Solstice™ SyMON™ 1.4 User's Guide

Solaris™ Version 2.6

Sun Microsystems
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A. 415-960-1300

Part No. 802-7302-10
August 1997, Revision A

Copyright 1997 Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX® system, licensed from Novell, Inc., and from the Berkeley 4.3 BSD system, licensed from the University of California. UNIX is a registered trademark in the United States and other countries and is exclusively licensed by X/Open Company Ltd. Third-party software, including font technology in this product, is protected by copyright and licensed from Sun's suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

Sun, Sun Microsystems, the Sun logo, Solaris, Solstice, Solstice SyMON, X11/NeWS, JumpStart, Sun-4, Ultra, Enterprise, Solstice Site Manger, Solstice Domain Manager, Solstice Enterprise Manager, SunVTS, and OpenWindows, are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. PostScript is a trademark of Adobe Systems, Inc., which may be registered in certain jurisdictions.

The OPEN LOOK® and Sun™ Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

X Window System is a trademark of X Consortium, Inc.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

Please
Recycle

Copyright 1997 Sun Microsystems Inc., 2550 Garcia Avenue, Mountain View, Californie 94043-1100, U.S.A.Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou de sa documentation associée ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Des parties de ce produit pourront être dérivées du système UNIX® licencié par Novell, Inc. et du système Berkeley 4.3 BSD licencié par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Sun, Sun Microsystems, le logo Sun, Solaris, Solstice, SyMON, X11/NeWS, JumpStart, Sun-4, Ultra, Enterprise, Solstice Site Manger, Solstice Domain Manager, Solstice Enterprise Manager, et OpenWindows, sont des marques déposées ou enregistrées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC, utilisées sous licence, sont des marques déposées ou enregistrées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

Les interfaces d'utilisation graphique OPEN LOOK® et Sun™ ont été développées par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant aussi les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

Le système X Window est un produit du X Consortium, Inc.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" SANS GARANTIE D'AUCUNE SORTE, NI EXPRESSE NI IMPLICITE, Y COMPRIS, ET SANS QUE CETTE LISTE NE SOIT LIMITATIVE, DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DES PRODUITS A RÉPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ILS NE SOIENT PAS CONTREFAISANTS DE PRODUITS DE TIERS.

Contents

Tables

Figures

Preface

Solstice™ SyMON™ is a system monitoring tool that collects various system information and displays it through a graphical user interface (GUI). This manual, which explains how to install and use Solstice SyMON, also describes how to modify the rules used to monitor the server. It also explains how to customize Solstice SyMON.

How This Book Is Organized

This manual is divided into six chapters and four appendixes:

Chapter 1, "Solstice SyMON Overview," provides an overview of the features and architecture.

Chapter 2, "Installing Solstice SyMON," describes how to install the software.

Chapter 3, "Using the Solstice SyMON Consoles," provides instructions on how to use the consoles.

Chapter 4, "Understanding and Writing Event Rules," explains how to modify event rules to meet specific monitoring needs. This chapter describes the basic Tcl commands and gives examples of event rules.

Chapter 5, "Command Line Options," describes the options you can invoke on the command line.

Chapter 6, "Troubleshooting" describes troubleshooting tips.

Appendix A, "Kernel Reader," provides a summary of the Kernel Reader data hierarchy.

Appendix B, "Config Reader," provides a summary of the Configuration Reader data hierarchy.

Appendix C, "Installing and Setting Up SNMP Traps," describes how to install and set up SNMP traps.

Appendix D, "Default Solstice SyMON Rules," describes the default Solstice SyMON rules.

Related Documents

Refer to the following documents for more information on the Tcl scripting language and Sun system performance tuning:

- *Tcl and the Tk Toolkit*, John K. Ousterhout, Addison-Wesley Publishing Company, 1994.
- *Practical Programming in Tcl and Tk*, Brent B. Welch, Prentice Hall, 1995.
- Reference card on Tcl, available from Specialized Systems Consultants, Inc., P.O. Box 55549, Seattle WA, 98155-0549.
- *Sun Performance and Tuning*, Adrian Cockcroft, SunSoft Press, 1995.
- *X Window System User's Guide, Motif Edition (Volume 3)*, O'Reilly and Associates, 1993.

Typographic Conventions

The following table describes the typographic conventions used in this book.

TABLE P-1 Typographic Conventions

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

Shell Prompts

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P-2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell superuser	#

Ordering Sun Documents

SunDocs[SM] is a distribution program for Sun Microsystems technical documentation. Contact SunExpress for easy ordering and quick delivery. You can find a listing of available Sun documentation on the World Wide Web.

TABLE P-3 SunExpress Contact Information

Country	Telephone	Fax
Belgium	02-720-09-09	02-725-88-50
Canada	1-800-873-7869	1-800-944-0661
France	0800-90-61-57	0800-90-61-58
Germany	01-30-81-61-91	01-30-81-61-92
Holland	06-022-34-45	06-022-34-46
Japan	0120-33-9096	0120-33-9097
Luxembourg	32-2-720-09-09	32-2-725-88-50
Sweden	020-79-57-26	020-79-57-27
Switzerland	0800-55-19-26	0800-55-19-27
United Kingdom	0800-89-88-88	0800-89-88-87
United States	1-800-873-7869	1-800-944-0661
World Wide Web: `http://www.sun.com/sunexpress/`		

Sun Welcomes Your Comments

Please use the *Reader Comment Card* that accompanies this document. We are interested in improving our documentation and welcome your comments and suggestions.

If a card is not available, you can email or fax your comments to us. Please include the part number of your document in the subject line of your email or fax message.

- Email:—`smcc-docs@sun.com`
- Fax:—SMCC Document Feedback, 1-415-786-6443

CHAPTER 1

Solstice SyMON Overview

This chapter gives an overview of the main features of the Solstice SyMON system monitor, which offers simple yet powerful monitoring capabilities for the following servers:

- Ultra™ Enterprise™ 3000, 4000, 5000, and 6000 servers
- Ultra Enterprise 450 server
- Ultra Enterprise 150 server
- SPARCserver™ 1000/1000E servers
- SPARCcenter™ 2000/2000E servers

Solstice SyMON uses a CDE/Motif™-based graphical user interface (GUI).

Product Features

Solstice SyMON identifies a range of hardware and system status states quickly. For example, it can monitor a major condition such as a CPU failure or a minor condition such as low swap space. You can also monitor hardware performance to detect incipient hardware failures, such as soft read errors on a disk.

To give you this critical performance information, Solstice SyMON analyzes system performance in real time; when performance problems occur, the event system alerts you, if desired, to the status of most system components.

TABLE 1-1 lists the main features.

TABLE 1-1 Solstice SyMON Features

Feature	Description
Performance Monitoring	Diagnoses and addresses potential problems such as capacity problems or bottlenecks. Solstice SyMON monitors four categories of performance data: CPU, disk, memory, and network.
Configuration Monitoring	Displays physical and logical views of exact server configurations; configuration monitoring improves system serviceability.
Remote Monitoring	Allows a server within a network running Solstice SyMON to be monitored from any location in the network.
Fault Management	Isolates potential problems or failed components. Provides access to system log file and maintains a log file of conditions for future analysis.
Graphical User Interface (GUI)	Ensures that users get the information they need quickly and easily with the configurable GUI. The GUI provides access to SunVTS™ diagnostics, which diagnoses hardware.

Software Components

Solstice SyMON consists of three interdependent subsystems:

- **Server subsystem**—The server subsystem consists of a set of *server agents* on the monitored server, the Kernel Reader, Config Reader, and Log Scanner (see FIGURE 1-2 on page 4). The agents continuously monitor the hardware status of the server.
- **Event Manager**—The Event Manager consists of the Event Generator and the Event Viewer. The Event Generator compares the conditions on the server to a set of pre-defined rules. If the conditions defined in the rule exist on the server, then the actions specified in the rule will be taken.

  You typically configure the Event Generator on a machine other than the one being monitored so that it will continue to run even if the monitored server is down. Each instance of the Event Generator monitors one server.

  For more information on the Event Manager, see “Event Manager” later in this chapter.

- **GUI**—You use the GUI, consisting of seven consoles, to view information generated by the Event Generator and Server subsystem, and to control the execution of SunVTS. You may install the GUI on several machines so others can remotely monitor from multiple locations. The software on the Event Generator communicates to the GUI machine by the GUI polling the server and the Event Generator. In general, the Solstice SyMON architecture is a polling/pulling once, not an asynchronous push type.

FIGURE 1-1 shows the relationship and flow of information between these subsystems.

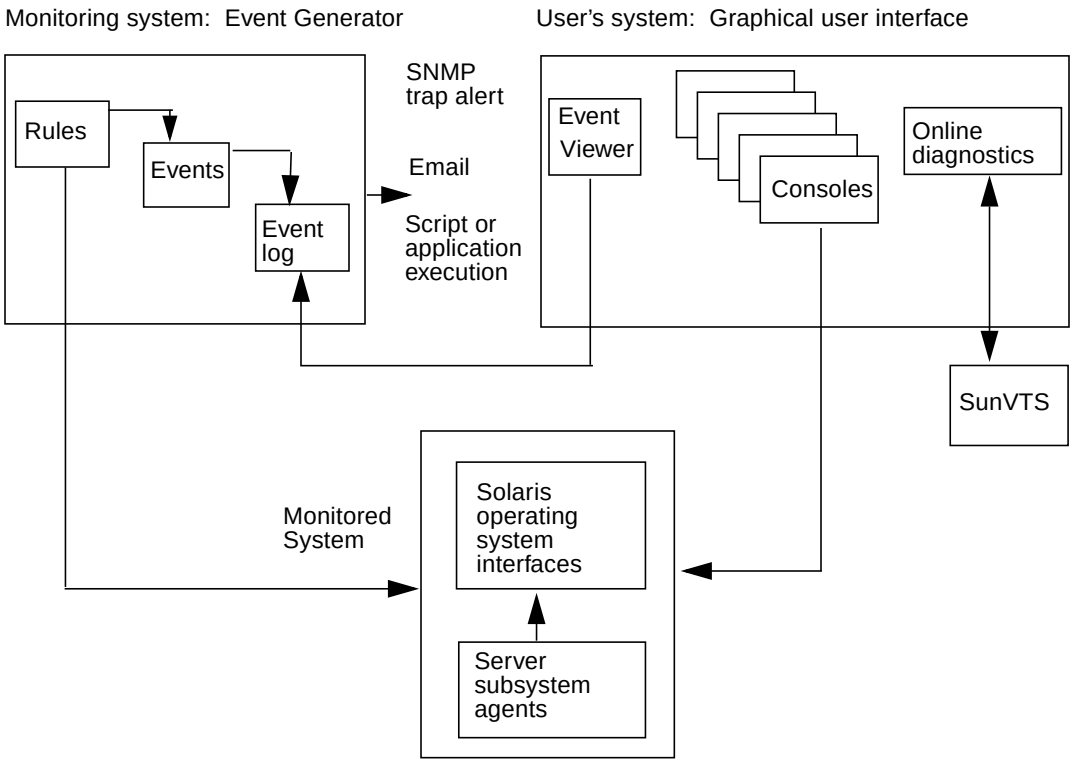

FIGURE 1-1 Process Flow Overview

Server Subsystem

The server subsystem consists of four agents (see FIGURE 1-2):

- Kernel Reader
- Config Reader
- Log Scanner
- Symond

These agents are briefly described in the following sections.

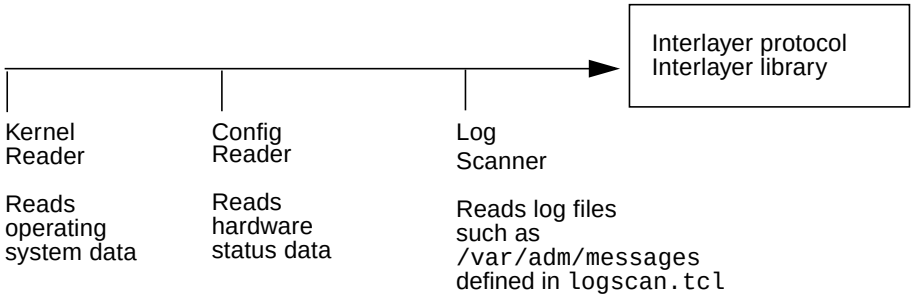

FIGURE 1-2 Server Subsystem Agents

Kernel Reader

The Kernel Reader extracts summary and process operating system data from the Solaris operating system kernel on a periodic basis. It determines which processes are CPU- and resource-intensive. Also, it monitors and reports on items such as:

- CPU usage
- Disk I/O
- Network I/O
- Memory usage
- Swap space usage
- Paging and swapping rates

Config Reader

The Config Reader monitors the server hardware. It reports system configuration data such as the number of CPU boards and the number and type of I/O boards. It monitors and tracks changes to the system configuration and the state of its components.

It reports on items such as:

- Disk hot plug
- Board hot swap
- Board temperature

Log Scanner

The Log Scanner monitors designated system log files searching for a user-specified list of regular expressions as specified in the rules. When the Log Scanner finds a set of messages that matches that list, it notifies the Event Generator and triggers the corresponding event rule.

The Log Scanner can also run in a demand-driven mode from the GUI. It looks for specific entries as indicated by user-specified search parameters.

It searches the `/var/adm/messages` system log. The Log Scanner can capture panic messages immediately preceeding a system crash.

`symond` Daemon

The `symond` daemon starts when the server or the event host is started. `symond` starts the agents, monitors their activity, and restarts them if they stop. It also provides a central contact point for any clients that want to connect to the agents.

Event Manager

The Event Manager subsystem consists of the Event Generator and the Event Viewer. The Event Generator evaluates the conditions on the server against a set of rules that defines an *event*. You can modify these rules. When an event is logged, you can set the software to take an action, such as highlighting a component in the display or executing a script.

A pre-defined set of rules, written in the Tcl scripting language, is provided with Solstice SyMON. FIGURE 1-3 illustrates the Event Manager monitoring system by showing how you can modify events or define new events that tell the software what to do when a particular event occurs.

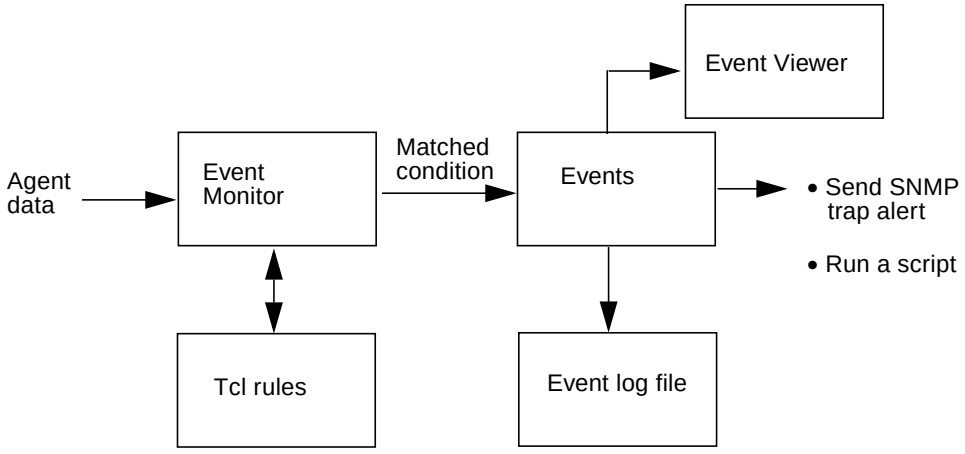

FIGURE 1-3 Event Manager Subsystem

The Event Generator polls for information about the server. Event Generator polling the Server Subsystem this is how the Server Subsystem communicates to the Event Generator (the monitoring machine). It evaluates the conditions of the server against a set of pre-defined *rules*. If the conditions of the server meet the defined set of conditions in the rule, an event is created and the defined actions of the rule are triggered. At this point, the event is known as an *open event*.

Point events are events that are generated due to log messages. Because they are closed right away, they are displayed in the All Events display of the Event Viewer but not by the Open Events display of the Event Viewer.

Once an event is open, the defined actions of the rule may include sending email or running a script that alerts you of this event. Routines are also available so events can generate a SNMP trap, which are directed to the Solstice Site Manager™ and Solstice Domain Manager™, Solstice Enterprise Manager,™ or other SNMP-listening management platforms.

When the condition that generated the event no longer exists, the Event Manager closes the event and the event is referred to as a *closed event*. The event log file records the opening and closing of events.

Graphical User Interface

When you start Solstice SyMON, the main GUI window, called the Launcher, is displayed (see FIGURE 1-4).

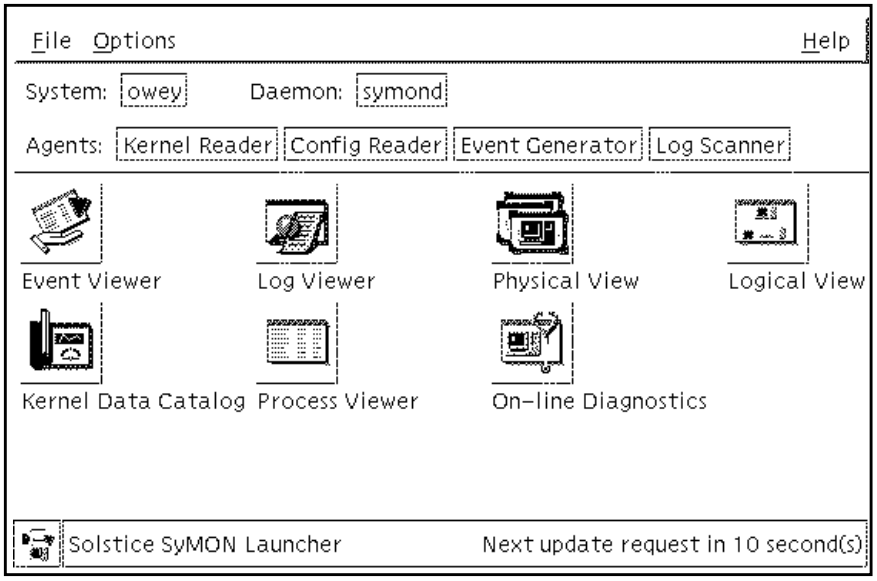

FIGURE 1-4 Launcher

Launcher and Consoles

The Launcher provides an easy to use, windowed display showing vital signs for Solstice SyMON and top level status information of the monitored server. The Launcher enables you to launch consoles (subwindows) to view data about the conditions of the server in a variety of ways. For example, you can view data on events, log files, physical or logical views, kernel information, or process information. The System: box blinks green when Solstice SyMON is operating normally.

The Launcher does the following:

- Tests for available agents (Kernel Reader, Config Reader, Event Generator, and Log Scanner)
- Displays the console icons and names; each icon displays an “unavailable” symbol (Ø) until it receives data from the appropriate agent; for more information on the unavailable symbol, see “Launcher Window” in **Chapter 3, “Using the Solstice SyMON Consoles**”

- Shows the status of agents; red indicates that the GUI is unable to contact the agent, yellow indicates that the GUI has found the agent but is not yet receiving data
- Initiates online diagnostics through SunVTS, a powerful diagnostic tool that incorporates multi-functional tests of the system through operating system-level calls

There are seven consoles on the Launcher. TABLE 1-2 describes the information presented by each console.

TABLE 1-2 Solstice SyMON Consoles

Console	Feature
Event Viewer	Displays events that have occurred on the server
Log Viewer	Displays log files; for example, `syslog`
Physical View	Displays a pictorial representation of the server; any component with an open event is highlighted
Logical View	Displays a hierarchical schematic view of the server; any component with an open event is highlighted
Kernel Data Catalog	Displays a hierarchy of metrics showing CPU, disk, memory, and network performance; any metric that causes an event is highlighted
Process Viewer	Displays resource-intensive processes running on the server
Online Diagnostics	Initiates the SunVTS diagnostics GUI

TABLE 1-3 describes menu items that are typical in each of the seven consoles listed in TABLE 1-2.

TABLE 1-3 Menu Items

Menu	Description
File	Gives you the option to close the window, exit the GUI, save a file or settings, or restore a file or settings
Edit	Offers different ways to manipulate windows, objects, or text
View	Lists options for viewing data or for displaying additional information about the data
Help	Provides access to online help

For detailed information on the Launcher and its consoles, see **Chapter 3, "Using the Solstice SyMON Consoles**."

Information Messages

When an operation that is monitored is incorrect, the software generates information messages in the footer of the active window.

Error and Warning Messages

Solstice SyMON displays warning messages in dialog boxes that appear above the console you are viewing. When a warning message is displayed, you should acknowledge it by selecting an action button such as O. K. or Quit.

Some warning messages state that a prerequisite for an operation has not been met. To acknowledge this message, click the Continue button before attempting the operation again.

Monitoring Hardware, System Performance, Alarms, and Events

This section provides an overview of the consoles used to monitor:

- Hardware status
- System performance
- Alarms and events

Chapter 3, "Using the Solstice SyMON Consoles" explains these consoles in more detail.

Monitoring Hardware Status

Use Solstice SyMON to view the configuration of your system, monitor the status of system hardware components, and launch diagnostic tests. The three consoles used to monitor hardware status are:

- Physical view
- Logical view
- Online diagnostics

Physical View

The Physical View, a pictorial representation of the front and/or back of the server, provides a picture of your system as it is actually configured. Information about each component and visual identification of a failed or troubled component is displayed.

Logical View

The Logical View, a companion view for the Physical View, displays the components in a schematic hierarchy.

Online Diagnostics

To test and validate the CPUs, disks, and network connections of a server, the software launches SunVTS, Sun Validation and Test Suite. For information about running SunVTS, refer to the *SunVTS 2.1 User's Guide* and the *SunVTS 2.1 Test Reference Manual* on the Solaris 2.6 on Sun Hardware AnswerBook on the Supplement for Solaris 2.6 for Sun Microsystems Computer Company CD.

Monitoring System Performance

Solstice SyMON monitors system performance to isolate and identify system bottlenecks such as:

- Overloaded CPU
- Excessive traffic to a single disk

The three consoles that monitor system status are:

- Kernel Data Catalog
- Log Viewer
- Process Viewer

Additionally, System Meters, which are not consoles, also monitor system status.

Kernel Data Catalog

The Kernel Data Catalog provides a hierarchical diagram of the system performance parameters by category (CPU, memory, disk, and network).

System Meters

You can display the performance parameters that the software captures as graphs in the System Meter console windows. These graphs in the System Meter console window show you the type and quantity of activities of the CPUs, disks, memory, and network interfaces of the server (see TABLE 1-4). System Meter consoles do not appear on the Launcher, since they are built using the Kernel Data Catalog and the Logical View.

TABLE 1-4 Performance Parameters

Component	Performance Parameters
CPU	Performance across all CPUs on a server or individual CPU performance; parameters that the software monitors are: user, system, total busy, wait and idle times, context switches, interrupts, and so on
Memory	Memory available, page scan rate, and total swap space
Disk	Number of active disks, projected fastest service time, number of bytes written, and average disk queue length
Network	Packet load and the number of packet errors that are occurring on the network interfaces of the server (overflow, underflow, CRC, frame, output, and input errors, as well as collisions)

Log Viewer

The Log Viewer displays entries from the log files, for example, `/var/adm/messages`. It searches the log using a list of regular expressions. When a match is found, the corresponding log file entry is sent to the Log Viewer. The Log Viewer can also search the SunVTS error log `/var/opt/SUNWvts/logs/sunvts.err`, if desired.

Process Viewer

The Process Viewer gathers and displays information about resource-intensive processes. You can configure the columns of data.

Monitoring Alarms and Events

Events are system conditions that require the attention of a system administrator. Alarms are used to alert the system administrator of an event, which appears as a message in the Event Viewer and a visual highlight on a console.

Event Generator

As the name implies, the Event Generator monitors the status and performance of the server. The software on the Server Subsystem communicates to the Event Generator (monitoring machine) by the Event Generator polling the server. The Event Generator detects any condition that violates the predefined performance parameters that are specified in the set of event rules.

If the Event Generator detects problems with the system, the affected component can be highlighted in red, yellow, (depending on the event severity) or blue (for capacity-related events). All related components are also highlighted. For example, if a CPU chip is highlighted, then the board on which it is mounted and its chassis slot are also highlighted.

For more information on the Event Generator and the Event Manager, see "Event Manager" earlier in this chapter and **Chapter 4, "Understanding and Writing Event Rules**."

Event Viewer

The console used to monitor events is the Event Viewer. The Event Viewer alerts you to any problems detected by the Event Manager. Each entry displayed in the Event Viewer window represents an event and gives you the following information:

- The rule number that generated the event
- The severity level (red, yellow, or blue)
- The time and date the event was detected
- A message indicating the type of event that was captured
- The node where the event occurred
- The event priority and severity

Interaction with SunReMon

SunReMon™ and the Remote Systems Monitoring service are part of the SunSpectrum℠ program offering from SunService℠. The Remote Systems Monitoring service uses the SunReMon software to collect key system data from critical servers. This data is automatically transferred to the local Sun Solution Center on a regular basis. When you report a problem, the support engineer has this data available to assist in problem analysis.

SunReMon also has the ability to generate a "call-home" email alert to the local Sun Solution Center when a critical problem is detected. This requires an Internet email connection at your site.

Solstice SyMON also has the ability to generate a “call-home” alert. Due to support requirements, the default Solstice SyMON rules do not use the “call-home” feature of the Remote Systems Monitoring service. For information on configuring the Remote Systems Monitoring service, contact your local Sun Solution Center.

The Remote Systems Monitoring service is available to SunSpectrum[SM] Gold[SM] and SunSpectrum[SM] Platinum[SM] contract customers throughout the world. For local availability, contact your SunService sales representative

CHAPTER 2

Installing Solstice SyMON

Solstice SyMON was first supported on the Solaris 2.5.1 software environment. You must be running at least the Solaris 2.5.1 software environment in order to use Solstice SyMON.

Installation Requirements

TABLE 2-1 lists the required disk space for each of the packages and the system on which the packages should be installed.

TABLE 2-1 Solstice SyMON Packages

Package	Package Name	Description	Install On	Disk Space
Server Subsystem	SUNWsys SUNWsyrt	Server subsystem package Runtime package	Monitored server	~ 1.15 Mbytes ~ 0.30 Mbytes
Event Manager	SUNWsye SUNWsyrt	Event Generator package Runtime package	Any SPARC system other than the monitored server	~ 1.26 Mbytes ~ 0.30 Mbytes
GUI	SUNWsyu SUNWsyua SUNWsyub SUNWsyuc SUNWsyud SUNWsyrt	GUI subsystem package Images for Ultra Enterprise 3000, 4000, 5000, and 6000 servers Images for the SPARCserver 1000(E) and SPARCcenter 2000(E) Images for the Ultra Enterprise 2 and Ultra Enterprise 150 Images for the Ultra Enterprise 450 Runtime package	Desktop SPARC system	~ 4.06 Mbytes ~ 9.81 Mbytes ~1.65 Mbytes ~ 4.57 Mbytes ~ 3.00 Mbytes
man Pages	SUNWsym	Man pages	Any SPARC system	~ 0.03 Mbytes
SunVTS Online Diagnostics	SUNWsyc SUNWodu SUNWvts	Standalone configuration reader; prerequisite for SunVTS Online Diagnostics SunVTS Online Diagnostic utility SunVTS kernel GUI and tests	Monitored server Monitored server Monitored server	~ 0.29 Mbytes ~ 1.6 Mbytes ~ 23.0 Mbytes

Note – The SUNWsyrt package is required on the Server Subsystem, Event Manager, and GUI configurations.

Only one instance of the Server Subsystem and the Event Manager operate on a given server. However, you can have many instances of the GUI subsystem on different systems to monitor one server.

Installing the Software

Installing Solstice SyMON requires these steps:

1. Installing the CDE packages from the Solaris Desktop CD
2. Mounting the Updates CD
3. Saving any user-customized Solstice SyMON event rules and removing Solstice SyMON 1.0 or 1.1 and all previously installed packages (if upgrading from a previous release)
4. Installing the packages
5. Setting up your environment
6. Configuring the software
7. Starting agents

Installing CDE from the Solaris Desktop CD

In order to have Solstice SyMON online help available (this help is CDE-based online help), you must install the following in this order:

1. Solaris CD
2. Solaris Desktop CD (This CD contains the CDE packages. One of these packages, `SUNWdtbas`, is necessary in order to use Solstice SyMON online help)
3. Install Solstice SyMON from the Updates CD (see the next section)

Installing Solstice SyMON from the Updates CD

The Solstice SyMON software is located on the SMCC Updates CD. You need to mount the SMCC Updates CD so you can retrieve the Solstice SyMON software from the SMCC Updates CD.

Mounting the CD

Follow the procedure “To Mount the Locally Connected CD-ROM” or “To Mount the Remotely Connected CD-ROM.”

▼ To Mount the Locally Connected CD-ROM

1. **Become superuser and create the** `/cdrom` **directory (if one doesn't exist).**
2. **Place the SMCC Updates CD in the CD-ROM drive and close the drive door.**

   If Volume Manager is running, the CD is mounted and the path is: `/cdrom/upd_sol_2_6_smcc/Product`.

   If Volume Manager is *not running* complete Step 3, mounting the `/cdrom` directory manually.
3. **Mount the CD manually by typing:**

```
# mount -r -F hsfs /dev/dsk/c0t6d0s0 /cdrom
```

▼ To Mount the Remotely Connected CD-ROM

1. **Export the** `/cdrom` **directory from the server:**

   a. **Edit the** `/etc/dfs/dfstab` **file by adding the following line at the** *end* **of the** `/etc/dfs/dfstab` **file:**

```
share  -F nfs  -o ro /cdrom/supp_sol_2_6_smcc/Product
```

   b. **Start NFS and the mount daemon:**

```
# /usr/lib/nfs/nfsd -a 16
# /usr/lib/nfs/mountd
```

   c. **Export the** `/cdrom` **file system:**

```
# shareall
```

2. **Mount the** `/cdrom` **directory from the server.**
   - **If** `automountd` **is running, type**:

```
# cd /net/server_ hostname/cdrom/supp_sol_2_6_smcc/Product
```

■ **If** `automountd` **is** *not* **running, type**:

```
# mkdir /cdrom
# mount server_ hostname:/cdrom /cdrom
# cd /cdrom/supp_sol_2_6_smcc/Product
```

Installing Packages

Note – The packages you install for the Server Subsystem, Event Manager, and GUI subsystem are dependent on `SUNWsyrt`, the SyMON Runtime Library package, and `SUNWdtbase`, the CDE-base package. Ignore warning messages if you do not install the `SUNWsyrt` package first.

Updating Solstice SyMON from a Previous Release

Before you can install the Solstice SyMON packages, you must completely remove the currently installed Solstice SyMON configuration and remove the previously installed packages.

▼ To Save Customized Rules Files and Remove Solstice SyMON

1. **Save any customized rules files.**

   There were syntax and pattern matching changes to rules between Solstice SyMON 1.1 and Solstice SyMON 1.3. Some rule files from Solstice SyMON 1.1 and Solstice SyMON 1.3 SyMON are incompatible.

2. **Completely remove the currently installed Solstice SyMON configuration by typing as superuser:**

   ```
   # /opt/SUNWsymon/sbin/sm_confsymon -D
   ```

3. **Install Solstice SyMON by following the procedures that follow.**

   The first step in each procedure describes how to remove the existing packages.

Identifying Previously Installed Packages with pkginfo

Before removing all previously installed packages, you must identify which previously installed packages are installed on each system with the pkginfo command. For example:

```
# pkginfo SUNWsyrt
```

If the SUNWsyrt package is installed, the following is displayed:

```
application scripts SUNWsyrt Solstice SyMON Runtime library and Tcl
```

If the SUNWsyrt package is not installed, the following is displayed:

```
ERROR: information for “SUNWsyrt” was not found
```

Use pkginfo to identify the remaining previously installed packages.

▼ To Install the Server Packages

1. **If you are upgrading from a previous release, remove the existing** SUNWsys **and** SUNWsyrt **packages on the monitored server by typing the following as superuser:**

   ```
   # /usr/sbin/pkgrm SUNWsys SUNWsyrt
   ```

   Answer yes to all questions.

2. **Go to the installation directory on the monitored server:**

   ```
   # cd dirname
   ```

   where:

   dirname is the directory where the Solstice SyMON packages are located

3. **Add the** SUNWsyrt **and** SUNWsys **packages:**

   ```
   # /usr/sbin/pkgadd -d source_directory SUNWsyrt SUNWsys
   ```

where:

source_directory is the full path name of the directory where the packages you want to add are located. You can also use `'pwd'` for the *source_directory*.

▼ To Install SunVTS for Online Diagnostics

- **Install the packages necessary for SunVTS Online Diagnostics on the monitored system by typing:**

```
# /usr/sbin/pkgadd -d source_directory SUNWsyc SUNWvts SUNWodu
```

where:

source_directory is the full path name of the directory where the packages you want to add are located. You can also use `'pwd'` for the *source_directory*.

▼ To Install the Event Generator Packages

Note – If you are installing both the Event Generator and the User GUI subsystem packages on the same system, install `SUNWsyrt` only one time.

---

1. **If you are upgrading from a previous release, remove the existing Solstice SyMON Event Generator software from your desktop system (or on any other system other than the server) by typing:**

```
# /usr/sbin/pkgrm SUNWsye SUNWsyrt
```

2. **Install the Event Generator packages on your desktop system (or on any other system other than the server) by typing:**

```
# /usr/sbin/pkgadd -d source_directory SUNWsyrt SUNWsye
```

where:

source_directory is the full path name of the directory where the packages you want to add are located.

▼ To Install the GUI Subsystem Packages

Note – If you are installing the Event Generator and User GUI Subsystem packages on the same system, *do not* remove `SUNWsyrt` (see Step 1, which follows) since you previously installed `SUNWsyrt` in Step 2 of the previous procedure.

---

1. **If you are upgrading from a previous release, remove the existing** `SUNWsyu` **and** `SUNWsyrt` **packages from the desktop system by typing:**

```
# /usr/sbin/pkgrm SUNWsyu SUNWsyrt
```

2. **Install the GUI subsystem packages on the desktop system by typing:**

```
# /usr/sbin/pkgadd -d source_directory SUNWsyrt SUNWsyu [SUNWsyua
SUNWsyub SUNWsyuc SUNWsyud]
```

where:

- *source_directory* is the full directory path name where the packages are located
- `SUNWsyua` contains images for the Ultra Enterprise 3000, 4000, 5000, and 6000 systems
- `SUNWsyub` contains the images for the SPARCserver 1000(E) and SPARCcenter 2000(E)
- `SUNWsyuc` contains the images for the Ultra Enterprise 2 and 150
- `SUNWsyud` contains the images for the Ultra Enterprise 450

Installing the man Pages

You can install the man pages in two ways :

- Without using the Windex database for man pages
- Using the Windex database for man pages

Review the following information before deciding which method to use.

The Windex database is generated with the `catman` command. The Windex man page database is used by the `man -k` and `apropos` commands. For example, if you need to know every man page that mentions disk, you would type `man -k disk`. You can also invoke the `apropos` command, by typing `apropos logout`, which returns man pages related to `logout`.

Use `pkgadd` to install the Windex database of man pages. The `catman` command indexes the Windex database.

▼ To Install the Man Pages

- If you do not have a Windex database for man pages:

1. **If you are upgrading from a previous release, remove the existing `SUNWsym` package by typing:**

```
# /usr/sbin/pkgrm SUNWsym
```

2. **Add the new man page package by typing:**

```
# /usr/sbin/pkgadd -d source_directory SUNWsym
```

where:

source_directory is the full path name of the directory where the packages you want to add are located.

- If you have a Windex database for the man page:

1. **If you are upgrading from a previous release, remove the existing `SUNWsym` packages and the Windex database by typing:**

```
# /usr/sbin/pkgrm SUNWsym
# rm /opt/SUNWsymon/man/Windex
```

2. **Add the new man page package by typing:**

```
# /usr/sbin/pkgadd -d source_directory SUNWsym
```

where:

source_directory is the full path name of the directory where the packages are located that you want to add.

3. **Create the new Windex database by typing:**

```
# catman -w M /opt/SUNWsymon/man
```

Setting Up Your Environment

You can set up your environment for the GUI using the Bourne, Korn, or C shells as follows.

▼ To Set Up the Environment for the GUI

- For Bourne and Korn shells

1. **Add the following lines to the `.profile` file:**

```
SYMONHOME=/opt/SUNWsymon
PATH=$SYMONHOME/bin:/opt/SUNWvts/bin:$PATH
MANPATH=$SYMONHOME/man:$MANPATH
LD_LIBRARY_PATH=$SYMONHOME/lib:$LD_LIBRARY_PATH
export SYMONHOME
export PATH
export MANPATH
export LD_LIBRARY_PATH
```

Note – If the software is not installed in `/opt/SUNWsymon`, replace the `/opt/SUNWsymon` path with the path you used to install the software in the line beginning with `SYMONHOME`.

---

2. **Make the previous setup parameters effective by logging out and then logging in.**

- For the C Shell

1. **Add the following lines to the `.cshrc` file:**

```
setenv SYMONHOME /opt/SUNWsymon
set path=$SYMONHOME/bin /opt/SUNWvts/bin $path
setenv MANPATH $SYMONHOME/man:$MANPATH
setenv LD_LIBRARY_PATH $SYMONHOME/lib:$LD_LIBRARY_PATH
```

Note – If the software is not installed in `/opt/SUNWsymon`, replace the `/opt/SUNWsymon` path with the path you used to install the software in the line beginning with `setenv`.

---

2. **Make the previous setup parameters effective by typing:**

```
# source .cshrc
# rehash
```

Configuring Solstice SyMON

Before you can run Solstice SyMON in your environment, you must configure the host and the Event Generator host (Event Host). For example, if you have two systems, server A and a desktop system B, and you want desktop system B to monitor server A, a diagram of the monitoring configuration would look similar to FIGURE 2-1.

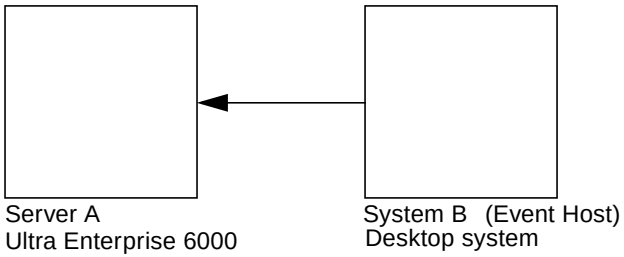

FIGURE 2-1 Diagram of Desktop System B Monitoring Server A

- **To configure the host, type the following on server A:**

```
# /opt/SUNWsymon/sbin/sm_confsymon -s EventMonitor_host
```

where:

EventMonitor_host is the host that monitors this server, which is Server A in this example.

- **To configure the Event Generator, invoke the** `uname -i` **command on server A.**

This command returns the platform name of the system. Use the platform name (for example `SUNW,Ultra-Enterprise`) as an argument to the `sm_confsymon -e` command.

Note – If the `uname -i` command returns `SUNW,Ultra-1,` this platform name is for the Ultra Enterprise 150, which is supported by Solstice SyMON. The same platform name is used for the Ultra Enterprise 1 system, which is not supported by Solstice SyMON.

Also, the `uname -i` command returns `SUNW,Ultra 2` for both the Ultra Enterprise 2 server, which is supported by Solstice SyMON and the Ultra Enterprise 2 workstation, which is not supported by Solstice SyMON.

3. **Type the following on the Event Host:**

```
# /opt/SUNWsymon/sbin/sm_confsymon -e agent_host -P platformname
```

where:

- *agent_host* is the host name of the server running the Server Subsystem, which is Server A in this example.
- *platformname* (`SUNW,Ultra-Enterprise` in this example) is the platform type returned by the `uname -i` command.

Starting Solstice SyMON

This section describes how to start running Solstice SyMON. The Server Subsystem and the Event Manager start automatically after you reboot your system. To start the server subsystem without rebooting, follow the next procedure, "To Start Solstice SyMON without Rebooting."

▼ To Start Solstice SyMON *without* Rebooting

Note – You can also use the `-v` flag (verbose installation) with the `sm_confsymon` command. The system echoes all actions and requests permission to continue when you use the `-v` flag.

1. **Start Solstice SyMON on the Server Subsystem host by typing:**

```
# /opt/SUNWsymon/sbin/sm_control start
```

2. **Start the Event Generator on the Event Generator machine by typing:**

```
# /opt/SUNWsymon/sbin/sm_control start
```

3. **Run the GUI from your desktop as a regular user by typing:**

```
CTRL-d (to become a regular user)
% rehash
% /opt/SUNWsymon/bin/symon -t target_system &
```

where:

target_system is the name of the server running the monitor agents.

Solstice SyMON also recognizes command line options (see **Chapter 5, "Command Line Options**)." When you invoke the software at the shell prompt, you can use these options on the command line. For example, in the previous screen, the `-t` option refers to the target system to monitor.

Exiting the GUI

This section describes how to exit the GUI. Exiting the GUI shuts down the GUI but does not shut down the software application, which continues to monitor the server.

▼ To Exit the GUI

The Launcher window contains a File menu that includes an Exit button.

- **Close the current window and all other windows by clicking the "yes" button to the exit inquiry.**

  If you need to view the information that the software has collected about the server, restart the GUI.

Shutting Down Solstice SyMON

Note – Normally, the agents and the Event Generator should run continuously, even if GUIs are not attached. Running them continuously monitors the server for events and gathers historical data.

- **To shut down Solstice SyMON, type the following command as superuser on the server or the Event Host:**

```
# /opt/SUNWsymon/sbin/sm_control stop
```

Removing Solstice SyMON

Caution – `SUNWsyrt` is the runtime library. Remove `SUNWsyrt` only if all the SyMON packages are removed from that machine.

▼ To Remove Solstice SyMON

Caution – If the Event Generator is also installed on this machine and you want to continue running the Event Generator on this machine, *do not* remove the `SUNWsyrt` package.

TABLE 2-2 Definition of Solstice SyMON Packages

If you remove	The following is removed
`SUNWsye` package	Event Generator binaries and data
`SUNWsyrt` package	Runtime library
`SUNWsys` package	Server binaries and executables

TABLE 2-2 Definition of Solstice SyMON Packages *(Continued)*

If you remove	The following is removed
SUNWsyu package	GUI binaries and executable
SUNWsyua package	Images for the Ultra™ Enterprise™ 3000, 4000, 5000, and 6000
SUNWsyub package	Images for the SPARCserver™ 1000/E and SPARCcenter™ 2000/E
SUNWsyuc package	Images for the Ultra Enterprise 2 and 150
SUNWsyud package	Images for the Ultra Enterprise 450 system
SUNWsyc package	Standalone configuration reader (prerequisite for SunVTS)
SUNWvts package	SunVTS kernel, GUI, and tests
SUNWodu package	SunVTS Online Diagnostic utility

1. **Remove the Server Subsystem, runtime library, and SunVTS packages on the Server Subsystem host by typing:**

```
# /usr/sbin/pkgrm SUNWsys SUNWsyrt SUNWsyc SUNWodu SUNWvts
```

2. **Remove the Event Manager and runtime library on a desktop system or any system except for the server host by typing:**

```
# /usr/sbin/pkgrm SUNWsye SUNWsyrt
```

3. **Remove the GUI and runtime library on the desktop system by typing:**

```
# /usr/sbin/pkgrm SUNWsyu [SUNWsyua SUNWsyub SUNWsyuc SUNWsyud]
SUNWsyrt
```

4. **Remove the man pages.**
   - **If you** *do not* **have the Windex database of man pages, type**:

```
# /usr/sbin/pkgrm SUNWsym
```

- **If you have the Windex database of man pages, type:**

```
# /usr/sbin/pkgrm SUNWsym
# rm /opt/SUNWsymon/man/Windex
```

Troubleshooting

If you have problems installing Solstice SyMON, see **Chapter 6, "Troubleshooting**."

Installing and Setting Up SNMP Traps

To install and set up SNMP traps, see Appendix C, "Installing and Setting Up SNMP Traps." A SNMP trap is an asynchronous message directed at a specific monitor.

CHAPTER 3

Using the Solstice SyMON Consoles

This chapter describes how to invoke and use the Solstice SyMON GUI and provides information on the menus and associated subwindows.

Invoking the Solstice SyMON GUI

This section describes how to launch a Solstice SyMON console.

▼ To Invoke the Solstice SyMON GUI

1. **Type the following as a regular UNIX user:**

```
% /opt/SUNWsymon/bin/symon -t monitored_servername &
```

where:

monitored_servername is the name of the server you want to monitor.

After a few moments, the main window, the Launcher, is displayed (see FIGURE 3-1). The Launcher establishes communication between the GUI, the agents, and the Event Generator and launches different consoles.

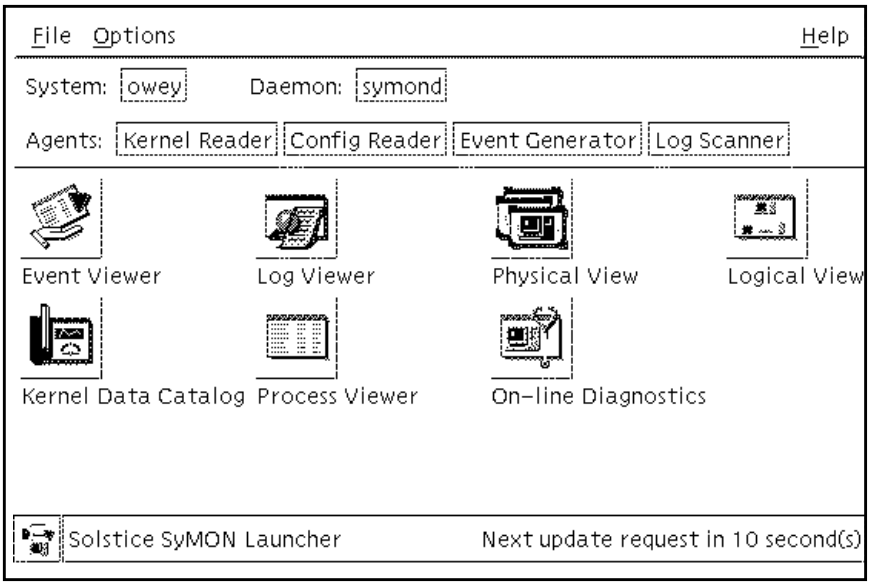

FIGURE 3-1 Launcher Window

2. **To launch a console, click the icon.**

Launcher Window

FIGURE 3-1 shows that the software is installed and is running on the server being monitored (owey). TABLE 3-1 describes the meanings of the colors you might see for the System: box in the upper left-hand corner of the Launcher window.

TABLE 3-1 The System Box on the Launcher Window

System Box Colors	Meaning
Blinking green	Target system is working; Solstice SyMON is operating normally
Red	Target system is not running

Below the System: box, the next line shows the status of the agents. The Launcher polls the agents and various log files and displays the information in the appropriate consoles.

The lower section of the Launcher shows consoles (icons) for viewing information collected by other agents. The Launcher provides six consoles that monitor server conditions:

- Event Viewer
- Log Viewer
- Physical View
- Logical View
- Kernel Data Catalog
- Process Viewer

The seventh console, Online Diagnostics, connects you directly to the Online SunVTS diagnostics. If the console is unavailable, a slashed circle is displayed (∅).

Any console with an associated event is highlighted in color. The colors (only red, yellow, and blue) are programmable using event rules. Note that TABLE 3-2 lists the meanings of the colors.

TABLE 3-2 Meanings of Colors Used

Color	Indications
Red	Serious condition detected by the Event Generator
Yellow	Warning condition
Blue	Long term capacity-related condition

Monitoring Hardware Status

You use the following consoles, described in this section, to monitor hardware status:

- Physical View, which displays detailed information about the server
- Logical View, which displays the components in a schematic hierarchy
- Online Diagnostics, which provides a launcher to SunVTS

Using the Physical View Console

The Physical View is a graphic (pictorial) representation of the server. Use this console to monitor the configuration and status of your server. If the Event Generator detects problems with a system component, the affected component is highlighted in red, yellow, or blue (see TABLE 3-2).

If you are monitoring a multi-sided system such as the Ultra Enterprise 6000, when you launch the Physical View, a submenu with three viewing options is displayed:

- Front view
- Rear view
- Both views

The Physical View shows detailed information about the status of individual components in the server. With this option you know exactly what hardware is installed on the server and if there is problem with a component on the server (the component is highlighted). More information on the Physical View is available through the Show Information window (see "Using the Show Information Window" later in this chapter.)

You can switch to an alternate view from any view in the Physical View by using the File menu. For example, if the system you are viewing has Front, Rear, and Side views and you selected Front view, from the Front view you can select the other two views using the File menu.

The Physical View also provides a way to access components that don't have an image in the current view. For example, disk enclosures don't have an image in the I/O card. You can access these disk enclosures by clicking on a connector to these disk enclosures. This displays a menu, which lists all possible viewpoints from all disk enclosures.

Displaying a Detailed View of a Component

When you view a system component, you can also view its internal components. For example, when you view a CPU/Memory board on the Ultra Enterprise 6000 system, you can also view a more detailed image that includes the memory and UltraSPARC modules on the board. When a more detailed sub-view is available, the Physical View cursor changes to a pointing hand. If you don't see the pointing hand, a more detailed view is not available.

- **To display a detailed picture of a component, click the component with the pointing hand cursor.**

  Another Physical View window is displayed with a detailed picture of the component. FIGURE 3-2 shows the detailed picture of the CPU/Memory board in the Ultra Enterprise 6000 system.

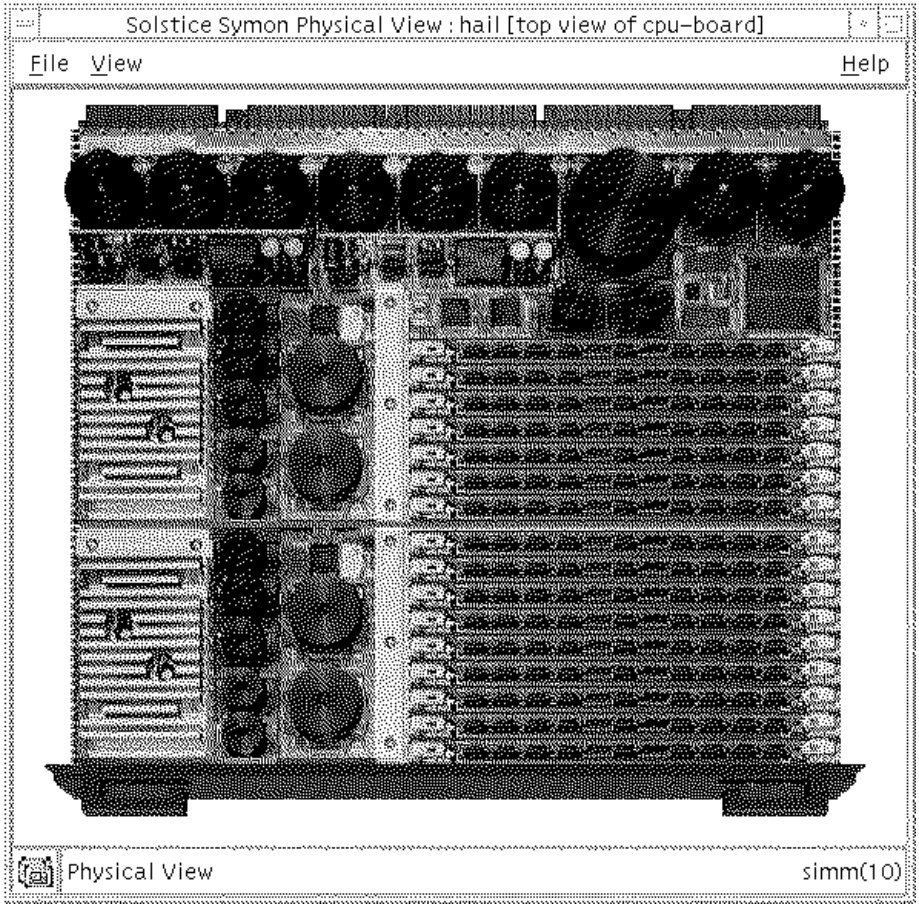

FIGURE 3-2 Expanded Physical View of a CPU/Memory Board

Limitations of the Physical View

The SPARCserver 1000 and SPARCcenter 2000 system can contain a disk board. However, Solstice SyMON does not sense the disk board because the hardware does not provide a way to sense the disk board. The Physical View does not show the disk board but shows an empty slot. Also, the SPARCserver 1000 and SPARCcenter 2000 do not provide a way to sense fiber optic modules. Their existence is inferred if a SPARCstorage Array is connected to the server.

Using the Show Information Window

The View menu on the Physical View and any sub-views of individual components include a Show Information window, which views more specific information about each system component. The information is continuously updated with information about the component the cursor is over as the cursor moves on the screen.

▼ To Display Component Specific Information in the Physical View

1. **Select Show Information from the View menu.**

   An Information window is created.

2. **Move the pointer to a component displayed on the Physical View.**

   Information about the component is displayed in the Information window (see FIGURE 3-3). For example, if you place the pointer over a CPU, critical information about the CPU is displayed, including: CPU type, status, cache-size, board number, port ID, processor ID, and model number.

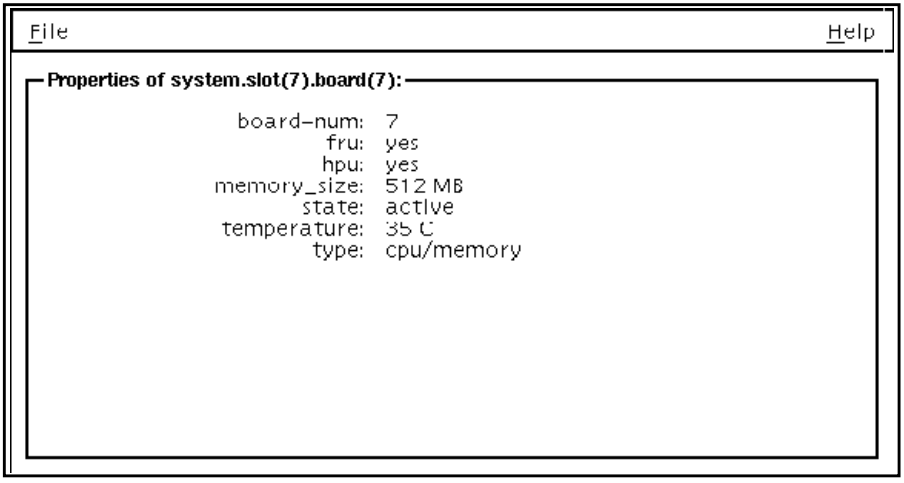

FIGURE 3-3 Show Information Window

Using the Copy Information Window

In addition to viewing the status of a component with the Show Information window, you may also need a static screen of this information. To do this, use the Copy Information window. It presents the same information as the Show Information window, except that the information in the Copy Information window relates only to the component you selected and is static.

▼ To Select the Copy Information Window

1. **Move the mouse on a component in the Physical View or node in the Logical View.**
2. **Hold down the Menu button of the mouse, which is the mouse button furthest away from the keyboard.**

   This is the right mouse button for a right-handed person and the left mouse button for a left-handed person.
3. **Select the copy information dialog box.**

   A Copy Information window is displayed showing information about the component or node.

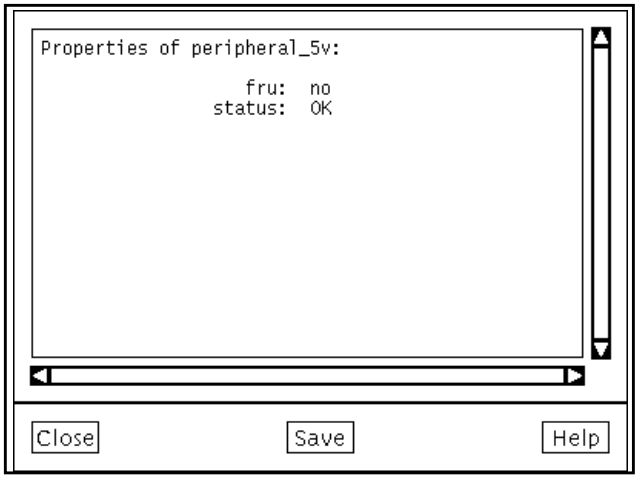

FIGURE 3-4 Copy Information Window Showing the FRU Status of the `peripheral_5v` Node in the Logical View

Using the Close Options

This section describes how to close a Physical View.

▼ To Close a Physical View

- **Select one of the following options from the File menu**:
  - **Close Self**, which only closes the window
  - **Close All Views of This Component and Descendants**, which closes the window and more detailed views of the component
  - **Close All Physical View**, which closes all Physical View windows

Locating Troublesome Components

If a component has problems or fails, it is highlighted in a color (see TABLE 3-2). If the component is not highlighted, the component is operating normally.

▼ To Quickly Locate Components that May Cause Problems

- **Look at the Physical View and trace events that are highlighted from the top level of the hierarchy down to the component that generated the event.**

Using the Logical View

The Logical View, a companion view of the Physical View, displays the components in a schematic hierarchy (see FIGURE 3-5) instead of showing pictures of the front or rear views. Like the Physical View, the Logical View shows detailed information about the status of many components in the server. If a component is responsible for generating an event, it and all higher level components are highlighted in a color (see TABLE 3-2), so you can quickly locate the component causing problems.

The bottom of the window contains instruments that may be opened. To display any instruments for a particular component, select (click the left mouse button) in the text of the component. In the Logical View, there are currently only board temperature instruments that are displayed by selecting the component of the SysMeter under the appropriate board.

▼ To Display a System Meter of an Instrument for a Specific Parameter

1. **Expand the hierarchy by clicking the + sign.**
2. **Click the component label (such as `disk`) to display available instruments in the instrument panel.**
3. **Display the instrument using one of the following tasks:**
   - Click the instrument (parameter) you want displayed, for example, ops `(op/sec)`, at the right of the display. A System Meter window is created with a graph of the selected parameter (FIGURE 3-19).
   - Drag the instrument and drop it in an existing System Meter**.** Use the middle or left mouse buttons as you move the mouse.

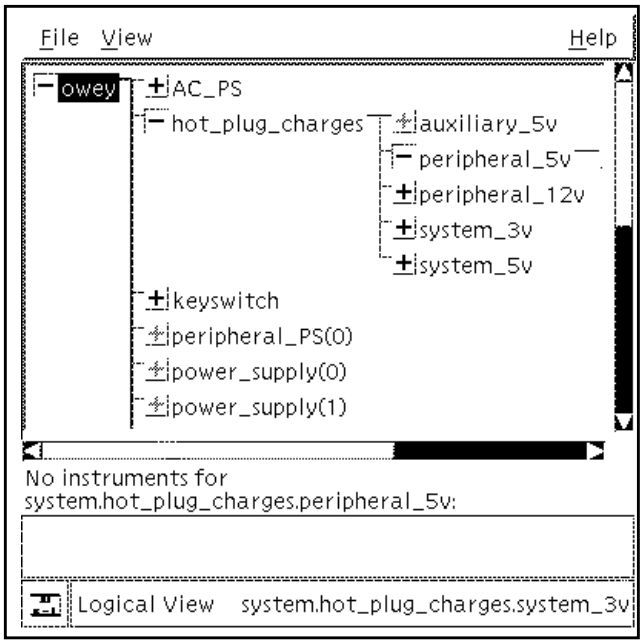

FIGURE 3-5 The Logical View

Expanding and Collapsing the Hierarchical Diagram

A button to the left of each component in the hierarchy is displayed (see FIGURE 3-5). The two states of the button are:

- Collapsed (+), which contracts the entire hierarchy below the point you are at. The AC_PS component in FIGURE 3-5, noted by a + sign, shows a collapsed display. The hierarchy below the AC_PS component is *not* displayed.
- Expanded (-), which displays the hierarchy below the point you are at one level at a time. The server `owey` in FIGURE 3-5, noted by a - sign, shows the expanded display to the right.

You cannot mix Logical View instruments with Kernel Data Catalog instruments in the same System Meter.

- **To expand the hierarchy, click the + button one level.**

  The hierarchy below that point is expanded and is noted by a - sign. To view the entire hierarchy use the scroll bars.

- **To hide the contents of the hierarchy, click the - button.**

  The button changes to a + sign and the contents of the lower level of the hierarchy are hidden.

Limitations of the Logical View

The SPARCserver 1000 and SPARCcenter 2000 systems can contain a disk board. However, Solstice SyMON does not sense the disk board because the hardware does not provide a way to sense the disk board. The Logical View shows the disks that are located in the disk board. However, it does not show them as children of the disk board. Instead, it shows them as children nodes of an SBus card, which is a child node of one of the system boards. Also, the SPARCserver 1000 and SPARCcenter 2000 do not provide a way to sense fiber optic modules. Their existence is inferred if a SPARCstorage Array is connected to the server.

Using the Show Information Window

The View menu on the Logical View includes a Show Information window that is identical to the Physical View.

Note – The Show Information window is shared between the Physical View and the Logical View. If you launched the Show Information Window from the Physical View, you do not need to re-launch it from the Logical View (or vice-versa).

▼ To Display Component-Specific Information in the Logical View

1. **Select Show Information from the View menu.**
2. **Move the pointer to a component displayed on the Logical View or Process Viewer.**

   Information about the component is displayed in the Show Information window.

Using the Copy Information Window

You can use the Show Information window to obtain a static screen of the same information presented in the Show Information window. The information in the Copy Information window relates only to the selected component.

For a step-by-step procedure on how to use the Copy Information window, see “To Select the Copy Information Window” earlier in this chapter. Using the Copy Information window works the same way for the Physical and Logical Views.

Using the Online Diagnostics Console

The Online Diagnostics icon launches the SunVTS Online GUI. This GUI is used to perform online diagnostics and troubleshoot components (see FIGURE 3-6).

SunVTS is a suite of diagnostics that test and validate the configuration and functionality of most hardware devices of a server. It is a diagnostic tool that handles sequential testing of system resources and internal and external peripheral equipment.

The SunVTS diagnostics have the following features:

- Online diagnostics tests are run one at a time
- Tests are safe to run in an online environment
- Information of how the test is progressing appears in the test message console as the tests execute

To schedule testing at a future time, either periodically or for one time, use the Periodic Testing feature under the Commands menu. When you click the Online Diagnostics console, the following screen is displayed (see FIGURE 3-6).

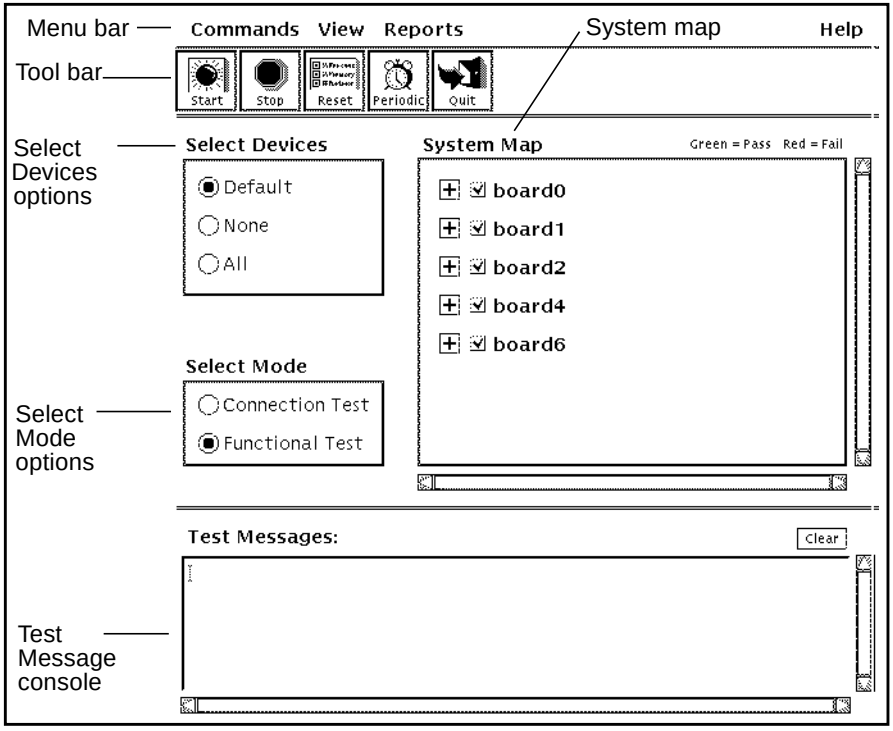

FIGURE 3-6 SunVTS Online Diagnostic Tool

TABLE 3-3 describes the features shown in FIGURE 3-6.

TABLE 3-3 SunVTS Components Online Diagnostic Tool Screen

Function	Description
Menu Bar	Includes these menus at the top of the GUI: • Commands • View • Reports • Help
Tool Bar	Provides a shortcut for executing commands in the Commands menu
System Map	Creates a hierarchy of devices that you can select for testing; during testing, the System Map provides testing status for each device by changing the color of the device name Green = pass and red = failure
Select Devices options	Contains buttons from which you can select or deselect a group of devices from the System Map on which to run tests.
Select Mode	Contains two options from which you can select the mode to run a test: • Connection Test mode—Quick check that each device is connected properly and ready to use • Functional Test mode—A more thorough functional check of the device in a non-destructive operation
Test Message Console	Displays test error messages

Commands Menu

The Commands menu (see FIGURE 3-7) contains several commands that you can use to control the testing of the target server system.

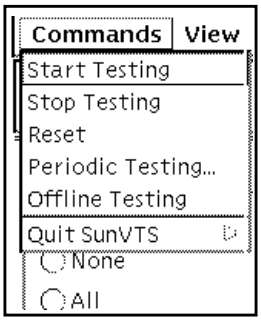

FIGURE 3-7 Commands Menu

The following selections are available from the Commands menu (see TABLE 3-4):

TABLE 3-4 Description of Commands in the Command Menu

Command	Description
Start Testing	Begins executing tests on each device (after all the devices are selected and the testing mode is set.) Test execution proceeds sequentially.
Stop Testing	Stops all device testing and returns SunVTS to an idle state.
Reset	Resets all device name colors in the System Map to black.
Periodic Testing	Schedules testing at a future time, either periodically or for one time (see “To Schedule Tests on Devices to Run at a Future Time”) later in this chapter.
Offline Testing	Terminates the SunVTS Online GUI and starts up the Offline GUI. Offline testing includes testing that may be destructive to the media (for example disk writes).

▼ To Run a SunVTS Test

1. **Select the devices you want tested:**
   - To run a predefined set of tests, select Default.
   - To run tests on all of the devices, select All.
   - To run a test on one device or just a few devices, select None. Select the device(s) you want tested by clicking the device name or checkbox (see FIGURE 3-6).

2. **Select the testing mode**:
   - Connection Test is a quick check that each device is connected properly and ready to use
   - Functional Test is a more thorough functional check of the device in a non-destructive operation
3. **Start the test by clicking the Start button or select Start Testing from the Commands menu.**

   As the tests are run on each device, an asterisk (*) appears next to the device in the System Map. The device name turns green if the tests pass; or red if the tests fail.
4. **To stop all testing and return SunVTS to an idle state, stop the test by clicking the Stop button or select Stop Testing from the Commands menu.**

Scheduling Tests

The Periodic Testing option (see FIGURE 3-7) schedules testing at a future time, either periodically or for one time. When creating a schedule, you can select tests and test options in the test selector window as described in the previous section "To Run a SunVTS Test." Alternatively, you can select a prepackaged test configuration, which includes such useful tests as a complete system connection check and a complete functional check of all devices

You can schedule tests to run in the following ways:

- At a certain time on a range of days, such as Monday through Sunday (periodic)
- At a certain time on a specific date (one time)

You can also create schedules to run after hours when the system use is low in order to validate that the components of the server are operating properly or are connected.

Schedule Menu

Options available under the Schedule menu are listed in TABLE 3-5.

TABLE 3-5 Schedule Menu Selections

Command	What it does
Create	Selecting Create lets you schedule a test. See the following procedure "To Schedule Tests on Devices to Run at a Future Time."
Open	Highlighting on the schedule you want to select with the left mouse button and selecting Open from the menu opens the filled in schedule window (see FIGURE 3-8). You can also double-click with the left mouse button on the schedule to open the window.
Delete	Highlighting the schedule you want to delete with the left mouse button and selecting Delete from the menu deletes the schedule (see FIGURE 3-8).
Delete All	Selecting Delete All deletes all schedules.

▼ To Schedule Tests on Devices to Run at a Future Time

1. **Select Periodic Testing under the Commands menu using the left mouse button or click the Periodic icon in the Tool Bar.**

   The following screen (see FIGURE 3-8) is displayed. If you have not set up schedules, a blank Schedule List window is displayed.

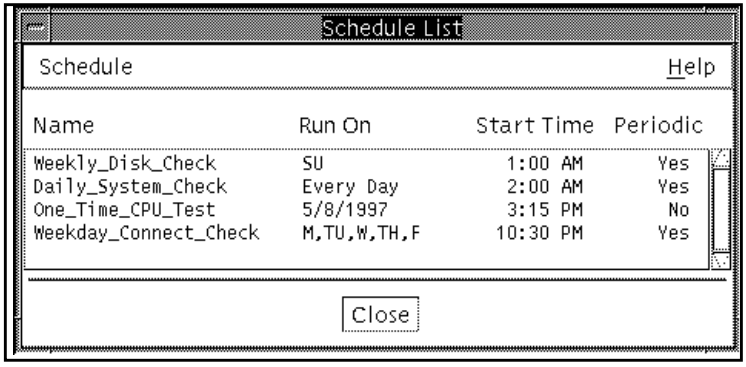

FIGURE 3-8 Schedule List of Periodic Testing

2. **Select the Create option under the Schedule menu using the left mouse button.**

   The following screen is displayed (see FIGURE 3-9).

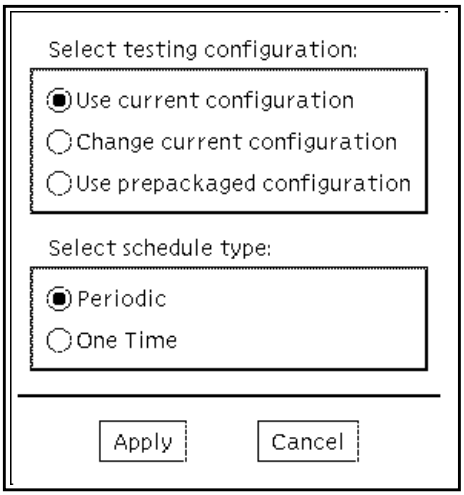

FIGURE 3-9 Create Option Under the Schedule Menu

3. **Select the schedule type (Periodic or One Time) on the bottom of the screen shown in** FIGURE 3-9**.**

4. **Select the testing configuration at the top of the screen.**
   - If you select “Use current configuration,” you will next see the screen for either testing type: Periodic (see FIGURE 3-11) or One Time (see FIGURE 3-12.)
   - If you select “Change current configuration” a pop up window is displayed prompting you to change the testing configuration in the main window. Click the OK button when you are done.
   - If you checked “Use prepackaged configuration,” a screen is displayed noting the types of checks that you can run (see FIGURE 3-10).

     a. Select one type of check.

     b. Click “More Information” to obtain more information on each type of check.

     c. Click Apply to make the selection.

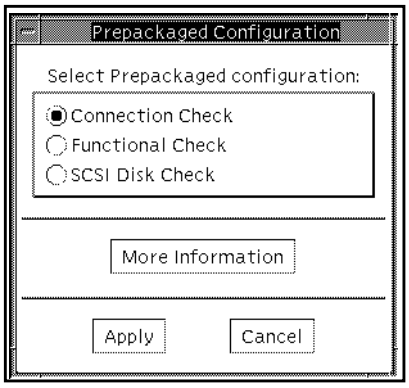

FIGURE 3-10 Selecting the type of Check to Run under Prepackaged Configuration

The schedule form appears:

- FIGURE 3-11 if you checked Periodic in FIGURE 3-9
- FIGURE 3-12 if you checked One Time in FIGURE 3-9

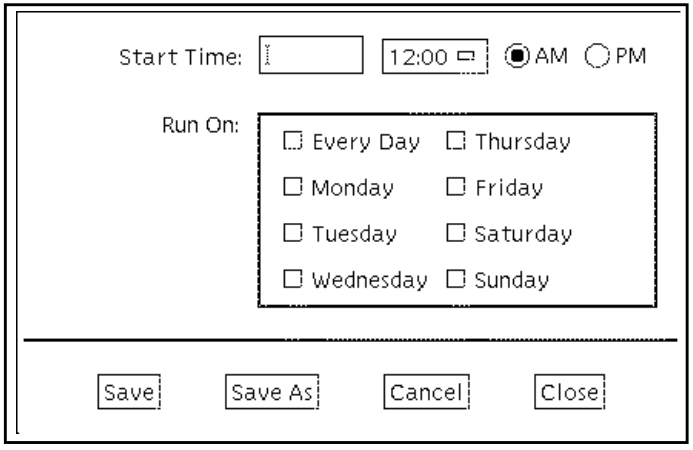

FIGURE 3-11 Periodic Schedule Form Display

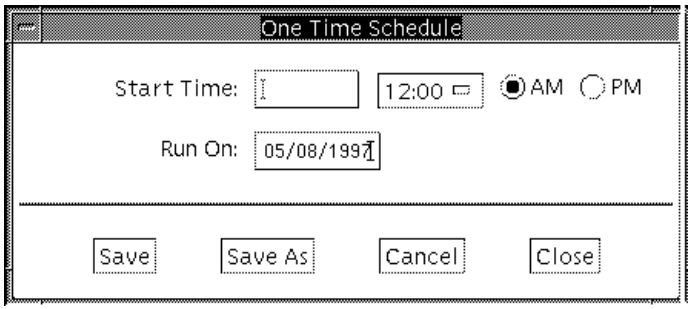

FIGURE 3-12 One Time Schedule Form Display

5. **Fill out the schedule then click on Save or Save As with the left mouse button (see** FIGURE 3-11 **or** FIGURE 3-12**).**

   A Save As screen appears (see FIGURE 3-13).

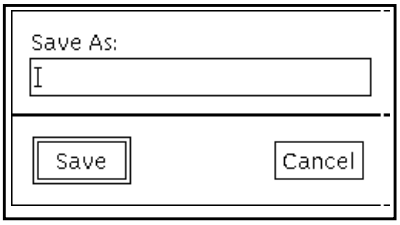

FIGURE 3-13 Save As Screen

6. **Type the schedule name (any name you choose without spaces) in the window and click Save with the left mouse button.**

   The schedule name is displayed in the Schedule List (see FIGURE 3-8). If you clicked on Save in Step 4, a unique schedule name is automatically generated. To check the results of a scheduled test, examine the information log.

   When a schedule is invoked, SunVTS loads the stored test configuration and begins testing.

   You view the results of the scheduled testing in the information log file by selecting Log files from the Reports menu (see also “Reports Menu“ later in this chapter.)

   If a test is scheduled to run periodically, the phrase `Periodic testing starting` appears along with the schedule name in the log file when the schedule is invoked.

   An example of running the schedule `midnight.processor.test` follows:

```
SUNWvts.ptexec.2000 05/29/97 00:00:00 ptexec midnight.processor.test INFO:
Periodic testing starting
```

View Menu

The View menu on the main SunVTS Online Diagnostics screen (see FIGURE 3-14) lets you expand or contract the system hierarchy (also known as the System Map in FIGURE 3-6) and has two options:

- Open System Map
- Close System Map

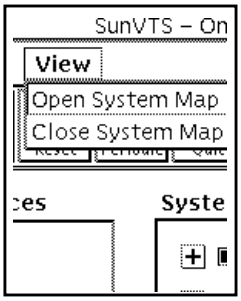

FIGURE 3-14 View Menu

▼ To Expand the System Map

- **Select Open System Map from the View menu.**

  The hierarchy of devices in the System Map is expanded to show all devices in the System Map.

▼ To Contract the System Map

- **Select Close System Map from the View menu.**

  The hierarchy of devices in the System Map is contracted. Only major devices are displayed.

Reports Menu

The Reports menu on the main SunVTS Online Diagnostics screen (see FIGURE 3-6 and FIGURE 3-15) enables you to view the system configuration or various log files.

FIGURE 3-15 Reports Menu

- **To display a listing of the hardware on the test system and their corresponding hardware tests (see** FIGURE 3-16**) select System configuration.**

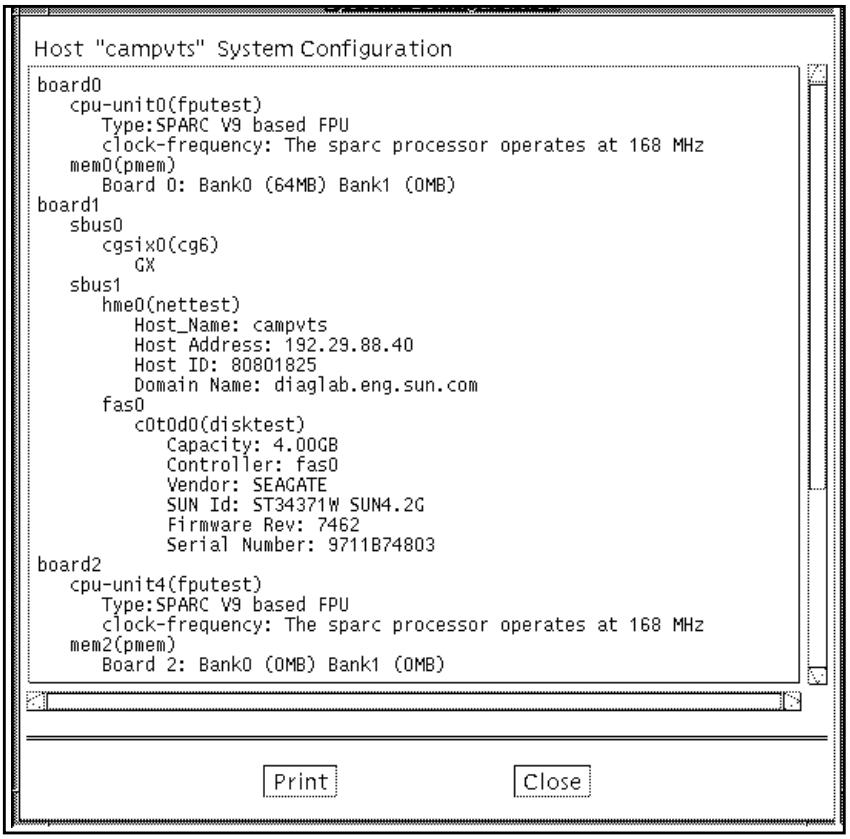

FIGURE 3-16 System Configuration

- **To view the SunVTS Error and Information log files as well as the** `/var/adm/messages` **file, which is the Solaris system message log, select Log files (see** FIGURE 3-17**).**

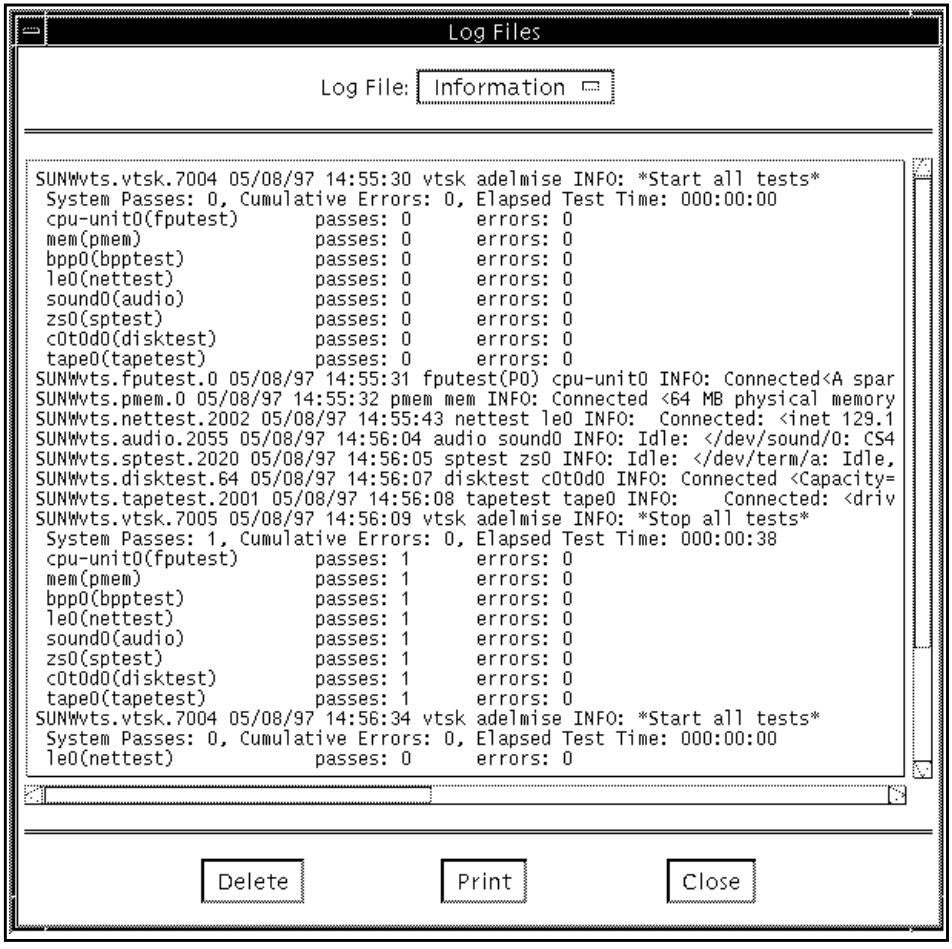

FIGURE 3-17 Log File Screen

Additional Information on SunVTS

For more information about SunVTS, refer to the *SunVTS 2.1 User's Guide* and the *SunVTS 2.1 Test Reference Manual* in the Solaris 2.6 on Sun Hardware AnswerBook on the Solaris 2.6 Supplement CD.

Monitoring System Performance

Solstice SyMON also monitors system performance and identifies potential bottlenecks. Solstice SyMON uses the Log Viewer and the Process Viewer consoles to monitor system performance. Solstice SyMON also uses System Meters to monitor performance. Using these two consoles, the System Meters and the Kernel Data Catalog, is explained in the sections that follow.

Using the Kernel Data Catalog

System Meters are graphical displays of system performance. You build System Meter displays using the Kernel Data Catalog.

- **To evaluate the extended performance of the server, create one or more System Meters using the Kernel Data Catalog.**

  System Meters can help to identify bottlenecks and anticipate potential capacity and hardware problems. To build System Meters, see "Building System Meter Windows" later in this chapter.

Kernel Data Catalog Instruments

The Kernel Data Catalog console displays the system performance parameters (see FIGURE 3-18). These parameters are displayed as *instruments* within the instrument panel of the Kernel Data Catalog. When you click a particular text of a component, a menu of available parameters (instruments) for that component is displayed in the instrument panel

Like the Logical View, you can choose whether the instruments associated with each hierarchy are displayed (-) or hidden (+). You can use instruments in the System Meters and the Kernel Data Catalog. However, you cannot mix Kernel Data Catalog instruments with Logical View instruments in the same System Meter.

For a complete listing of the Kernel Reader data hierarchy, see **Appendix A, "Kernel Reader**."

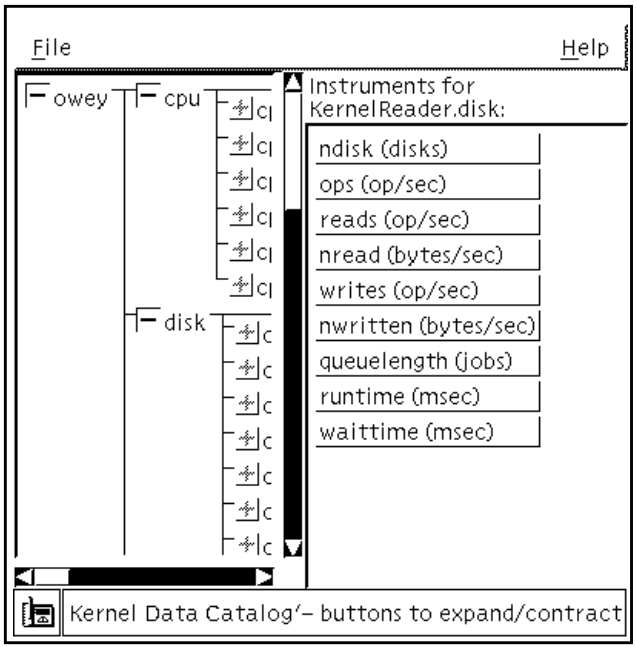

FIGURE 3-18 Kernel Data Catalog Console

When the status of a parameter results in an event, the Event Manager highlights the parameter *and* its ancestors on the Kernel Data Catalog.

Building System Meter Windows

The System Meter windows provide a graphical display of system performance parameters. Use the instruments displayed in the Kernel Data Catalog to:

- Create a graph for one parameter in a System Meter
- Combine multiple parameters on the same graph in a System Meter
- Combine multiple graphs in the same System Meter
- Locate multiple System Meters

For information on locating troublesome components, see "To Quickly Locate Components that May Cause Problems" earlier in this chapter. To expand or collapse the hierarchical diagram, see "Expanding and Collapsing the Hierarchical Diagram" earlier in this chapter.

▼ To Display a System Meter of an Instrument for a Specific Parameter

1. **Expand the hierarchy by clicking the + sign.**
2. **Click the component label (such as `disk`) to display available instruments in the instrument panel.**
3. **Display the instrument using one of the following tasks**:
   - Click the instrument (parameter) you want displayed, for example, ops `(op/sec)`, at the right of the display.

     A System Meter window is created with a graph of the selected parameter (FIGURE 3-19).
   - Drag the instrument and drop it in an existing System Meter**.** Use the middle mouse button or use the left mouse button as you move the mouse.

▼ To Display Multiple System Meters in the Same Window

1. **Click the System Meter window you want to activate.**
2. **Select the legend for the desired parameter (for example, `writes (ops/sec)`) and drag it to the footer of the second System Meter window (see** FIGURE 3-19**).**

   A second pane is added to the System Meter window and the graph for the new parameter is displayed in the new pane.

- **To select one graph at a time per system meter, click the selected graph within the System Meter window.**

  The graph with a darker background is selected.

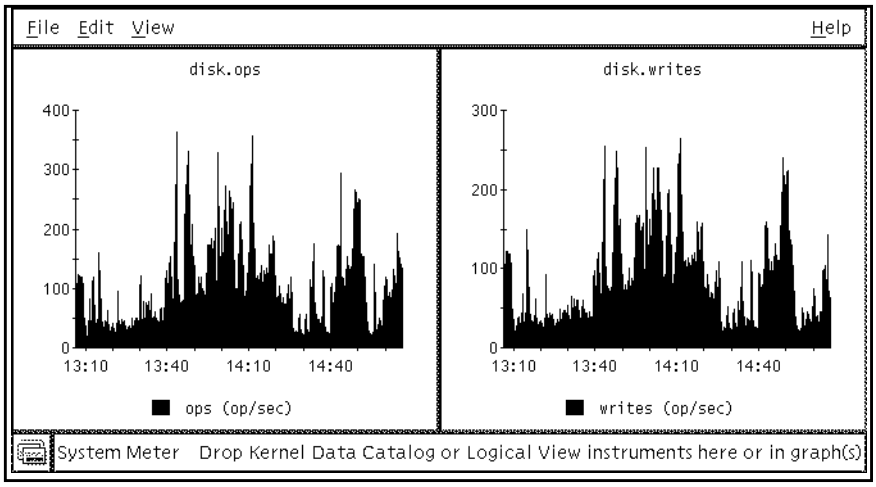

FIGURE 3-19 Displaying Two System Meters in the Same Window

- **To display multiple parameters on the same graph, drag an instrument and drop it in the graph to which you want added.**
- **To print a SysMeter, select Print from the File menu.**

Customizing a System Meter

To customize a System Meter window, use the following options under the View menu. The View menu options are:

- Layout
- Graph style
- Title

- **To use the Layout option to control the appearance of a System Meter, use these options:**
  - Horizontal—Displays graphs horizontally, from left to right
  - Vertical—Displays multiple graphs vertically, from top to bottom
  - Matrix—Displays multiple graphs in a grid format

▼ To Use the Graph Style Option

1. **Select Graph Style from the View menu.**

   The Graph Style window is displayed (see FIGURE 3-20).

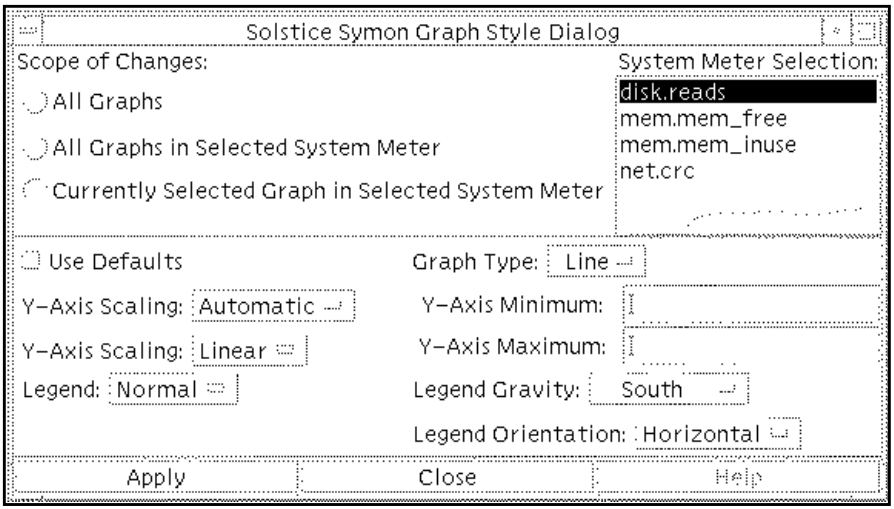

FIGURE 3-20 Graph Style Window Under the View Menu

2. **Select one of three options under the submenu "Scope of Changes:" in the top portion of the Graph Style window.**
   - All Graphs
   - All Graphs in Selected System Meter
   - Currently Selected Graph in Selected System Meter

   For an explanation of the bottom half of the window, see TABLE 3-6.

- **To change the title of the System Meter or the currently selected graph, use the Title option.**.

▼ To Modify the Selected SysMeter

1. **Select the SysMeter using the "System Meter Selection" field (**FIGURE 3-22**).**
2. **Modify graph attributes of the SysMeter using the bottom half of the window (**TABLE 3-6**).**

   For example, you can change graph scaling, Y axis minimum and maximum, legend gravity, and legend orientation.

TABLE 3-6 shows the graph attributes that you can modify.

TABLE 3-6 Graph Style Window Attributes

Field	Option	Description
Use Defaults	Active/Inactive	Selects default settings for all fields at the bottom of the form
Y-Axis Scaling	Automatic	Makes Y-axis scaling changes to reflect changing values (default)
	Fixed	Makes Y-axis scaling constant, regardless of changes in values
Y-Axis Scaling	Linear	Makes Y-axis scaling linear (default)
	Log	Makes Y-axis scaling logarithmic; useful when a graph includes two parameters with vastly different values
Legend	Normal	Shows the legend name for each variable(default)
	None	Omits the legend
	Long	Extended label for each variable; for example, `cpu.cpu0.busy` (%)
Graph Type	Area	Draws an area graph
	Line	Draws a line graph (default)
Y-Axis Minimum		Sets the minimum value for the Y-axis; used only if Y-Axis is “Fixed”
Y-Axis Maximum		Sets the maximum value for the Y-axis; used only if the Y-Axis if “Fixed”
Legend Gravity	Top	Displays the legend at the top
	Top-right	Displays the legend at the top right
	Right	Displays the legend at the right
	Bottom-Right	Displays the legend at the bottom right
	Bottom	Displays the legend at the bottom (default)
	Bottom-Left	Displays the legend at the bottom left
	Left	Displays the legend at the left
	Top-Left	Displays the legend at the top left
Legend Orientation	Vertical	Displays the legend in a vertical format
	Horizontal	Displays the legend in a horizontal format (default)

▼ To Change the Title of the System Meter or the Currently Selected Graph

1. **Select Title from the View menu with the left mouse button.**

   A window is displayed.

2. **Type the new title in the text field and click Apply.**

   The title of the System Meter or graph is changed.

▼ To Change a Graph Within a System Meter

1. **Choose the Graph Styles option from the View menu of any System Meter.**
2. **Select the graphs that you want to change, using the top section of the window (see** FIGURE 3-20**).**
3. **Make the desired changes to the controls on the bottom half of the window.**
4. **Click Apply.**

▼ To Delete a Graph Within a System Meter Window

1. **Select a graph.**
2. **Select Delete from the SysMeter Edit menu (see** FIGURE 3-19**)**

- **To undo a change in a graph, select Undo from the SysMeter Edit menu.**

▼ To Delete a Variable from a Graph

Note – If you have more than one legend in a graph, you can delete one of the legends. See the following procedure for details.

---

1. **Place the pointer over the legend for the variable (see** FIGURE 3-19**) and click the left mouse button.**

   A dialog box asks, “Do you want to delete the following data set?” with the following information in the dialog box:

   - System Meter: Variable name
   - Graph: Defines parameters being displayed
   - Data Set: Lists the operation being monitored

2. **Delete the variable by clicking the YES button.**

Displaying the System Meter Graphs

You can move, rotate, scale, or zoom the contents of your graph to better display the information with the following procedures:

▼ To Shift the Graph Within its Framer

1. **Position the pointer within the System Meter.**
2. **Press and hold the Shift key and the middle mouse button.**
3. **Move the graph using the mouse.**

▼ To Scale a Graph

1. **Position the pointer within the selected graph.**
2. **Press and hold both the Control key and the middle mouse button.**
3. **Move the mouse to scale the graph.**

   Releasing the mouse button stops graph movement within the graph.

▼ To Zoom a Graph

1. **Press and hold down the Control and Shift keys and the left mouse button at one corner of the graph.**
2. **Drag the mouse to form a rectangle around the area you want to magnify while holding down the left mouse button.**
3. **Release the mouse button to define the rectangle and to complete the zoom.**
4. **Release the Control and Shift keys.**

Note – If you release the Control or Shift keys too early, you start to drag-and-drop instead of zoom.

- **To restore a graph to the default setting, position the cursor within the desired graph and type** `r`.

Saving and Redisplaying System Meter Configurations

You can save and recall the System Meter and your defined layouts and configurations. For example, you may have a specific set of CPU monitoring graphs that you would like to view every time you run the software.

▼ To Save the System Meter Configuration into a New File

1. **Choose the Save As option from the File menu.**

   The File Selection Dialog box is displayed.

2. **Select the directory and file name from the Directories and Files submenu or enter a new file name in the Selection submenu.**

▼ To Save an Altered System Meter Configuration

- **Choose the Save option from the File menu.**

  The SysMeter contents are written to the existing file. The name of the saved file is displayed in the footer of the System Meter window.

▼ To Recall a Saved System Meter

You can save or restore sets of System Meters by using the File menu in the Kernel Data Catalog.

1. **Choose the Open option from the File menu.**

   The File Selection Dialog box is displayed.

2. **Select the file name from the Directories and Files submenu or enter the file name in the Selection panel.**

- **To save or restore sets of System Meters, use the File menu in the Kernel Data Catalog.**

Using the Log Viewer

Use the Log Viewer to search `/var/adm/messages` as defined in `/etc/opt/SUNWsymon/log_scan.tcl`, for entries that occur at a specific time or for entries that contain a specific keyword or word pattern. With this version of Solstice SyMON you can also search `/var/opt/SUNWvts/logs/sunvts.err`. You can use keywords or expressions to search for particular types of items in `/var/adm/messages` without using UNIX commands.

- **To display the Log Viewer console, open the Log Viewer console as shown in** FIGURE 3-21**.**

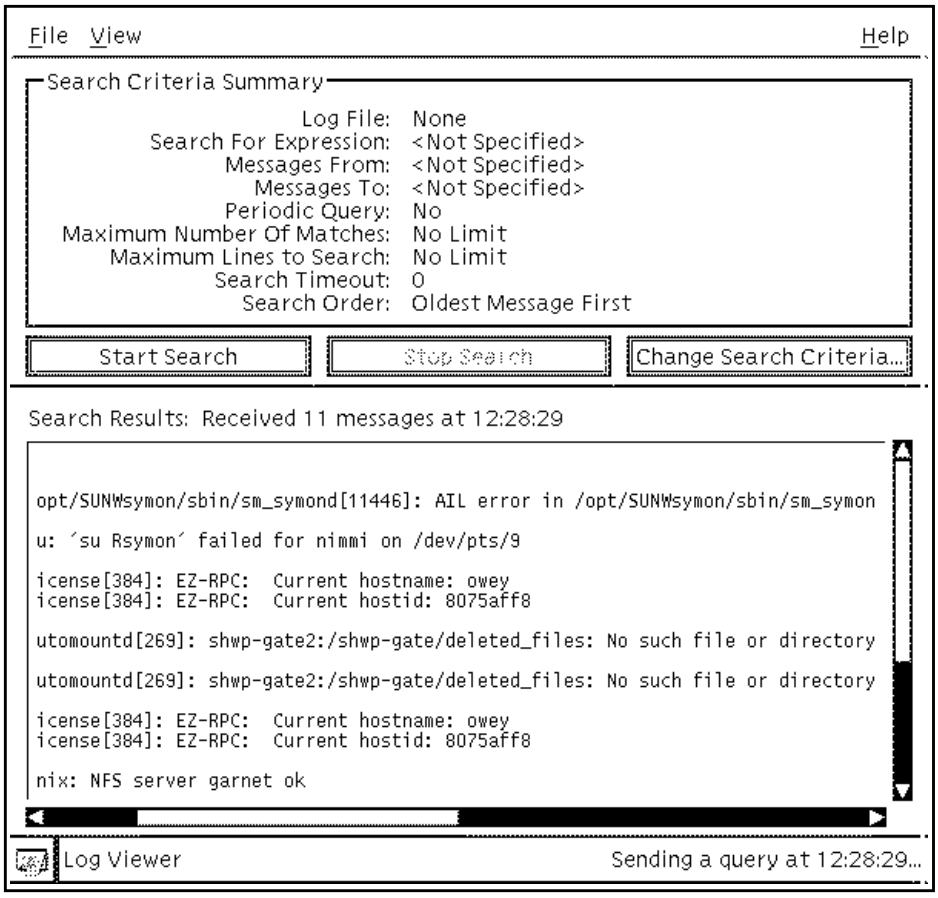

FIGURE 3-21 Log File Viewer Window

Log Viewer and Log Scanner

Until the Log Scanner is completely initialized, the Log Viewer console remains inoperable or inactive and a slashed circle is displayed (∅). After Log Scanner initialization, the Log Viewer console is available. Because your log files may be large, it may take a while for the Log Scanner to initialize and for the Log Viewer to be available.

Note that if the Log Scanner quits operation, the console for the Log Viewer becomes inactive. After the agents are restarted and the Log Scanner is completely initialized, the message “Log Scanner is back” is displayed.

View Menu in the Log File Viewer Window

The View menu of the Log File Viewer controls the appearance of the window and provides status information. The options are:

- **Log File Status**—Describes the Log Scanner status
- **Show Search Criteria Summary**—Displays the Search Criteria Summary pane of the window
- **Don't Show Search Criteria Summary**—Hides the Search Criteria Summary pane of the window
- **Display Oldest Messages First**—Defines how results are displayed in the Search Result window; displays the oldest message first
- **Display Newest Messages First**—Defines how results are displayed in the Search Result window; displays the newest message first

Searching for Log File Entries

The Search Results window is empty until you specify the search criteria in the Search for Expression entry (see FIGURE 3-21) and initiate the search. You can specify your own Search Criteria, retrieve and use a stored Search Criteria, or retrieve and modify a stored Search Criteria. Log entries retrieved by a search include the items listed in TABLE 3-7.

TABLE 3-7 Log Entry Search Criteria

Searches for	Description
Log File	Name of the log file viewed
Search for Expression	User-defined keyword, string, or regular expression
Messages From	Start date and start time of search
Messages To	End date and end time of search
Periodic Query	Interval (in minutes) between repeated searches
Maximum Number of Matches	Only this number of messages specified is reported
Maximum Lines to Search	The number of lines (a positive integer) to perform the search on
Search Timeout	Time elapsed when the search is ended; the default is 20 seconds
Search Order	Oldest (or newest) message searched first

If Solstice SyMON locates entries in the selected log file that meet the search criteria, they are displayed in the Search Results display.

- **To initiate a search, select the Start Search button from the Log File Viewer window (**FIGURE 3-21**).**
- **To stop a search, select the Stop Search button from the Log File Viewer window.**

Note – Due to the client-server structure of Solstice SyMON, results may not appear for several seconds after submitting the search. This condition is more evident when the server is fully loaded.

- **To sort the results of the search, select either the Display Oldest Messages First or the Newest Messages First from the View menu.**

▼ To Define Your Search Criteria for Specific Log File Entries

1. **Select the Change Search Criteria button in the Log File Viewer window (**FIGURE 3-21**).**
2. **Specify the Start Date, Start Time, End Date, and End Time**

   a. **Move the pointer to the “From” field text field.**

   Enter the time in 24-hour format (FIGURE 3-22). The “From” time defaults to the time that the log was created.

   b. **Move the pointer to the “To” text field.**

   Enter the time and date field. The “To” time defaults to the current time. See FIGURE 3-22.

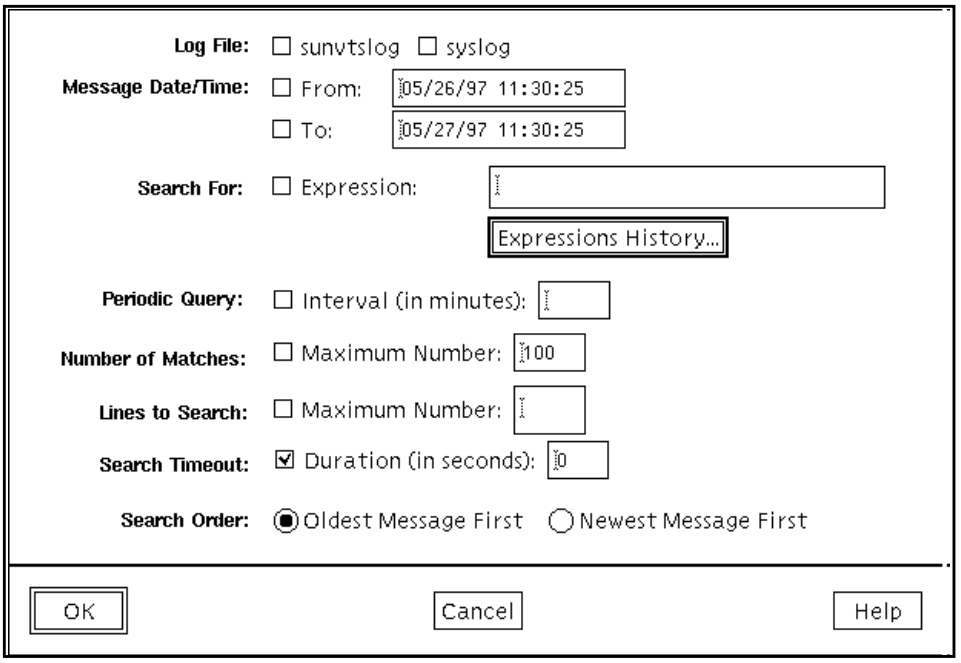

FIGURE 3-22 Log Viewer Change Criteria Worksheet Window

- **To search for Log File entries that contain a specific expression or keyword, move to the Search for Expression field and enter the expression or keyword.**
- **To perform more complex searches, specify a `grep`-like regular expression instead of a keyword.**

  For regular expression syntax, refer to the `grep` man page.

Search Criteria and Results

From the File menu of the Log File Viewer window you can:

- Load Search Criteria
- Save Search Criteria
- Save Search Result
- Close

These tasks are self-explanatory.

Using the Process Viewer

To monitor the most demanding processes and determine if you need to reallocate resources, use the Process Viewer.

By viewing what particular processes are running on the CPU, you can determine if a particular server is busy due to a temporary overload or because of a progressive increase in system usage. Processes that are inactive or consume little CPU time are not reported in the Process Viewer.

Each entry displayed on the Process Viewer console (FIGURE 3-23) represents a process with the following data:

- When the process was started
- Who started the process
- How much CPU time the process used
- Current process status

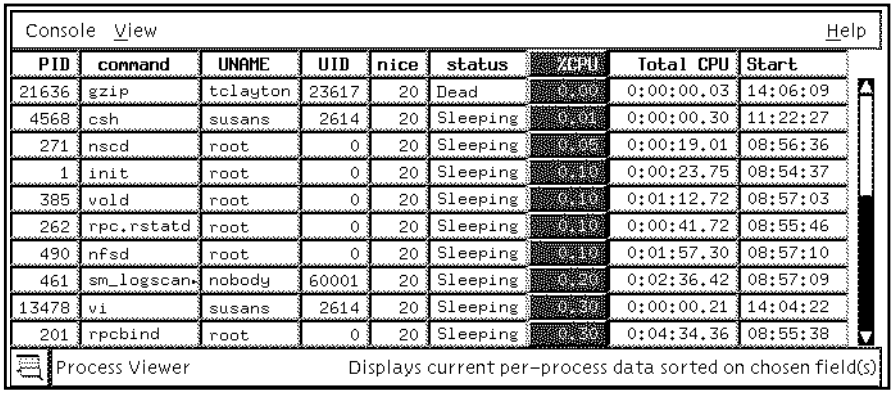

FIGURE 3-23 The Process Viewer

TABLE 3-8 describes the information displayed in each Process Viewer entry.

TABLE 3-8 Process Viewer Entry Descriptions

Field	Description
PID	Process ID
command	Name of program
UNAME	User name
UID	User ID
nice	Determines when the process can voluntarily lower its priority; as the nice value increases, the process is more willing to give up CPU scheduling priority (value defaults to 20); for more information, see the nice man page
status	Lists the current status of the process: - Sleeping: No activity generated - Running: Generating activity - Runable: Waiting for I/O - Trace: Running a debugger
% CPU	Percentage of CPU time used over the life of the process
Total CPU	The CPU time used by the system, user, and short-lived processes
Start	Time the process started; if the process has been running for more than 24 hours, it displays the date the process started

Customizing the Process Viewer Display

You can customize the Process Viewer window to display specific information. You can set the order in which columns are displayed, their default width, and the default sort column through the `common.tcl` file, which is read when Solstice SyMON is first started.

▼ To Customize the Process Viewer

- If you are responsible for customizing Process Viewer for all Solstice SyMON users, you must edit the `/opt/SUNWsymon/lib/tcl/C/common.tcl` file.
- If you are a Solstice SyMON user, copy the `/opt/SUNWsymon/lib/tcl/C/common.tcl` file to `$HOME/.symon/lib/tcl/C/common.tcl`. Replace any symbolic links for `common.tcl` in that directory. Edit the personal copy.

The `common.tcl` file is read and interpreted when you start the GUI. This file controls the appearance of these consoles.

You can set the following attributes:

- Order in which columns are displayed
- Default column width
- Default sort column
- Justification for column titles

The `common.tcl` file sets the following Tcl variable, `symon_process_columns_2`, which controls the appearance of the Process Viewer.

Within this variable is a line for each data item. The order in which these data item lines appear in the Tcl variable is the order in which their respective columns appear in the displays.

A typical line looks like this:

```
{"tpcnt"  "  Total (%)  "    "+"}
```

TABLE 3-9 describes the meaning of the strings in the line listed above.

TABLE 3-9 Description of a Typical Line for Each Data Item

String	Meaning
`"tpcnt"`	Internal name for the data item
`" Total (%) "`	Title for the column in which it is displayed; the total length of this string (including spaces) controls the default width of the column
`"+"`	Controls the default sort order of the column (+ for ascending, - for descending); this string can be only - or +

Some lines may have a fourth string with a format specifier such as `%s` or `%d`; these are not used and may be omitted.

▼ To Control the Sorting of Entries in the Process Viewer Display

- **Use the options under the View menu.**
  - **Sort**—The menu displays a list of the fields (columns) for each process view. You can choose one of these fields as the basis for sorting the process views. You can also click the column's header to change the sort order.

- **Ascending Sort Order**—Presents the display in ascending sort order for the selected column.
- **Descending Sort Order**—Presents the display in descending sort order for the selected column.

Resizing Columns

You may need to resize the columns to read the displayed information. You'll know if the column is too narrow because a small black triangle in the column is displayed. To view all of the information in the column, resize the column.

▼ To Resize a Column

1. **Place the pointer in the column that you want to resize.**
2. **Move the pointer to the left or right margin of the column.**
   The pointer changes from a cross hair to an arrow (⇥).
3. **Press and hold the left mouse button.**
4. **Drag the margin to the desired spot and release the mouse button.**

Monitoring Alarms and Events

The Event Manager monitors data from the server and generates events when abnormal conditions occur on your server. The condition, and their corresponding alarms, are described by a set of event rules. **Chapter 4, "Understanding and Writing Event Rules**," describes the syntax of the Tcl commands used to write event rules and provides examples of event rules.

Using the Event Viewer

The Event Viewer gives a tabular listing of potential problems or failures. Use this information to react proactively to prevent problems or to quickly identify and repair failures. When you launch the Event Viewer, the information in the Event Log file is displayed (see FIGURE 3-24).

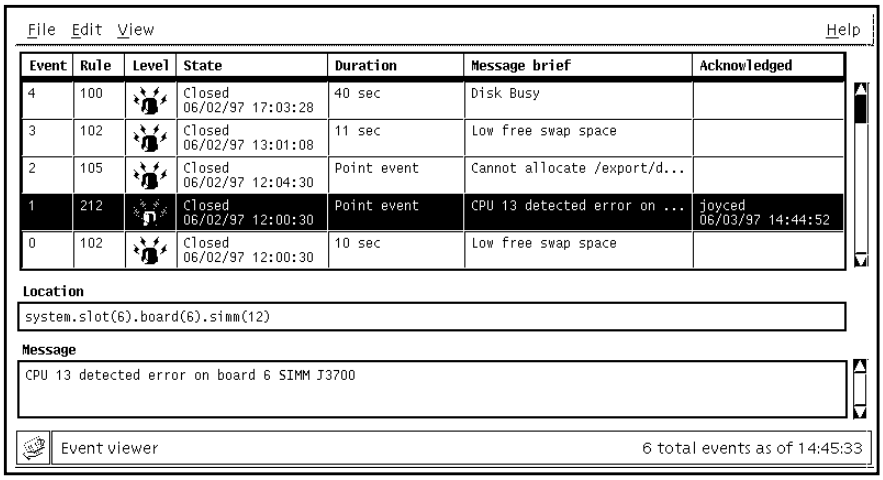

FIGURE 3-24 Event Viewer

When the Event Viewer is notified of an event, it adds a description of the event to its display, which includes the information listed in TABLE 3-10. In the Message brief column, a lengthy message is abbreviated with ellipses (. . .).

TABLE 3-10 Description of Entries in the Event Viewer for Open Events

Field	Description
Event	Sequential number assigned to the event
Rule	Number of the rule that caused the event
Level	Level of events: yellow (caution), red (danger), blue (capacity warning)
State	This field can be Open, Closed, or Fixed; also notes the time the event was open, closed, or fixed.
Duration	Length of the event in days, hours, minutes, and seconds
Message brief	A shortened message of the entire event; you can see the entire message in the Message text area below the table when you double click on the row with the message in it
Acknowledged	User ID of person acknowledging the event and time the user acknowledged the event

Event Viewer Menus

The Event Viewer has three menus:

- File
- Edit
- View

The menu choices available from the File menu are:

- **Save as**—Presents a dialog window from which you can save the table data to a file
- **Print**—Presents a dialog window from which you can print the table data
- **Close**—Iconizes the Event Viewer

The menu choices available from the Edit menu are:

- Select all
- Acknowledge
- Fix
- Delete
- Delete all

Most of these menu items are self explanatory. To mark an event as fixed, you use the Fix menu item. See "To Mark an Event as Fixed" later in this chapter. To acknowledge an event, see "To Acknowledge an Event" later in this chapter.

The menu choices available from the View menu of the Event Viewer are:

- **All Events**—Displays all events (even if closed), until they are deleted.
- **Open Events**—Displays only currently open events. This view is displayed when you select the Event Viewer icon.
- **Sort . . .** —Brings up a dialog window that allows you to select the item to sort on such as Event, Rule, Duration, and the Ascending or Descending sort order (see FIGURE 3-25.)

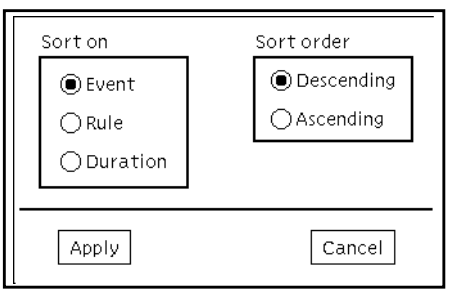

FIGURE 3-25 Sort . . . Dialog Window from the View Menu

When an event is open, the Event Manager highlights the event in the Logical View and Physical View or the Kernel Data Catalog until the cause of the event is corrected or the event is closed.

When the condition that caused an event ends, the description of the event is removed from the Open Event view and any associated highlighted consoles are also removed. The Event Manager notes the closing of the event in the event log.

Acknowledging and Deleting Events

When you acknowledge an event, it has no effect on the event condition or alarms. However, it informs all users that the event has been observed.

▼ To Acknowledge an Event

1. **Select the event to be acknowledged by clicking any column in the row with the left mouse button.**

   The entire row is highlighted.

2. **Select Acknowledge from the Edit menu.**

   The Event Manager enters the time, date, and User ID in the Acknowledge column.

Note – If you acknowledge or delete an event, the change appears in all GUIs.

After the event is closed, it remains in the All Events view until you delete it.

▼ To Mark an Event as Fixed

1. **Select the event you want marked as fixed by clicking any column in the row with the left mouse button.**

2. **Select fix from the Edit menu.**

   If the event is allowed to be marked as fixed, the Event Manager marks the event as fixed and enters the time, date, and the user ID in the state column. The state of the event changes from Open to Fixed by *username*.

▼ To Delete a Closed Event

1. **Select All Events from the View menu.**

2. **Select the event you want deleted by clicking the event (which highlights it) and select Delete from the Edit menu.**

Note – You cannot delete an Open event from the event list.

Troubleshooting

To troubleshoot problems with the GUI, see **Chapter 6, "Troubleshooting**."

CHAPTER 4

Understanding and Writing Event Rules

This chapter explains how to write or modify event rules that are written in the Tcl scripting language. It describes the categories of event rules, the syntax of the Tcl functions, and how to write or modify event rules for agents including: Config Reader, Kernel Reader, Log Scanner, and Event Generator. At the end of this chapter, examples of simple and complex rules are presented.

Overview of Event Rules

This section describes and explains:

- Terminology used with event rules
- Categories of event rules
- Rules files
- Special characters and reserved words
- Attributes

Event rules are not required to include an action and can set a value that is used in other rules. All rules are combined into a single Tcl variable called `Rules.tcl`, which is case sensitive.

For a list of books describing the Tcl language, see “Related Documents” in the Preface.

Caution – You must know Tcl to modify event rules. Do not modify event rules if you do not know Tcl. Incorrectly modifying an event rule will cause an error or an incorrect result to occur.

Terminology

Events are hardware or operating system conditions that may require the attention of a system administrator. Examples of events include:

- Processor overload
- Excessive swapping
- Failed power supply
- Failed fan
- Soft disk errors
- Extreme temperature conditions

Before the Event Generator subsystem can alert you of an event, you must first write a rule that defines the event. A *rule* includes a *condition* and other attributes that define the state of the rule and the subsequent *actions* to take.

A condition is an expression that defines when a rule is active. An example of a condition is a failed board. Actions identify how to tell users about a situation that may require attention. For example, an action tells Solstice SyMON what to do when a condition is true, if the condition changes, or if the system shuts down.

In addition to conditions and actions, a rule may also include the attributes listed in TABLE 4-1.

TABLE 4-1 `LEVEL`, `PRIORITY`, and `SEVERITY` Attribute Descriptions

Attribute	Description
`LEVEL`	Causes the appearance of an appropriately colored icon in the `LEVEL` column of the event display. By convention, YELLOW is caution, RED is danger, and BLUE is a capacity warning.
`PRIORITY`	Arbitrary integer value. For the rules that accompany Solstice SyMON, 1 is the highest priority and 4 is the lowest priority. You can swap or customize these values. For example, you can make 4 the highest priority and 1 the lowest priority. You can also customize the priority by assigning other numeric values, such as 99, to `PRIORITY`.
`SEVERITY`	Arbitrary integer value. For the rules that accompany Solstice SyMON, 1 is the highest priority and 4 is the lowest priority. You can swap or customize these values. For example, you can make 4 the highest priority and 1 the lowest priority. You can also customize the severity by assigning other numeric values, such as 99, to `SEVERITY`.

Types of Event Rules

Sun supplies a set of pre-defined rules with Solstice SyMON (see **Appendix D, "Default Solstice SyMON Rules**." They are divided into four major categories (see TABLE 4-2).

TABLE 4-2 Types of Event Rules

Category	Description
Non-hardware	Monitors non-hardware, such as CPU usage and swap space
Hardware	Monitors hardware for potential trouble, such as a failed processor or extreme temperature conditions
Capacity Planning	Takes a long term view of events; monitors CPU, memory, disk and network processes to identify potential bottlenecks and provides information about how to upgrade a system
Predictive Failure Analysis	Looks at the soft error rates of SIMMs and disks; issues a warning if a potential component failure is identified

Description of Rules Files

The `rules.tcl` file defines the location of the rules that tell Solstice SyMON what to do in a given situation. Each type of event rule has a rule file (see TABLE 4-3). To tell the Event Generator where to look for theses rules files, set the path in `event_gen.`*servername*`.tcl`. For a list of the default paths, see "Locating Rule Files" later in this chapter.

The `rules.tcl` file reads the rest of the rules files listed in TABLE 4-3 into the input stream by doing a `psource` of each of the rule files.

```
psource n swrules.tcl
psource n syrules.tcl
psource y rultext.tcl
```

CODE EXAMPLE 4-1 `psource` Code Example

where:

`n` after the `psource` command in the `rules.tcl` file means not to issue an error message if it can't find the file.

`y` after the `psource` command in the `rules.tcl` file means to issue an error message if it can't find the file

All of the rules files listed in TABLE 4-3 are located in `rultext.tcl`. There is a different `rultext.tcl` file for each country in which the software is localized. These `rultext.tcl` files are located in subdirectories. For example, the English `rultext.tcl` file is located in the `C` subdirectory.

The following rules files (see TABLE 4-3) contain generic rules, which apply to any system type. However, platform specific rules (see TABLE 4-4) apply only to one type or one class of system.

TABLE 4-3 Description of Rule Files

Rule	Description
`rules.tcl`	Organizer (master file)
`cprules.tcl`	Capacity planning rules
`hwrules.tcl`	Hardware monitoring rules
`syrules.tcl`	Monitoring rules
`egrules.tcl`	Event Generator rules
`pfrules.tcl`	Predictive failure rules
`swrules.tcl`	System monitoring rules
`rultext.tcl`	Message string definition rules

TABLE 4-4 lists platform-specific rules files and the platforms to which they apply.

TABLE 4-4 Description of Platform-Specific Rule Files

Rule	Description
`SS1000.tcl`	SPARCserver 1000 / 1000E
`SC2000.tcl`	SPARCcenter 2000 / 2000E
`UEnterprise.tcl`	Ultra Enterprise 3000, 4000, 5000, and 6000 servers
`UEnterpriseI.tcl`	Ultra Enterprise 2 and 150 servers
`UE450.tcl`	Ultra Enterprise 450 server

For each hardware platform, there is a corresponding `rultext_<`*hardwarename*`>.tcl` file. For example, for the SPARCserver 1000 and 1000E, there is a `rultext_SS1000.tcl` file and for the SPARCcenter 2000 there is a `rultext_SC2000.tcl` file. The `rultext_<`*hardwarename*`>.tcl` files contain messages that can be translated into different languages.

Special Characters

The following special characters are used in the Tcl language:

- " " (double quotes)
- { } (curly brace)
- [ ] (square bracket)
- # (pound sign)
- * (asterisk)

Follow these guidelines when using these special characters:

- Enclose each Tcl function string in curly braces { }.
- If you include spaces in an attribute data item, enclose the data item in double quotes.

Reserved Words

TABLE 4-5 describes the Tcl reserved variable names, which contain certain values that have specific meaning for the Event Generator.

Note – Do not redefine these variable names.

TABLE 4-5 Reserved Tcl Variable Names

Variable Name	Description
`symond_status`	Tells whether the `symond` daemon is running
`server_status`	Tells whether the monitored system is up
`LogScanner_status`	Tells if the Log Scanner agent is running
`ConfigReader_status`	Tells if the Config Reader agent is running
`KernelReader_status`	Tells if the Kernel Reader agent is running
`node`	Contains the root hierarchy that is being evaluated in the rule; accessible only from within a `MULTI` rule (see "MULTI Rules" later in this chapter)

TABLE 4-5 Reserved Tcl Variable Names *(Continued)*

Variable Name	Description
`my_root`	Contains the root node for the Event Generator hierarchy; this variable should not be changed
`value`	Contains the value of the node returned by the `findlist` function in a `MULTI` rule; accessible only from within a `MULTI` rule (see "MULTI Rules" later in this chapter)
`timestamp`	Contains the timestamp of the matched message

Attributes

A rule is a list of attributes. Attributes list conditions under which the rules should activate or close down and what actions to take when the rule is activated. A condition of a rule can look at time or other values in the environment.

Each attribute has a label followed by a value. Separate the components of an attribute with spaces, tabs, or new lines and separate each attribute with a new line or space.

TABLE 4-6 describes attribute descriptions.

TABLE 4-6 Attribute Descriptions

Label	Value	Attribute Description
`RULE`	Integer	Rule number displayed by the Event Viewer to show which rule generated the event. The rule number must be unique across all rule files.
`ON_OPEN`	Tcl script	Tcl string giving actions to take when the condition of the rule becomes true
`ON_CONTINUE`	Tcl script	Tcl string giving actions to take when the condition of a rule continues to be true
`ON_CLOSE`	Tcl script	Tcl string giving actions to take when the condition of a rule is no longer true
`ON_ACKNOWLEDGE`	Tcl script	Interpreted when the Event Generator receives an acknowledgment of an event from the GUI
`ON_SHUTDOWN`	Tcl script	Tcl string to be interpreted when the Event Generator shuts down
`PARAMETERS`	String	User-defined parameters
`MULTI`	Tcl script	Tcl string to provide a list of data node variables for multiple events of a single rule

TABLE 4-6 Attribute Descriptions *(Continued)*

Label	Value	Attribute Description
SEVERITY	Integer	Severity of the event (user-defined and interpreted); 1 = most severe and 4 = least severe; see also TABLE 4-1, earlier in this chapter, which describes this attribute in more detail
PRIORITY	Integer	Priority of the event (user-defined and interpreted); 1 = most severe and 4 = least severe; see also TABLE 4-1, earlier in this chapter, which describes this attribute in more detail
RATE	Integer	The frequency to execute the rule (seconds); for example, `RATE 60` means the rule will evaluate every 60 seconds; the default is to execute the rule every time the Event Generator receives new data from the server
COMMENTS or C	String	Comments field
LOG_RULES	Log script	Tcl script that defines the log scanner activity that supports rules
ON_FIX	Tcl script	Specifies if an event can be marked as fixed; the Tcl script is interpreted by the Event Generator when an event is marked as fixed

MULTI Rules

This section describes `MULTI` rules, which are used in Tcl functions. The following paragraph explains this code example:

```
MULTI { expr { [ findlist KernelReader.disk.*.busy .busy ] } }
```

The `MULTI` statement tells the Event Generator to repeat the same rule on each node that is returned from the `findlist` function. The `findlist` function finds the node whose name matches `KernelReader.disk.*.busy`. Since the strip string (strip string means remove or strip out) is defined as `.busy`, the node name that is returned in the `node` variable by the `findlist` function contains only the matched node name, which excludes `.busy`. For example, if the `findlist` function matches a node named `KernelReader.disk.sd8.busy`, the `$node` variable contains `KernelReader.disk.sd8`. Because the strip string is not null, it does not have to be in double quotes.

Rule Functions

This section describes the Tcl rule functions associated with the `Rules.tcl` variable.

`alarm`

The `alarm` function makes an event active, highlights the hierarchy node (`RED`, `YELLOW`, or `BLUE`) and creates entries in both the Event Log and the Event Viewer with a predefined message of the event. The syntax of the `alarm` function is:

```
alarm level node "message" "Tcl command"
```

where:

level can be: `RED`, `YELLOW`, or `BLUE`. By convention, these levels mean

- `RED` is used for a situation or event that requires immediate attention
- `YELLOW` is used for warning events
- `BLUE` is used for capacity planning events

node is the path name for a hierarchy node to be highlighted on the display. Using an empty string (" ") instead of the node name means you do not want the node to be highlighted. *node* contains the name of the node that is in question or in error.

message is a string that is displayed in the Event Viewer. Messages can be either text strings or variables. For internationalization, the text strings are set to variables in double quotes, which expands to the expected string. The variables are used to reference entries in `rultext.tcl`. For example, the following line of a Tcl function uses the "`r0mess`" variable for internationalization. For the complete rule, see CODE EXAMPLE 4-3.

```
ON_OPEN { alarm RED "" "$r0mess" "" }
```

Tcl command is a Tcl string that is executed by the GUI when it receives the event. If you do not want to execute a Tcl function, use an empty string (" ").

To close an open `alarm` condition, see the `end_alarm` function.

crnode

The crnode function generates a Config Reader node based on elements returned from the Log Scanner. The syntax is:

```
crnode keyword
```

Examples of the *keyword* are: DISK, SIMM, CPU, QLGC, PLN, SOC.

TABLE 4-7 defines the keywords.

TABLE 4-7 Keywords for the crnode Command

Keyword	Description
DISK	Uses one argument; examples are sd0 or ssd0
SIMM	Uses two arguments (for example 0 J3201); in this example, 0 is the board number and J3201 is the SIMM reference number
CPU	Uses one argument, the CPU number
QLGC	Uses one argument, the instance number; for more information, see the isp man page
PLN	Uses one argument, the instance number; for more information, see the pln man page
SOC	Uses one argument, the instance number; for more information, see the soc man page

Examples of the crnode function are: `crnode DISK sd0` and `crnode SIMM 0 J3201.`

debug

The debug function provides output to a person who is writing rules to explain what is happening to a rule. This function is analogous to puts in Tcl.

The syntax is as follows in which *msg* is a text string that is placed in debug and must appear in double quotes (“ ”).

```
debug "msg"
```

The debug function writes debug messages to:
/var/opt/SUNWsymon/<*monitored_machine*>/EG/eg_debug.<*pid*>

dynlink

The dynlink function takes a shared object file (.so) and procedure name and dynamically links the shared object and calls the specific procedure. The syntax of the dynlink function is:

```
dynlink file name_of_function
```

- *file* is the full path of the shared object (typically ending in .so).
- *name_of_function* is the function name within the dynamically loadable object.

An example of the dynlink function is:

```
dynlink /opt/SUNWsymon/lib/eg_pfa.so pfainit
```

- /opt/SUNWsymon/lib/eg_pfa.so is the file name.
- pfainit is the function name. Only pfainit is needed to do the initialization.

end_alarm

The end_alarm function has no arguments. Use end_alarm to close an open alarm condition. An example of this function is:

```
end_alarm
```

findlist

The findlist function takes a hierarchy path in which wildcards replace nodes, and replaces it with a list of valid paths in the hierarchy that match the wildcard. It builds the list of matching nodes and returns the most recent value associated with the path, such as set foo $value.

The syntax is:

```
findlist hierarchy_path "strip_string"
```

An example of a *hierarchy_path* is `KernelReader.cpu.cpu1.busy.` The hierarchy path uses the period (.) as a path separator.

"strip_string" (strip string means remove or strip out) is any subpath that is on the full path. Enclosing an item in the "*strip_string*" deletes that item from the expanded path (see FIGURE 4-1). If the strip string is null, you must use double quotes. If the strip string is not null, you don't need to enclose the strip string in double quotes.

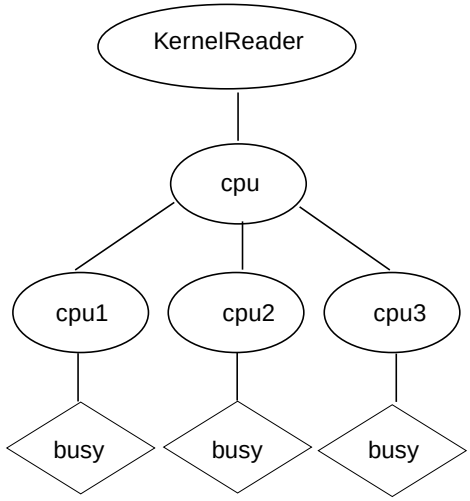

FIGURE 4-1 `KernelReader.cpu.*.busy` Hierarchy Example

For example, when you invoke the `findlist` function, the following function:

```
findlist KernelReader.cpu.*.busy ""
```

is expanded to:

```
KernelReader.cpu.cpu1.busy
KernelReader.cpu.cpu2.busy
KernelReader.cpu.cpu3.busy
```

Another example of the `findlist` function is:

```
findlist KernelReader.cpu.*.busy ".busy"
```

which expands to:

```
KernelReader.cpu.cpu1
KernelReader.cpu.cpu2
KernelReader.cpu.cpu3
```

Note that `.busy` is not in the expanded paths because in the `findlist` function, `.busy` was enclosed in double quotes in the *strip_string*. By placing `.busy` in the *strip_string*, this deletes `.busy` from the expanded path.

Use the `findlist` function only in a Tcl script related to the `MULTI` attribute. The entire list that is created is kept internal to the Event Generator and is not available all at once to the script.

Each member of the list is available to the script, one at a time. The action is called once for each node-value pair. If this were the rule:

```
MULTI {command}
{command2 }
```

It could look like this in the C shell:

```
foreach i ( ‘command‘ )
command 2
```

For an example of the `findlist` function, see Rule 10000 later in this chapter.

Implicit Instance Matching

The `findlist` and `hotlist` functions have the implicit instance matching feature. This feature allows any node entry in a `findlist` or `hotlist` hierarchy path that is *not* a wildcard entry and does *not* contain an instance number to match nodes that have the same name but have an instance number. An example of an entry that meets these criteria is: `system.slot(0).board(0).cpu`. Note that the node `cpu` in the previous path is not a wildcard entry and it does not contain an instance number, such as `cpu(1)`.

For example, a single rule for: `system.*.*.cpu.temperature` matches all CPU nodes with temperature properties. The string `system.slot.board.cpu.temperature` also matches all CPU nodes with temperatures properties since you can have several instance matches in the `findlist` or `hotlist` path.

Multilevel Wildcard String Support

The `findlist` snd `hotlist` functions also have multilevel wildcard string support, "**". This feature was first introduced in Solstice SyMON 1.4. Multilevel wildcard string support allows a single rule to match nodes at different levels of the hierarchy. The string ** matches any number of nodes.

Note – You must use only one instance of this wildcard in a hierarchy path expression.

As an example:

- **To find all** `sd` **nodes in a system, use the expression:** `system.**.sd`

findstatus

The `findstatus` function finds the status of a node. The syntax is as follows in which *node* is the hierarchy endpoint. Two valid values of *node* are `dead` or `alive`.

```
findstatus node
```

For an example of a rule that uses the `findstatus` function, see RULE 201 in CODE EXAMPLE 4-6.

findvalue

The `findvalue` function takes the name of any data hierarchy variable and returns the value of the variable. The syntax is:

```
findvalue node
```

node is the hierarchy endpoint.

An example of the `findvalue` function is:

```
findvalue $node
```

You cannot specify a time value for the value. The value used depends on the agents providing the data. For an example of a rule using the `findvalue` function, see RULE 201 in CODE EXAMPLE 4-6.

getfield

The `getfield` function returns the internal values associated with a rule or a rule-node combination (list from a `MULTI` string). It accesses publicly available fields of data from within rules.

The syntax is:

```
getfield [rule#] field_type
```

The optional variable *rule#* is the rule number, which is assigned when you write the rule. For example, 0, 1, 2, and so on.

The *field_types* are listed in TABLE 4-8:

TABLE 4-8 `getfield` Field Type Descriptions

Field Type	Description
`PARAMETERS`	User-defined string associated with the rule; use this field type to pass data between invocations of the rule
`RATE`	Frequency of the execution of the rule (seconds)
`ACTIVE`	Returns a value of either true or false; `ACTIVE` equals true when the event is active or open (when the `ON_OPEN` section of the rule is executed but the `ON_CLOSE` section is not executed;) the event is closed or inactive once the `ON_CLOSE` is executed
`COUNT`	Returns the number of consecutive iterations of the rule (read only)
`PRIORITY`	Returns the priority value of the rule (see TABLE 4-1)

TABLE 4-8 `getfield` Field Type Descriptions

Field Type	Description
SEVERITY	Returns the severity value of the rule (see TABLE 4-1)
RULE	Returns the rule number (read only)
START_TIME	Returns the time the rule became active (read only)

An example of the `getfield` function is:

```
getfield COUNT
```

The `getfield COUNT` function returns a count of the number of consecutive iterations where the condition of the rule has been true for a non-`MULTI` rule. This function also returns the number of consecutive iterations where the condition of a rule has been true for the specified rule or node combination for a `MULTI` rule. In a `MULTI` rule, the rule is evaluated for more than one node. If the rule number, which is optional, is not specified (as in the example), the rule number you are presently in is used.

For an explanation of the converse function to `getfield`, see "putfield" later in this chapter.

Caution – If you are using `get_parameter` and/or `put_parameter`, do not use `putfield` or `getfield` with the PARAMETERS field type on the same rule. Your data may become corrupt.

get_parameter

The `get_parameter` function takes a single argument and gets a value that is unique to the rule from the PARAMETERS attribute that is associated with that argument. Rules can pass values to each other. A parameter is a way to pass the value to a different rule. Parameters are placeholders for values that are used at a later time.

The `get_parameter` function manages a parameter list. One use of `get_parameter` is for `MULTI` rules that require historical records.

The syntax is:

```
get_parameter argument
```

- *argument* is an identification string (without spaces) where a particular value is stored in memory.

You initialize the `get_parameter` function by using the `PARAMETERS` attribute to set the initial default value. For example, `PARAMETERS {default 1000}`. If the `get_parameter` argument does not have a value currently associated with it, the default value is returned.

An example of the `get_parameter` function follows:

```
{
     set oldtempdegree [ get_parameter $node ]
.
.
.
```

For a description of a rule containing the `PARAMETERS` attribute and the `get_parameter` function, see CODE EXAMPLE 4-7.

Additionally, see the `put_parameter` function in "put_parameter" later in this chapter which is the compliment to `get_parameter`.

gettime

The `gettime` function returns the last sample time of the monitored machine as a long integer. There are no arguments. The syntax is:

```
gettime
```

hotlist

The `hotlist` function works like `findlist` except that it returns only the nodes whose values changed. Do not use the `hotlist` function in `cprules`, `pfrules`, or rules that contain `ON_CONTINUE` in them. That is because in these circumstances you want every node to be returned. In these instances, use the `findlist` function instead.

The following is a partial code example for `RULE 1201` The complete rule is listed later in this chapter in CODE EXAMPLE 4-6.

```
{
   COMMENTS { for hot plug charge DC status }
   RULE 1201
   MULTI {
     expr { [ hotlist system.hot_plug_charges.*.status "" ] } }
}
     set hpustatus  [ findstatus $node ]
     expr { ("$value" != "OK") && ("$hpustatus" == "alive") }
}
.
.
.
```

Note – If any component of a hierarchy path contains a period (.) in it, you may not be able to use the `hotlist` function to locate it. For example, if you search for the mount point `/export/a0.test` reported by the Config Reader via the following path:
`system.slot.board.io-unit.sbi.dma.esp.sd./export/a0.test`
the node will not be found. Instead, use the following path to find this node:
`system.slot.board.io-unit.sbi.dma.esp.sd.*` .

For more information on `hotlist`, refer to "findlist."

There are two new features for both `findlist` and `hotlist` in Solstice SyMON 1.4. They are:

- Implicit instance matching
- Multilevel wildcard string "**" support

To learn how to use these new features, refer to "Implicit Instance Matching" and "Multilevel Wildcard String Support" in "findlist."

load

The `load` function, which is specific to the Log Scanner, takes more than one argument to create a Log Scanner hierarchy that is checked as part of the condition of the rule. An example is:

```
LOG_RULES { { grep { Ecc error on board ([0-9]), reference number (J[0-9]) } syslog }
                      { load LR10000 "$logword(1)" "$logword(2)"} }
```

For a complete listing of this code example, see “Partial Listing of RULE 10000 and Description” later in this chapter.

The `load` command has three arguments:

- Rule number `LR10000`
- `$logword(1)`
- `$logword(2)`

`$logword(1)` is substituted for the value that matches the first regular expression between parentheses ( ); in this case, the board number `$logword(2)` is substituted for the value that matches the second regular expression enclosed between parentheses ( ), which in this case is the reference number. For example, the following line in the `syslog` file:
“`Ecc error on board 5, reference number J3200`” generates the following hierarchy: `LogScanner.LR10000.5.J3200`.

For more information on how to use the `load` command, see “Log Scanner” later in this chapter.

mailto

The `mailto` function mails a message string to the specified address. The syntax of the function is:

```
mailto address "msg_string"
```

If there are spaces in *msg_string*, include *msg_string* in double quotes (“ ”). All existing rule messages are in `rultext*.tcl`, to facilitate translation into other languages.

An example of the `mailto` function is:

```
mailto root "$mess"
```

Another example of the `mailto` function is:

```
mailto root "We have a problem."
```

putfield

The `putfield` function is the converse of `getfield`. It uses a field type and data and assigns the data to the field of the current rule.

The syntax is:

```
putfield  [rule#] field_type "value"
```

The optional variable *rule#* is the rule number that is assigned when you write the rule. For example, 0, 1, 2, 3, and so on.

The *field_types* are described in TABLE 4-9.

TABLE 4-9 `putfield` Field Type Descriptions

Field Type	Description
PARAMETERS	Sets the user-defined string associated with the rule. Use this field type to pass data between invocations of the rule.
RATE	Sets the frequency of the execution of the rule (seconds).
PRIORITY	Sets the priority value of the rule. For meanings of the values of PRIORITY, see "Terminology" earlier in this chapter.
SEVERITY	Sets the severity value of the rule. For meanings of the values of SEVERITY, see "Terminology" earlier in this chapter.

If there are spaces in *value,* use double quotes (" ").

An example of the `putfield` function is:

```
putfield 1 RATE 50
```

This example changes the frequency with which RULE 1 is checked. RULE 1 is checked at 50 second intervals.

For an explanation of the converse function to `putfield`, see "getfield."

put_parameter

The syntax of the put_parameter function is:

```
put_parameter tag value
```

The put_parameter function manages a parameter list. It provides a way for rules to pass values to each other (or between iterations of the same rule).

One use of the put_parameter function is for MULTI rules that require historical records. It saves a part of the data that is associated with a tag.

The arguments have the following meanings:

- *tag* is the key that is used to store data; the legal values can be any string.
- *value* is the data to be stored.

You initialize the put_parameter function by using the PARAMETERS attribute to set the initial default value. For example, PARAMETERS {default 1000}.

An example of the put_parameter function follows:

```
put_parameter [$node] RED
```

Using the $node parameter is optional. Leaving out $node makes the put_parameter function more efficient. For a complete code example that includes the put_parameter function, see "RULE 100 Code Example."

Additionally, see the get_parameter function in "get_parameter,"which is related to put_parameter.

snmp

The snmp function takes a string as an argument and generates an SNMP trap. A trap message is sent to every machine in the snmp_hosts variable, which is defined in event_gen.tcl. The syntax is as follows in which *string* is the message sent to programs such as Solstice Site Manager and Solstice Domain Manager, which monitors the system:

```
snmp "string"
```

An example follows:

```
snmp  "$imsg"
```

Messages can either appear explicitly in `snmp` or the identifier of a message in `rultext.tcl` can be given. All existing rule messages are in `rultext.tcl`, which facilitates translation into other languages. The `snmp` function goes to the `rultext.tcl` file, picks up the variables from `rultext.tcl`, runs it through the `format` function, and then displays the messages (see CODE EXAMPLE 4-3).

syslog

The `syslog` function takes a string as an argument and places the string in the `syslog`. Use this function for debugging purposes only. Use this function as you use the `printf` function in C. The syntax is as follows in which *msg* is a text string that is placed in `syslog` and must be in double quotes (“ ”). All existing event messages are in `rultext.tcl`, which can be translated into other languages.

```
syslog "msg"
```

An example of the `syslog` function is:

```
syslog "Show me the value of $variablename"
```

This example prints out the message and gives the value of the variable name.

Output from the `syslog` function is treated as warning messages. The message level for the `syslog` call is `LOG_WARNING`. To be sure that the warning messages appear in the system log file, make sure that the `/etc/syslog.conf` file is set correctly. For more information, see the `syslog.conf` man page.

Hierarchies

The next part of this chapter explains how hierarchies work and how to write event rules.

Hierarchies are used to structure data and group-related information. Each node and subnode organize the data beneath it. For example, in this hierarchy, the following is true:

```
KernelReader.cpu.cpu1.busy
```

- Top level, `KernelReader,` indicates that this is `KernelReader` data.
- `KernelReader.cpu` indicates that this is `cpu` data.
- `KernelReader.cpu.cpu1` indicates that this data is related to `cpu1`.
- `KernelReader.cpu.cpu1.busy` is the percentage of time that the `cpu1` was busy.

Hierarchies use the period (.) as a separator for hierarchy paths. For specifics on how to use the `findlist` function for locating hierarchy paths, see “findlist” earlier in this chapter. The `findlist` function cannot find the hierarchy path if any component of the hierarchy path contains a period.

For more information on the Kernel Reader see “Kernel Reader” later in this chapter. For information on the Kernel Reader hierarchy, see **Appendix A, “Kernel Reader**.”

Writing Event Rules

This section provides procedures for writing event rules and explains how to create rules from any of the four agents:

- Config Reader
- Kernel Reader
- Log Scanner
- Event Generator

▼ To Create New or Modified Rules

1. **As root or superuser, use a text editor to call up the appropriate rules files to edit (see** TABLE 4-3**.)**

   For example, if you need to add a capacity planning rule, call the `cprules.tcl` file and add the new rule to this file. Go on to Step 3.

2. **If a rules file does not exist for your rule category, create a new rules file using a text editor and add the rule to this file.**

   a. **Modify the master file** `rules.tcl` **after you create the new rule by adding a** `psource` **command to tell the** `rules.tcl` **file to read the new rule file. See** CODE EXAMPLE 4-1 **for information on** `psource` **in "Description of Rules Files" earlier in this chapter.**

   b. **Add the new rule file name to Tcl variable** `Rules` **in the** `set Rules "`*file_names*`"` **statement.**

      Note that within the quotes, the file names should be separated by spaces.

3. **Verify the rule.**

   See "Verifying New Event Rules" later in this chapter.

4. **Activate the rule.**

   See "Activating New or Modified Event Rules" later in this chapter.

The following sections describe the four agents in more detail, explain how to determine the path for the Config Reader and Kernel Reader hierarchies, explain the `LOG_RULES` that are used for the Log Scanner, and describe the functions of the Event Generator.

Config Reader

The Config Reader collects the data for the system hardware configuration and status and provides data continuously to the Event Generator. Rules written against the Config Reader describe hardware-related failures, such as power supply or fan failures. The Config Reader data begins with `system.<`*complete_path*`>`. An example is: `system.hot_plug_charges.auxillary_5v.status.`

The following screen shows a partial display of Rule 1201, which describes hot plug charge DC status. The complete rule is presented in "Complex Rule Example 2: RULE 1201" later in this chapter.

```
{
  COMMENTS { for hot plug charge DC status }
  RULE 1201
  MULTI {
    expr { [ hotlist system.hot_plug_charges.*.status "" ] } }
}
    set hpustatus  [ findstatus $node ]
    expr { ("$value" != "OK") && ("$hpustatus" == "alive") }
}
.
.
.
```

The `MULTI` label indicates that this rule might apply to more than one node. Each node found using the `hotlist` function matches the pattern `system.hot_plug_charges.*.status`. For example:

- `system.hot_plug_charges.auxiliary_5v.status`
- `system.hot_plug_charges.peripheral_12v.status`
- `system.hot_plug_charges.peripheral_5v.status`

Since the strip string is null (empty double quotes), the value assigned to the `$node` variable after the evaluation does not change.

The statements in the condition do the following:

- Obtains the status of the `node` variable and assigns it to the `hpustatus` variable
- Checks if the value of the `value` variable is not equal to "`OK`" and the value of the variable `hotplugstatus` is equal to `"alive"`

Determining the Path for the Config Reader

To write rules for the Config Reader, use the Logical View console to determine the Config Reader path.

▼ To Determine the Path for the Config Reader

1. **Click the Logical View icon.**
2. **Expand the hierarchy by clicking the + signs in the hierarchy.**

   FIGURE 4-2 shows an example of the `owey.*` hierarchy. The expanded path is as follows:

```
owey.hot_plug_charges.peripheral_5v.peripheral_5v_precharge.status
```

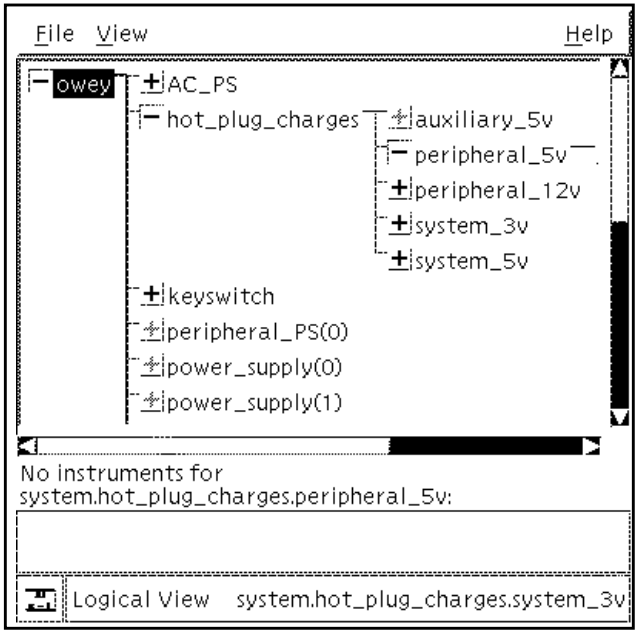

FIGURE 4-2 `ConfigReader` Data Hierarchy Example

3. **Replace** `owey` **with** `system`**.**

   The path is now:

```
system.hot_plug_charges.peripheral_5v.peripheral_5v_precharge.status
```

For information on the Config Reader hierarchy, see **Appendix B, "Config Reader**."

Kernel Reader

The Kernel Reader monitors system performance data such as disk queue length, forks, and CPU load. Rules written against the Kernel Reader data test Kernel Reader data. The Kernel Reader, like the Config Reader, provides data continuously to the Event Generator.

Kernel Reader data path names begin with `KernelReader.` CODE EXAMPLE 4-2 shows a partial display of RULE 100, which uses a `MULTI` attribute to define a Kernel Reader path. The complete rule is presented later in "Complex Rule Example 3: RULE 100" later in this chapter.

```
{
     RULE 100
     COMMENTS {
               This rule generates a YELLOW alarm if it finds any
               disk with an increasing wait queue while busy.
.
.
.
               }
.
.
.
MULTI { expr { [ hotlist KernelReader.disk.*.busy .busy ] } }
.
.
.
```

CODE EXAMPLE 4-2 Partial Rule 100 Display

The `MULTI` statement tells the Event Generator to repeat the same rule on each node that is returned from the `hotlist` function. The `hotlist` function finds the node whose name matches `KernelReader.disk.*.busy.` Since the strip string is defined as `.busy` (because the strip string is not null, it has to be in double quotes), the node name that is returned in the `node` variable by `hotlist` function contains only the matched node name, which excludes `.busy`. For example, if the `hotlist` function matches a node named `KernelReader.disk.sd8.busy`, the `$node` variable contains `KernelReader.disk.sd8.`

Determining the Path for the Kernel Reader

To write rules for the Kernel Reader, determine the Kernel Reader path.

▼ To Determine the Path for the Kernel Reader

1. **Double click the Kernel Data Catalog.**
2. **Expand the hierarchy by clicking the + signs in the hierarchy.**
   FIGURE 4-3 shows an example of the `owey.disk` hierarchy.
3. **Replace the machine name,** `owey`**, with** `KernelReader`**.**
   An example of a path is `KernelReader.disk.ssd93.runtime`.

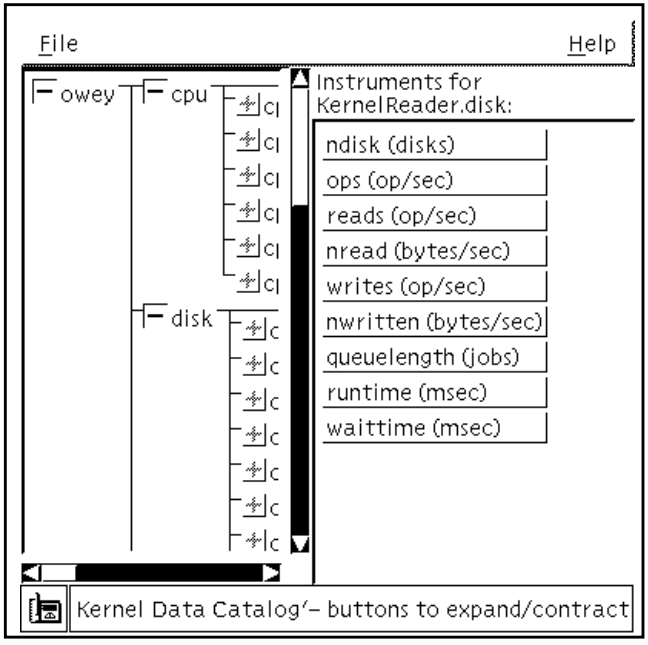

FIGURE 4-3 Example of the Kernel Reader Data Hierarchy

For information on the Kernel Reader hierarchy, see **Appendix A, "Kernel Reader**."

Log Scanner

Event rules can search the system log files for specific strings or regular expressions. The `LOG_RULES` attribute specifies the desired string. The Log Scanner parses the system log file and converts any matching message to a hierarchy that is similar to hierarchies generated by the Kernel Reader and Config Reader.

Rules written against the Log Scanner describe the following type of conditions and messages:

- Panic conditions (such as a system dying)
- Conditions that require immediate attention, such as a corrupted database
- Critical conditions such as hard disk errors
- Errors and warning messages
- Conditions that require special handling (these conditions are user-defined)
- Informational messages (such as hot plug information)
- Messages that contain information normally used when debugging a program

For more information on these conditions and warnings, see the `syslog(3)` man page.

Event Generator and Log Scanner Tasks

The Event Generator and the Log Scanner work together to perform the following tasks:

- The Event Generator reads the rules. When it finds the `LOG_RULES` statement, it places the rules on its hierarchy. The Log Scanner reads this hierarchy and performs actions.
- The Log Scanner goes through the system log files and uses the pattern supplied in `LOG_RULES` to find a match in the log file. If found, it constructs another hierarchy beginning with `LogScanner.`*<completepath>*, which is similar to the Config Reader and Kernel Reader hierarchies. By convention, the complete path is `LogScanner.LR`*<rule#>*.*<what is dynamically assigned by the Log Scanner>*. It obtains the complete path by reading this hierarchy as explained earlier. The `LS_MONITOR` statement in the `/etc/opt/SUNWsymon/log_scan.tcl` file identifies the file that contains the system log for the Log Scanner.
- The Event Generator executes the rules.

Partial Listing of RULE 10000 and Description

The following shows RULE 10000, an oversimplified rule that describes memory errors. This rule is oversimplified to present the following concepts:

```
{
      COMMENTS { memory error }
      RULE 10000
        LOG_RULES { { grep { Ecc error on board ([0-9]), reference number (J[0-9]) } syslog }
                        { load LR10000 "$logword(1)" "$logword(2)"} }
         MULTI { expr { [ findlist LogScanner.LR10000.* "" ] } }
      {
      set found 1
      }
```

The LOG_RULES statement tells the Log Scanner what message to look for, and what to do if the message is found in the syslog. In this example, the Log Scanner looks for a system message that matches the regular expression defined in a pair of brackets { } followed by in syslog.

If a system message that matches the regular expression is found, the load command (which is specific to the Log Scanner) is called to construct a hierarchy that begins with LogScanner. The load command has three arguments:

- Rule number LR10000
- $logword(1)
- $logword(2)

$logword(1) is substituted for the value that matches the first regular expression enclosed between parentheses ( ), in this case, the board number. $logword(2) is substituted for the value that matches the second regular expression enclosed between parentheses ( ), in this case, the reference number. For example, the following line in the syslog file: "Ecc error on board 5, reference number J6" generates the following hierarchy: LogScanner.LR10000.5.J6.

The Event Generator executes the findlist function in the MULTI statement searches and returns the list of hierarchies or nodes that match LogScanner.LR10000. The next statement assigns the value 1 to the found variable.

Guidelines for Using LOG_RULES

When using LOG_RULES follow these guidelines:

- The load function takes more than one argument to create a Log Scanner hierarchy that is checked as part of the condition of the rule.

- LOG_RULES are always MULTI rules. For example:

```
MULTI {expr {[findlist LogScanner.* ""]}}
```

- To get data loaded by the Log Scanner, use `findvalue $node`.
- To parse the logword data, use the `split` and `lindex` Tcl functions. The logword data is created by specifying subexpressions in the regular expression. These subexpressions are loaded by the `load` function on a Log Scanner property. Use the `split` and `lindex` Tcl functions in the Event Generator action to parse the string data on that Log Scanner property.

For a complete listing of a rule for the Log Scanner, see CODE EXAMPLE 4-7.

Event Generator

The Event Generator collects information from the Config Reader, Kernel Reader, and Log Scanner, and evaluates the data against its rules. It also maintains data about itself and the state of the server agents. This allows rules to be written against the status of agents (including itself) as well as data from the server. One example of a rule that is written against the Event Generator would be a warning of a high number of open events.

When a condition is true, the Event Generator logs an event and carries out the appropriate action in the rule. When the condition that generated the event is no longer true, the Event Generator may run a special action such as closing the event.

Getting Functions

The Event Generator gets functions from the `/opt/SUNWsymon/etc/event_gen.tcl` file. The `sm_confsymon -e` command copies the `/opt/SUNWsymon/etc/event_gen.tcl`file to the `/etc/opt/SUNWsymon/event_gen.servername.tcl` file, where `servername` is the server being monitored.

The `event_gen.servername.tcl` file defines variables and procedures used by the Event Generator. For example, the `psource` procedure is defined in `event_gen.servername.tcl` and is used in `rules.tcl`.

The Event Generator looks first in `/etc/opt/SUNWsymon` for rules files. If it cannot find the files, it then looks in `/opt/SUNWsymon`.

▼ To Customize Procedures and Commands

- **Edit the** `event_gen.servername.tcl` **file.**

  Editing the `event_gen.servername.tcl` enables you to include your own procedures and commands, such as adding the `psource` procedure. See CODE EXAMPLE 4-1 for information on `psource` in "Description of Rules Files" earlier in this chapter.

Caution – Do not modify defaults of procedures and commands unless you are extremely sure of your Tcl knowledge. Modifying the defaults incorrectly can cause massive errors or malfunctioning rules

Locating Rule Files

The Event Generator reads a file called `rules.tcl`. This file tells what files to read and where the rules files are located. For more information on this file, see "Description of Rules Files" earlier in this chapter.

The `event_gen.`*servername*`.tcl` file defines the path used to search for the rule files. By default, the paths are:

- `/etc/opt/SUNWsymon`
- `/opt/SUNWsymon/etc`
- `/opt/SUNWsymon/etc/lib`
- `/etc/opt/SUNWsymon/platform`
- `/opt/SUNWsymon/etc/platform`
- `/opt/SUNWsymon/etc/`*locale*

Verifying and Activating Rules

This section describes how to verify and activate rules. After you write or modify the event rule, you need to verify it then activate it.

Verifying New Event Rules

- **To check for the correct syntax, invoke the** `verify_rules` **command by typing**:

```
$ verify_rules [-I filename] [-R filename] [-o]
```

The command has three optional arguments (see TABLE 4-10.)

TABLE 4-10 Arguments to the `verify_rules` Command

Argument	Description
`-I`	Checks the file that contains supporting functions; the default is `event_gen.tcl`
`-R`	Checks the file that contains the `EVENTS` variable; the default is `./rules.tcl`
`-o`	Provides debugging information on rules to the standard error output

When you run `verify_rules` and the rules are correct, the software responds, "`GOOD RULES`." If a syntax error is detected, the program responds, "`BAD RULES`." However, this does not guarantee that the rule will work as expected.

Activating New or Modified Event Rules

You can change rules by stopping and restarting the Event Generator or sending a signal to the Event Generator process.

▼ To Activate New or Modified Event Rules

To Stop and Restart the Event Generator:

1. **Kill the Event Generator: by typing:**

```
% ps -ef | grep sm_egd
% kill pid
```

   a. **Type the *pid* number for the `kill` command that was returned from the `ps -ef | grep sm_egd` command.**

2. **Restart the Event Generator.**

   By sending a signal:

- **Activate the new or modified rule by typing:**

```
% kill -HUP pid
```

This causes the Event Generator to re-read and execute all rules files.

Debugging Tips

Use the following debugging tips to debug event rules:

- Run `verify_rules` to make sure the syntax is accurate. See “Verifying and Activating Rules” earlier in this chapter.
- If you have a problem with the rules, change the `rules.tcl` file to test only one rule at a time.
- Look for error messages in the event log file: `/var/opt/SUNWsymon/`*monitored_machine_name*`/event_log`.
- Use the `debug` function to do debugging. For information on the debug function see “debug” earlier in this chapter.

Troubleshooting

If you have problems with event rules, such as when the Event Generator fails to start properly, you may have to upgrade to Solstice SyMON 1.1 event rules. For information on troubleshooting event rules problems, see **Chapter 6, “Troubleshooting**.”

Rule Examples

This section describes rule examples. The rule description follows the Tcl code example.

Simple Rules

The sections that follow present Tcl code and a description of a simple rule.

Simple Rule Example 1: RULE 0

```
{
            RULE 0
            COMMENTS {
                    This rule generates an event if the symond
                    process on the server machine goes down,as
                    reported by symond on the Event Generator's
                    machine to the Event Generator.

                    symond_status is a Tcl variable provided by
                    the Event Generator,and is either "alive"
                    or "dead". It is changed when the Event
                    Generator gets a callback from symond.
                    }
            { expr { "$symond_status" == "dead" } }
            ON_OPEN { alarm RED "" "$r0mess" "" }
               set imsg [ format "$ir0msg" "$target" ]
                    snmp  "$imsg" }
            ON_CLOSE { end_alarm }
            SEVERITY 1
            PRIORITY 1
}
```

CODE EXAMPLE 4-3 RULE 0 Code Example

RULE 0 checks whether the value of variable `symond_status` is equal to `"dead."` If so, `ON_OPEN` does the following:

- Calls the `alarm` function with level `RED`, null node name, message `$r0mess`, which is defined in `rultext.tcl`, and a null Tcl function to the GUI (empty double quotes).
- Assigns the first formatted message `format "$ir0msg"`, which is defined in `rultext.tcl` and its target, `"$target"` to the `imsg` variable.
- Takes the string `imsg` as an argument and generates SNMP traps on every machine in the `snmp_host` variable, which is defined in `event_gen.tcl`. The string `imsg` is sent to Solstice Site Manager, Solstice Domain Manager, Solstice Enterprise Manager, or other SNMP-listening management platforms.

If the condition changes from True to False, `ON_CLOSE` calls the `end_alarm` function. For an explanation of `SEVERITY` and `PRIORITY`, see TABLE 4-1 earlier in this chapter.

Simple Rule Example 2: RULE 403

```
{
        RULE 403
        C { Test for meta CPU events }
        {
                set level [ check_cpa_cpu ]
                expr { "$level" == "BLUE" }
        }
        ON_OPEN { alarm BLUE KernelReader.cpu "$r403mess" ""
                  set imsg [ format "$ir403msg" "$target" ]
                  snmp  "$imsg" }
        ON_CLOSE { end_alarm }
        RATE 3600
}
```

CODE EXAMPLE 4-4 RULE 403 Code Example

RULE 403, which tests for meta CPU events, assigns the returned value of the `check_cpa_cpu` function to the `level` variable and checks whether the value of variable `level` is equal to `"BLUE."`

The `check_cpa_cpu` function checks the cpu cpa-related data that is collected from another rule, compares it with the predefined threshold and then returns a capacity panning level. If there is a capacity concern, the variable level returned is "`BLUE`." `ON_OPEN` does the following if the variable `level` is equal to `"BLUE`":

- Calls the `alarm` function with level `BLUE`, node name `KernelReader.cpu`, message `$r403mess`, which is defined in `rultext.tcl`, and a null Tcl function to the GUI (empty double quotes).
- Formats a message with `"$ir403msg"` (defined in `rultext.tcl`) as the base and `"$target"` as its first argument. It stores the formatted message in the variable `imsg`.
- Broadcasts an SNMP message, which is defined in the variable `imsg`, to machines defined in `snmp_host` variable (defined in `event_gen.tcl`). The string `imsg` is sent to Solstice Site Manager, Solstice Domain Manager, Solstice Enterprise Manager, or other SNMP-listening management platforms.

If the condition changes from True to False, `ON_CLOSE` calls the `end_alarm` function. The statement `RATE 3600` means that the rule is evaluated every 3600 seconds.

Complex Rules

This section contains examples and explanations of complex rules.

Complex Rule Example 1: RULE 102

```
{
     RULE 102
     COMMENTS {
           This generates a YELLOW event if 90% of the swap
           space is in use. Event stays open until swap space
           in use is less 80% of the swap space.
           }
   MULTI expr { [hotlist KernelReader.mem.swap_free ] }}
     {
          if { [getfield ACTIVE] } { set ts [ expr 0.20 * [ findvalue \
                                       KernelReader.mem.swap_total ] ] } \
                    else { set ts [ expr 0.10 * [ findvalue \
                                         KernelReader.mem.swap_total ] ] }
           expr { $value < $ts}
      }
      ON_OPEN { alarm YELLOW KernelReader.mem.swap_free\
           "$r102mess" "" }
           set imsg [ format "$ir102msg" "target" ]
           snmp "$imsg"  }
      ON_CLOSE { end_alarm }
      SEVERITY 3
      PRIORITY 3
}
```

CODE EXAMPLE 4-5 RULE 102 Code Example

The `MULTI` attribute checks the current value of free swap space by getting the value of `KernelReader.mem.swap_free` using the `hotlist` function.

There are two statements in the condition for this rule:

- The first statement checks whether this rule is currently active by getting the value of the `ACTIVE` field with the `getfield` function. If this rule is active, it gets the value of total swap space (`KernelReader.mem.swap_total`) using the `findvalue` function; it then calculates twenty percent of the total swap space and assigns the value to the `ts` variable. If the rule is not active, it gets the value of the total swap space as in the previous statement and then calculates ten percent of the total swap space and assigns the value to the `ts` variable.
- The second statement compares the value (`$value`) with the threshold and determines the current condition of this rule.

If the condition changes from False to True, `ON_OPEN` does the following:

- Calls the `alarm` function with level `YELLOW`, node name `KernelReader.mem.swap_free`, message `$r102mess` (defined in `rultext.tcl`), and no Tcl function to the GUI (empty quotes).

- Assigns the first formatted message `format "$ir0msg"` that is defined in `rultext.tcl` and its target, `"$target"` to the `imsg` variable.
- Takes the string `imsg` as an argument and generates SNMP traps on every machine in the `snmp_host` variable, which is defined in `event_gen.tcl`. The `imsg` string is sent to Solstice Site Manager, Solstice Domain Manager, Solstice Enterprise Manager, or other SNMP-listening management platforms.

If the condition changes from True to False, `ON_CLOSE` calls the `end_alarm` function.

For an explanation of `SEVERITY` and `PRIORITY`, see TABLE 4-1 earlier in this chapter.

The GUI highlights the node in the Physical and Logical Views, or the Kernel Data Catalog, because a node name is passed to the `alarm` function.

The sections that follow present Tcl code and descriptions of three complex rules.

Complex Rule Example 2: RULE 1201

```
{
   COMMENTS { for hot plug charge DC status }
   RULE 1201
   MULTI {
      expr { [ hotlist system.hot_plug_charges.*.status "" ] } }
   {
      set hpustatus  [ findstatus $node ]
      expr { ("$value" != "OK") && ("$hpustatus" == "alive") }
   }
   ON_OPEN {
      set mess [ format "$r1201mess" "$value" ]
      alarm RED $node "$mess" ""
       set imsg [ format "$ir1201msg" "$target" "$value" ]
       snmp   "$imsg"
      }
   ON_CLOSE { end_alarm }
   SEVERITY 2
   PRIORITY 2
}
```

CODE EXAMPLE 4-6 RULE 1201 Code Example

The MULTI label indicates that this rule might apply to more than one node. Each node found using the hotlist function matches the pattern system.hot_plug_charges.*.status. For example:

- system.hot_plug_charges.auxiliary_5v.status
- system.hot_plug_charges.peripheral_12v.status
- system.hot_plug_charges.peripheral_5v.status

Since the strip string is null (empty double quotes), the value assigned to the $node variable after the evaluation doesn't change.

The statements in the condition do the following:

- Obtains the status of the node variable and assigns it to the hpustatus variable.
- Checks if the value of the value variable is not equal to "OK" and the value of the variable hotplugstatus is equal to "alive".

If the condition changes from False to True, the following actions in ON_OPEN take place:

- Assigns the first formatted message $r1201mess (defined in rultext.tcl) to the mess variable with its argument, $value.
- Calls the alarm function with the RED level, the $node node name, the formatted $mess message, and no Tcl function to be executed (empty double quotes).
- Assigns the first formatted message format "$ir0msg", defined in rultext.tcl, and its target, "$target," with its $value argument to the imsg variable.
- Takes the string imsg as an argument and generates SNMP traps on every machine in the snmp_host variable, which is defined in event_gen.tcl. The imsg string is sent to Solstice Site Manager, Solstice Domain Manager, Solstice Enterprise Manager, or other SNMP-listening management platforms.

If the condition changes from True to False, ON_CLOSE calls the end_alarm function.

For an explanation of SEVERITY and PRIORITY, see TABLE 4-1 earlier in this chapter.

The process is repeated until all nodes are found using the findlist function in the MULTI statement.

Complex Rule Example 3: RULE 100

```
{
     RULE 100
     COMMENTS {
               This rule generates a YELLOW alarm if it finds any
               disk with an increasing wait queue while busy.

               YELLOW flag when disk is over 75% busy and average
               queue length is over 10 and increasing wait queue.
               Flag stays on till disk is not over 70 and average
               queue length is no longer than 8.

               This is a transitory event.
               }
     {
               set oldwait [ get_parameter]
               set projwait [ expr $oldwait*1.25 ]
               set diskwait [ findvalue $node.queuelength ]
               put_parameter $diskwait
              if { [getfield ACTIVE] } { set th1 0.7; set th2 8 } \
                              else      { set th1 0.75; set th2 10}
               expr { (($value>$th1)&&($diskwait>$th2)&& \
                       ($diskwait>$projwait)) }
     }
     ON_OPEN { alarm YELLOW $node.busy "$r100mess" ""
       set imsg [ format "$ir100msg" "$target" "$node" ]
       snmp  "$imsg" }
     MULTI { expr { [ hotlist KernelReader.disk.*.busy .busy ] } }
     ON_CLOSE {end_alarm }
     PARAMETERS { default 1000 }
     SEVERITY 3
     PRIORITY 2
}
```

CODE EXAMPLE 4-7 RULE 100 Code Example

The first part of the rule has seven statements:

- The first statement gets a user-defined value using the `get_parameter` function. The first time the rule is evaluated, the value is the default value defined after the `PARAMETERS` value. The `oldwait` variable (old wait time) stores the value.
- The second statement multiplies the old wait time, `$oldwait`, by 1.25 and stores it in the `projwait` variable (projected wait time).
- The third statement gets the current disk wait time. It obtains the value of the `queuelength` field of the node that is currently being evaluated. It stores the value in the `diskwait` variable.

- The fourth statement saves the current disk wait time, `$diskwait`, using the `put_parameter` function.
- The fifth and sixth statements check if the rule is currently active (if the condition is currently True) and assign the threshold values to the `th1` and `th2` variables. They compare the amount of time the node is busy (`$diskbusy`) with the `th1` threshold. Also, they compare the wait time of the node (`$diskwait`) with the `th2` threshold.

If the condition changes from False to True, `ON_OPEN` does the following:

- Calls the `alarm` function with the level `YELLOW`, the node name with its property name (`$node.busy`), event message (`r100mess`), which is defined in `rultext.tcl` and no Tcl function for the GUI (empty double quotes).
- Assigns the first formatted message `format "$ir000msg"`, defined in `rultext.tcl` and its target, `"$target,"` to the `imsg` variable.
- Takes the `imsg` string as an argument and generates SNMP traps on every machine in the `snmp_host` variable, which is defined in `event_gen.tcl`. The string `imsg` is sent to Solstice Site Manager, Solstice Domain Manager, Solstice Enterprise Manager, or other SNMP-listening management platforms.

The `MULTI` statement tells the Event Generator to repeat the same rule on each node that is returned from the `hotlist` function. The `hotlist` function finds the node whose name matches `KernelReader.disk.*.busy.busy`. Since the strip string is defined as `.busy`, (because the strip string is not null, it has to be in double quotes), the node name that is returned in the `node` variable by the `hotlist` function contains only the matched node name excluding `.busy`. For example, if the `hotlist` function matches a node named `KernelReader.disk.sd8.busy.busy`, the `$node` variable contains `KernelReader.disk.sd8`.

If the condition changes from True to False, `ON_CLOSE` calls the `end_alarm` function.

The `PARAMETERS` statement sets a default value to 1000. Thus, the condition will not be True the first time this rule is evaluated.

For an explanation of `SEVERITY` and `PRIORITY`, see TABLE 4-1 earlier in this chapter.

Complex Rule Example 4: RULE 106

```
{
      COMMENTS { no swap space }
      RULE 106
      RATE 300
      LOG_RULES { {grep { no swap space.*pid ([0-9]+)} syslog }
                    { load LR106 "$logword(1)" } }
      MULTI { expr { [ hotlist LogScanner.matches.LR106.* "" ] } }
      {
                  set found 1
   }
      ON_OPEN {
               set lsdata       [ split $value ]
               set pid           [ lindex $lsdata 0 ]
               set mess [ format "$r106mess" "$pid" ]
               alarm YELLOW "" "$mess" ""
               end_alarm
                set imsg [ format "$ir106msg" "$target" "$pid" ]
                snmp "$imsg"
               }
      SEVERITY 2
      PRIORITY 2
}
```

CODE EXAMPLE 4-8 Rule 106 Code Example

The `LOG_RULES` statement tells the Log Scanner what message to look for, and what to do if the message is found in the `syslog`. In this example, the Log Scanner looks for a system message that matches the regular expression, which is defined in a pair of brackets { } followed by `in syslog`.

If a system message that matches the regular expression is found, the Log Scanner creates a hierarchy named `LogScanner.LR106.*` The value of `LogScanner.matches.LR106.*` contains arguments following the `load` function. The element, `$logword(1),` contains the string that is matched in the regular expression defined between parentheses. You count the pairs of parentheses from left to right. In this example, it is the first pair of parentheses whose matched string is stored in `$logword` with an index of 1, `$logword(1)`. If there were a second pair of parentheses, you would find the matched information between the second pair of parentheses to the second string.

The `hotlist` function in the `MULTI` statement searches and returns the list of hierarchies or nodes created by the Log Scanner whose name can be matched with `LogScanner.matches.LR106.*` The condition has one statement, which assigns the value `1` to the `found` variable.

Next, the following seven actions take place in `ON_OPEN`.

- Splits the list into elements and stores them in the `lsdata` variable. For more information on the `split` function, refer to the Tcl books listed in the Preface.
- Obtains the process ID number from the first list element (index 0) and stores it in the `pid` variable.
- Formats the event message and stores it in the `mess` variable. This is done by passing the message body `$r106mess` and the process ID number `$pid`. The `$r106mess` variable is stored in `rultext.tcl`.
- Calls the `alarm` function with `YELLOW` level, no node name, formatted message `$mess,` and no Tcl function for the GUI (empty double quotes).
- Calls `end_alarm` to close the event.
- Assigns the first formatted message `format "$ir106msg"`, stored in `rultext.tcl,` and its target `$target` plus `$pid` to the `imsg` variable.
- Takes the string `imsg` as an argument and generates SNMP traps on every machine in the `snmp_host` variable, which is defined in `event_gen.tcl`. The string `imsg` is sent to Solstice Site Manager, Solstice Domain Manager, Solstice Enterprise Manager, or other SNMP-listening management platforms.

For an explanation of `SEVERITY` and `PRIORITY`, see TABLE 4-1 earlier in this chapter.

CHAPTER 5

Command Line Options

Use the Solstice SyMON options described in this chapter on the command line when you invoke Solstice SyMON. For example, the `-t` option refers to the target system to monitor:

```
% /opt/SUNWsymon/bin/symon -t targetsystemname &
```

Solstice SyMON also recognizes the standard X Window System toolkit options.

Options to Solstice SyMON

The following entries describe the Solstice SyMON command line options in man page type format.

dragthreshold

Name: `-dragthreshold`

Abbreviated Name: `None`

Default: `10`

Resource: `*dragthreshold`

Description: This is the mouse drag threshold (pixels) for SysMeters.

flashDuration

Name: `-flashDuration`

Abbreviated Name: `-fd`

Default: 30

Resource: `*flashDuration`

Description: Controls the duration (msec) of the flashes of the `System:` button. The `System:` button is the button in the upper left corner of the Launcher window, which tells the server being monitored. Once Solstice SyMON has established communication with the server, this button flashes according to the `-fd` and `-fi` options until communication with the server ceases.TABLE 5-1 describes the action for flash duration settings.

TABLE 5-1 Flash Duration Settings

Flash Duration Setting	Action
zero (0)	Turns off the feature
Less than 0	Error message is printed and flash duration is set to 0
Greater than the flash interval	Flash duration is set to the flash interval and an error message is printed to the standard error
Equal to the flash interval	Turns the system button green constantly; the button may flicker as it is updated at each interval

See Also: `flashInterval`

flashInterval

Name: -flashInterval

Abbreviated Name: -fi

Default: 10000

Resource: *flashInterval

Description: Controls the frequency of flashing (msec) of the System: button. The System: button is the button in the upper left corner of the Launcher window, which gives the name of the server being monitored. Once Solstice SyMON has established communication with the server, this button flashes according to the -fd and -fi options until communication is terminated.

If the interval is less than 1000 msec, an error message is printed to the standard error and the flash interval is set to 1000. Setting the flash interval and flash duration equally, turns the system button green constantly. The button may flicker as it is updated at each interval.

After each interval, Solstice SyMON executes a RPC ping of the target system. With the default flash interval and flash duration, each flash indicates a successful ping.

See Also: flashDuration

heartbeatInterval

Name: -heartbeatInterval

Abbreviated Name: -hi

Default: 10

Resource: *heartbeatInterval

Description: The heartbeat interval (seconds) is the interval between successive updates of the message in the Launcher footer, which states the amount of time remaining until the next data update. For example: "Next update request in 10 seconds."

help

Name: `-help`

Abbreviated Name: `-h`

Default: None

Resource: None

Description: Prints a list of options with default values and exits.

installDir

Name: `-installDir`

Abbreviated Name: `-I`

Default: `/opt/SUNWsymon`

Resource: `*installDir`

Description: Sets the root directory to examine Tcl files (default is `/opt/SUNWsymon`)

interval

Name: `-interval`

Abbreviated Name: `-i`

Default: 10

Resource: `*interval`

Description: Polling interval (in seconds). This is the time between requests for data from agents. The minimum polling interval is 5 seconds. The maximum polling interval is 120 seconds.

To change the polling interval while Solstice SyMON is running, use the Launcher Options menu. From the Options menu of the Launcher, select Data Controls. From the Data Controls menu, set the polling interval.

Solstice SyMON polls the agents for data every `interval` seconds. It does all GUI updates necessary, then waits for at least `minWait`, and possibly until `interval` seconds from the start of the last update (whichever is greater) before polling again.

See Also: `minWait`

minWait

Name: `-minWait`

Abbreviated Name: `-mw`

Default: 1

Resource: `*minWait`

Description: Minimum wait (in seconds) between polls or updates to guarantee user interaction time. This specifies how long to wait after processing one data update before asking the target system for another update. This creates a time for you to reduce the polling frequency.

Solstice SyMON polls the agents for data every `interval` seconds. It does all GUI updates necessary, then waits for at least `minWait`, and possibly until `interval` seconds from the start of the last update (whichever is greater) before polling again.

S**ee Also**: `interval`

pruneTime

Name: `-pruneTime`

Abbreviated Name: `-pt`

Default: 120

Resource: `*pruneTime`

Description: Length of time (in minutes) for the Kernel Reader data to be retained in the GUI. For example, if the prune time is 180 minutes, Solstice SyMON deletes data older than 180 minutes.

rpcNum

Name: `-rpcNum`

Abbreviated Name: `-rn` *or* `-n`

Default: 100244

Resource: `*rpcNum`

Description: Enables you to assign a custom RPC number to Solstice SyMON. All client applications that want to connect to Solstice SyMON must know this custom RPC number. The default Solstice SyMON RPC number of 100244 is usually supplied to client applications. If you change this RPC number, you must provide the new RPC number to all client applications.

The RPC number you provide should be the RPC number for Solstice SyMON running on the monitored host from which all clients retrieve connectivity information (not the RPC number of the spawning Solstice SyMON on a remote system with an Event Generator).

session

Name: `-session`

Abbreviated Name: `none`

Default:“”

Resource: `*session`

Description: The session option specifies a file to read when Solstice SyMON starts up. This file is a Tcl file in the same format as Tcl files that are saved in the System Meter, Kernel Data Catalog, or Logical View consoles. It defines the layout and contents of a Solstice SyMON instance.

This option is used by the CDE session manager to restore previously saved layouts. Not all windows are currently saved or restored—only the Launcher and System Meters. When you invoke Solstice SyMON at a later date, your previous environment is restored.

target

Name: `-target`

Abbreviated Name: `-t`

Default: `target_system_name`

Resource: `*target`

Description: Name of the server to be monitored.

tempPruneTime

Name: `-tempPruneTime`

Abbreviated Name: `-tpt`

Default: 1440

Resource: `*tempPruneTime`

Description: Length of time (in minutes) for Config Reader data (such as board temperature) to be retained by the GUI. For example, if the `tpt` is 180 minutes, Solstice SyMON deletes data more than three hours old.

vtsui

Na**me**: `-vtsui`

Abbreviated Name: None

Default: vtsui.online

Resource: `*vtsui`

Description: Path of SunVTS user interface binary.

CHAPTER 6

Troubleshooting

This chapter presents troubleshooting tips when using Solstice SyMON.

Troubleshooting Tips

TABLE 6-1 presents troubleshooting problems and the suggested corrective actions.

TABLE 6-1 Troubleshooting Tips

Problem	System Display	Corrective Action
Event Generator fails to start properly	"We need the TCL variable Rules to proceed" is displayed in the Event Log file	The upgraded Event Generator is running with an older rules file. Upgrade the event rules by installing the new rules you obtained from the CD-ROM (see Chapter 2).
All agents are displayed in red on the Solstice SyMON console and there are no daemons running on your server	Agents are displayed in red	You may have installed Solstice SyMON agents on an unsupported platform such as an Ultra Enterprise 1 system. Install the Solstice SyMON agents on one of the supported platforms listed in Chapter 1.
Agents are displayed in red on the Solstice SyMON console or no daemons are running on your server	Agents are displayed in red	Check /var/adm/messages for Solstice SyMON-related error messages.

TABLE 6-1 Troubleshooting Tips *(Continued)*

Problem	System Display	Corrective Action
Solstice SyMON agents are running but the GUI or Event Generator cannot connect to the agents	Event Generator messages will be written to: `/var/opt/SUNWsymon/`*host*`/EG/eg_debug.`<*pid*>	Set the debug flag to 4 to trace connectivity; for example: `symon -aildebug 4` or in `sm_symond.conf`, use: `sm_egd -D 4` in addition to other arguments.
You are experiencing problems with the Event Generator	(See Corrective Action)	Check for error messages in the Event Generator log file either by using the Log file Viewer or examining this file: `/var/opt/SUNWsymon/`<*host*>`/EG/event_log`
You are experiencing difficulties with the log file scanner	(See Corrective Action)	Check for error messages in the Log file scanner log file either by using the Log file Viewer or by examining this file: `/var/opt/SUNWsymon/`<*host*>`/LS/ls_log`
Solstice SyMON requires CDE 1.0.1 and later	"`ld.so.1: ./symon: fatal: relocation error: symbol not found: braelist:` referenced in `/usr/dt/lib/libDtHelp.so.1`"	Upgrade your CDE installation to CDE version 1.01 or later; refer to your CDE documentation.
Your system is too heavily loaded	"`do_accept : too many open files`" appears in the `/var/adm/messages` file	Close some of the GUIs polling the system.
The Event Generator is saturated with polling requests	"`get_all_event_string: data not ASCII type =`"	Kill and restart the Event Generator; refer to "To Activate New or Modified Event Rules" in Chapter 4.
Solstice SyMON is installed improperly	A message similar to "`ld.so.1: /opt/SUNWsymon/sbin/sm_symond: fatal: libaip.so.1: can't open file: errno=2`" appears on the system console or the command terminal	Remove the `SUNWsyrt` package for an older release of Solstice SyMON. Install the `SUNWsyrt` package for this release of Solstice SyMON (see Chapter 2).
Solstice SyMON is installed improperly	When starting the SyMON GUI, a message like "`ld.so.1: symon: fatal: relocation error: symbol not found: called_on_incoming_fd_data: referenced in /opt/SUNWsymon/lib/libaip.s`" appears	Remove the `SUNWsyu` package for an older Solstice SyMON release and install the `SUNWsyu` package for this version of Solstice SyMON (see Chapter 2).

TABLE 6-1 Troubleshooting Tips *(Continued)*

Problem	System Display	Corrective Action
Solstice SyMON is installed improperly	When starting the SyMON agents or Event Generator, a message such as "`ld.so.1: /opt/SUNWsymon/sbin/sm_symond: fatal: relocation error: symbol not found: called_on_incoming_fd_data referenced in /opt/SUNWsymon/lib/libaip.so`" appears on the system console, or the command terminal in `/var/adm/messages`	Remove the `SUNWsys` or `SUNWsye` package for an older Solstice SyMON release and install the `SUNWsys` or `SUNWsye` packages, respectively for this release of Solstice SyMON (see Chapter 2).

TABLE 6-1 Troubleshooting Tips *(Continued)*

Problem	System Display	Corrective Action
A Solstice SyMON client not running this version of Solstice SyMON is attempting to monitor a host running this version of Solstice SyMON	When starting the SyMON GUI, a message appear such as "`symon: _AIP_request_V14: RPC: Program/version mismatch; low version = XX, high version = XX`" where *XX* is a number greater than 14	Upgrade the client to this release of Solstice SyMON (see Chapter 2).
The SyMON GUI shows the Event Generator in red; this means the Event Viewer is not accessible	On the agent host, a message in `/var/adm/messages like ... /opt/SUNWsymon/sbin/sm_symond[...]: Version mismatch. Symond on` *<Event Generator host>* `is not version 1.4` or on the Event Generator host, a message such as `..../opt/SUNWsymon/sbin/sm_egd[...]: Got an error: <AIL error in /opt/SUNWsymon/sbin/sm_egd: Version mismatch. Symond does not use AIP version 14>`	Upgrade the Event Generator host to this release of Solstice SyMON (see Chapter 2).
The SyMON GUI shows the Event Generator in red; this means the Event Viewer is not accessible; the version of Solstice SyMON on the agent host is earlier than the current version	In `/var/adm/messages` a message such as "`... /opt/SUNWsymon/sbin/sm_symond[...]: AIL error in /opt/SUNWsymon/sbin/sm_symond: Version mismatch. Symond does not use AIP version 14` or on the Event Generator host, a message in `/var/adm/messages like ... /opt/SUNWsymon/sbin/sm_egd[...]: Got an error: <AIL error in /opt/SUNWsymon/sbin/sm_egd: Version mismatch. Symond on` *<agent host>* `is not version 1.4>`	Upgrade the agent host to this Solstice SyMON release (see Chapter 2).

APPENDIX **A**

Kernel Reader

This appendix provides a summary of the Kernel Reader data hierarchy.

Kernel Reader Data Hierarchy

The Kernel Reader is one of the three system agents. Use the information in this appendix if you want to write event rules that monitor the output of the Kernel Reader.

TABLE A-1 lists the Kernel Reader hierarchy.

The first column contains the name of the node or property.

The second column specifies whether the parameter is a node (N) or a property (P). A node in the hierarchy is a starting point for a branch of the hierarchy. All properties that belong to a node appear below the node and are marked with a P. For example, `cpu.busy` and `cpu.user` fall under the `cpu` node. The node and its properties are grouped together.

The third column lists the value type. The Kernel Reader generates five types of values:

- (I) Instant value at the sample time
- (S) Summary over the last sample interval
- (A) Average value over last sample interval
- (%) Percentage value (100.0 == 100%) over the last sample interval
- (R) Rate value (per second) over the last sample interval

The fourth column describes the node or property.

TABLE A-1 Kernel Reader Data Hierarchy

Name of Node or Property	Node or Property	Value Type	Description
KernelReader	N		Kernel Reader root
control	N		Control information
control.pid	P	I	Process ID for the Kernel Reader
cpu	N		CPU usage over all CPUs
cpu.busy	P	%	CPU busy time (user + system)
cpu.user	P	%	User time
cpu.sys	P	%	System time
cpu.wait	P	%	Wait time
cpu.idle	P	%	Idle time
cpu.ncpu	P	I	Number of CPUs
cpu.context_switch	P	R	Context switches rate
cpu.interrupt	P	R	Interrupts rate
cpu.syscalls	P	R	System call rate
cpu.mutex	P	R	Mutex rate
cpu.forks	P	R	Fork call rate
cpu.execs	P	R	Exec call rate
cpu.swapout	P	R	Number of swapouts per second
cpu.swapin	P	R	Number of swapins per second
cpu.pgout	P	R	Number of page-outs per second
cpu.pgin	P	R	Number of page-ins per second
cpu.pgpgout	P	R	Number of pages paged out per second
cpu.pgpgin	P	R	Number of pages paged in per second
cpu.run_queue_length	P	A	Number of jobs waiting to be run
cpu.swap_queue_length	P	A	Number of jobs waiting for page swap
cpu.io_queue_length	P	A	Number of jobs waiting for I/O
cpu.cpu*X*	N		Individual CPU data (*X*=0, 1, 2...)
cpu.cpu*X*.instance	P	I	CPU instance (the *X* in the label)
cpu.cpu*X*.busy	P	%	CPU busy time (user + system)

TABLE A-1 Kernel Reader Data Hierarchy *(Continued)*

Name of Node or Property	Node or Property	Value Type	Description
cpu.cpu*X*.user	P	%	User time (100.00 == 100%)
cpu.cpu*X*.sys	P	%	System time
cpu.cpu*X*.wait	P	%	Wait time
cpu.cpu*X*.idle	P	%	Idle time)
cpu.cpu*X*.context_switch	P	R	Context switches rate
cpu.cpu*X*.interrupt	P	R	Interrupts rate
cpu.cpu*X*.syscalls	P	R	System call rate
cpu.cpu*X*.mutex	P	R	Mutex rate
cpu.cpu*X*.forks	P	R	Fork call rate
cpu.cpu*X*.execs	P	R	Exec call rate
cpu.cpu*X*.swapouts	P	R	Number of swapouts per second
cpu.cpu*X*.swapins	P	R	Number of swapins per second
cpu.cpu*X*.pgpgout	P	R	Number of pages paged out per second
cpu.cpu*X*.pgpgin	P	R	Number of pages paged in per second
cpu.cpu*X*.pgout	P	R	Number of page-outs per second
cpu.cpu*X*.pgin	P	R	Number of page-ins per second
mem	N		Physical memory
mem.mem_avail	P	I	Total memory available (physmem) (Mbytes)
mem.mem_inuse	P	A	Physical memory in use (Mbytes)
mem.mem_inuse_p	P	%	Physical memory in use
mem.mem_free	P	A	Physical memory free (Mbytes)
mem.mem_free_p	P	%	Physical memory free (%)
mem.swap_avail	P	I	Swap space available (Mbytes)
mem.swap_resv	P	I	Swap space reserved (Mbytes)
mem.swap_alloc	P	I	Swap space allocated (Mbytes)
mem.swap_used	P	I	Swap space used (Mbytes)
disk	N		For disk I/O, summary on all disks
disk.ndisk	P	I	Number of disks
disk.ops	P	R	Average disk operations (r+w) rate (op/s)
disk.reads	P	R	Average disk read rate (op/s)

TABLE A-1 Kernel Reader Data Hierarchy *(Continued)*

Name of Node or Property	Node or Property	Value Type	Description
`disk.write`	P	R	Average disk write rate (op/s)
`disk.nread`	P	R	Number of bytes read (bytes/s)
`disk.nwritten`	P	R	Number of bytes written (bytes/s)
`disk.queuelength`	P	A	Average disk queue length
`disk.waittime`	P	A	Average wait time (ms)
`disk.runtime`	P	A	Average run time
`disk.`*X*	N		Individual disks (disk name as label, such as sd3)
`disk.`*X*`.reads`	P	R	Average disk read rate (op/s)
`disk.`*X*`.write`	P	R	Average disk write rate (op/s)
`disk.`*X*`.nread`	P	R	Number of bytes read (Kbytes/s)
`disk.`*X*`.nwritten`	P	R	Number of bytes written (Kbytes/s)
`disk.`*X*`.busy`	P	%	Disk busy
`disk.`*X*`.svctime`	P	A	Average service time (ms)
`disk.`*X*`.waittime`	P	A	Average wait time (ms)
`disk.`*X*`.runtime`	P	A	Average run time (ms)
`disk.`*X*`.queuelength`	P	A	Average queue length
`net`	N		Network-related values, all interfaces
`net.nnet`	P	I	Number of net interface
`net.oflo`	P	R	Overflow error per second
`net.uflo`	P	R	Underflow error per second
`net.crc`	P	R	CRC error per second
`net.framming`	P	R	Frame error per second
`net.collisions`	P	R	Collisions per second
`net.oerrors`	P	R	Output errors per second
`net.ierrors`	P	R	Input errors per second
`net.opackets`	P	R	Output packet count (packets/second)
`net.ipackets`	P	R	Input packet count (packets/second)
`net.`*X*	N		Each net interface, name as label (for example `le0`)
`net.`*X*`.oflo`	P	R	Overflow error per second
`net.`*X*`.uflo`	P	R	Underflow error per second

TABLE A-1 Kernel Reader Data Hierarchy *(Continued)*

Name of Node or Property	Node or Property	Value Type	Description
net.*X*.crc	P	R	CRC error per second
net.*X*.framming	P	R	Frame error per second
net.*X*.collisions	P	R	Collisions per second
net.*X*.oerrors	P	R	Output error rate
net.*X*.ierrors	P	R	Input error rate
net.*X*.opackets	P	R	Output packet count (packets/second)
net.*X*.ipackets	P	R	Input packet count (packets/second)

If any component of a hierarchy path contains a period (.), you may not be able to use the `findlist` function to find it. For example, if you search for the mount point `/export/a0.test` reported by the Config Reader via the path: `system.slot.board.io-unit.sbi.dma.esp.sd./export/a0.test` the node will not be found. Instead, use: `system.slot.board.io-unit.sbi.dma.esp.sd.*` to find this node.

APPENDIX **B**

Config Reader

The Config Reader determines the configuration of a machine, along with the status of its components. Its output is depicted in these consoles:

- Physical View
- Logical View

Tables B-2 through B-4 list the Config Reader data hierarchy. Use the information in this appendix if you want to write event rules that monitor the output of the Config Reader.

Terminology

The terms in TABLE B-1 are used to explain the information in TABLE B-2.

TABLE B-1 Definitions of Common Config Reader Terms

Property	Term	Definition
P	`instance`	This is a decimal or hexadecimal number acquired by reading `/etc/path_to_inst`, which maps physical device names to the instance number; refer to the man page for more information.

TABLE B-1 Definitions of Common Config Reader Terms

Property	Term	Definition
P	`upa-portid`	This property exists for SBus and CPU units. Each board has two CPU slots, A and B, which have unique numbers. For example, board 0 has `upa-portid` 0 and 1, and board 1 has `upa-portid` 2 and 3, and so on. `portid`/2 = slot number.
P	`upa-mid`	This property is displayed for `fhc`, `sbus`, `cpu-unit` and central modes. It is always equal to `upa-portid`.
P	`hpu`	This stands for hot plugable unit. You can replace hot plugable units without powering the system down.

Data Hierarchy

These abbreviations are used in the following tables in this chapter:

- I = Integer
- H = Hexadecimal
- A = ASCII

Table B-2 lists the Config Reader data hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 systems that describe the configuration of the host. They can also be electrical entities. For example, the `system_3v node` is not a physical entity but is the input voltage to system boards.

Each node has a set of properties that describes its state; for example, if an LED is on or off, or if a voltage level is acceptable. The node and its properties are grouped together in TABLE B-2 in the following order:

- The first column contains the node or property name.
- The second column specifies whether the parameter is a node (N) or a property (P). A node in the hierarchy is a starting point for a branch of the hierarchy. All properties that belong to a node appear below the node and are marked with a P. For example, `cpu.busy` and `cpu.user` fall under the `cpu` node. The node and its properties are grouped together.
- The third column contains the data type of the property. Note that the Config Reader always encloses the value of a property between double quotes (“ ”).
- The fourth column lists the default value, which is not listed for every entry.
- The last column explains the meaning of the node or property.

Ultra Enterprise 3000, 4000, 5000, and 6000

The following table is the Config Reader data hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 server systems.

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems

Node or Property and Name	Node or Property	Type	Default Value	Description
AC_PS	N			
status	P	"A"	"OK"/"FAIL"	AC connector and switch
SUNW,fas	N			Host Bus Adapter driver
tape_count	P	"I"		Number of tapes attached to a particular slot
disk_count	P	"I"		Number of disks attached to a particular slot
instance*	P	"I"		
device_type*	P	"A"	"scsi"	
SUNW,ffb(N)	N			UPA color frame buffer and graphics accelerator (N) = instance number
buffer	P	"A"	"Single"/ "Double"	
3DRAM	P	"A"		3-D RAM model and version number
DAC	P	"A"		Revision of DAC chip
FBC_version	P	"H "		
revision	P	"A"		Revision of FFB board
instance*	P	"I"		
upa-portid*	P	"I"		
width	P	"I"		Pixels
height	P	"I"		Pixels
device_type	P	"A"	"display"	
model	P	"A"		
SUNW,hme	N			Fast Ethernet device driver

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
instance*	P	"I"		
device_type	P	"A"	"network"	
SUNW,leo	N			Double-buffered SBus color frame buffer and graphics accelerator
instance*	P	"I"		
width	P	"I"		Pixels
height	P	"I"		Pixels
model	P	"A"		
device_type	P	"A"	"display"	
SUNW,pln(N)	N			SPARCstorage Array
disk_count	P	"I"		Number of attached disks
device_type	P	"A"		SPARCstorage Array
instance*	P	"I"		
SUNW,rtvc	N			Real-time video capture
instance*	P	"I"		
model	P			
SUNW,soc(N)*	N			Serial Optical Controller device driver; (N) = instance number
device_type	P	"A"	See Description	Serial Optical Channel Processor Host Adapter
instance*	P	"I"		
model	P			
soc-wwn	P	"I"		World Wide Web device name
port-wwns	P	"I"		
board(N)	N			(N) = board number
hpu	P	"A"	"yes"	Hot plug unit
temperature	P	"A"		"*N* C< " where *N* is an integer

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
`state=`	P	"A"		• "active:" Normal state for boards after the operating system has booted • "disabled:" Disabled by the PROM because it was listed in the 'disabled-board-list' property in the options node • "failed:" A board that is failed by POST and powered off in the testing phase of PROM operations • "hot-plug:" Intermediate state after a board is hotplugged • "low-power:" The state that hotplug boards are in before they are removed • "unknown:" Unused
`fru`	P	"A"	`"yes"`	Field replaceable unit
`type`	P	"A"	See Description	One of: `clock`, `cpu/memory`, `Dual SBus I/O`, `disk-board`, `SBus/FFB I/O`, `pci I/O`
`board-num`	P	"I"		For example, Ultra-Enterprise 3000: up to 4
`memory_size`	P	"A"		"*N* MB," *N* is an integer (CPU/memory boards only)
cdrom	N			
`mounted_partitions`	P	"I"		
`device_type`	P	"A"		"CD-ROM"
`name`	P	"A"		For example `c0t6d0`
`instance*`	P	"I"		
`central`	N			Only exists under the clock board
`instance*`	P	"I"		
`upa-mid*`	P	"I"		
`clock-board`	N			

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
instance*	P	"I"		
cpu-unit(*N*)	N			(*N*) = unit
fru	P	"A"	"yes"	Field replaceable unit
clock-frequency	P	"A"		*N* MHz, where *N* is an integer
cpu-type	P	"A"		For example sparc
status	P	"A"	"online"/ "offline"	
model	P	"A"		
processor-id	P	"I"		2*board number (+1 if cpu-unit=B)
unit	P	"A"		"A" or "B" (each board takes two CPUs)
upa-mid*	P	"I"		
upa-portid*	P	"I"		
icache-size*	P	"A"		"*N* KB" or "*N* MB," where *N* is a fixed point number
dcache-size	P	"A"		"*N* KB" or "*N* MB," where *N* is a fixed point number
ecache-size	P	"A"		"*N* KB" or "*N* MB," where *N* is a fixed point number
device_type	P	"A"	"cpu"	
board#	P	"A"		Board number on which this node resides
disk_fan	N			Exists only in Ultra Enterprise 3000 systems
hpu	P	"A"	"yes"	Hotplug unit?
status	P	"A"	"OK"/"FAIL"	
fan	N			In the Ultra Enterprise 4000, 5000, and 6000, the AC_PS has a fan node
status	P	"A"	"OK"/"FAIL"	

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
fhc	N			Fire Hose controller; one associated with each AC; contains board CSR, which controls on-board devices such as PROMs, RAM and UARTs
instance*	P	"I"		
upa-mid*	P	"I"		
version#	P	"I"		Version number
board#	P	"I"		Board number on which this node resides
model	P	"A"		Address controller
ac	N			
instance*	P	"I"		
version#	P	"I"		
model	P	"A"		
device_type	P	"A"	"memory-controller"	
eeprom	N			Electrically programmable read-only memory
model	P	"A"		
environment	N			
instance*	P	"I"		
flashprom	N			
instance*	P	"I"		
model	P	"A"		
sbus-speed	P			
keyswitch				
position	P	"A"		"Normal Mode," "Secure mode," or "Diagnostic mode"

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
led(N)	N			Power supplies have two LEDs: green (off = no power) and amber (on=FAIL); boards have three LEDs; refer to the Ultra Enterprise Maintenance Manual for LED combinations
led-num	P	"I"		1, 2, or 3
state	P	"A"	"on"/"off"	
peripheral_PS(N)	N			184w power supply for SCSI peripherals, additional power for boards in system and pre-charge power for hot plugging boards
hpu	P	"A"	"yes"	Hot plug unit
status	P	"A"	"OK"/"FAIL"	
unit#	P	"I"		Unique number of this power supply
fru	P	"A"	"yes"	Field replaceable unit
nf(N)	N			SBus FDDI Driver
ether	P	"A"		Ethernet address; for example 8:0:20:75:af:f8
inet	P	"I"		Internet address; for example 129.146.65.45
symbol	P	"A"		Network name corresponding to the inet address above
name	P	"A"		Driver name; for example, nf0
instance*	P	"I"		
model	P	"A"		
device_type	P	"A"	"network"	
power_supply(*N*)	N			300w board supply, (*N*) = unit number
hpu	P	"A"	"yes"	Hot plug unit
status	P	"A"	"OK"/"FAIL"	

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
unit#	P			Unique number of the power supply
fru	P	"A"	"yes"	Field replaceable unit
sbus(N)	N			Peripheral bus; N=instance number
instance*	P	"I"		
upa-portid*	P	"I"		
upa-mid*	P	"I"		
version#*	P			
board#	P	"A"		Board number on which this node resides
device_type*	P	"A"	"sbus"	
model	P			
network_count	P	"I"		Network devices attached to this SBus
remote_console	N			
status=	P	"A"	"Enabled/ Disabled"	
sd(0)	N			
fru	P	"A"	"yes"	Field replaceable unit
mounted_partitions	P	"I"		
device_type	P	"A"		
name	P	"A"		"disk" for example, "c0t0d0"
instance	P	"I"		
The following data, which is a continuation of the properties that go with the sd(0) node (above), is the output of Config Reader if the disk has mounted partition.				
name	P	"A"		For example, c0t0d0s0
percent_used	P	"I"		For example, 49%
total_bytes	P	"I"		*n* MBytes or *n* KBytes
avail_bytes	P	"I"		*n* MBytes or *n* KBytes

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
used_bytes	P	"I"		*n* MBytes or *n* KBytes
device_type*	P	"A"		partition
export name	P	"A"		For example, c0t0d0s7
simm(0)	N			
board_reference_number	P	"A"		
slot	P	"I"		CPU / memory slot where this SIMM resides
size	P	"A"		
type	P	"A"		dram
fru	P	"A"	"yes"	Field replaceable unit
sram	N			Static RAM
instance*	P	"I"		
st(*N*)	N			SCSI tape, *N* is the instance number
fru	P	"A"	"yes"	Field replaceable unit
model	P	"A"		For example, Exabyte EXB-8500 8mm
name	P	"A"		For example, /dev/rmt/0n
status	P	"A"	"OK/FAIL"	
device_type	P	"A"	"tape drive"	
instance*	P	"I"		
system	N			Contains general properties of the system
total_processors	P	"I"		Number of processors in the machine
total_memory	P	"A"		*N* MB, where N is an integer
total_disks	P	"I"		
total_tape_devices	P	"I"		
sample number	P	"I"		Since daemon started on server

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
timestamp	P	"A"		For example, `Tue Mar 26 10:33:08 1996`
hostname	P	"A"		Machine name
OS	P	"A"		Value as returned by `uname(2)` in release field
OS_version	P	"A"		Value as returned by `uname(2)` in release field
architecture	P	"A"		For example, `sparc`, `mc68030`, `m32100`, or `8386`
machine	P	"A"		For example, `sun4u`
platform	P	"A"		Value as returned by the `uname -i` command
serial_number	P	"I"		
System_clock_frequency	P	"A"		*N* MHz, where *N* is an integer
The following nodes are not physical entities. They are the output of different power supplies in the system that is used for purposes such as hotplugging, and peripherals.				
hot_plug_charges	N			
fru	P	"A"	"no"	Field replaceable unit
auxiliary_5v	N			
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit
peripheral_12v				
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit
peripheral_12v_precharge	N			
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit
peripheral_5v	N			
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit

*** Refer to Table B-1.**

TABLE B-2 Config Reader Data Hierarchy for the Ultra Enterprise 3000, 4000, 5000, and 6000 Systems *(Continued)*

Node or Property and Name	Node or Property	Type	Default Value	Description
peripheral_5vprecharge	N			
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit
system_5v	N			
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit
system_5v_ precharge	N			
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit
system_3v	N			
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit
system_3v_ precharge	N			
status	P	"A"	"OK"/"FAIL"	
fru	P	"A"	"no"	Field replaceable unit

*** Refer to Table B-1.**

SPARCserver 1000 / 1000E and SPARCcenter 2000 / 2000E

The following table is the Config Reader data hierarchy for the SPARCstation 1000 / 1000E and the SPARCcenter 2000 / 2000E.

TABLE B-3 SPARCserver 1000 / 1000E and SPARCcenter 2000 / 2000E Config Reader Data Hierarchy

Name	Node or Property	Type	Default Value	Description
QLGC, isp(*N*)	N			(*N*) = unit number
device_type	P	"A"	"SCSI controller"	
disk_count	P	"I"		
fru	P	"A"	"yes"	Field replaceable unit
instance*	P	"I"		
model	P	"A"		
tape_count	P	"I"		
board(*N*)	N			(*N*) = board number; this is a unique number ranging from 0 to the maximum number of boards per machine
board-num	P	"I"		same as *N*, above
fru	P	"A"	"yes"	Field replaceable unit
state	P	"A"	One of "Active," "disabled," or "failed."	• "active:" Normal state for boards after the operating system has booted • "disabled:" Disabled by the PROM because it was listed in the 'disabled-board-list' property in the options node • "failed:" A board that is failed by POST and powered off in the testing phase of PROM operations
type	P	"A"	"cpu/memory/io"	
cgsix(*N*)	N			(*N*) = unit number
device_type	P	"A"	"display"	

*** Refer to Table B-1.**

TABLE B-3 SPARCserver 1000/1000E and SPARCcenter 2000/2000E Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
instance*	P	"I"		
model	P	"A"		
cpu-unit(*N*)	N			(*N*) = unit number
board#	P	"I"		Board number on which this node resides
clock-frequency	P	"A"	"N MHz"	
cpu-type	P		"sparc"	
device_type	P		"cpu"	
fru	P	"A"	"yes"	Field replaceable unit
module	P	"A"		
processor-id	P	"I"	2*board number (+1 if cpu-unit is "B")	
sparc-version	P	"I"		
status	P	"A"	"online"/ "offline"	
unit	P	"A"	"A" or "B" (each board takes two CPUs)	
dma(*N*)	N			(*N*) = unit number
instance*	P	"I"		
model	P	"A"		
esp(*N*)	N			(*N*) = unit number
chip	P	"A"		
device_type	P	"A"	"SCSI controller"	
disk_count	P	"I"		Number of disks attached to this node
instance*	P	"I"		
tape_count	P	"I"		Number of tapes attached to this node
io-unit(N)	N			

*** Refer to Table B-1.**

TABLE B-3 SPARCserver 1000/1000E and SPARCcenter 2000/2000E Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
board#	P	"I"		Board number on which this node resides
device-id	P	"I"		
instance*	P	"I"		
mem-unit board#	N	"I"		Board number on which this node resides
device-id	P	"I"		
device_type	P	"A"	"memory"	
size	P	"A"	"N MB"	
status	P	"A"	"OK"/"FAIL"	
le(*N*)	N			(*N*) = unit number
alias	P		"le"	
device_type	P		"network"	
ether	P		"8:0:20:74:30:43"	
inet	P		"129.146.235.47"	
instance*	P	"I"		
name	P		"le0"	
symbol	P		"pump"	
lebuffer(*N*)	N			(*N*) = unit number
instance*	P	"I"		
model	P	"A"		
network_count	P	"I"		
sbi(*N*)	N			(*N*) = unit number
device_type	P		"SBus card"	
instance*	P	"I"		
simm(*N*)	N			(*N*) = unit number
group#	P	"I"		
memory-type	P		"RAM"	

*** Refer to Table B-1.**

TABLE B-3 SPARCserver 1000/1000E and SPARCcenter 2000/2000E Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
size	P		"32MB"	
sd(N)	N			(N) = disk number
device_type	P		"disk"	
fru	P	"A"	"yes"	Field replaceable unit
instance*	P	"I"		
mounted_partitions	P	"I"		
name="c1t3d0"	P	"A"	"N mbytes"	Partition name
directory_name	P	"A"		For example, / or /usr
avail_bytes	P	"A"	"N mbytes"	
device_type	P		"partition"	You don't see these partitions unless mounted_partitions != 0
name	P		"c1t3d0s0"	You don't see these partitions unless mounted_partitions != 0
percent_used	P	"A"	"N%"	You don't see these partitions unless mounted_partitions != 0
total_bytes	P	"A"	"N mbytes"	You don't see these partitions unless mounted_partitions != 0
used_bytes	P	"A"	"N mbytes"	You don't see these partitions unless mounted_partitions != 0
slot(*N*)	N			Slot where boards are physically installed; (*N*) = unit number
slot-num	P	"I"		
sd	N			This node only exists in cases where no disks are attached to the SCSI controller
device_type	P	"A"	"block"	
fru	P	"A"	"yes"	Field replaceable unit

*** Refer to Table B-1.**

TABLE B-3 SPARCserver 1000/1000E and SPARCcenter 2000/2000E Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
st	N			This node only exists in cases where no tapes are attached
device_type	P	"A"	"byte"	
fru	P	"A"	"yes"	Field replaceable unit
system	N			
OS	P	"A"		Value as returned by uname(2) in the release field
OS_version	P	"A"		Value as returned by uname(2) in the release field
System_clock_frequency	P	"A"	"N MHz"	
architecture	P	"A"		For example, sparc, mc68030, m32100, or 8386
hostname	P	"A"		Machine name
machine	P	"A"	"sun4d"	
platform	P	"A"		Value as returned by the uname -i command
serial_number	P	"I"		
timestamp	P	"A"		
total_disks	P	"I"		Number of disks attached to the machine
total_memory	P	"A"	"N MB"	
total_processors	P	"I"		Number of processors in the machine
total_tape_devices	P	"I"		Number of tapes attached to the machine

*** Refer to Table B-1.**

Ultra Enterprise 2 and150 System

The following table is the Config Reader data hierarchy for the Ultra Enterprise 2 and Ultra Enterprise 150 systems.

TABLE B-4 Ultra Enterprise 2 and 150 Config Reader Data Hierarchy

Name	Node or Property	Type	Default Value	Description
SUNW,fas	N			Host Bus Adapter driver
tape_count	P	"I"		Number of tapes attached to a slot
disk_count	P	"I"		Number of disks attached to a slot
instance*	P	"I"		
device_type*	P	"A"	"scsi"	
SUNW,ffb(*N*)				UPA color frame buffer and graphics accelerator (*N*) = instance number
buffer	P	"A"	"Single"/ "Double"	
3DRAM	P	"A"		3-D RAM model and version number
DAC	P	"A"		Revision of DAC chip
FBC_version	P	"H "		
revision	P	"A"		Revision of FFB board
instance*	P	"I"		
upa-portid*	P	"I"		
width	P	"I"		Pixels
height	P	"I"		Pixels
device_type	P	"A"	"display"	
model	P	"A"		
SUNW,hme	N			Fast Ethernet device driver
instance*	P	"I"		
device_type	P	"A"	"network"	

*** Refer to Table B-1.**

TABLE B-4 Ultra Enterprise 2 and 150 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
SUNW,leo	N			Double-buffered SBus color frame buffer and graphics accelerator
instance*	P	"I"		
width	P	"I"		Pixels
height	P	"I"		Pixels
model	P	"A"		
device_type	P	"A"	"display"	
SUNW,pln(N)	N			SPARCstorage Array
disk_count	P	"I"		Number of attached disks
device_type	P	"A"		SPARCstorage Array
instance*	P	"I"		
SUNW,rtvc	N			Real-time video capture
instance*	P	"I"		
model	P			
SUNW,soc(*N*)*	N			Serial Optical Controller device driver; (*N*) = instance number
device_type	P	"A"	See Description	Serial Optical Channel Processor Host Adapter
instance*	P	"I"		
model	P			
soc-wwn	P	"I"		World Wide Web device name
port-wwns	P	"I"		
system-board	N			
state=	P	"A"		"active:" Normal state for boards after the operating system has booted "unknown:" Unused
fru	P	"A"	"yes"	Field replaceable unit
board-num	P	"I"		Always 1
memory_size	P	"A"		*N* MB, where *N* is an integer (CPU / memory boards only)
cdrom	N			

*** Refer to Table B-1.**

TABLE B-4 Ultra Enterprise 2 and 150 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
mounted_partitions	P	"I"		
device_type	P	"A"		CD-ROM
name	P	"A"		For example, c0t6d0
instance*	P	"I"		
cpu-unit(*N*)	N			(*N*) = unit
fru	P	"A"	"yes"	Field replaceable unit
clock-frequency	P	"A"		*N* MHz, where *N* is an integer
cpu-type	P	"A"		For example, sparc
status	P	"A"	"online"/ "offline"	
model	P	"A"		
processor-id	P	"I"		0 or 1 (only 0 possible on the Ultra Enterprise 150 system)
unit	P	"A"		"A" or "B" (only on UE-2)
upa-mid*	P	"I"		
upa-portid*	P	"I"		
icache-size*	P	"A"		*N* KB or *N* MB, where *N* is a fixed point number
dcache-size	P	"A"		N KB or N MB, where N is a fixed point number
ecache-size	P	"A"		N KB or N MB, where N is a fixed point number
device_type	P	"A"	"cpu"	
nf(N)	N			SBus FDDI Driver
ether	P	"A"		Ethernet address; for example, 8:0:20:75:af:f8
inet	P	"I"		Internet address; for example, 129.146.65.45
symbol	P	"A"		Network name corresponding to the inet address above
name	P	"A"		Driver name; for example. nf0
instance*	P	"I"		

*** Refer to Table B-1.**

TABLE B-4 Ultra Enterprise 2 and 150 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
model	P	"A"		
device_type	P	"A"	"network"	
sbus(*N*)	N			Peripheral bus; *N*=instance number
instance*	P	"I"		
upa-portid*	P	"I"		
upa-mid*	P	"I"		
version#*	P			
device_type*	P	"A"	"sbus"	
model	P			
network_count	P	"I"		Network devices attached to this SBus
sd(0)	N			
fru	P	"A"	"yes"	Field replaceable unit
mounted_partitions	P	"I"		
device_type	P	"A"		
name	P	"A"		"disk" for example, c0t0d0
instance	P	"I"		
The following data, which is a continuation of the properties that go with the sd(0) node (above), is the output of Config Reader if the disk has mounted partition.				
name	P	"A"		For example, c0t0d0s0
percent_used	P	"I"		For example, 49%
total_bytes	P	"I"		*n* MBytes or *n* KBytes
avail_bytes	P	"I"		*n* MBytes or *n* KBytes
used_bytes	P	"I"		"*n* MBytes or *n* KBytes
device_type*	P	"A"		"partition"
export name	P	"A"		For example, c0t0d0s7
simm(*N*)	N			*N* is simm instance.
slot	P	"I"		CPU/memory slot where this SIMM resides

*** Refer to Table B-1.**

TABLE B-4 Ultra Enterprise 2 and 150 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
size	P	"A"		
type	P	"A"		"dram"
fru	P	"A"	"yes"	Field replaceable unit
board-reference-number	P	"A'		Address of SIMM on board
st(*N*)	N			SCSI tape, *N* is the instance number
fru	P	"A"	"yes"	Field replaceable unit
model	P	"A"		For example, Exabyte EXB-8500 8mm
name	P	"A"		For example, /dev/rmt/0n
status	P	"A"	"OK/FAIL"	
device_type	P	"A"	"tape drive"	
instance*	P	"I"		
system	N			Contains general properties of the system
total_processors	P	"I"		Number of processors in the machine
total_memory	P	"A"		*N* MB, where *N* is an integer
total_disks	P	"I"		
total_tape_devices	P	"I"		
sample number	P	"I"		Since daemon started on server
timestamp	P	"A"		For example, Tue Mar 26 10:33:08 1996
hostname	P	"A"		Machine name
OS	P	"A"		Value as returned by uname(2) in release field
OS_version	P	"A"		Value as returned by uname(2) in release field
architecture	P	"A"		For example, sparc, mc68030, m32100, or 8386
machine	P	"A"		For example, sun4u

*** Refer to Table B-1.**

TABLE B-4 Ultra Enterprise 2 and 150 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
platform	P	"A"		Specific model of the hardware platform
serial_number	P	"I"		
System_clock_frequency	P	"A"		*N* MHz, where *N* is an integer
sbus-board(*N*)	N			The SBus board in slot *N*
board-num	P	"I"		The number of the board in its particular sbus
fru	P	"A"	"yes"	Field replaceable unit
model	P	"A"		Board model—either a model name (if the board is identified by Solstice Symon) or the Open Boot PROM model string (for example, SUNW,501-2739)
state	P	"A"		"active:" Normal state for boards after the operating system has booted
le	N			Ethernet device driver
instance*	P	"I"		
device_type	P	"A"	"network"	
ether	P	"A"		Ethernet address; for example, "8:0:20:75:af:f8"
inet	P	"I"		Internet address; for example, "129.146.65.45"
name	P	"A"		Driver name; for example, le0
device_type	P	"A"	"network"	
SUNW,fdtwo	N			Floppy disk device
instance*	P	"I"		
device_type	P	"A"	"block"	
fru	P	"A"	"yes"	Field replaceable unit

*** Refer to Table B-1.**

If any component of a hierarchy path contains a period (.) in it, you may not be able to use the `findlist` function to find it. For example, if you search for the mount point `/export/a0.test` reported by the Config Reader via the path:
`system.slot.board.io-unit.sbi.dma.esp.sd./export/a0.test`
the node will not be found. Instead, use:
`system.slot.board.io-unit.sbi.dma.esp.sd.*` to find this node.

Ultra Enterprise 450 System

TABLE B-5 lists the Config Reader data hierarchy for the Ultra Enterprise 450 system.

TABLE B-5 Ultra Enterprise 450 Config Reader Data Hierarchy

Name	Node or Property	Type	Default Value	Description
`glm`	N			Host Bus Adapter driver
`chassis-disk-group`	P	“I”		Shows which of the 5 in-chassis disk groups is attached to this controller; this property is not present if the controller is not attached to internal disks
`clock-frequency`	P	“I”		Clock frequency of the controller (MHz)
`logical-disk-controller`	P	“A”		The logical controller name used to refer to this controller in the `/dev/dsk` directory
`compatiblen`	P	“A”		Alternate name for this node; may determine driver being used; will have value `glm` for SUN fast-wide SCSI controller
`tape_count`	P	“I”		Number of tapes attached to a slot
`disk_count`	P	“I”		Number of disks attached to a slot
`instance*`	P	“I”		
`model`	P	“A”	Symbios,53C875	Model name
`device_type*`	P	“A”	`“scsi-2”`	

*** Refer to Table B-1.**

TABLE B-5 Ultra Enterprise 450 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
SUNW,ffb(*N*)	N			UPA color frame buffer and graphics accelerator; *N* = instance number
buffer	P	"A"	"Single"/ "Double"	
3DRAM	P	"A"		3-D RAM model and version number
DAC	P	"A"		Revision of DAC chip
FBC_version	P	"H "		
revision	P	"A"		Revision of FFB board
instance*	P	"I"		
upa-portid*	P	"I"		
width	P	"I"		Pixels
height	P	"I"		Pixels
device_type	P	"A"	"display"	
model	P	"A"		
hme	N			Network device
compatiblen	P	"A"		Alternate device name; will have value SUNW,hme for Sun Fast Ethernet device
instance*	P	"I"		
device_type	P	"A"	"network"	
pln(N)	N			SPARCstorage Array
disk_count	P	"I"		Number of attached disks
device_type	P	"A"		SPARCstorage Array
instance*	P	"I"		
soc(*N*)*	N			Serial Optical Controller device driver; *N* = instance number
device_type	P	"A"	See Description	Serial Optical Channel Processor Host Adapter
instance*	P	"I"		
model	P			

*** Refer to Table B-1.**

TABLE B-5 Ultra Enterprise 450 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
soc-wwn	P	"I"		World Wide Web device name
port-wwns	P	"I"		
system-board	N			
state=	P	"A"		"active:" Normal state for boards after the operating system has booted"unknown:" Unused
fru	P	"A"	"yes"	Field replaceable unit
board-num	P	"I"		Always 1
cdrom*N*	N			
mounted_partitions	P	"I"		
device_type	P	"A"		CD-ROM
name	P	"A"		For example, c0t6d0
instance*	P	"I"		
cpu-unit(*N*)	N			*N* = unit; Possible values are 1 through 4
fru	P	"A"	"yes"	Field replaceable unit
clock-frequency	P	"A"		*N* MHz, where *N* is an integer
cpu-type	P	"A"		For example, sparc
status	P	"A"	"online"/ "offline"	
model	P	"A"		
processor-id	P	"I"		0 through 3
unit	P	"A"		"A1" or "B1"or "A2" or "B2"
upa-mid*	P	"I"		
upa-portid*	P	"I"		
icache-size*	P	"A"		*N* KB or *N* MB, where *N* is a fixed point number
dcache-size	P	"A"		N KB or N MB, where N is a fixed point number
ecache-size	P	"A"		*N* KB or *N* MB, where *N* is a fixed point number

*** Refer to Table B-1.**

TABLE B-5 Ultra Enterprise 450 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
device_type	P	"A"	"cpu"	
temperature	P	"A"		*N* C< where *N* is an integer
pci(*N*)	N			Peripheral bus; *N* = instance number
instance*	P	"I"		
clock-frequency	P	"I"		Speed of the PCI bus in MHz (33 MHz or 66 MHz)
upa-portid*	P	"I"		
upa-mid*	P	"I"		
version#*	P			
device_type*	P	"A"	"sbus"	
model	P			
network_count	P	"I"		Network devices attached to this SBus
sd*N*	N			SCSI disk; *N* is the instance number
fru	P	"A"	"yes"	Field replaceable unit
mounted_partitions	P	"I"		
device_type	P	"A"		
name	P	"A"		"disk"; for example, c0t0d0
instance	P	"I"		
The following data, which is a continuation of the properties that go with the sd*N* node above, is the output of the Config Reader if the disk is physically located in the Ultra Enterprise 450 chassis.				
error_led	P	"A"	On / Off	State of the error LED of the disk
disk-slot	P	"I"		Slot number in the chassis front: 9 to 19
The following data, which is a continuation of the properties that go with the sd*N* node (above), is the output of Config Reader if the disk has mounted partition.				
name	P	"A"		For example, c0t0d0s0
percent_used	P	"I"		For example, 49%
total_bytes	P	"I"		*n* MBytes or *n* KBytes

*** Refer to Table B-1.**

TABLE B-5 Ultra Enterprise 450 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
avail_bytes	P	"I"		*n* MBytes or *n* KBytes
used_bytes	P	"I"		*n* MBytes or *n* KBytes
device_type*	P	"A"		Partition
export name	P	"A"		For example, c0t0d0s7
mc(*N*)	N			Memory controller; instance *N*
device_type	P	"A"	memory-bank	Device type
bank(*N*)	N			Bank; instance *N*
device_type	P	"A"	memory-bank	Device type
bank-size	P	"A"	*N* Mb	Bank size in Mb
status	P	"A"	"ok"	Status of the bank—"missing DIMMs," "mis-sized," or "ok"
interleave-group	P	"I"		Group of interleaved banks; only present if this bank is interleaved
dimm(*N*)	N			*N* is dimm instance within the bank
slot	P	"I"		Memory slot where this DIMM resides
status	P	"A"	"ok"	Status of DIMM—"unused," "partially used, " or "ok"
module-size	P	"A"		
type	P	"A"		"dram"
socket-name	P	"A'		Address of DIMM on board
fru	P	"A"	"yes"	Field replaceable unit
st(*N*)	N			SCSI tape, *N* is the instance number
fru	P	"A"	"yes"	Field replaceable unit
model	P	"A"		For example, Exabyte EXB-8500 8mm
name	P	"A"		For example, /dev/rmt/0n
status	P	"A"	"OK/FAIL"	
device_type	P	"A"	"tape drive"	

*** Refer to Table B-1.**

TABLE B-5 Ultra Enterprise 450 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
instance*	P	"I"		
system	N			Contains general properties of the system
total_processors	P	"I"		Number of processors in the machine
ambient-temperature	P	"A"	*nn* C	Ambient temperature of the system in degrees C
total_memory	P	"A"		*N* MB, where *N* is an integer
total_disks	P	"I"		
total_tape_devices	P	"I"		
sample number	P	"I"		Since daemon started on server
timestamp	P	"A"		For example, `Tue Mar 26 10:33:08 1996`
hostname	P	"A"		Machine name
OS	P	"A"		Value as returned by `uname(2)` in the release field
OS_version	P	"A"		Value as returned by `uname(2)` in the release field
architecture	P	"A"		For example, `sparc`, `mc68030`, `m32100`, or `8386`
machine	P	"A"		For example, `sun4u`
platform	P	"A"		Specific model of the hardware platform
serial_number	P	"I"		
System_clock_frequency	P	"A"		*N* MHz, where *N* is an integer
pci-board(*N*)	N			The PCI board in slot *N* of the particular PCI bus
board-num	P	"I"		The number of the board in its particular SBus
fru	P	"A"	"yes"	Field replaceable unit

*** Refer to Table B-1.**

TABLE B-5 Ultra Enterprise 450 Config Reader Data Hierarchy *(Continued)*

Name	Node or Property	Type	Default Value	Description
model	P	"A"		Board model — either a model name (if the board is identified by Solstice Symon) or the Open Boot PROM model string (for example, SUNW,501-2739)
state	P	"A"		"active:" Normal state for boards after the operating system has booted
fdthree	N			Floppy disk device
instance*	P	"I"		
fru	P	"A"	"yes"	Field replaceable unit
device_type	P	"A"	"block"	
cpu-fans	N			Rack of 3 fans cooling the processors and peripheral boards
status	P	"A"	"OK"	Status of fan group; "OK" if all fans OK
power-supply-fans	N			Rack of 3 fans cooling the power supplies
status	P	"A"	"OK"	Status of fan group; "OK" if all fans OK
fan(N)	N			Individual fan node
status	P	"A"	"OK"	Fan status
front-panel-keyswitch	N			
position	P	"A"		Position of key
power-supply(N)	N			Power-supply node
current-share	P	"A"		State of current—sharing
limit	P	"A"		State of current—-limiting
status	P	"A"		State of power supply
power-rating	P	"A"		Rating of power supply: "*N* W"
temperature	P	"A"		*N* C< where *N* is an integer

*** Refer to Table B-1.**

If any component of a hierarchy path contains a period (.), you may not be able to use the `findlist` function. For example, if you search for the mount point `/export/a0.test` reported by the Config Reader via the path:
`system.slot.board.io-unit.sbi.dma.esp.sd./export/a0.test`
the node will not be found. Instead, use:
`system.slot.board.io-unit.sbi.dma.esp.sd.*` to find this node.

APPENDIX **C**

Installing and Setting Up SNMP Traps

This appendix describes how to install and set up SNMP traps. A SNMP trap is an asynchronous message directed at a specific monitor.

The Event Generator can send SNMP traps to applications that listen for SNMP traps, such as Solstice Site Manager and Solstice Domain Manager. Traps include information about events on the system that may be useful when running a Solstice Site Manager and Solstice Domain Manager console.

Installing SNMP Traps

Note – Before you begin this procedure, make sure you know the machine name of the *snmphost* and the IP address. You will need to enter this information when you install SNMP traps.

To install SNMP version 2.2.3 packages on the machine named “snmphost,” refer to the SNMP installation menu.

▼ To Install SNMP Traps

1. **Make sure the software that receives SNMP traps is installed and configured.**
2. **On the machine** *snmphost***, copy the following three files in the** `/opt/SUNWsymon/etc/snm` **directory on the event host to the** `/opt/SUNWconn/agents` **directory as superuser. This is the default SNMP directory.**
   - `symon.mib.oid`
   - `symon.mib.traps`
   - `symon.mib.schema`
3. **Alter the** `snmp_hosts` **variable by running the following command on the event host:**

```
# /opt/SUNWsymon/sbin/sm_confsymon -e event_host -S "list_of_snmphost"
```

   where:

   `-S` "*list_of_snmphost*" indicates a space-separated list of hosts to receive the SNMP traps.

4. **Start the Event Generator daemon by entering:**

```
# /opt/SUNWsymon/sbin/sm_control start
```

5. **Run** `snm -i` **and select BasicStart.**

   Wait for the SNMP Console to come up from the main menu.

6. **Select** `Edit->Create`**.**

   a. **Make sure your "Category" is** `Component`.

   b. **Select your "Type" to be** `WorkStation`**.**

   c. **Click the Create button.**

   A Create Form pop-up window is displayed.

7. **Type the following:**
   - Name: *snmp_hostname*
   - IP address: *snmp_hostname's IP address*
   - SNMP Rdcommunity: `public`
   - SNMP Wrcommunity: `public`
8. **Select the** `SYMONMIB` **check button, then click Apply.**

9. **Select the workstation icon you just created and from the main menu, select** `View—>Event/Trap` **Report.**

10. **Send test traps using this command on the event host machine:**

```
# /opt/SUNWsymon/sbin/trapsend -a "1.3.6.1.4.1.42 STRING (STATUS | FAIL)"
```

If you receive the message above in double quotes in your `Event/Trap` Report, you are ready to receive SNMP traps from the server.

APPENDIX **D**

Default Solstice SyMON Rules

This appendix describes the default rules for this release of Solstice SyMON. Use this appendix to determine the rules that are incorporated into this Solstice SyMON release. By identifying the default rules, you can determine whether or not you need to write a new rule.

Note – The `sample.tcl` file in Solstice SyMON 1.1 through 1.3 contained only a comment.

TABLE D-1 describes the default rules supplied with this release of Solstice SyMON. The columns in the table describe the following:

- Column 1 is the rule number.
- Column 2 is a description of the rule.
- Column 3 is the action taken when the condition is true.

TABLE D-1 Default Solstice SyMON Rules

Rule Number	Description	Action
General: `syrules.tcl`		
0	Symond status	`alarm, snmp`
1	Server status	`alarm, snmp`
3	Log Scanner status	`alarm, snmp`
4	Kernel Reader status	`alarm, snmp`
6	Delay initial Log Scanner and death status	none
7	Delay initial Kernel Reader death status	none
8	Warning if label contains `"."`	
9	Warning if label contains `"*"`	`alarm, snmp`
General: `egrules.tcl`		

TABLE D-1 Default Solstice SyMON Rules *(Continued)*

Rule Number	Description	Action
50	Trim event log if getting full	`alarm, snmp`
51	Warning when maximum number of events is near	`alarm, snmp`
52	Warning when `ls_hist` is corrupted	`alarm, snmp`
53	Warning on incorrect Tcl rule syntax	`alarm, snmp`
General: `swrules.tcl`		
100	Disk busy	`alarm, snmp`
102	Swap space	`alarm, snmp`
103	CPU swapping and paging	`alarm, snmp`
104	CPU load	`alarm, snmp`
105	File system full	`alarm, snmp`
106	No swap space	`alarm, snmp`
General: `hwrules.tcl`		
209	SIMM error	`alarm, snmp`
210	`ssd` hardware error	`alarm, snmp`
211	`ssd` fatal error	`alarm, snmp`
212	SIMM error	`alarm, snmp`
214	`configd.*.status` failing	`alarm, snmp`
216	CPU panic	`alarm, snmp`
217	Tape error rate high	`alarm, snmp`
219	`ssd` off-line	`alarm, snmp`
220	`ssd` okay (online)	`alarm, snmp`
222	Fatal SunVTS error	`alarm, snmp`
223	SunVTS error	`alarm, snmp`
General: `pfrules.tcl`		
300	SIMM errors	`add_pfa SIMM`
301	Disk soft errors	`add_pfa DISK`
302	Disk soft errors from smart drives	`alarm, snmp`
303	Check SIMM (test period 1)	`alarm, snmp`
304	Check SIMM (test period 2)	`alarm, snmp`

TABLE D-1 Default Solstice SyMON Rules *(Continued)*

Rule Number	Description	Action
305	Check SIMM (test period 3)	`alarm, snmp`
306	Check disk `pfa` (warning)	`alarm, snmp`
307	Check disk `pfa` (severe)	`alarm, snmp`
308	Trims `pfa` history	none
General: `cprules.tcl`		
400	CPU overload over *x* hours	`alarm, snmp, add_cpa`
401	Disk busy over *x* hours	`alarm, snmp, add_cpa`
402	Swap space low over *x* hours	`alarm, snmp, add_cpa`
403	Meta CPU events (`check_cpa_cpu`)	`alarm, snmp`
404	Meta disk events (`check_cpa_disk`)	`alarm, snmp`
405	Meta swap events (`check_cpa_swap`)	`alarm, snmp`
406	Trims `cpa` history	
407	Initializes timestamp for rules 400 and 402	
Platform-Specific Rules: SPARCserver 1000(E), SPARCcenter 2000(E) - `SS1000.tcl, SC2000.tcl`		
1002	Config Reader status	`alarm, snmp`
1005	Delay initial Config Reader death status	none
1207	`system.*.*.*.status`	`alarm, snmp`
1208	`system.*.*.*.*.*.*.status`	`alarm, snmp`
1224	`system.*.*.*.*.*.*.*.status`	`alarm, snmp`
1225	Corrected ECC error	`alarm, snmp`
Platform-Specific Rules: Ultra Enterprise 450: `UE450.tcl`		
1002	Config Reader status	`alarm, snmp`
1005	Delay initial configd death status	none
1204	`system.*.fan.status`	`alarm, snmp`
1205	`system.*.*.*.temperature`	`alarm, snmp, mailto`
1207	`system.*.*.*.status`	`alarm, snmp`

TABLE D-1 Default Solstice SyMON Rules *(Continued)*

Rule Number	Description	Action
1208	`system.*.*.*.*.*.*.status`	`alarm, snmp`
1212	Power supply state	`alarm, snmp`
1213	Power supply limit	`alarm, snmp`
1214	Power supply current share	`alarm, snmp`
1215	Power supply temperature	`alarm, snmp, mailto`
1224	`system.*.*.*.*.*.*.*.status`	`alarm, snmp`
1225	Corrected ECC error	`alarm, snmp`
Platform-Specific Rules: Ultra Enterprise 3000, 4000, 5000, 6000: `UEnterprise.tcl`		
1002	Config Reader status	`alarm, snmp`
1005	Delay initial Config Reader death status	none
1201	Hot plug charge DC status	`alarm, snmp`
1202	Hot plug precharge DC status	`alarm, snmp`
1203	`system.*.status`	`alarm, snmp`
1204	Fan status	`alarm, snmp`
1205	CPU board temperature	`alarm, snmp, mailto`
1206	I/O and clock board temperature	`alarm, snmp, mailto`
1207	`system.*.*.*.status`	`alarm, snmp`
1208	`system.*.*.*.*.*.status`	`alarm, snmp`
1209	`system.*.*.*.*.*.*.status`	`alarm, snmp`
1213	Hot plug "removed"	`alarm, snmp`
1215	Hot plug "hotplugged into"	`alarm, snmp`
1218	`system.*.*.*.*.status`	`alarm, snmp`
1220	Hot plug "installed"	`alarm, snmp`
1221	Redundant power status	`alarm, snmp`
Platform-Specific Rules: Ultra Enterprise 150: `UEnterpriseI.tcl`		
1002	Config Reader status	`alarm, snmp`

TABLE D-1 Default Solstice SyMON Rules *(Continued)*

Rule Number	Description	Action
1005	Delay initial `configd` death status	none
1207	`system.*.*.*.status`	`alarm, snmp`
1208	`system.*.*.*.*.*.status`	`alarm, snmp`
1224	`system.*.*.*.*.*.*.*.*.status`	`alarm, snmp`

Glossary

action Tells Solstice SyMON what to do when a condition is true, if the condition changes, or if the system shuts down. Use actions to notify users of a situation that may require attention. For example, an action might be to create an event, to send a record of the event to the Event Log; to generate an email message about the event; to use a modem to dial a beeper, or to execute a program. You can define new actions as needed.

agent A daemon that collects data from the server.

`alarm` A predefined procedure for event rules that makes an event active; opens a log entry in the Event Log for the event; optionally posts a message to the Event Viewer; and highlights the affected node in the appropriate console.

attributes Elements of an event rule. Attributes include the rule number, condition, actions, level, priority, and severity.

closed event When the condition that generated the event no longer exists, the Event Manager closes the event and the event is referred to as a *closed event*. The event log file records closing events

complex rule An event rule in which a single rule can generate multiple events and log entries by searching a range of parameters. The `MULTI` parameter in a rule designates it as a complex rule. Complex rules make the Event Manager evaluate the condition for multiple node variables. For example, complex rules can be used to make the Event Manager check the condition for each board on a system.

condition A Tcl string that defines when a rule is active. This is the only mandatory part of a rule. A condition does not have a tag. Examples of conditions include a failed board or a DNLC (directory name lookup cache) long name ratio greater than 25 percent.

`end_alarm` A predefined procedure for event rules that closes the log entry in the Event Log for the event, deletes the event from the Event Viewer, and removes any console.

event An action that is used to notify other applications of a condition, typically hardware or operating system conditions that may require the attention of the system administrator.

Event Generator Collects information from the Config Reader, Kernel Reader, and Log Scanner and evaluates the data against its rules. It also maintains data about itself and the state of the server agents. This enables rules to be written against the status of agents (including itself) as well as data from the server.

Event Log A log in which the Event Generator records events.

Event Manager A subsystem of Solstice SyMON that reports events based on a set of rules. The Event Manager collects data from server agents and evaluates them for conditions defined within its rules. When a condition is true, an event occurs and the Event Manager implements any specified action(s) in the rule. When the condition that generated the event no longer exists, the Event Manager may run a special action such as closing the event.

Event Viewer Displays the Event Log information, which includes `ID, LVL, OPEN, CLOSED, DURATION, MESSAGE LOCATION, PRI, SEVERITY, ACKED, ACKED BY, RULE NUMBER`.

Kernel Data Catalog Displays a hierarchical schematic view of the Solaris performance parameters that Solstice SyMON monitors.

Launcher The main window of Solstice SyMON. Each icon on the console launches one of the seven consoles: Event Viewer, Log Viewer, Physical View, Logical View, Kernel Data Catalog, Process Viewer, and the On-Line Diagnostics screen. It also reports status of the server and the Solstice SyMON agents.

Log Viewer Displays matching entries in a selected log file.

Logical View Displays a hierarchical schematic view of the configuration and state.

`MULTI` A special event rule attribute used to permit the rule to check a list of node variables for multiple events off a single rule. For example, the `MULTI` string can be used to make the Event Generator check a condition of a rule for each board in the system.

`ON_OPEN` An event rule attribute that is interpreted when a condition for a rule becomes true.

`ON_CLOSE` An event rule attribute that is interpreted when a rule is no longer true.

`ON_CONTINUE` An event rule attribute that is interpreted when a rule continues to be true two or more times in a row.

`ON_SHUTDOWN` An event rule attribute that is interpreted when the system shuts down.

Physical View Displays graphical images of the server, which are updated dynamically as the server changes configuration or state.

point event An event that is generated due to log messages.

PRIORITY — A rule attribute. A priority for the rule for the user to interpret that is used in conjunction with SEVERITY.

procedures file — Defines procedures used by rules. The file must exist but can be empty. The file is located in /etc/opt/SUNWsymon/event_gen.*servername*.tcl. Note that *servername* is the name of the server.

Process Viewer — Displays information on processes running on the server. Information includes the time the process was started and by whom; total CPU that the process has used; and its current status.

rule — A set of conditions and actions associated with that condition. Rules are always in the RULES Tcl variable, which is located in the file /etc/opt/SUNWsymon/rules.tcl.

rules functions — A rules function is a predefined procedure such as ON_OPEN or ON_CLOSE; it is used within an event rule, such as alarm and end_alarm.

Rules **variable** — A list of event rules specified in the file /etc/opt/SUNWsymon/rules.tcl. This variable is case sensitive.

SEVERITY — An event rule attribute that indicates the severity of the rule for the user to interpret. Used in conjunction with PRIORITY.

simple rule — An event rule that has a one-to-one correspondence between the rule and an event. Because a simple rule does not check a wildcard list of variables, it can only generate a single event.

strip string — The strip string is any subpath that is on the full path. Enclosing an item in the strip string deletes that item from the expanded path. If the strip string is null, you must use double quotes. If the strip string is not null, enclosing the strip string in double quotes is not required, but allowable.

System Meter — A GUI window within the Kernel Data Catalog that displays graphs of monitored performance parameters.

Tcl — A scripting language used to define rules.

Index

D

E

F

G

H

I

K

L

M

S

T

U

V

W

Y

Z