Sun Fire™ V20z Server—
Server Management Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 817-5249-10
817-5249-10, Revision A

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

Please
Recycle

Preface

This user guide explains the ways in which a user can manage the Sun Fire™ V20z server.

How This Book Is Organized

Chapter 1 provides an overview of the ways in which a user can manage the Sun Fire V20z server. See “Introduction” on page 1.

Chapter 2 describes how to manage the Sun Fire V20z server through the Intelligent Platform Management Interface (IPMI). See “IPMI Server Management” on page 19.

Chapter 3 describes how to manage the Sun Fire V20z server through the Simple Network Management Protocol (SNMP). See “SNMP Server Management” on page 33.

Chapter 4 describes the Sun Control Station (SCS) software. See “Sun Control Station” on page 45.

Chapter 5 provides further management information, such as how to enable Console Redirection over Serial on a Linux-based server and enable Serial-over-LAN. See “Further Management Information” on page 49.

Appendix A describes the command-line-interface (CLI) commands mentioned in the Sun Fire V20z server documentation. See “Commands” on page 57.

Using UNIX Commands

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

- Software documentation that you received with your system
- Solaris™ operating system documentation, which can be found at

  `http://docs.sun.com`

Related Documentation

Application	Title	Part Number
Installation	*Sun Fire V20z Server Installation Guide*	817-5246-xx
Installation	*Sun Fire V20z Server Linux Operating-System Installation Guide*	817-5250-xx
Service	*Sun Fire V20z Server User Guide*	817-5248-xx

Accessing Sun Documentation

You can view, print, or purchase a broad selection of Sun documentation, including localized versions, at:

`http://www.sun.com/documentation`

Third-Party Web Sites

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

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in this document, go to:

`http://www.sun.com/service/contacting`

Sun Welcomes Your Comments

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

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

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

Sun Fire V20z Server—Server Management Guide, part number 817-5249-10

Contents

CHAPTER 1

Introduction

Overview

Strong server-management capabilities are crucial to maintaining mission-critical servers. Advance notification of problems and rapid diagnosis and correction are critical functions to an environment in which a few servers bear the bulk of the workload. The Sun Fire™ V20z server and its extensive server-management capabilities lower costs by reducing failure and potentially eliminating hands-on management.

This document describes how to perform remote management on the Sun Fire V20z server.

The Sun Fire V20z server is an AMD Opteron processor-based, enterprise-class one-rack-unit (1U), two-processor (2P) server. The Sun Fire V20z server provides performance and value to an enterprise environment, offering significantly better performance than current 32-bit Intel-based solutions. The AMD Opteron processor implements the x86-64-bit architecture, which delivers significant memory capacity and bandwidth with twice the memory capacity and up to three times the memory bandwidth of existing x86-32-bit servers. The balanced server design maximizes overall performance through industry-leading I/O options, and delivers compelling, real-world, workload performance.

The Sun Fire V20z server includes an embedded Service Processor (SP), flash memory, RAM, a separate Ethernet interface and server-management software. It comes equipped with superior server-management tools for greater control and minimum total cost of ownership. You can use the command-line interface (CLI) or SNMP integration with third-party frameworks to configure and manage the platform with the SP. The dedicated SP provides complete operating-system independence and maximum availability of server management.

Acronyms

TABLE 1-1 explains the acronyms found in this document.

TABLE 1-1 Acronyms

Acronym	Explanation
ACPI	Advanced Configuration and Power Interface
ARP	Address Resolution Protocol
BMC	Baseboard Management Controller
CRU	Customer-Replaceable Unit
DPC	Direct Platform Control
FRU	Field-Replaceable Unit
grub	Grand Unified Bootloader
IPMI	Intelligent Platform Management Interface
KCS	Keyboard Controller Style
KVM	Keyboard, video and monitor
LAN	Local Area Network
LILO	Linux Loader
LOM	Lights Out Management
MIB	Management Information Base
RMCP	Remote Management Control Protocol
SCS	Sun Control Station
SDR	Sensor Data Record
SEL	System Event Log
SNMP	Simple Network Management Protocol
SOL	Serial Over LAN
SP	Service Processor
SSU	System Setup Utility
SunMC	Sun Management Center
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
WAN	Wide Area Network

Server Management

There are several options for remotely managing a Sun Fire V20z server:

- Lights Out Management (LOM) through IPMItool
- Simple Network Management Protocol (SNMP)
- Sun Control Station (SCS)

Service Processor

The Sun Fire V20z server includes a dedicated chipset for complete operating-system independence and maximum availability of server-management functions. This chipset, called Service Processor (SP), is an embedded PowerPC chip providing the following:

- Environmental monitoring of the platform (such as temperatures, voltages, fan speeds, panel switches)
- Alert messages when problems occur
- Remote control of server operations (boot, shutdown and reboot of the server's operating system, turning the server's power on and off, stopping the server's boot process in BIOS, upgrading the BIOS)

Note – In this document, you might see references to a Baseboard Management Controller (BMC).

A BMC is a dedicated IPMI controller. The SP found in the Sun Fire V20z server is a general-purpose, embedded CPU that contains software to emulate a BMC.

Server-Management Interfaces

The Sun Fire V20z server includes remote server-management capabilities through the SP; the SP supports four server-management interfaces:

- IPMI using a Keyboard Controller Style (KCS) interface and an IPMI kernel driver (in-band)
- IPMI over local area network (LAN) (out-of-band)
- SNMP integration with third-party SNMP management consoles
- Command-line-interface (CLI) LOM

Command Line Interface

Server-management capabilities are available from the command line.

See Appendix A for a list of server-management commands that you can use with the Sun Fire V20z server, as well as a description, the command format, a list of arguments and a list of return codes for each command.

Scripting and SSH Capabilities

An administrator can log in to the SP using an `ssh` client and issue commands or, more commonly, write a shell script that remotely invokes these operations.

The server-management commands enable you to manage efficiently each area of the server. From the command line, you can write data-driven scripts that automate the configuration of multiple machines. For example, a central management system can cause many servers to power on and boot at a specified time or when a specific condition occurs.

SNMP Integration

SNMP management provides remote access by SNMP-compliant entities to monitor the health and status of the Sun Fire V20z server. The SP sends SNMP alerts to external management functions when warranted.

For more information about SNMP, refer to “SNMP Server Management” on page 33.

The diagram in FIGURE 1-1 illustrates the communications paths for the different server-management options.

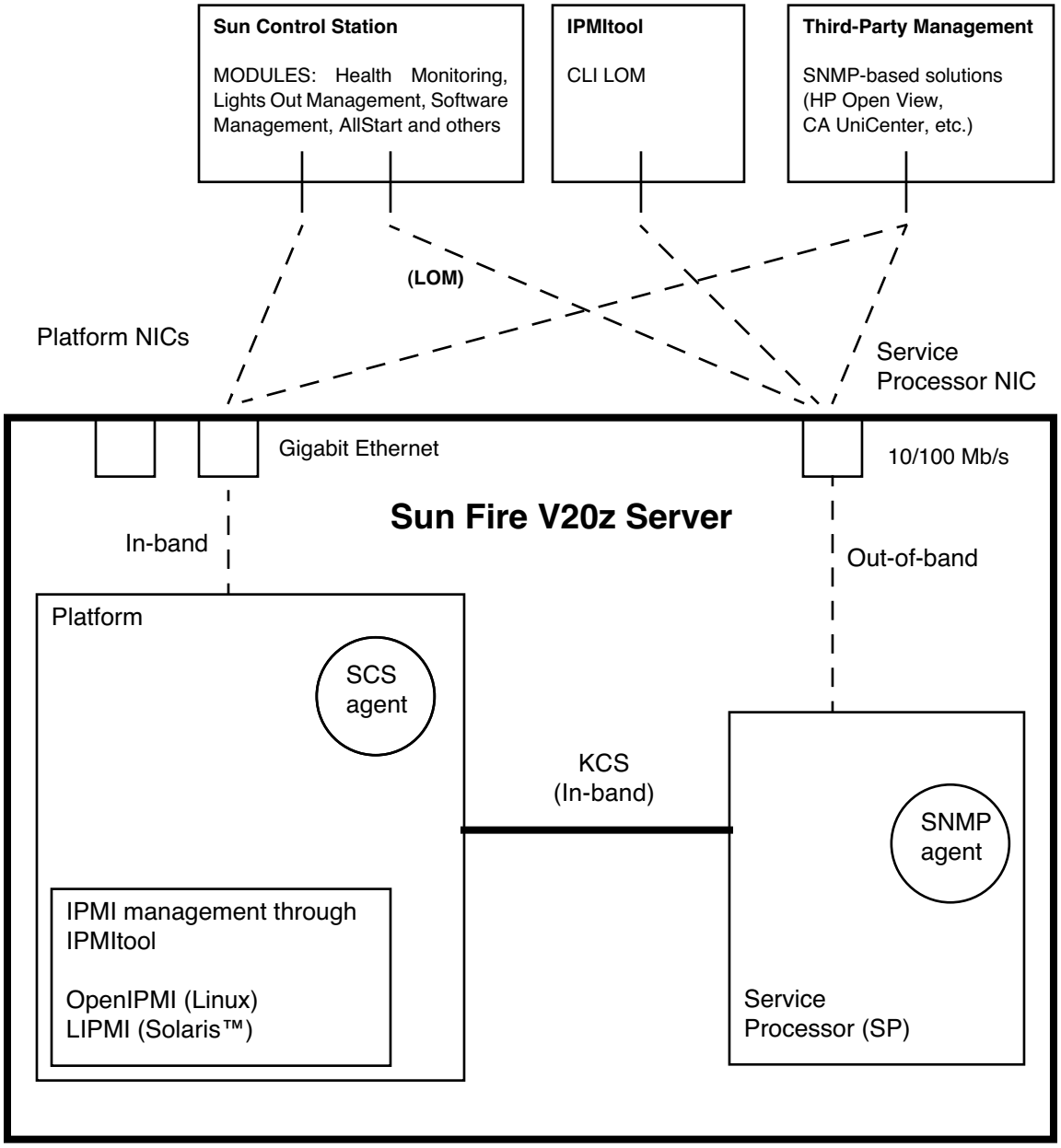

FIGURE 1-1 Diagram of the Server-Management Options

Operator Panel

You can use the operator panel to configure network settings for the SP. See FIGURE 1-2.

Note – The SP defaults to Dynamic Host Configuration Protocol (DHCP) networking if the operator panel is not interactively engaged on the first power-up.

Note – For more information about the operator panel, refer to the *Sun Fire V20z Server User Guide*, 817-5248-xx.

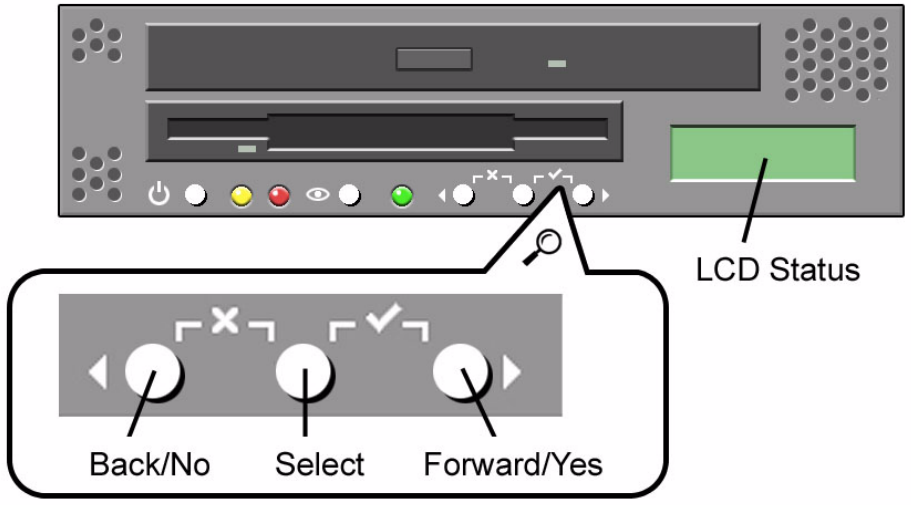

FIGURE 1-2 Operator Panel and Buttons

The operator panel displays information on the LCD in two lines, and you respond to prompts or initiate actions using the following buttons:

TABLE 1-2 Operator Panel Buttons

Buttons	Function
	Back/No
	Select
	Forward/Yes
	Enter
	Cancel

If a menu or data-entry screen displays for more than 30 seconds with no action taken, the menu or data entry is cancelled and the display returns to the idle/background state.

For every action that you confirm, feedback displays on the panel to indicate success, failure, or that the action has been initiated.

The Back and Forward buttons automatically scroll, repeating the action as long as the button is held down. After holding the button down a few seconds, auto scrolling begins and rapidly increments or decrements the value.

Initial Setup of the Service Processor

This procedure describes the steps for the initial setup of the SP.

Part I: Assign an IP Address to the SP

1. **Connect one of the two SP Ethernet ports to a LAN.**

   It does not matter which port you use. The unused port can be used to daisy-chain a number of Sun Fire V20z servers to a management console.

2. **If a DHCP server services the LAN, turn on the main server power switch (on the rear panel) and observe the IP address of the SP on the LCD.**

3. **If no DHCP server is available, you can assign an IP address through the LCD console.**

4. **If no DHCP server or physical access is available, you can configure the SP from the local operating system (OS) using IPMItool in conjunction with an IPMI kernel driver.**

   You can also enter an IP address directly through the operator panel (see the *Sun Fire V20z Server Installation Guide*, 817-5246-xx).

Note – To configure the SP from the local OS, proceed first to "Part IV-a: Enabling IPMI LAN Access on a Linux-based Server (in-band)" on page 11 or to "Part IV-b: Alternate Method for Enabling IPMI LAN Access (out-of-band)" on page 12 for the network setup.

Following this, return to Part II instructions.

Part II: Secure the Service Processor

1. **Using an SSHv1 client or SSHv2 client, connect to the IP address of the SP.**
2. **Log in to the SP and authenticate as the user *setup* with no password required.**

```
ssh <spipaddr> -l setup
```

Note – If you are prompted for a password, this indicates that the SP has already been secured.

If you do not know the management user name and password, you can reset the SP from the operator panel. For more information about the menu options on the operator panel, refer to the *Sun Fire V20z Server User Guide*, 817-5248-xx.

---

3. **Follow the prompts to create a management user account. Remember the user name and password assigned to this account.**

Note – The IP address, user name and password that you configure in "Part I: Assign an IP Address to the SP" on page 8 and "Part II: Secure the Service Processor" on page 9 are referred to as the <`spipaddr`>, <`spuser`> and <`sppasswd`>.

Part III: Enabling IPMI Access on a Linux-based Server (in-band)

1. **Log in to the server and authenticate as the user *root*.**
2. **Install the custom openIPMI Linux kernel driver from the *Documentation and Support Files* CD-ROM located in the directory `/support/sysmgmt/`.**

   Browse to the OS variant installed on your Sun Fire V20z server. The options are:

   - redhat/rhel3 for Red Hat Enterprise Linux, version 3 (32-bit mode uses the architecture type "i386"; 64-bit mode uses architecture type "x86_64")
   - suse/sles8 for SuSE Enterprise Linux, version 8 (32-bit mode uses the architecture type "i386"; 64-bit mode uses architecture type "x86_64")
   - suse/suse9 for SuSE 9 Professional

3. **Ensure that the kernel-source RPM is already installed on your distribution by running the command:**

   ```
   rpm -qvi kernel-source
   ```

   If this utility reports that the kernel-source software package is not installed, install the kernel-source RPM that is current for your installed Linux distribution.

   a. **On SuSE distributions, install the kernel-source RPM by running the command:**

      ```
      yast2
      ```

   b. **On RedHat distributions, download the current kernel-source RPM to a temporary directory (such as `/tmp`). Install the package by running the command:**

      ```
      rpm -ivh /tmp/kernel-source*.rpm
      ```

4. **Install the openIPMI Linux kernel driver RPM.**

   a. **Browse to the OS variant installed on your Sun Fire V20z server. The options are:**

      - redhat/rhel3 for Red Hat Enterprise Linux, version 3 (32-bit mode uses the architecture type "i386"; 64-bit mode uses architecture type "x86_64")
      - suse/sles8 for Suse Enterprise Linux, version 8 (32-bit mode uses the architecture type "i386"; 64-bit mode uses architecture type "x86_64")
      - suse/suse9 for Suse 9 Professional

   b. **Install the openIPMI RPM file by running the command:**

      ```
      rpm -ivh openipmi*.rpm
      ```

Note – The kernel driver will be compiled using the kernel-source code during installation.

---

5. **Install IPMItool.**

   IPMItool is the command-line-interface (CLI) server-management client.

   a. **If the installed Linux distribution uses the 32-bit "i386" architecture, run the following command:**

      ```
      rpm -ivh ipmitool*.i386.rpm
      ```

   b. **If the installed Linux distribution uses the 64-bit "x86_64" architecture, run the following command:**

      ```
      rpm -ivh ipmitool*.x86_64.rpm
      ```

6. **Test the IPMI kernel device driver and client application by running the following command:**

   ```
   ipmitool -I open chassis status
   ```

   Successful output should look something like the following:

   ```
   "

   System Power: on

   Power Overload: false

   Power Interlock: inactive

   Main Power Fault: false

   Power Control Fault: false

   Power Restore Policy: unknown

   Last Power Event:

   Chassis Intrusion: inactive

   Front-Panel Lockout: inactive

   Drive Fault: false

   Cooling/Fan Fault: false

   "
   ```

Note – On a subsequent reboot, the IPMI kernel driver may have to be loaded with the following command:
`modprobe ipmi_kcs_drv`

Note – If you upgrade your Linux kernel, refer to "Upgrading the Kernel" on page 12.

Part IV-a: Enabling IPMI LAN Access on a Linux-based Server (in-band)

1. **If the server is powered off, boot the local OS.**
2. **Log in to the server and authenticate as the user *root*.**
3. **Load the OpenIPMI kernel device driver (as installed in Part III, Step 3).**

   ```
   modprobe ipmi_kcs_drv
   ```

4. **Using IPMItool, configure the network setting for the SP.**

Note – For more information on the syntax for IPMItool commands, refer to "Syntax" on page 22.

```
ipmitool -I open lan set 6 ipaddr <ipaddr>
ipmitool -I open lan set 6 netmask <netmask>
ipmitool -I open lan set 6 defgw ipaddr <gwipaddr>
ipmitool -I open lan set 6 password <ipmipasswd>
```

Part IV-b: Alternate Method for Enabling IPMI LAN Access (out-of-band)

1. **Using an SSHv1 client or SSHv2 client, log in to the IP address of the SP.**
2. **Authenticate as the newly created management user (see "Part II: Secure the Service Processor" on page 9).**

   ```
   ssh <spipaddr> -l <spuser>
   ```

3. **Enable IPMI LAN access and assign a password when prompted.**

   ```
   ipmi enable channel lan
   exit
   ```

Note – This password will be referred to as <`ipmipasswd`>.

---

4. **Using IPMItool, test the IPMI LAN access.**

   ```
   ipmitool -I lan -H <spipaddr> -P <ipmipasswd> chassis status
   ```

Upgrading the Kernel

Upgrading the installed Linux kernel to a newer version requires you to recompile the upgraded IPMI kernel device driver.

1. **Install the kernel-source RPM that matches the version of the upgraded kernel binary RPM package**
2. **Log in to the server and authenticate as the user *root*.**
3. **Change to the following directory:**

   ```
   cd /usr/src/kernel-modules/openipmi
   ```

4. **Recompile the module by running the following commands:**

```
make clean
make
make install
```

5. **Re-test the IPMI kernel device driver and client application by running the following command:**

```
ipmitool -I open chassis status
```

Successful output should look similar to the following:

```
"
System Power: on
Power Overload: false
Power Interlock: inactive
Main Power Fault: false
Power Control Fault: false
Power Restore Policy: unknown
Last Power Event:
Chassis Intrusion: inactive
Front-Panel Lockout: inactive
Drive Fault: false
Cooling/Fan Fault: false
"
```

Note – On a subsequent reboot, the IPMI kernel driver may have to be loaded with the following command:
`modprobe ipmi_kcs_drv`

Daisy-chaining the Sun Fire V20z servers

The SP uses one 10/100 network interface but integrates a 10/100 Ethernet switch. Both external management Ethernet ports connect to the internal switch.

The unused management Ethernet port can be used to daisy-chain to one of the platform Gigabit Ethernet ports or, through a crossover Ethernet cable, can be connected to another Sun Fire V20z server.

Site Integration

When deploying your Sun Fire V20z server, ensure that you determine the best integration strategy for your environment.

The Sun Fire V20z server includes network connections for the Service Processor (SP) that are separate from network connections for the platform. This allows you to configure the server so that the SP is connected to an isolated, management network, and is not accessible from the production network.

Updating the SP Software

Note – For complete information on the menu options available through the operator panel, refer to the *Sun Fire V20z Server User Guide*, 817-5250-xx.

If you attempt to update the SP software using the operator panel when the IP address for the SP has not been set, the update fails. Ensure that the IP address has been set prior to attempting an update.

For more information, refer to the *Sun Fire V20z Server Installation Guide*, 817-5246-xx.

Refer to "Operator Panel" on page 6 for general orientation and usage of the operator panel.

Note – Prior to executing this procedure, you must start the Java™ Update Server. Refer to "Updating the Service Processor Base Component" on page 16 for details about starting the Java Update Server.

To update the SP software:

1. **When the LCD displays the Service Processor information (as shown in the following example), press any button.**

```
123.45.67.89
OS running
```

The LCD displays the first menu option:

```
Menu:
Server Menu
```

2. **Press the Forward button until you reach the SP menu.**

   ```
   Menu:
   SP menu
   ```

3. **Press Select or Enter to display the SP menu's options.**

   ```
   SP Menu:
   Set SP IP info?
   ```

4. **Press the Forward button until you reach the Update SP Flash menu option.**

   ```
   SP Menu:
   Update SP Flash?
   ```

5. **Press Select or Enter.**

6. **A string of 0s displays with the cursor at the left digit. Use the Forward and Back buttons to increment or decrement a digit.**

Note – You are prompted for an IP address. If you attempt to update the SP software using the operator panel when the IP address for the SP has not been set, the update fails.

Note – If you need to supply a port address, it can be any number between 0 and 65535. The leading 0s are removed.

See Step 3 in "Updating the Service Processor Base Component" on page 16 for more information.

---

7. **Press Select to move to the next digit.**

8. **Press Select when finished to return to the left-most column.**

9. **Press the button combination for Enter.**

Updating the Service Processor Base Component

To update the SP base component:

1. **Start the spupdate server on a machine with a Java Runtime Environment (JRE) by running the following command:**

   ```
   java -jar spupdate.jar -f filename [ -p port ]
   ```

   The `spupdate.jar` file is located in the folder `spupdate` of the Network Share Volume.

   In this command, *filename* is an SP `.image` file located in `sw_images/sp/spbase/<version>`. The `sw_images` directory contains an SP base `.image` file for each version available.

   By default, the server uses port number 52708. If this port number is already in use, specify another port using the optional `-p` flag.

   The update server does not start if the file is not found in the specified path. Otherwise, the server is ready to receive update requests from any SP. The update server can simultaneously accept multiple update requests from different SPs.

2. **Log in to the SP by running the following command:**

   ```
   ssh <spipaddr> -l <spuser>
   ```

3. **Execute the sp command to start the update process on the SP.**

   ```
   sp update flash all {-i | --ipaddress} IPADDRESS [{-p | --port} PORT]
   ```

   Note – This command includes the optional `-p` flag to denote that the server is running on a port other than the default port. This command pings the update server to see if the update server is up and running. If successful, your connection is closed when the SP reboots and the update process begins.

   Refer to Appendix A for more information about the sp commands.

4. **Monitor progress of the update process on the server.**

   Messages display as the installation process progresses. When complete, the SP reboots with the new version installed.

Autoconfiguring the SP

Autoconfiguration replicates the majority of configuration files from an SP that has already been configured to another SP, so that the two servers have identical configurations except for the host name and IP address.

For example, after you configure a single SP (set up users, hosts, certificates, mounts and so on), you then run autoconfiguration on each additional SP so that the settings are identical. In addition, if you modify the configuration of one SP, you can update all of them by re-running autoconfiguration on each one. (For this reason, set the IP address of the autoconfigure server to x.x.x.1.)

Note – Autoconfiguration does not merge configurations, it overwrites the existing configuration.

To perform autoconfiguration of an SP, follow these steps:

You can start autoconfiguration either when you are prompted at the completion of setting the IP address of the SP, or by selecting Autoconfigure from the SP menu option on the operator panel at any time.

1. **On the operator panel, press the Forward or Back buttons until the following prompt shows `Yes`.**

   ```
   SP Auto Setup?
   No
   ```

   For instructions on setting an IP address, refer to the section "Defining SP Network Settings" in the *Sun Fire V20z Server Installation Guide*, 817-5246-xx.

2. **Press the Select button.**

   The SP attempts to locate an IP address.

   - If the SP successfully locates an IP address, the following prompt appears, displaying an IP address for this SP:

     ```
     Setup Server IP:
     x.x.x.1
     ```

     Where *x.x.x* is the first three octets of the SP IP address. For example, if the address is 10.10.30.19, the address that displays in the prompt appears as 10.10.30.1.

     In this case, press the Select operator panel button to start autoconfiguration.

   - If the SP does not locate an IP address, the following message appears:

     ```
     Unable to get
     SP IP address
     ```

     In this case, you must manually enter an IP address before you press the Select operator panel button to start autoconfiguration.

3. **Wait until the autoconfiguration is complete, at which point the SP automatically reboots.**

   Th following message displays when autoconfiguration is running.

```
SP AutoConfigure
in progress
```

Note – If the autoconfiguration is unsuccessful, a failure message displays. Press any button to clear it.

CHAPTER 2

IPMI Server Management

Server manufacturers today have to re-invent how each new server manages itself. The hardware and software design for one server does not necessarily work with another. Every server supplier provides basic monitoring and data collection functions but no two do it exactly the same. These proprietary implementations for manageability only complicate the problem.

The standardization of server-based management, called Intelligent Platform Management Interface (IPMI), provides a solution. IPMI allows you to interconnect the CPU and devices being managed. It allows for:

- Easy replication of the monitoring functions from server to server
- Support for a reasonably large number of monitoring devices
- Common driver-level access to management instrumentation
- More cost-effective implementations
- Increased scalability of the server management functions

IPMI is an industry-standard, hardware-manageability interface specification that provides an architecture defining how unique devices can all communicate with the CPU in a standard way. It facilitates platform-side server management and remote server-management frameworks, by providing a standard set of interfaces for monitoring and managing servers.

With IPMI, the software becomes less dependent on hardware because the management intelligence resides in the IPMI firmware layer, thereby creating a more intelligently managed server. The IPMI solution increases server scalability by distributing the management intelligence closer to the devices that are being managed.

Base Management Controller

In order to perform autonomous platform-management functions, the processor runs embedded software or firmware. Together, the processor and its controlling firmware are referred to as the Base Management Controller (BMC), which is the core of the IPMI structure. Tightly integrating an IPMI BMC and management software with platform firmware facilitates a total management solution.

The BMC is a service processor integrated into the motherboard design, providing a management solution independent of the main processor. The monitored server can communicate with the BMC through one of three defined interfaces which are based on a set of registers shared between the platform and the BMC.

Note – In the Sun Fire V20z server, the Service Processor (SP) has software that emulates a BMC.

The BMC is responsible for:

- Managing the interface between server management software and platform management hardware
- Interfacing to the system sensors, such as fan speed and voltage monitors
- Providing access to the system event log
- Providing autonomous monitoring, event logging and recovery control
- Acting as a gateway between the management software and the IPMB/ICMB
- Monitoring the system watchdog timer
- Facilitating the remote-management tasks, even when the main server hardware is in an inoperable state

The BMC provides the intelligence behind IPMI. In the Sun Fire V20z server, the SP serves as the BMC, providing access to sensor data and events through the standard IPMI interfaces.

Manageability

IPMI defines a mechanism for server monitoring and recovery implemented directly in hardware and firmware. IPMI functions are available independent of the main processors, BIOS and operating system.

IPMI monitoring, logging and access functions add a built-in level of manageability to the platform hardware. IPMI can be used in conjunction with server-management software running under the operating system, which provides an enhanced level of manageability.

IPMI provides the foundation for smarter management of servers by providing a methodology for maintaining and improving the reliability, availability and serviceability of expensive server hardware.

IPMI Compliance and Channel Access on the Sun Fire V20z server

The Sun Fire V20z server supports IPMI through the SP software version 2.0 and later. The Sun Fire V20z server meets compliance standards for IPMI version 1.5.

The IPMI implementation on the Sun Fire V20z server also supports LAN channel access. (Refer to the IPMI specification version 1.5 for details.) The LAN channel access is disabled by default. To enable it, use the `ipmi enable channel` command and specify the ID of the channel to enable for the LAN Interface.

Note – This ID is case-sensitive and must be lowercase.

```
ssh <spipaddr> -l <spuser> ipmi enable channel {sms | lan}
```

For more information about enabling or disabling the IPMI channel, refer to Appendix A, “Commands”.

Usernames and Passwords

Operator and administrator-level access over the LAN channel requires a valid user ID and password. The Sun Fire V20z server comes pre-configured with an administrator-level user with a null user ID and password. However, you can re-add the anonymous user at a later time if you wish. You can configure both the user ID and password to be null.

Note – For security reasons, the LAN channel access is disabled by default.

Note – IPMI user identities are in no way associated with user accounts defined for server-management capabilities. Refer to “Initial Setup of the Service Processor” on page 8 for more information about these server-management user accounts.

Lights Out Management (LOM)

On the Sun Fire V20z server, Lights Out Management is performed through IPMItool, a utility for controlling IPMI-enabled devices.

Description

IPMItool is a simple command-line interface (CLI) to servers that support the Intelligent Platform Management Interface (IPMI) v1.5 specification. It provides the ability to:

- Read the Sensor Data Record (SDR) and print sensor values
- Display the contents of the System Event Log (SEL)
- Print information about Field Replaceable Units (FRUs)
- Read and set LAN configuration parameters
- Perform chassis power control

Originally written to take advantage of IPMI-over-LAN interfaces, IPMItool is also capable of using a system interface as provided by a kernel device driver such as OpenIPMI.

Further Information

For up-to-date information about IPMItool, visit:

`http://ipmitool.sourceforge.net/`

For more information about the IPMI specification, visit:

`http://www.intel.com/design/servers/ipmi/spec.htm`

For more information about the OpenIPMI project (MontaVista IPMI kernel driver), visit:

`http://openipmi.sourceforge.net/`

Syntax

The syntax used by IPMItool is as follows:

`ipmitool [`**`-ghcvV`**`] `**`-I`**` `*`lan`*` `**`-H`**` `*`address`*` [`**`-P`**` `*`password`*`] <`*`expression`*`>`

`ipmitool [`**`-ghcvV`**`] -I `*`open`*` <`*`expression`*`>`

Options

TABLE 2-1 lists the options available for IPMItool.

TABLE 2-1 Options for IPMItool

Option	Description
-h	Provides help on basic usage from the command line.
-c	Makes the output suitable for parsing, where possible, by separating fields with commas instead of spaces.
-g	Attempts to make IPMI-over-LAN communications more robust.
-V	Displays the version information.
-v	Increases the amount of text output. This option may be specified more than once to increase the level of debug output. If given three times, you receive hexdumps of all incoming and outgoing packets.
-I <*interface*>	Selects the IPMI interface to use. The possible interfaces are LAN or open interface.
-H <*address*>	Displays the address of the remote server; it can be an IP address or host name. This option is required for the LAN interface connection.
-P <*password*>	Displays the password for the remote server; the password is limited to a maximum of 16 characters. The password is optional for the LAN interface; if a password is not provided, the session is not authenticated.

Expressions

TABLE 2-2 lists the expressions and parameters available for IPMItool.

Note – For each of these expressions, the beginning command is always `ipmitool`, followed by the expression and parameter(s).

Note – The `sol` command is not supported in the Sun Fire V20z server, but you can enable a Serial-over-LAN feature. See "Serial Over LAN" on page 54.

TABLE 2-2 Expressions and Parameters for IPMItool *(1 of 4)*

Expression	Parameter	Sub-parameter	Description and examples

help			Can be used to get command-line help on IPMItool commands. It may also be placed at the end of commands to get help on the use of options.
			
			EXAMPLES:
			`ipmitool -I open help`
			`Commands:  chassis, fru, lan, sdr, sel`
			
			`ipmitool -I open chassis help`
			`Chassis Commands:  status, power, identify, policy, restart_cause`
			
			`ipmitool -I open chassis power help`
			`Chassis Power Commands: status, on, off, cycle, reset, diag, soft`

raw	*<netfn>*	*<cmd>* [*data*]	Allows you to execute raw IPMI commands (for example, to query the POH counter with a raw command).
			
			EXAMPLE:
			`ipmitool -I open raw 0x0 0xf`
			
			`RAW REQ (netfn=0x0 cmd=0xf data_len=0)`
			`RAW RSP (5 bytes)`
			`3c 72 0c 00 00`

TABLE 2-2 Expressions and Parameters for IPMItool *(2 of 4)*

Expression	Parameter	Sub-parameter	Description and examples

chaninfo	[*channel*]		Displays information about the selected channel. If no channel is specified, the command displays information about the channel currently being used.
			
			EXAMPLES:
			`ipmitool -I open chaninfo`
			`Channel 0xf info:`
			`  Channel Medium Type: System Interface`
			`  Channel Protocol Type: KCS`
			`  Session Support: session-less`
			`  Active Session Count: 0`
			`  Protocol Vendor ID: 7154`
			
			`ipmitool -I open chaninfo 7`
			`Channel 0x7 info:`
			`  Channel Medium Type: 802.3 LAN`
			`  Channel Protocol Type: IPMB-1.0`
			`  Session Support: multi-session`
			`  Active Session Count: 1`
			`  Protocol Vendor ID: 7154`
			`  Alerting: enabled`
			`  Per-message Auth: enabled`
			`  User Level Auth: enabled`
			`  Access Mode: always available`

userinfo	<*channel*>		Displays information about configured user information on a specific LAN channel.
			
			NOTE: This command fails on system interfaces. You can try channel 6 or 7.
			
			EXAMPLE:
			`ipmitool -I open userinfo 6`
			`Maximum User IDs      : 4`
			`Enabled User IDs      : 1`
			`Fixed Name User IDs   : 1`
			`Access Available      : call-in / callback`
			`Link Authentication   : disabled`
			`IPMI Messaging        : enabled`

chassis	status		Returns information about the high-level status of the server chassis and main power subsystem.

TABLE 2-2 Expressions and Parameters for IPMItool *(3 of 4)*

Expression	Parameter	Sub-parameter	Description and examples
	identify	*<interval>*	Controls the front panel *identification* light. The default value is 15 seconds. Enter "0" to turn it off.
	Note: At the time of the Sun Fire V20z server's release, IPMItool did not support the `chassis identify` command. You can use the following commands: To enable the locate light: `ssh -l <spuser> <spidpaddr> sp set locatelight blink` To disable the locate light: `ssh -l <spuser> <spidpaddr> sp set locatelight off` For more information, see "Commands" on page 57.		
	restart_cause		Queries the chassis for the cause of the last server restart.
power			Performs a chassis control command to view and change the power state.
	status		Shows the current status of the chassis power.
	on		Powers on the chassis.
	off		Powers off chassis into the *soft off* state (S4/S5 state). NOTE: This command does not initiate a clean shutdown of the operating system prior to powering off the server.
	cycle		Provides a power-off interval of at least 1 second. No action should occur if chassis power is in S4/S5 state, but it is recommended to check the power state first and then only issue a power-cycle command if the server power is on or in a lower sleep state than S4/S5.
	reset		Performs a hard reset.
lan	print	*<channel>*	Prints the current configuration for the given channel.
	set	*<channel> <parameter>*	Sets the given parameter on the given channel.
		ipaddr *<x.x.x.x>*	Sets the IP address for this channel.
		netmask *<x.x.x.x>*	Sets the netmask for this channel.
		macaddr *<xx:xx:xx:xx:xx:xx>*	Sets the MAC adddress for this channel.
		defgw ipaddr *<x.x.x.x>*	Sets the default gateway IP address.
		defgw macaddr *<xx:xx:xx:xx:xx:xx>*	Sets the default gateway MAC address.
		bakgw ipaddr *<x.x.x.x>*	Sets the backup gateway IP address.
		bakgw macaddr *<xx:xx:xx:xx:xx:xx>*	Sets the backup gateway MAC address.

TABLE 2-2 Expressions and Parameters for IPMItool *(4 of 4)*

Expression	Parameter	Sub-parameter	Description and examples
		password <*pass*>	Sets the null user password.
		user	Enables the user-access mode.
		access <on | off>	Sets the LAN-channel-access mode.
		ipsrc <*source*>	Sets the IP address source. As a source, you can indicate: **none** = unspecified **static** = manually configured static IP address **dhcp** = address obtained by BMC running DHCP **bios** = address loaded by BIOS or system software
		arp respond <on | off>	Sets the BMC-generated ARP responses.
		arp generate <on | off>	Sets the BMC-generated gratuitous ARPs.
		arp interval <seconds> s	Sets the interval for the BMC-generated gratuitous ARPs.
		auth <*level,...*> <*type,...*>	This command sets the valid authtypes for a given auth level. Levels can be: `callback`, `user`, `operator`, `admin` Types can be: `none`, `md2, md5`
fru	print		Reads all inventory data for the Field Replaceable Units (FRUs) and extracts such information as serial number, part number, asset tags and short strings describing the chassis, board or product.
sdr	list		Reads the Sensor Data Record (SDR) and extracts sensor information, then queries each sensor and prints its name, reading and status.
sel	info		Queries the BMC for information about the system event log (SEL) and its contents.
	clear		Clears the contents of the SEL. The `clear` command cannot be undone.
	list		Lists the contents of the SEL.

IPMI Linux Kernel Device Driver

The IPMItool application utilizes a modified MontaVista OpenIPMI kernel device driver found on the *Documentation and Support Files* CD-ROM. The driver has been modified to use an alternate base hardware address and modified device IO registration.

This driver must be compiled and installed from the *Documentation and Support Files* CD-ROM.

The following kernel modules must be loaded in order for IPMItool to work:

1. `ipmi_msghandler`

   The message handler for incoming and outgoing messages for the IPMI interfaces.

2. `ipmi_kcs_drv`

   An IPMI Keyboard Controller Style (KCS) interface driver for the message handler.

3. `ipmi_devintf`

   Linux-character-device interface for the message handler.

To force IPMItool to use the device interface, you can specify it on the command line:

```
ipmitool -I open [option...]
```

Installing and Compiling the Driver

To install and compile this kernel device driver, see "Initial Setup of the Service Processor" on page 8.

LAN Interface for the BMC

Note – In the Sun Fire V20z server, the Service Processor (SP) has software that emulates a BMC.

The IPMItool LAN interface communicates with the BMC over an Ethernet LAN connection using User Datagram Protocol (UDP) under IPv4. UDP datagrams are formatted to contain IPMI request/response messages with IPMI session headers and Remote Management Control Protocol (RMCP) headers.

Remote Management Control Protocol is a request-response protocol delivered using UDP datagrams to port 623. IPMI-over-LAN uses version 1 of the RMCP to support management both before installing the OS on the server or if the server will not have an OS installed.

The LAN interface is an authenticated, multi-session connection; messages delivered to the BMC can (and should) be authenticated with a challenge/response protocol with either a straight password/key or an MD5 message-digest algorithm. IPMItool attempts to connect with administrator privilege level as this is required to perform chassis power functions.

With the `-I` option, you can direct IPMItool to use the LAN interface:

```
ipmitool -I lan [option...] <address> [password]
```

To use the LAN interface with IPMItool, you must provide a host name on the command line.

The password field is optional; if you do not provide a password on the command line, IPMItool attempts to connect without authentication. If you specify a password, it uses MD5 authentication if supported by the BMC; otherwise, it will use straight password/key.

Files

The file `/dev/ipmi0` is a character-device file used by the OpenIPMI kernel driver.

Examples

If you want to control remotely the power of an IPMI-over-LAN-enabled server, you can use the following commands:

```
ipmitool -I lan -H <spipaddr> -P <ipmipasswd> chassis power on
```

The result returned is:

```
Chassis Power Control: Up/On
```

```
ipmitool -I lan -H <spipaddr> -P <ipmipasswd> chassis power status
```

The result returned is:

```
Chassis Power is on
```

Viewing the System Event Log

To view the System Event Log (SEL), use IPMItool.

The out-of-band command is:

```
ipmitool -I lan -H <spipaddr> -P <ipmipasswd> sel list
```

The in-band command (using OpenIPMI on a Linux-based server or LIPMI on a Solaris™-based server) is:

```
ipmitool -I open sel list
```

Note – To receive more verbose logging messages, you can run the following command: `ssh -l <spuser> <spipaddr> sp get events`

IPMI Troubleshooting

TABLE 2-3 describes some potential issues with IPMI and provides solutions.

TABLE 2-3 IPMI Troubleshooting

Issue	Solution
You cannot connect to the management controller using IPMItool over LAN.	Verify the network connection to the management controller and its IP address.
You cannot authenticate to the management controller using IPMItool over LAN.	Ensure that you are using the password assigned when you enabled IPMI LAN access from the management-controller shell prompt.

You have forgotten the password for IPMI access over LAN.	1. You can reset the IPMI setting, reset the SDR and purge the SEL from the management-controller shell by running the command:
	`ssh <spipaddr> -l <spuser> ipmi reset -a`
	2. Now re-enable IPMI on LAN with the following commands:
	`ssh <spipaddr> -l <spuser>`
	`ipmi enable channel lan`
	`exit`

IPMItool fails when using the "open" interface.	Ensure that the Linux kernel module `ipmi_kcs_drv` is loaded by running the command `lsmod`.

CHAPTER 3

SNMP Server Management

You can manage your Sun Fire™ V20z server using the Simple Network Management Protocol (SNMP). This chapter also explains how to update your Service Processor (SP) software using the operator panel.

Simple Network Management Protocol

Simple Network Management Protocol (SNMP) is a network-management protocol used almost exclusively in TCP/IP networks. SNMP provides a means to monitor and control network devices, and to manage configurations, statistics collection, performance and security on a network.

SNMP-based management allows for third-party solutions to be used. This includes products such as HP OpenView and CA Unicenter.

The base component of an SNMP solution is the Management Information Base (MIB). The MIB is included on the Sun Fire V20z server *Network Share Volume* (NSV) CD-ROM.

This configuration is beneficial when, for example, you have a cluster of machines serving Web content and the platform is connected to the Internet, but the SP is protected and only accessible on an internal network.

SNMP Integration

Simple Network Management Protocol (SNMP) is an open network-management technology that enables the management of networks and entities connected to the network. Within the SNMP architecture is a collection of network-management stations and managed nodes. Network-management stations execute management applications which monitor and control managed nodes. Managed nodes are devices such as hosts, gateways and so on, which have management agents responsible for performing the management functions requested by the management stations. SNMP is used to communicate management information between the management stations and the agents. In other words, SNMP is the protocol by which the agent and the management station communicate.

The monitoring of state through SNMP at any significant level of detail is accomplished primarily by polling for appropriate information on the part of the management station. Managed nodes may also provide unsolicited status information to management stations in the form of traps, which is likely to guide the polling at the management station.

Communication of information between management entities in a network is accomplished through the exchange of SNMP-protocol messages, both in the form of queries (get/set) by the management station and in the form of unsolicited messages (traps) indicated by the agent.

Your Sun Fire V20z server includes SNMP agents that allow for health and status monitoring. The SNMP agent runs on the SP and therefore all SNMP-based management of the server should occur through the SP. The SNMP agent on the Sun Fire V20z server provides the following capabilities:

- Event management
- Inventory management
- Sensor and system state monitoring
- SP configuration monitoring

SNMP Management Information Base (MIB)

The Management Information Base (MIB) is a text file that describes SNMP data as managed objects. The Sun Fire V20z server provides SNMP MIBs so that you can use any SNMP-capable network management system, such as HP OpenView Network Node Manager (NNM), Tivoli, CA Unicenter, IBM Director and so on, to manage and monitor your Sun Fire V20z server. The MIB data describes the information being managed, reflects current and recent server status, and provides server statistics.

Sun Fire V20z Server MIB Tree

FIGURE 3-1 illustrates the Sun Fire V20z server MIB tree:

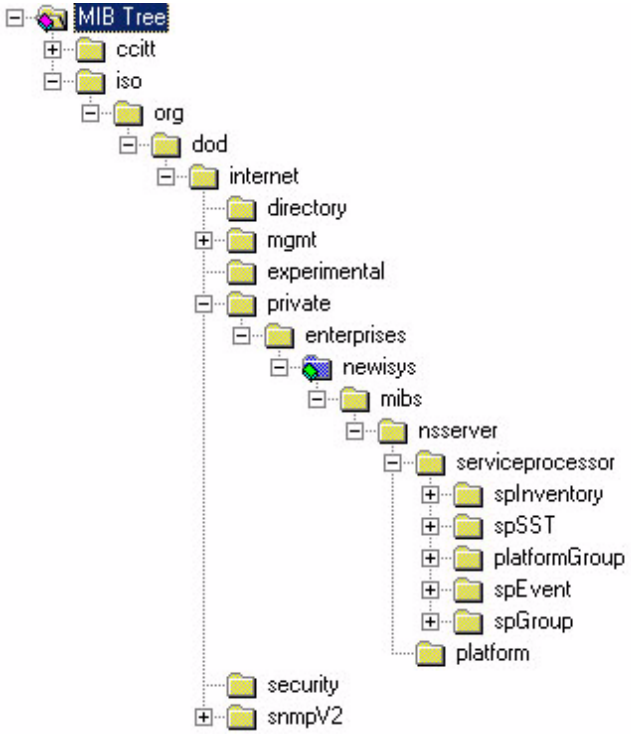

FIGURE 3-1 Sun Fire V20z server MIB Tree

Integrating MIBs with Third-Party Consoles

You use the Sun Fire V20z server MIBs to integrate the management and monitoring of Sun Fire V20z server into SNMP management consoles. The Sun Fire V20z server MIB branch is a private enterprise MIB, located at object identifier (OID) 1.3.6.1.2.1.9237. The standard SNMP port 161 is used by the SNMP agent on the SP.

Configuring SNMP on Your Sun Fire V20z Server

Note – There are several services that are supplied by the SNMP agent on the Sun Fire V20z server. Depending on your business needs and configuration of your current office network and management environment, you may wish to take advantage of these services.

There are certain prerequisites and setup required on both the SP and the platform in order to enable and utilize each of these services:

- SNMP agent on the SP
- Proxy forwarder application/ProxyAgent [RFC 2271]
- Agent X [RFC 2741]

The following diagram illustrates the SNMP architecture and communication paths between the SP and the platform.

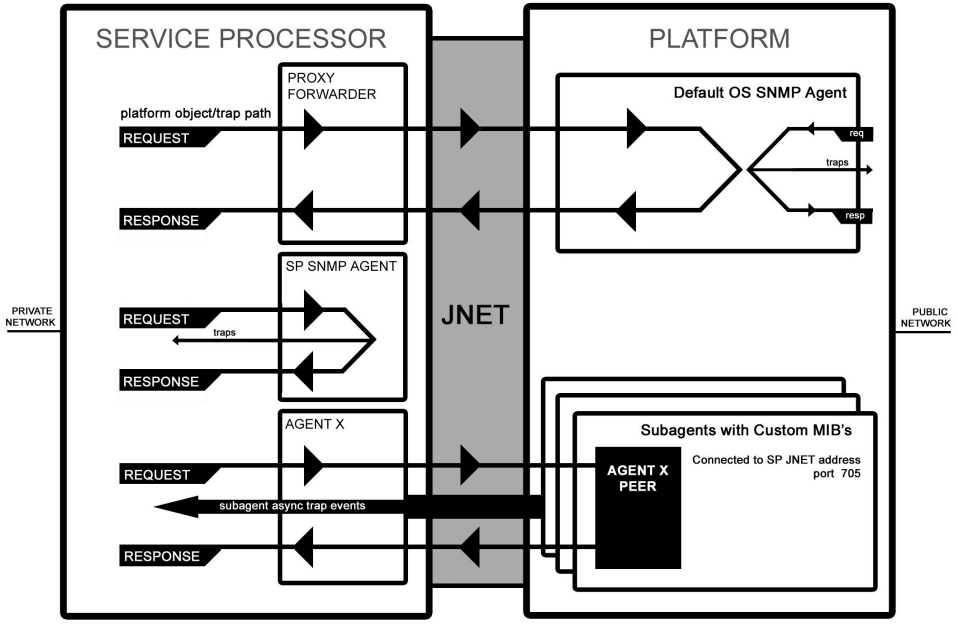

FIGURE 3-2 SNMP Architecture and Communications

SNMP Agent on the Service Processor

The SNMP agent running on the SP facilitates the management and monitoring of the server. The SNMP agent can be used to query various types of SP information. Refer to FIGURE 3-1 for a list of the MIBs; refer to TABLE 3-3 for a detailed description of the MIBs.

There is no configuration required to use this functionality other than integrating the Sun Fire V20z server MIBs with your desired management station.

Refer to the procedure for using the SNMP agent on the SP, as explained in "Integrating MIBs with Third-Party Consoles" on page 35.

Note – The SNMP agent on the Sun Fire V20z server supports SNMP v1/v2c. For security reasons, there are no settable attributes in this agent.

Proxy Agent

The SP acts as an SNMP proxy agent intermediary for the platform. Queries made from a management station to the SNMP agent on the SP are intercepted by the proxy agent on the SP and forwarded to the platform; the SP proxy agent contacts the platform to retrieve the requested information. The proxy agent then receives the data from the platform and sends the request back to the management station. The management station never knows that the request was proxied. The SP and the platform communicate over an internal private network.

To enable this facility, you must first run an SNMP agent on your platform operating system (see your operating system vendor to obtain this agent). This enables platform-level management transparently through the SP. Querying MIBs other than the Sun Fire V20z server MIB (for example, the Host Resource MIB) and the MIBII System MIB on the SP obtains information from the platform by proxying the request to the platform SNMP agent.

Ensure that the SP can identify the read-only and read-write community names that are configured for your platform SNMP agent. Refer to "Setting the Community Name" on page 38.

Setting the Community Name

The SNMP agent on the SP acts as a proxy for the SNMP agent running on the platform. (Refer to "Configuring SNMP on Your Sun Fire V20z Server" on page 36.) To properly proxy, you must use the community string. The community string needed to do so is the value defined when you configured the platform for SNMP.

If you find that your SNMP queries are not being proxied to the platform SNMP agents, validate that the community string on the SP matches that on the platform. The SP proxy community string can be changed to match the platform community string using the following command:

```
sp set snmp proxy community
```

There are no restrictions on the length of the community strings; common names are *private* and *public*. The default name is *public*.

For more information, refer to "SP Set SNMP Proxy Community Subcommand" on page 86.

Agent X

A sub-agent using SNMP Agent X protocol on the platform can connect to the SNMP agent on the SP (through a special port) and forward query responses or unsolicited traps through the SP. This allows server-management traffic to be kept secure from the production network connected to the platform, if required.

To properly enable this facility, you must identify the IP address and port number pair associated with the SP (as seen from the platform). The Agent X port is fixed at 705 (TCP). However, the private-network IP address is configurable and, by default, this address is 169.254.101.2.

Refer to your application documentation for instructions on configuring the sub-agents.

Note – You can use the subcommand `sp get jnet` on the SP to retrieve the JNET IP address of the SP.

Using a Third-Party MIB Browser

The following example demonstrates integrating the Sun Fire V20z server MIBs into an SNMP node manager.

1. **From the Manager Preferences menu, choose Load/Unload MIBS: SNMP.**
2. **Locate and select the SP-MasterAgent-MIB.mib.**
3. **Click Load.**
4. **Specify the directory in which the Sun Fire V20z server MIBs are placed and click Open.**
5. **Repeat steps 2 through 4 to load other MIBS (for example, SP-SST-MIB.mib, SP-INVENTORY-MIB.mib, SP-EVENT-MIB.mib, SP-PLATFORM-MIB.mib, SP-GROUP-MIB.mib and so on).**
6. **Exit the Manager Preferences menu.**
7. **Open an SNMP MIB browswer.**

   The SNMP standard tree displays in the MIB Browser.
8. **Locate the Newisys branch located under private.enterprises.**

   Refer to FIGURE 3-1 for a sample view of the MIB tree.

Setting Logging Options

You can also easily integrate SP-generated traps and set logging options. The following example demonstrates the necessary steps when using HP OpenView NNM:

1. **Load the SP-EVENT-MIB.mib according to the previous procedure.**
2. **Choose Options>EventConfiguration**
3. **Select the spEvent module from the Enterprises list.**
4. **Double-click an event from the Events for Enterprise spEvent list.**
5. **Select the Event Message tab.**
6. **Select the Log and display in category radio dialog and choose a category from the corresponding dropdown list, or create your own event category.**
7. **Select the severity of the event from the Severity dropdown list.**
8. **Enter a message or $* to display all information in the Event Log Message field.**
9. **Click OK.**

SNMP Traps

SNMP traps are network-management notifications of an event occurring at a managed network node. These events can identify problems in the network, machines up or down, and so on. The Sun Fire V20z server uses traps to signal conditions related to the server's health, including critical conditions related to physical components, the return to a normal state for these components, and other situations related to the state of the software running on the SP (for example, network settings being reconfigured).

Traps are defined in the MIB files and are generated, received and processed by an SNMP management station. SNMP trap data is uniquely identified by the MIB. Each SNMP trap contains information identifying the server's name, IP address and other relevant data about the event.

Within the Sun Fire V20z server event MIB, each trap has the following variables and event bindings; see TABLE 3-1.

TABLE 3-1 Sun Fire V20z server Event Traps

Event	Description
EventID	Uniquely identifies the event on the SP from where it came.
EventSource	Denotes the source module that generated the event.
EventComponent	Denotes the component ID about which the event refers.
EventDescription	The event message received from its source.
EventTimeStampInitial	The time at which this event ID was initially generated.
EventTimeStampLast	The most recent time at which this event ID was generated.

Configuring SNMP Trap Destinations

Although SNMP traps are generated for events that occur on the SP, you must configure where these traps are to be sent. There is no default destination for traps. You can use the server-management subcommands (see TABLE 3-2) on the SP to configure SNMP destinations.

For more information on these subcommands, refer to Appendix A.

TABLE 3-2 Subcommands for Configuring SNMP Destinations

Subcommand	Description
`sp get snmp-destinations`	Displays all the available SNMP destination IP addresses and host names to which the SP will send.
`sp add snmp-destination`	Adds a new SNMP destination one IP address or host name at a time.
`sp delete snmp-destination`	Removes an existing SNMP destination one IP address or host name at a time.

Configuring SNMP Destinations

Administration- and manager-level users can define SNMP destinations to which SNMP events (alerts) will be sent using this option. All users can view the current destinations (through read-only access).

The number of destinations you can create is limited due to memory constraints.

Sun Fire V20z Server MIB Details

SNMP uses object identifiers (OIDs) to provide name variables by which objects are grouped together for easier reference. The Sun Fire V20z server provides agents for the MIBs shown in TABLE 3-3:

TABLE 3-3 SNMP MIBs

MIB	OID	Description
SP-MasterAgent-MIB .mib	.1.3.6.1.4.1.9237	Creates the main trunk of the Sun Fire V20z server MIB tree. All other MIBs of the Service Processor branch from this tree. To be loaded first while integrating with any third-party framework.
SP-INVENTORY-MIB .mib	.1.3.6.1.4.1.9237.2.1.1.1 .1.3.6.1.4.1.9237.2.1.1.1.2 .1.3.6.1.4.1.9237.2.1.1.1.3	Used for querying inventory information for all Sun Fire V20z server hardware and software components. **Hardware Inventory Table**: Collects all hardware component inventory. **Software Inventory Table**: Collects all software component inventory.
SP-SST-MIB.mib	.1.3.6.1.4.1.9237.2.1.1.4	Defines objects for the System State Table in the SP. Contains all sensor readings, including the name of the sensor, its current value, maximum allowed value, measurement type, scale and scanning interval.
SP-PLATFORM-MIB.mib	.1.3.6.1.4.1.9237.2.1.1.5	Defines objects for the platform SNMP which includes osstate, platform state, and platform IP table.
SP-EVENT-MIB.mib	.1.3.6.1.4.1.9237.2.1.1.6	Identifies the OIDs associated with all SNMP traps originated from the Service Processor.
SP-GROUP-MIB.mib	.1.3.6.1.4.1.9237.2.1.1.7	Defines objects for the SP including host name, DNS, a reboot node, a node to hold the last port 80 postcode, a clone tree and an IP table.

The events listed in TABLE 3-4 are sent to the SNMP destination by `SP-EVENT-MIB.mib`.

TABLE 3-4 SP Events *(1 of 2)*

Enterprise Trap ID	Event
1	spGenericEventInformational
2	spGenericEventWarning
3	spGenericEventCritical
4	spTemperatureEventInformational
5	spTemperatureEventWarning
6	spTemperatureEventCritical
7	spVoltageEventInformational
8	spVoltageEventWarning
9	spVoltageEventCritical
10	spFanEventInformational
11	spFanEventWarning
12	spFanEventCritical
13	spPlatformMachineCheckEventInformational
14	spPlatformMachineCheckEventWarning
15	spPlatformMachineCheckEventCritical
16	spPlatformStateChangeEventInformational
17	spPlatformStateChangeEventWarning
18	spPlatformStateChangeEventCritical
19	spPlatformBIOSEventInformational
20	spPlatformBIOSEventWarning
21	spPlatformBIOSEventCritical
22	spGenericEventInformational
23	spGenericEventWarning
24	spGenericEventCritical
25	spTemperatureEventInformational
26	spTemperatureEventWarning
27	spTemperatureEventCritical
28	spVoltageEventInformational

TABLE 3-4 SP Events *(2 of 2)*

Enterprise Trap ID	Event
29	spVoltageEventWarning
30	spVoltageEventCritical
31	spFanEventInformational
32	spFanEventWarning
33	spFanEventCritical
37	spPlatformStateChangeEventInformational
38	spPlatformStateChangeEventWarning
39	spPlatformStateChangeEventCritical
40	spPlatformBIOSEventInformational
41	spPlatformBIOSEventWarning
42	spPlatformBIOSEventCritical

CHAPTER 4

Sun Control Station

A Sun-branded product, the Sun Control Station (SCS) is a server-management solution for the Sun Fire V20z server.

Note – At the time of the Sun Fire V20z server's release, the Sun Control Station software did not support the Sun Fire V20z server. Support will be available through future releases of SCS.

For more information on the Sun Control Station software, visit the Web site at: `http://wwws.sun.com/software/controlstation/index.html`

The Sun Control Station is a native-Linux application that allows administrators to take control of their servers: tracking and applying software updates, deploying new services, and monitoring the health and performance of servers. More than just a typical server-management device, the Sun Control Station is a platform that helps you to manage the complete life cycle of your servers, from initial setup through eventual redeployment at the end of useful life. As an added benefit, customers can continue to use third-party or homegrown software in conjunction with the Sun Control Station.

There is both a server-side component and a client-side component for the Sun Control Station. The server-side component can be installed on any x86-based server running Red Hat OS 7.3 or Red Hat Enterprise Linux AS 2.1; the client-side component, known as an *agent*, will be available for the Sun Fire V20z server, for both Linux and Solaris™ customers.

Sun Control Station supports both in-band and out-of-band server management.

The server-side component consists of two parts: a core framework that is the engine for executing control modules, and the built-in control modules themselves. These control modules can come from Sun Microsystems, Inc., from third-party vendors or from your own in-house design team. Numerous modules are available from Sun: Appliance Inventory, Performance Monitoring, Software Management, Health Monitoring, Lights Out Management (LOM) and AllStart, which includes the JumpStart™ utility for Solaris-based servers, the KickStart utility for servers running Red Hat operating systems and the AutoYaST utility for servers running SUSE operating systems.

Through the Sun Control Station, you can fully control the distribution of software payloads, offering customized and tailor-made services to downstream and end-user customers. Service Providers can offer unique payloads, data or software monitoring services. By leveraging the Sun BlueLinQ technology, all available software updates and patches can be accessed and distributed by a Sun Control Station, as designated by the administrator. Custom-built and third-party software can also be distributed.

Services on the Sun Control Station

Here is a sample of what you can do with the Sun Control Station:

- **Inventory Management.** You can import and group from hundreds up to thousands of servers, as well as obtain detailed information on the servers.
- **Software Management.** The Server Administrator can keep servers current using the version-tracking feature for software patches and updates. You can also deploy custom software or data.
- **Local Software Repository.** In conjunction with Software Management, you can use the local software repository to "publish" software package files and allow the Server Administrator or end users of various servers to install available package files as desired or needed. You can configure your Sun Control Station to view "published" package files and share package files with other BlueLinQ-enabled servers.
- **Health and Performance Monitoring.** These control modules provide system alerts and the metrics of the basic operations of the servers, allowing you to pinpoint potential causes of failure.
- **Lights Out Management (LOM)** allows you to perform certain management functions remotely on servers that are compliant with the Intelligent Platform Management Interface (IPMI) version 1.5. This control module allows you to:
  - Power on and power off a server
  - Perform a hardware reset
  - Illuminate a blue light-emitting diode (LED) on the server for identification and location
  - View the current sensor data and System Event Log (SEL) from the motherboard in the server

- **AllStart** provides a common user interface for creating software payloads, defining client profiles, and monitoring and validating system installations and updates. This module allows you to:
  - Select files or RPMs to load onto a client
  - Select the distributions of different OSs to load onto a client
  - Create customized payloads made up of files and OS distributions
  - Create profiles containing configuration information
  - Add clients on to which the payloads and profiles are loaded (using the Media Access Layer [MAC] address of the client)

These services can be used within an extranet or an intranet environment, or across the Internet.

CHAPTER 5

Further Management Information

Console Redirection Over Serial on a Linux-based Server

Caution – Redirecting the console over serial is a procedure intended for advanced users of Linux only.

You can seriously disrupt the proper functioning of Sun Fire™ V20z server or render the server unbootable if you introduce a problem in the configuration files.

Note – Instructions for console redirection on a Solaris™-based server are not yet available.

Redirecting the console interaction over the serial port allows the user another method to monitor the server. The goal of these configurations is to configure the bootloader to redirect its output, pass the kernel the proper parameters and configure a login session on the serial port.

This chapter specifies how to configure these options.

The BIOS redirects console output to serial by default (9600, 8N1, no handshake) until a bootloader program is run from the hard disk drive. The bootloader must be configured to support the serial console in addition to the keyboard, video and monitor (KVM) console.

Two common bootloaders are `grub` and Linux Loader (LILO).

Caution – Do not edit the working-image section of your configuration files directly.

Copy the working-image section and paste it within the configuration file. Make your editing changes to this copied section.

---

grub

If you use `grub`, there are three steps to enable console redirection over serial; these steps all involve editing the `grub` configuration file `/etc/grub.conf`.

Note – The file `/etc/grub.conf` is a symbolic link to the file `/boot/grub/grub.conf`.

---

1. Passing the proper console parameters to the kernel.
2. Configuring the grub menu system to redirect to the proper console.
3. Removing any splash images that would prevent the proper serial-console display.

For more information on the parameters, refer to the file `kernel-parameters.txt` in your kernel documentation.

For more information on `grub`, run the command `info grub`.

Note – If the arrow keys do not work through your remote serial concentrator, you can use the keystroke combinations of <CTRL+P> and <CTRL+N> work to highlight the Previous and Next entry, respectively. Pressing Enter then boots that entry.

The parameter `console=ttyS1` tells the system to send the data to the serial port first. The parameter `console=tty0` tells the system to send the data to the KVM second.

A working-image section in your file `/etc/grub.conf` should have an entry for the kernel image to boot. The stock kernel entry looks like:

```
kernel /vmlinuz-<kernelrevision> ro root=/dev/sda5
```

where <*kernelrevision*> is simply the kernel version that you are using.

You need to change the stock kernel entry of your image to include the console-kernel parameters, as follows:

```
kernel /vmlinuz-<kernelrevision> ro root=/dev/sda5
console=ttyS1,9600 console=tty0
```

Note – These options should be all on one line with no wrap to a second line.

Add the following two lines to the file `/etc/grub.conf`:

```
serial --unit=1 --speed=9600
terminal serial console
```

Adding these two lines sets up your serial port or your KVM as your `grub` console so that you can remotely or locally select a boot image from the `grub` menu.

Comment out or remove the following line from the file `/etc/grub.conf`:

```
splashimage=(hd0,1)/boot/grub/splash.xpm.gz
```

Removing the `splashimage` line allows for greater compatibility during your serial connection; with this line removed, the splash image does not prevent the proper `grub` menu from displaying.

LILO

LILO uses the `append` feature in an image section in order to pass to the kernel the proper parameters for using the serial console.

You can enter the consoles in the `append` statement of the file `/etc/lilo.conf`:

```
append="console=ttyS1,9600 console=tty0"
```

After modifying the file `/etc/lilo.conf`, the user must run `lilo` from the command line to activate the change.

For more information on LILO, run the commands `man lilo` or `man lilo.conf`.

getty

You can run a service called `getty` to log out of idle shell sessions automatically on the serial interface.

To enable `getty`, append the following line to the list of `gettys` in the file `/etc/inittab` file:

```
7:12345:respawn:/sbin/agetty 9600 ttyS1
```

Note – It does not matter where you append this line in the list.

The list of `gettys` currently looks like the following:

```
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

securetty

To add the serial-console device `/dev/ttyS1 to the file /etc/securetty`, run the following command:

```
echo ttyS1 >> /etc/securetty
```

Enabling and Configuring BIOS Console Redirection

Note – Console redirection is enabled by default in the BIOS.

If the default settings have been changed in the BIOS, the following procedure explains how to change the console-redirection settings.

1. **Boot or reboot the Sun Fire V20z server.**
2. **When prompted, press <F2> to enter BIOS setup.**
3. **Select the Advanced menu from the category selections along the top.**
4. **Select Console Redirection.**

Note – Make note of all settings in this menu, as they are required for configuring the remote-console access and the Serial Over LAN (SOL) feature.

- To disable console redirection to serial, select Disabled from the option Port.
- To enable console redirection, select On-board COM A from the option Port.
- To change the baud rate, select the desired bit rate from the option Baud Rate.

5. **Save the changes to the BIOS settings.**
6. **Press <F10> to exit the BIOS setup.**

For the new settings to take effect, you must reboot the server.

Network Share Volume (NSV) CD-ROM

A network share volume (NSV) structure is included with the Sun Fire V20z server on CD-ROM.

Although the SP functions normally without access to an external file system, a file system is required to enable several features, including event log files, software updates, diagnostics, the troubleshooting dump utility and online help. You can configure the NSV to be shared among multiple SPs. Admin- and manager-level users can configure the external file system; regular users can only view the current configuration with read-only access.

The following software components are included with the Sun Fire V20z server:

- Platform BIOS (preinstalled)
- SP Base Software (preinstalled)
- SP Value-Add Software (preinstalled)
- Java™ Runtime Environment (JRE) installation packages
- Network Share Volume
- Platform Software
- Motherboard platform drivers

All of these software packages are packaged with the NSV and are installed on the file server when the external file system is installed and configured.

For more complete information about the NSV, such as extracting and installing the NSV software, refer to the *Sun Fire V20z Server Installation Guide*, 817-5246-xx.

Serial Over LAN

The Serial Over LAN (SOL) feature lets servers transparently redirect the serial character stream from the baseboard Universal Asynchronous Receiver/Transmitter (UART) to and from the remote-client system over LAN. Serial over LAN has the following benefits compared to a serial interface:

- Eliminates the need for a serial concentrator
- Reduces the amount of cabling
- Allows remote management of servers without video, mouse or keyboard (headless servers)

Serial over LAN requires a properly configured LAN connection and a console from which an `ssh` session can be established.

In a Linux environment, you can use a shell such as `csh` or `ksh` as your console. This console works well in a scripting environment in which you might want to monitor many servers.

Enabling or Disabling the SOL Feature on the Server

Note – When the SOL feature is enabled, you cannot access the Sun Fire V20z server through the external DB9 serial port (COM A).

Note – The variable `spuser` is the user account created when securing the SP. The variable `spipaddr` is the IP address assigned to the SP.

For more information, see "Initial Setup of the Service Processor" on page 8.

You can enable or disable the SOL feature through the SP.

Enabling the SOL feature

To enable the feature, run the following command:

```
ssh -l <spuser> <spipaddr> platform set console -s sp -e
```

Disabling the SOL feature

To disable the feature, run the following command:

```
ssh -l <spuser> <spipaddr> platform set console -s platform
```

Launching an SOL Session

To launch an SOL session, run the following command:

```
ssh <spipaddr> -l <spuser> platform console
```

Terminating an SOL Session

To terminate an SOL session:

1. **Press Control-e.**
2. **Press 'c'.**
3. **Press '.'.**

You can also terminate an SOL session by terminating the `ssh` session:

1. **Press the tilde key (~).**
2. **Press '.'.**

APPENDIX **A**

Commands

This appendix describes the commands and subcommands mentioned in the *Sun Fire™ V20z Server Installation Guide* (817-5246-xx), the *Sun Fire V20z Server User Guide* (817-5246-xx) and this server management guide.

Using the `ssh` Protocol

You must use `ssh` to execute these commands on the Server Processor (SP). There are two ways to do this:

- Use the interactive shell on the SP.
- Preface each command with a set piece of text.

Interactive Shell on the SP

To use the interactive shell:

- **Log in to and authenticate on the interactive shell by running the command:**

```
ssh <spipaddr> -l <spuser>
```

Preface Text

- **Preface each command with the following text**

```
ssh <spipaddr> -l <spuser>
```

Commands

The following commands are described in this appendix:

- The `ipmi` commands manage IPMI functions on the server.
- The `platform` commands report or change some aspect of the state of the Sun Fire V20z server platform.
- The `sp` commands get or set configuration values for the Service Processor (SP), and generate or manage events and notices.

Note – For a complete listing of the diagnostics commands (`diags`), refer to the *Sun Fire™ V20z Server User Guide* (817-5246-xx).

TABLE A-1 lists the subcommands described in this appendix. A more-detailed explanation of each subcommand follows this table.

TABLE A-1 Subcommands in Appendix A

Subcommand	Description
`ipmi disable channel`	Disables one of two IPMI channels.
`ipmi enable channel`	Enables one of two IPMI channels.
`platform get console`	Retrieves the configuration of the SP access to the platform serial console.
`platform set console`	Configures the SP access to the platform serial console.
`platform get os state`	Retrieves the current state of the platform operating system (for example, running, booting, off and so on).
`platform set os state`	Reboots the platform into the default OS, BIOS setup or BIOS update, or shuts down the platform.
`sp update diags`	Updates the diagnostics to a newer version.
`sp delete event`	Clears an existing event using the event ID.
`sp get events`	Returns detailed information about all active SP events.
`sp get jnet`	Retrieves the JNET address.

TABLE A-1 Subcommands in Appendix A

Subcommand	Description
`sp set jnet`	Sets the JNET address.
`sp get locatelight`	Reads the value of the locatelight switch (which represents the state of the front and rear panel identification lights).
`sp set locatelight`	Sets the state of the locatelight switch.
`sp add mount`	Creates or resets a mountpoint.
`sp delete mount`	Deletes the specified mountpoint.
`sp get mount`	Displays the current mountpoints on the SP.
`sp get port 80`	Retrieves the last port 80 postcode from the PRS Port80 register.
`sp add snmp-destination`	Adds an SNMP destination.
`sp delete snmp-destination`	Deletes the SNMP destination.
`sp get snmp-destinations`	Displays the available SNMP destinations (IP address or host name) to which the SP is configured to send.
`sp get snmp proxy community`	Returns the community name currently being used by the SP SNMPD to proxy the platform SNMP agent.
`sp set snmp proxy community`	Sets the proxy entries that specify the OID to be referred, the IP address to which they are referred, and the community string to use while proxying.
`sp get tdulog`	Captures debug-dump data and stores it on the SP in compressed format.
`sp update flash all`	Sets the update flag to start the full flash update on the next SP reset.

Note – Every command returns a return code upon completion.

IPMI Commands

The `ipmi` command manages IPMI functions on the server.

IPMI Disable Channel Subcommand

Description: Allows you to disable one of two IPMI channels.

Command format:

```
ipmi disable channel {sms | lan}
```

TABLE A-2 lists the arguments for this subcommand.

TABLE A-2 Arguments for Subcommand `ipmi disable channel`

Arguments	Description
`sms`	The ID of the channel to disable for the System Interface; it is not case sensitive.
`lan`	The ID of the channel to disable for the LAN Interface; it is not case sensitive.

Return Codes

TABLE A-3 lists the return codes for this subcommand.

TABLE A-3 Return Codes for Subcommand `ipmi disable channel`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_InvalidArgument`	`4`	One or more arguments were incorrect or invalid.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.

IPMI Enable Channel Subcommand

Description: Allows you to enable one of two IPMI channels.

Command format:

```
ipmi enable channel {sms | lan}
```

TABLE A-4 lists the arguments for this subcommand.

TABLE A-4 Arguments for Subcommand `ipmi enable channel`

Arguments	Description
`sms`	The ID of the channel to enable for the System Interface; is not case sensitive.
`lan`	The ID of the channel to enable for the LAN Interface; is not case sensitive. If you are activating the LAN channel for the first time, you are prompted for a password to associate with the *null* user.

Return Codes

TABLE A-5 lists the return codes for this subcommand.

TABLE A-5 Return Codes for Subcommand `ipmi enable channel`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_InvalidArgument`	`4`	One or more arguments were incorrect or invalid.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.
`NWSE_NotImplemented`	`10`	Function not implemented.
`NWSE_ServiceNotAvailable`	`24`	Requested service is not available.

Platform Commands

The `platform` command reports or changes some aspect of the state of the Sun Fire V20z server's platform.

Platform Get Console Subcommand

Description: Retrieves the configuration information regarding the Service Processor access to the platform serial console.

Command format:

`platform get console [{-H|--noheader}] [{-D | --delim <DELIMITER>}]`

TABLE A-6 lists the arguments for this subcommand.

TABLE A-6 Arguments for Subcommand `platform get console`

Arguments	Description
`{-H |--noheader}`	Suppresses column headers.
`{ -D | --delim }`	Delimits columns with the specified delimiter. Headings are also delimited unless suppressed. The delimiter can be any character or string.

The following output displays when successful:

```
Rear Panel        Enabled Speed   Pruning     Log Trigger
SP Console        Yes     115200  No          1024KB
```

or

```
Platform COMA     No      19200   Yes         64KB
```

One of the other lines of data displays depending on whether the rear-panel serial port is connected to the platform or to the SP. TABLE A-7 describes the headings for the output columns.

TABLE A-7 Headings for the Output Columns

Heading	Description
`Enabled`	Displays `No` if the external serial port is connected to the platform. Otherwise, the external serial port is connected to the SP console and you can access the platform serial console through the SP command line.
`Speed`	Indicates the communications speed of the link.
`Prune`	Indicates whether ANSI escape code and duplicate information pruning is enabled.
`Log Trigger`	Indicates the approximate size at which log rotation occurs (for example, when the `console.0` file is removed, the current log is moved to `console.0` and a new log file is opened). Pruning of the log-file contents happens only when rotation occurs. The minimum size for a log file is 64KB, the maximum size is 1024KB.

Return Codes

TABLE A-8 lists the return codes for this subcommand.

TABLE A-8 Return Codes for Subcommand `platform get console`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	`2`	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	`3`	Unable to connect to the RPC server.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.

Platform Set Console Subcommand

Description: Enables configuration of SP access to the platform serial console, sets the speed of the connection, and limits the size of the log files created.

```
platform set console {--serial|-s} platform
```

This option configures the external serial port so that it is connected to the platform serial console. This is the default setting.

Command format:

```
platform set console {--serial|-s} sp
[{{--enable|-e}|{--disable|-d}}]
[{{--prune|-p}|{--noprune|-n}}] [{--speed|-S}
{1200|2400|4800|9600|19200|38400|115200}] [{--log|-l} size]
```

This option configures the external serial port so that it is connected to the SP serial console. You can then access the platform serial console through the SP command line.

Note – None of the arguments listed in TABLE A-9 can be used with `-s` set to `platform`.

TABLE A-9 lists the arguments for this subcommand.

TABLE A-9 Arguments for Subcommand `platform set console`

Arguments	Description
`{-S | --speed} {1200|2400|4800| 9600|19200|38400| 115200}`	Select the port speed for the platform console. BIOS, the platform operating system and the console must all be configured for the same speed.
`{-d | --disable}`	Indicates that the platform console monitor is inactive. Cannot be used with: `-e`.
`{-e | --enable}`	Indicates that the platform console monitor is active. Cannot be used with: `-d`.
`{-l | --log}size`	Select the trigger size in KB for console-log rotation. The acceptable values for log size are between 64 and 1024 inclusive.

TABLE A-9 Arguments for Subcommand `platform set console`

Arguments	Description
`{-n | --noprune}`	Indicates that the platform console log should be the raw console data. Cannot be used with: `-p`.
`{-p | --prune}`	Indicates that the platform console log is to be cleaned of ANSI sequences and pruned of duplicated information. Cannot be used with: `-n`.
`{-s | --serial} {sp|platform}`	Specify whether the serial port is connected to the platform COMA port or to the SP serial console. Cannot be used with: `-e` [platform] `-d` [platform] `-p` [platform] `-n` [platform] `-S` [platform] `-l` [platform].

Return Codes

TABLE A-10 lists the return codes for this subcommand.

TABLE A-10 Return Codes for Subcommand `platform set console`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	`2`	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	`3`	Unable to connect to the RPC server.
`NWSE_InvalidArgument`	`4`	One or more arguments were incorrect or invalid.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.
`NWSE_NoMemory`	`8`	Insufficient memory.
`NWSE_DeviceError`	`25`	Unable to read or write to the device.

Platform Get OS State Subcommand

Description: Retrieves the current state of the platform operating system.

Command format:

```
platform get os state
```

The values for the current state include:

- Off
- On
- Communicating
- Diagnostics
- Sleeping
- BIOS booting
- BIOS setup
- OS booting
- OS shutting down

These states display in the help panel when you move your mouse over the platform-operating-system button. As the state changes, the image on the button also changes.

When the platform is in the *communicating* state (in which the operating system is communicating with the SP), if the platform drivers are uninstalled, the SP remains in the *communicating* state even though it can no longer communicate with the platform.

For more information about setting the state, refer to "Platform Set OS State Subcommand" on page 67.

Return Codes

TABLE A-11 lists the return codes for this subcommand.

TABLE A-11 Return Codes for Subcommand `platform get os state`

Return Code	ID	Description
`NWSE_Success`	0	Command successfully completed.
`NWSE_InvalidUsage`	1	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	2	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	3	Unable to connect to the RPC server.
`NWSE_NoPermission`	6	Not authorized to perform this operation.

Platform Set OS State Subcommand

Description: Provides the ability to reboot the platform into the default OS, BIOS setup or BIOS update, or to shut down the platform. Rebooting to BIOS setup allows you to configure the BIOS parameters; rebooting to BIOS update allows you to reflash the BIOS image.

Command format:

```
platform set os state reboot [{-W | --nowait}]
[{-b | --bios}] [{-f |--forced}] [-q | --quiet]

platform set os state reboot-to-diags
[{-f |--forced}] [START | STOP]

platform set os state shutdown[{-W | --nowait}]
[{-f |--forced}] [-q | --quiet]

platform set os state update-bios [-q | --quiet] [{-W | --nowait}]
BIOS_IMAGE
```

TABLE A-12 lists the arguments for this subcommand.

TABLE A-12 Arguments for Subcommand `platform set os state`

Arguments	Description
`[-W | --nowait]`	If specified, the command returns immediately instead of waiting for the operation to complete.
`{-f | --forced}`	Results in a hard power off.
`[-b | --bios]`	Only applicable to `set os state reboot`. Allows reflashing of the BIOS image.
`[-q | --quiet]`	Suppresses interative warning messages. No error messages are blocked.
`BIOS_IMAGE`	Only applicable to `set os state update-bios`. Indicates the name of the file containing the new BIOS image to use to update the BIOS.

The `platform set os state` command waits for the platform to boot.

When the platform is in the *communicating* state (in which the operating system is communicating with the SP), if the platform drivers are uninstalled, the SP remains in the *communicating* state even though it can no longer communicate with the platform.

For a list of possible states, refer to "Platform Get OS State Subcommand" on page 66.

Return Codes

TABLE A-13 lists the return codes for this subcommand.

TABLE A-13 Return Codes for Subcommand `platform set os state`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	`2`	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	`3`	Unable to connect to the RPC server.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.

TABLE A-13 Return Codes for Subcommand `platform set os state`

Return Code	ID	Description
`NWSE_Busy`	9	Device or resource is busy.
`NWSE_FileError`	18	File open, missing, or read or write error occurred.
`NWSE_InvalidOpForState`	22	Invalid operation for current state.

SP Commands

The `sp` command gets or sets the configuration values for the SP, and generates or manages events and notices.

SP Update Diags Subcommand

Description: Updates the current version of diagnostics available.

While the SP functions normally without access to an external file system, a file system is required to enable several features, including diagnostics. The SP software uses a default version of diagnostics. However, if a new version is released and stored on the Network Share Volume, you must explicitly point to that new version to use it.

Command format:

`sp update diags {-p | --path} <`*`PATH_TO_DIAGS_FOLDER`*`>`

TABLE A-14 lists the argument for this subcommand.

TABLE A-14 Argument for Subcommand `sp update diags`

Argument	Description
`{-p	--path}`	Points to the location of the new diagnostics.

Return Codes

TABLE A-15 lists the return codes for this subcommand.

TABLE A-15 Return Codes for Subcommand `sp update diags`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_InvalidArgument`	`4`	One or more arguments were incorrect or invalid.
`NWSE_UnknownError`	`15`	Miscellaneous error not captured by other errors.

SP Delete Event Subcommand

Description: Clears an existing event using the event ID.

Command format:

`sp delete event` *`EVENT ID`* `[-a | --all] [-q | --quiet]`

TABLE A-16 lists the arguments for this subcommand.

TABLE A-16 Arguments for Subcommand `sp delete event`

Argument	Description
`EVENT ID`	Specifies the existing event to clear. This argument is repeatable to clear multiple events at one time.
`[-a | --all]`	Removes all events.
`[-q | --quiet]`	If the event to delete is not found, this argument specifies that no error be returned.

Return Codes

TABLE A-17 lists the return codes for this subcommand.

TABLE A-17 Return Codes for Subcommand `sp delete event`

Return Code	ID	Description
`NWSE_Success`	0	Command successfully completed.
`NWSE_InvalidUsage`	1	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	2	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	3	Unable to connect to the RPC server.
`NWSE_NotFound`	5	Entity (user, service, file, path, etc.) was not found.
`NWSE_NoPermission`	6	Not authorized to perform this operation.
`NWSE_NoMemory`	8	Insufficient memory.
`NWSE_InvalidOpForState`	22	Invalid operation for current state.

SP Get Events Subcommand

Description: Returns detailed information about all active Service Processor events. By default, event ID, last update, component, severity, and a message are displayed.

Administrators can view detailed information about all the currently active system events and perform various actions related to each event.

You can view this information in the System Events table, which contains a row for each unique active system event, or using this command. For a list of all possible events, refer to the TABLE 3-4 in Chapter 3.

Command format:

```
sp get events [ {-i | --id} <EVENT ID> ]
[{-d | --detail} ] [{ -v | --verbose}]
[{-H | noheader}][{-D | --delim <DELIMITER>}]
```

TABLE A-18 lists the arguments for this subcommand.

TABLE A-18 Arguments for Subcommand `sp get events`

Argument	Description
`{-i | --id}`	Specifies to display only information about this event; otherwise information for all existing events returns.
`{-d | --detail}`	Specifies to display the history of either one or all events.
`{ -v| --verbose}`	Specifies to display all columns.
`{ -H | --noheader }`	Suppresses column headings.
`{ -D | --delim }`	Specifies to delimit columns with the specified delimiter. Headings are also delimited unless suppressed. The delimiter can be any character or string.

Return Codes

TABLE A-19 lists the return codes for this subcommand.

TABLE A-19 Return Codes for Subcommand `sp get events`

Return Code	ID	Description
`NWSE_Success`	0	Command successfully completed.
`NWSE_InvalidUsage`	1	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	2	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	3	Unable to connect to the RPC server.
`NWSE_InvalidArgument`	4	One or more arguments were incorrect or invalid.
`NWSE_NotFound`	5	Entity (user, service, file, path and so on) not found.
`NWSE_NoMemory`	8	Insufficient memory.

SP Get JNET Subcommand

Description: Retrieves the IP address of the platform JNET driver.

Command format:

```
sp get jnet [{-H | --noheader}] [{-D | --delim <DELIMITER>}]
```

TABLE A-20 lists the arguments for this subcommand.

TABLE A-20 Arguments for Subcommand `sp get jnet`

Argument	Description
`{ -H | --noheader }`	Suppresses column headings.
`{ -D | --delim }`	Delimits columns with the specified delimiter. Headings are also delimited unless suppressed. The delimiter can be any character or string.

Return Codes

TABLE A-21 lists the return codes for this subcommand.

TABLE A-21 Return Codes for Subcommand `sp get jnet`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_NoMemory`	`8`	Insufficient memory.
`NWSE_Busy`	`9`	Device or resource is busy.
`NWSE_HostDown`	`14`	Host is down.

SP Set JNET Subcommand

Description: Sets or modifies the SP and platform network addresses for JNET. Because of the firewall between these drivers, you must specify both addresses at the same time.

Both the Service Processor and Platform JNET addresses must be on the same Class C subnet.

Command format:

`sp set jnet {-p | --platform}` *`IP ADDRESS`* `{-s | --sp}` *`IP ADDRESS`*

TABLE A-22 lists the arguments for this subcommand.

TABLE A-22 Arguments for Subcommand `sp set jnet`

Argument	Description
`{-p	--platform}`	Specifies the IP address for the platform.
`{-s	--sp}`	Specifies the IP address for the SP.

Note – If you change the default addresses of JNET using this command and then re-install the platform operating system or reset the SP through the `sp reset to default-settings` command, you must re-issue the `sp set jnet` command to re-establish the JNET connection.

Otherwise the connection will be out-of-sync (one address will be modified and one will be re-set to the default address.)

Return Codes

TABLE A-23 lists the return codes for this subcommand.

TABLE A-23 Return Codes for Subcommand `sp set jnet`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.
`NWSE_NoMemory`	`8`	Insufficient memory.
`NWSE_Busy`	`9`	Device or resource is busy.
`NWSE_HostDown`	`14`	Host is down.

SP Get Locatelight Subcommand

Description: Reads the value of the locatelight switch (which represents the state of the front and rear panel identification lights).

Command format:

```
sp get locatelight
```

Return Codes

TABLE A-24 lists the return codes for this subcommand.

TABLE A-24 Return Codes for Subcommand `sp get locatelight`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	`2`	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	`3`	Unable to connect to the RPC server.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.

SP Set Locatelight Subcommand

Description: Sets the state of the locatelight switch (which describes the state of the front and rear panel identification lights).

Command format:

```
sp set locatelight {blink | off}
```

Return Codes

TABLE A-25 lists the return codes for this subcommand.

TABLE A-25 Return Codes for Subcommand `sp set locatelight`

Return Code	ID	Description
`NWSE_Success`	0	Command successfully completed.
`NWSE_InvalidUsage`	1	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	2	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	3	Unable to connect to the RPC server.
`NWSE_NoPermission`	6	Not authorized to perform this operation.

SP Add Mount Subcommand

Description: Creates or resets a mountpoint.

Command format:

```
sp add mount [{-l | --local} MOUNTPOINT ]
{-r | --remote} SERVER:FILESYSTEM
[{-u|--user} USER] [{-p|--password} PASSWORD]
```

TABLE A-26 lists the arguments for this subcommand.

TABLE A-26 Arguments for Subcommand `sp add mount`

Argument	Description
`{-l | --local}`	*Optional*; Specifies the local mount point. The only mount point supported is `/mnt`.
`{-r | --remote}`	Specifies the remote server and file system. If *SERVER* specifies a host name, DNS must be properly configured.
`{-u | --user}`	Specifies the user name for the mount. Only required for SMB.
`{-p | --password}`	Specifies the password for the mount user. Only required for SMB.

Note – Several error messages may appear when executing an `smb` mount while mounting windows partitions. Check that the mount succeeded after the call by running the subcommand `sp get mount`.

Return Codes

TABLE A-27 lists the return codes for this subcommand.

TABLE A-27 Return Codes for Subcommand `sp add mount`

Return Code	ID	Description
`NWSE_Success`	0	Command successfully completed.
`NWSE_InvalidUsage`	1	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_InvalidArgument`	4	One or more arguments were incorrect or invalid.
`NWSE_NoPermission`	6	Not authorized to perform this operation.
`NWSE_NoMemory`	8	Insufficient memory.
`NWSE_Busy`	9	Device or resource is busy.

TABLE A-27 Return Codes for Subcommand `sp add mount`

Return Code	ID	Description
`NWSE_RPCConnected`	11	RPC client already connected.
`NWSE_RPCConnRefused`	12	RPC connection refused.
`NWSE_NoRouteToHost`	13	No route to host (network down).
`NWSE_HostDown`	14	Host is down.
`NWSE_UnknownError`	15	Miscellaneous error not captured by other errors.

SP Delete Mount

Description: Deletes a mountpoint.

Command format:

`sp delete mount` *`LOCAL MOUNT POINT`* `[-q | --quiet]`

TABLE A-28 lists the arguments for this subcommand.

TABLE A-28 Arguments for Subcommand `sp delete mount`

Argument	Description
`LOCAL MOUNT POINT`	Specifies the mount point to remove. If you do not specify the local mount point, `/mnt` is implicit as the default value.
`[-q	--quiet]`	If the mount point to delete is not found, this argument specifies that no error be returned.

Return Codes

TABLE A-29 lists the return codes for this subcommand.

TABLE A-29 Return Codes for Subcommand `sp delete mount`

Return Code	ID	Description
`NWSE_Success`	0	Command successfully completed.
`NWSE_InvalidUsage`	1	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_InvalidArgument`	4	One or more arguments were incorrect or invalid.
`NWSE_NoPermission`	6	Not authorized to perform this operation.
`NWSE_NoMemory`	8	Insufficient memory.

TABLE A-29 Return Codes for Subcommand `sp delete mount`

Return Code	ID	Description
`NWSE_Busy`	9	Device or resource is busy.
`NWSE_RPCConnected`	11	RPC client already connected.
`NWSE_RPCConnRefused`	12	RPC connection refused.
`NWSE_NoRouteToHost`	13	No route to host (network down).
`NWSE_HostDown`	14	Host is down.
`NWSE_UnknownError`	15	Miscellaneous error not captured by other errors.

SP Get Mount Subcommand

Description: Displays the current mount points on the SP.

Command format:

`sp get mounts [{-l | --local} MOUNTPOINT] [-H | --noheader] [{-D | --delim <DELIMITER>}]`

TABLE A-30 lists the arguments for this subcommand.

TABLE A-30 Arguments for Subcommand `sp get mount`

Arguments	Description
`{-l	--local}`	Specifies the local mountpoint. If you do not specify `-l`, `/mnt` is implicit as the local mount point.
`{ -H	--noheader }`	Suppresses column headings.
`{ -D	--delim }`	Delimits columns with the specified delimiter. Headings are also delimited unless suppressed. The delimiter can be any character or string.

Return Codes

TABLE A-31 lists the return codes for this subcommand.

TABLE A-31 Return Codes for Subcommand `sp get mount`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_InvalidArgument`	`4`	One or more arguments were incorrect or invalid.
`NWSE_NotFound`	`5`	Entity (user, service, file, path and so on) not found.
`NWSE_NoMemory`	`8`	Insufficient memory.
`NWSE_Busy`	`9`	Device or resource is busy.
`NWSE_RPCConnected`	`11`	RPC client already connected.
`NWSE_RPCConnRefused`	`12`	RPC connection refused.
`NWSE_NoRouteToHost`	`13`	No route to host (network down).
`NWSE_HostDown`	`14`	Host is down.
`NWSE_NotMounted`	`21`	File system is not mounted.

SP Get Port 80 Subcommand

Description: Retrieves the last Port 80 post code from the PRS Port80 register. The register is written by platform BIOS during platform boot. The command is used to debug platform boot problems.

Command format:

```
sp get port80 {-m | --monitor}
```

TABLE A-32 lists the argument for this subcommand.

TABLE A-32 Argument for Subcommand `sp get port80`

Argument	Description
`{-m | --monitor}`	Allows for continuous monitoring of the port 80 traffic.

You can also retrieve the last ten Port 80 post codes using the operator panel.

For more details about using the operator-panel menus or for a complete list of the BIOS Power On Self Test (POST) codes, refer to the *Sun Fire V20z Server User Guide* (817-5248-xx).

Return Codes

TABLE A-33 lists the return codes for this subcommand.

TABLE A-33 Return Codes for Subcommand `sp get port80`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.
`NWSE_NoMemory`	`8`	Insufficient memory.
`NWSE_ServiceNotAvailable`	`24`	Requested service is not available.

SP Add SNMP Destination Subcommand

Description: Adds a single SNMP destination, (either IP address or host name).

Command format:

`sp add snmp-destination` *`IP ADDRESS/HOSTNAME`*

TABLE A-34 lists the argument for this subcommand.

TABLE A-34 Argument for Subcommand `sp add snmp-destination`

Arguments	Description
`IP ADDRESS/HOSTNAME`	Specifies the IP address or name of the host for the destination you wish to add. This argument is repeatable to add multiple destinations at one time; however, the number of destinations you can create is limited due to memory constraints.

Return Codes

TABLE A-35 lists the return codes for this subcommand.

TABLE A-35 Return Codes for Subcommand `sp add snmp-destination`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	`2`	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_InvalidArgument`	`4`	One or more arguments were incorrect or invalid.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.
`NWSE_NoMemory`	`8`	Insufficient memory.
`NWSE_RPCConnRefused`	`12`	RPC connection refused.
`NWSE_UnknownError`	`15`	Miscellaneous error not captured by other errors.
`NWSE_FileError`	`18`	File open, missing, or read or write error occurred.
`NWSE_Exist`	`19`	Entity (user, service, etc.) already exists.

SP Delete SNMP Destination Subcommand

Description: Deletes a single SNMP destination (either IP address or host name)

Command format:

`sp delete snmp-destination` *`IP ADDRESS/HOSTNAME`* `[-a | --all] [-q | --quiet]`

TABLE A-36 lists the arguments for this subcommand.

TABLE A-36 Arguments for Subcommand `sp delete snmp-destination`

Arguments	Description
`IP ADDRESS/HOSTNAME`	Specifies the IP address or hostname of the destination to remove. This argument is repeatable to remove multiple destinations at one time.
`[-a | --all]`	Removes all SNMP destinations.
`[-q | --quiet]`	If the SNMP destination to delete is not found, this argument specifies that no error be returned.

Return Codes

TABLE A-37 lists the return codes for this subcommand.

TABLE A-37 Return Codes for Subcommand `sp delete snmp-destination`

Return Code	ID	Description
`NWSE_Success`	0	Command successfully completed.
`NWSE_InvalidUsage`	1	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	2	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_InvalidArgument`	4	One or more arguments were incorrect or invalid.
`NWSE_NotFound`	5	Entity (user, service, file, path and so on) not found.
`NWSE_NoPermission`	6	Not authorized to perform this operation.
`NWSE_NoMemory`	8	Insufficient memory.
`NWSE_RPCConnRefused`	12	RPC connection refused.
`NWSE_UnknownError`	15	Miscellaneous error not captured by other errors.
`NWSE_FileError`	18	File open, missing, or read or write error occurred.

SP Get SNMP Destinations Subcommand

Description: Displays the available SNMP destinations (IP address or host name) to which the Service Processor is configured to send. Many networking programs use this information to identify the machine.

Command format:

```
sp get snmp-destinations
```

Return Codes

TABLE A-38 lists the return codes for this subcommand.

TABLE A-38 Return Codes for Subcommand `sp get snmp-destinations`

Return Code	ID	Description
`NWSE_Success`	0	Command successfully completed.
`NWSE_InvalidUsage`	1	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	2	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	3	Unable to connect to the RPC server.
`NWSE_NoMemory`	8	Insufficient memory.
`NWSE_UnknownError`	15	Miscellaneous error not captured by other errors.
`NWSE_FileError`	18	File open, missing, or read or write error occurred.

SP Get SNMP Proxy Community Subcommand

Description: Returns the community name the Service Processor is currently using to proxy the platform SNMP agent.

Command format:

`sp get snmp proxy community` *`COMMUNITY_STRING`*

TABLE A-39 lists the argument for this subcommand.

TABLE A-39 Argument for Subcommand `sp get snmp proxy community`

Argument	Description
`COMMUNITY_STRING`	Specifies the name of the community the SP is currently using.

There are no restrictions on the length of the community strings; common names are *private* and *public*. The default name of the community string is *private*.

If you run the `sp get snmp proxy community` command without setting it, the return valule is `private`. Otherwise, you can set it to any string.

Return Codes

TABLE A-40 lists the return codes for this subcommand.

TABLE A-40 Return Codes for Subcommand `sp get snmp proxy community`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	`2`	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	`3`	Unable to connect to the RPC server.

SP Set SNMP Proxy Community Subcommand

Description: The SNMP agent on the Service Processor acts as a proxy for the master SNMP agent running on the platform. These proxy entries specify the OID to be referred, the IP to which they are referred, and the community string to use while proxying. The community string is the value configured on the platform-side SNMP configuration.

Command format:

`sp set snmp proxy community` *`COMMUNITY STRING`*

TABLE A-41 lists the argument for this subcommand.

TABLE A-41 Argument for Subcommand `sp set snmp proxy community`

Argument	Description
`COMMUNITY STRING`	Specifies the name of the community to configure.

There are no restrictions on the length of the community strings; common names are *private* and *public*. The default name of the community string is *private*, but you can set it to any string.

Return Codes

TABLE A-42 lists the return codes for this subcommand.

TABLE A-42 Return Codes for Subcommand `sp set snmp proxy community`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_RPCTimeout`	`2`	Request was issued, but was not serviced by the server. RPC procedure timed out and the request may or may not have been serviced by the server.
`NWSE_RPCNotConnected`	`3`	Unable to connect to the RPC server.

SP Get TDULog Subcommand

Description: The Troubleshooting Dump Utility (TDU) captures debug data. When you execute this command, this data is gathered and stored on the SP in a compressed tar file.

Command format:

```
sp get tdulog [{-f | --filename} FILENAME or STDOUT
[{-c | --cpuregs} CPU REGISTERS]
[{-p | --pciregs} PCI REGISTERS]
[{-r | --reset} RESET PLATFORM]
```

TABLE A-43 lists the arguments for this subcommand.

TABLE A-43 Arguments for Subcommand `sp get tdulog`

Argument	Description
`{-f | --filename}`	*Optional*. The name of the output file to which the log files are copied, or the fully qualified path name. File names cannot contain the forward slash character (/), backward relative path reference (..), or the less than symbol (<). The following log files are created by default: `envLog`: contains the environment variables `vpdLog`: contains raw VPD data Additional log files are created for CPU2 and CPU3 registers. The TDU data can also be redirected to `stdout`. If the file name is `stdout`, the output is sent to `stdout` and the log files are not created. An NFS-mounted file share must be used to store the output file. If you do not provide a file name, it creates a file named `tdulog.tar` in `/logs/<`*hostname*`>`, where the <*hostname*> is the host name of the SP. If the host name is `localhost`, then the MAC address is used instead.
`{-c | --cpuregs}`	Reads the K-8 registers (GPRs, MSRs, TCB and machine check) from up to four CPUs.
`{-p | --pciregs}`	Reads all PCI registers on the system.
`{-r | --reset}`	Resets the platform if unable to access HDT mode.

The register name, address and data are logged to a file. For example, the information for CPU0 is shown in TABLE A-44.

TABLE A-44 Sample Information for Subcommand `sp get tdulog` on CPU0

Reg Name	Reg Addr	Reg Data
MSR_MCG_CAP_MSR	0xc0020179	0x0000000000000105
MSR_MCG_STAT_MSR	0xc002017a	0x0000000000000000
MSR_MCG_CTL_MSR	0xc002017b	0x000000000000001F
MSR_MC0_CTL	0xc0020400	0x000000000000007F

Return Codes

TABLE A-45 lists the return codes for this subcommand.

TABLE A-45 Return Codes for Subcommand `sp get tdulog`

Return Code	ID	Description
NWSE_Success	0	Command successfully completed.
NWSE_InvalidUsage	1	Invalid usage: bad parameter usage, conflicting options specified.
NWSE_NotFound	5	Entity (user, service, file, path and so on) was not found.
NWSE_NoPermission	6	Not authorized to perform this operation.
NWSE_MissingArgument	7	Missing argument(s).
NWSE_UnknownError	15	Miscellaneous error not captured by other errors.
NWSE_FileError	18	File open, missing, or read or write error occurred.
NWSE_NotMounted	21	File system is not mounted.
NWSE_ServiceNotAvailable	24	Requested service is not available.

SP Update Flash All Subcommand

Description: Checks the server availability and sets environment variables for the `update sp flash image` components from PPCBOOT. After verifying that the update server is running, the command sets the update flag to start the full flash update at the next SP reset. It also sets the server IP address and optional server port number in the environment variables.

Command format:

```
sp update flash all {-i | --ipaddress}
<IP ADDRESS xxx.xxx.xxx.xxx> [{-p | --port}] <PORT#>
```

TABLE A-46 lists the arguments for this subcommand.

TABLE A-46 Arguments for Subcommand `sp update flash all`

Argument	Description
`{-i | --serverip}`	The IP address of the update server on which the update server is running. The update server also contains the flash images.
`{-p | --port}`	*Optional*: The port number on the update server to which the Service Processor connects for the image updates. If the port number is not provided, the Service Processor tries to connect to the default port. The default port number is 52708.

Return Codes

TABLE A-47 lists the return codes for this subcommand.

TABLE A-47 Return Codes for Subcommand `sp update flash all`

Return Code	ID	Description
`NWSE_Success`	`0`	Command successfully completed.
`NWSE_InvalidUsage`	`1`	Invalid usage: bad parameter usage, conflicting options specified.
`NWSE_NoPermission`	`6`	Not authorized to perform this operation.
`NWSE_NoMemory`	`8`	Insufficient memory.
`NWSE_UnknownError`	`15`	Miscellaneous error not captured by other errors.
`NWSE_ServiceNotAvailable`	`24`	Requested service is not available.