (add, subtract, multiply, divide, and square root) before it performs complex operations (parallel operations, trigonometric operations, root findings, and matrix operations).

Separate images of SVP are available for the following IRIX operating system versions:

- IRIX 5.3 for all systems
- IRIX 6.2 for all systems
- IRIX 6.3 for all systems
- IRIX 6.4 for all systems
- IRIX 6.5 through 6.5.4 for all systems
- IRIX 6.5.5 through 6.5.8 for all systems
- IRIX 6.5.9 for Origin 3000 and Onyx 3 systems
- IRIX 6.5.9 for non Origin 3000 and non Onyx 3 systems
- IRIX 6.5.10 for Origin systems
- IRIX 6.5.10 for non Origin systems

SVP automatically recognizes the system configuration and performs appropriate tests based on this environment. (Some features of the newer images may not apply to older versions of the operating system.)

Note: Test coverage may vary, depending on the hardware and software configurations and the version of the IRIX operating system.

1.5 List of Product Tasks and Deliverables

SVP develops and reports the following information during its execution:

- Provides date stamp and system name
- Logs *SVP start* and *SVP end* events into the ESP framework. *SVP end* event contains the SVP test results that are logged into the ESP database.
- Lists software patches installed (IRIX 6.4 and 6.5)
- Lists mounted and unmounted file systems
- Provides software inventory
- Provides system configuration
- Provides hardware inventory (IRIX 6.4, 6.5 through 6.5.4, and 6.5.5 - IP27, IP29, IP30, and IP31)
- Provides current memory inventory (IRIX 6.4, 6.5 through 6.5.4, and 6.5 - IP27, IP29, IP30, and IP31)
- Provides disk inventory (IRIX 6.4, 6.5 through 6.5.4, and 6.5 - IP27, IP29, IP30, and IP31)
- Provides graphic's pipes inventory (IRIX 6.4, 6.5 through 6.5.4, and 6.5 - InfiniteReality and InfiniteReality2)
- Performs confidence tests on peripheral devices
- Performs board-level tests
- Performs IRIX level tests
- Performs communication network tests

Refer to Appendix B for an example of a typical SVP report listing.

1.6 Description of Product Limitations

The System Verification Program runs under the IRIX operating system (OS) and requires Perl to be installed. (Perl is distributed with the IRIX OS in the `eoe.sw.gifts_perl` and `eoe.sw.gifts_perl_lib` subsystems.) If the hardware is incapable of loading and running the operating system, you will have to troubleshoot and repair the system using other software. SVP must be run after all the system software has been installed.

Do not run other tests or applications when you are running SVP. If you ignore this restriction, you may receive false results during board-level tests. Problems can also arise from improper or insufficient allocation of memory or from an insufficient amount of logical swap space for diagnostics.

SVP is a test suite that calls out a subset of diagnostics and other IRIX related commands to measure the functional integrity of the system. If the system fails, you will need to use other diagnostic methods to restore the system to operation.

Chapter 2

Getting Started

This chapter describes how to install and operate the System Verification Program.

2.1 Installing the System Verification Program

SVP can be distributed either locally or remotely.

2.1.1 Local Distribution

Local distribution means that SVP is distributed by compact disk (CD). To install SVP on the site workstation from a CD, use the following procedure:

1. Become a superuser of the workstation that has the CD-ROM drive.

   ```
   % /bin/su -
   Password:
   #
   ```

2. Insert the CD that contains the SVP distribution, with the label facing up, into the CD-ROM caddy and push the caddy into the CD-ROM drive.

3. Determine whether the `mediad(1M)` process is running on your system by entering

   ```
   # ps -efdl | grep mediad | grep -v grep
   ```

   If the `mediad` process is running, skip to Step 6. Otherwise proceed to Step 4.

4. If it does not already exist, create a mount point directory called `/CDROM` on the workstation that has the CD-ROM drive by entering

   ```
   # mkdir /CDROM
   ```

5. Determine whether `objectserver(1M)` is running on the workstation that has the CD-ROM drive by entering

   ```
   # ps -efdl | grep objectserver | grep -v grep
   ```

   If an `objectserver` process is running, run `mediad` on the workstation that has the CD-ROM drive by entering

   ```
   # /usr/etc/mediad
   ```

   After running `mediad`, continue to Step 6. If an `objectserver` process is not running, use the `mount(1M)` command to mount the CD.

First, run `hinv(1M)` to determine the <cntlr> number and the <unit> number on the CD-ROM drive by entering

```
# hinv | grep CDROM
  CDROM: unit 4 on SCSI controller 0
```

The <unit> number and the <cntrl> number are returned on the “CDROM” line.

Next, mount the CD on the workstation that has the CD-ROM drive. For systems running IRIX 5.3 through IRIX 6.3, enter

```
# /sbin/mount -o ro -t iso9660 /dev/scsi/sc<cntrl>d<unit>vol/CDROM
```

For systems running IRIX 6.3 (or later), enter

```
# /sbin/mount -o ro -t iso9660 /dev/rdsk/dks<cntrl>d<unit>vol/CDROM
```

This command will properly mount the CD on the local system. If this command fails, please consult your local SGI technical support center for assistance. Otherwise, skip to Step 7.

6. Verify that the CD-ROM has mounted properly by entering

   ```
   # ls /CDROM
   ```

   If files appear in /CDROM, proceed to Step 7. If files do not appear, then turn off `mediad(1M)` and mount the CD by hand. To turn off `mediad`, enter

   ```
   # /usr/etc/mediad -k
   ```

   After turning off `mediad`, return to Step 4.

7. Once the CD is mounted, change the working directory to the path where the SVP images are located.

   # **cd <***path***>**

   where *<path>* is the path determined in Step 6.

8. Invoke `inst` by entering the command:

   ```
   # /usr/sbin/inst -f.
   ```

   When the Inst prompt appears, enter the following commands (in order shown from top to bottom):

   ```
   Inst> list
   Inst> go
   Inst> quit
   ```

The SVP software loads from the CD to a directory, `/usr/SVP`. You are now ready to run the System Verification Program.

2.1.2 Remote Distribution

Remote distribution means that SVP is available through the System and Site Support Tools Web site at:

`http://ist.csd.sgi.com`

When you reach this Web page, select *Products & Tools* and then select the latest release for more information on installing software or documentation with various IRIX releases.

2.2 Options for Running the System Verification Program

The System Verification Program offers a variety of options that control the operation of the program. These options include:

- Run all tests (diagnostics, and IRIX commands)
- Run diagnostics only
- Run `irsaudit`
- Run peripheral confidence tests
- Gather configuration report only
- Run IRIX command-level tests only
- Report in verbose mode
- Report version of SVP
- Disable memory test
- Set Field Mode (for IP21 and IP25 only)
- Set the run time
- Set the pass count (the number of times SVP will loop through the tests)

The details of running SVP with the various options are located in the SVP man page in Appendix A. The support engineer selects the appropriate options according to the needs of the site.

2.3 Starting the System Verification Program

To run SVP, execute the following command as root:

/usr/SVP/svp [*options*]

Refer to Appendix A for a description of SVP options.

The program prompts you to enter the following information before SVP will start the test process:

- Your name
- The name of the test site
- The reason for the test run (MFG / INS / UDP / REP)

  where MFG=manuafacturing, INS=installation, UPD=upgrade, and REP=repair or replacement of a part. Ensure that the manufacturing mode is never selected in the field. (In manufacturing, an automated process sends data back to the IB database. In the field, this process should be controlled by the support engineer.)

  Note: References to MFG do not apply to the Channel Support community.
- The other system name (the default localhost) for network testing

  If not localhost:

  - Login name (default guest)
  - Password to login
- The default localhost e-mail address (use the `whoami` command to determine the default), if the system has Mail set up
- Any optional one-line comment that you want logged
- Whether you want to transmit the SVP results automatically to SGI

Note: Always enter your name and the name of your workstation. If you do not know how to respond to the other requests, enter an empty field for each of them.

The System Verification Program should begin to run at this time.

Chapter 3

Viewing Test Results

This Chapter describes how to view test results and errors, analyze errors for probable causes, and report config / results to the home office.

3.1 Viewing Test Results, Configurations, and Errors

All SVP test results are stored in the file `/usr/SVP/RESULT/svp_date.LOG`. The results of `irsaudit` are stored in the file `/usr/SVP/RESULT/irsaudit_date.LOG`. The results of Digital Video diagnostics are stored in the file `/usr/SVP/RESULT/divo_date.LOG`. All errors are also logged into the file `/usr/SVP/RESULT/svp_date.ERROLOG`. You may view these files with your preferred editor (for example, vi).

All comments that you have input during the run are inserted with a date entry into the file `/usr/SVP/CONFIG/run_reason`. With IRIX operating system versions 6.4, 6.5 through 6.5.4, and 6.5.5; all configuration information is logged into the directory `/usr/SVP/CONFIG`. This directory contains the files `configdata_1.1`, `software_installed`, `patches`, and `mosaic_data.email`.

If you select the verbose option before you run SVP, the test results will indicate the actual command that is executed for each test. In both the verbose mode and the nonverbose mode, error messages provide detailed information about a failure.

IRIX command-level tests provide tips when a command cannot be run because the necessary external software is not available. These tips indicate which software is needed to run the particular command. Generally, a test is skipped if its prerequisite software is not installed.

Refer to Appendix B for an example of a nonverbose SVP report listing.

3.2 Analyzing Errors for Probable Causes

Following an analysis of the test results, SVP attempts to assign a probable cause to the failure of a particular test.

3.2.1 Memory Tests

The following error messages may appear during memory testing:

```
main memory test (all cpu's) failed
CPU # main memory test failed
ERROR-can't allocate buffer
```

The diagnostics attempt to load 85% less 5 MB of per-process system memory when the overall memory size is greater than 64 MB, or 80% less 5 MB of per-process memory when the overall memory size is less than or equal to 64 MB.

Note: If any other application is running, memory cannot be allocated to this SVP test. Ensure that no other application is running during this test.

An error may also occur if the test runs out of logical swap space. Check `/usr/adm/SYSLOG` to collect additional information about memory test errors.

If multiple, identical errors appear within a single application, determine whether this error is common to all CPUs for which the test is run. If so, then the `malloc` (memory allocation) test failed and a memory board is suspect. If the `tagram` test also fails and uses the same CPU as in the memory test, then the CPU is suspect. Swap CPUs and determine whether the failure moves to another slot.

```
ERROR - addr:AAAA exp:BBBB got CCCC
```

(where `AAAA`, `BBBB`, and `CCCC` are hexadecimal numbers)

This test uses the address-under-test in a Flippancies series to test the high-order bits. After the test writes data into each memory address, the data is regenerated and compared to the memory contents. A test failure usually indicates a problem with the memory board. However, if the failures occur only with an individual CPU, the CPU itself may be the cause.

(CHALLENGE systems only)

```
CPU # cache test failed
ERROR: data miscompare
At page XX, address AAAA data was BBBB should be CCCC
```

(where `AAAA`, `BBBB`, and `CCCC` are hexadecimal numbers)

This error occurs during the tag test. The test determines the total number of pages into which the allocated memory can be divided. Then the test writes four locations and their addresses in each page. The test compares data read from these pages to expected data patterns. If the failure is associated with one CPU, move that CPU to another slot and retest. If, however, multiple CPUs report a common failing address, the cause of the failure may be the memory board.

3.2.2 Cache Tests (CHALLENGE Systems Only)

```
ERROR: CPU # cache test failed
ERROR - memaddr.Bit: Addr XXXX XXXX XXXX XXXX
```

This test runs only when the field mode option (`-f`) is selected. The test writes two types of data patterns to memory: a fixed pattern and a moving inversion pattern. The contents of memory are read and compared to the written data. A difference between the written data and the read-out data could indicate a cache failure; therefore, the cache is cleared and the data pattern is read again. A persistent problem may be either a cache or memory failure. Swap the CPU to determine whether the failure is related to the CPU or memory.

3.2.3 Matrix Tests

Failures of the matrix test indicate a CPU- or FPU-related problem. The matrix test runs FORTRAN programs that solve multidimensional matrices. Typically, matrix test failures indicate that the CPU board is failing.

3.2.4 Disk Tests

A SCSI thrasher test stresses the SCSI subsystem, including the MSCSI and IO6 boards. The SCSI thrasher test forks multiple processes to the disks being tested. All disk reads and writes are synchronized by semaphores. When each subsystem finishes its random-seek write phase, it waits for all other processes to complete and then begins the read phase. The read data and the write data are compared. The test uses five data patterns.

The SCSI thrasher test generates the following output messages:

```
PASS: Diskthrasher Passed All Patterns
```

The preceding message states that the test completed successfully; it did not detect any errors.

```
ERROR: Diskthrasher Thrasher Failed. Check error.log for error
ERROR: Data ERROR on device_name
   Disk mismatched at disk offset xxxx Expected data xxxx Act. xxxx
   Disk mismatched at disk offset xxxx Expected data xxxx Act. xxxx
   Disk mismatched at disk offset xxxx Expected data xxxx Act. xxxx
   Disk mismatched at disk offset xxxx Expected data xxxx Act. xxxx
   Disk mismatched at disk offset xxxx Expected data xxxx Act. xxxx
```

These error messages indicate a read / write data compare error. Note that the disk offset value is listed in the error message.

```
Data transfer on disk # by CPU # failed
```

This error appears when one of the mounted disks fails during a data transfer. The test writes two `tar` images of the current system directory into two temporary directories and compares them. If the two images do not compare, the error message above is issued. This error could be caused by a failure of the disk drive, disk controller, or interconnecting cables.

3.2.5 Network Tests

The Network Thrasher test, a client/server model, stresses the network hardware on the IO6 board by sending large blocks of data to a remote system and reading the data back. The test compares the data that is returned with the data that was sent.

The Ethernet Thrasher test has the following prerequisites:

- You must have the IRIX operating system booted and running on the system that you are testing.
- You must have access to a functional remote system that is running the IRIX operating system.
- You must have disk space available on the system that you are testing for the creation of temporary files that this test uses. The amount of disk space required depends on the number of clients that you specify the test will fork.
- You must correctly set the `remote host` parameter for the system that you are testing.
- You must provide an IP address with a fully qualified hostname to the `/etc/hosts` file for the system that you are testing. For example:

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

- You must provide the account login name and the account password for the remote system.

This test generates the following output messages:

```
PASS: Network Thrasher passed
```

The test completed successfully; it did not detect any errors.

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

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

```
ERROR: Remote system is not responding. Cannot continue
```

The test failed with an error. This error occurs while the test attempts to determine whether the remote system is operational. The remote system may not be responding for many reasons (such as the network is not functioning, the remote system is down, or the test 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 (an IRIX system call; refer to `man fork(2)`). If this message appears, you should kill the forked processes on the remote system and the test system as follows:

1. On the remote system, enter **`killall server1`**.
2. On the Origin 3000 series system, enter **`killall c_server`** and then enter **`killall e_client`**.

3.2.6 HIPPI and ATM Tests

The LINC DMA test is called out by SVP in superuser mode. It transfers 8 KB of DMA data from the host to LINC0 on the network card (ATM or HIPPI), executes programmed I/O (PIO) reads to read back the data, and compares the transmit and receive host buffers to verify that the data was transferred correctly. The test repeats for LINC DMA engine 1 and LINC1 DMA engine 0.

3.2.7 CPU Instruction Test

The CPU instruction test (`torpedo`) is a floating-point unit (FPU) stress test that uses several standard floating-point algorithms to test the FPU in each CPU. It compares the results from all CPUs and reports any discrepancies. The `torpedo` test verifies basic FPU functionality (add, subtract, multiply, divide, and square root) before it performs complex operations (parallel operations, trigonometric operations, root findings, and matrix operations).

The `torpedo` test has the following prerequisites:

- You must have the IRIX operating system booted and running on the system that you are testing.
- You must be in superuser mode.

Refer to *Online Diagnostics*, part number 108-0286-002, for additional information about the `torpedo` test.

Chapter 4

SVP Configuration Information

This chapter applies only to systems that are running IRIX 6.4 through 6.5.4 operating systems (OS). For information on systems running IRIX 6.5.5 OS and above, refer to the *Embedded Support Partner (ESP) User Guide*, publication number 007-4065-004.

4.1 General Description

When SVP is run at the site, it builds a database on the system disk, which is a snapshot of the hardware configuration. This snapshot is different from the one that is created during the manufacturing process because overpack items are not yet installed on the system. The information gathered at the site is essential to enable SGI to evaluate service contract renewal, service calls to the Technical Assistance Center (TAC), and escalation calls for serious system problems.

Origin 2000 servers can provide additional detailed board-level information such as part numbers, serial numbers, and revisions, which are embedded electronically in the NICs (Number In a Can). SVP takes advantage of the availability of this data and includes it in the configuration database.

As system configurations evolve because of fault isolation or upgrades, SVP can archive the history of these changes whenever it is run. This history is stored in directory `/usr/SVP/CONFIG` in the following files:

- `configdata_1.1` for hardware information
- `patches` for patch information
- `mosaic_data.email`

  These files contain hardware and test results packaged for transmission.

4.2 Configdata

The output of `configdata_1.1` is a set of tables instead of the single table that `configdata` provides. Tables generated by `configdata_1.1` provide information about node board memories, disk drives, graphic pipes, and RAID units.

Most of the table fields are self-explanatory. The Flag ID field in the earlier version of `configdata` has been renamed Deinstall; it indicates the date on which a board or disk was deinstalled. Items that have never been deinstalled are marked as xxxxxxxxxx.

If any part listed is missing information, the corresponding table field (for example, Part #) will contain NA instead of ???????? as in the previous version of `configdata`. Table 4-1 through Table 4-5 are examples of the configuration data that SVP collects.

Table 4-1 System Information

System Serial#: K0002056

Mod Sl#	Mod#	Slot #	Sl#	Part#	Rev	Deinstall	Install	Type
K0005150	1	io5	DYJ729	030-0927-002	B	xxxxxxxxxx	1997/07/08	FIBRE-CHANNEL
K0005150	1	io6	ENZ658	030-0927-002	B	xxxxxxxxxx	1997/07/08	FIBRE-CHANNEL
K0005150	1	n1	DAR081	013-1839-001	D	xxxxxxxxxx	1997/07/08	4PIG5_MPLN
K0005150	1	n1	DBA334	030-0733-033	J	xxxxxxxxxx	1997/07/08	IP27

Table 4-2 Memory Information

Module Serial #	Part Serial #	Module	Slot	NodeID	Memory Size	Enable
K0005150	DBA334	1	n1	0	128	Y

Table 4-2 is not generated for Origin 200 and Octane systems.

Table 4-3 Disk Information

Module Serial #	Module Number	Part Serial #	Slot #	Rev	Disk ID	Disk Mfg	Disk Serial #	Deinstall	Install
K0002788	1	DFM450	io1	3232	1	IBM DCHS09Y	13059385RAMSG032	xxxxxxxxxx	1997/07/18
K0002788	1	DFM490	io1	3232	1	IBM DCHS04Y	68113330RAMSG032	xxxxxxxxxx	1997/07/18
K0002780	2	CJK376	io1	3232	1	IBM DCHS09Y	68022350RAMSG032	xxxxxxxxxx	1997/07/18
K0002169	3	DBX398	io1	3232	1	IBM DCHS04Y	68032034RAMSG032	xxxxxxxxxx	1997/07/18

Table 4-4 Fibre Channel RAID Information

Module Serial #	Module	Ctrl	Part Serial #	Slot #	Rev	Disk ID	Disk Mfg	Disk Serial #	Deinstall	Install
K0005150	1	2	DYJ722	io3	SG79	9	SEAGATE ST19171F	LA142176	1997/07/11	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	0	SEAGATE ST19171F	LA373435	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	1	SEAGATE ST19171F	LA382977	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	2	SEAGATE ST19171F	LA209858	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	3	SEAGATE ST19171F	LA421777	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	4	SEAGATE ST19171F	LA059195	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	5	SEAGATE ST19171F	LA393473	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	6	SEAGATE ST19171F	LA407193	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	7	SEAGATE ST19171F	LA407935	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	8	SEAGATE ST19171F	LA413783	xxxxxxxxxx	1997/07/11
K0005150	1	2	DYJ722	io3	SG79	9	SEAGATE ST19171F	LA409355	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	7	SEAGATE ST19171F	LA406376	1997/07/11	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	0	SEAGATE ST19171F	LA416044	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	1	SEAGATE ST19171F	LA445367	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	2	SEAGATE ST19171F	LA415326	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	3	SEAGATE ST19171F	LA447744	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	4	SEAGATE ST19171F	LA407192	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	5	SEAGATE ST19171F	LA410097	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	6	SEAGATE ST19171F	LA408613	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	7	SEAGATE ST19171F	LA416793	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	8	SEAGATE ST19171F	LA432117	xxxxxxxxxx	1997/07/11
K0005150	1	3	DYJ722	io3	SG79	9	SEAGATE ST19171F	LA417004	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	9	SEAGATE ST19171F	LA409355	1997/07/11	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	0	SEAGATE ST19171F	LA378815	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	1	SEAGATE ST19171F	LA403398	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	2	SEAGATE ST19171F	LA402907	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	3	SEAGATE ST19171F	LA407927	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	4	SEAGATE ST19171F	LA404461	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	5	SEAGATE ST19171F	LA416592	xxxxxxxxxx	1997/07/11

Table 4-4 (continued) Fibre Channel RAID Information

Module Serial #	Module	Ctrl	Part Serial #	Slot #	Rev	Disk ID	Disk Mfg	Disk Serial #	Deinstall	Install
K0005150	1	4	ENZ658	io6	SG79	6	SEAGATE ST19171F	LA395016	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	7	SEAGATE ST19171F	LA407862	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	8	SEAGATE ST19171F	LA249785	xxxxxxxxxx	1997/07/11
K0005150	1	4	ENZ658	io6	SG79	9	SEAGATE ST19171F	LA142176	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	0	SEAGATE ST19171F	LA389380	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	1	SEAGATE ST19171F	LA405702	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	2	SEAGATE ST19171F	LA142912	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	3	SEAGATE ST19171F	LA396993	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	4	SEAGATE ST19171F	LA395790	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	5	SEAGATE ST19171F	LA211324	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	6	SEAGATE ST19171F	LA407599	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	7	SEAGATE ST19171F	LA413533	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	8	SEAGATE ST19171F	LA397131	xxxxxxxxxx	1997/07/11
K0005150	1	6	DYJ729	io5	SG79	9	SEAGATE ST19171F	LA406376	xxxxxxxxxx	1997/07/11

Table 4-5 Graphic Pipes Information

Module of Pipe#	Graphics Pipe SI#	Serial	Part Number	Rev	Type	Deinstall Date	Install Date
K0009813	EZC646	FSY432	030-1184-001	E	GV0	xxxxxxxxxx	1998/01/02
K0009813	EZC646	EDY463	030-1054-001	F	RM7	xxxxxxxxxx	1998/01/02
K0009813	EZC646	EYP971	030-1242-001	C	DG5-2	xxxxxxxxxx	1998/01/02
K0009813	EZC646	EDW278	030-1053-001	E	TM7-64	xxxxxxxxxx	1998/01/02

Patch Information, as in the following example, is available at `/usr/SVP/CONFIG/patches`. The Install/Remove Flag field may contain either I or R, but the Installation Date field always indicates when a patch is installed, regardless of the flag setting.

Table 4-6 Patch Information

Patch #	Install/ Remove Flag	Installation Date	Description
SG0001809	I	07/07/97	SpeedShop 1.1 Patch
SG0001820	I	02/20/97	specfs patch for IRIX 6.4
SG0001821	I	02/20/97	fixes to hwgfs (mount, link counts, pathconf)
SG0001868	I	05/23/97	XFS patch #1 for IRIX 6.4
SG0001943	I	05/29/97	6.4 S2MP+Octane IP27/BASEIO proms
SG0001978	I	07/08/97	6.4 kernel rollup
SG0002061	I	05/29/97	PCI Rollup
SG0002073	I	07/07/97	SCSI tape
SG0002118	I	07/07/97	6.4 SCSI rollup
SG0002173	I	07/07/97	Hinv rollup for 6.4.1
SG0002175	I	06/27/97	fibre channel update

4.3 Mosaic_data

During SVP execution, the configuration data and the results are packaged into a data format that is sent to the corporate IB database. The packaged information that will be e-mailed to SGI is stored in a file named `mosaic_data.email` at `/usr/SVP/CONFIG`. The following text illustrates a typical file:

```
#--- begin data ---
MSG_TYPE=4501
||char::Sn::K0009813::
||char::ModuleSerialNumber::K0009813::
||char::ModuleId::1::
||char::SlotNo::io1::
||char::PartSerialNumber::ESL146::
||char::PartNumber::030-0734-002::
||char::PartRev::J::
||char::CaseId::::
||char::DateRun::19980102::

MSG_TYPE=4501
||char::Sn::K0009813::
||char::ModuleSerialNumber::K0009813::
||char::ModuleId::1::
||char::SlotNo::io1::
||char::PartSerialNumber::EYZ086::
||char::PartNumber::030-0880-003::
||char::PartRev::E::
||char::CaseId::::
||char::DateRun::19980102::

MSG_TYPE=4503
||char::Sn::K0009813::
||char::ModuleSerialNumber::K0009813::
||char::PartSerialNumber::DKH301::
||char::SlotNo::io1::
||char::node-id::0::
||char::mem_size::1024::
||char::mem_enabled::Y::

MSG_TYPE=4504
||char::Sn::K0009813::
||char::ModuleSerialNumber::K0009813::
||char::PartNumber::030-0880-003::
||char::PartSerialNumber::EYZ086::
||char::SlotNo::io1::
||char::disk_rev::5252::
||char::disk_id::1::
||char::disk_manf::IBM DCHS09Y::
||char::disk_serial_number::680778A3RAMSG052::
||char::DateRun::19980102::

MSG_TYPE=4505
||char::Sn::K0009813::
||char::patch_number::SG0001776::
||char::description::Origin FRU analyzer fix::
||char::DateRun::08/27/97::
```

```
MSG_TYPE=4505
||char::Sn::K0009813::
||char::patch_number::SG0001975::
||char::description::gmemusage security hole::
||char::DateRun::12/18/97::

MSG_TYPE=4502
||char::Sn::K0009813::
||char::TestNumber::0::
||char::DateCreated::19980102::
||char::TestType::INS::
||char::ResultFlag::F::
||char::NumberOfTest::84::
||char::Comments:::
#--- end data ---
MSG_TYPE=4506
||char::Sn::K0009813::
||char::ModuleSerialNumber::K0009813::
||char::PartSerialNumber::EZC646::
||char::GraphicsSerialNumber::FSY432::
||char::PartNumber::030-1184-001::
||char::PartRev::E::
||char::DateRun::19980102::
```

Each `MSG_TYPE` indicates a record in the database. The `MSG_TYPE` also defines the type of data being sent back:

- `MSG_TYPE 4501` is a hardware part type
- `MSG_TYPE 4502` is a result type
- `MSG_TYPE 4503` is for memory
- `MSG_TYPE 4504` is for disks
- `MSG_TYPE 4505` is for patches
- `MSG_TYPE 4506` is for component serial numbers

If all tests pass, only one record is sent back; it contains the number of tests that passed. If any failure is observed, the “number of tests passed” record indicates how many of the tests passed. There is also one additional record for each failed test, which indicates the type of test that failed.

This information is e-mailed to the Mosaic database automatically if you chose that option at test setup; or you may e-mail it manually by entering the following command:

```
mail svp@corp.sgi.com</usr/SVP/CONFIG/mosaic_data.email
```

The Mosaic database is updated nightly. Data received from the site is available the following morning. Web sites that are located within the SGI firewall enable SSEs to view this data. The Web addresses for the configuration data and the test results are as follows:

`http://illiad.corp.sgi.com/mosaic_queries/svp_config.html`
and
`http://illiad.corp.sgi.com/mosaic_queries/svp_testresult.html`

Refer to the following figures for examples of Mosaic database query displays.

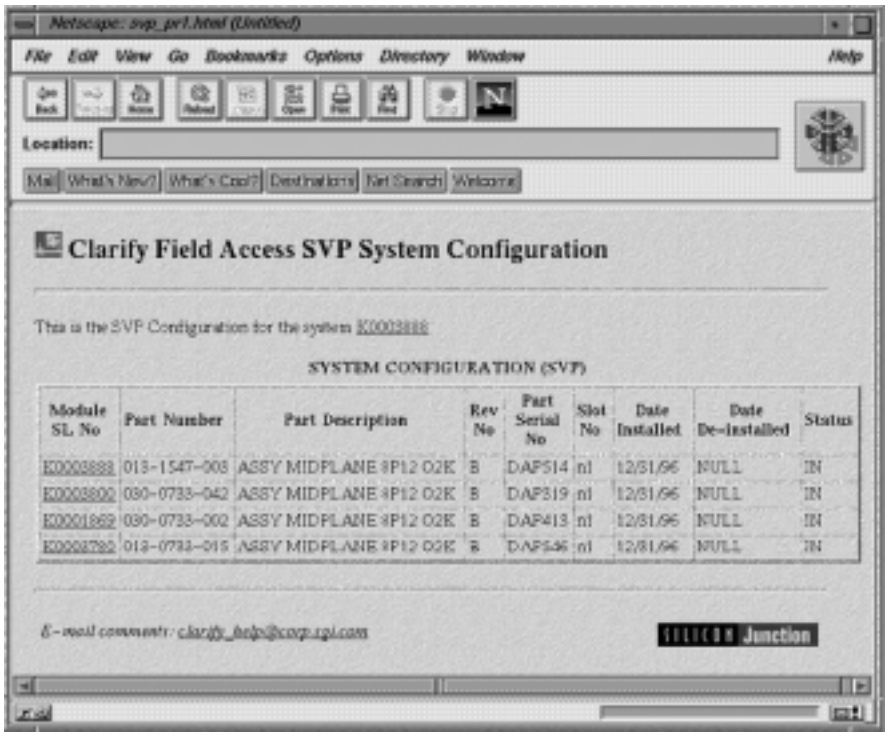

Figure 4-1 SVP Configuration Web Page

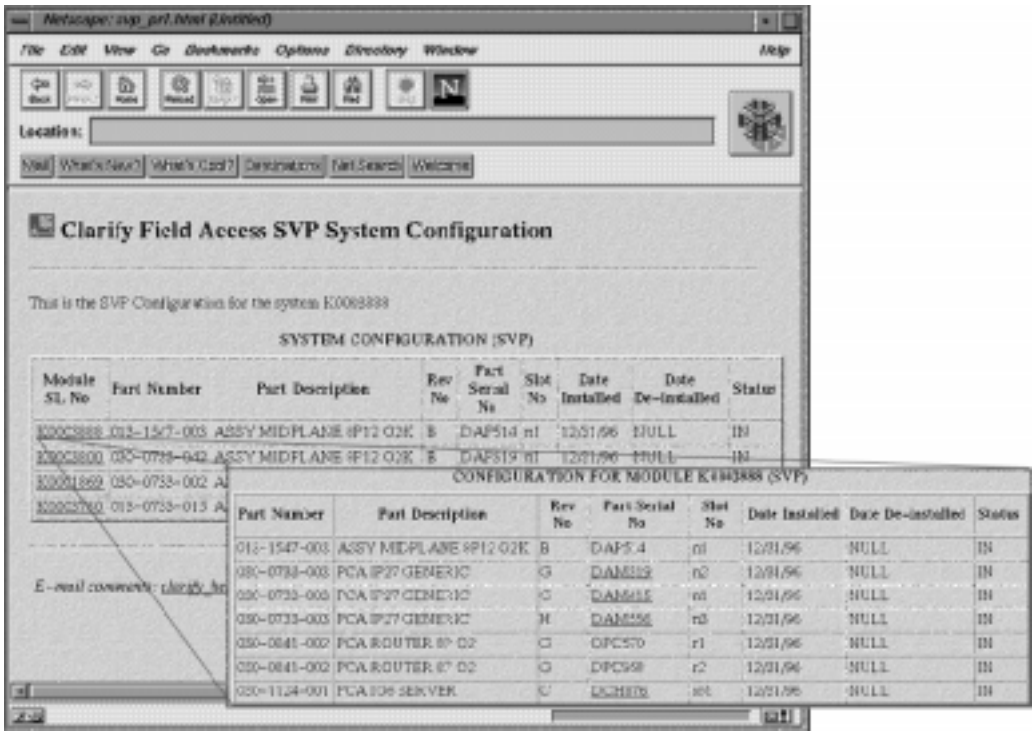

Figure 4-2 Module Configuration Screen

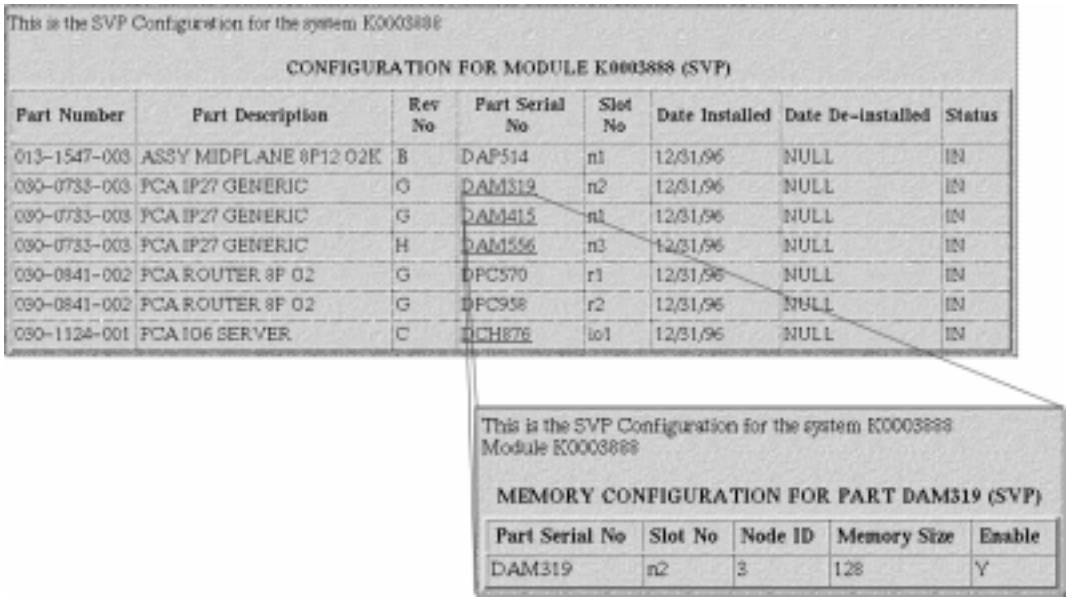

Figure 4-3 Memory Configuration Screen

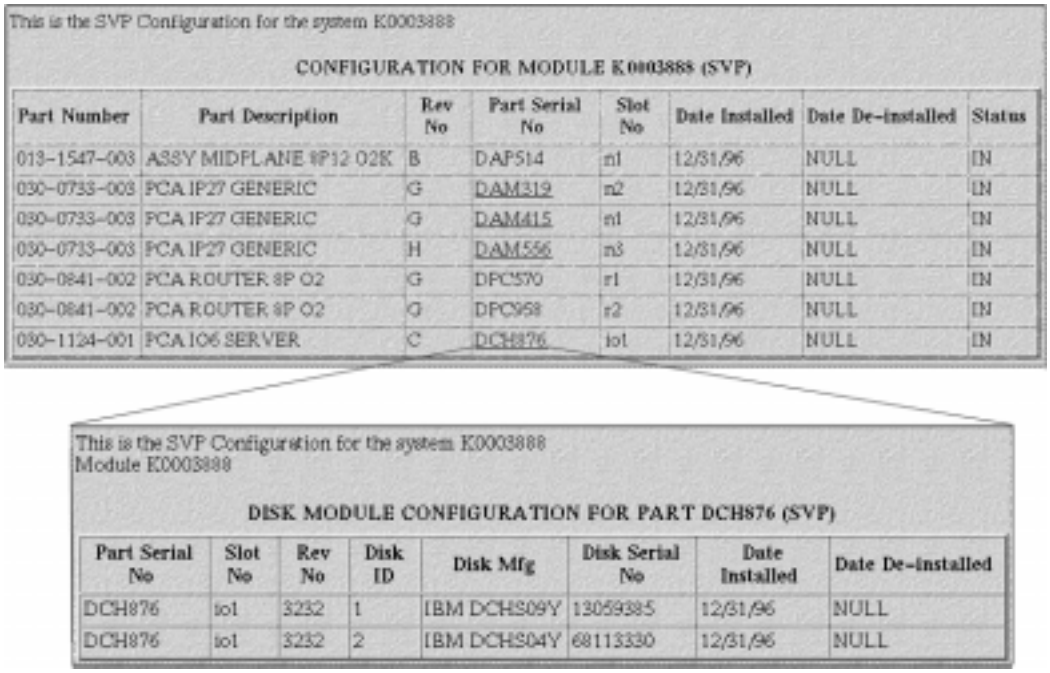

Figure 4-4 Disk Module Configuration Screen

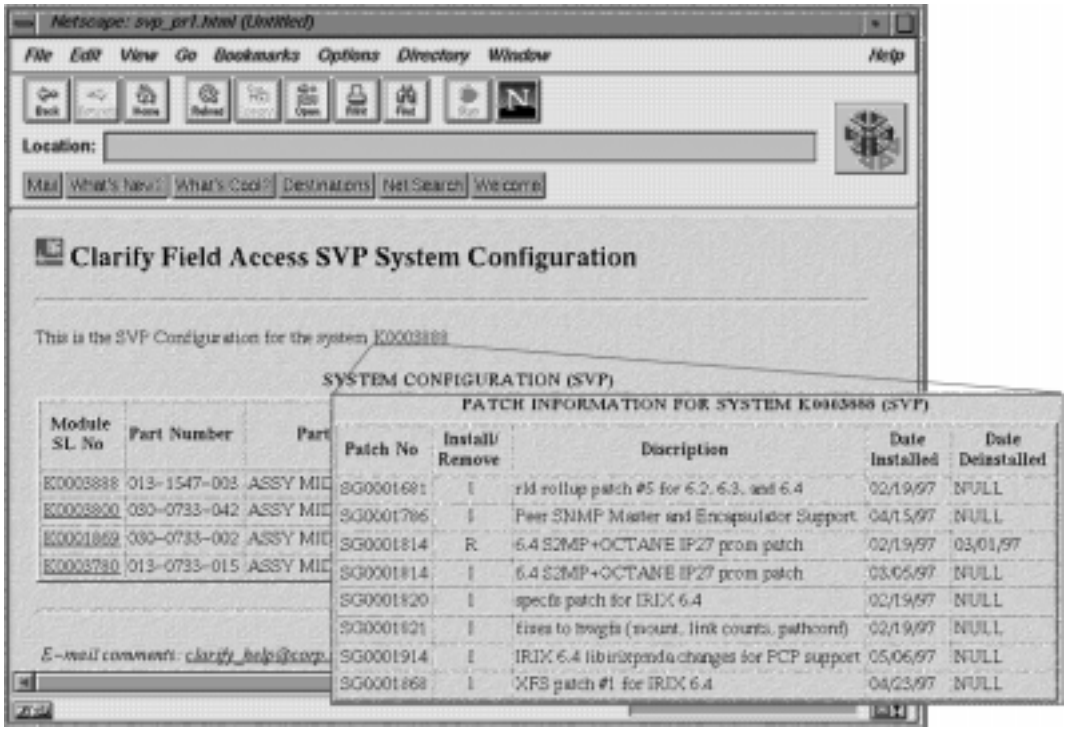

Figure 4-5 Patch Information Screen

GRAPHIC CONFIGURATION FOR PIPE ESA422 (SVP)

Part Serial Number	Graphic Part	Graphic Serial No	Revision	Date Installed	Date De Installed
ESA422	080-1054-001	EDW084	E	Aug 28 1997 12:00:00:000AM	NULL
ESA422	080-1053-001	EDW105	E	Aug 28 1997 12:00:00:000AM	NULL
ESA422	080-1053-001	EDW283	E	Aug 28 1997 12:00:00:000AM	NULL
ESA422	080-1054-001	EDW361	E	Aug 28 1997 12:00:00:000AM	NULL
ESA422	080-1054-001	EDW456	F	Aug 28 1997 12:00:00:000AM	NULL
ESA422	080-1054-001	EDW458	F	Aug 28 1997 12:00:00:000AM	NULL
ESA422	080-1053-001	EEA252	E	Aug 28 1997 12:00:00:000AM	NULL
ESA422	080-1053-001	EEA263	E	Aug 28 1997 12:00:00:000AM	NULL
ESA422	080-1087-001	ENG749	H	Aug 28 1997 12:00:00:000AM	NULL
ESA422	013-1760-001	K43624	A	Aug 28 1997 12:00:00:000AM	NULL

Figure 4-6 Graphic Pipe Information Screen

This is the SVP Test Result for the system K0002608

SVP TEST RESULT

Test No	Run Date	Test Type	Result	No of test	Comments
0	12/28/1996	MFG	P	51	Final Configuration Test

Figure 4-7 SVP Test Results

Appendix A

SVP man Page

```
svp(1)                                                          svp(1)

NAME
     svp - System Verification Program

SYNOPSIS
     svp [ -bCdfhIMuvV ] [ -l Loopcount ] [ -m Duration ] [ -o Logfile ]

DESCRIPTION
     svp is an IRIX based tool that is designed to help support engineers
     verify the basic functionality of high-end systems during installation at
     the customer site.  svp is a test suite that comprises peripheral
     confidence tests, board-level diagnostics, and IRIX command tests.  The
     board-level diagnostics exercise the CPU, FPU, memory, I/O,
     multiprocessor graphics (KONA only), and video (DIVO).  The IRIX command
     tests cover a wide range of commands, graphics, network and some
     application packages.  The test results are displayed as well as saved in
     a logfile.  A history of hardware changes is recorded under
     /usr/SVP/CONFIG whenever svp is run.  Refer to the SVP Users Guide for
     information about testing your configuration.  Errors are logged in
     svp_date.ERRORLOG under /usr/SVP/RESULT.  svp logs an "SVP start" event
     shortly after execution begins and an "SVP end" event just before
     exiting.  Test results are also logged into the ESP framework.

     To run svp, su to root and execute:

          /usr/SVP/svp [ options ]

     Options are:

     -b   Disable UNIX tests and execute only the board-level diagnostics

     -c   Generate configuration data without running tests (IRIX
          6.4 - 6.5.4 only)

     -C   Run peripheral confidence tests.  Confidence tests are interactive
          and must be exited using the exit option in the file menu before svp
          continues to run.  Confidence tests are run only once, regardless of
          any command line arguments to loop.

     -d   Enable shell xtrace (debug) output.  If this option is used, it is
          recommended that svp be invoked from an sh-style shell and that
          stderr be redirected to a file.  This option is intended for testing
          and troubleshooting problems with svp itself.  It is not
```

```
          recommended for field use.

     -f   Run cache tests (IP21 and IP25 systems only)

     -h   List usage

     -I   Run irsaudit diagnostics for InfiniteReality systems (unless the -u
          option is also used)

     -l Loopcount
          Run Loopcount loops of the svp tests.  If used with the -m option,
          svp exits when the earliest condition (loops or duration) is
          satisfied.  svp will run no more than one loop if the -l option is
          not used.

     -M   Disable memory tests

     -m Duration
          Run svp for Duration minutes.  If the -l option is also used, svp
          exits when the earliest condition (loops or duration) is satisfied.
          By default, the duration of an svp run is not limited.

     -o Logfile
          Log test results to Logfile instead of /usr/SVP/RESULT/svp_date.LOG.
          ("date" is replaced by the date and time of the svp invocation.)

     -u   Disable board-level diagnostics and execute only the UNIX command
          tests

     -v   Enable verbose mode.  In the verbose mode, test information is more
          detailed.  This option displays the paths of all commands that are
          executed.  For board-level diagnostics, detailed information on the
          CPU and other components is displayed.  If a test fails, details of
          the failure are displayed on the terminal and logged into the
          logfile.  By default, only minimal information regarding the tests
          is displayed.

     -V   Print svp version information

EXAMPLE 1
          /usr/SVP/svp -v -b -l 4 -m 120 -o SVP.log

     This example runs board level diagnostics only.  The mode settings are
     verbose mode with a loop count of 4 and a maximum runtime of 120 minutes.
     The results will be logged in SVP.log in the current directory.

EXAMPLE 2
          /usr/SVP/svp

     This example uses all default values.  The test will run board
     diagnostics and IRIX command tests.  Verbose mode is off and only minimal
     test information is provided.  The loop count is set to one and no
     specific time limit imposed on the test.  The result will be logged in
     /usr/SVP/RESULT/svp_date.LOG.  Errors will be logged in
     /usr/SVP/RESULT/svp_date.ERRORLOG.

NOTE
     svp should not be run concurrently with any user processes which consume
```

non-trivial amounts of system resources. If this should occur, svp may report incorrect information or it may be denied the resources it needs to successfully test the system.

Memory tests are not supported on IP17, IP20, IP22, IP26 or IP28 systems.

SEE ALSO

irsaudit(1) (infiniteReality systems only), divotest(1), ESP(1)

Appendix B

Example of an SVP Report

```
        ******************************************
        *                                        *
        *                                        *
        *             SGI SVP Test Report        *
        *                                        *
        *                                        *
        ******************************************

        Test start time:        Mon May 18 13:40:35 PDT 1998
        Tester name:            nadia
        Test site name:         ist
        Case id:
        System serial number:   K0002780
        Machine name:           bootleg
        Machine type:           IP27 mips
        Network addr:           150.166.14.31
        Host name:              bootleg
        Operating system:       IRIX64 6.4 Version# 02121744
        svp versions:           Version 1.2
        svp test options:       ./svp


  *** Disk Inventory Information: ***


DETECT MOUNTED FILESYSTEMS
dks0d1s0
dks4d6s7

3 NOT MOUNTED
dks0d2
dks16d1
dks4d1

# ftn77_eoe.sw.lib - NOT INSTALLED, some tests will be skipped
# ftn90_eoe.sw.lib - NOT INSTALLED, some tests will be skipped
# showcase.sw.showcase - NOT INSTALLED, some tests will be skipped

  ************************
  *   Board Level Tests   *
  ************************
```

```
*1   Test Program: Matrix 300X300 SP         .............Passed

*2   Test Program: Matrix 300X300 DP         .............Passed

*3   Test Program: Matrix 1000X1000 SP       .............Passed

*4   Test Program: Matrix 1000X1000 DP       .............Passed

*5   Test Program: MP Matrix 1000X1000 DP    .............Passed

*6   Test Program: System Disk Test          .............Passed

*7   Test Program: Mounted Disks Tests       .............Passed

*8   Test Program: Router Test               .............Passed

*9   Test Program: HIPPI Test                .............Passed

*10  Test Program: SCSI Thrasher             .............Passed

*11  Test Program: Ethernet Thrasher         .............Passed


     ************************
     *   UNIX Level Tests    *
     ************************


*12  Test Program: C compiler and make       .............Passed

*13  Test Program: /usr/bin/strip            .............Passed

*14  Test Program: C compiler -n32           .............Passed

*15  Test Program: C compiler -64            .............Passed

*16  Test Program: /usr/bin/size             .............Passed

*17  Test Program: /usr/bin/nm               .............Passed

*18  Test Program: /usr/bin/pixie            .............Passed

*19  Test Program: C++ compiler              .............Passed

*20  Test Program: sn -a                     .............Passed

*21  Test Program: /bin/df -k                .............Passed

*22  Test Program: /sbin/ps                  .............Passed

*23  Test Program: /usr/bin/cat              .............Passed

*24  Test Program: /sbin/nice date           .............Passed

*25  Test Program: /usr/bin/sync             .............Passed
```

```
*26  Test Program: /usr/bsd/whereis          .............Passed
*27  Test Program: /usr/bsd/finger           .............Passed
*28  Test Program: /usr/bsd/which            .............Passed
*29  Test Program: /usr/bsd/hostid           .............Passed
*30  Test Program: /usr/bsd/last             .............Passed
*31  Test Program: /sbin/who                 .............Passed
*32  Test Program: /bin/whatis               .............Passed
*33  Test Program: mpadmin -s                .............Passed
*34  Test Program: /usr/sbin/ipcs            .............Passed
*35  Test Program: /etc/fuser                .............Passed
*36  Test Program: dircmp                    .............Passed
*37  Test Program: /usr/sbin/runon           .............Passed
*38  Test Program: /usr/bin/time             .............Passed
*39  Test Program: /usr/bin/timex            .............Passed
*40  Test Program: ls -l /unix               .............Passed
*41  Test Program: /bin/chmod                .............Passed
*42  Test Program: /bin/chown                .............Passed
*43  Test Program: /usr/bin/file             .............Passed
*44  Test Program: /usr/bin/more             .............Passed
*45  Test Program: man                       .............Passed
*46  Test Program: id                        .............Passed
*47  Test Program: whoami                    .............Passed
*48  Test Program: /bin/sort                 .............Passed
*49  Test Program: /bin/find                 .............Passed
*50  Test Program: /bin/tail                 .............Passed
*51  Test Program: /usr/bin/domainname       .............Passed
*52  Test Program: /usr/bin/rup              .............Passed
*53  Test Program: /usr/sbin/pwck            .............Passed
```

```
*54  Test Program: /sbin/tar without tape    .............Passed

*55  Test Program: /sbin/cpio without tape   .............Passed

*56  Test Program: /sbin/dd without tape     .............Passed

*57  Test Program: /usr/sbin/bru without tape.............Passed

*58  Test Program: /usr/etc/ping             .............Passed

*59  Test Program: /usr/etc/netstat          .............Passed

*60  Test Program: /usr/etc/nfsstat          .............Passed

*61     Test Program: /usr/bsd/rcp 5 times   .............ERROR

*61     Test Program: /usr/bsd/rcp 5 times   .............ERROR

*61     Test Program: /usr/bsd/rcp 5 times   .............ERROR

*61     Test Program: /usr/bsd/rcp 5 times   .............ERROR

*61     Test Program: /usr/bsd/rcp 5 times   .............ERROR

*62     Test Program: mount -a               .............ERROR

Test 62 : mount -a                    return failed got 2 should be 0
*63  Test Program: /bin/mail                .............Passed

*64  Test Program: /usr/sbin/Mail           .............Passed


     ************************************************
     All Automated Tests Completed
     Important!! For FAILURE REPORT please check
     /usr/SVP/RESULT/svp_05_18_98.ERRLOG
     check the following operations:
     autoconfig
     reboot
     /etc/halt
     /etc/init.d/network stop
     /etc/init.d/network start
     fsck certain file system
     test vi or emacs
     o************************************************
     End Test Time: Mon May 18 13:57:07 PDT 1998

     TEST RESULTS: Pass/61 Fail/2 Total/64
     TOTAL TEST TIME: 18 minutes
```