

SunNet Manager™ 2.2.3

Reference Manual

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 801-6986-10
Revision A of September 1995



© 1995 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 UNIX Systems Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and from the Berkeley 4.3 BSD system, licensed from the University of California. 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.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, the Sun logo, and Solaris, SunSoft, the SunSoft logo, SunOS, OpenWindows, DeskSet, ONC, SNM, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. OPEN LOOK is a registered trademark of Novell, Inc. PostScript and Display PostScript are trademarks of Adobe Systems, Inc.

All SPARC trademarks are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. SPARCcenter, SPARCcluster, SPARCcompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC-II, and UltraSPARC are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

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 GUI's 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.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN, THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAMS(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.



Contents

Preface	xix
1. Introduction.....	1-1
1.1 SunNet Manager Architecture	1-2
1.2 Console	1-4
1.2.1 Management Database.....	1-5
1.2.2 Console Tools	1-6
1.3 Agents and Proxies	1-6
1.3.1 Agents Included with SNM 2.2.3	1-8
1.3.2 Ancillary Daemons	1-12
1.4 SunNet Manager Directories and Files	1-13
1.4.1 Installation Directory Files.....	1-14
1.4.2 Database Files	1-15
1.4.3 Log Files.....	1-15
1.4.4 Miscellaneous Files.....	1-16
1.5 Environment Variables Used with SunNet Manager....	1-17

1.6	Extending SunNet Manager	1-20
2.	SunNet Manager Console	2-1
2.1	Control Panel Buttons and Menus	2-1
2.1.1	File Button	2-2
2.1.2	View Button	2-8
2.1.3	Edit Button	2-12
2.1.4	Props Button	2-18
2.1.5	Requests Button.	2-21
2.1.6	Tools Button	2-23
2.1.7	Goto Button	2-24
2.2	Element Glyph Menu.	2-26
2.2.1	Quick Dump	2-27
2.2.2	Send Request	2-28
2.2.3	Send Predefined	2-28
2.2.4	Set Request.	2-28
2.2.5	Alarm Reports	2-28
2.2.6	Tools	2-30
2.2.7	Glyph State	2-31
2.2.8	Show Subview	2-32
2.2.9	Properties	2-34
2.2.10	Connect	2-39
2.2.11	Change Type	2-41
2.2.12	Auto Manage Off	2-42
2.3	Viewing and Modifying Requests	2-44

2.3.1	Request Viewer	2-44
2.3.2	Request Glyphs	2-48
3.	Data and Event Requests.....	3-1
3.1	Using Quick Dump to Send a Data Report.....	3-2
3.1.1	Quick Dump Option via the Requests Button.....	3-2
3.1.2	Quick Dump Option via the Glyph Menu.....	3-4
3.2	Predefined Requests Supplied with SNM 2.2.3	3-5
3.2.1	Predefined Data Request Records.....	3-6
3.2.2	Predefined Event Request Records.....	3-7
3.3	Creating, Modifying, or Deleting Predefined Requests..	3-7
3.3.1	Creating, Modifying, or Deleting a Predefined Data Request	3-9
3.3.2	Creating, Modifying, or Deleting a Predefined Event Request	3-16
3.4	Sending Data and Event Requests	3-23
3.4.1	Sending a Data Request	3-23
3.4.2	Sending an Event Request	3-32
3.5	Sending Predefined Data and Event Requests	3-36
3.5.1	Sending a Predefined Data Request	3-36
3.5.2	Sending a Predefined Event Request	3-37
4.	Viewing Reports and Logs	4-1
4.1	Data Reports	4-1
4.1.1	Data Reports Log	4-1
4.1.2	Indicators	4-3
4.1.3	Strip Charts	4-4

4.1.4	Graph Tool	4-6
4.2	Event Reports	4-6
4.2.1	Event/Trap Reports Log	4-7
4.2.2	Errors Log	4-8
4.2.3	Events Summary	4-9
4.3	Viewing Alarm Reports	4-9
4.3.1	Finding a Device	4-11
4.3.2	Switching Views	4-13
4.3.3	Obtaining Device-specific Alarm Reports	4-14
4.3.4	Features of Device-specific Alarm Reports Window	4-16
5.	Discover Tool	5-1
5.1	Invoking Discover	5-2
5.2	Discover Tool Configuration	5-5
5.2.1	Discover Function Configuration	5-7
5.2.2	Example: Using Discover to Create a Routers-only View	5-12
5.2.3	Monitor Function Configuration	5-14
5.3	snm_discover Command	5-18
5.3.1	Discover Command-Line Options	5-18
5.3.2	Discovery on Non-Subnetted Class B Networks ..	5-21
5.4	The discover.conf File	5-21
5.4.1	OID Section	5-23
5.4.2	COLORS Section	5-23
5.4.3	COMPONENTS Section	5-24

5.4.4	DEFAULTS Section	5-24
5.4.5	MAPPINGS Section	5-25
5.4.6	MONITOR COMPONENTS Section.	5-26
5.4.7	AGENTS Section	5-26
6.	Results Browser.	6-1
6.1	Starting the Results Browser	6-2
6.2	Loading Files.	6-6
6.3	Report Streams	6-7
6.4	Streams Menu	6-11
6.5	Selecting Streams	6-12
6.5.1	Sending Data to the Results Grapher	6-13
6.6	Folders	6-15
6.7	Customizing the Browser	6-17
7.	Results Grapher	7-1
7.1	Starting the Results Grapher.	7-2
7.2	Results Grapher Window	7-3
7.3	Graph Properties Window	7-4
7.4	Displaying Graphs	7-8
7.5	Merging Graphs	7-11
8.	Set Tool.	8-1
8.1	Set Tool Window	8-4
8.2	Set Information List	8-5
8.3	Invoking Set Tool from the Command Line	8-6

9. Console Properties	9-1
9.1 SunNet Manager Console Properties.....	9-2
9.2 Windows	9-5
9.3 Requests.....	9-6
9.4 Automatic Management	9-7
9.5 Events and Traps	9-12
9.6 Errors	9-18
9.7 Locations	9-21
9.8 Miscellaneous	9-22
9.9 Other Configuration.....	9-24
9.9.1 Forking Programs on Element Creation or Modification	9-25
9.9.2 Redirecting SNMP Requests	9-26
9.9.3 Activating Console Database Manager Traps for File Loads.....	9-26
10. Management Database.....	10-1
10.1 Element Type Definition	10-3
10.1.1 Glyph Definition (for Component or View Element Type)	10-6
10.1.2 User Command Definition for an Element Type ..	10-8
10.2 Element Instance Definition	10-9
10.3 Connection Definition	10-12
10.4 Background Definition.....	10-12
10.5 Tools Menu Definition	10-13
10.6 Definition of Requests	10-14

10.6.1	Data Requests	10-15
10.6.2	Event Requests	10-18
11.	SNMP Support	11-1
11.1	SNMP Proxy Agent Operation	11-2
11.2	Creating a Schema File.....	11-7
11.3	Creating an SNMP Host File.....	11-9
11.4	Asynchronous Event Reports (Traps)	11-11
11.4.1	Trap Daemon Operation.....	11-12
11.4.2	SNMP Trap File	11-14
11.4.3	Trap Reports on the Console	11-23
11.4.4	Interpreting a Trap Report	11-24
11.5	Mapping SNMP Object IDs to Strings	11-25
12.	SNMP Version 2 Support.....	12-1
12.1	SNMP Version 2 Support.....	12-1
12.1.1	SNMPv2 Enhancements.....	12-2
12.2	SNMPv2Files.....	12-4
12.3	Using the v2mib2schema Program.....	12-4
13.	Network Management Security	13-1
13.1	Authentication	13-2
13.2	Access Control	13-2
13.3	The Security Algorithm.....	13-3
13.4	Conferring Right-of-Access.....	13-4
A.	Error Messages	A-1
A.1	If Agents or Daemons Don't Start	A-2

A.2	NetISAM Messages	A-3
A.3	Types of SNM Errors	A-4
A.4	SNM Error Messages	A-5
B.	Man Page Summary	B-1
B.1	Setting the MANPATH on Solaris 1.x	B-1
B.2	Setting the MANPATH on Solaris 2.x	B-2
B.3	User Commands.....	B-2
B.4	File Formats.....	B-4
B.5	Agents and Daemons.....	B-5
	Glossary	Glossary-1
	Index	Index-1

Figures

Figure 1-1	SunNet Manager Console Block Diagram	1-3
Figure 1-2	Using Proxy Agents for Distributed Network Management	1-7
Figure 2-1	SunNet Manager Console Window	2-2
Figure 2-2	File Menu	2-3
Figure 2-3	File—Load and Save Window	2-4
Figure 2-4	File—Load—Management Database Window	2-5
Figure 2-5	File—About Window	2-8
Figure 2-6	View Button Menu	2-9
Figure 2-7	View—Find Window	2-11
Figure 2-8	Edit Button Menu	2-12
Figure 2-9	Edit—Create	2-15
Figure 2-10	Edit—Create—Element Properties Window	2-16
Figure 2-11	Element Properties Window	2-19
Figure 2-12	Console Properties Window	2-20
Figure 2-13	Requests Menu	2-21
Figure 2-14	Tools Menu	2-23

Figure 2-15	Sample Goto Menu	2-25
Figure 2-16	Glyph Menu	2-27
Figure 2-17	Alarm Reports Window	2-29
Figure 2-18	Glyph Menu—Tools	2-30
Figure 2-19	Glyph Menu—Glyph State.	2-32
Figure 2-20	Sample Show Subview Window	2-33
Figure 2-21	Glyph Menu—Element Properties Window	2-34
Figure 2-22	Glyph Menu—Properties—Alias Window	2-38
Figure 2-23	Glyph Menu—Connect Window	2-40
Figure 2-24	Connection Created Between two Elements	2-41
Figure 2-25	Glyph Menu—Change Type Window	2-42
Figure 2-26	Glyph Menu—Auto Manage Off	2-43
Figure 2-27	Requests Menu—Requests Summary Window	2-45
Figure 2-28	Selecting Requests	2-47
Figure 2-29	Request Glyph Menu	2-49
Figure 2-30	Sample Request Properties Window	2-50
Figure 3-1	Requests — Quick Dump Request Window	3-3
Figure 3-2	Sending a Quick Dump Request	3-4
Figure 3-3	Quick Dump Report Pop-Up Window.	3-5
Figure 3-4	Requests — Create Predefined Request Window.	3-8
Figure 3-5	Sample Predefined Data Request Template	3-10
Figure 3-6	Sample Predefined Event Request Template	3-17
Figure 3-7	Glyph — Send Data Request Window	3-24
Figure 3-8	Requests — Send Data Request Window	3-25
Figure 3-9	Data Request Template.	3-26

Figure 3-10	Glyph Menu — Send Event Request Window	3-33
Figure 3-11	Requests — Send Event Request Window.	3-34
Figure 3-12	Event Request Template.	3-35
Figure 3-13	Sample Glyph — Send Predefined Data Request Menus	3-37
Figure 3-14	Sample Glyph — Send Predefined Event Request Menus . . .	3-38
Figure 4-1	View — Data Reports Window.	4-2
Figure 4-2	Strip Chart	4-4
Figure 4-3	Strip Chart Properties.	4-5
Figure 4-4	View — Event/Trap Reports.	4-7
Figure 4-5	Alarm Reports Summary Window.	4-10
Figure 4-6	Alarm Reports Find Window	4-12
Figure 4-7	Alarm Reports: Show View Window.	4-13
Figure 4-8	Device-specific Alarm Reports Window	4-15
Figure 4-9	Device-specific Alarm Reports View Menu	4-16
Figure 4-10	Device-specific Alarm Reports Filter Window	4-17
Figure 4-11	Device-specific Alarm Reports Sort Menu.	4-18
Figure 4-12	Device-specific Alarm Reports Find Window.	4-19
Figure 4-13	Save to Logfile Window	4-20
Figure 5-1	Discover Tool Base Window	5-3
Figure 5-2	Discover Configuration Window: Discover Properties.	5-6
Figure 5-3	Discover Configuration Category Menu	5-7
Figure 5-4	Subview Created without Coordinates	5-11
Figure 5-5	Example of Routers-Only Discover Configuration	5-13
Figure 5-6	Subview in Routers-only Hierarchy	5-14
Figure 5-7	Discover Configuration: Monitor Window	5-15

Figure 6-1	Invoking the Browser from the Console	6-3
Figure 6-2	Results Browser Window.....	6-5
Figure 6-3	Load Window.....	6-6
Figure 6-4	Results Browser Report Streams.....	6-7
Figure 6-5	Agent Reports from Selected Stream	6-9
Figure 6-6	Report Menu.....	6-10
Figure 6-7	Results Browser Streams Menu.....	6-12
Figure 6-8	Streams Selection by System	6-13
Figure 6-9	Sending Data to the Grapher	6-14
Figure 6-10	Copying Streams to a Folder	6-16
Figure 6-11	Tool Properties Window	6-18
Figure 7-1	Invoking the Grapher from the Console	7-2
Figure 7-2	Results Grapher Window.....	7-4
Figure 7-3	Graph Properties	7-5
Figure 7-4	Graph.....	7-9
Figure 7-5	Controlling Grapher Rotation Angles	7-10
Figure 8-1	Set Request Menus	8-2
Figure 8-2	Set Tool Window	8-3
Figure 9-1	Selecting Props Button	9-2
Figure 9-2	Console Properties Window	9-3
Figure 9-3	Console Properties Window Categories	9-4
Figure 9-4	Console Properties Requests Category	9-6
Figure 9-5	Console Properties Automatic Management Category.....	9-8
Figure 9-6	Console Properties Events and Traps Category	9-13
Figure 9-7	Events and Traps—Upon Opening Menu.....	9-14

Figure 9-8	Console Properties Errors Category	9-19
Figure 9-9	Console Properties Locations Category	9-21
Figure 9-10	Console Properties Miscellaneous Category	9-22
Figure 11-1	SNMP Proxy Agent	11-2
Figure 11-2	MIB and Schema Definitions	11-3
Figure 11-3	Example of Glyphs for Pseudo-devices	11-17
Figure 11-4	Sample Properties Sheet for Pseudo-Device	11-22
Figure 11-5	Trap Report.	11-24

Tables

Table 1-1	Summary of SNM 2.2.2 Agents and Proxies	1-9
Table 1-2	Agents Specific to Solaris 2.x	1-11
Table 1-3	Summary of Default SNM File Locations	1-19
Table 2-1	Glyphs Used for Multiple Element Types	2-17
Table 2-2	Summary of Edit Operations	2-17
Table 3-1	SNM Supplied Predefined Data Requests	3-6
Table 3-2	SNM Supplied Predefined Event Requests	3-7
Table 10-1	Data Request — dataRequest Record	10-15
Table 10-2	Data Request—dataAttribute Record	10-17
Table 10-3	Data Request — rqstState Record	10-17
Table 10-4	Data Request — membership Record	10-18
Table 10-5	Event Request — eventRequest Record	10-19
Table 10-6	Event Request — eventAttribute Record	10-20
Table 10-7	Event Request — rqstState Record	10-21
Table 10-8	Event Request — membership Record	10-21
Table 11-1	Supported SNMP and Schema Types	11-23

Table A-1	Common NetISAM Errors	A-4
-----------	---------------------------------	-----

Preface

The *SunNet Manager 2.2.3 Reference Manual* provides information on SunNet Manager™ functions and features.

Who Should Use This Book

The document is intended both for first-time and more experienced SunNet Manager users.

Before You Read This Book

If you have just acquired the product, you should read the *SunNet Manager 2.2.3 User's Guide* to learn how to use basic product features. As you gain experience with SunNet Manager (SNM™), the *SunNet Manager 2.2.3 Reference Manual* will provide additional information about the product.

How This Book Is Organized

This document is organized as follows:

Chapter 1, “Introduction,” provides a high-level description of SunNet Manager and its underlying architecture.

Chapter 2, “SunNet Manager Console,” describes the functions of the SunNet Manager Console.

Chapter 3, “Data and Event Requests,” describes how to create and send normal and predefined data and event requests.

Chapter 4, “Viewing Reports and Logs,” describes how to view the reports and logs that are generated as a result of data and event requests being sent.

Chapter 5, “Discover Tool,” describes the Discover Tool.

Chapter 6, “Results Browser,” describes the Browser Tool.

Chapter 7, “Results Grapher,” describes the Grapher Tool.

Chapter 8, “Set Tool,” describes the Set Tool.

Chapter 9, “Console Properties,” describes the configuration of the SunNet Manager Console.

Chapter 10, “Management Database,” describes data definitions in the SunNet Manager Management Database.

Chapter 11, “SNMP Support,” discusses SunNet Manager SNMP support.

Chapter 12, “SNMP Version 2 Support,” discusses SunNet Manager SNMP Version 2 support.

Chapter 13, “Network Management Security,” presents guidelines for using SunOS security and authentication mechanisms with SunNet Manager.

Appendix A, “Error Messages,” describes the error messages that can occur while running the SunNet Manager.

Appendix B, “Man Page Summary,” lists and summarizes the on-line man pages for SunNet Manager commands, utilities, file formats, agents and daemons.

A “Glossary” provides a glossary of terms used throughout the SunNet Manager documents.

Compatibility

See the *SunNet Manager 2.2.3 Important Product Information (IPI)* for compatibility information.

Conventions Used in This Book

Command Line Examples

All command line examples in this guide use the C-shell environment. If you use either the Bourne or Korn shells, refer to `sh(1)` and `ksh(1)` man pages for command equivalents to the C-shell.

What Typographic Changes and Symbols Mean

The following table describes the type changes and symbols 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 <code>.login</code> file. Use <code>ls -a</code> to list all files. system% You have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	system% su Password:
<AaBbCc123>	Command-line placeholder: replace with a real name or value	To delete a file, type <code>rm</code> <filename>.
<i>AaBbCc123</i>	Book titles, new words or terms, or words to be emphasized	These are called <i>class</i> options. You <i>must</i> be root to do this.
Code samples are included in boxes and may display the following:		
%	UNIX C shell prompt	system%
\$	UNIX Bourne and Korn shell prompt	system\$
#	Superuser prompt, all shells	system#

Introduction



<i>SunNet Manager Architecture</i>	<i>page 1-2</i>
<i>Console</i>	<i>page 1-4</i>
<i>Agents and Proxies</i>	<i>page 1-6</i>
<i>SunNet Manager Directories and Files</i>	<i>page 1-13</i>
<i>Environment Variables Used with SunNet Manager</i>	<i>page 1-17</i>
<i>Extending SunNet Manager</i>	<i>page 1-20</i>

The number and size of computer networks is growing at an ever increasing rate, and there is a critical need for tools to manage them. More products are becoming available, increasing the heterogeneity of the networks. Managing a network means managing devices from different manufacturers—often with very different architectures.

A successful network management solution must be flexible enough to solve today's management challenges in a broad variety of networking environments. It must also continue to address future challenges as the networks (and their uses) grow and change.

SunNet Manager (SNM) is a platform for the management of distributed work group networks. The product features mechanisms for extending the platform, providing the ability to continue effective management as requirements change. It also features a collection of tools immediately useful for management.

This chapter introduces SunNet Manager features and presents some of the underlying concepts and architecture in order to help both engineers interested in extending the product and users trying to manage their networks.

1.1 SunNet Manager Architecture

As a management platform, SunNet Manager relies on the manager/agent model described in the Open Systems Interconnection (OSI) management framework and native Solaris Transmission Control Protocol/Internet Protocol (TCP/IP) communication. The manager is a process initiated by the user. The agent is a process that accesses the managed object (the entity being managed) and collects data on behalf of the manager. Figure 1-1 presents a functional block diagram of SunNet Manager.

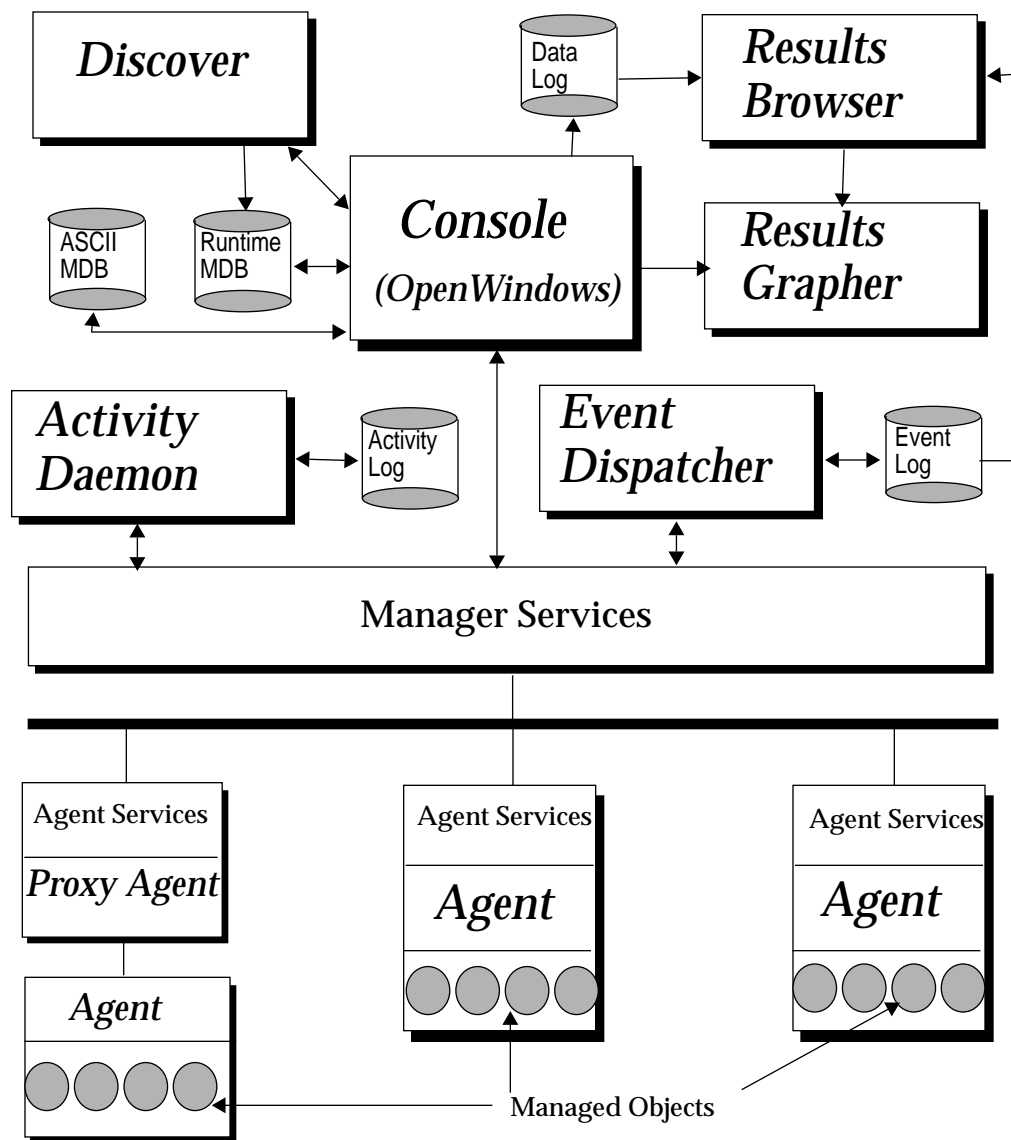


Figure 1-1 SunNet Manager Console Block Diagram

The remaining sections of this chapter discuss the following components of the block diagram:

- The Console, including the management database (MDB) and tools such as Discover, Results Browser, and Results Grapher.
- Agents and proxy agents.
- Ancillary daemons (activity daemon and event dispatcher).

1.2 Console

The SunNet Manager Console is the central management application in the SunNet Manager package—the place where you initiate management tasks and management information is returned.

The SunNet Manager Console presents an object-oriented interface which may be tailored to depict a particular management domain. The Console uses the OPEN LOOK Graphical User Interface running under OpenWindows™ 3.0 or later. OpenWindows supports the X11 protocol, which allows the Console and other application windows to be displayed on a network display that is managed by an X11 server. X terminals must be fully compatible with the MIT X11 release 4 server and use an OPEN LOOK-compliant window manager.

The SNM Console supports X terminals by allowing multiple instances of the Console to run on a single machine at the same time. Each instance of the Console is mapped to the name of the user who invokes it. Each instance of the Console will only work with other SNM tools that use the same user name.

The Console provides mechanisms for initiating requests for *data* reporting and *event* reporting. Data reporting allows you to direct agents to send reports of raw management data on a periodic basis. Event reporting allows you to direct agents to report only when specified conditions are met (that is, when an event occurs).

You direct an agent by making a request in which you specify the parameters for a desired management task. The request contains information on the object to be managed and how often the agent is to report.

When requesting data reports, you can elect to gather data in groups or individually, have data reported immediately or on a deferred basis, and store returned data in files. For event reports, you have many options for setting the condition(s) of the event.

The Console also supports the display of reported data and event indications (including audible, visual, and programmatic mechanisms). Visual changes resulting from predefined event conditions are propagated through the Console so that you can see at a glance if an event has occurred. From the Console, you can also use the Set Tool, a window interface that allows you to change attribute values. Currently, only the Simple Network Management Protocol (SNMP) proxy agent supports Set Tool operations. With the Set Tool, you can request to change (set) the value of one or more attributes in a group or among different groups. The Console can also be extended to support user specified commands as well.

Context sensitive help is available on-line for Console screens through popup Help windows. See Chapter 3 of the *SunNet Manager 2.2.3 Installation Guide* for the requirements for invoking on-line help. Refer to Chapter 2, “SunNet Manager Console,” for detailed information about Console functions.

1.2.1 Management Database

The Console relies on the definitions and information contained in a management database (MDB). Emerging open management standards (for example, OSI and SNMP) specify that agents abstract the properties (or attributes) of managed objects into data items (for example, “how busy a CPU is” may become a value between 0 and 100). In SunNet Manager, the attributes of a managed object are described in a portion of the MDB called the agent schema. The agent is able to respond to the manager’s request because both use the same data definitions for the managed object.

Each *instance* of the Console requires a *separate*, runtime database. In other words, multiple Console instances *cannot* share a single runtime database.

The MDB data and any updates made using the graphical editing capabilities constitute a dynamic runtime database, which may be saved to an ASCII database file at any time. The ASCII format supports easy modification and portability to other systems. Chapter 10, “Management Database,” describes MDB files and how to modify them.

SNM provides an application programming interface (API) that allows an application program to query or modify the runtime database. The API functions are described in the *SunNet Manager 2.2.3 Programmer’s Guide*.

1.2.2 Console Tools

SunNet Manager includes several tools that you can invoke from the Console:

- The Discover Tool seeks out network elements and automatically creates a graphical representation of your network. As it finds elements, Discover adds them to the runtime database. Refer to Chapter 5, “Discover Tool,” for a description of the Discover functions or see the `snm_discover(8)` man page.
- The Results Browser allows you to examine and organize log files. Chapter 6, “Results Browser,” describes Browser functions.
- The Results Grapher allows you to visualize data reports and log file information. You can send data directly from data reports to the Grapher, or you can send log file data from the Browser to the Grapher. Chapter 7, “Results Grapher,” describes Grapher functions.
- The Set Tool allows you to change SNMP attribute values. Refer to Chapter 8, “Set Tool,” for more information about Set Tool.

Context-sensitive help is available on-line for these tools through popup Help windows. See Chapter 3 of the *SunNet Manager 2.2.3 Installation Guide* for information about the environment variable required for SNM on-line help.

1.3 Agents and Proxies

There are two types of SunNet Manager agents: those that directly access managed objects and those that indirectly access managed objects. Most of the agents provided with this release manage objects on the Sun workstations where they are installed. For example, the `hostmem` agent uses the same mechanism as `netstat -m` to get memory utilization data.

The second type of agent provides the ability to manage objects that reside in other Sun workstations or in other vendors' devices. Such agents are called *proxy agents*. Proxy agents run on Sun workstations, called *proxy systems*, and use protocol translation mechanisms to provide the necessary access to the managed objects. The proxy system may be the workstation on which the SunNet Manager Console is running or another workstation on the network. The proxy system can also be a workstation in a different subnet or domain from where the Console is running. This allows SunNet Manager to extend into virtually any domain. Figure 1-2 illustrates how SNM proxy agents can provide a distributed network management solution.

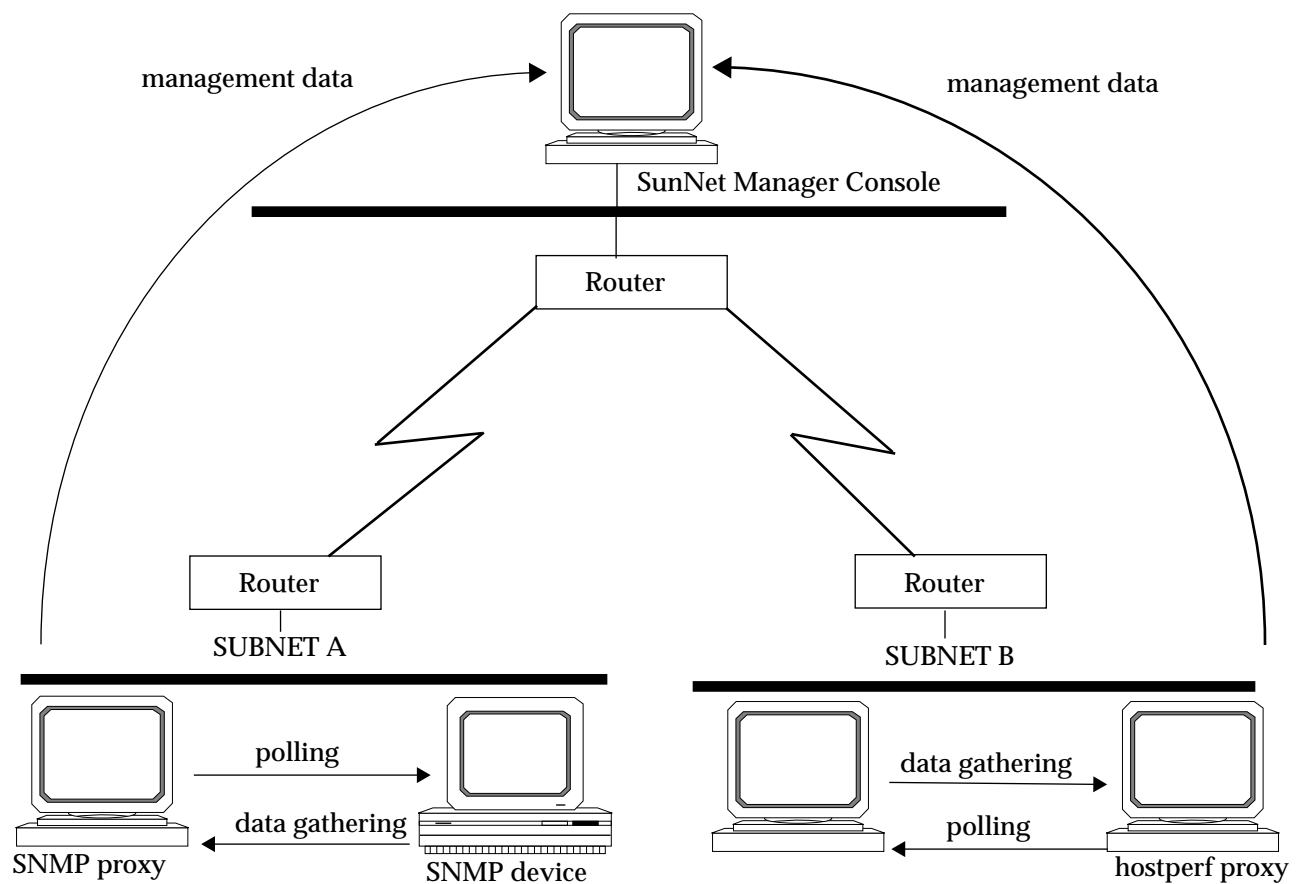


Figure 1-2 Using Proxy Agents for Distributed Network Management

In the figure above, the *hostperf* proxy agent uses the *rstat* protocol to gather statistics for remote hosts, then returns management data to the Console. The Simple Network Management Protocol (SNMP) proxy agent can manage any device that supports SNMP, the standard management protocol for the TCP/IP world. In both cases, the polling of the target devices is confined within the local subnets; only the relevant management information is sent across the wide area networks. This configuration can reduce the amount of traffic between the management station and the managed devices.

The manager and agent communicate through the Manager/Agent Services library, which provides the management infrastructure and handles the communication services. The manager and agent processes use the Services through Application Programming Interfaces (APIs), which in turn use the Remote Procedure Call/External Data Representation (RPC/XDR) capability in Solaris. The agent and manager need not be concerned with the underlying networking involved in their communication. The agent processes need be concerned only with collecting data from the managed object.

When you first make a request to an agent, two agent processes are started: one is a parent process and one is a child process to handle the request. Subsequent requests for the same agent may start additional child processes. The SunNet Manager `hostperf`, `ping`, and `SNMP proxy` agents can handle multiple requests with a single child process. These proxy agents create a new subprocess under the following conditions:

- The proxy agent receives a Quick Dump request. The subprocess handling the Quick Dump exits right after sending the report.
- The proxy agent receives a request with a different manager ID, manager IP address, rendezvous IP address, rendezvous RPC program number, or rendezvous RPC version number from any request it is currently handling.
- The maximum number of requests that a subprocess can handle is exceeded.
- A subprocess is too busy to handle a new request.

The maximum number of subprocesses that each proxy agent can fork is set by the keyword `na.<proxy>.max-subprocs` in the `snm.conf` file. (The default location for `snm.conf` is `/etc` for the Solaris 1.1.1 version of SNM and in `/etc/opt/SUNWconn/snm` for the Solaris 2.3 version.) At installation, this value is set to 20. The maximum number of requests that a subprocess can handle is set by the keyword `na.<proxy>.max-requests` in the `snm.conf` file. At installation, this value is set to 50. See the `snm.conf(1)` man page for more information.

1.3.1 Agents Included with SNM 2.2.3

A SunNet Manager 2.2.3 Console can manage both SunOS 5.1, SunOS 5.2, and SunOS 4.x clients. All SNM agents that run on SunOS 4.x clients are also available for the Solaris 2.x environment. Note however, that certain of the agents that run on Solaris 2.x clients support different attributes than their SunOS 4.x counterparts. The Solaris 2.x versions of these agents have been

renamed to `<agentname>2`. For example, `na.iostat2` is the Solaris 2.x version of the (SunOS 4.x) agent `na.iostat`. Table 1-1 provides a summary of the agents shipped with SNM 2.2.3.

Table 1-1 Summary of SNM 2.2.3 Agents and Proxies

Name	Description	Related UNIX Command	Type
diskinfo	Reports disk usage information	df	Agent
etherif	Ethernet interface statistics for SunOS 4.x clients	—	SunOS 4.x agent
etherif2	Ethernet interface statistics for Solaris 2.x clients	—	Solaris 2.x agent
hostif	Monitors interfaces that send IP packets	netstat -i	Agent (see note)
hostmem	Memory utilization information for SunOS 4.x clients	netstat -m	SunOS 4.x agent
hostmem2	Memory utilization information for Solaris 2.x clients	netstat -m	Solaris 2.x agent
hostperf	Reports host system performance data	rup and perfmeter	Proxy agent
iostat	Reports input/output statistics for SunOS 4.x clients	iostat	SunOS 4.x agent
iostat2	Reports input/output statistics for Solaris 2.x clients	iostat	Solaris 2.x agent
ippath	Reports IP packet trace information	—	Proxy agent
iproutes	Reports IP route table and statistics	netstat -r	Agent
layers	Protocol layer statistics for SunOS 4.x clients	netstat -rs netstat -s	SunOS 4.x agent
layers2	Protocol layer statistics for Solaris 2.x clients	netstat -rs, netstat -s	Solaris 2.x agent
lpstat	Reports printer status	lpq and lpstat	Proxy agent
ping	IP connectivity information	ping	Proxy agent
rpcnfs	Remote Procedure Call and Network File System statistics	nfsstat	Agent
snmp	Returns information from MIB I-compliant SNMP devices.	—	Proxy agent
snmpv2	For managing SNMP Version 1 and Version 2 devices	—	Proxy agent
snmp-mibII	Returns information from MIB II-compliant SNMP devices.	—	Proxy agent

Table 1-1 Summary of SNM 2.2.3 Agents and Proxies

Name	Description	Related UNIX Command	Type
sun-smnp	Returns MIB I-compliant and Sun-specific information from Sun workstations.	—	Proxy agent
sync	Monitors synchronous serial lines	syncstat	Agent
traffic	Ethernet traffic analyzer	—	Proxy Agent

Note – `snmp`, `snmp-mibII`, and `sun-smnp` are schema files that can be used with the SNMP proxy agent `na.smnp`. For each of the other agents shipped with SNM 2.2.3, there is only one corresponding schema file.

Note – `na.hostif` is the same for SunOS 4.x and Solaris 2.x except that the metric table is inaccessible for Solaris 2.x.

Table 1-2 lists the agents that have substantial differences between their SunOS 4.x and Solaris 2.x versions. Table 1-2 shows the attributes that were added to the Solaris 2.x agents; it also shows attributes from the SunOS 4.x version that were deleted for the Solaris 2.x version of these agents.

Table 1-2 Agents Specific to Solaris 2.x

Agent	SunOS 4.x attributes removed	New attributes added for Solaris 2.x
na.etherif2	input group: ibufs ibufdesc idiscard iframedesc output group: ojams odrops watchdog obufdesc copies nocfs obufs oframedesc obuferr	input group: imis inocanput output group: oinits
na.hostmem2	mbuf group mbuf_uses group streams group: mblkused mblkfree mblk% mblkcom mblkfail dbls table	streams group: msgused msgfree msg% msgcum msgfail lblkused lblkfree lblk% lblkcum lblkfail streveused strevefree streve% strevecum strevefail

Table 1-2 Agents Specific to Solaris 2.x

Agent	SunOS 4.x attributes removed	New attributes added for Solaris 2.x
na.iostat2	disk table: mbytes xfers seeks activeT xferT seekT %xferT avgxferT kbps	disk table: kreads kwrites svcwait acttrans svctime %wait %busy
na.laters2	no changes	no changes

Appendix B, “Man Page Summary,” contains brief descriptions of all the SunNet Manager agents. These agents represent examples of the data gathering that you can accomplish using SunNet Manager, and are not meant to be an exhaustive set. To write your own agent, refer to Part II of the *SunNet Manager 2.2.3 Programmer’s Guide*.

Many Sun products include agents that can be used with SunNet Manager. Refer to your product documentation or call your local Sun representative for more information.

1.3.2 Ancillary Daemons

SunNet Manager includes two ancillary daemons: the activity daemon (na.activity) and the event dispatcher (na.event). The activity daemon is a process that uses the Manager/Agent Services to ensure ongoing requests (activities) continue to be serviced by agents. The activity daemon uses the activity log, a record of active requests started from the Console.

The event dispatcher is a process that uses the Manager/Agent Services to direct reports of events to the proper destinations (*rendezvous process*), such as the SunNet Manager Console. Any other management application can use the event dispatcher, registering to receive some or all event reports based on a variety of selection criteria. The Event Dispatcher also logs all event reports in the Event/Trap Log. The Event/Trap Log is in ASCII format, so you can examine and modify it using standard UNIX tools.

Two copies of the event dispatcher are started when you invoke the Console. When you make the first request to any agent, two copies of the activity daemon are started. When you exit the Console, two copies of `na.activity` and one copy of `na.event` remain.

1.4 SunNet Manager Directories and Files

This section discusses the files and directories that are created during SunNet Manager installation. The Solaris 1.x and Solaris 2.x versions of SunNet Manager differ in the default locations assigned to these files and directories by the installation scripts. The default locations for these files are summarized in Table 1-3.

The `snm.conf` file is the configuration file used by SNM agents and daemons. (For Solaris 1.x installations of SNM, `snm.conf` is located in `/etc`; for Solaris 2.x installations `snm.conf` is located in `/etc/opt/SUNWconn/snm`.) See the `snm.conf(5)` man page for more information about changing keyword values.

The SunNet Manager installation script prompts you for the following locations:

- The directory where the SNM software will be installed. This normally is `/usr/snm` for Solaris 1.x installations; for Solaris 2.x installations the default location is `/opt/SUNWconn/snm`.

If you install SNM into a directory other than the default, you must set the environment variable `SNMHOME` to the directory where SNM is installed before you can use the software.

- The directory where the SNM database will be written. (If you do not specify otherwise, this is `/var/adm/snm` for Solaris 1.x installations; the default location for Solaris 2.x installations is `/var/opt/SUNWconn/snm`.)

Caution – You should have at least 10 Mbytes of space for the initial database directory. You can change the location of the database directory after installation by setting the environment variable `SNMDBDIR` to a writable directory.

- The directory where SNM log files will be written. (If you do not specify otherwise, this is `/var/adm/snm` for Solaris 1.x installations; the default location for Solaris 2.x installations is `/var/opt/SUNWconn/snm`.)

The log files grow during Console operation. This could cause a disk space problem if the `/var` directory is part of the root file system. You can change the location of the log files after installation by specifying fully-qualified path names for the keywords `activity-log`, `event-log`, `monitor-log`, and `request-log` in the `snm.conf` file. Make sure that the new directories are writable by root or mode 777 so that the Console can write to the log files.

1.4.1 Installation Directory Files

If you install the SNM software in the default path (`/usr/snm` for Solaris 1.1.1 installations, `/opt/SUNWconn/snm` for Solaris 2.3 installations), the installation script will create the following directories and files under the default directory:

- `agents` directory, which contains SNM agents and their corresponding schema files. You can add your own agent and schemas to this directory. This directory also contains the files `enterprises.oid`, which contains mappings for enterprise identifiers, and `snmp.oid`, which contains mappings for SNMP MIB I object identifiers.
- `bin` directory, which contains binaries used by SNM. One of the binaries, `snm_asroot`, is undocumented.
- `icons` directory, which contains SNM icon files. You can add your own icons to this directory.
- `include` directory, which contains header files for building agents.
- `lib` directory, which contains SNM libraries.
- `man` directory, which contains SNM man pages.
- `5.x`, which contains the Solaris 2.x version of the SunNet Manager agents, for managing SunOS 5.x machines.
- `src`, which contains sample source code for selected agents and manager applications.
- `struct` directory, which contains the following files:
 - `elements.schema` defines the element types. If you need to add site-specific elements, create your own file with the `.schema` extension (for example, `myelements.schema`).
 - `example.db` is an example database that you can load into the Console.

- `snm.glue` is a file that contains a starting set of definitions for the Console. Do *not* modify this file.

1.4.2 Database Files

A database directory is created in the path specified by the environment variable `SNMDBDIR`. If `SNMDBDIR` is not specified, then the database path specified during installation is used. The database directory is `db.<user_name>`, where `<user_name>` is the value of the environment variable `SNM_NAME`, or the environment variables `LOGNAME` or `USER`, if `SNM_NAME` is not set. The following database files are created for SNM:

- `snm+lock` is the Console lock file; this prevents more than one user from accessing the same runtime database.
- `snmdb+lock` is the database API lock file; this file enables the database to be locked when entries are changed programmatically.
- The files `events.ind`, `events.rec`, `nc.ind`, and `nc.rec` make up the runtime database used by the Console.

1.4.3 Log Files

Log files are created in the directory specified during installation. You can also change the location of individual log files after installation by modifying the appropriate log file entry in the `snm.conf` file. The following log files are created for SNM:

- `activity.log` is a log of active requests started from the SNM Console host; this file is used by the activity daemon.
- `event.log` is the log file written by event dispatcher.
- `monitor.log` is written by the `na.logger` agent. This agent logs data reports launched by the `snm_cmd` command.
- `request.log` is a record of requests that are to be restarted by the appropriate agent; this file is used by the agents.

1.4.4 Miscellaneous Files

- The `snmp.hosts` file contains information about SNMP hosts. (The default location of the `snmp.hosts` file for Solaris 1.x installations is `/var/adm/snm`; for Solaris 2.x installations the default location is `/var/opt/SUNWconn/snm`.) You can add entries into this file to specify enterprise-specific host information.
- The `snmp.trapfile` file contains information about enterprise-specific traps. (For Solaris 1.x installations the default location is `/var/adm/snm`; for Solaris 2.x installations, the default directory is `/var/opt/SUNWconn/snm`.) You can add entries into this file to specify enterprise-specific trap information.
- The `$HOME/.SNMdefaults` file contains information pertaining to SNM Console properties. For example, when the icon size is set to a new value (i.e., 32X32), or the schema path changed, these values and other properties are written to your `$HOME/.SNMdefaults` file. If you subsequently remove SNM, then reinstall it (e.g., migrating from SNM 2.0 to 2.2.3), the old `.SNMdefaults` values will continue to be used. You can modify the values by using the SNM Console Props button, or by editing your `.SNMdefaults` file. You can ensure that the SNM 2.2.3 Console property defaults are used by moving a preexisting `.SNMdefaults` file from your home directory or deleting your `$HOME/.SNMdefaults` file.
- The `.SNMpredefined` file contains predefined data and event request records provided with the SNM 2.2.3 product. (The default location of this file for Solaris 1.x installations is `/usr/snm/struct`; for Solaris 2.x installations, the default path is `/opt/SUNWconn/snm/struct`.) For information about creating and sending predefined data and event requests, refer to Chapter 3 in the *SunNet Manager 2.2.3 Reference Manual*. When you modify or create predefined data and event requests, they are stored in your `$HOME/.SNMpre-defined` file along with the predefined requests provided with this product.
- The `linkmap` file is an ASCII file that contains information used by the SNM Console to provide link management capabilities. (The default location of the `linkmap` file is `/var/adm/snm` for Solaris 1.x installations and `/var/opt/SUNWconn/snm` for Solaris 2.x installations.) It is either created by the network manager or the Discover Tool; the Discover Tool will populate this file with discovered links (if 2 or more hops are specified as the scope of a network discovery operation). Refer to the `linkmap` (5) manual page for information about the format and use of the `linkmap` file.

1.5 Environment Variables Used with SunNet Manager

You can use the following environment variables with SunNet Manager:

- The `PATH` environment variable should include `<installation_path>/snm/bin` where:
 - `<installation_path>` will be `/usr` if SNM is installed in the default location for SunOS 4.x installations.
 - `<installation_path>` will be `/opt/SUNWconn` if SNM is installed in the default location for the Solaris 2.x version.
- If the SNM 2.2.3 software is installed in a directory *other* than the default location (`/usr/snm` for Solaris 1.x, `/opt/SUNWconn/snm` for Solaris 2.x), you *must* set the `SNMHOME` environment variable to the installation directory.
- The `HELPPATH` environment variable should include `<installation_path>/snm/help` where:
 - `<installation_path>` will be `/usr` if the default location is used for a Solaris 1.x installation.
 - `<installation_path>` will be `/opt/SUNWconn` if the default location is used for a Solaris 2.x installation.

This enables you to get SunNet Manager on-line help.

- The `MANPATH` environment variable should include `<installation_path>/snm/man` where:
 - `<installation_path>` will be `/usr` if the default location is used for a Solaris 1.x installation.
 - `<installation_path>` will be `/opt/SUNWconn` if the default location is used for a Solaris 2.x installation.

This enables use of the SunNet Manager man pages.

- The `SNM_NAME` environment variable can be used to allow multiple instances of the Console and database to be run from a single user ID. One or more names can be specified with `SNM_NAME`. Each instance of the Console is associated with its own runtime database, located in the `db.<snm_name>` directory, where `<snm_name>` is a name specified by the environment variable `SNM_NAME`. Each instance of the Console will use the user's `$HOME/.SNMdefaults` and `$HOME/.SNMpre-defined` files.

- The `SNMDBDIR` environment variable is used to specify the directory where the predefined data and event request ASCII data record file resides.
`/var/adm/snm` is the default directory for the Solaris 1.x version of SNM 2.2.3; the default location for Solaris 2.x is `/var/opt/SUNWconn/snm`.
- The `SNMLINKMAP` environment variable is used to specify the directory where the `linkmap` file used by the link management feature is located. If this environment variable is undefined, the default location is:
 - `/var/adm/snm` for the Solaris 1.x version
 - `/var/opt/SUNWconn/snm` for the Solaris 2.x version.
- The `SNMDISCOVERMAP` environment variable is used to specify the directory where the `discover.conf` file, used by the Discover Tool, is located.
`/var/adm/snm` is the default directory for Solaris 1.x installations;
`/var/opt/SUNWconn/snm` is the default location for Solaris 2.x installations.

Table 1-3 Summary of Default SNM File Locations

Files	Default Location in Solaris 1.x Version	Default Location in Solaris 2.x Version
Agents and schema files	/usr/snm/agents	/opt/SUNWconn/snm/agents
Agents for Solaris 2.x machines	/usr/snm/5.x	/opt/SUNWconn/snm/5.x
discover.conf file	/var/adm/snm	/var/opt/SUNWconn/snm
elements.schema file	/usr/snm/struct	/opt/SUNWconn/snm/struct
Example database	/usr/snm/struct	/opt/SUNWconn/snm/struct
Header files	/usr/snm/include	/opt/SUNWconn/include
Icon files	/usr/snm/icons	/opt/SUNWconn/snm/icons
linkmap file	/var/adm/snm	/var/opt/SUNWconn/snm
Log files	/var/adm/snm	/var/opt/SUNWconn/snm
Management database	/var/adm/snm	/var/opt/SUNWconn/snm
Manager Services libraries	/usr/snm/lib	/opt/SUNWconn/snm/lib
man pages	/usr/snm/man	/opt/SUNWconn/snm/man
messages file	/var/adm/snm	/var/opt/SUNWconn/snm
Predefined data/ event request file	/var/adm/snm	/var/opt/SUNWconn/snm
On-line help files	/usr/snm/help	/opt/SUNWconn/snm/help
Sample agent and application source code files	/usr/snm/src	/opt/SUNWconn/src
snm.conf file and snmpd.conf file	/etc	/etc/opt/SUNWconn/snm
.SNMdefaults file	\$HOME	\$HOME
SNM executables	/usr/snm/bin	/opt/SUNWconn/snm/bin
snm.glue file	/usr/snm/struct	/opt/SUNWconn/snm/struct

Table 1-3 Summary of Default SNM File Locations

Files	Default Location in Solaris 1.x Version	Default Location in Solaris 2.x Version
.SNMpredefined file	/usr/snm/struct	/opt/SUNWconn/snm/struct
snmp.hosts file	/var/adm/snm	/var/opt/SUNWconn/snm
snmp.traps file	/var/adm/snm	/var/opt/SUNWconn/snm

1.6 Extending SunNet Manager

The SunNet Manager infrastructure provides a platform extensible to any application. The SunNet Manager Console user interface is an application that uses the underlying services. SunNet Manager also includes a command-line mechanism for initiating data and event reporting called `snm_cmd(1)`.

You will probably want to extend the capabilities of SunNet Manager. Some of the mechanisms to support this are discussed later in this document. Refer to the *SunNet Manager 2.2.3 Programmer's Guide* to implement the extensions you need.

<i>Control Panel Buttons and Menus</i>	<i>page 2-1</i>
<i>Element Glyph Menu</i>	<i>page 2-26</i>
<i>Viewing and Modifying Requests</i>	<i>page 2-44</i>

This chapter describes the following components of the SunNet Manager Console:

- Control panel buttons and menus.
- The element Glyph menu.
- Data request Properties window.
- Event request Properties window.
- Viewing and modifying request results.

2.1 Control Panel Buttons and Menus

The SunNet Manager Console base window contains the following “control panel” buttons: File, View, Edit, Props, Requests, Tools, and Goto. They are discussed in the following sections. The base SunNet Manager Console window is shown in Figure 2-1.

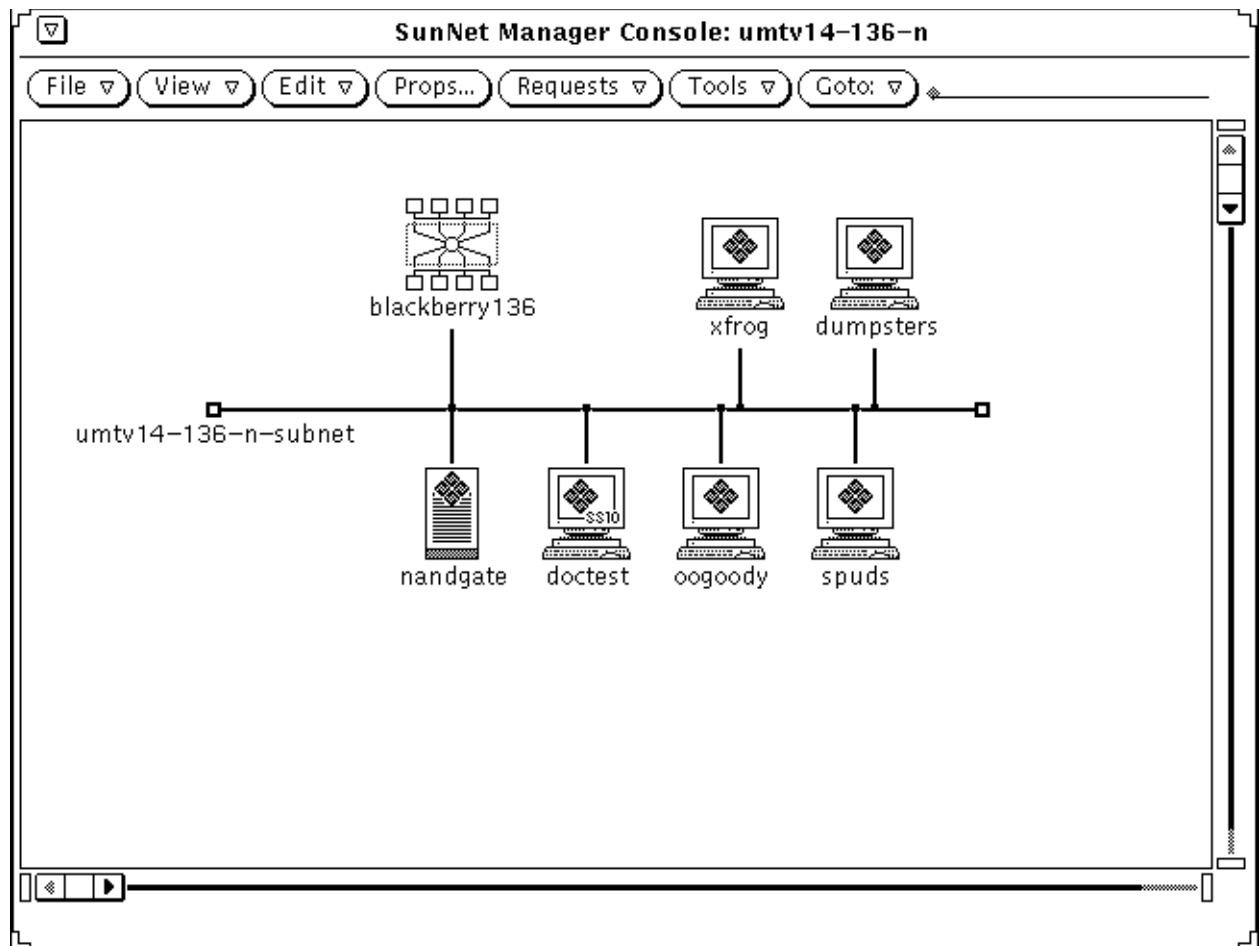


Figure 2-1 SunNet Manager Console Window

2.1.1 File Button

The File button menu allows you to load (read) management database (MDB) or schema files, and predefined request records, or save predefined requests, or instances in the runtime database to a file. The File menu is shown in Figure 2-2.

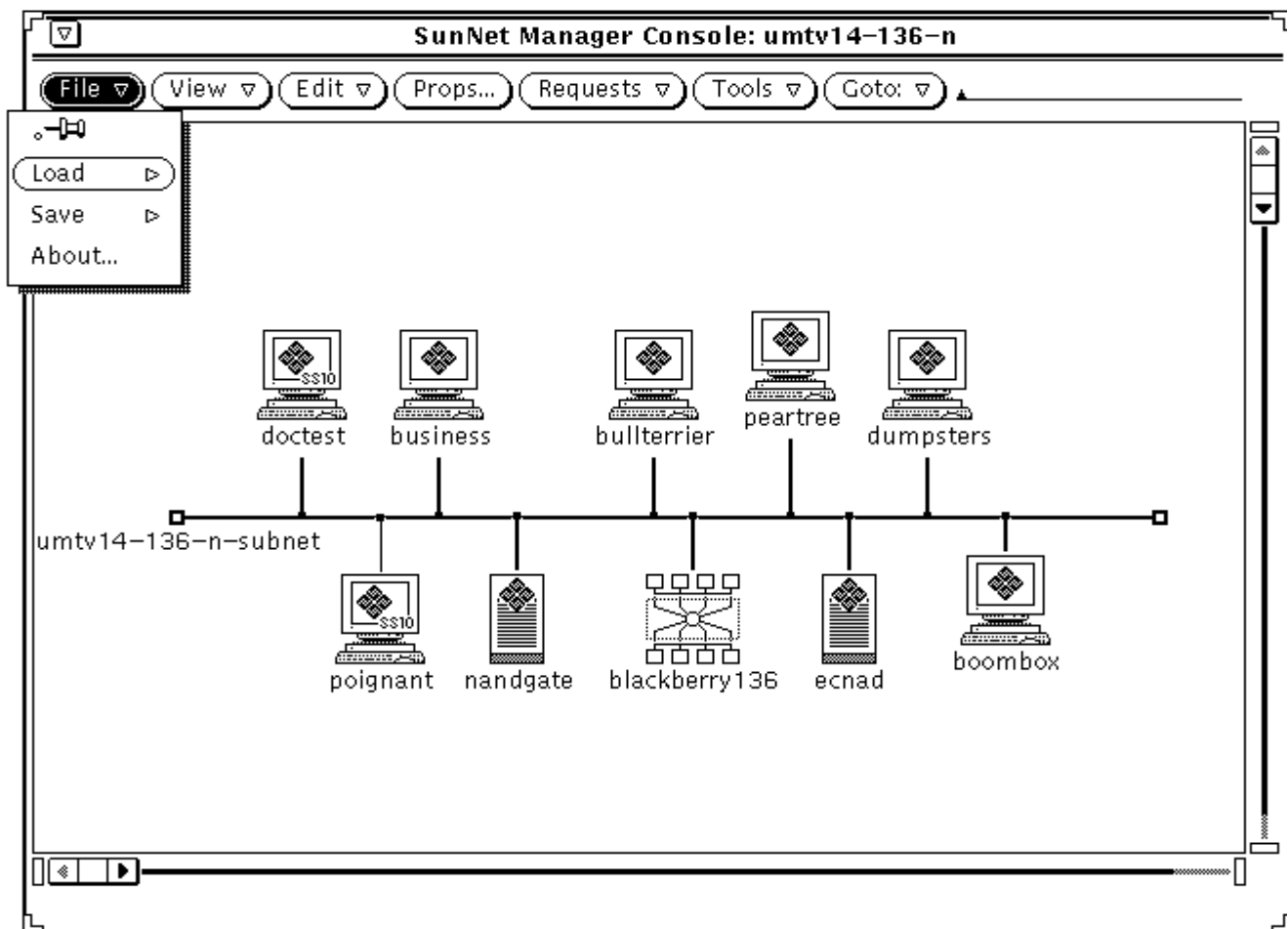


Figure 2-2 File Menu

2.1.1.1 Load and Save Options

The Load and Save options bring up windows that allow you to accomplish the following tasks:

- specify a file to load (read) into the runtime database
- load a predefined data and/or event request file into the runtime database.

- save the runtime database version of the predefined data and event request file into an ASCII-format file.
- save the runtime database into an ASCII-format file.

The File button's Load and Save menus are shown in Figure 2-3.

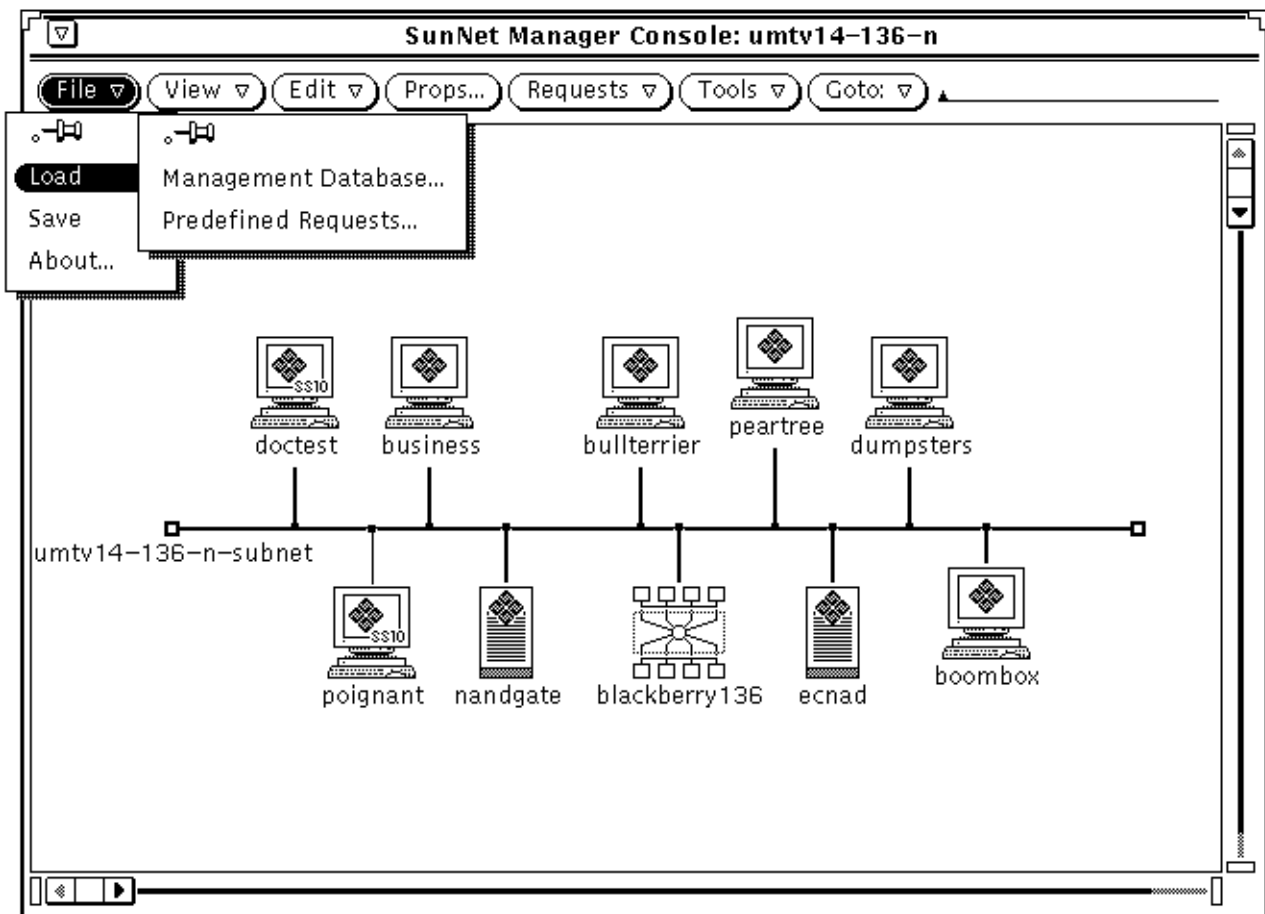


Figure 2-3 File—Load and Save Window

2.1.1.2 Load/Save Management Database Option

The Load and Save options' Management Database window is shown in Figure 2-4.

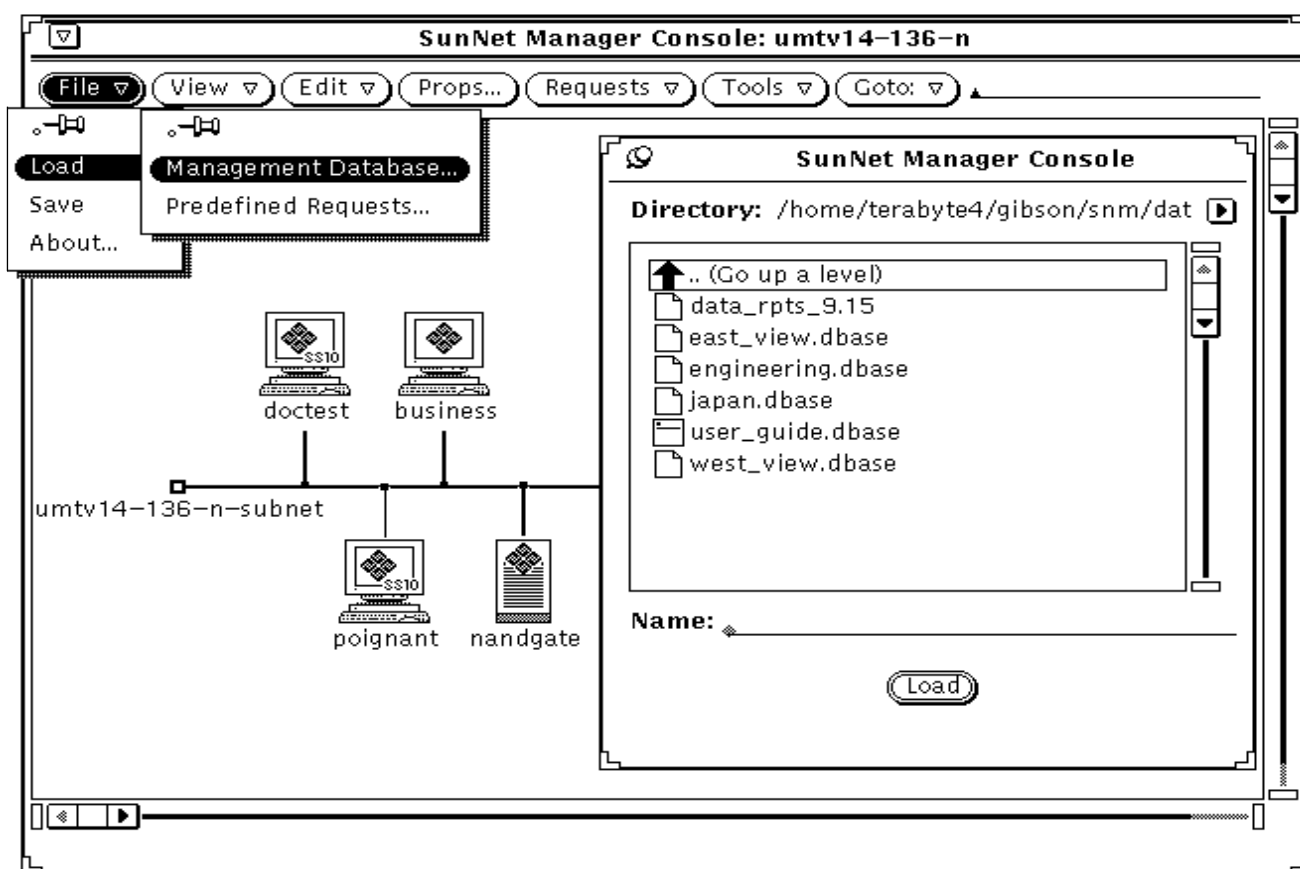


Figure 2-4 File—Load—Management Database Window

The directory from which you started the SunNet Manager Console is supplied as the default directory in the Directory field, and its contents are displayed in the scrolling list. To load a file from the scrolling list, double-click SELECT on the file name or click SELECT on the file name, and then click SELECT on the Load button. In the Name field, you can specify a file name in the default directory, a fully-qualified file name, or an alternate directory path. If you specify a directory path, the directory becomes the default directory, and its contents are displayed in the scrolling list.

If the parser detects an error (for example, bad syntax) while loading an MDB file, it displays error messages in the Load/Save window footer and backs out all changes made to the runtime database for the particular file. You must fix and reload the entire file.

Note – The file that you attempt to load must be an ASCII-format file (for example, a runtime database saved using the File►Save►Management Database option). If you attempt to load a non-ASCII file, an error message is displayed in the Console window footer:

`<file name>: Not Readable by SunNet Manager...`

You can use the File►Save►Management Database option to save the current version of the SNM Console's runtime database to a file. This is especially useful if you have made modifications to the Console and you wish to save the modified Console's runtime database for future use. If for instance, the machine you are running the Console on goes down and you have saved the runtime database to a file via the File►Save►Management Database option, you can reinvoke SNM and either specify the saved runtime database file using the `-i` option, or you can use the File►Load►Management Database option to load the modified version you saved earlier once you have invoked the Console.

Note – There is no way to unload the runtime database. To use another database, you must quit the Console session, then restart the Console with the `-i` option.

2.1.1.3 Load Predefined Requests Option

Select the File►Load►Predefined Requests menu option to load a file containing predefined data and/or event request records into the runtime database. This option is intended for use by knowledgeable users who have a file containing their own predefined data and event requests records that they prefer to use instead of, or in conjunction with, the predefined data and event requests provided with the SNM 2.2.3 product. Refer to Section 3.2, "Predefined Requests Supplied with SNM 2.2.3," on page 3-5 for more detailed information about the predefined data and event requests provided with the SNM 2.2.3 product. Refer to Section 3.3, "Creating, Modifying, or Deleting Predefined Requests," on page 3-7 for a complete description on how to build, modify, or delete predefined data and event requests.

Note – To load files containing predefined event or data requests, these files must be ASCII-format files. For example, predefined requests that are saved using the File►Save►Predefined Requests menu option are saved in ASCII format. If you attempt to load a non-ASCII file, an error message is displayed in the footer of the Console window:

`<file name>: Not Readable by SunNet Manager...`

Warning – Attempting to load a file containing predefined data and/or event request records that have duplicate names of records already existing in the runtime database will cause the “Load of <filename> failed -- see error report for details” error message to appear. You can view the error report via the View►Error Reports menu option. Upon running into this problem, you must use your favorite text editor to remove or rename the duplicate predefined data and/or event record name(s) from the file being loaded. Once accomplished you can use the File►Load►Predefined Requests menu option to load the file.

2.1.1.4 *Save Predefined Requests Option*

Select the File►Save►Predefined Requests menu option to save the runtime database version of the predefined data and event requests records to the ASCII file you specify. Refer to Section 3.2, “Predefined Requests Supplied with SNM 2.2.3,” on page 3-5 for a complete description of predefined data and event requests.

2.1.1.5 *About Option*

The About option provides a SunNet Manager Bulletin Board containing information pertaining to the current release of the SNM product. The information provided by the SNM Bulletin Board includes the product’s version number, the available software packages, and the product’s documentation titles. Upon clicking on the About button, the window shown in Figure 2-5 will be displayed.

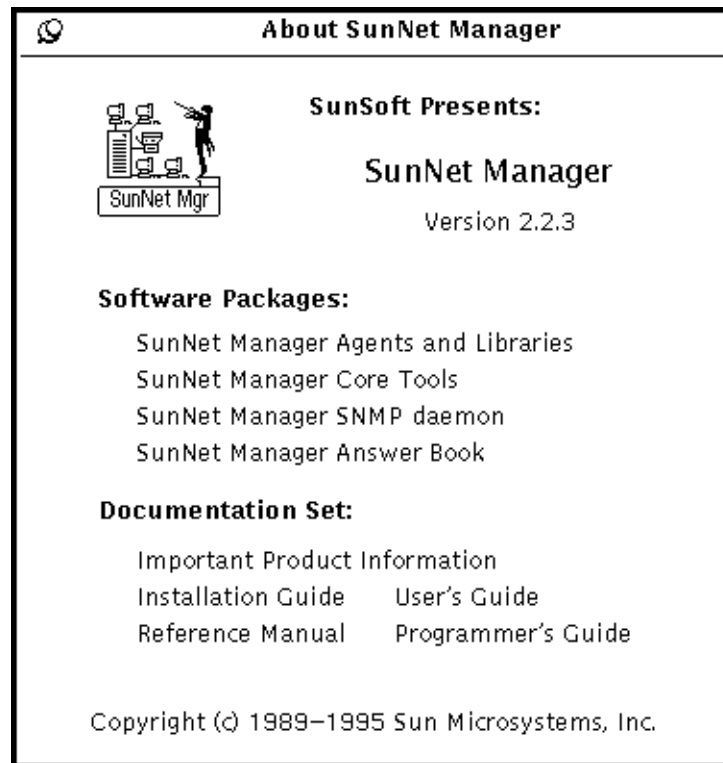


Figure 2-5 File—About Window

2.1.2 View Button

The View button menu allows you to:

- Display a list of the alarm reports received.
- Display a log of the data reports received.
- Display a log of the event/trap reports received.
- Display a log of the error reports received.
- Display a list of machines that have had an event occur on them.
- Specify a background image to be displayed for the current view.

- Remove a background image from the current view.
- Find a particular element.
- Look at the contents of the clipboard.

When you press MENU on the View button, the menu shown in Figure 2-6, is displayed.

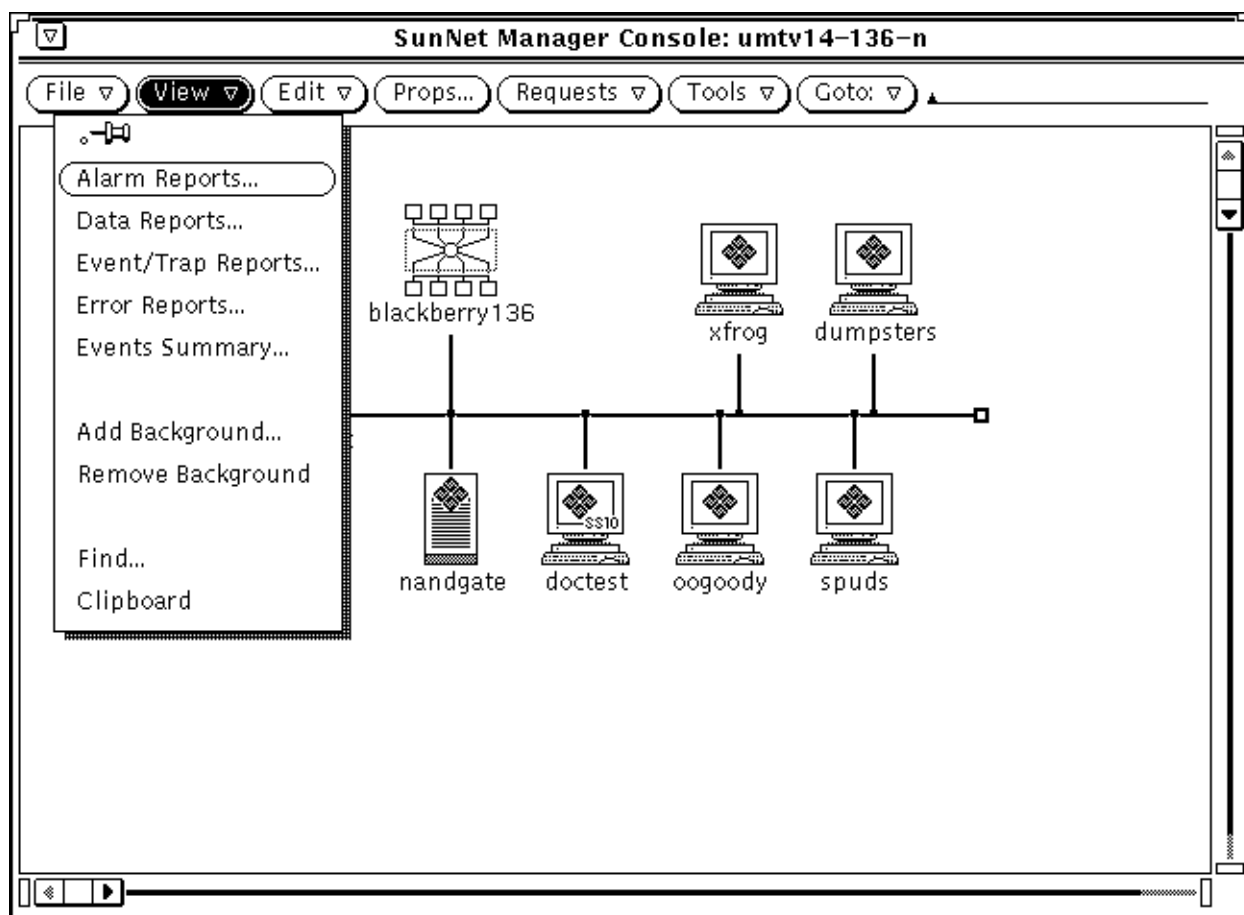


Figure 2-6 View Button Menu

Following is a description of each item in the View menu.

- Alarm Reports facilitates alarm/fault management. It displays the number and priority (high/medium/low) of events, traps, and errors to have occurred. The reports can pertain to a specific element or to an entire view. Refer to Section 4.3, “Viewing Alarm Reports,” on page 4-9 for more information about viewing alarm reports.
- Data Reports displays data reports returned from agents. The information in the reports consists of timestamp, element name, request name, and attribute name/value pairs. The reports can pertain to a specific element or to an entire view. Refer to Section 4.1, “Data Reports,” on page 4-1 for more information about viewing data reports.
- Event/Trap Reports displays event reports as well as trap reports. The information in the reports consists of timestamp, element name, request name, and attribute name/value pairs. The reports can pertain to a specific element or to an entire view. Refer to Section 4.2, “Event Reports,” on page 4-6 for more information about viewing event/trap reports.
- Error Reports displays messages that are sent when a request is not successfully launched and when other miscellaneous errors occur. The reports can pertain to a specific element or to an entire view.
- Events Summary displays a list of machines that have had at least one event or trap occur on them. You can use the Event/Trap Reports option to see the event(s) and trap(s) that have occurred for a specific device.
- Add Background allows you to specify the background image to be used for the current view. The background image is a Raster file.
- Remove Background allows you to remove the background image from the current view.
- Find displays a window which allows you to specify an element to find (see Figure 2-7). It then displays the view containing the glyph for the element specified. The message at the bottom of the Find window tells you the number of views that the particular glyph appears in. If the glyph appears in only one view, the Next button is dimmed. Otherwise, the Console displays the next view in which the glyph appears when Next is selected.

Note – Element names are *not* case-sensitive.

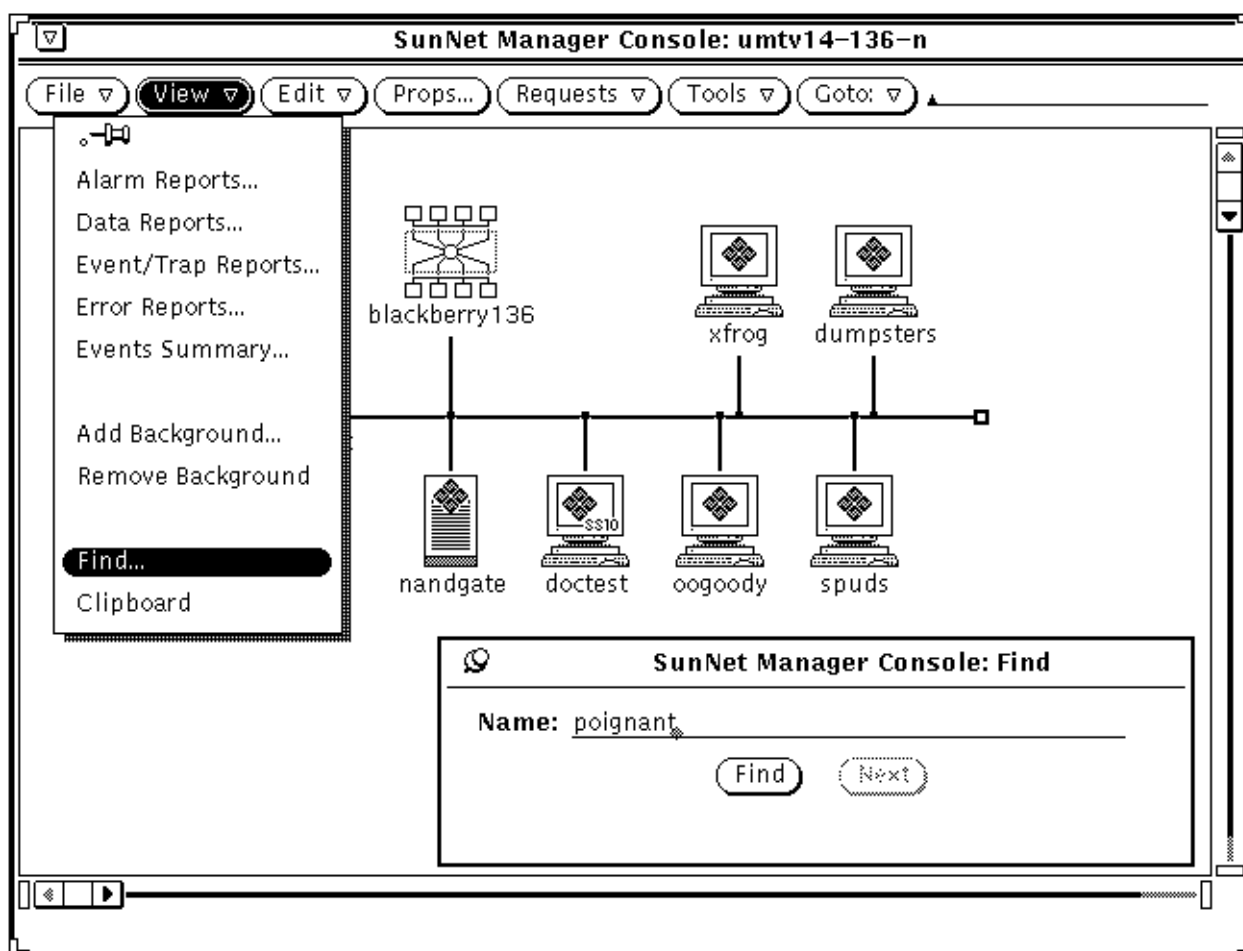


Figure 2-7 View—Find Window

- Clipboard displays the contents of the clipboard. For example, glyphs for elements and requests edited via a Cut or Copy operation are displayed. See “Edit Button” in the following subsection for a discussion of the Cut and Copy operations.

2.1.3 Edit Button

The Edit button menu provides a graphical editing capability for creating/deleting, and cutting/copying/pasting glyphs. To access the Edit menu, move the mouse pointer to either the Edit button, or to any open area (where there is no glyph) in the view, and press MENU. The SunNet Manager Console displays the Edit menu. See Figure 2-8.

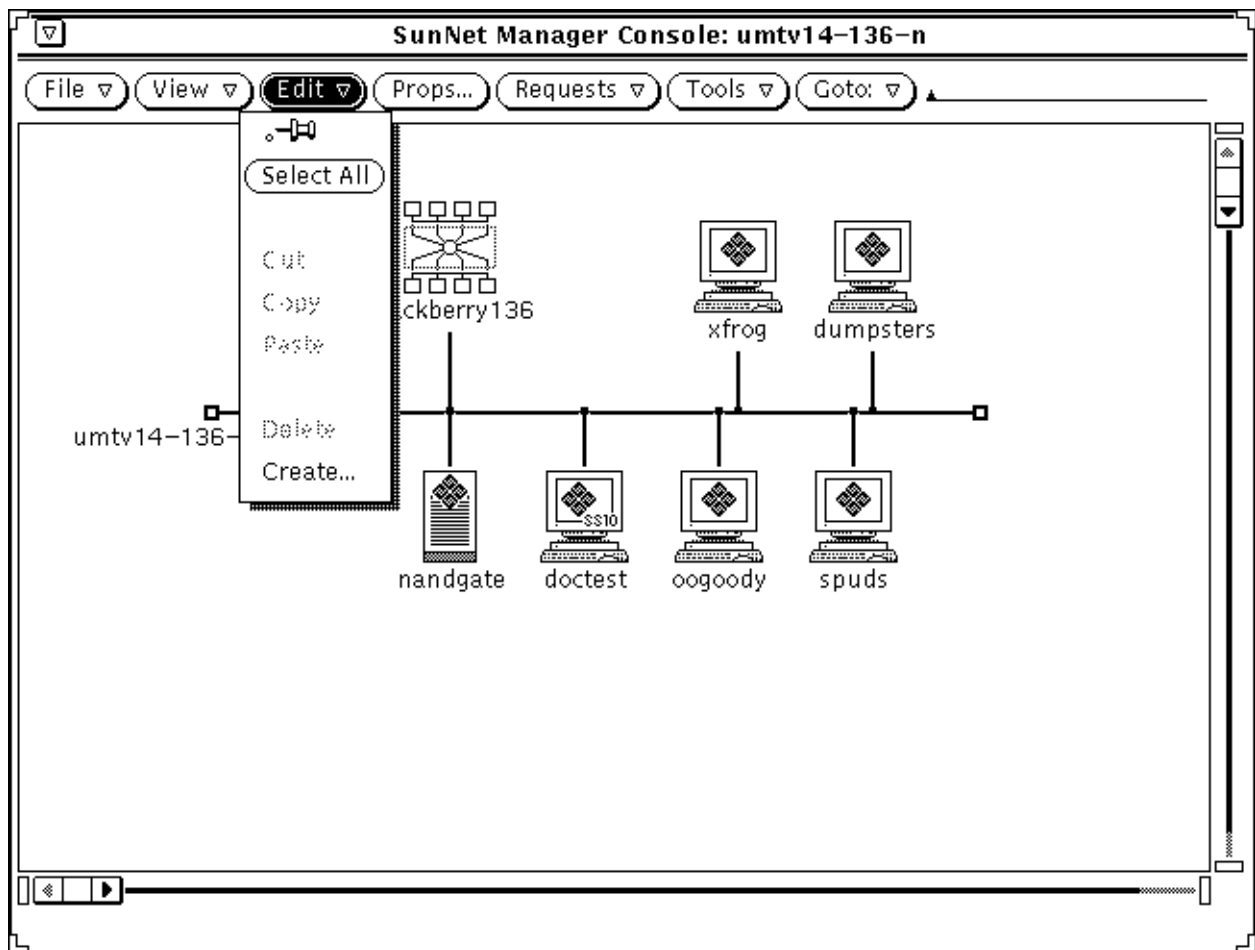


Figure 2-8 Edit Button Menu

Edit operations make use of a clipboard. The clipboard holds elements edited via the SunNet Manager Console graphical editor. The clipboard is read-only — you cannot modify clipboard elements.

Note – Copy, Cut, and Delete are highlighted (active) only when one or more elements are selected. Paste is highlighted only when there is something in the clipboard.

The following describes each item in the Edit menu.

- Select All selects all elements in the current view for copying, cutting, or deleting.
- Cut removes the selected elements from the view and places them in the clipboard. As an alternative, you can press the Cut function key. You can only cut an element if it has a non-empty subview.
- Copy copies the selected element(s)—element or request instances—to the clipboard. As an alternative, you can press the Copy function key.

Caution – Perform only one Copy operation at a time. You cannot append to the clipboard—the latest Copy overwrites the clipboard’s previous contents.

- Paste inserts the clipboard contents at the current mouse location. Alternatively, you can press the Paste (also known as Get) function key.

You cannot paste an element (a component, view, connection, or bus) in a view where it already is displayed. Duplicate element names are not allowed in a single view.

You can copy and paste a request in a different element’s subview. (See Section 2.2.8, “Show Subview,” on page 2-32 for a discussion of how to navigate to different views using the Glyph►Show Subview menu entry.) Pasting a request into the subview of a particular element inserts the request into that subview and also launches the request targeted at that element. You can use this method to launch the same requests for multiple elements.

To copy elements into a view, you can do one of the following:

- Select one or more elements, Copy them, then go to the target view and Paste.
- Select one or more elements, Copy them, then point at the glyph for the target view and press the Paste function key.

- Move the mouse pointer to the desired element and press SELECT. To copy additional elements, move the mouse pointer to them and press ADJUST. Drag the mouse pointer to the target view. For a single element, release SELECT to place it in the target view. For multiple elements, release ADJUST.

Caution – Perform only one Cut operation at a time. You cannot append to the clipboard—the latest Cut overwrites the clipboard’s previous contents.

- Delete removes the selected element from the view but does *not* place it in the clipboard. Deleting an element causes the record for that element to be removed from the runtime database. You cannot Delete an element that has an active data or event request associated with it. You also cannot Delete an element that has a non-empty subview.
- Create displays a window that allows you to create a new element (a component, view, connection, or bus). See Figure 2-9.

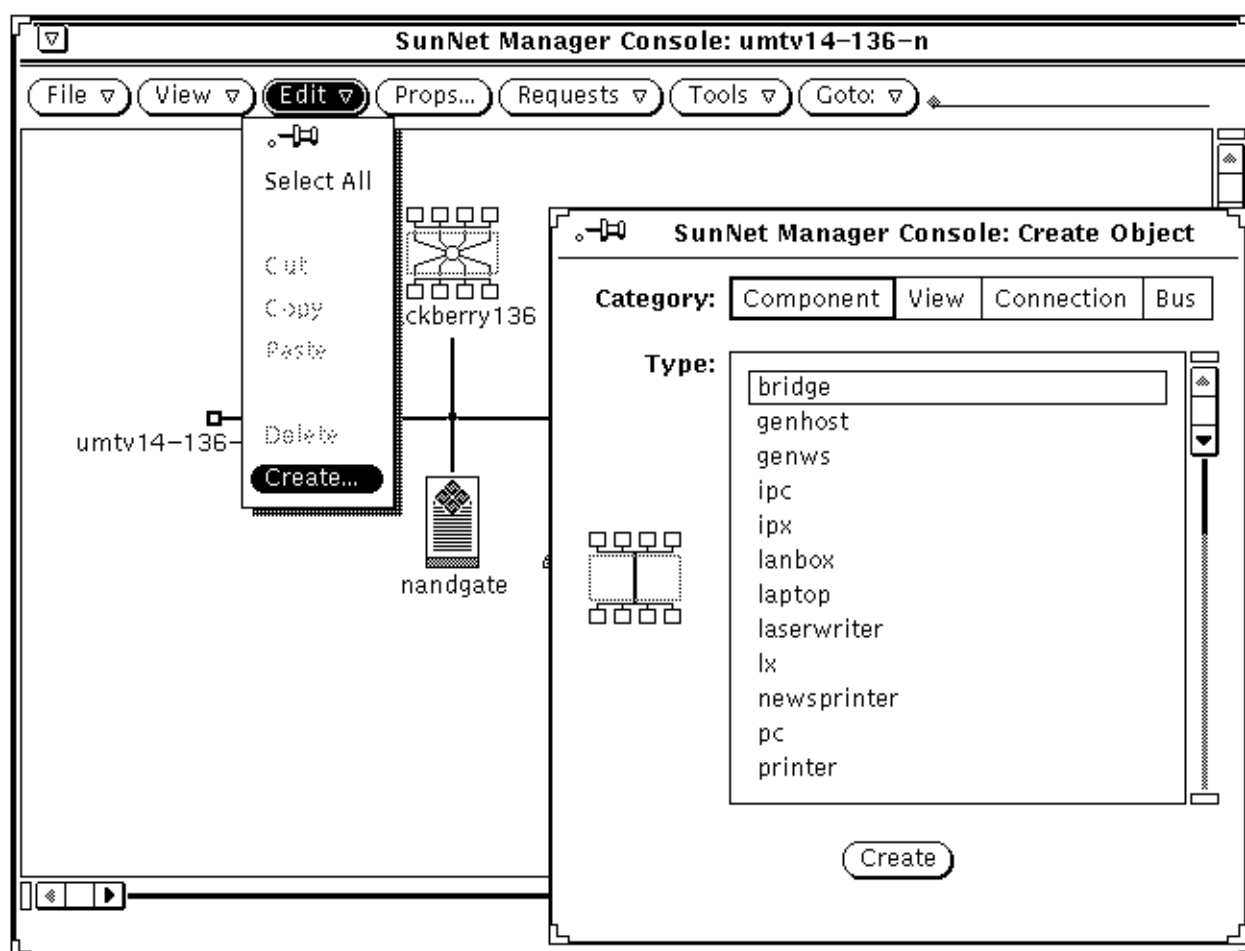


Figure 2-9 Edit—Create

You specify the Category and Type. The glyph image will be displayed for you to preview. When you are satisfied with your selections, click on the Create button. A Properties window for the selected element type appears, as shown in Figure 2-10.

SunNet Manager Console: New component.bridge

Name: _____

Net Address1: _____

Net Address2: _____

Contact: _____

Location: _____

Description: _____

SNMP RdCommunity: _____

<input type="checkbox"/> diskinfo	disk space information
<input type="checkbox"/> etherif	ethernet interface statistics
<input type="checkbox"/> etherif2	ethernet interface statistics (Solaris
<input type="checkbox"/> hostif	interface statistics
<input type="checkbox"/> hostmem	memory utilization
<input type="checkbox"/> hostmem2	memory utilization (Solaris 2.x)
<input type="checkbox"/> hostperf	Host performance data
<input type="checkbox"/> iostat	I/O statistics (SunOS 4.x)
<input type="checkbox"/> iostat2	I/O statistics (Solaris 2.x)
<input type="checkbox"/> ippath	IP path information

Red: 0

Green: 0




Blue: 0

Figure 2-10 Edit—Create—Element Properties Window

See Section 2.2.9, “Properties,” on page 2-34, for a description of the Element Properties window. The Alias button at the bottom of the Properties window is greyed out when using Edit►Create to create the new element.

The glyphs in Table 2-1 are used for more than one type of element.

Table 2-1 Glyphs Used for Multiple Element Types

Glyph	Element Category	Element Type
	Component	ss330 ss370 sun-deskside
	Component	sun-server sun470
	Connection	link rs232

If you use the Edit button to create an element, the glyph is positioned after any explicitly-positioned glyphs. Otherwise, the glyph is positioned as near as possible to the upper left-hand corner of the view window. You may also access the Edit functions by pressing MENU while pointing at any blank space in a view. Creating a new element in this manner positions the glyph at the pointer location. (This is an explicitly-positioned glyph.)

Table 2-2 shows a summary of whether an operation can (yes) or cannot (no) be done on the various SunNet Manager elements.

Table 2-2 Summary of Edit Operations

Operation	Component/ Bus/View	Connection	Simple Connect	Request
Simple Connect	yes	no	no	no
Copy	yes	yes	no	yes
Create	yes	yes	no	no
Cut	yes	yes	yes	yes
Delete	yes	yes	yes	yes
Double Click	yes	yes	no	no
Drag	yes	yes	no	yes

Table 2-2 Summary of Edit Operations

Drop into	yes	yes	no	no
Paste	yes	yes	no	yes
Properties	yes	yes	no	yes

Note that “simple” connects have not yet been described; see Section 2.2.10, “Connect,” on page 2-39, for more information.

2.1.4 Props Button

The Props button allows you to view, modify, and configure specific element or SNM Console properties. If you want to bring up the Properties window for a specific element, click SELECT on the appropriate element glyph, then click SELECT on the Props button to bring up the Element Properties window as shown in Figure 2-11.

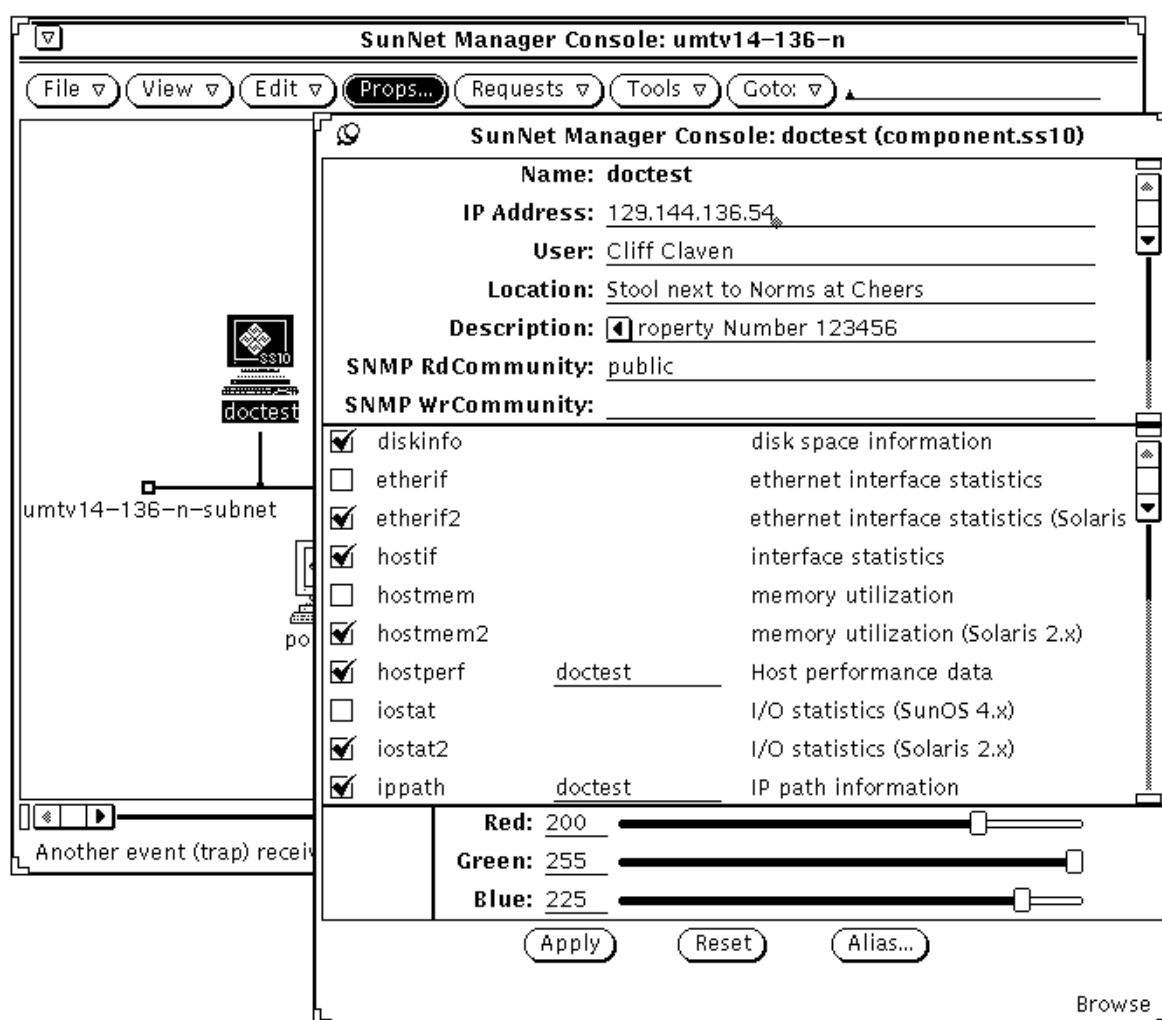


Figure 2-11 Element Properties Window

See Section 2.2.9, "Properties," on page 2-34, for descriptions of the fields in the element Properties window.

To bring up the SNM Console Properties window as shown in Figure 2-12, click SELECT on the Props button without having any elements selected. Refer to Chapter 9, “Console Properties,” for a description of the functions of the SNM Console Properties window.

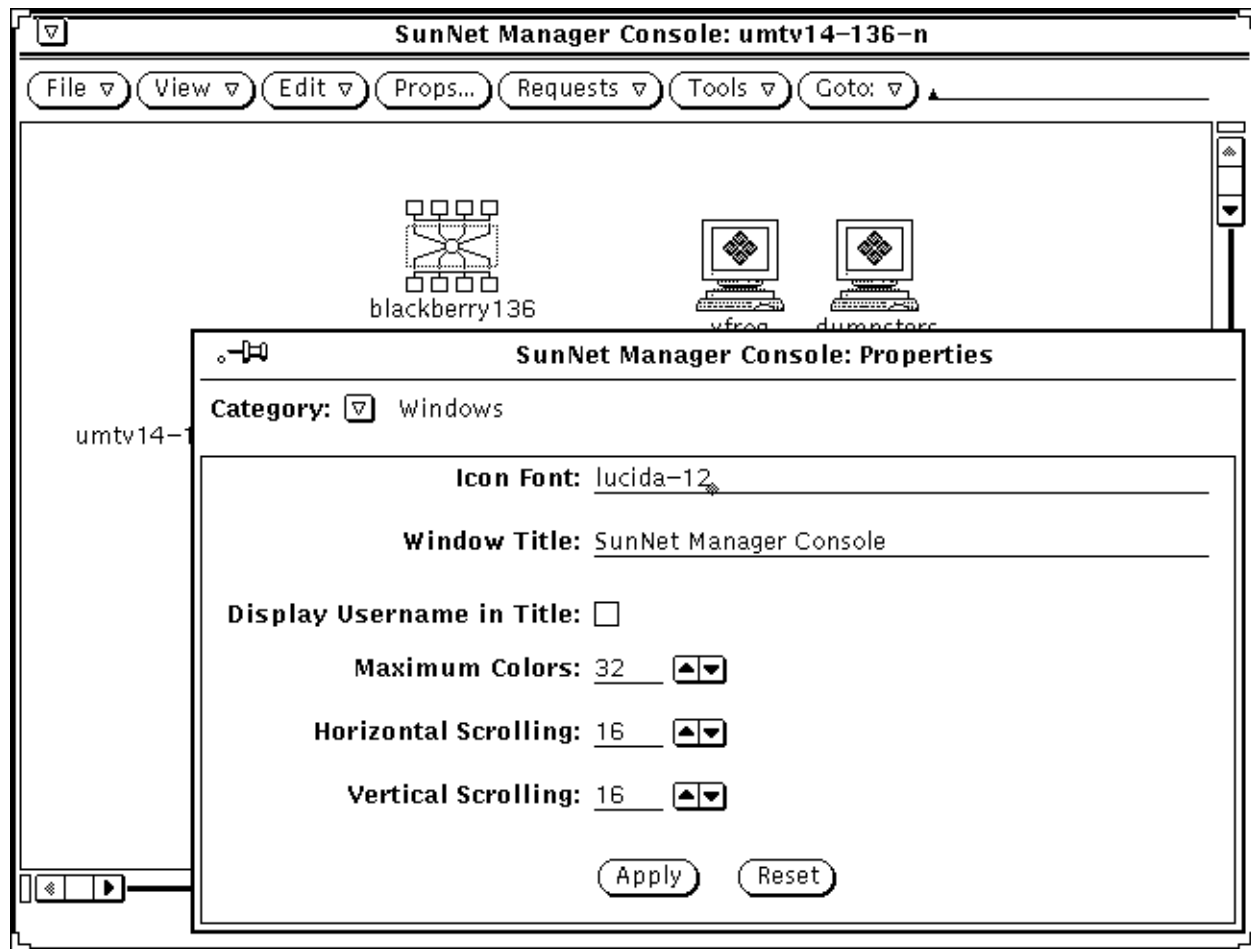


Figure 2-12 Console Properties Window

2.1.5 Requests Button

The Requests button menu allows you to send a Quick Dump request, send data and event requests, build predefined data and event requests, or get a summary of requests that are being sent. When you press MENU on the Requests button and no elements have been selected, the Requests menu shown in Figure 2-13, is displayed with the Requests Summary option as the default option and the Quick Dump and Send Request options dimmed.

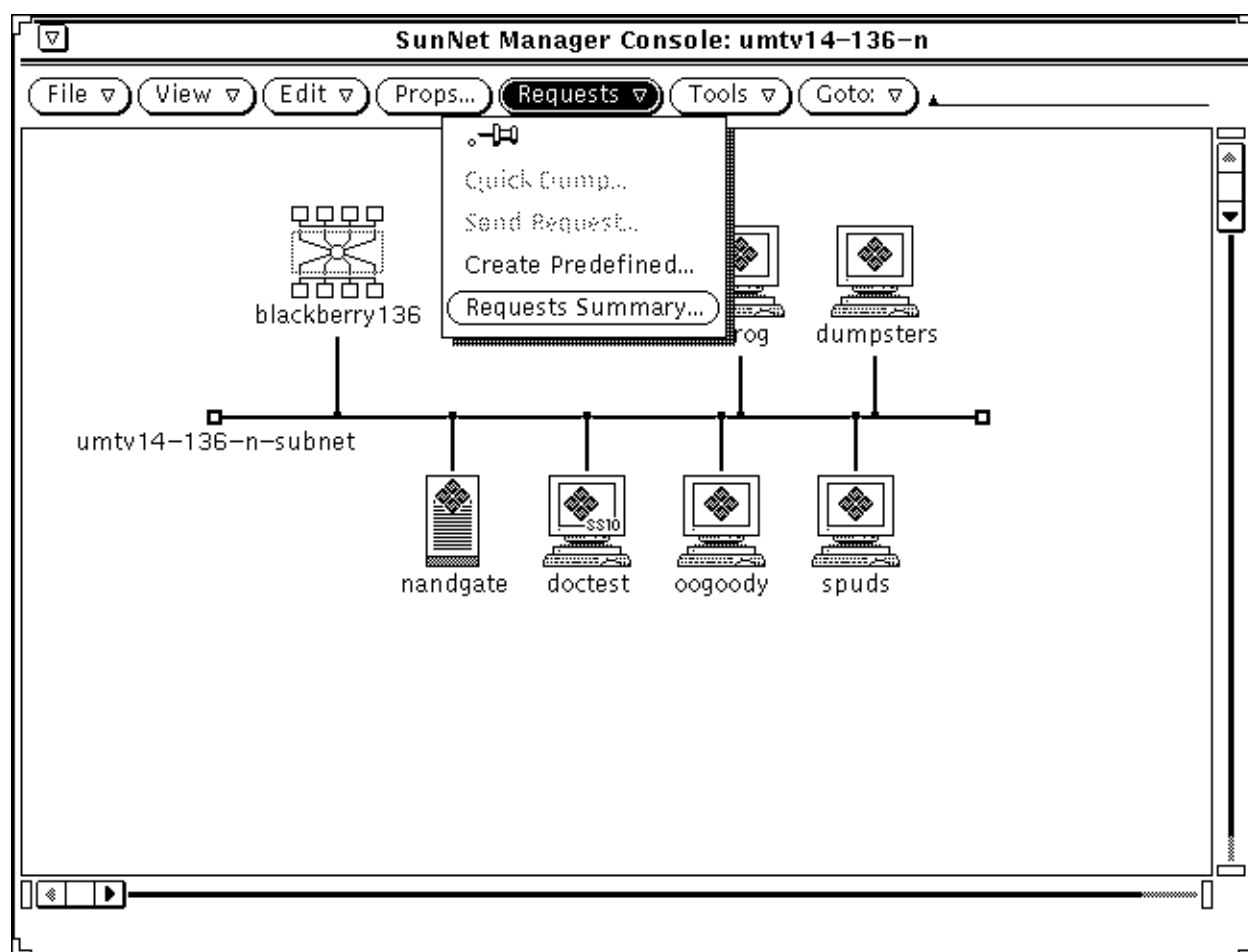


Figure 2-13 Requests Menu

2.1.5.1 *Quick Dump Option*

Quick Dump launches a request for a one-time data report for all the attributes in an agent group. Refer to Section 3.1, “Using Quick Dump to Send a Data Report,” on page 3-2 for information about using the Quick Dump option.

2.1.5.2 *Send Request Option*

The Send Request option is used to send a data or event request from the selected element. The Send Request option will be dimmed if an element was not selected prior to selecting the Requests button or if the selected element has no agents specified. Refer to Section 3.4, “Sending Data and Event Requests,” on page 3-23 for information about using the Send Request option.

2.1.5.3 *Create Predefined Option*

The Create Predefined option is used to build a predefined data or event request. This is a global operation, not element based. Therefore, this menu item will be dimmed if an element has been selected prior to clicking the Requests button. Refer to Section 3.3, “Creating, Modifying, or Deleting Predefined Requests,” on page 3-7 for information about using the Create Predefined option.

Note – The predefined data and event record names are case insensitive, as is the case with most of the SNM Console related name fields. Since the predefined data and event records reside in the runtime database with all of the elements, it is *highly* recommended that the names assigned to newly created predefined requests consist of at least two words, as is the case with the predefined data and event requests provided with this product. Otherwise, a conflict may arise between the names of hosts and the names of the predefined data and event request records, as they all must be unique.

2.1.5.4 *Requests Summary Option*

To get a summary of the data and event requests that are to be sent, click SELECT on the Requests Summary button. The window shown in Figure 2-27 on page 2-45 will be displayed. For a description of the fields in the Requests Summary Window, refer to Section 2.3.1, “Request Viewer,” on page 2-44.

2.1.6 Tools Button

The Tools button menu allows you to invoke SunNet Manager utilities from the Console. When you click MENU on the Tools button, the menu shown in Figure 2-14, is displayed.

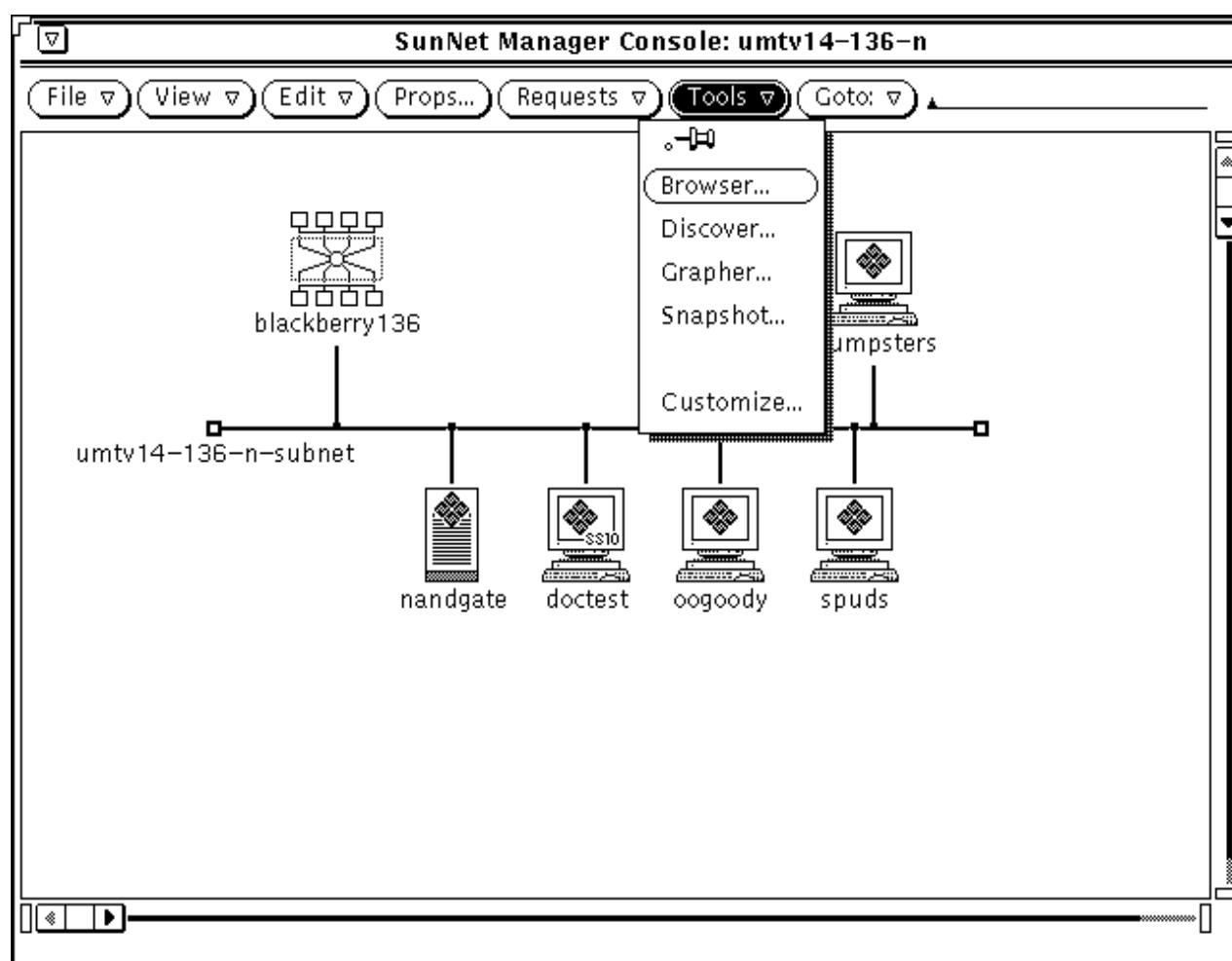


Figure 2-14 Tools Menu

The following options are available via the Tools menu:

- Browser allows you to examine and organize log files. Chapter 6, “Results Browser,” describes the Browser Tool functions.
- Discover automatically creates a graphical representation of your network. See Chapter 5, “Discover Tool,” for a description of the Discover Tool.
- Grapher allows you to visualize data reports and log file information. Chapter 7, “Results Grapher,” describes the Grapher Tool functions.
- Snapshot enables you to capture and print Console screens. For more information about using the OpenWindows Snapshot utility, refer to the *SunOS 4.x DeskSet Environment Reference Guide*.
- Customize allows you to create a customized Tools menu. The customized Tool menu remains in the runtime database, and as such, will carry over to any subsequent invocations of SNM as long as the `-i` parameter is not specified with the `snm` command.

2.1.7 Goto Button

The Goto button menu allows you to navigate through views defined in your management database. The Home option displays the home or top-level view. The other options in the Goto menu are the most recently-displayed views. Up to 16 view names in addition to Home can appear in the menu. An example menu is shown in Figure 2-15.

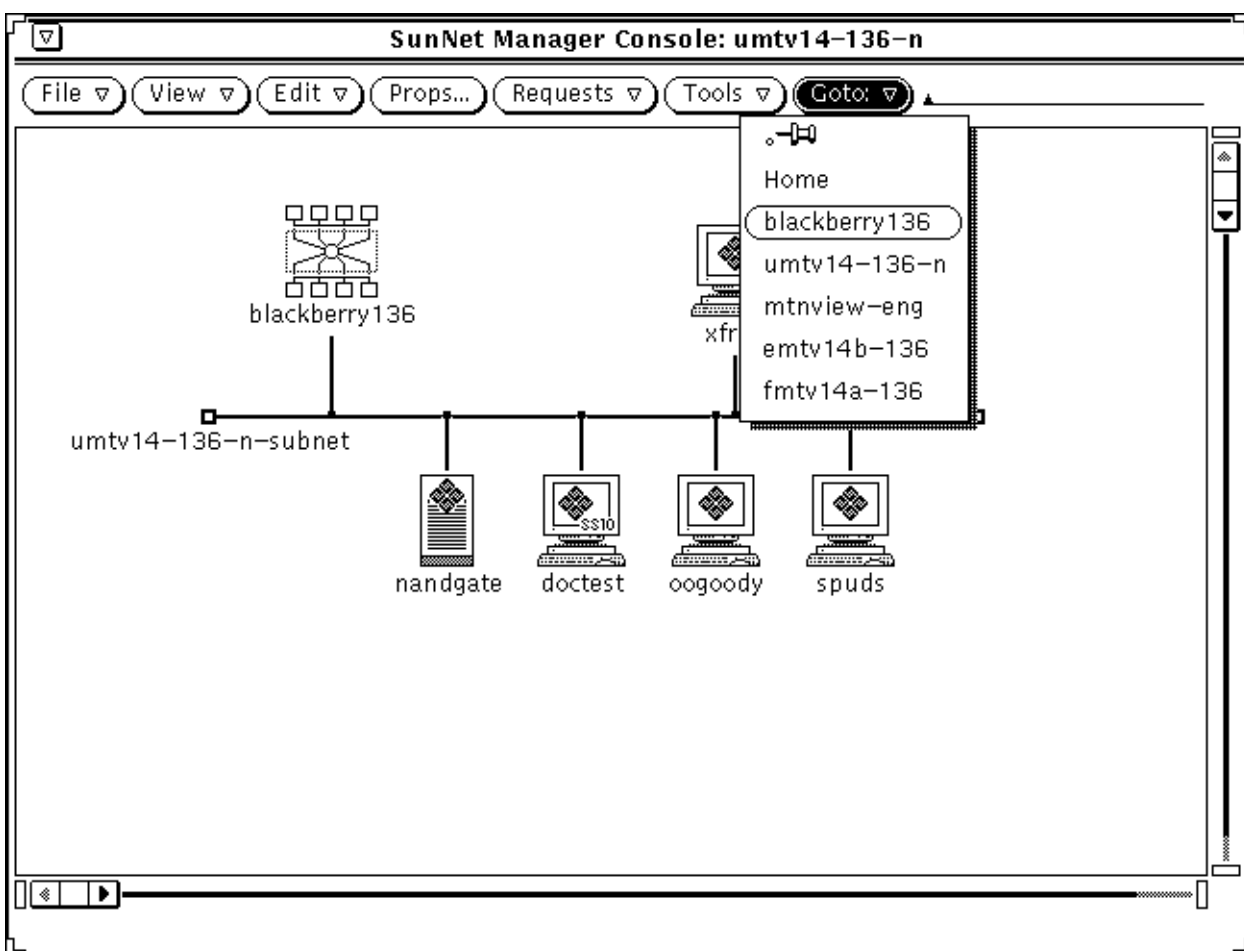


Figure 2-15 Sample Goto Menu

Clicking SELECT on the Goto button causes the last-visited view to be displayed. You can optionally change this operation so that clicking SELECT on the Goto button causes the Home view to be displayed. This is specified in the Miscellaneous category of the Console Properties window. (See Chapter 9, "Console Properties," for more information.) If you know the name of the view you want to see, you can enter the view name on the line next to the Goto button and press Return.

2.2 *Element Glyph Menu*

The element Glyph menu allows you to:

- Launch a Quick Dump request for an element.
- Launch standard and predefined data and event requests.
- Start a user-defined command on an element.
- Analyze event/trap and error reports.
- Invoke SunNet Manager tools.
- Change the glyph state of an element.
- Display the subview of an element.
- Display and modify the properties of an element.
- Connect an element to another element.
- Change the element type of a component.
- Turn automatic node management off for a subview, if automatic node management is enabled in the Console Properties window. See Section 9.4, “Automatic Management,” on page 9-7 for more information. (Note that this menu option only appears if automatic node management is enabled.)

You display the Glyph menu of an element, as shown in Figure 2-16, by pressing MENU on the selected element glyph.

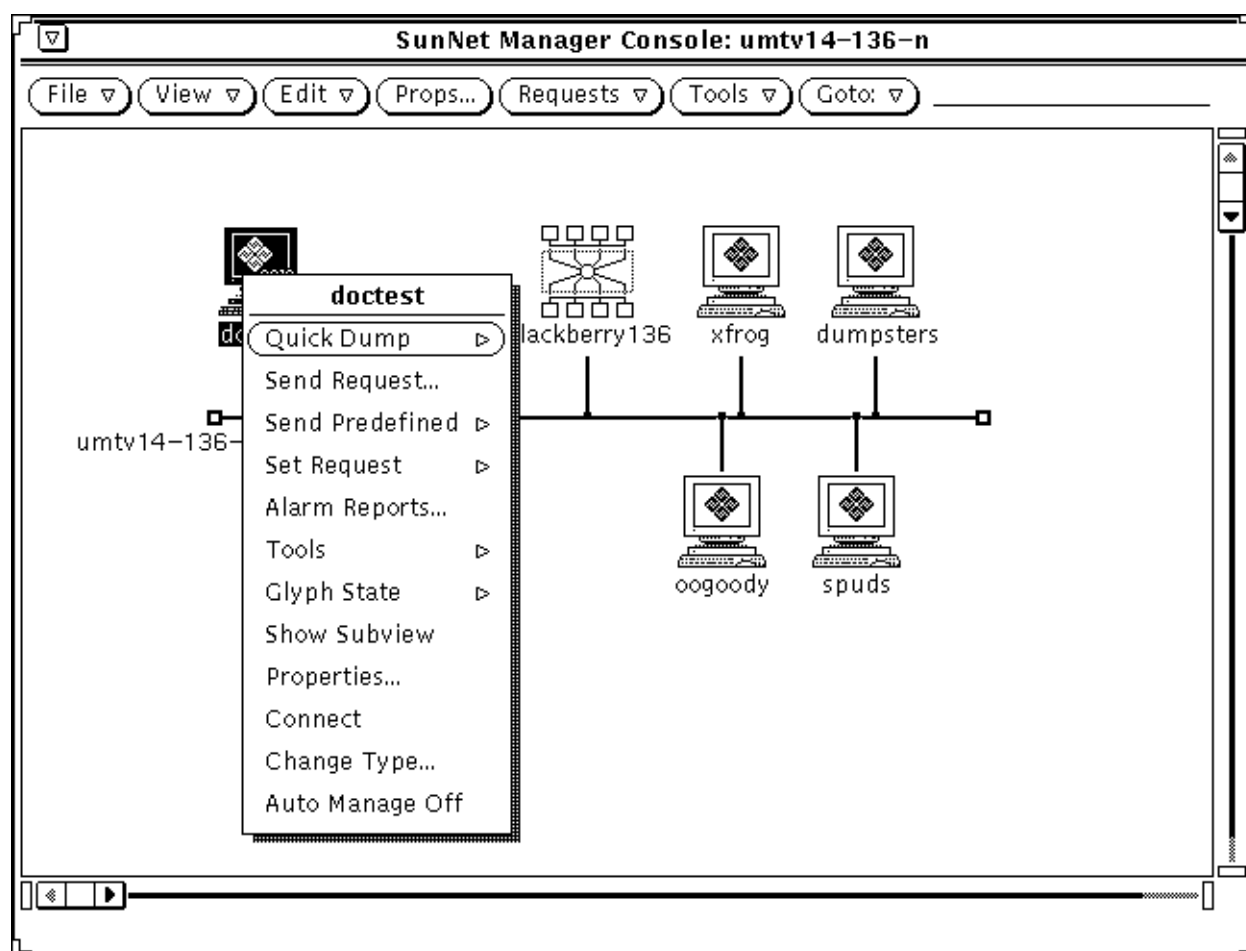


Figure 2-16 Glyph Menu

Following is a description of each item in the glyph menu.

2.2.1 Quick Dump

Quick Dump launches a request for a one-time data report of all the attributes in an agent group. Refer to Section 3.1, "Using Quick Dump to Send a Data Report," on page 3-2 for information about using the Quick Dump option.

2.2.2 Send Request

Send Request allows you to send a data or event request. Refer to Section 3.4, “Sending Data and Event Requests,” on page 3-23 for information about using the Send Request option.

2.2.3 Send Predefined

Send Predefined allows you to send a previously created predefined data or event request. Predefined data and event requests are built by using the SNM Console Requests ► Create Predefined option and assigning a Request Name to the request being built. Refer to Section 3.5, “Sending Predefined Data and Event Requests,” on page 3-39 for information about using the Send Predefined option.

2.2.4 Set Request

Set Request allows you to set attribute values. The Set Request selection is dimmed if there are no writable attributes for any agent on the target system. Refer to Chapter 8, “Set Tool,” for a detailed discussion.

2.2.5 Alarm Reports

Alarm Reports allows you to analyze event/trap and error reports for a specific element. Refer to the brief discussions on the viewing alarm reports under Section 2.1.2, “View Button,” on page 2-8, for a description of the view options available under the View button in the SNM Console. For a more detailed description of the Alarm Reports option, see the discussion under Section 4.3, “Viewing Alarm Reports,” on page 4-9. Figure 2-17 shows the format of the Alarm Reports window.

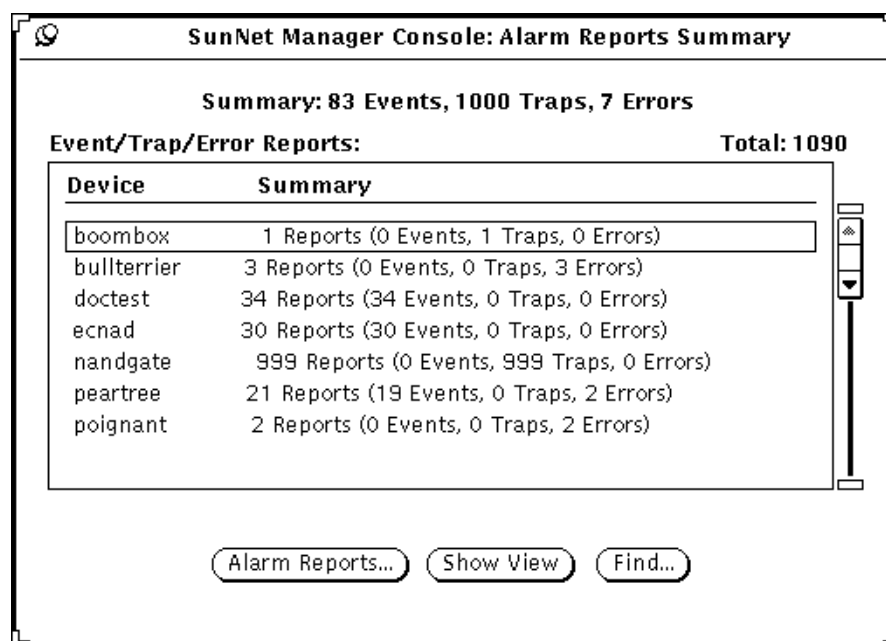


Figure 2-17 Alarm Reports Window

2.2.6 Tools

The Tools menu allows you to execute user-designated commands. See Figure 2-18.

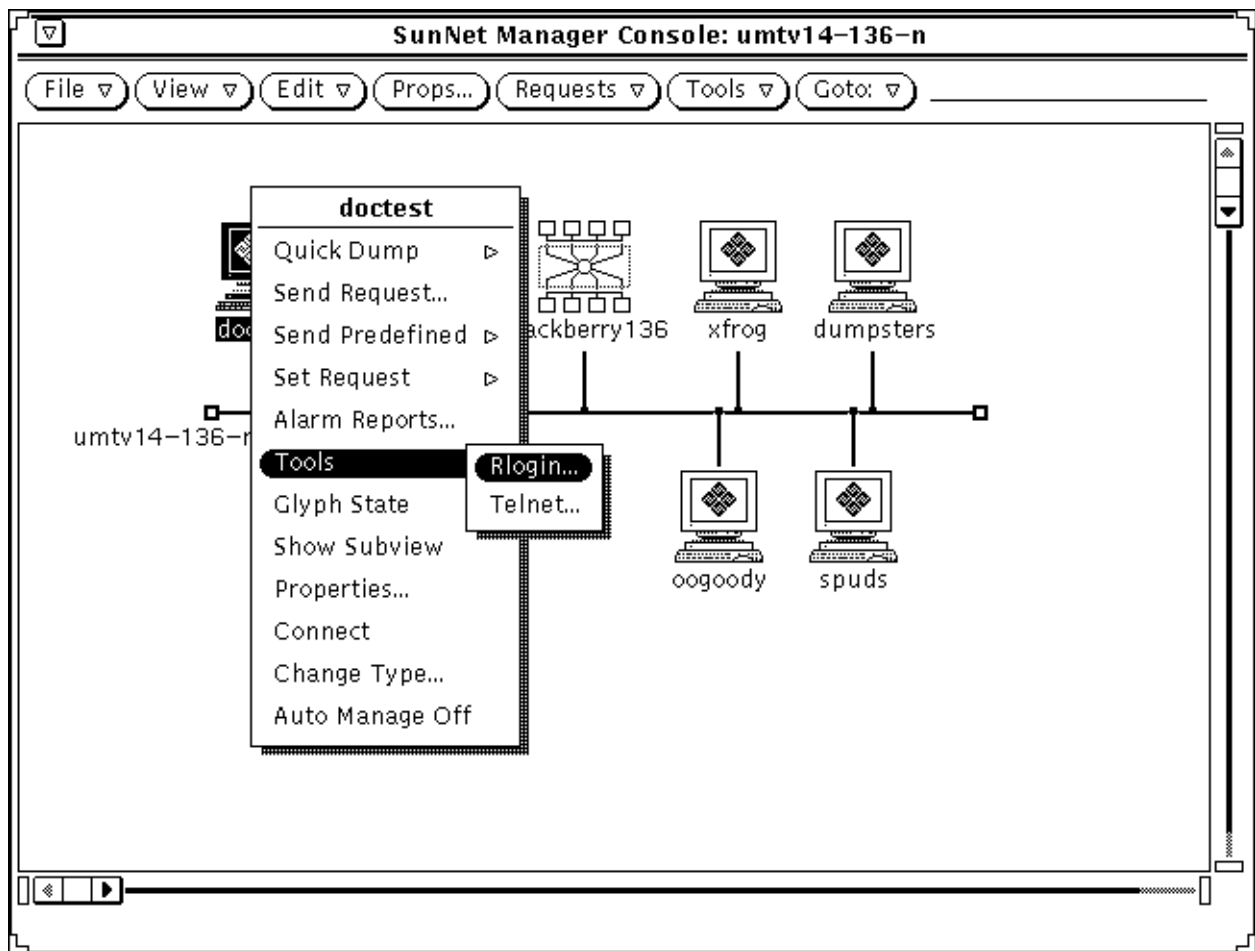


Figure 2-18 Glyph Menu—Tools

The commands in this menu are defined in the `elements.schema` file. (Refer to Chapter 10, "Management Database," for more information.) The following tools are defined by default:

- Rlogin—allows you to log in to the selected machine. This command is defined for workstations only.
- Telnet—allows you to communicate with the selected machine, using the TELNET protocol. This command is defined for all devices. However, it may not be valid for devices such as bridges and LAN boxes.

2.2.7 Glyph State

Glyph State allows you to turn blinking or dimming on/off for a glyph or turn a glyph that has changed color back to its normal color (see Figure 2-19). For example, you can turn blinking off or turn a glyph back to its normal color once you have recognized that an event has been reported for a particular element. By default, glyph state changes are propagated through the Console's view hierarchy. Refer to Section 9.5, "Events and Traps," on page 9-12 for more information about glyph state propagation. Refer to Chapter 9 of the *SunNet Manager 2.2.3 User's Guide* for information on how to have your link(s) blink and turn a different color when an event has occurred.

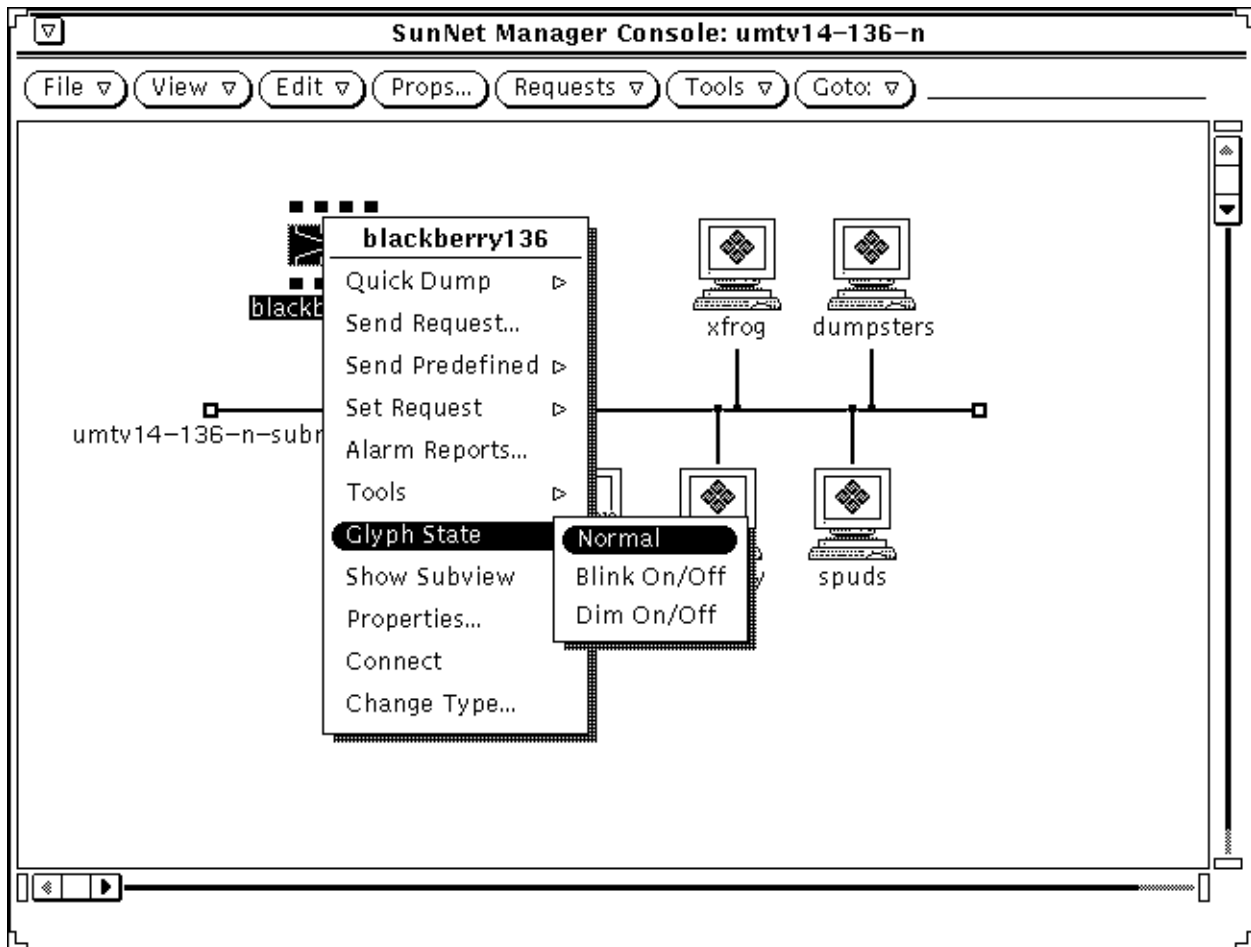


Figure 2-19 Glyph Menu—Glyph State

2.2.8 Show Subview

Show Subview changes views to the next lower level view in the containment hierarchy. The particular subview depends on the context of the current view. The Show Subview option for a subnet shows the element instances that are members of that view. Show Subview for a particular element instance shows all requests targeted at that element.

You can also display an element's view by double-clicking SELECT on the element's glyph. Figure 2-20 shows a sample Show Subview window.

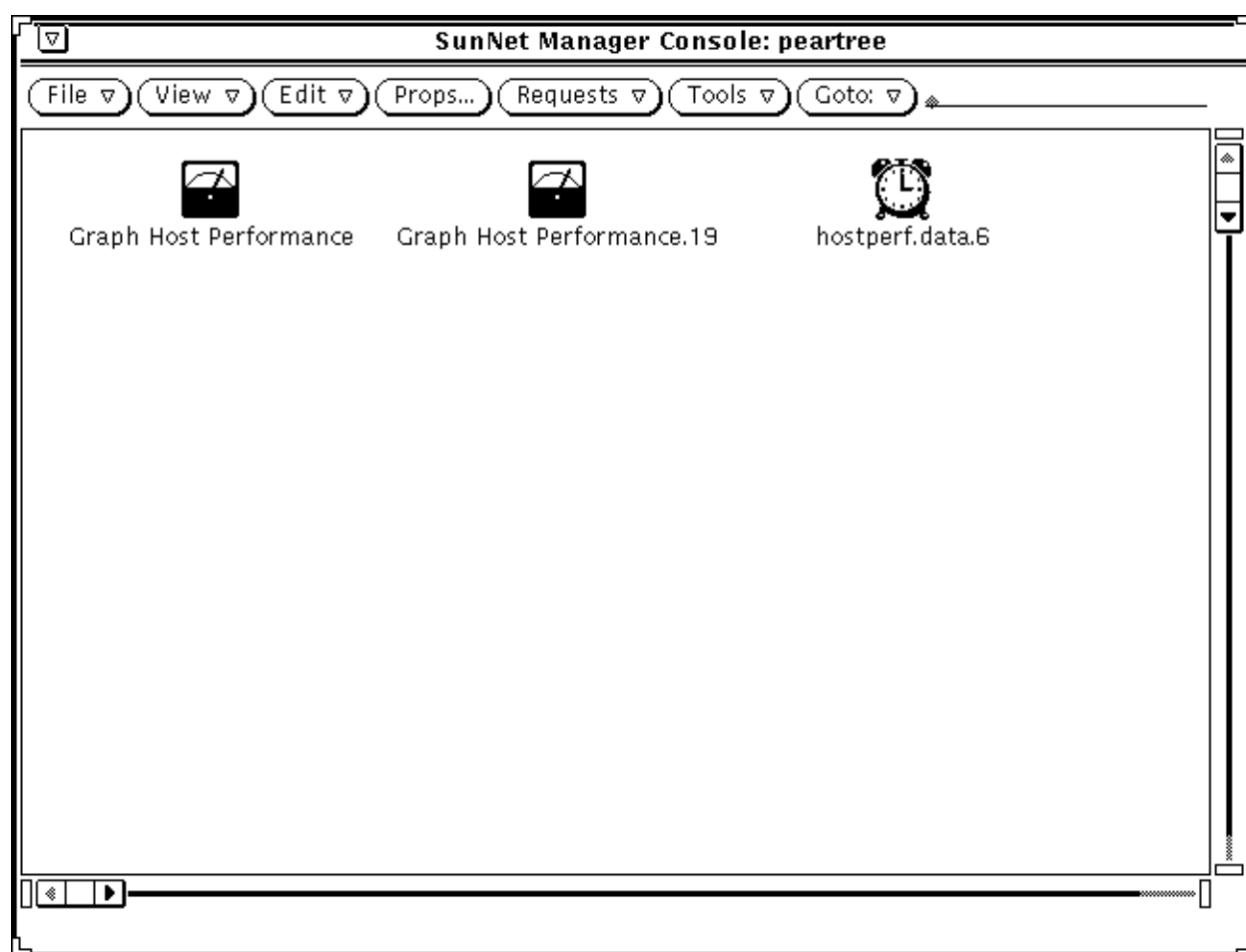


Figure 2-20 Sample Show Subview Window

2.2.9 Properties

The Properties item of the Glyph Menu displays the properties of a particular element glyph. Figure 2-21 shows the properties of the element instance boombox, which is running under the Solaris 1.x operating environment.

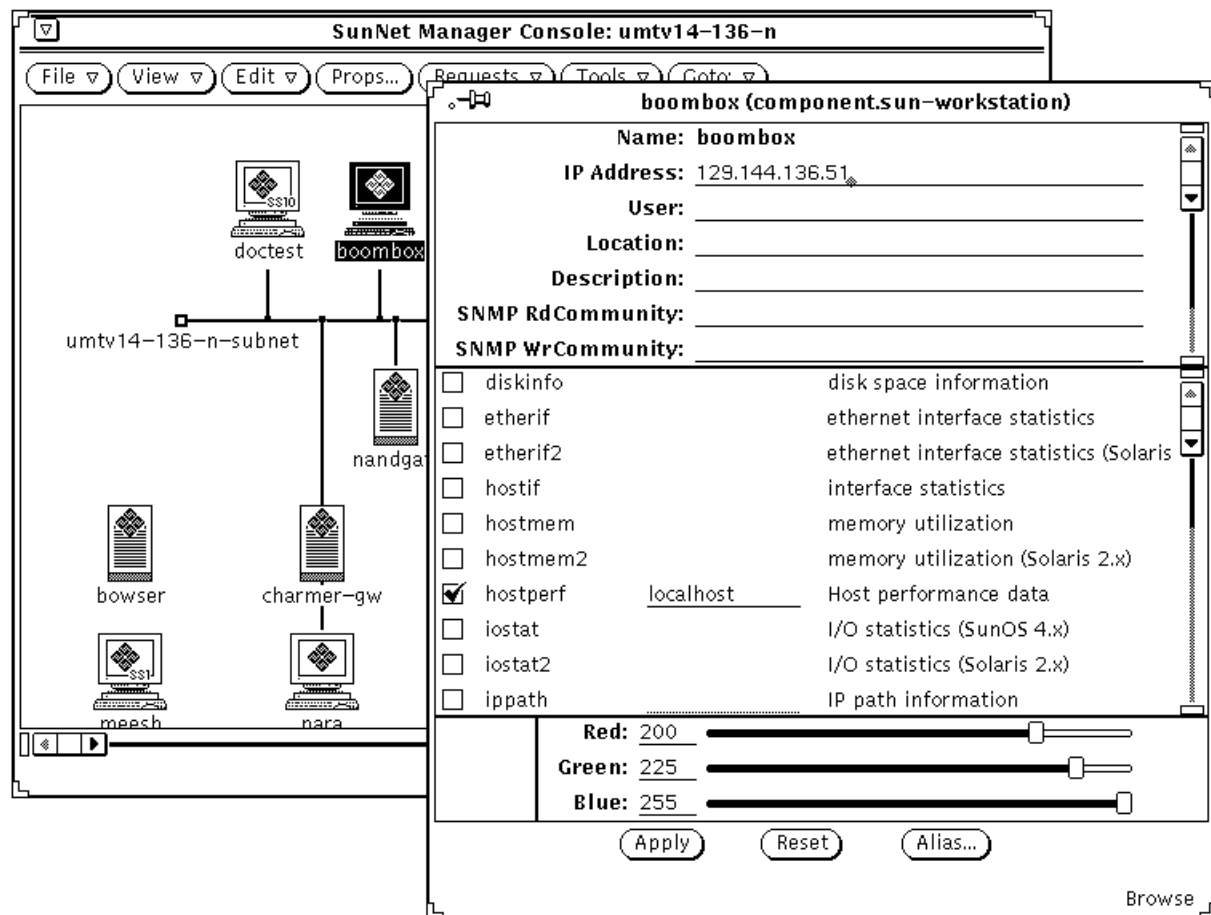


Figure 2-21 Glyph Menu—Element Properties Window

The Properties pop-up window is divided into four sections: element data, agent/proxy list, color, and a control panel.

- The element data consists of panel text items representing the fields in the element type data record as defined in the `elements.schema` file. These fields are displayed in a scrollable panel.

The values in the fields SNMP RdCommunity (read), SNMP WrCommunity (write), SNMP Vendor Proxy, and SNMP Timeout are parameters that are used only for requests to the SNMP proxy agent. (Refer to Chapter 11, “SNMP Support,” for more information about the SNMP proxy agent.) If you do not specify values in the SNMP RdCommunity, SNMP WrCommunity, and SNMP Timeout fields, the SNMP proxy agent uses the following values:

- read community is “public”
- write community is “public”
- timeout is the value (in seconds) specified by the keyword `na.snmp.request_timeout` in the `snm.conf` file on the system where the SNMP proxy agent resides. The keyword’s supplied value is 5 (seconds).

SNMP Vendor Proxy is an optional field that specifies the name of a proxy system with which the SNMP proxy agent will communicate. If this field is specified, the SNMP request is passed through the element to the secondary proxy. This field should only be specified when a vendor has supplied an SNMP proxy agent to manage a particular device or set of devices. The vendor’s SNMP proxy agent communicates with the SunNet Manager SNMP proxy agent via SNMP, but communicates with the element using either SNMP or a different protocol.

Glyph State is a field that appears for an element of types ‘view’ or ‘bus’. This field allows you to specify whether the glyph states of elements in the subview are propagated to the element.

Default Proxy is a field that appears in subnets that correspond to the element category ‘bus’ and element type ‘ethernet’. This field specifies the default proxy system for any elements that are discovered in the subnet. When the Discover Tool is run for the subnet, the name specified in this field appears in the proxy system fields in the Properties window for each element in the subnet. The name you specify for Default Proxy should be the host name of a system on the subnet. This would confine polling and data gathering to the subnet and only relevant management information would pass from the proxy to the Console.

- The agent schema list—agent schema file name, proxy system (if applicable), and brief description—shows all the agent schemas the SunNet Manager Console knows about.

Note – Most of the agents supplied with SunNet Manager have only one agent schema file. The SNMP proxy agent can use Sun-supplied SNMP schema files and/or device-specific schema files created from a MIB file.

The “check” box to the left of the agent schema list provides a convenient method of indicating that this element can be managed using a particular schema. To toggle the check box state between checked and not checked, click SELECT on the check box.

Note – Merely checking an agent schema box does not make the element manageable through the particular agent. The appropriate agent software must be installed and running on the system.

In the case of a proxy agent, the SunNet Manager Console allows you to specify the name of the Sun workstation or server where the proxy agent is running. This proxy system is used as the default in Quick Dump, data, and event reporting requests. If you used the Discovery Tool to create your management database, each discovered SNM host is the default proxy system for agents that are installed on the host. For discovered systems that are not SNM hosts, `localhost`—the Console system—is the default proxy system.

The agent schema list is generated from files in the directories specified by the Schema Locations setting in the Console Properties Locations category (Refer to Section 9.7, “Locations,” on page 9-21 for more information). If you don’t want an agent schema to appear in the list, remove the corresponding file from the directory. (Before you remove an agent schema file, make sure that the schema is not defined for any elements in an existing instance file. Otherwise, you will not be able to load the instance file.)

- The color section of the Properties window provides three fields—red, green, and blue—that specify the hue of the glyph associated with a particular machine. To change a hue, press SELECT on the color slider and drag the mouse pointer to the right or the left. Or, enter in numeric values. The intensity for each of the three colors falls in the range of 0 through 255.

All zeros makes the glyph transparent; the glyph appears the same color as the background of the Console window. White is all 255s. The box to the left of the color fields displays the currently-defined color.

If you are running the SunNet Manager Console on a monochrome monitor, modifying the color palette has no effect on the glyphs displayed on your machine. However, these modifications are kept in the runtime database. If you save your runtime database to an instance file and use this instance file when running the Console on a color workstation, the color palette values would be used.

- The control panel has the following buttons: Apply, Reset, and Alias. Use the Apply button to put into effect changes made to the properties. Use the Reset button to set the data to its initial state or to the state of the most recent Apply. Use the Alias button to apply one or more alternate names for a machine that has multiple network interfaces, such as a router, for using with SunNet Manager. Figure 2-22 shows the Alias window for the element named boombox.

Note – The Alias button will be grayed out if the Edit►Create option was used to create the new element. Refer to Section 2.1.3, “Edit Button,” on page 2-12 for information regarding the Edit►Create option.

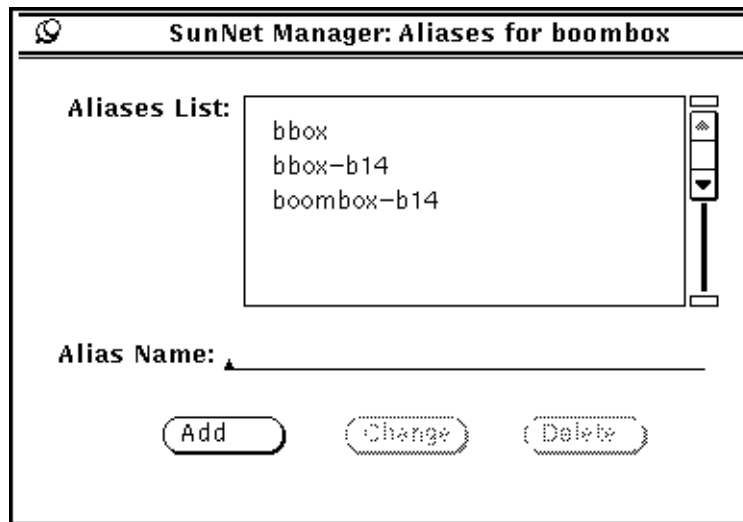


Figure 2-22 Glyph Menu—Properties—Alias Window

Enter a name at the Alias Name prompt and click SELECT on Add to add the name to the Aliases List. To change a name, click SELECT on the name in the Aliases List, make your change to the name as it subsequently appears on the Alias Name line, and click SELECT on Change. To delete a name, click SELECT on the name in the Aliases List, then click SELECT on Delete. At any time, you can click SELECT on the pushpin in the upper right corner of the aliases window to unpin and thereby dismiss the window.

The control panel at the bottom right corner of the element Properties window, indicates if you are in Browse mode (reviewing the properties of an existing element) or Create mode (creating a new element). See Figure 2-21 on page 2-34.

Note – Instead of using the Glyph►Properties pull-down menu, you can access the properties of a particular element by moving the mouse pointer over the element glyph and pressing the Props function key.

2.2.10 Connect

Connect draws connections from selected (highlighted) elements to the element glyph on which the mouse arrow is currently pointing. This type of connection is “simple”, in contrast to the connections created through the Edit menu’s Create function. Simple connections are not true elements but are useful for graphically representing connectivity. They can be selected and deleted, but cannot be copied, pasted, or managed. Simple connections inherit the color of the element to which they belong (that is, the connected-to element). Selecting an element also highlights its simple connections; however, connections cannot be moved.

To connect two glyphs, select (highlight) one of them. Move the mouse pointer over the other glyph and choose the Glyph►Connect option, as shown in Figure 2-23, for the element. As an example, using two elements called doctest and blackberry136, the connection will be drawn as shown in Figure 2-24 on page 2-41. To delete the connection created by using this feature, click SELECT on the connection, then click SELECT the Edit►Delete option.

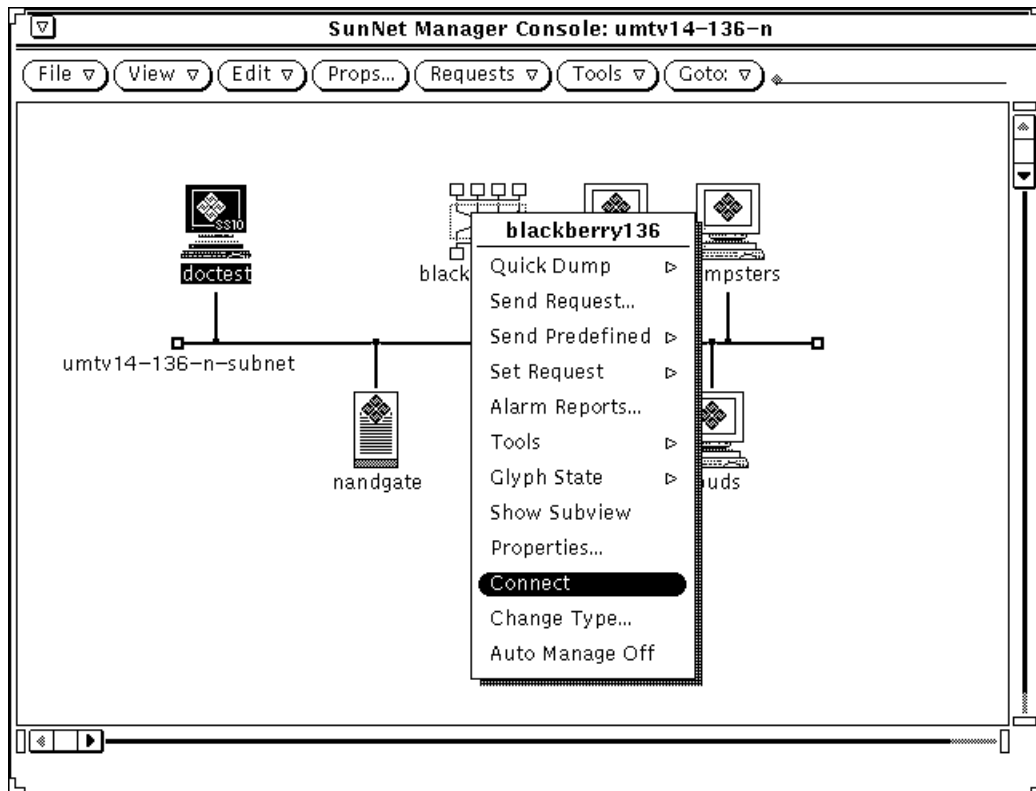


Figure 2-23 Glyph Menu—Connect Window

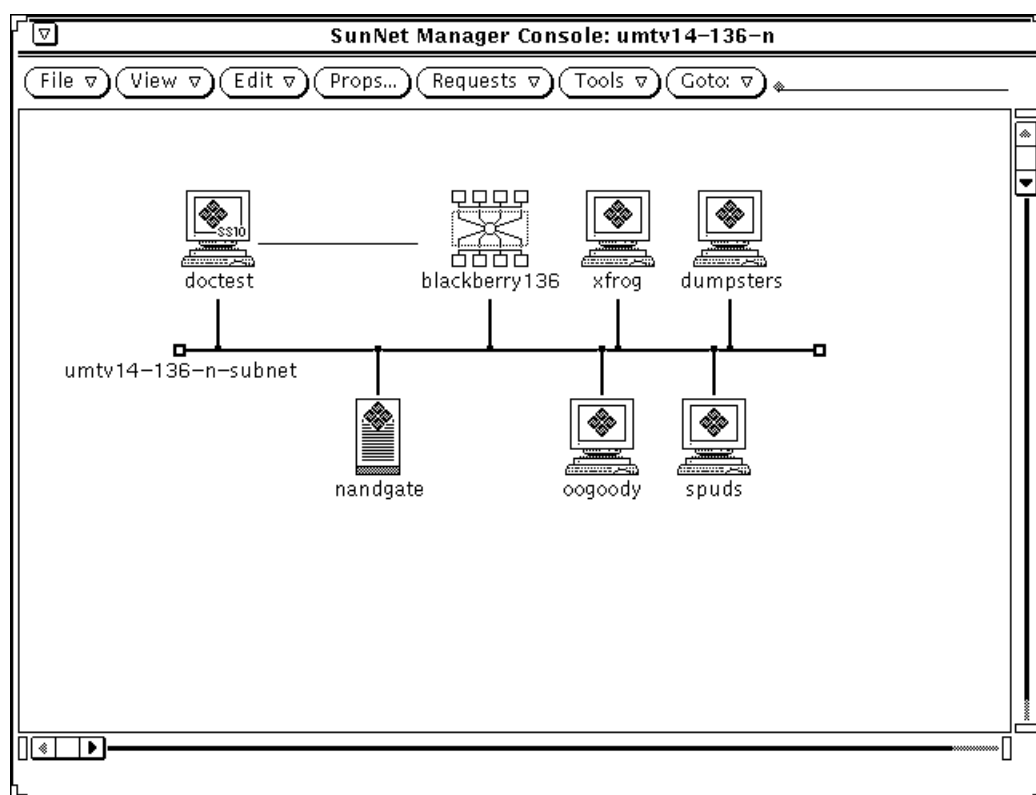


Figure 2-24 Connection Created Between two Elements

2.2.11 Change Type

Change Type allows you to change the element type of a component. Change Type displays a submenu that allows you to select a new component element type. Pull right to the desired element type and release the menu button. The glyph changes to reflect the glyph that is associated with the new element type. Figure 2-25 shows the Change Type pop-up window that allows you to change an element type.

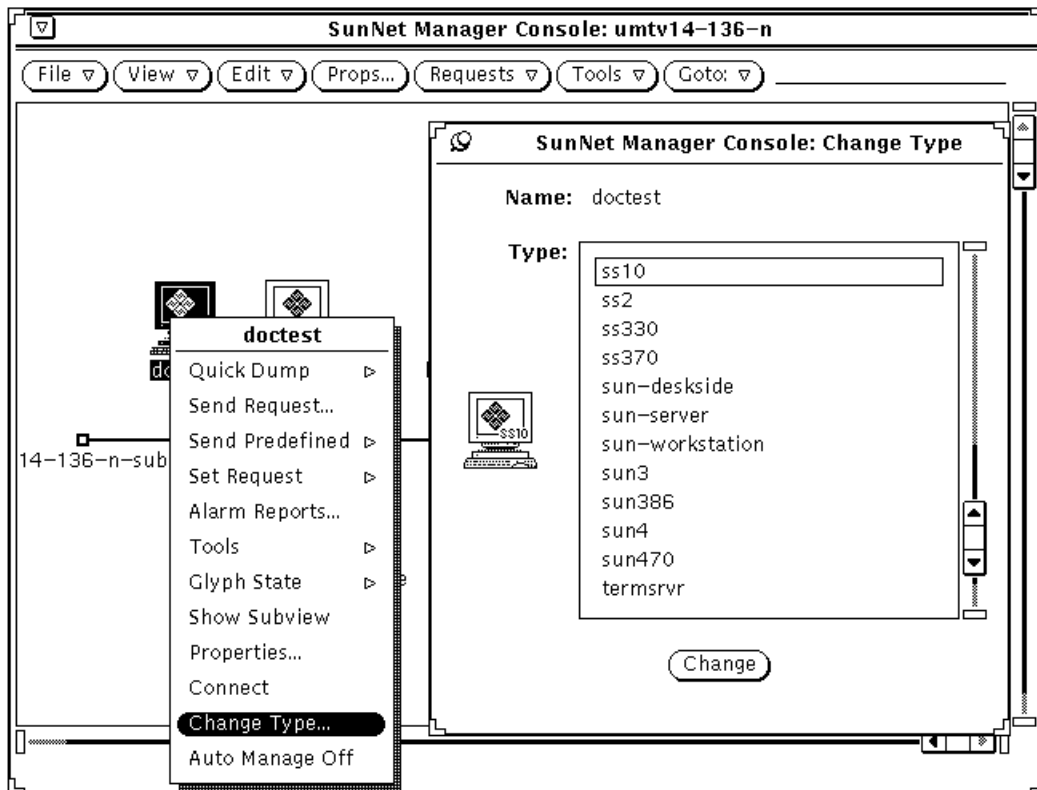


Figure 2-25 Glyph Menu—Change Type Window

When an element type is changed, in some cases not all of the element data fields for the previous element type can be copied over to the new element type. In this case, appropriate warnings are logged in the Error Reports window.

2.2.12 Auto Manage Off

Note – This menu option only appears if you enable automatic node management in the Console Properties window. Refer to Section 9.4, “Automatic Management,” on page 9-7 for more information on this feature.

If automatic node management is enabled, automatic requests are launched for all elements that support at least one of the agents associated with automatic requests. Selecting the Auto Manage Off option, shown in Figure 2-26, specifies that no automatic requests are launched for the element or view. Any active automatic requests for the element or view are killed. Once you choose the Auto Manage Off option, the menu option changes to Auto Manage On. You can use the Auto Manage On option to toggle automatic node management back on for the element or view.

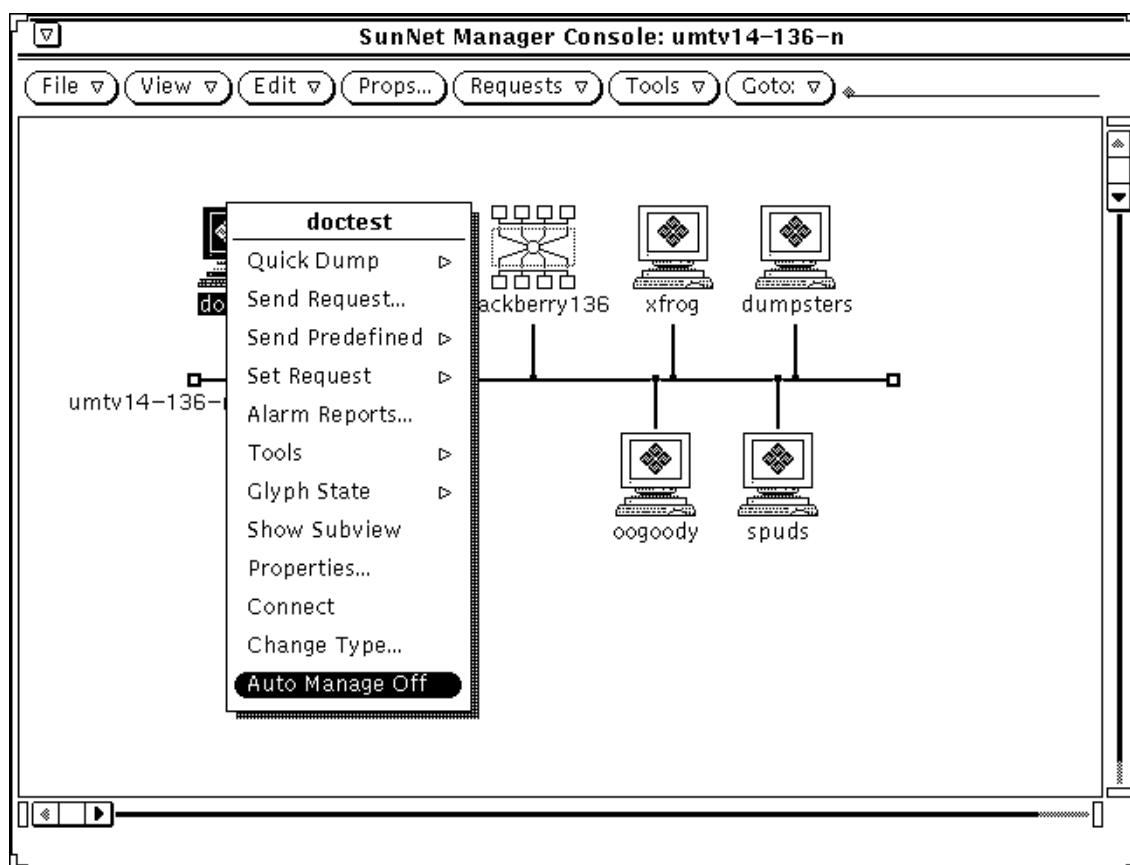


Figure 2-26 Glyph Menu—Auto Manage Off

If the element is a bus or a view, the Auto Manage Off option prevents and kills automatic requests for elements contained in the bus or view. You can toggle automatic management on for individual elements in the view by choosing the Auto Manage On option in the Glyph menu for the element.

2.3 *Viewing and Modifying Requests*

Requests that have been started or held may be examined or modified. There are several ways to do this:

- The SunNet Manager Console Requests►Requests Summary option allows you to examine the status of requests. You can also modify, stop, kill, or restart one or more requests at a time.
- Request glyphs that reside in the view of the target element allow you to modify or kill requests, one at a time. You may also select multiple requests and kill them.

2.3.1 *Request Viewer*

To display the Requests Summary window, press MENU over the SunNet Manager Console Requests►Requests Summary option and release. The Requests Summary pop-up window appears as shown in Figure 2-27.

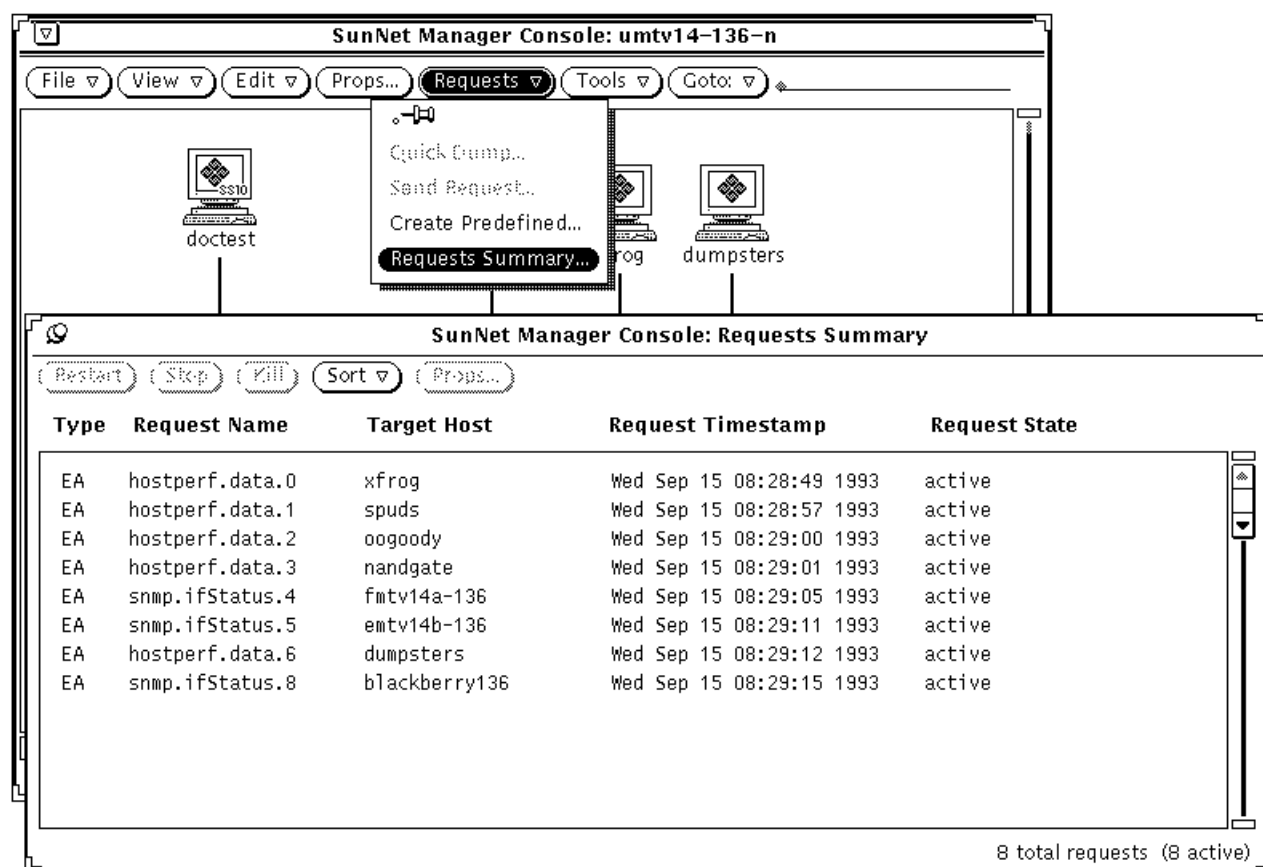


Figure 2-27 Requests Menu—Requests Summary Window

Following is a description of the information that is displayed in the Requests Summary window:

- Type defines the request as one of the following:
 - D: data request
 - E: event request
 - EA: event request launched by the Automatic Node Management feature of the Console

- Request Name lists the name you have assigned to the request or, if you have not assigned a name, a name of the format *<agent>.<group>.<number>*, for example, *ping.reach.4*.
- Target Host is the name of the target element for the request.
- Request Timestamp is the date that the request was started or restarted. Dates are only displayed for requests that have been started at least once.
- Request State is one of the following:
 - Active
 - Stopped
 - Awaiting stop
 - Stopping
 - Awaiting activation
 - Being activated

The footer of the window indicates the total number of requests (whether active or stopped) and the number of active requests.

Using the buttons at the top of the window, you can restart, stop, sort, or kill requests. The operation applies to all requests that you have selected (highlighted). To select requests, press MENU in the scrolling list area of the window to bring up the Select menu. The Select menu is shown in Figure 2-28.

If the On Completion field in the properties of a request is set to Delete Request (rather than the default Save Request), then stopping that request is the same as killing the request. A description of the fields in requests can be found in Section 3.4.1.3, “Data Request Template.”

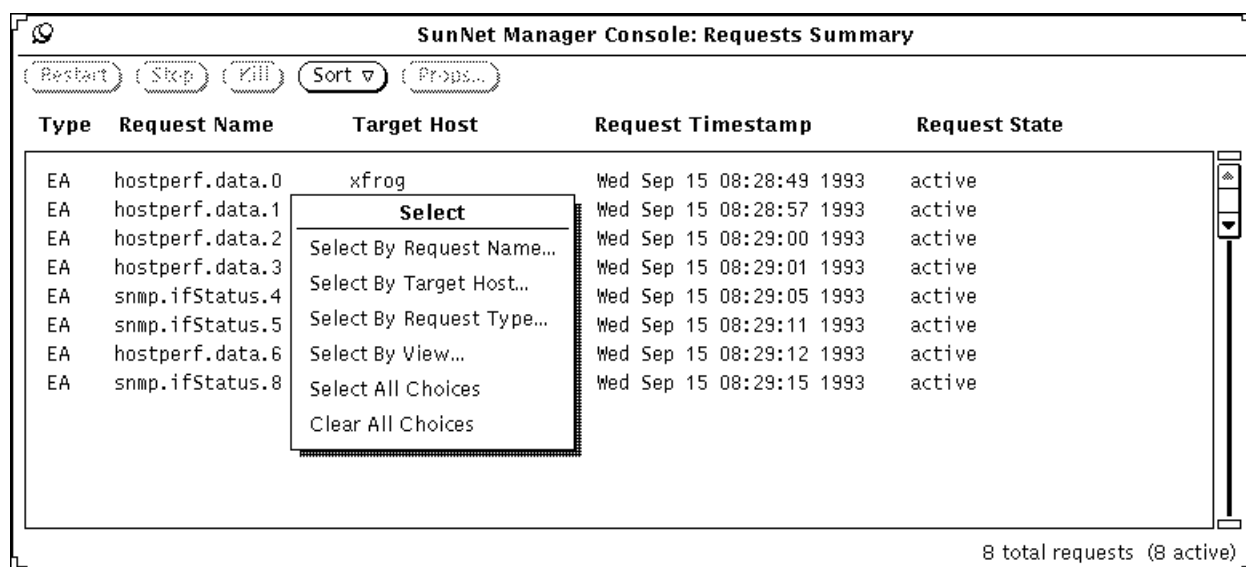


Figure 2-28 Selecting Requests

If you choose Select By Request Name, Select By Target Host, Select by Request Type, or Select By View, a pop-up window is displayed where you can specify the request name, target element, request type, or view name. Wildcard characters (* and ?) may be used when specifying the request name, target element, or request type. View name cannot be specified with wildcard characters.

Note – If an element is selected (highlighted) in a Console view, the element name appears in the target element or view name pop-up window.

The Sort button menu allows you to sort the displayed requests by request name, target host, request timestamp, request type, or request state. Sorting by timestamp displays the requests from earliest to latest start times. Sorting by request state displays requests according to the following order of request states:

1. Stopped
2. Awaiting activation
3. Being activated
4. Active
5. Awaiting stop
6. Stopping

All other sorting is done in alphabetical order.

The Props button allows you to display the Request Properties window. You can then modify any of the properties of the request. If more than one request is selected when you click SELECT on the Props button, only the Properties window for the first selected request is displayed.

2.3.2 Request Glyphs

Requests are represented by glyphs in the subview of the target element. To display the request glyph view, you can either double-click SELECT on the target element or choose the element Glyph►Show Subview menu option.

Note – Held requests or requests that are completed and saved are displayed as dimmed.

There is a pop-up menu associated with each request glyph. To display the request glyph's menu, move the mouse pointer over a the request glyph and press MENU. This is shown in Figure 2-29.

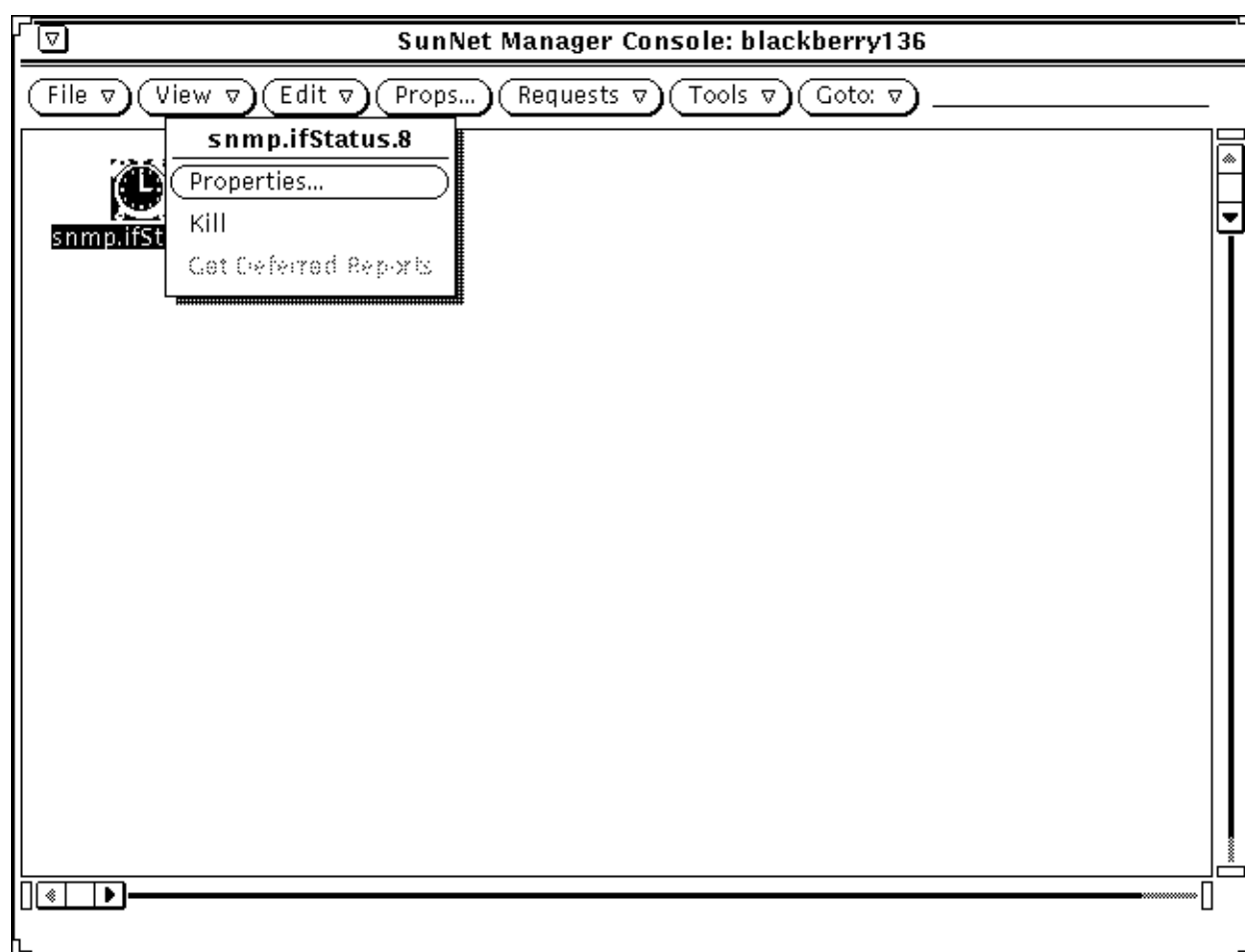


Figure 2-29 Request Glyph Menu

Following is a description of the options in the Request Glyph Menu:

- Properties shows the properties of a particular request. See Figure 2-30.

SunNet Manager Console: Event Request (doctest.hostperf.data)

Name: if System Reboot	Attributes: uptime
Proxy System: doctest	
Interval: 600	
Count: 0	
Key:	
Restart: <input checked="" type="checkbox"/>	Attribute: <input type="text" value="cpu%"/>
Send Once: <input type="checkbox"/>	Relation1: <input type="text" value="Threshold Not Set"/>
Defer Reports: <input type="checkbox"/>	Threshold1:
On Completion: <input checked="" type="checkbox"/> Save Request	Relation2: <input type="text" value="Threshold Not Set"/>
Options:	Threshold2:
	Priority: <input type="text" value="Low"/>
	Glyph Effect: <input type="text" value="Blink Glyph"/>
	Audio Effect: <input type="text" value="None"/>
	Audio File:
	Mail To:
	To Program:

Host performance data CPU utilization

Start Hold Reset Apply Reset Delete

Figure 2-30 Sample Request Properties Window

You can modify the request in several ways. For example, you can choose a different attribute, change the report frequency, stop an active request, or start a request that was placed on hold. When you modify a request, you are actually changing the specifications for that request, not launching a new one. The request manager stops the original request and restarts it with the new specifications.

Clicking the Stop button stops the request and saves it only if you have previously specified that the request be saved upon completion. Otherwise, the Stop button effectively kills the request. Use the Restart button to launch held or saved requests. If the request has already been started, Restart restarts the modified request with the new attribute specification and report characteristics.

-
- Kill allows you to kill a request that you no longer wish to run. Kill deletes a request even if you have previously set the Upon Completion request properties field to Save Request.

Note – Quitting the Console causes any active requests to be discarded unless the Restart field is on (a check mark appears in the accompanying box) or the On Completion field is set to Save Request.

- Send Deferred indicates you are now ready to receive deferred reports. The agent system now sends the deferred reports (up to 32) to the Console. Refer to the Defer Reports discussion in Section 3.3.1.1, “Reporting Characteristics,” on page 3-10 for a description of deferred reports.

Data and Event Requests



<i>Using Quick Dump to Send a Data Report</i>	<i>page 3-2</i>
<i>Predefined Requests Supplied with SNM 2.2.2</i>	<i>page 3-5</i>
<i>Creating, Modifying, or Deleting Predefined Requests</i>	<i>page 3-7</i>
<i>Sending Data and Event Requests</i>	<i>page 3-23</i>
<i>Sending Predefined Data and Event Requests</i>	<i>page 3-39</i>

This chapter describes the following aspects of data and event requests:

- Sending a one-time data report using the Quick Dump option via the Requests button or the Glyph menu.
- Provides a list and description of the predefined data and event requests supplied with SNM 2.2.3.
- Creating, modifying, and deleting predefined data and event requests.
- Sending data and event requests via the Requests button or the Glyph menu.
- Sending predefined data and event requests via the Glyph menu.

3.1 Using Quick Dump to Send a Data Report

The Quick Dump option is used to launch a request for a one-time data report of all the attributes in an agent group. There are two ways to send a Quick Dump request; via the SNM Console Requests►Quick Dump menu item or via the element Glyph►Quick Dump menu item.

3.1.1 Quick Dump Option via the Requests Button

If the Quick Dump menu item in the Requests button menu is dimmed, it means there are no readable attributes for any agent on the target system. To build a Quick Dump request, select an element then click SELECT on the Requests button and SELECT the Quick Dump menu item. The window shown in Figure 3-1, will be displayed.

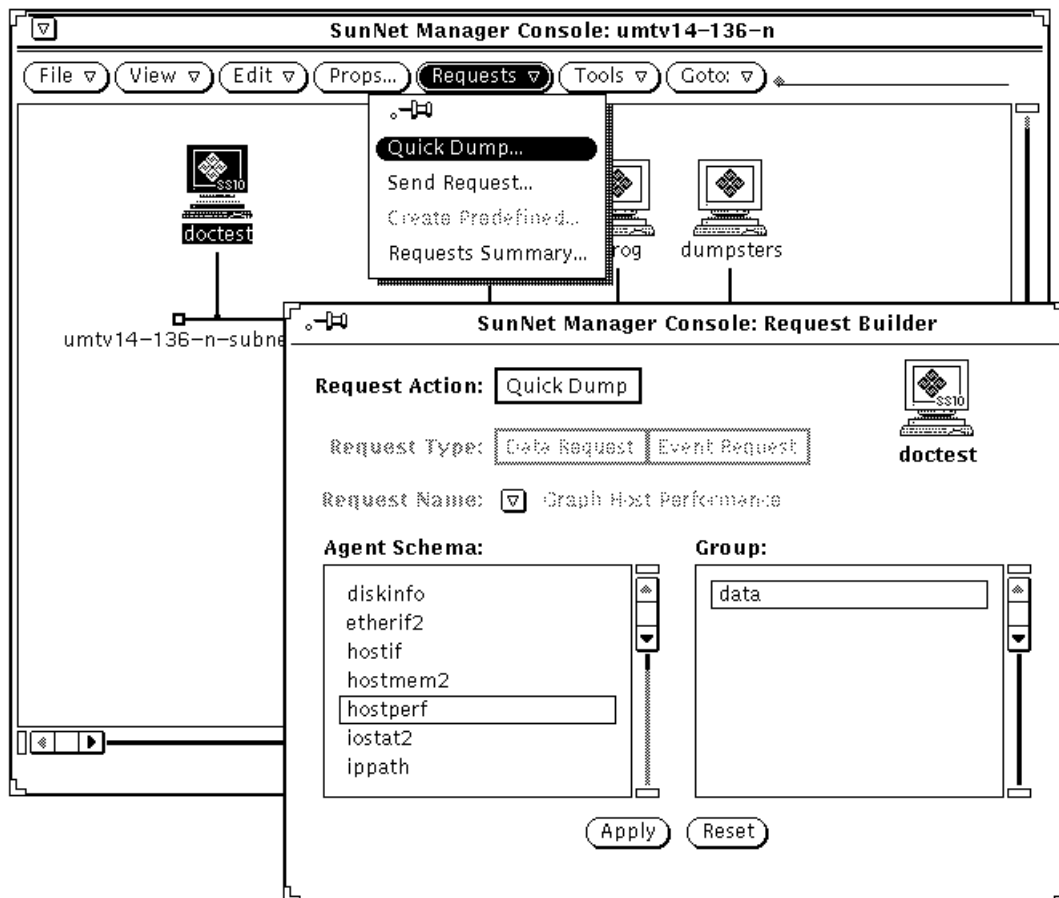


Figure 3-1 Requests — Quick Dump Request Window

Once you have selected the Agent Schema and Group that you desire, click **SELECT** on the **Apply** button. The **Reset** button can be used at any time to return to the base Quick Dump Request Builder window. See Figure 3-3 on page 3-5, for an example of a Quick Dump Report window that will be displayed as a result of sending a Quick Dump request.

3.1.2 Quick Dump Option via the Glyph Menu

A Quick Dump request for a one-time data report of all the attributes in an agent group can also be sent by using the Quick Dump option of the element Glyph menu. If the Quick Dump item is dimmed, it means that there are no readable attributes for any agent on the target system. Figure 3-2 is an example of launching a Quick Dump request for the Agent Schema named `diskinfo` of Group type `diskSpace` from the element `doctest`'s Glyph menu.

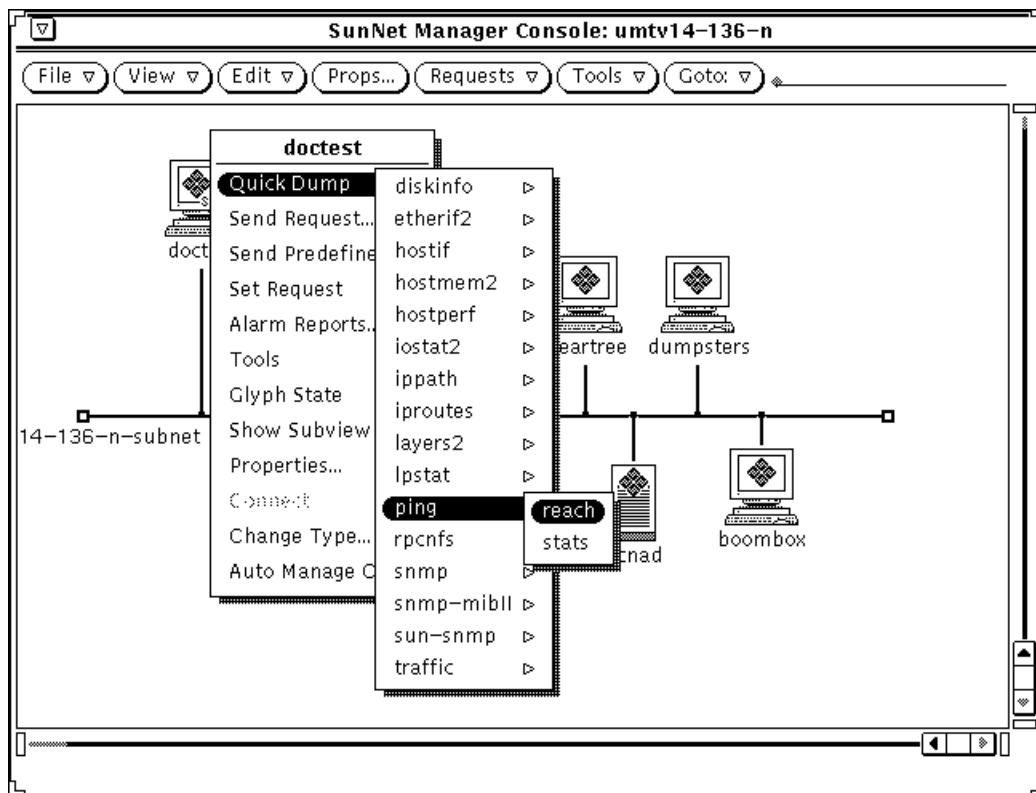


Figure 3-2 Sending a Quick Dump Request

Quick Dump results are displayed in a pop-up window, as shown in Figure 3-3, and are *not* stored in the Data Reports window.

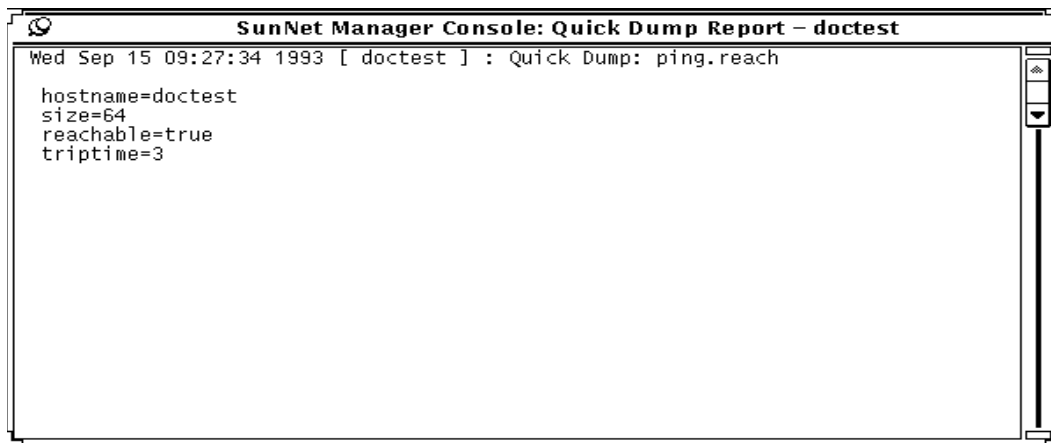


Figure 3-3 Quick Dump Report Pop-Up Window

Table attributes are displayed in tabular form when possible; the pop-up window is stretched as necessary. To dismiss the window, click on the pushpin. If an error occurs during a Quick Dump request, the pop-up window does not appear and a message is displayed in the Console footer; the text of the error can be displayed through the View►Error Reports window.

3.2 Predefined Requests Supplied with SNM 2.2.3

The SNM 2.2.3 product provides a number of predefined data and event request records in the `.SNMpredefined` file in the `struct` directory. Upon invoking the SNM Console, a check is made to see if the `$HOME/.SNMpredefined` file exists. If it does, the file is loaded into the runtime database for use by the Console. If it does not exist, the `.SNMpredefined` file is loaded into the runtime database. The first time the SNM Console is invoked, the `.SNMpredefined` file will be loaded into the runtime database. The records in this file, along with any predefined data and event requests you modify or create after the SNM Console is started, are stored in your `$HOME/.SNMpredefined` file upon exiting from the SNM Console. You also have the option of saving the runtime database version of the predefined data and event requests records by using the File►Save►Predefined Requests option. Using this option you can save these

records to your `$HOME/.SNMpredefined` file or to any other file you desire. For information about creating, modifying, deleting, and sending predefined data and event requests, refer to the following sections of this chapter.

3.2.1 Predefined Data Request Records

The following table lists the predefined data requests that are provided with this product, their Agent, Group, and Attribute settings, as well as a description of their intended function(s).

Table 3-1 SNM Supplied Predefined Data Requests

Request Name	Agent Name	Group Name	Attributes Supported	Function
Graph Host Performance	hostperf	data	cpu% intr disk ipkts opkts	CPU utilization # of device interrupts # of disk transfers # of if input pkts # of if output pkts
Record Disk Space	diskinfo	diskSpace	all	disk space information
Record Host Performance	hostperf	data	all	Statistics of Host
Show Host Interfaces	hostif	if	all	host interface statistics
Show NFS Statistics	rpcnfs	client	all	RPC and NFS statistics
Show Path to Host	ippath	path	all	trace IP packet's path between proxy and target system
Show Routing Statistics	iproutes	routes	all	Routing statistics
Show snmp System Info	snmp	system	all	System information
Show snmp-mibII System Info	snmp-mibII	system	all	System information
Show sun-snmp System Info	sun-snmp	system	all	System information

3.2.2 Predefined Event Request Records

The following table lists the predefined event requests that are provided with this product, their Agent, Group, and Attribute settings, as well as a description of their intended function(s).

Table 3-2 SNM Supplied Predefined Event Requests

Request Name	Agent Name	Group Name	Attributes Supported	Event to be Reported
when Disk is Full	diskinfo	diskSpace	capacity	disk file system is full
If System Reboot	hostperf	data	uptime	if system reboots
when Printer Error	lpstat	status	statusCode	line printer error
when System is not Reachable	ping	reach	reachable	system not reachable

3.3 Creating, Modifying, or Deleting Predefined Requests

Predefined data and event requests are created using the SNM Console's Requests ► Create Predefined option. This is a global operation, not element based. Therefore, this menu item will be dimmed if an element has been selected prior to clicking the Requests button. To build a predefined data or event request, click

SELECT on the Create Predefined menu item with no elements having been selected. The window shown in Figure 3-4, will be displayed.

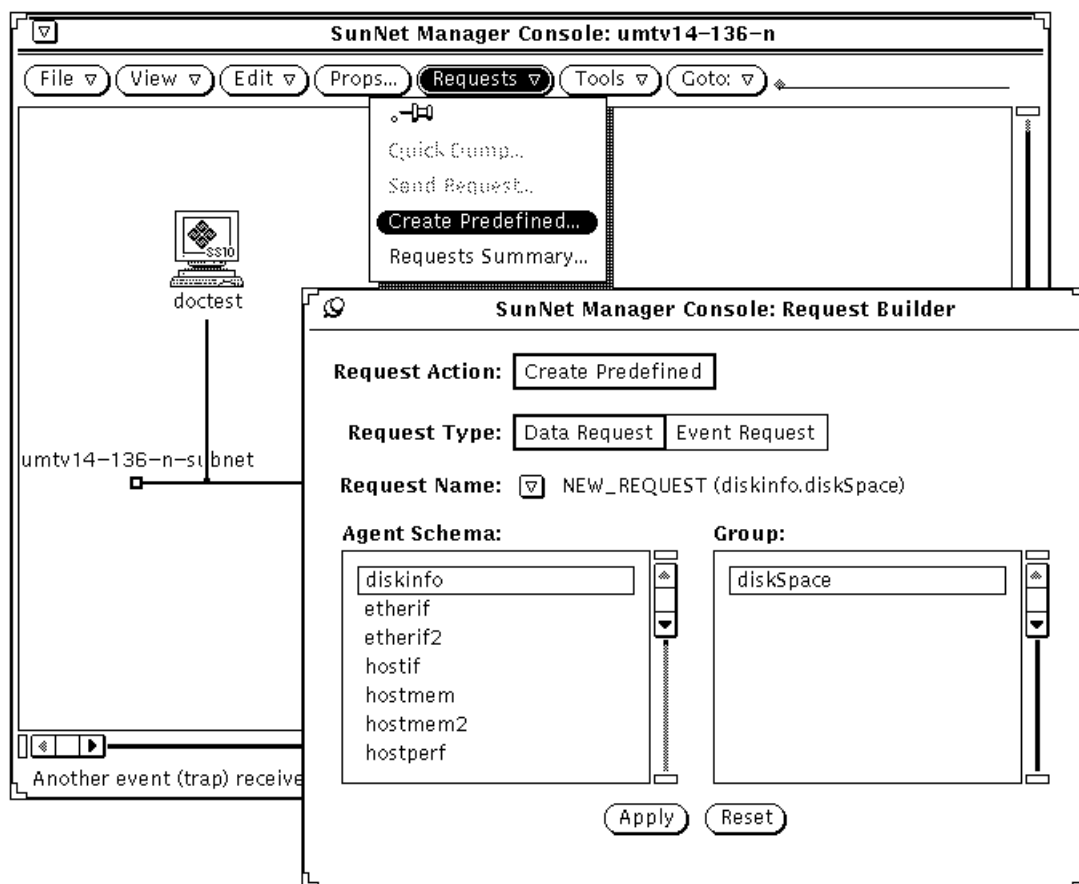


Figure 3-4 Requests — Create Predefined Request Window

where:

Request Action:

Is set to Create Predefined automatically.

Request Type:

Click SELECT on either the Data Request or Event Request field, depending on the type of predefined request you wish to build.

Request Name:

Press MENU on the abbreviated menu button and release over the name of the predefined data or event request you wish to use. This can be either a Sun provided predefined data or event request, a predefined data or event request you previously built and wish to modify, or a new predefined data or event request you wish to create (by choosing the default selection NEW_REQUEST.)

Agent Schema:

Click SELECT on the appropriate agent schema name that you want the data or event request predefined for.

Group:

Click SELECT on the appropriate group name that you want the data or event request predefined for.

Once you have specified the options that you desire, click on the Apply button. The Reset button can be used at any time to return to the base Create Predefined request window. See Section 3.3.1, “Creating, Modifying, or Deleting a Predefined Data Request,” below, for information on creating or modifying a predefined data request. See Section 3.3.2, “Creating, Modifying, or Deleting a Predefined Event Request,” on page 3-16 for information on creating or modifying a predefined event request.

3.3.1 Creating, Modifying, or Deleting a Predefined Data Request

The window shown in Figure 3-5 will be displayed once you have selected the Request Type, Request Name, Agent Schema, and Group Name and clicked on the Apply button in the Request Builder window shown in Figure 3-4.

SunNet Manager Console: Predefined Data Request

Request Name: Graph Host Performance

Agent Schema: hostperf

Group/Table: data

Polling Interval: 30 seconds

Count: 0 times

Key:

Options:

Restart: ☒

Defer Reports: ☐

On Completion: ☒ Save Request

Log to File:

To Program:

Selected Attributes:

- cpu%
- intr
- disk
- ipkts
- opkts

Available Attributes: ☒ cpu%

Data Log: ☒ True

Indicator: ☐

Strip Chart: ☒ None

Graph Tool: ☒ Absolute Values

Buttons: Define, Reset, Undefine, Add, Change, Delete

Request Name: Graph Host Performance.

Figure 3-5 Sample Predefined Data Request Template

3.3.1.1 Reporting Characteristics

The fields on the left side of the Predefined Data/Event Request window specify reporting characteristics. Each of the fields are described below.

Request Name:

The name you are assigning to the predefined request. You *must* specify a name for this request. Request Names are limited to a maximum of 64 characters. If you selected a predefined data or event request, the Request Name field will contain the name of the predefined data or event request you selected.

Note – The predefined data and event record names are case insensitive. Since the predefined data and event records reside in the runtime database with all of the elements, it is *highly* recommended that the names assigned to newly created predefined requests consist of at least two words, as is the case with the predefined data and event requests provided with this product. Otherwise, a conflict may arise between the name of an element and the name of the predefined data and/or event request record being created, as they must be unique. When an attempt to create a predefined data or event request with a duplicate name of an element or an existing predefined request is made, an error message will be placed in the error reports log and the predefined request will not be created.

Agent Schema:

The name of the agent schema the predefined data request is associated with.

Group/Table:

The name of the group or table associated with the agent schema the predefined data request is associated with.

Polling Interval:

Specifies the interval (in seconds) for the agent on the target system to send reports. An Interval value of 0 indicates that the agent should use its default interval. See the specific agent's `man` page for its default interval value.

Count:

Specifies the number of times for the agent on the target system to send reports. A count of 0 specifies that the agent is to send reports until the request is stopped or killed.

Key:

Identifies a particular row in a table (for example, `ie0` or `le0` in an Ethernet interface table). The use of a key is agent-specific and is indicated in the bottom left corner of the Data Request Properties window. See the `man` page for the particular agent for information regarding keys. If no key is specified, the entire table is returned.

Options:

Specifies any options (such as arguments) that the agent expects. To specify Options, type in an option string. For example, with SNMP requests you can specify an SNMP read-community name. The information that you can specify in the Options field is agent-specific—not all agents accept options. See the `man` page for the particular agent for information regarding options.

Restart:

Specifies whether the agent should attempt to restart the request if the system on which the agent is running reboots, the agent terminates unexpectedly, or the Console itself is restarted. If Restart is off (no check mark appears in the accompanying box), the request is discarded if the agent fails or if you quit the Console. If Restart is on (a check mark appears in the accompanying box), the request is restarted. The default setting of this field is determined by the Restart Request upon Agent Failure setting in the Console Properties Requests category. (Refer to Section 9.3, “Requests,” on page 9-6 for more information.) Click SELECT on the box next to this field to toggle the check mark on or off.

Defer Reports:

Specifies whether or not the agent should cache reports and send them to the Console only when asked to. If Defer Reports is on (a check mark appears in the accompanying box), the agent is directed to collect the statistics, but send them to the Console only when directed. Defer Reports tells the agent to cache the last 32 reports. If 32 reports are cached, or if the agent runs out of memory, the oldest report is deleted when a new report is cached. You should not specify Defer Reports for agents that return a large amount of data (for example, a routing table).

In many cases an agent collects information useful for debugging problems. This information might not be of interest in your normal daily operation. Furthermore, if a request were started before the error occurred, data reports will continually stream back to the Console causing unnecessary network traffic and increased CPU load. If, on the other hand, the request was started after the error happened, the debugging information would not have been collected. By setting Deferred Sending on, reports are held in the agent’s system until you need to ask for them.

The data request must be active for Defer Reports to work properly; that is, issue the request with a long Interval and high Count. Refer to Section 2.3, “Viewing and Modifying Requests,” on page 2-44 for information on how to obtain deferred reports.

On Completion:

Specifies whether the request should be deleted or saved upon completion. To specify that the request be saved, drag MENU over the On Completion abbreviated menu button—release MENU over Save Request. If you choose to save the request, the request glyph is dimmed after the request has been completed but remains in the view of the target element. You can examine, modify, or restart the saved request.

If you select Delete Request On Completion, then stopping the request is the same as killing it. (Refer to Section 2.3, “Viewing and Modifying Requests,” on page 2-44 for more information.)

Log to File:

Specifies the file name where reports are to be stored. If you specify a file name with no path, the SunNet Manager Console uses the current directory. If you fully qualify the file name and the left-most characters scroll left, they are not lost.

Note – The file specification is independent of the attribute-specific selections in the Data Request Properties window. Values for all attributes in the selected group are written to the specified file.

To Program:

A shell command line specifying a program or shell script to be run. Include the directory path, if necessary. Data reports are passed to the standard input of the program or shell script. Every active program to which data reports are sent keeps a file descriptor open. The default maximum number of file descriptors allowed for a single process is 64. You can increase this number up to 256 by using the limit command in the C-shell.

3.3.1.2 *Selecting Attributes*

The fields in the right portion of the Predefined Data/Event Request window are used to specify attributes.

Selected Attributes:

A scrolling list that contains those attributes for which you are requesting data. You choose an attribute and how you want the attribute data displayed with the fields beneath the Available Attribute values scrolling list

(described below). When you click SELECT on the Add button at the right of the window, the attribute is added to the Selected Attributes values scrolling list.

Following is a description of the Available Attributes and attribute display fields:

Available Attributes:

Specifies the attribute you want the agent to return data on. When the Predefined Data Request window is first displayed, the Available Attributes field displays the name of the first attribute in the group/table from the agent schema file. To specify a particular Available Attribute, press MENU over the Available Attributes abbreviated menu button and release MENU on the desired entry. When an attribute is selected, a description is displayed in the lower-right footer.

Note – If no attribute is specified, *all* attributes for the selected group or table are written to the Data Reports log. For tables, the values of the keys are also returned. However, if you specify a particular table attribute, the key values are not returned.

Data Log:

Specifies whether or not the attribute data should be written in the Data Reports Log. To specify that the data *not* be written in the Data Reports log, drag MENU over the Data Log abbreviated menu button—release MENU over False.

Indicator:

Specifies whether or not the attribute data should be displayed in an Indicator. An Indicator shows the last reported value for a particular attribute. Refer to Section 4.1.2, “Indicators,” on page 4-3 for more information.

To specify the Indicator display option, click SELECT on the Indicator box.

Strip Chart:

Specifies whether or not the attribute data should be displayed in a Strip Chart. A Strip Chart contains an auto-scaled strip chart of values for an attribute. Refer to Section 4.1.3, “Strip Charts,” on page 4-4 for more information.

To specify the Strip Chart display option, drag MENU over the Strip Chart abbreviated menu button—release MENU over either Absolute Values (chart the received values) or Delta Values (chart the differences between received values).

Graph Tool:

Specifies whether or not the attribute data should be displayed by the Grapher. The Graph Tool, also known as the Results Grapher, is an application that allows the display of data received by the Console. Only data of types integer, float, counter, gauge, timestamp, or UNIX time can be plotted. Refer to Chapter 7, “Results Grapher,” for more information about the Graph Tool. To specify the Graph Tool display option, drag MENU over the Graph Tool abbreviated menu button—release MENU over either Absolute Values (graph the received values) or Delta Values (graph the differences between received values).

3.3.1.3 *Adding Attributes*

The buttons to the right of the Selected Attributes field in the Predefined Data Request window allow you to add, change, and delete attribute selections in the Selected Attributes values scrolling list. The buttons are described below:

- Add tells the request manager to add the Available Attribute selected to the Selected Attributes scrolling list.
- Change tells the request manager to apply the changes made to the attribute setting(s).
- Delete tells the request manager to remove the Available Attribute selected from the Selected Attributes scrolling list.

An entry in the scrolling list may be selected at any time by pointing the mouse pointer at the entry and clicking the SELECT button. This updates the window to the settings for the selected attribute. An entry may be deleted by selecting it and clicking on the Delete button.

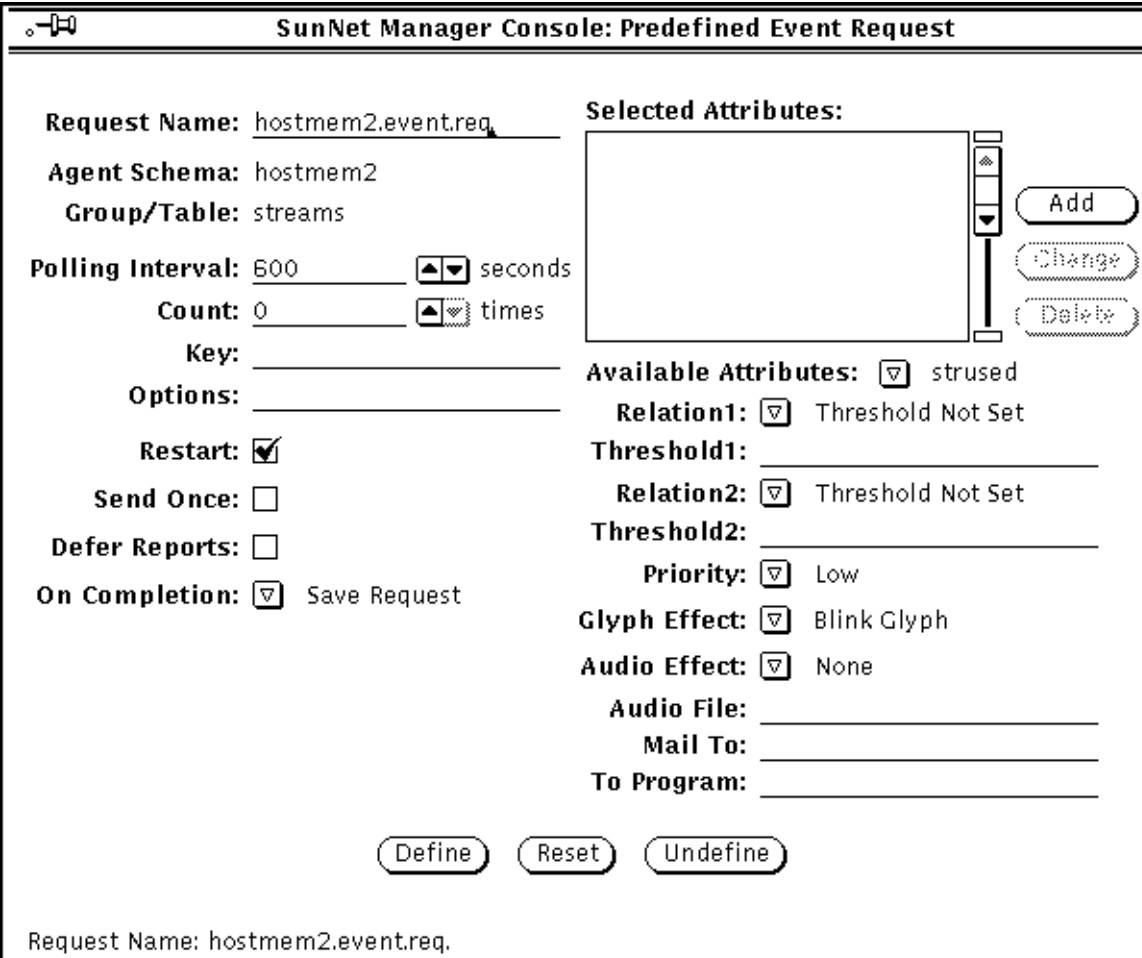
3.3.1.4 Creating, Modifying, and Deleting Predefined Data Records

The buttons at the bottom of the Predefined Data Request window allow you to define and reset all of the data record fields for the predefined data request, as well as delete the record from the runtime database. The buttons are described below:

- Define creates or updates the predefined data request runtime database record to the values specified in the Predefined Data Request Window.
- Reset returns the request specifications back to the originally applied selections.
- Undefine deletes the selected predefined data request from the runtime database record.

3.3.2 Creating, Modifying, or Deleting a Predefined Event Request

The window shown in Figure 3-6 will be displayed once you have selected the Request Type, Request Name, Agent Schema, and Group Name and clicked on the Apply button in the Request Builder window.



SunNet Manager Console: Predefined Event Request

Request Name: hostmem2.event.req

Agent Schema: hostmem2

Group/Table: streams

Polling Interval: 600 seconds

Count: 0 times

Key: _____

Options: _____

Restart: ☒

Send Once: ☐

Defer Reports: ☐

On Completion: Save Request

Selected Attributes:

Available Attributes: strused

Relation1: Threshold Not Set

Threshold1: _____

Relation2: Threshold Not Set

Threshold2: _____

Priority: Low

Glyph Effect: Blink Glyph

Audio Effect: None

Audio File: _____

Mail To: _____

To Program: _____

Request Name: hostmem2.event.req.

Figure 3-6 Sample Predefined Event Request Template

Descriptions of most of the reporting characteristics fields are provided in Section 3.3.1.1, "Reporting Characteristics," on page 3-10. There is however, an important distinction to note in the Polling Interval and Count fields for predefined event requests. For predefined event requests, these fields simply specify when the agent is to send a report. A report is forwarded to the Console system only if it has been determined that an event has occurred. For

agents that are shipped with SNM, the value reported for an attribute is the value noted at the reporting interval. Thus, it is possible for event conditions to occur *between* reporting intervals and not cause an event report.

If Send Once has been selected (a check mark appears in the accompanying box), the request is terminated after the Console receives the first event report. Click SELECT on the Send Once box to toggle the check mark on or off.

3.3.2.1 *Selecting Attributes*

The fields in the right portion of the Predefine Event Request properties window are used to specify attributes.

Selected Attributes:

A scrolling list that contains those attributes for which you are requesting event information. You specify attributes, event thresholds, and how you want the event signalled with the fields under the Selected Attributes values scrolling list. When you click SELECT on the Add button to the right of the Selected Attributes window, the attribute is added to the Selected Attributes values scrolling list.

Following is a description of the available attributes, event threshold, signal options, receiver, and priority fields:

Available Attributes:

Specifies the attributes that you want to use to specify an event. When the window is first displayed, the Available Attributes field displays the name of the first attribute in the group/table from the agent schema file. To specify an attribute, press MENU on the Available Attributes field and release the mouse button on the desired entry. When an attribute is selected, a description is displayed in the lower-right footer of the window.

Relation1, Relation2:

Specify threshold conditions for reporting an event. You can specify up to two threshold relational operators; the event is reported if *either* of the two threshold conditions are met. To set the threshold relational operator(s) for

Relation1 or Relation2, drag MENU over the Relation abbreviated menu button to display the following list of relational operators in a pop-up menu—release MENU over the desired relational operator.

Threshold Not Set means you have not set a threshold for this attribute. This is the default choice.

Equal To tells the agent to report an event if the attribute value equals “Threshold n ”.

Not Equal To tells the agent to report an event if the attribute value does not equal “Threshold n ”.

Less Than tells the agent to report an event if the attribute value is less than “Threshold n .”

Less Than Or Equal To tells the agent to report an event if the attribute value is less than or equal to “Threshold n .”

Greater Than tells the agent to report an event if the attribute value is greater than “Threshold n .”

Greater Than Or Equal To tells the agent to report an event if the attribute value is greater than or equal to “Threshold n .”

Changed tells the agent to report an event if the attribute value differs from the last sampled value.

Increased By tells the agent to report an event if the attribute value has increased by “Threshold n ” compared to the last sampled value.

Decreased By tells the agent to report an event if the attribute value has decreased by “Threshold n ” compared to the last sampled value.

Increased By More Than tells the agent to report an event if the attribute value has increased by more than “Threshold n ” compared to the last sampled value.

Increased By Less Than tells the agent to report an event if the attribute value has increased by less than “Threshold n ” compared to the last sampled value.

Decreased By More Than tells the agent to report an event if the attribute value has decreased by more than “Threshold n ” compared to the last sampled value.

Decreased By Less Than tells the agent to report an event if the attribute

value has decreased by less than “Threshold n ” compared to the last sampled value.

Note – For the Changed, Increased By, Decreased By, Increased By More Than, Increased By Less Than, Decreased By More Than, and Decreased By Less Than operators, the Count field of the report characteristics must not be ‘1.’ A Key must be supplied if a table of attributes is selected.

Threshold1, Threshold2:

Specify the threshold value if the selected operator is other than Threshold Not Set, or Changed.

Priority:

Specifies the priority of the request. Click SELECT on the desired priority button (Low, Medium, or High) for the request.

Glyph Effect:

Specifies a visible indicator when the specified event is reported. Press MENU on the Glyph Effect abbreviated menu button to display a signal options pop-up menu. Release MENU on the signal option you want. You can choose only one visual signal option. In order to have the decay-to-blue feature in effect, you must select the Priority by Color option. This feature is described under Priority by Color, below.

The following options are available:

Blink Glyph

Blinks the glyph of the element for which an event has been reported.

Dim Glyph

Dims the glyph of the element for which an event has been reported.

Priority by Color

Causes the glyph to change color, based on the priority of the reported event. Red indicates a high priority event, orange indicates medium priority, and yellow indicates low priority. If multiple reports are received for a system, the highest priority is used. In association with the Priority by Color option is the decay-to-blue feature. The decay-to-blue feature pertains to events and traps that have reached a threshold and then fallen below it, or that have had an event or trap stop being

reported. In response to these conditions, the glyph for the affected element turns blue. Once a glyph has decayed to blue, the event must be acknowledged in order for the glyph to return to its original color. To get a glyph to return to its original color, you can either:

- i. Select the Console's View►Event Summary option
 - ii. Highlight the name of the glyph
 - iii. Press the Drop from List button.
- Or

From the Glyph Menu for the specific element, pull right over the Glyph State►Normal option.

Audio Effect:

Specifies an audible indicator when the specified event is reported. Press MENU on the Audio Effect abbreviated menu button to display a signal options pop-up menu—drag and release MENU on the desired signal option. Only one audio signal option can be chosen. The following options are available:

Ring Bell sounds an audible bell on the SunNet Manager Console workstation.

Play Audio File causes the file specified in the Audio File field to be played.

Audio File:

Defines the audio file to be played when the specified event is reported. Type in the file name of the audio file. The option Play Audio File must be specified in the Audio Effect field. You can use the volume function of the Audio Tool in OpenWindows 3.2 to control the sound level.

Note – The Console must be running (not merely displaying) on a machine with an audio port. If you are running the Console on a server but displaying the Console windows on a local workstation, the Console will attempt to play the audio file on the server.

Mail To:

Defines one or more mail recipients to send the event report to when the specified event is reported. Type in a list of mail recipients. If there is more than one recipient, use a space between each entry.

To Program:

A shell command line specifying a program or shell script to be run. Include the directory path, if necessary. A new copy of the program or shell script is forked for each event report. The event report is passed to the standard input of the program or shell script. Every active program that receives an event report keeps a file descriptor open. The default maximum number of file descriptors allowed for a single process is 64. You can increase this number up to 256 by using the `limit` command in the C-shell.

3.3.2.2 Adding Attributes

The buttons to the right of the Selected Attributes field in the Predefined Event Request window allow you to add, change, and delete attribute selections in the Selected Attributes values scrolling list. The buttons are described below:

- Add tells the request manager to add the Available Attribute selected to the Selected Attributes scrolling list.
- Change tells the request manager to apply the changes made to the attribute setting(s).
- Delete tells the request manager to remove the selected attribute from the Selected Attributes scrolling list.

An entry in the scrolling list may be selected at any time by pointing the mouse pointer at the entry and clicking the SELECT button. This updates the window to the settings for the selected attribute. An entry may be deleted by selecting it and clicking on the Delete button.

3.3.2.3 Creating, Modifying, and Deleting Predefined Event Records

The buttons at the bottom of the Predefined Event Request window allow you to define and reset all of the event record fields for the predefined event request, as well as delete the record from the runtime database. The buttons are described below:

- Define creates or updates the predefined event request runtime database record to the values specified in the Predefined Event Request Window.
- Reset returns the request specifications back to the originally applied selections.

- **Undefine** deletes the selected predefined event request from the runtime database record.

3.4 *Sending Data and Event Requests*

This section describes how to build and send data and event requests. Data and event requests, when launched, send information such as the number of reports and reporting intervals to the specified agents on the target machine. The agents then automatically generate reports at specified intervals without any intervention (polling) from the Console. The activity daemon running on the Console workstation periodically verifies that the specified agent is running on the target machine. In the case of proxy agents, the proxy agent polls the target system at the intervals specified in the data or event request.

Data and event requests can be created and sent via the SNM Console Requests►Send Request menu item or via the Glyph►Send Request menu item (as in the past), or you may create predefined data and event requests via the SNM Console Requests►Create Predefined menu item. Refer to Section 3.3, “Creating, Modifying, or Deleting Predefined Requests,” on page 3-7 for information about creating predefined data and event requests. Predefined data and event requests can only be sent via the Glyph►Send Predefined menu item. Refer to Section 3.4.1, “Sending a Data Request,” below, for more information on sending data requests. Refer to Section 3.4.2, “Sending an Event Request,” on page 3-32 for more information on sending event requests.

3.4.1 *Sending a Data Request*

Data requests can be sent in one of two ways; via the SNM Console Requests►Send Request menu item or via the Glyph►Send Request menu. For a description of sending a data request via the Glyph►Send Request menu, refer to Section 3.4.1.1, “Glyph Menu – Send Data Request Option,” below. For a description of sending a data request via the SNM Console Requests►Send Request menu, refer to Section 3.4.1.2, “Requests Button – Send Data Request Option.”

3.4.1.1 Glyph Menu – Send Data Request Option

The Glyph►Send Request option allows you to send a data or event request. Upon selecting the Glyph►Send Request menu item, the window in Figure 3-7 will be displayed.

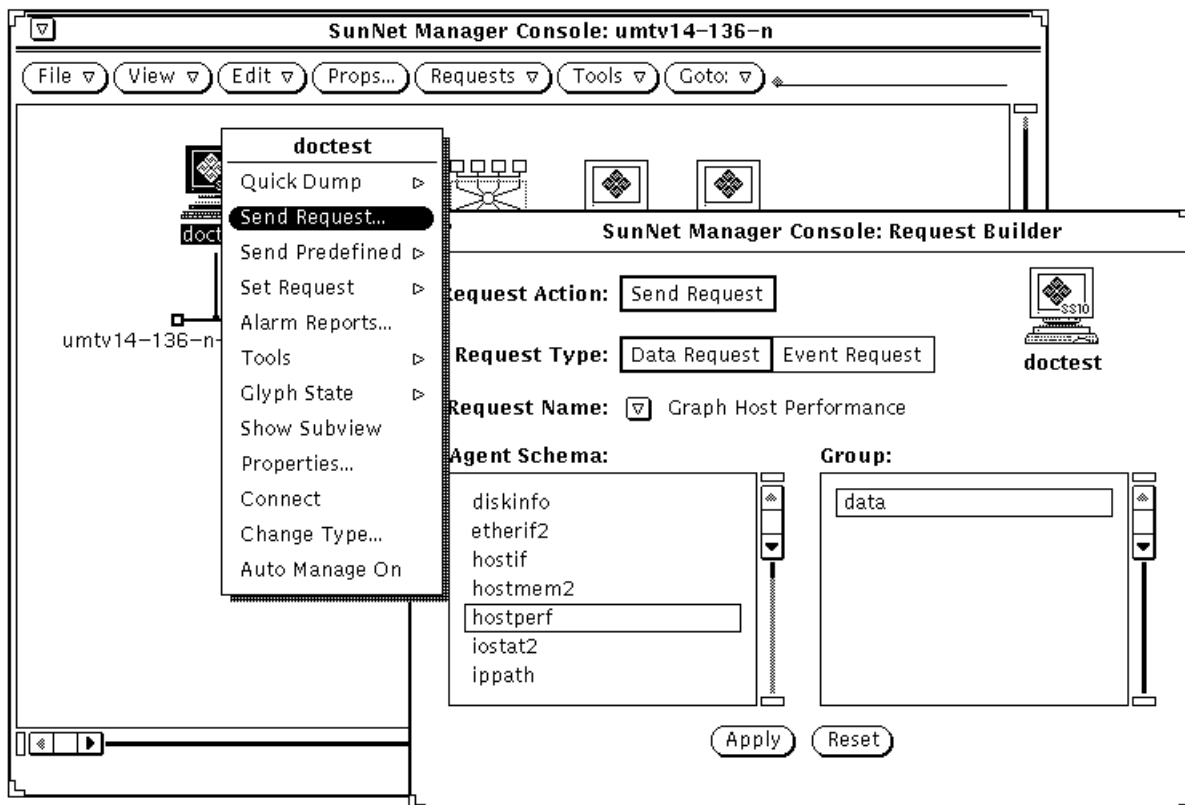


Figure 3-7 Glyph — Send Data Request Window

Once you have specified the options that you desire in the Request Builder template, click SELECT on the Apply button. Refer to Section 3.4.1.3, “Data Request Template,” on page 3-26 for a description of the template that will be displayed next. Again, specify the desired options and then click on the Start button to send the data request to the desired agent.

3.4.1.2 Requests Button – Send Data Request Option

The SNM Console's Requests►Send Request option can also be used to send a data or event request to the selected element. The Send Request option will be dimmed if an element was not selected prior to clicking the Requests button or if the selected element has no agents specified. Upon selecting the Requests►Send Request menu item, the window in Figure 3-8 will be displayed.

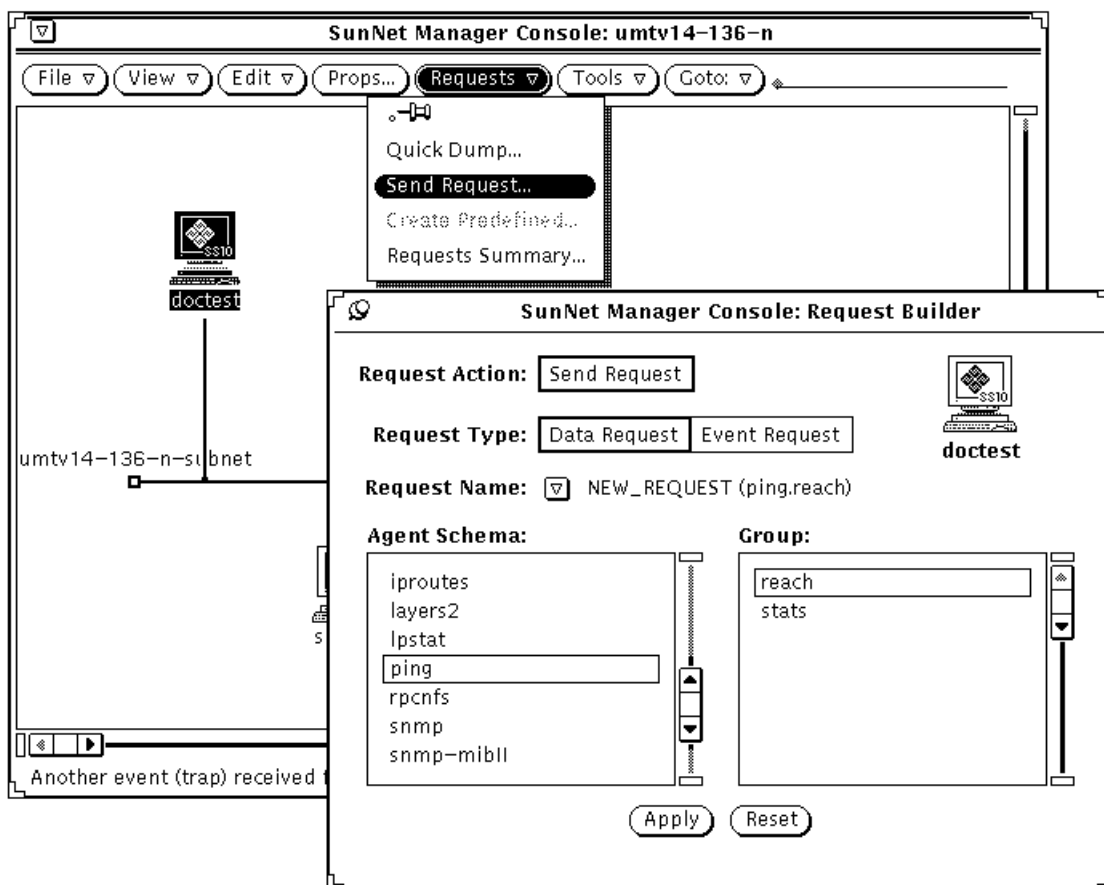


Figure 3-8 Requests — Send Data Request Window

The Request Action field is automatically set to Send Request. You select whether you want a data or event request to be built by clicking SELECT on the appropriate Request Type box. The Request Name is a pull down menu containing a list of the predefined data or event requests that exist (depending on the Request Type you specified) as well as a NEW_REQUEST (default) entry. You select the NEW_REQUEST entry to create a custom request.

Once you have specified the desired options in the Request Builder template, click on the Apply button. Refer to Section 3.4.1.3, “Data Request Template,” below, for a description of the template that will be displayed next. Again, specify the desired options and then click on the Start button to send the data request to the desired agent. If you selected one of the predefined requests, the template will have the appropriate information displayed in the various Data/Event Request template fields.

3.4.1.3 Data Request Template

Figure 3-9 is an example of a data request named DataReachable. A description of the fields in the template are provided below.

SunNet Manager Console: Data Request (doctest.ping.reach)

Name: DataReachable

Proxy System: localhost

Interval: 60

Count: 0

Key:

Restart: ☒

Defer Reports: ☐

On Completion: ☒ Save Request

Options:

Log to File:

To Program:

Attributes:

- hostname
- reachable

Attribute: ☒ reachable

Data Log: ☒ True

Indicator: ☒

Strip Chart: ☐ None

Graph Tool: ☐ None

Buttons: Start, Hold, Reset, Apply, Reset, Delete

Status: reachable information, Attribute reachable added.

Figure 3-9 Data Request Template

The description of the agent group for the request is displayed in the lower left corner of the window. The properties fields for this data request need to be specified at this time. Each of these fields is described below.

Reporting Characteristics

The fields on the left side of the Data/Event Request window specify reporting characteristics. Each of the fields are described below.

Name:

The optional name you wish to assign to the request. This allows you to assign a recognizable name to a request. The request manager assigns a request name if one is not specified, using the following format:

`<agent>.<group>.<number>` (for example, `hostperf.data.0`)

Proxy System:

The name of the system the data request is to be sent to.

Interval:

Specifies the interval (in seconds) for the agent on the target system to send reports. An Interval value of 0 indicates that the agent should use its default interval. See the agent's `man` page for its default interval.

Count:

Specifies the number of times for the agent on the target system to send reports. A count of 0 specifies that the agent is to send reports until the request is stopped or killed.

Key:

Identifies a particular row in a table (for example, `ie0` or `le0` in an Ethernet interface table). The use of a key is agent-specific and is indicated in the bottom left corner of the Data Request Properties window. See the `man` page for the particular agent for information regarding keys. If no key is specified, the entire table is returned.

Restart:

Specifies whether the agent should attempt to restart the request if the system on which the agent is running reboots, the agent terminates unexpectedly, or the Console itself is restarted. If Restart is off (no check mark appears in the accompanying box), the request is discarded if the agent fails or if you quit the Console. If Restart is on (a check mark appears in the accompanying box), the request is restarted. The default setting of this

field is determined by the Restart Request upon Agent Failure setting in the Console Properties Requests category. (Refer to Section 9.3, “Requests,” on page 9-6 for more information.) Click SELECT on the box next to this field to toggle the check mark on or off.

Defer Reports:

Specifies whether or not the agent should cache reports and send them to the Console only when asked to. If Defer Reports is on (a check mark appears in the accompanying box), the agent is directed to collect the statistics, but send them to the Console only when directed. Defer Reports tells the agent to cache the last 32 reports. If 32 reports are cached, or if the agent runs out of memory, the oldest report is deleted when a new report is cached. You should not specify Defer Reports for agents that return a large amount of data (for example, a routing table).

In many cases an agent collects information useful for debugging problems. This information might not be of interest in your normal daily operation. Furthermore, if a request were started before the error occurred, data reports will continually stream back to the Console causing unnecessary network traffic and increased CPU load. If, on the other hand, the request was started after the error happened, the debugging information would not have been collected. By setting Deferred Sending on, reports are held in the agent’s system until you need to ask for them.

The data request must be active for Defer Reports to work properly; that is, issue the request with a long Interval and high Count. Refer to Section 2.3, “Viewing and Modifying Requests,” on page 2-44 for more information on how to obtain deferred reports.

On Completion:

Specifies whether the request should be deleted or saved upon completion. To specify that the request be saved, drag MENU over the On Completion abbreviated menu button—release MENU over Save Request. If you choose to save the request, the request glyph is dimmed after the request has been completed but remains in the view of the target element. You can examine, modify, or restart the saved request.

If you choose Delete Request on Completion, stopping the request is the same as killing it. (Refer to Section 2.3, “Viewing and Modifying Requests,” on page 2-44 for more information.)

Options:

Specifies any options (such as arguments) that the agent expects. To specify Options, type in an option string. For example, with SNMP requests you can specify an SNMP read-community name. The information that you can specify in the Options field is agent-specific—not all agents accept options. See the `man` page for the particular agent for information regarding options.

Log to File:

Specifies the file name where reports are to be stored. If you specify a file name with no path, the SunNet Manager Console uses the current directory. If you fully qualify the file name and the left-most characters scroll left, they are not lost.

Note – The file specification is independent of the attribute-specific selections in the Data Request Properties window. Values for all attributes in the selected group are written to the specified file.

To Program:

A shell command line specifying a program or shell script to be run. Include the directory path, if necessary. Data reports are passed to the standard input of the program or shell script. Every active program to which data reports are sent keeps a file descriptor open. The default maximum number of file descriptors allowed for a single process is 64. You can increase this number up to 256 by using the `limit` command in the C-shell.

Selecting Attributes

The fields in the right portion of the Data/Event Request window are used to specify attributes.

Attributes:

A scrolling list that contains those attributes for which you are requesting data. You choose an attribute and how you want the attribute data displayed with the fields beneath the Attributes scrolling list (described below). When you click SELECT on the Apply button at the bottom right of the window, the attribute is added to the Attributes scrolling list.

Following is a description of the Attribute and attribute display fields:

Attribute:

Specifies the attribute you want the agent to return data on. When the Data Request window is first displayed, the Attribute field displays the name of the first attribute in the group/table from the agent schema file. To specify a particular Attribute, press MENU on the Attribute abbreviated menu button and release MENU on the desired entry. When an attribute is selected, a description is displayed in the lower-right footer.

Note – If no attribute is specified, *all* attributes for the selected group or table are written to the Data Reports log. For tables, the values of the keys are also returned. However, if you specify a particular table attribute, the key values are not returned.

Data Log:

Specifies whether or not the attribute data should be written in the Data Reports Log. To specify that the data *not* be written in the Data Reports log, drag MENU over the Data Log abbreviated menu button—release MENU over False.

Indicator:

Specifies whether or not the attribute data should be displayed in an Indicator. An Indicator shows the last reported value for a particular attribute. Refer to Section 4.1.2, “Indicators,” on page 4-3 for more information.

To specify the Indicator display option, click SELECT on the Indicator box.

Strip Chart:

Specifies whether or not the attribute data should be displayed in a Strip Chart. A Strip Chart contains an auto-scaled strip chart of values for an attribute. Refer to Section 4.1.3, “Strip Charts,” on page 4-4 for more information.

To specify the Strip Chart display option, drag MENU over the Strip Chart abbreviated menu button—release MENU over either Absolute Values (chart the received values) or Delta Values (chart the differences between received values).

Graph Tool:

Specifies whether or not the attribute data should be displayed by the Grapher. The Graph Tool, also known as the Results Grapher, is an application that allows the display of data received by the Console. Only

data of types integer, float, counter, gauge, timestamp, or UNIX time can be plotted. Refer to Chapter 7, “Results Grapher,” for more information about the Graph Tool. To specify the Graph Tool display option, drag MENU over the Graph Tool abbreviated menu button—release MENU over either Absolute Values (graph the received values) or Delta Values (graph the differences between received values).

Adding Attributes

The buttons below the Attributes fields in the Data/Event Request window allow you to add, change, and delete attribute selections in the Attributes scrolling list. The buttons are described below:

- Apply tells the request manager to add the Attribute selected to the Attributes scrolling list.
- Reset tells the request manager to reset the Attributes scrolling list back to its default value.
- Delete tells the request manager to remove the Attribute selected from the Attributes scrolling list.

An entry in the scrolling list may be selected at any time by pointing the mouse pointer at the entry and clicking the SELECT button. This updates the window to the settings for the selected attribute. An entry may be deleted by selecting it and clicking on the Delete button.

Sending the Data/Event Request

The buttons at the bottom left of the Data/Event Request window allow you to send, hold, and reset the data/event request. The buttons are described below:

- Start tells the request manager to send the data/event request.
- Hold tells the request manager to hold the data/event request.
- Reset tells the request manager to set the data/event request to its default values.

3.4.2 Sending an Event Request

An event request allows the SunNet Manager Console user to tell an agent to report only when specific conditions are met. All event reports are logged to the Event/Trap Log. You can send an event request to a specific element by using the Glyph►Send Request option or the SNM Console's Requests►Send Requests option for the target element.

3.4.2.1 Glyph Menu – Send Event Request Option

The Glyph►Send Request option allows you to send a data or event request. Upon selecting the Send Request menu item, the window in Figure 3-10 will be displayed.

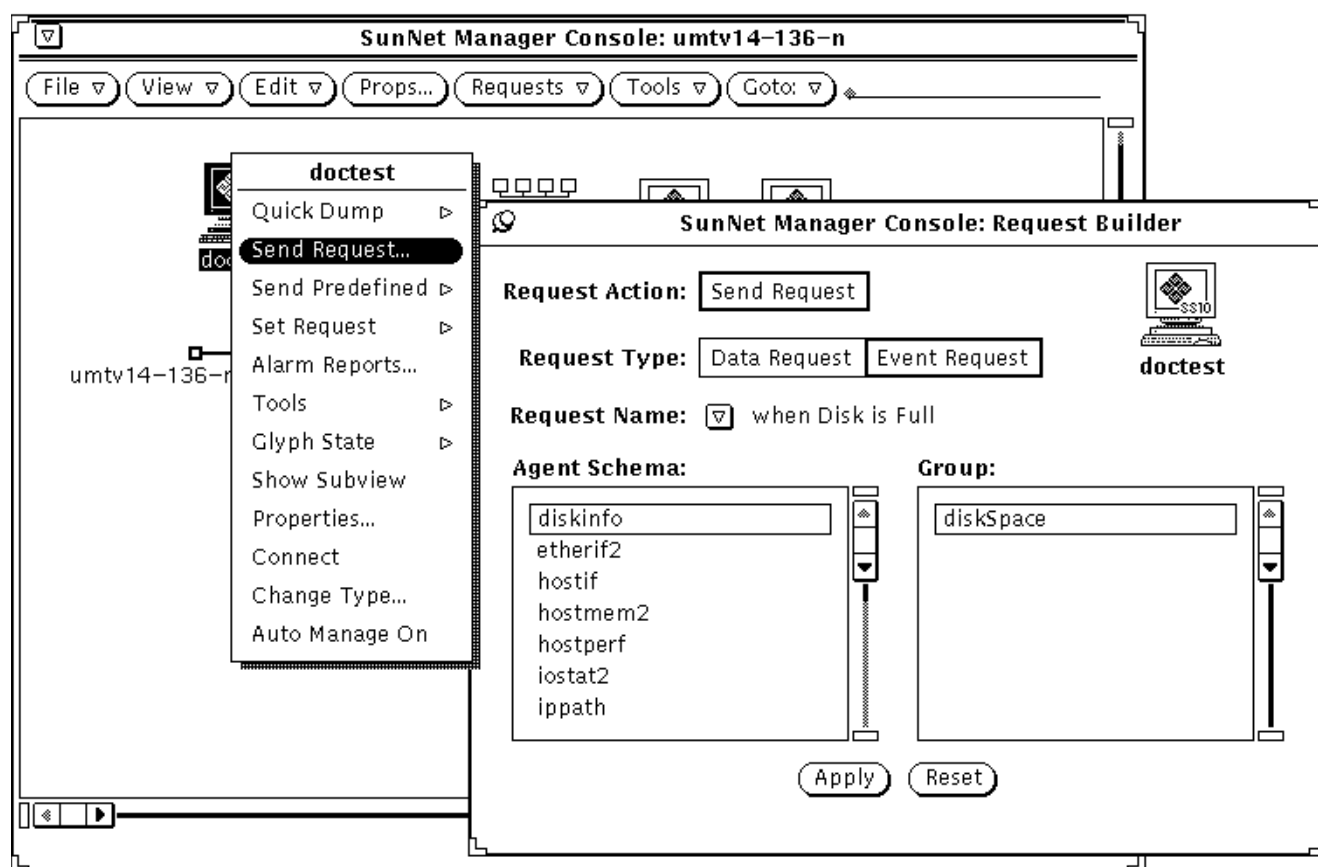


Figure 3-10 Glyph Menu — Send Event Request Window

Select Event Request as the Request Type. Once you have specified the options that you desire in the Request Builder template, click on the Apply button. Refer to Section 3.4.2.3, “Event Request Template,” on page 3-35 for a description of the template that will be displayed next. Again, specify the desired options and then click on the Start button to send the data request to the desired agent.

3.4.2.2 Requests Button – Send Event Request Option

The SNM Console's Requests►Send Request option is used to send a data or event request from the selected element. The Send option will be dimmed if an element was not selected prior to clicking the Requests button or if the selected element has no agents specified. Upon selecting the Requests►Send Request menu item, the window in Figure 3-11 will be displayed.

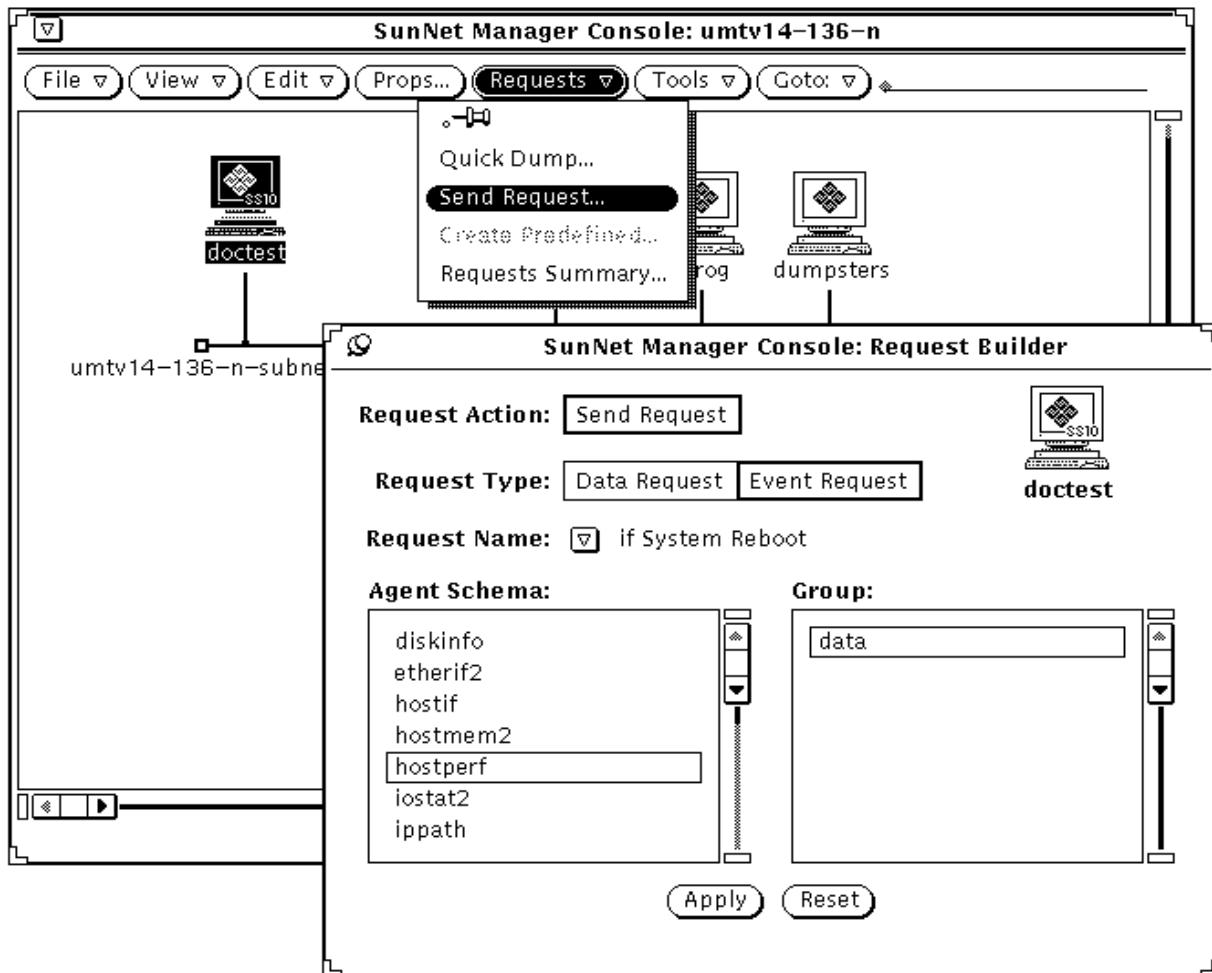


Figure 3-11 Requests — Send Event Request Window

Select Event Request as the Request Type. Once you have specified the options that you desire in the Request Builder template, click on the Apply button. Refer to Section 3.4.2.3, “Event Request Template,” below, for a description of the template that will be displayed next. Again, specify the desired options and then click on the Start button to send the event request to the desired agent.

3.4.2.3 Event Request Template

Figure 3-12 is an example of an event request named if System Reboot. A description of the fields in the template are provided below.

SunNet Manager Console: Event Request (doctest.hostperf.data)

Name: if System Reboot

Proxy System: doctest

Interval: 600

Count: 0

Key:

Restart: ☒

Send Once: ☐

Defer Reports: ☐

On Completion: ☒ Save Request

Options:

Attributes: uptime

Attribute: cpu%

Relation1: Threshold Not Set

Threshold1:

Relation2: Threshold Not Set

Threshold2:

Priority: Low

Glyph Effect: Blink Glyph

Audio Effect: None

Audio File:

Mail To:

To Program:

Buttons: Start, Hold, Reset, Apply, Reset, Delete

Host performance data **CPU utilization**

Figure 3-12 Event Request Template

Descriptions of most of the reporting characteristics fields are provided in Section , “Reporting Characteristics,” on page 3-27. There is however, an important distinction to note in the Interval and Count fields for event requests. For event requests, these fields simply specify when the agent is to

send a report. A report is forwarded to the Console system only if it has been determined that an event has occurred. For agents that are shipped with SNM, the value reported for an attribute is the value noted at the reporting interval. Thus, it is possible for event conditions to occur *between* reporting intervals and not cause an event report.

If Send Once has been selected (a check mark appears in the accompanying box), the request is terminated after the Console receives the first event report. Click SELECT on the Send Once box to toggle the check mark on or off.

Descriptions of the Attribute fields are provided in Section 3.3.2.1, "Selecting Attributes," on page 3-18.

3.4.2.4 Example: Using an Event Request to Manage SNM Log Files

You can use event requests to help you manage the management station itself. In this example we define an event request to generate an alarm when the file system where the SNM Console log files are mounted (`/var`) is more than 90% full.

We first click SELECT the glyph representing “gibber” — our management station — and then we invoke the Request Builder via the Requests►Send Request option. As shown in Figure 3-13, we select the `diskinfo` agent and Event Request in the Request Builder.

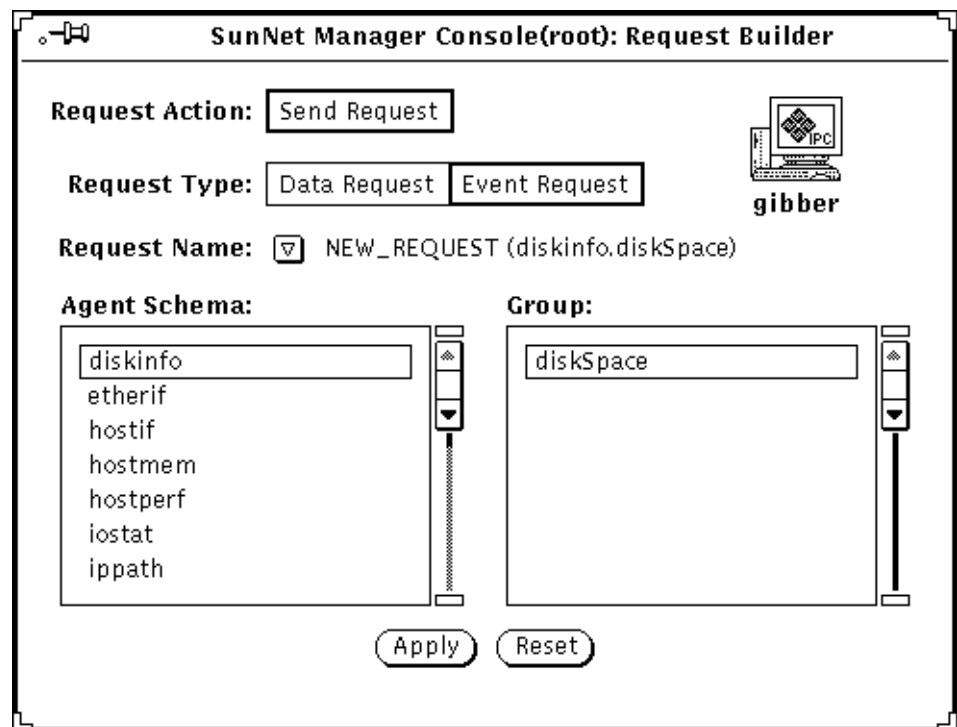


Figure 3-13 Managing SNM Log Files: Request Builder Selection

When we click SELECT the “Apply” button, we get the Event Request template, as shown in Figure 3-14.

SunNet Manager Console(root): Event Request (gibber.diskinfo.diskSpace)

Name: check_SNM_logfiles

Proxy System: _____

Interval: 900 [▲▼]

Count: 0 [▲▼]

Key: /var

Restart: ☒

Send Once: ☐

Defer Reports: ☐

On Completion: [▼] Save Request

Options: _____

Attributes:

capacity

Attribute: [▼] capacity

Relation1: [▼] Greater Than

Threshold1: 90

Relation2: [▼] Threshold Not Set

Threshold2: _____

Priority: [▼] High

Glyph Effect: [▼] Color by Priority

Audio Effect: [▼] None

Audio File: _____

Mail To: _____

To Program: _____

[Restart] [Stop] [Reset]

[Apply] [Reset] [Delete]

disk space information: Key is mountedOn

Figure 3-14 Managing SNM Log Files: Defining the Event Request

We've defined the various fields as follows:

- We've selected the "capacity" attribute which designates the percentage of total capacity that is used. The condition defining the event is capacity being Greater Than 90.
- The meaning of the "Key" field varies depending on the agent. For the diskinfo agent, Key specifies the relevant disk partition — the file system that would be indicated in the "Mounted On" column in a `df` command output, for example. In this case `/var`, where the log files are situated, is a separate partition.
- We've selected a polling interval of 15 minutes.

- If the event condition is met, the glyph “gibber” will turn red since we’ve selected Color By Priority.
- Count is set to 0 to indicate the request should continue until we stop it, and “Save Request” is selected so that the request will not be deleted if we do stop it.

3.5 *Sending Predefined Data and Event Requests*

The Glyph►Send Predefined option allows you to send a previously created predefined data or event request. Predefined data and event requests are built by using the SNM Console’s Requests►Create Predefine option and assigning a Request Name to the request being built. The Predefined data and event request information are stored in ASCII data record files. The location of the files is specified using the \$HOME environment variable. The default location is the login user’s home directory. Send Predefined displays a submenu that allows you to select a data or event request. Pull right to the desired element and release the menu button.

3.5.1 *Sending a Predefined Data Request*

Figure 3-15 shows an example of selecting the Glyph menu’s Send Predefined option to send a predefined data request. If the Send Predefined menu item is greyed out, it means the user selected agent schema (in the element’s Property sheet) associated with the predefined request does not exist on the target system. Refer to Section 3.3.1, “Creating, Modifying, or Deleting a Predefined Data Request,” on page 3-9, for a description of how to create or modify a predefined data request. Refer to Section 3.2, “Predefined Requests Supplied with SNM 2.2.2,” on page 3-5 for information pertaining to the predefined data and event requests that are provided with this product.

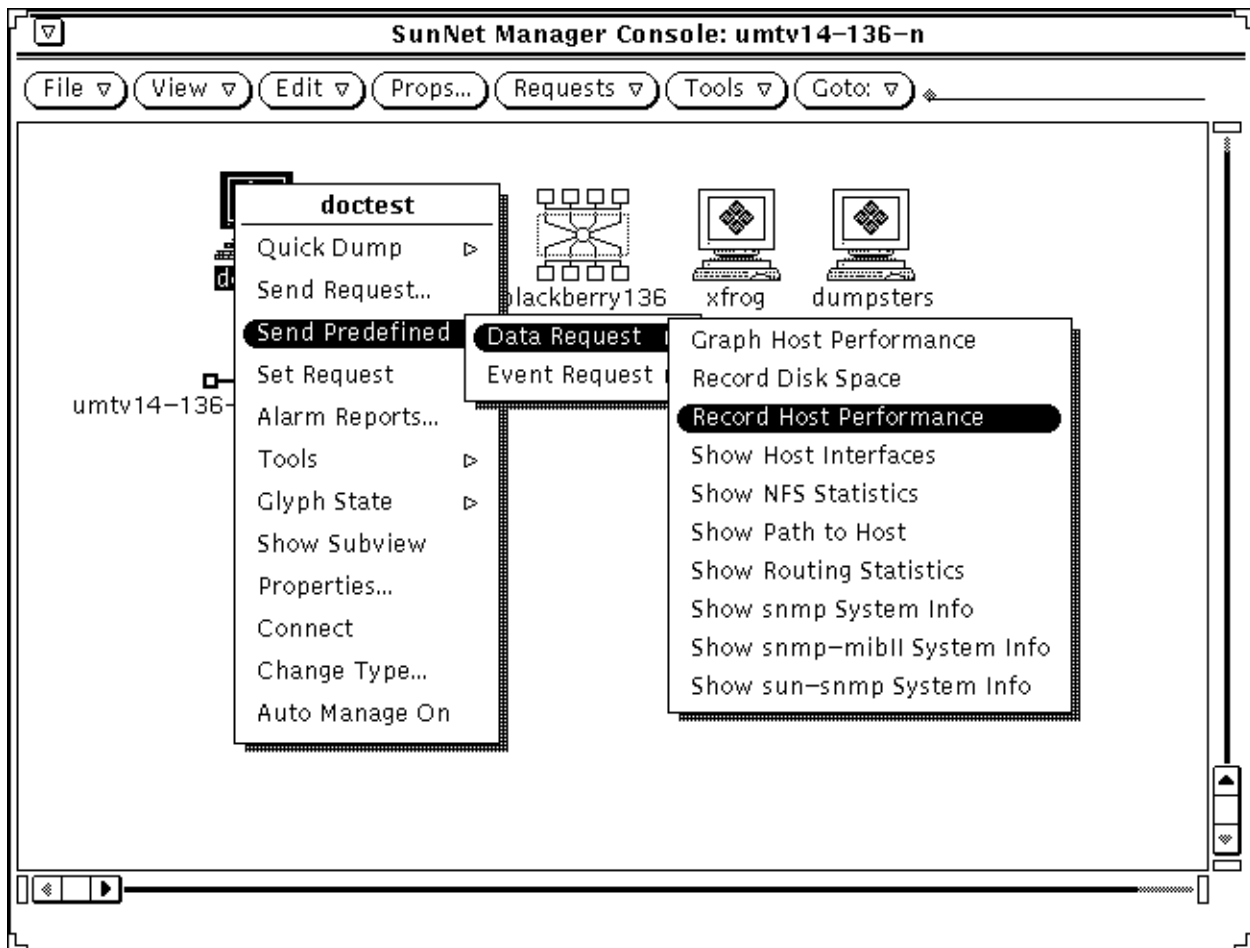


Figure 3-15 Sample Glyph — Send Predefined Data Request Menus

3.5.2 Sending a Predefined Event Request

Figure 3-16 shows an example of selecting the Glyph menu's Send Predefined option to send a predefined event request. If the Send Predefined menu item is greyed out, it means the user selected agent schema (in the element's Property sheet) associated with the predefined request does not exist on the target system. Refer to Section 3.3.2, "Creating, Modifying, or Deleting a Predefined

Event Request,” on page 3-16 for a description of how to create a predefined event request. Refer to Section 3.2, “Predefined Requests Supplied with SNM 2.2.2,” on page 3-5 for information pertaining to the predefined data and event requests that are provided with this product.

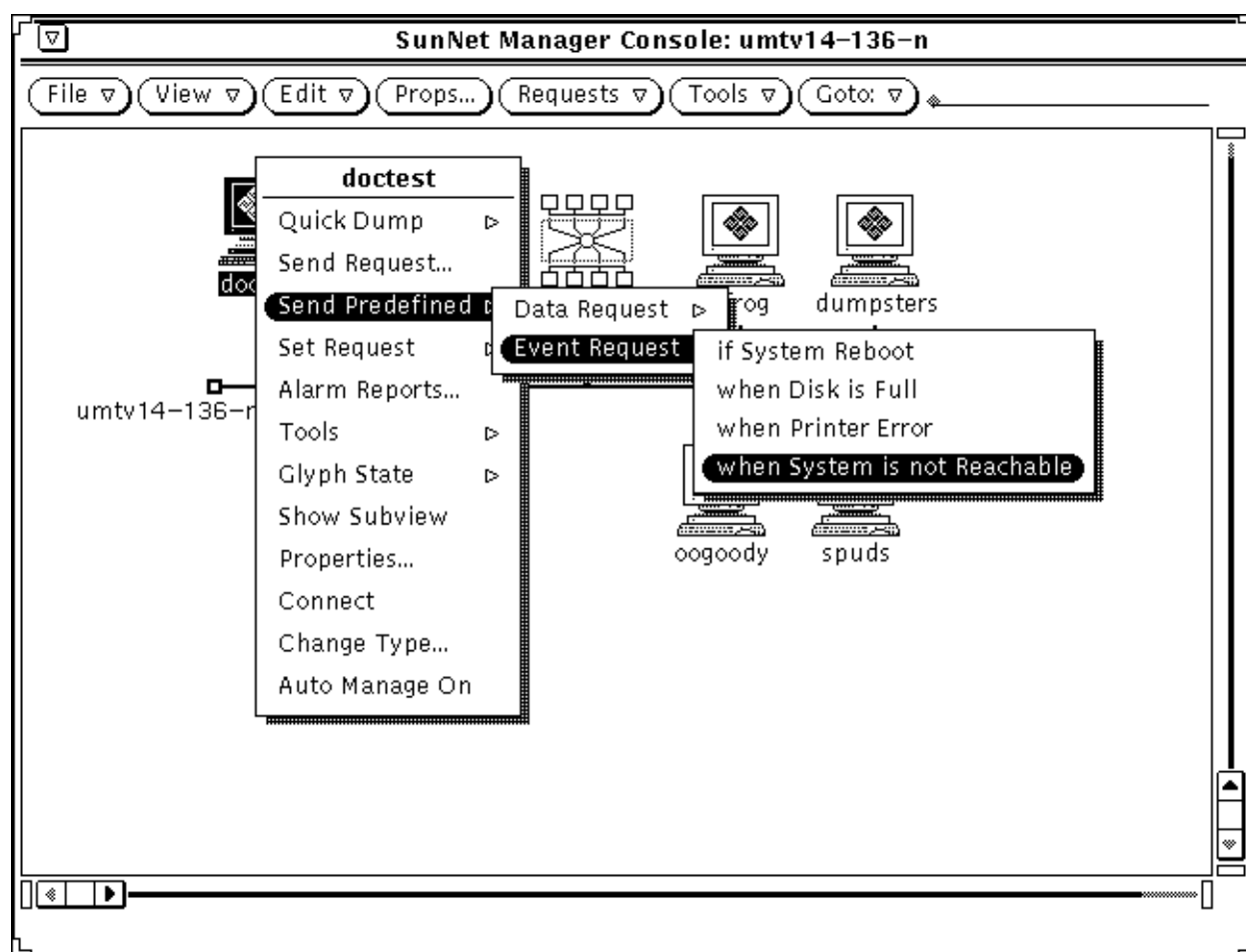


Figure 3-16 Sample Glyph — Send Predefined Event Request Menus

Viewing Reports and Logs



<i>Data Reports</i>	<i>page 4-1</i>
<i>Event Reports</i>	<i>page 4-6</i>
<i>Viewing Alarm Reports</i>	<i>page 4-9</i>

This chapter describes several ways of viewing output from data and event requests. Additionally, you can view data reports or event/trap reports with the SunNet Manager Results Browser and Results Grapher. (See Chapter 6, “Results Browser,” and Chapter 7, “Results Grapher,” for more information.)

4.1 Data Reports

For data requests, you specify how you want the attribute data displayed (data reports log, Strip Chart, or Indicator) in the Data Request window. You view the Data Reports log through the SNM Console View menu, while Strip Charts and Indicators are displayed in the view from which the data request was launched. The following subsections discuss each of these types of displays.

4.1.1 Data Reports Log

The Data Reports log contains data reports, in ASCII, for all elements managed by the SunNet Manager Console (see Figure 4-1). You display the Data Reports log by selecting the SNM Console View ► Data Reports menu item.

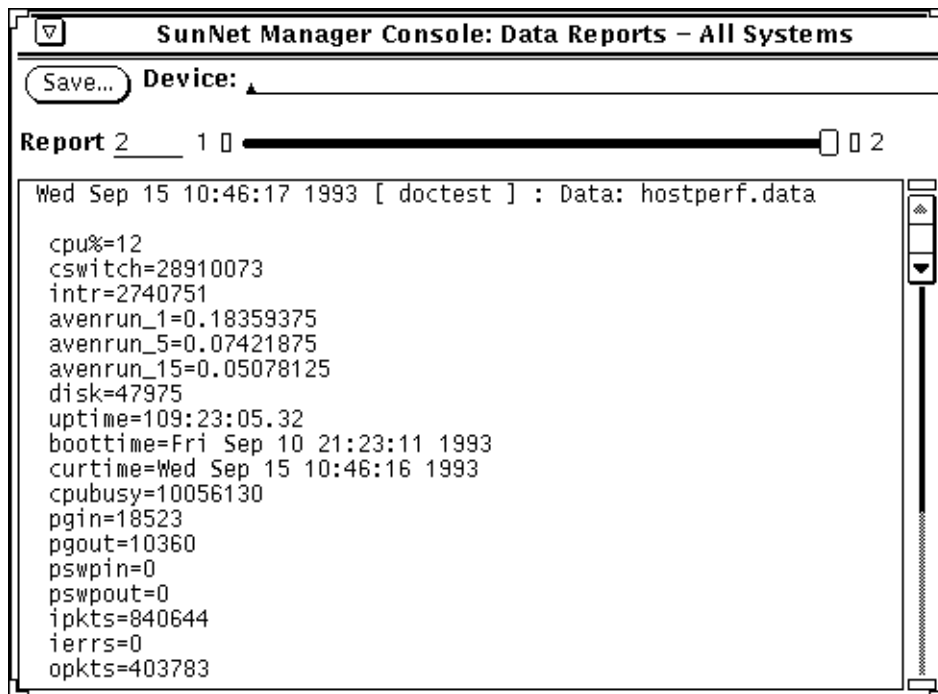


Figure 4-1 View — Data Reports Window

Data report entries have the following format:

`<timestamp> [<element_name>] : <report type>: '<request_name>'`

Table attributes are displayed in tabular form when possible. Attributes that are not tables are displayed in the format:

`<attribute_name> = <attribute_value>`

To view data reports, drag the slider to the right or the left. To look at all data reports for all elements, leave the Device field blank. To examine data reports for a particular element instance, type in the element name after the Device prompt and press Return. If you wish to look at Data Report log entries for all the elements again, press ctrl-u to clear the Device field and press Return.

The Data Reports log contains the most recently-received data reports for all systems. The currently displayed report is indicated on the Report line, while the total number of entries is displayed at the right of the slider. If you specify an element name in the Device field, the number of entries reflects the total number of entries for that element.

Data reports are allocated a maximum number of entries in the Data Reports log. By default, the maximum number of data reports is 1000. When the allocation limit is reached for either data reports, the oldest data report or error message is deleted. You can change the maximum number of entries by modifying the Maximum Data Reports setting in the Console Properties Miscellaneous category (refer to Section 9.8, “Miscellaneous,” on page 9-22 for more information).

Save writes the contents of the Data Reports log to disk. Selecting this button displays a pop-up window in which you can specify the file name and path of the disk file. You can use Save after specifying a Device name to save a subset of the Data Reports log to disk.

Use the scroll elevator to scroll through an entry that spans multiple panes of the Data Report window. Any number of Data Report log windows can be displayed simultaneously, allowing you to view the log entries for individual systems side by side.

Note – The contents of the Data Report log window are subject to configured XView limitations. This may cause data requests for very large tables to not be displayed in the Data Reports log window. If this happens, increase the value of the `text.maxDocumentSize` parameter in your `.Xdefaults` file, reload the resource into the X resource manager, and restart the Console.

4.1.2 Indicators

An Indicator shows the last reported value for a particular attribute. Indicators are displayed in the format of `<label>:<value>`, where the `<label>` consists of the element name, agent name, and attribute. For example:

```
poignant.etherif.ipkts: 6892892
```

The SunNet Manager Console displays a maximum of 64 indicators in the View where the data request selection process is begun. Indicators are positioned in the View using the same algorithm applied to newly-created elements.

Indicators may be edited (cut, copied, or pasted) in the same manner as other elements. For example, you can position an Indicator adjacent to the glyph for the element being monitored.

As with any other element in the SunNet Manager Console, an Indicator has properties that may be modified using the Properties option. The Indicator Properties window is similar to the Properties window for an element. The middle portion of the window for selecting agents does not apply, nor do the color slide bars in the lower portion of the window. The scrollable top portion of the window displays the Name, Label, and Data fields.

If an Indicator is associated with a request that is loaded from an ASCII file and the request's Restart field is set to True, the Indicator is displayed in the view from which the request is launched. This is the case even if you have moved the Indicator into a different view before saving the ASCII file.

4.1.3 Strip Charts

A Strip Chart contains an auto-scaled strip chart of values for a single attribute reported from a particular element. See Figure 4-2.

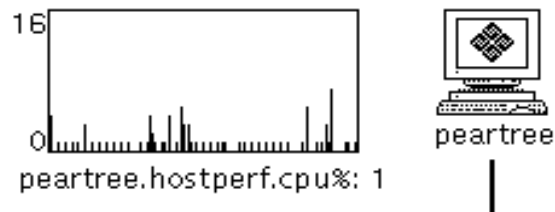


Figure 4-2 Strip Chart

The SunNet Manager Console displays up to 128 Strip Charts in the view where the data request selection process is begun. Strip Charts are positioned in the view using the same positioning algorithm applied to newly created elements.

Strip Charts may be edited (cut, copied, pasted) in the same manner as other elements. For example, all Strip Charts can be collected in a separate view.

If a Strip Chart is associated with a request that is loaded from an ASCII file and the request's Restart field is set to True, the Strip Chart is displayed in the view from which the request is launched. This is the case even if you have moved the Strip Chart into a different view before saving the ASCII file.

As with any other SunNet Manager Console element, the Strip Chart has properties that can be modified. Click MENU on the appropriate strip chart, and click SELECT on the Properties option. The window shown in Figure 4-3 will be displayed.

The screenshot shows a window titled "SunNet Manager Console: _chart.3643.1070 (view._graph)". The window contains the following fields and controls:

- Name:** _chart.3643.1070
- Label:** peartree.hostperf.cpu%
- Data:** 0
- Width:** 128 (with up/down arrow buttons)
- Height:** 64 (with up/down arrow buttons)
- Fixed Range:** ☐
- Max Y Value:** 0 (with up/down arrow buttons)
- A list of monitoring options, each with a checkbox and a description:
 - ☐ diskinfo: disk space information
 - ☐ etherif: ethernet interface statistics
 - ☐ etherif2: ethernet interface statistics (Solaris
 - ☐ hostif: interface statistics
 - ☐ hostmem: memory utilization
 - ☐ hostmem2: memory utilization (Solaris 2.x)
 - ☐ hostperf: Host performance data
 - ☐ iostat: I/O statistics (SunOS 4.x)
 - ☐ iostat2: I/O statistics (Solaris 2.x)
 - ☐ ippath: IP path information
- Color selection fields:
 - Red:** 0 (with a color bar)
 - Green:** 0 (with a color bar)
 - Blue:** 0 (with a color bar)
- Buttons at the bottom: Apply, Reset, Alias...
- A "Browse" button in the bottom right corner.

Figure 4-3 Strip Chart Properties

Note that the Strip Chart Properties window is similar to the Properties window for an element. However, the middle portion of the window for selecting agents does not apply and the strip chart color cannot be changed. The scrollable top portion of the window displays the following:

Name:

is an internal element identifier and may not be changed.

Label:

sets the label displayed underneath the chart.

Data:

is the latest attribute value displayed in the chart and may not be changed.

Width:

is the displayed width of the chart in pixels.

Height:

is the displayed height of the chart in pixels.

Fixed Range:

determines whether the chart is auto-ranging (no check mark in box) or fixed-range (check mark in box).

Max Y Value:

is the maximum value to use in a fixed range chart.

Min Y Value:

is the minimum value to use in a fixed range chart.

4.1.4 Graph Tool

The Graph Tool, also known as the Results Grapher, is an application that allows the display of real-time data. See Chapter 7, “Results Grapher,” for more information about displaying information with the Grapher.

4.2 Event Reports

The Event/Trap Reports log displays reported values for selected attributes, as well as asynchronous (trap) reports. You display the Event/Trap Reports log by selecting the SNM Console View ► Event/Trap Reports menu item.

4.2.1 Event/Trap Reports Log

The Event/Trap Reports log contains event and trap reports, in ASCII, for all elements managed by the SunNet Manager Console. Event/trap reports entries are time-stamped and show attribute values for the entire agent group. The attribute and value that caused the event are flagged with the relational operator and threshold value shown to the right in parentheses. See Figure 4-4.

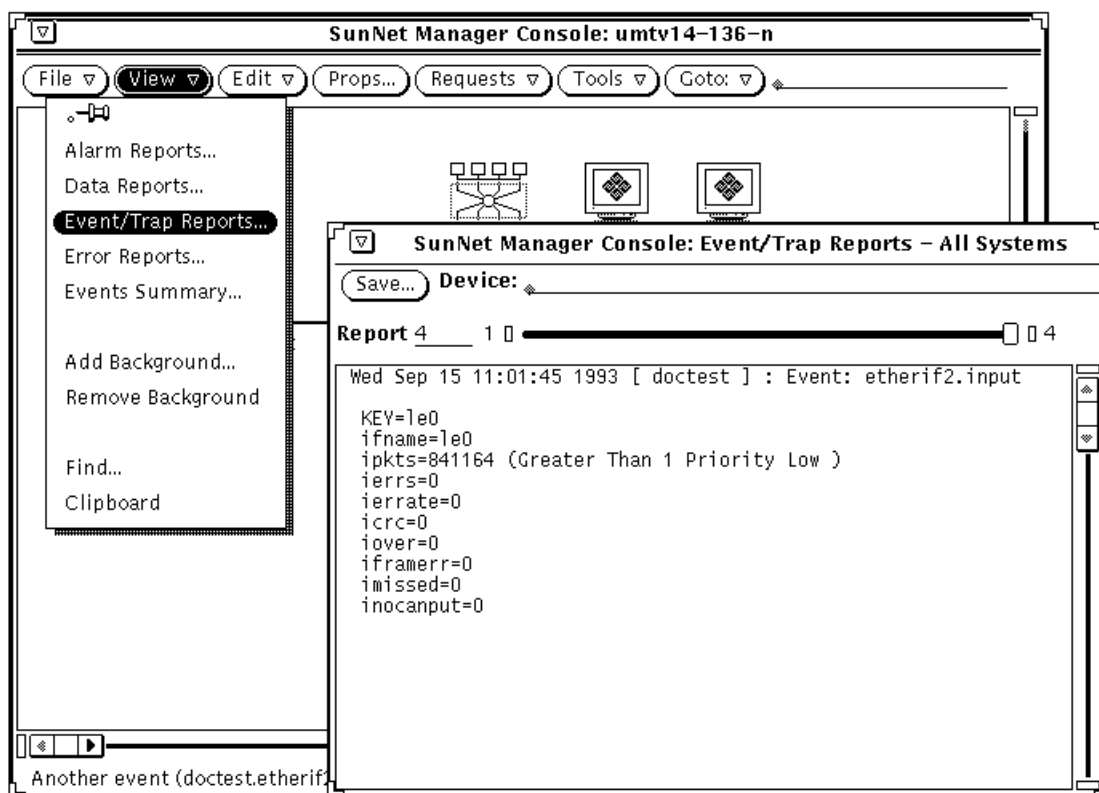


Figure 4-4 View — Event/Trap Reports

To view event/trap reports, drag the slider to the right or the left. To look at all Event/Trap log entries for all elements, leave the Device field blank. To examine event/trap reports for a particular element instance, type in the

element name after the Device prompt and press Return. If you wish to look at Event/Trap log entries for all the elements again, press ctrl-u to clear the Device field and press Return.

The Event/Trap log contains the most recently received event reports and trap reports for all systems. The currently displayed report is indicated on the Report line, while the total number of entries is displayed at the right of the slider. If you specify an element name in the Device field, the number of entries reflects the total number of entries for that element.

Event reports and trap reports are each allocated a maximum number of entries in the Event/Trap log. By default, the maximum number of reports for either events or traps is 1000. When the allocation limit is reached for either event reports or trap reports, the oldest event report or trap report is deleted. You can change the maximum number of entries by modifying the Maximum Event Reports or Maximum Trap Reports settings in the Console Properties Miscellaneous category (refer to Section 9.8, "Miscellaneous," on page 9-22 for more information).

Save writes the contents of the Event/Trap Reports log to disk. Selecting this button displays a pop-up window in which you can specify the file name and path of the disk file. You can use Save after specifying a Device name to save a subset of the Event/Trap Reports log to disk.

Use the scroll elevator to scroll through the entry that spans multiple panes of the Event/Trap log window. Any number of Event/Trap log windows can be displayed simultaneously, allowing you to view log entries from individual systems side-by-side.

4.2.2 Errors Log

The Errors log contains a list of machines, in ASCII, that have had at least one error occur on them, for all elements managed by the SunNet Manager Console. You display the Error Reports log by selecting the SNM Console View►Error Reports menu item. You are then able to use the View►Event/Trap Reports menu item to specify a specific device name in the Event/Trap Reports window to see the event(s) that have occurred on the machine(s) of your choice.

4.2.3 Events Summary

The Events Summary log contains a list of machines, in ASCII, that have had at least one event occur on them, for all elements managed by the SunNet Manager Console. You display the Events Summary log by selecting the SNM Console View►Events Summary menu item. You are then able to use the View►Event/Trap Reports menu item to specify a specific device name in the Event/Trap Reports window to see the event(s) that have occurred on the machine(s) of your choice. If you press the MENU button over the Events Summary, the Drop All popup menu appears, as shown in Figure 4-5. Selecting Drop All clears all outstanding events.

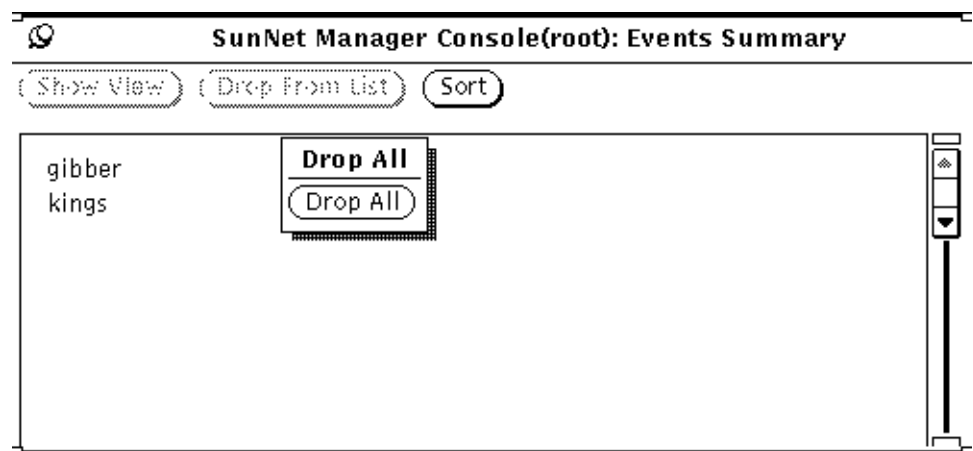


Figure 4-5 Drop All Option Popup Menu

4.3 Viewing Alarm Reports

There are four types of reports returned by SunNet Manager: data, event, error, and trap. Of these four, the Console groups the last three under the heading of “Alarms.” SNM allows you to view a summary of alarm reports for all elements within a given view. To do this, with no glyphs selected in the Console, press MENU on the View button in the Console window and release MENU over Alarms. You receive a window such as the following:

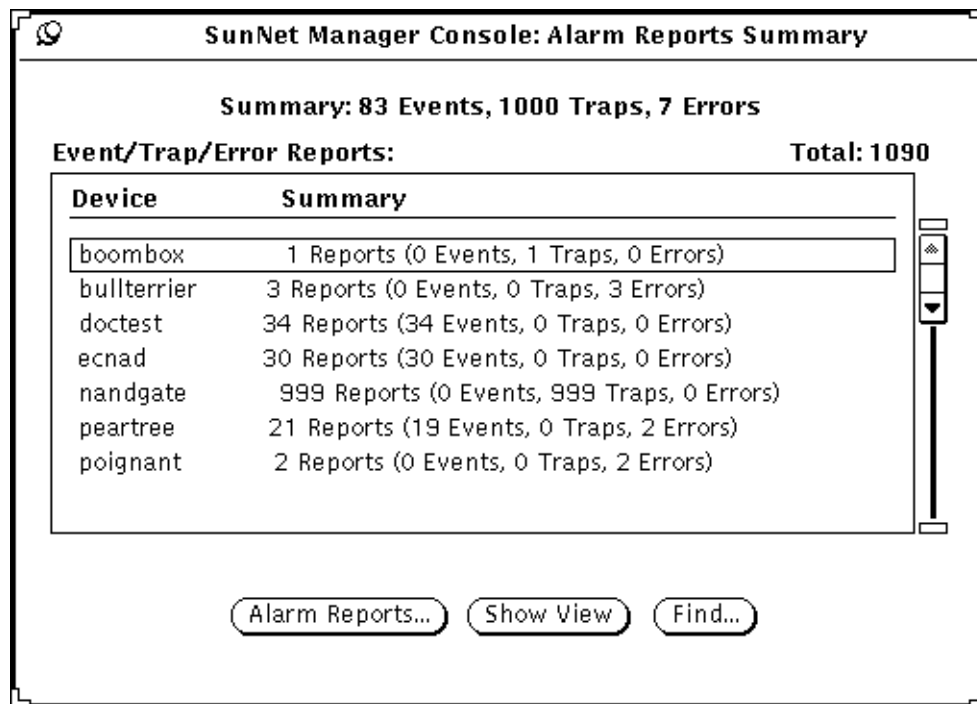


Figure 4-6 Alarm Reports Summary Window

The Alarm Reports Summary window is updated every 30 seconds. You might find it useful to leave the window pinned up while the Console is running. The Alarms Reports Summary displays reports only for the current view. If there are no reports pertaining to the current view, the Alarms Reports Summary window is blank. Upon switching to another view, the Alarms Reports Summary window is updated with the switched-to view's alarm reports summary, if such reports exist.

Note – If both the SNM Console Alarm Report Summary window and a device specific Alarm Summary window are being displayed simultaneously and reports continue to arrive for the device, the number of reports received totals for the two windows will differ until the SNM Console Alarm Reports Summary window is next updated (every 30 seconds.)

The Alarm Reports Summary window has three buttons at the bottom, described as follows:

Alarm Reports

Allows you to view a device-specific alarm report for a device selected in the Alarm Reports Summary window's scrolling list.

Show View

After having selected a specific device from the scrolling list, clicking SELECT on the Show View button selects the specified device and brings it into view in the SNM Console window and tells you whether the device resides under one or multiple views. If a device resides under multiple views, clicking the Next button allows you to switch to the next view under which this device resides. Refer to Section 4.3.2, "Switching Views," on page 4-13 for more information.

Find

Allows you to find an element entry not visible in the Alarm Reports Summary window's scrolling list.

The following subsections describe functions available through the Alarm Reports feature, including (but restricted to) the functions available through the buttons described above.

4.3.1 Finding a Device

To find a device that is not visible in the window's scrolling list, you can scroll through the list or, more conveniently, click SELECT on the Find button at the bottom of the window. After clicking on Find, you receive the window shown in Figure 4-7.

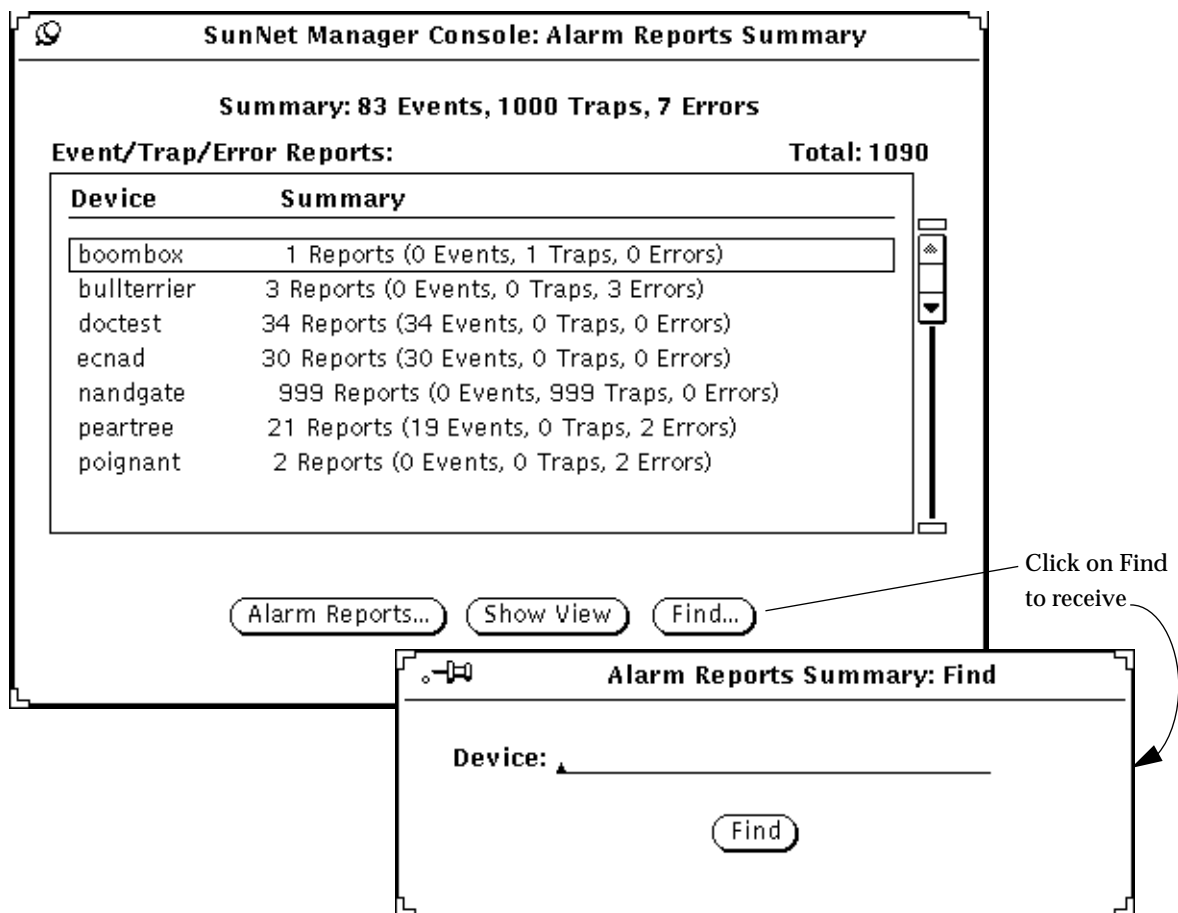


Figure 4-7 Alarm Reports Find Window

When you click SELECT on Find in the Find window, if a line for a device is present, that line is displayed and highlighted in the Alarm Reports Summary window's scrolling list. If a line for a device is not present, you receive the message "Device name not found" in the Alarm Reports Summary Find pop-up window.

4.3.2 Switching Views

Through the Alarm Reports Summary window, SNM allows you to switch views for a specific device. (This function is also available through the View►Find option in the Console window.) To show the current view for a device and to be able to switch views, first select the desired device in the Alarm Reports Summary window's scrolling list, then click SELECT on the Show View button. You receive a window such as the one shown in Figure 4-8.

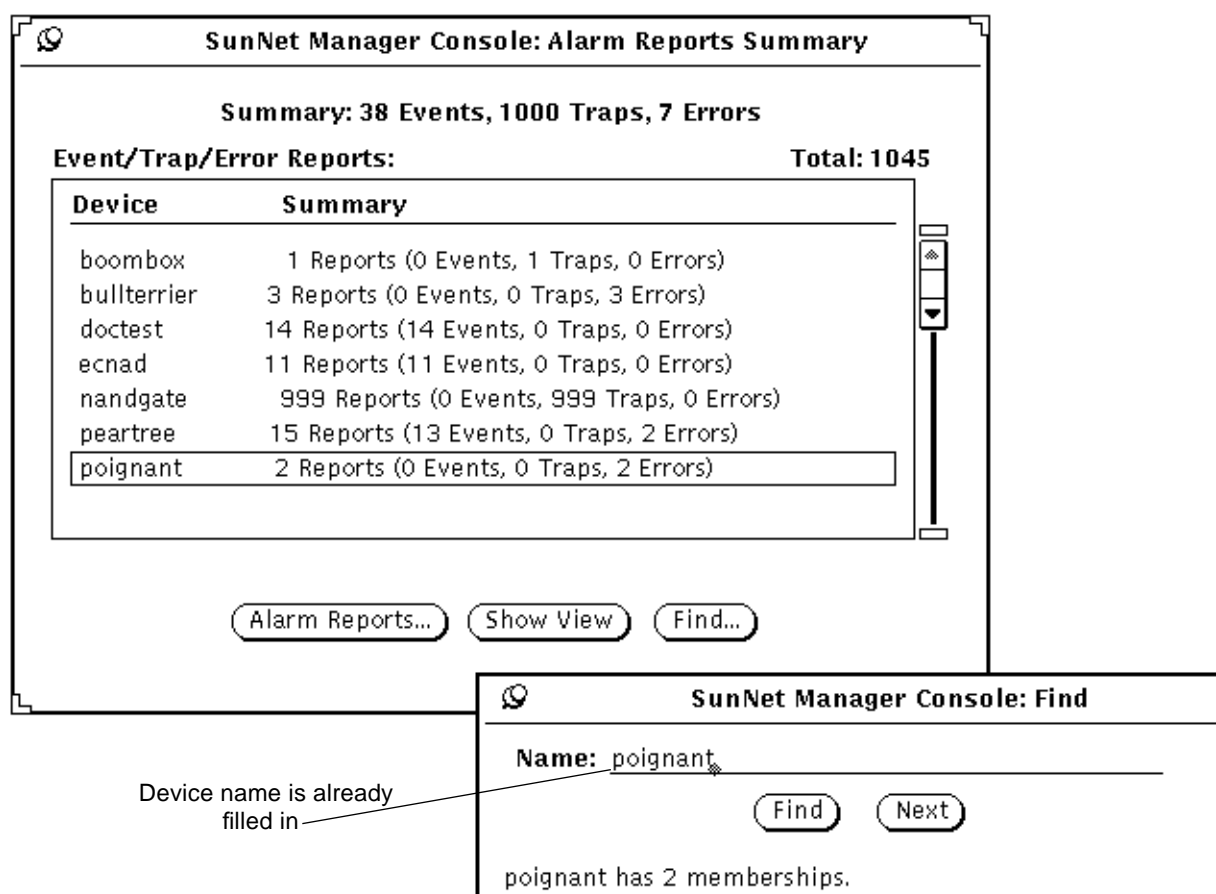


Figure 4-8 Alarm Reports: Show View Window

If a device is in only one view, the Next button in the window above is greyed-out. If a device is in multiple views, click SELECT on Next to switch from the current view (as displayed in the Console window) to the next view in which the specified device appears. In this context, “next” means the next view on the list of views as displayed by the Console’s Goto menu.

4.3.3 *Obtaining Device-specific Alarm Reports*

SNM gives you the ability to view further detail for a specific device. To obtain a device-specific report, you can use any of the following methods:

- Double click on the line for a device in the scrolling list in the Alarm Reports Summary window.
- Click SELECT on the line for a device in the scrolling list in the Alarm Reports window, then click SELECT on the Alarm Reports button at the bottom of the window.
- Click SELECT on the glyph for a device and invoke View►Alarm Reports in the Console menu
- Invoke Alarm Reports from an element’s glyph menu.

Using any of the preceding methods, you then receive an Alarm Reports window for the specified device, as shown in Figure 4-9.

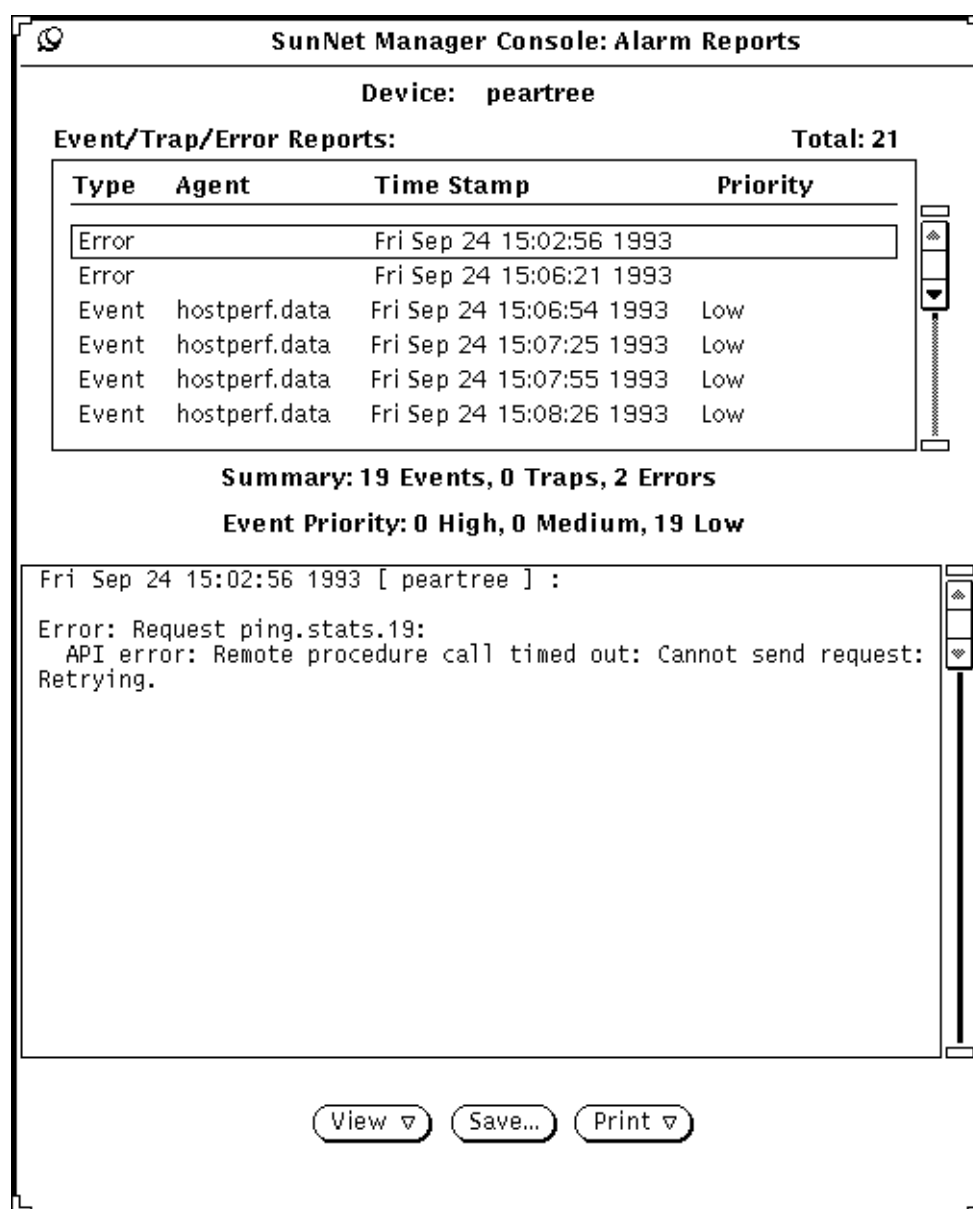


Figure 4-9 Device-specific Alarm Reports Window

4.3.4 Features of Device-specific Alarm Reports Window

A device-specific Alarm Reports window, such as the one shown in Figure 4-9, allows you to:

- sort alarm reports
- filter reports
- find reports associated with a specific agent
- save reports to a file (which can be viewed using the Results Browser)
- print reports

These functions are available through the View, Save, and Print buttons at the bottom of the Alarm Reports window.

4.3.4.1 Filtering Alarm Reports

Press MENU in the View button, to receive the following menu;

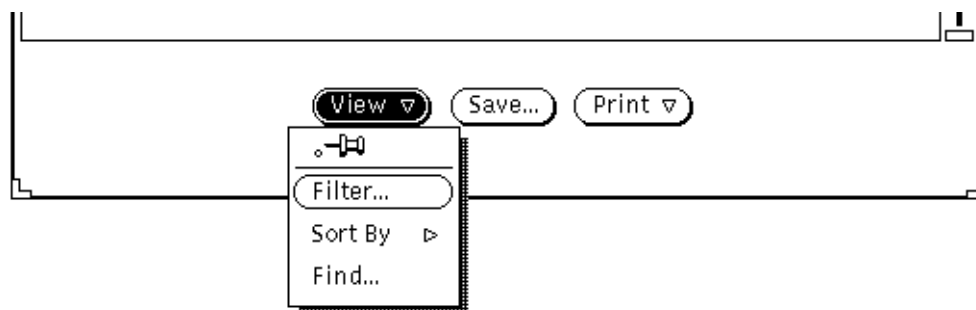


Figure 4-10 Device-specific Alarm Reports View Menu

Release MENU over Filter to receive the following window:

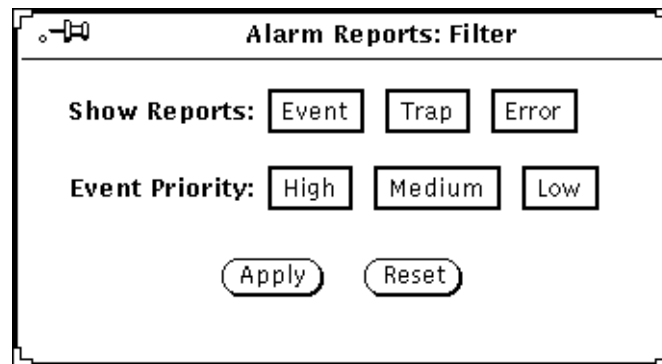


Figure 4-11 Device-specific Alarm Reports Filter Window

Your selections in the Filter window determine which types of reports are displayed in the device-specific Alarm Reports window. The default is that all alarm reports are displayed. Note that priority choices apply only to Event Reports. If you click on Apply, your choices take effect immediately. Click on Reset to restore the choices to the way they were when you brought up the Filter window.

Note – If you plan to both sort (described below) and filter your device-specific alarm reports, you must sort before filtering. If you want to sort again, after filtering, you must click SELECT on Reset, then Apply, in the Filter window, sort again, as described below, then reinvoke Filter.

4.3.4.2 *Sorting Alarm Reports*

In a device-specific Alarm Reports window, press MENU on View►Sort By. You receive the following menu:

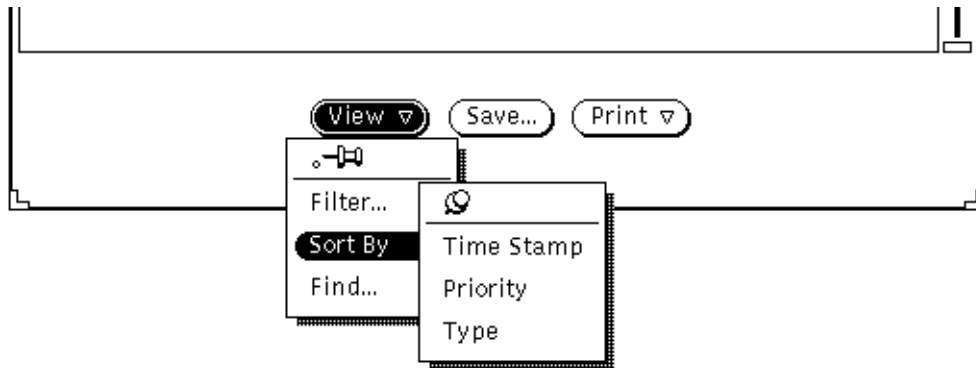


Figure 4-12 Device-specific Alarm Reports Sort Menu

These choices are mutually exclusive. Release MENU over the sorting method you want. Your selection takes effect immediately.

The sorting choices are explained as follows:

Time Stamp

The default choice. Reports are sorted in chronological order with the most recent being last on the list.

Priority

Event reports are sorted according to priority—high, medium, or low—with high priority reports listed first on the list. This option makes most sense if you are filtering out trap and error reports. If you are not filtering out reports and specify sorting by priority, event reports come first, followed by error reports, then trap reports.

Type

Reports are sorted by type, in the order: event, error, trap. Note that alarm reports that are received after you specify sorting by type are appended to the appropriate list of reports, depending on the new reports' type.

Note – When sorting alarm reports, only the reports currently displayed in the Alarm Reports window will be sorted. All new reports received during the sorting operation are appended to the end of the sorted list.

4.3.4.3 Finding Agent-specific Alarm Reports

SNM allows you to search on an agent name to find instance(s) of event reports originating with that agent. To do this, in the device-specific Alarm Reports window, press MENU on View►Find and release MENU to receive the window shown in Figure 4-13. (The View menu is shown in Figure 4-10 on page 4-16.)

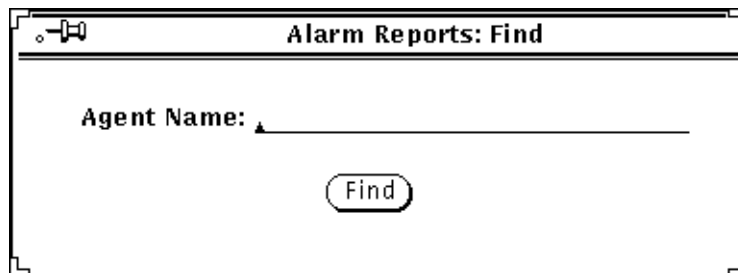


Figure 4-13 Device-specific Alarm Reports Find Window

In the Agent Name field, enter the agent *and* group name—not just the agent name. For example, specify `hostperf.data`, not just `hostperf`. When you click SELECT on Find, the window displays the contents of the first (oldest) report originating with the agent you specified. If there is no report originating with the agent you specify, you receive the message, “Agent name not found.”

4.3.4.4 Saving Device-specific Alarm Reports

SNM allows you to save device-specific alarm reports to a file. To do this, in the device-specific Alarm Reports window, click SELECT on Save. You receive a window such as the one in Figure 4-14.

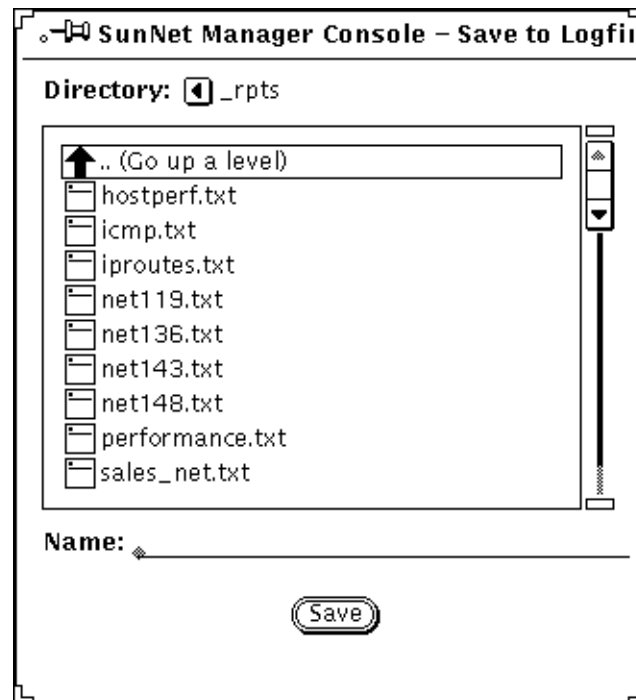


Figure 4-14 Save to Logfile Window

If the directory that is first displayed is not the directory you want, change directories by clicking SELECT on folder icons or the .. (level above) symbol. Alternatively, you can enter a pathname in the Name field and click SELECT on Save. If the path you enter ends in a valid directory, you are switched to that directory.

When you reach the directory you want, enter the name of the file to which the report will be written.

4.3.4.5 Printing Device-specific Alarm Reports

SNM allows you to print device-specific alarm reports. Click SELECT on the Print button on the bottom of the device-specific Alarm Reports window, or press MENU on Print►Report and release MENU. SNM sends the currently displayed alarm report to your default printer.

Discover Tool



<i>Invoking Discover</i>	<i>page 5-2</i>
<i>Discover Tool Configuration</i>	<i>page 5-5</i>
<i>snm_discover Command</i>	<i>page 5-18</i>
<i>The discover.conf File</i>	<i>page 5-22</i>

The discover function has a special mode that is used only upon startup, when you invoke the HeadStart option. In this mode the search for network elements is limited to no more than 10 elements on the local subnetwork.

The Discover Tool has two functions:

- A discover function, which finds hosts, routers, networks, and Simple Network Management Protocol (SNMP) devices reachable from the Console machine. On finding a network element, the discover function stores a record for that element in the runtime database.
- A monitor function, in the which the tool compares the elements stored in the runtime database with the elements it finds at a specified interval or specified time. If new elements are detected, the monitor function stores the elements in a holding-area view and records these elements in a log file. Through the same log file, the monitor function can also notify you if a previously discovered host is down or was down within a given period.

The hosts found by the monitor function are hosts added to a network since the last running of the discover function and hosts that the discover function did not find.

The Discover Tool offers a number of configuration options for both the discover and monitor functions. These options allow you to fine-tune the extent and depth of the tool's search and monitoring activities. The configuration options are described later in this chapter.

Note – When the Discover Tool locates SNMP devices, it cannot determine whether these devices support functionality specific to SNMPv2.

5.1 *Invoking Discover*

You invoke Discover using one of the following methods:

- from the Console window, when you press MENU on Tools►Discover and release MENU;
- from the SunOS command line, if you have the path to the SNM executables in your PATH variable, invoke `snm_discover`; for example, to run Discover and invoke the user interface, enter:

```
hostname# snm_discover -T
```

The path to `snm_discover` is:

- `/usr/snm/bin` on SunOS 4.x
- `/opt/SUNWconn/snm/bin` on Solaris 2.x

After Discover is invoked from the Console window or from the command line using the `-T` option, the window shown in Figure 5-1 is displayed. (You can start Discover directly, bypassing this window, if you use the `snm_discover` command without the `-T` option.)

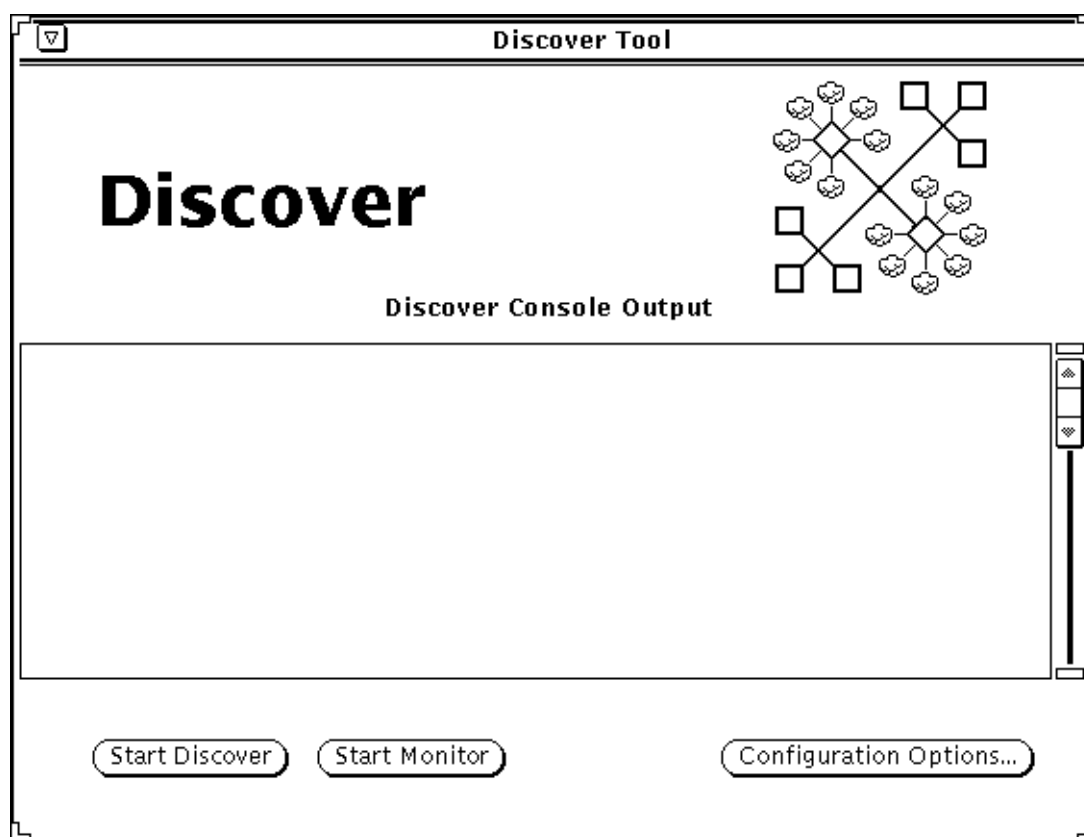


Figure 5-1 Discover Tool Base Window

The buttons in the Discover Tool base window are described as follows:

Start/Stop Discover

This button is a toggle. When Discover is first invoked, Start Discover is displayed. If Start Discover is selected, the discover function is invoked in a mode that is determined by the parameters in the Discover Configuration window (discussed below). This mode of the discover function can be limited or as extended as your time and machine resources allow. Once Start Discover is selected, the button display changes to read “Stop Discover”. Selecting Stop Discover will cause discovery operations to cease.

Start/Stop Monitor

A toggle for which the initial display is Start Monitor. This button invokes the monitor function, using the parameter values as they are set in the Monitor Properties window, described below.

Configuration Options

Invokes the Discover Configuration window, in which you can switch between discover and monitor function properties. These properties are discussed in the following subsections.

Once Discover is started, Discover searches for network devices in a two-stage process:

- Discover first builds a hierarchical linked-list structure that mirrors the network topology that it uncovers. For each Class A, B, or C network that is discovered, an entry is added to the list. Each of these entries contains a list of subnetworks (or a list of hosts directly connected to this network if the network is non-subnetted) and a list of connected networks.
- Once the network structure has been uncovered, Discover then issues Internet Control Message Protocol (ICMP) echo requests to find hosts on each of the subnetworks in the linked-list structure. By default, Discover pings each host address in the range 0 to 255 on each subnetwork. (The command-line version of Discover has a `-r` option that can be used to pass Discover a different range of host addresses to ping. This is discussed below in Section 5.3.2, “Discovery on Non-Subnetted Class B Networks.”)

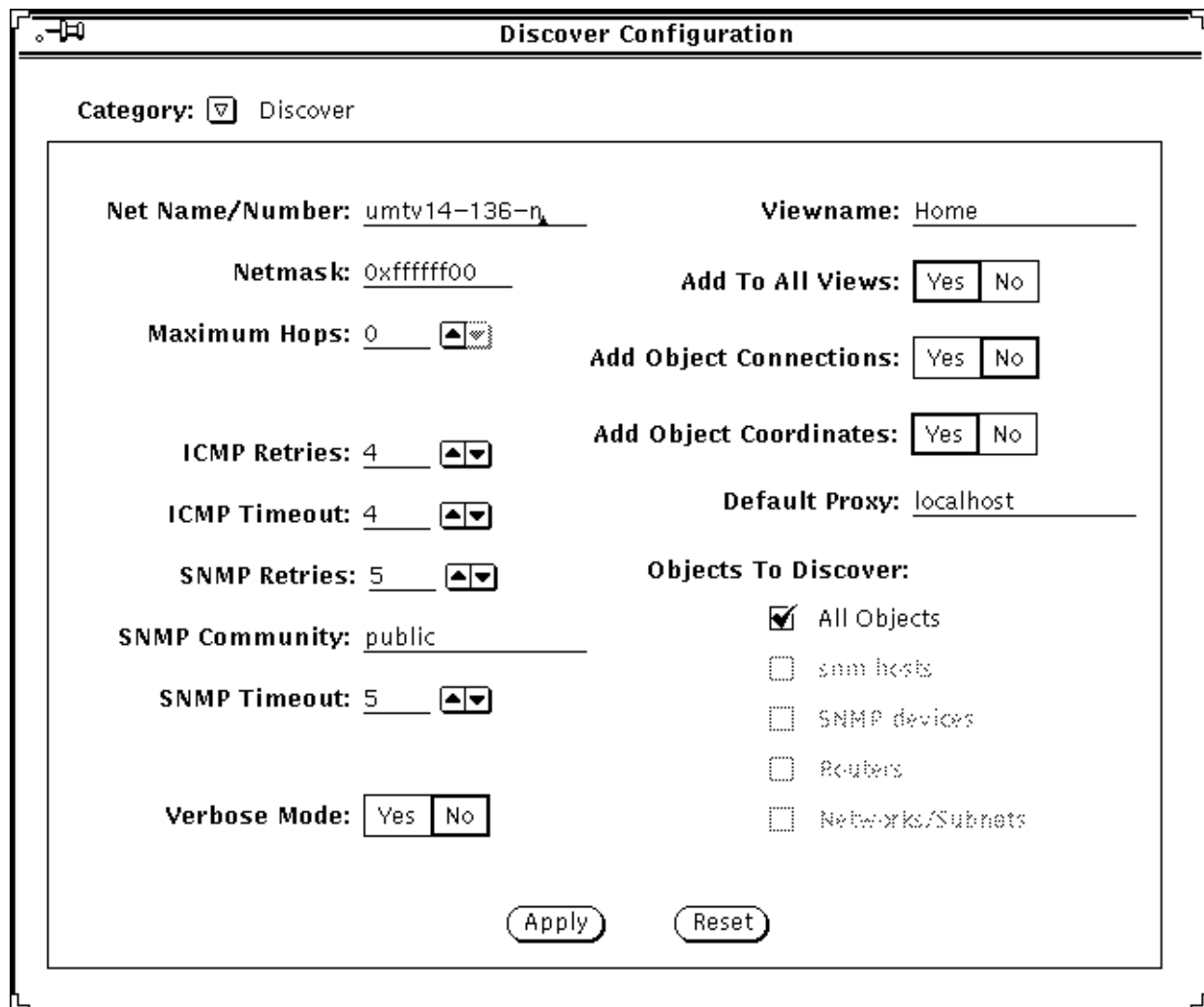
To build the hierarchical network topology structure, Discover proceeds through the following steps:

- The local routing table is accessed to find the location of the subnetwork’s default router.
- The router’s routing table is then retrieved using SNMP.
- Routing tables are then retrieved from all next hop gateways.
- Discover then leapfrogs from gateway to gateway as it retrieves the routing table from each “next hop” gateway.
- Discover performs a traceroutes operation to determine the number of hops to each “next hop” gateway. Each router that a packet must traverse to reach that gateway counts as an additional “hop.” The traceroutes operations may also reveal gateways that were not uncovered by previous queries.

- Discover continues to retrieve routing tables of “next hop” gateways until the “Maximum Hops” count set in the Discover Properties sheet is reached. By default, “Maximum Hops” is set to zero and the discovery process is limited to the local subnetwork.
- SNMP is used to obtain the routing tables of any additional gateways that were uncovered by tracing the route to gateways previously uncovered.

5.2 *Discover Tool Configuration*

When you click SELECT on the Configuration Options button in the Discover Tool base window, you receive the Discover Configuration window as shown in Figure 5-2.



The image shows a software window titled "Discover Configuration". At the top, there is a "Category:" dropdown menu with a small square icon to its left, currently set to "Discover". Below this, the window is divided into two main sections. The left section contains several configuration fields: "Net Name/Number:" with the value "umtv14-136-n", "Viewname:" with the value "Home", "Netmask:" with the value "0xffffffff", "Maximum Hops:" with a value of "0" and a small up/down arrow icon, "ICMP Retries:" with a value of "4" and a small up/down arrow icon, "ICMP Timeout:" with a value of "4" and a small up/down arrow icon, "SNMP Retries:" with a value of "5" and a small up/down arrow icon, "SNMP Community:" with the value "public", "SNMP Timeout:" with a value of "5" and a small up/down arrow icon, and "Verbose Mode:" with "Yes" and "No" radio buttons. The right section contains three pairs of "Yes/No" radio buttons for "Add To All Views:", "Add Object Connections:", and "Add Object Coordinates:". Below these is a "Default Proxy:" field with the value "localhost". At the bottom right, there is a section titled "Objects To Discover:" with a list of checkboxes: "All Objects" (checked), "snm hosts", "SNMP devices", "Routers", and "Networks/Subnets". At the very bottom of the window, there are two buttons: "Apply" and "Reset".

Discover Configuration

Category: Discover

Net Name/Number: umtv14-136-n Viewname: Home

Netmask: 0xffffffff Add To All Views:

Maximum Hops: 0 Add Object Connections:

ICMP Retries: 4 Add Object Coordinates:

ICMP Timeout: 4 Default Proxy: localhost

SNMP Retries: 5 Objects To Discover:

SNMP Community: public ☒ All Objects

SNMP Timeout: 5 ☐ snm hosts

Verbose Mode: ☐ SNMP devices

☐ Routers

☐ Networks/Subnets

Figure 5-2 Discover Configuration Window: Discover Properties

In this window, press MENU in the Category menu button to view the two categories of properties, Discover and Monitor, as shown in Figure 5-3.

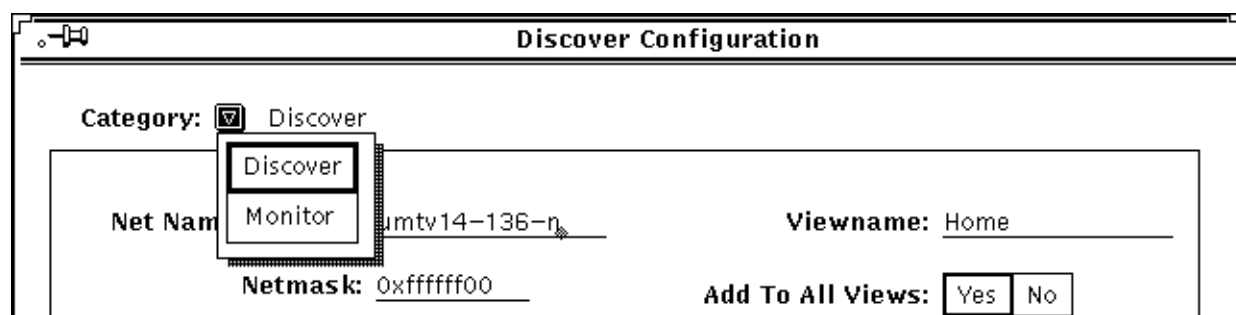


Figure 5-3 Discover Configuration Category Menu

Release MENU over the category of your choice. The parameters for these categories are discussed in the following subsections.

When a function, either discover or monitor, is active, the configuration window for that function is greyed-out. You must stop the function before changing any properties.

5.2.1 Discover Function Configuration

The window for discover function configuration is the window you receive when you click SELECT on the Configuration Options button in the Discover Tool base window. (See Figure 5-2.) The parameters in this window determine the operation of the discover function of the Discover Tool. The ICMP- and SNMP-related discover parameters also apply to the monitor function.

The parameters in the Discover Properties window are described as follows:

Net Name/Number

The name or subnetwork number—as specified in an NIS map, an NIS+ table, or the `/etc/networks` file—that identifies the subnetwork that will be the starting point for the discover function. By default, the discover uses the network number used by the local machine. When specifying a network number, be sure to enter a value for each byte within the number. For example, specify `129.144.41.0`, not `129.144.41`.

If you enter a network name or number outside of the local subnetwork, the discover function limits its search to the subnetwork you specify.

Netmask

The network mask—as specified in an NIS map, an NIS+ table, or the `/etc/netmasks` file—used in the network(s) on which the discover function is to operate. By default, the discover function uses the netmask number used on the local subnetwork. Enter the netmask number in hexadecimal (preceded by `0x`) or dotted decimal notation (for example, `255.255.255.0`).

For limiting the number of elements found, the discover function uses the inverse of the host portion of the subnet mask. For example, if you have a netmask of `0xffffffff00`, the host portion of the netmask, `0x00`, becomes `0xff`, or decimal 255.

The netmask parameter has two uses: one is for situations in which you want discovery to begin in a subnetwork other than the local subnetwork. In this case, the discover functions needs to know the netmask used on that subnetwork. The second use is as a limit to the number of hosts the discover function will add to the runtime database. While the latter use can be helpful in certain networks, most users need not be concerned with setting limits. The discover function will ping a maximum of 2048 network elements per subnetwork.

Maximum Hops

A measure of how far the discover function will extend its search. For a given route from the machine running the Discover Tool (or from the subnetwork specified in the Net Name/Number parameter), this parameter is the maximum number of routers the ICMP packets sent by the discover function will traverse. The default for this parameter is zero hops, which means searches are restricted to the local subnetwork. By setting Maximum Hops to 1 or greater, Discover will extend its search to additional subnets connected to the local subnetwork's router.

Note – If there are multiple routers attached to the local subnetwork, the number of subnets searched may increase exponentially if Maximum Hops is set higher than zero. Depending on the size of your network, setting Maximum Hops to any value above zero can result in very long searches. For example, on a moderately burdened machine, with Maximum Hops set to 0 (the default) and a local subnetwork with 200 hosts, the discover function would take about 20 minutes to complete.

ICMP Retries

The discover function uses the Internet Control Message Protocol (ICMP) to find network elements that are not SNMP devices. This parameter determines the number of times the tool will retry “finding” an element using ICMP following an initial non-response. If your network is very busy, you might want to increase this value.

ICMP Timeout

Determines how long (in seconds) the discover function waits after sending an ICMP packet that did not receive a response, before sending another ICMP packet. To speed up the discovery process, reduce this value.

SNMP Retries

For SNMP devices, the discover function uses SNMP for discovery purposes. This parameter determines the number of times the tool will retry contacting an element using SNMP following an initial response. This parameter comes into play primarily when querying routers. ARP and routing tables are often large and sometimes require multiple retries to obtain complete tables.

SNMP Community

The discover function limits its search for SNMP devices to devices that have the community name(s) you specify here. You can specify up to five SNMP community names. Use a colon to delimit multiple names. Note that using multiple community names makes the discover function take a longer time than it would with a single name. This is because the discover function tries each device with each community name until a response is received.

SNMP Timeout

Determines how long (in seconds) the discover function waits for a response to an SNMP packet before giving up or, for routers, sending another SNMP packet.

Verbose Mode

A yes/no switch for which the default is yes. Click SELECT on Yes to have the discover function report detailed progress in the Discover Console Output window in the Discover base window. A no selection means that the discover function reports only intermittent progress.

We recommend a yes setting for this parameter, particularly for long searches.

Viewname

Name of the view to which the discover function will add discovered network elements. If Add To All Views (see below) is set to no, the discover function preserves a separate view hierarchy under this name. If the name you specify here is not in the runtime database, the discover function creates a new view and adds discovered elements to it. The name specified here will be appended to all subviews in this view hierarchy.

The SNM database does not permit duplicate names of views (or other elements). By default, Discover uses network and subnetwork names in building the database representation of the network topology. If you want to have different views into the same subnetwork or network — such as a view of subnetwork A that shows all hosts and a view of subnetwork A that shows only routers — then the names specified for these views must be distinct. By appending Viewname to the names of the subviews in a view hierarchy, the separateness of the subviews in that view hierarchy is preserved. Discover will append Viewname to the names of subviews only if Add To All Views is set to “No.”

Add To All Views

A yes/no switch for which the default is yes. If you accept the default of Yes, the discover function adds discovered elements to all views in the runtime database, including the view specified in Viewname, if one is present. If you click SELECT on No, the discover function appends the name specified in the Viewname field to all views (that is, elements of category view) within the specified view. For an element that is in multiple, high-level views, this feature allows you to distinguish that element as it exists in different views, such as a router that is in a routers-only view of a subnetwork but also in a view that shows all devices on that subnetwork. Since the SNM database does not allow views (or other elements) with duplicate names, you can only have multiple views into the same subnetwork if you assign separate names to maintain the uniqueness of each view. The use of this parameter to create separate view hierarchies is illustrated in Section 5.2.2, “Example: Using Discover to Create a Routers-only View.”

Manageable connections—with a manageable object at each end—are distinct from the simple connections that you can draw between elements using the Connect option in an element’s glyph menu.

Add Object Connections

A yes/no switch for which the default is no. Click SELECT on Yes to have the discover function create manageable connections between discovered elements and to add the connection information to the `linkmap` file. However, the discover function only adds connection information to the

linkmap file if Maximum Hops has been set to 2 or more. Refer to the linkmap (5) manual page for information about the format of the linkmap file and its usage. A No selection means that the discover function does not make connections.

Add Object Coordinates

A yes/no switch for which the default is yes. By default, the discover function performs minimal layout of elements within a view. For example, in a subview representing an Ethernet subnetwork, glyphs representing hosts are distributed on both sides of the bus glyph. If you select no, elements are created with no layout of elements taking place. A sample subview created without coordinates is shown in Figure 5-4.

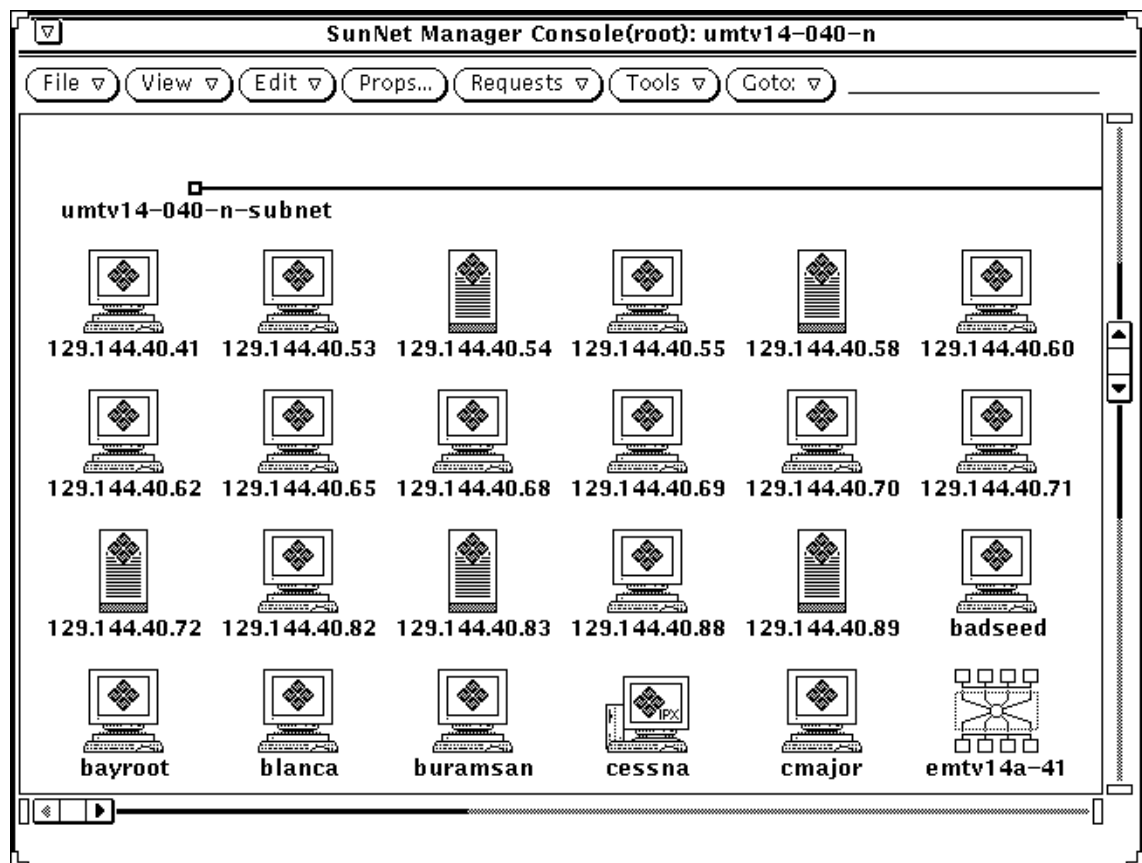


Figure 5-4 Subview Created without Coordinates

Default Proxy:

When the discover function finds an element that has SNM proxy agents, it specifies as the proxy host the host you enter here. If you leave this item blank, the proxy host is specified as `localhost`.

Objects to Discover

By default, the All Objects item is selected. If you click SELECT on the All Objects checkbox to deselect it, the following greyed-out items display normally:

snm hosts

Systems running SunNet Manager agents.

SNMP devices

Systems or other devices running SNMP. Note that the discover function cannot discover whether the SNMP device supports the added functionality of SNMPv2.

Routers

Systems that have multiple network interfaces to different IP networks. When the discover function finds a router, it fills in IP addresses for the first two interfaces and uses the alias feature to give a name to each interface (if, indeed, each interface has a unique name), including those beyond the first two. You can then specify any of these aliases when sending a data or event request.

Networks/Subnets

Collections of subnetworks or individual subnetworks. If you select this item, the discover function finds only networks and subnetworks, plus manageable connections and routers if the Add Object Connections item is selected.

Accept the default of All Objects or click SELECT on the checkbox(es) for object(s) of your choice.

5.2.2 Example: Using Discover to Create a Routers-only View

You might want to use Discover to create a separate view hierarchy — named “Routers” — that contains only routers in addition to another view hierarchy that contains all objects. To create a routers-only view hierarchy, you would need to make the following entries (as shown in Figure 5-5) in the Discover Configuration window:

- Specify “Routers” in the Viewname field
- Set Add To All Views to No
- Select Routers only in the Objects To Discover field.

Category: ☒ Discover

Net Name/Number:

Viewname:

Netmask:

Add To All Views:

Maximum Hops:

Add Object Connections:

ICMP Retries:

Add Object Coordinates:

ICMP Timeout:

Default Proxy:

SNMP Retries:

Objects To Discover:

SNMP Community:

☐ All Objects

SNMP Timeout:

☐ snm hosts

Verbose Mode:

☐ SNMP devices

☒ Routers

☐ Networks/Subnets

Figure 5-5 Example of Routers-Only Discover Configuration

A glyph named “Routers” will be added to the Home view. If there are multiple subviews under “Routers” that represent subnetworks, the names of these views will have “-Routers” appended to them, as shown in Figure 5-6.

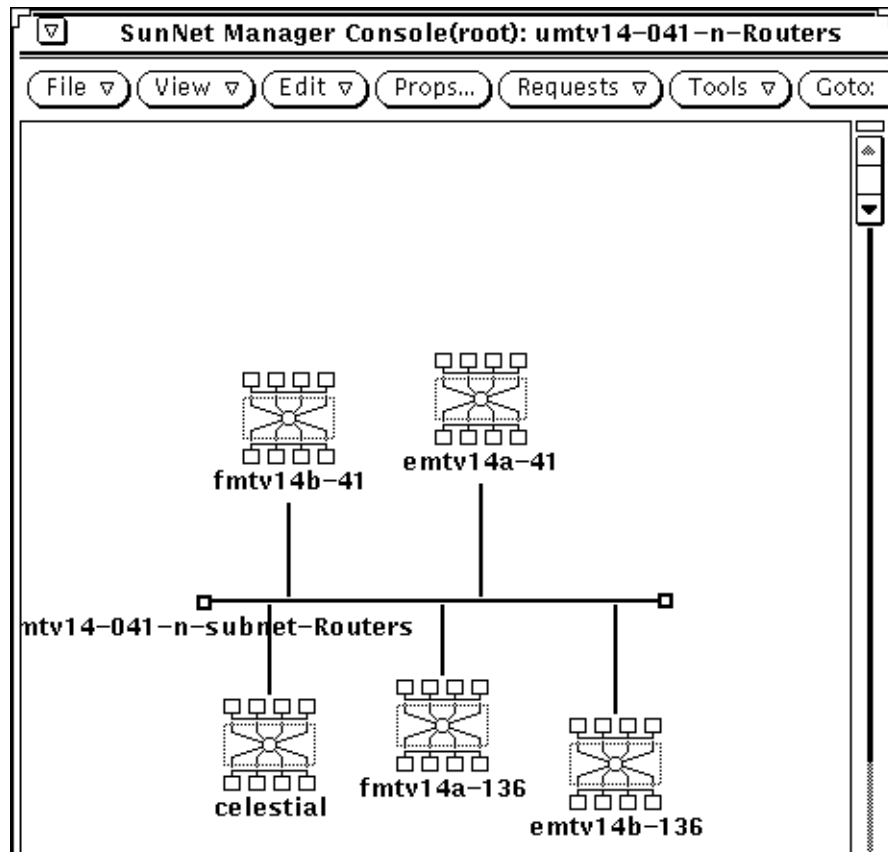


Figure 5-6 Subview in Routers-only Hierarchy

5.2.3 Monitor Function Configuration

The window for monitor function configuration is the window you receive when you press MENU in the Category button in the Discover Configuration window and release MENU over Monitor. The window shown in Figure 5-7 is displayed.

Monitor Configuration

Category: ☒ Monitor

Holding Area Viewname: holding-area **Verbose Mode:** ☐ Yes ☐ No

Non-response Timeout: 30 min **Start Time:** 8:00

Time Between Cycles: 15 min **Stop Time:** 11:00

ICMP Timeout: 4 sec **Start Date:** 10/19/93

Log History: **Stop Date:** 10/19/93

Log Filename: /tmp/monitor1393.log **Run Weekly:**

Mail Logfile To: _____ **Day of Week:** Tuesday

Ignore-host Filename: _____

Figure 5-7 Discover Configuration: Monitor Window

The parameters in the window above determine the operation of the monitor function only; they have no effect on the discover function. (However, the ICMP and SNMP-related discover parameters *do* apply to the monitor function.)

By default, the monitor function works silently. Set the Verbose Mode option (described below) to Yes to have the function display messages in the Discover Tool Console Output window. Also by default, the monitor function runs daily; you can set it to run weekly instead, by setting the Run Weekly and Day of Week parameters, described below.

To protect the integrity of already-discovered views, the monitor function adds hosts to a “holding area” view you define in the property sheet (see Holding Area Viewname, below).

The monitor parameters in the Discover Properties window are described as follows:

Holding Area Viewname

The name of the view to which the monitor function adds the hosts it finds.

Non-response Timeout

Determines how long (in minutes) the monitor function will continue to try to contact an unresponsive host before marking the host as down in the file specified in the Log Filename item (described below).

Time Between Cycles

Within the span specified by Start Time and Stop Time (see below), this parameter determines how long (in minutes) the monitor function will wait between monitor cycles. A monitor cycle is one complete pass of the monitor function through the network elements in a runtime database. A value of 0 means that the monitor function runs continuously until the Stop Time is reached.

ICMP Timeout

Determines how long (in seconds) the monitor function waits after sending an ICMP packet before sending another ICMP packet.

Log History

An on/off switch the default for which is on. Determines whether the monitor function keeps a record of its operation. You must set this parameter to On to have data sent to a log file and to have to the log file sent to an electronic mail address.

Log Filename

Full path name of the file in which the monitor function stores a record of its operation. Note that you must specify a valid path name for this parameter to obtain a record of monitor output.

Mail Logfile To:

An electronic mail address to which the monitor function sends a record of its operation upon completion of a cycle. The software performs no error checking on the address you enter.

Ignore-Host File Name

Full path name of the file listing names of hosts and views that you want the monitor function to ignore. The requirements for the file are that it be an ASCII file with one hostname or view per line, no blank lines, with host names beginning in column 1. The monitor function does not look inside a view listed in this file.

You might find the ignore-views feature useful if you have object-type-specific views. For example, if you have a router-only view, the monitor function, left unchecked, adds all of the hosts it finds on the routers' subnetworks to this view. If you insert the name of this view in the file you specify for the Ignore-Host File Name, the monitor function will not look inside this view.

Use the ignore-hosts feature if the monitor function frequently finds new or down hosts that are of no interest to you.

Verbose Mode

A yes/no switch for which the default is no. Click SELECT on Yes to have the monitor function report progress in the Discover Console Output window in the Discover base window. A no selection means that the monitor function works silently. A no selection for this parameter does not affect the logging-related parameters.

Start Time

A time at or after which the monitor function begins operation within the specified start and stop dates. The initial display is the current time. The start time takes effect for daily or for weekly use of the monitor function.

Stop Time

A time at which the monitor function ceases operation within the specified start and stop dates. The stop time takes effect for daily or for weekly use of the monitor function.

Start Date

A date on or after which the monitor function begins operation. The initial display is the current date. The start date takes effect for daily or for weekly use of the monitor function.

Stop Date

A date on which the monitor function ceases operation. The stop date takes effect for daily or for weekly use of the monitor function.

Run Weekly

A yes/no switch for which the default is no. A no selection means that the monitor function works on a daily rather than a weekly basis. When this item is set to no, the following item (Day of Week) is greyed-out. Click SELECT on Yes to have the monitor function run on a weekly rather than a daily basis.

Day of Week

If the previous item is set to yes, this item displays normally. The initial selection is the current day of the week. Press MENU in the abbreviated menu button to obtain a day-of-the-week menu and release MENU over the day on which you want to monitor function to run.

5.3 snm_discover *Command*

You can invoke the Discover Tool from the SunOS command line using the `snm_discover` command. Many of the options that you can use with this command correspond to fields in the Discover or Monitor menus of the Discover Configuration popup window. If you have installed the Solaris 1.1 version of SNM 2.2.3, you would enter:

```
hostname% /usr/snm/bin/snm_discover <options>
```

If you have installed SNM 2.2.3 on a Solaris 2.x machine, you should enter:

```
hostname% /opt/SUNWconn/snm/bin/snm_discover <options>
```

5.3.1 Discover Command-Line Options

`-c <community>`

`<community>` specifies the SNMP community string to use. Equivalent to the “SNMP Community” field in the Configuration window.

-d <objects>

By default, Discover searches for all objects. The **-d** option limits the discovery to types of objects specified in a list. Possible entries: routers, snmp, snm, or network. Entries in the list are separated by colons. Equivalent to the “Objects to Discover” field in the Configuration window.

-h <hops>

<hops> measures how far Discover will search. By default this value is 0; in that case, Discover limits its search to the local subnet. For a given route to a possible target, **<hops>** measures the number of routers that will be traversed by packets sent by Discover. This is equivalent to the “Maximum Hops” field in the Configuration window.

-l

Manageable connections are created between discovered elements and connection information is added to the `linkmap` file. Equivalent to the “Add Object Connections” field in the Configuration window.

-m <netmask>

<netmask> specifies the netmask used in Discover. This can be in hexadecimal or dotted decimal notation. Equivalent to “Netmask” field in Configuration window.

-M

Starts Monitor.

-n <network>

<network> specifies the name or subnetwork number—as specified in an NIS map, an NIS+ table, or the `/etc/networks` file—that identifies the subnetwork that will be the starting point for the discover function. By default, the discover uses the network number used by the local machine. Equivalent to the “Net Name/Number” field in the Configuration window.

-o

When a **<viewname>** is specified with the **-v** option, and the **-o** option is also specified, Discover adds elements only under the view specified in **<viewname>**. Subviews created under this view have **<viewname>** appended. This is equivalent to entering “No” for the “Add to All Views” field in the Configuration window. An example illustrating the use of this option is discussed in Section 5.2.2, “Example: Using Discover to Create a Routers-only View.”

- p *<num>*
<num> specifies the number of times Discover will reissue an ICMP echo request to a host address after an initial non-response. Equivalent to “ICMP Retries” field in Configuration window.
- q
 This invokes a quick Discover, which limits discovery to a maximum of ten devices on the local subnet.
- r *<IPAddress1>:<IPAddress2>*
 By default, Discover only pings host addresses in the range 0 to 2048 on each subnet that it finds. The -r option passes a range of IP addresses to use as targets for ICMP echo requests in searching for hosts. The use of this option is discussed in the next section.
- s *<seconds>*
<seconds> specifies the number of seconds to wait for a response to an SNMP packet before sending another SNMP packet. Equivalent to the “SNMP Timeout” field in the Configuration window.
- S *<num>*
 The number of SNMP retries when sending requests to routers. Equivalent to the “SNMP Retries” field in the Configuration window.
- t *<seconds>*
 Determines how many seconds Discover waits before sending another ICMP request after non-response to a prior request. Equivalent to “ICMP Timeout” field in Configuration window.
- T
 Invokes the Discover Tool window interface.
- v
 Turns on verbose mode. Equivalent to setting “Verbose Mode” field to “Yes” in Configuration window.
- V *<viewname>*
 Puts all discovered devices into *<viewname>*. By default, *<viewname>* is “Home.” Equivalent to the “Viewname” field in the Configuration window.
- x
 By default, Discover does some minimal layout of elements in views. If this option is used, elements are created without screen coordinates and no layout of elements takes place.

5.3.2 Discovery on Non-Subnetted Class B Networks

When Discover attempts to find elements on a network, ICMP echo requests will be sent to a maximum of 2048 host addresses on each subnetwork. By default, these host addresses will be those in the range 0 to 2048. However, non-subnetted Class B networks may have host addresses higher than 2048.

In a Class B IP network address, two octets (16 bits) of the four-octet (32-bit) IP address are available as host addresses. In a subnetted Class B network, a portion (typically one octet) of this 16-bit host address is used for subnet addresses.

On a non-subnetted Class B network, the entire 16-bit host portion of the IP address is available for host addresses, thus allowing host addresses higher than 2048. (Host addresses higher than 2048 can also occur on subnetted Class B networks if 4 or fewer bits of the host portion of the IP address are used for subnet addresses. In general, any subnetted Class A or Class B network that has 12 or more bits available for the host address can have host numbers higher than 2048.)

Although an individual Discover session will only ping 2048 devices on each network, the possible range of host addresses is not limited to the default 0–2048. The Discover `-r` option can be used to modify the range of addresses to locate devices with host addresses higher than 2048 on non-subnetted networks. This is done by invoking Discover with the `-r` option at the SunOS command line:

```
hostname# <snm_path>/snm_discover -r <IPaddress1>:<IPaddress2>
```

By default, `<snm_path>` is `/usr/snm/bin` for SunOS 4.x installations or `/opt/SUNWconn/snm/bin` for Solaris 2.x installations.

Progressively higher blocks of host addresses can be passed to Discover in subsequent sessions to exhaustively search the range of possible host addresses on large non-subnetted networks.

5.4 *The discover.conf File*

The material in this section is for those users who:

- want to modify the shape or the color of the glyphs that represent the elements found by the Discover Tool;
- want to use system descriptions returned by SNMP agents to specify which type of element to create;
- have created customized views that you want to monitor;
- want to use the Discover Tool to find agents in addition to the agents shipped with SNM.

Most users need not be concerned with this information.

The `discover.conf` file is an ASCII file stored in the following directory:

- `/var/adm/snm` on Solaris 1.x machines
- `/var/opt/SUNWconn/snm` on Solaris 2.x machines.

The file contains individual sets of configurable fields, each set identified by the following uppercase keywords:

- `OID`
- `COLORS`
- `COMPONENTS`
- `DEFAULTS`
- `MAPPINGS`
- `MONITOR COMPONENTS`
- `AGENTS`

In the `discover.conf` file, each of these keywords must be preceded by a hash mark (`#`). Except when preceding a keyword, a hash mark indicates a comment.

Within `discover.conf`, the order of fields *is* significant.

5.4.1 *OID Section*

The OID section maps object identifiers (OIDs) to element types. An element type determines the glyph used for a given element. An entry has the form `<oid> <element_type>`, where `<element_type>` is in a schema file that has been loaded into the Console. For example:

```
#OID
#
Sun Microsystems.2.1.1  component.ss10
HP.2.3.2.5             component.hpsnake
```

When a device is reachable through SNMP, the discover function tries to retrieve the OID for that device. If successful, the discover function maps the OID to a specific SNM element type. In the example above, any device returning the OID for Sun (which can be in either the textual form of the example or the numeric 1.3.6.1.4.1.42.2.1.1.1) is created as the SNM element type `component.ss10`. Using the same example, if the device returns HP.2.3.2.5 the element type is `component.hpsnake`.

5.4.2 *COLORS Section*

The COLORS section of `discover.conf` allows you to create your own color names for the two sections, COMPONENTS and DEFAULTS, that follow COLORS. As shipped with SNM, the COLORS section is as follows:

```
#COLORS
#
RED          255 0 0
BLUE         0 0 255
ORANGE       255 0 255
YELLOW       255 255 75
```

See the bottom area of an element's property sheet for slider bars that show values for different colors.

5.4.3 COMPONENTS Section

The COMPONENTS section of `discover.conf` associates element types with the colors defined in the COLORS section. For example:

```
#COMPONENTS
#
component.ssl          ORANGE
component.server       LIGHT_BLUE
bus.ethernet           RED
view.subnet            BROWN
view.network           DARK_GREEN
component.link         BLACK
```

5.4.4 DEFAULTS Section

The DEFAULTS section of `discover.conf` allows you to assign a specific color for all SNMP devices. For example:

```
#DEFAULTS
#
snmp_color             GREEN
```

With the entry above, glyphs for all SNMP devices will be colored green.

5.4.5 MAPPINGS Section

The MAPPINGS section allows you to map a keyword returned from the SNMP system description to an SNM element type. If a keyword has white space, it must be enclosed in quotes. The Discover Tool has the Sun machine types listed below as defaults.

"sparcstation 10"	component.ss10
"sparcstation 1"	component.ss1
"sparcstation 2"	component.ss2
"sparcstation 330"	component.ss330
"sparcstation 370"	component.ss370
sun386	component.sun386
sun3	component.sun3
sun4	component.sun3
sun470	component.sun3
sc2000	component.sc2000
sc1000	component.sc2000
ipc	component.ipc
ipx	component.ipx
lx	component.lx

The mappings in the `discover.conf` file are consulted before the defaults shown above. For example, if the system description returned by the remote SNMP agent is:

```
SNMP agent ibm pc
```

...and you have the following entries in your `discover.conf` file:

```
#MAPPINGS
"ibm pc"      component.pc
bridge component.bridge
```

...the type for the element created by the Discover Tool will be `component.pc`. Note that the keyword can appear anywhere in the system description returned by the SNMP agent. Using the preceding example, if a system description is "better to build a bridge," the Discover Tool will create a component of the type `component.bridge`.

5.4.6 MONITOR COMPONENTS Section

The MONITOR COMPONENTS section enables the monitor function to monitor views of types other than networks and subnets. An entry in this section has the following form:

```
<view type>
```

...where *<view type>* identifies the type of view that you want to the monitor function to monitor. The view type and view instance must be successfully loaded in the Console. The view type must have a netmask field; for the view instance, the netmask field must have a non-null value.

The view types shipped with SNM (to see the list, invoke Edit►Create in the Console's control area, then click SELECT on the View category in the Create Object window) do not qualify for monitoring because they do not have a netmask field.

5.4.7 AGENTS Section

The AGENTS section enables the Discover Tool to search for agents in addition to those shipped with SNM. An entry has the following form:

```
<RPC number>    <agent_name>    [proxy]
```

The entries in this file are described as follows:

<RPC number>

The RPC number of the agent as it is specified in the `/etc/services` file or a directory service map or table.

<agent_name>

The agent name as it appears in the properties sheet for an element. Note that this is not `na.<name>`, but simply *<name>*. *<agent_name>* must identify an agent that is in a schema file that is loaded in the Console.

proxy

If the agent is a proxy agent, use the word `proxy` to indicate its proxy status.

Results Browser



<i>Starting the Results Browser</i>	<i>page 6-2</i>
<i>Loading Files</i>	<i>page 6-6</i>
<i>Report Streams</i>	<i>page 6-7</i>
<i>Streams Menu</i>	<i>page 6-11</i>
<i>Selecting Streams</i>	<i>page 6-12</i>
<i>Folders</i>	<i>page 6-15</i>
<i>Customizing the Browser</i>	<i>page 6-17</i>

This chapter describes the SunNet Manager Results Browser Tool. The Results Browser allows you to retrieve, organize, and view stored management data.

The Browser allows you to examine data stored in the following files:

- Console Event/Trap log file. This file is normally located at:
 - `/var/adm/snm/event.log` for Solaris 1.x installations
 - `/var/opt/SUNWconn/snm/event.log` for Solaris 2.x installations.
- Console Data Report log entries that have been saved to a disk file. (Refer to Section 4.1, “Data Reports,” on page 4-1 for more information.)
- Data Report files. You use Save option in the Console Data Report window to create a log file of data reports.
- Other files that use the data format specified in `snm.logfile(5)`.

Each instance of the Browser can only work with other SNM tools that use the same user name.

6.1 Starting the Results Browser

The Browser can be invoked from either the Tools menu in the SunNet Manager Console or from a command line. It runs as a separate process from the Console and remains when you quit the Console. To start the Results Browser from the Console, press MENU on the Tools button and drag and release on the Browser menu option (see Figure 6-1).

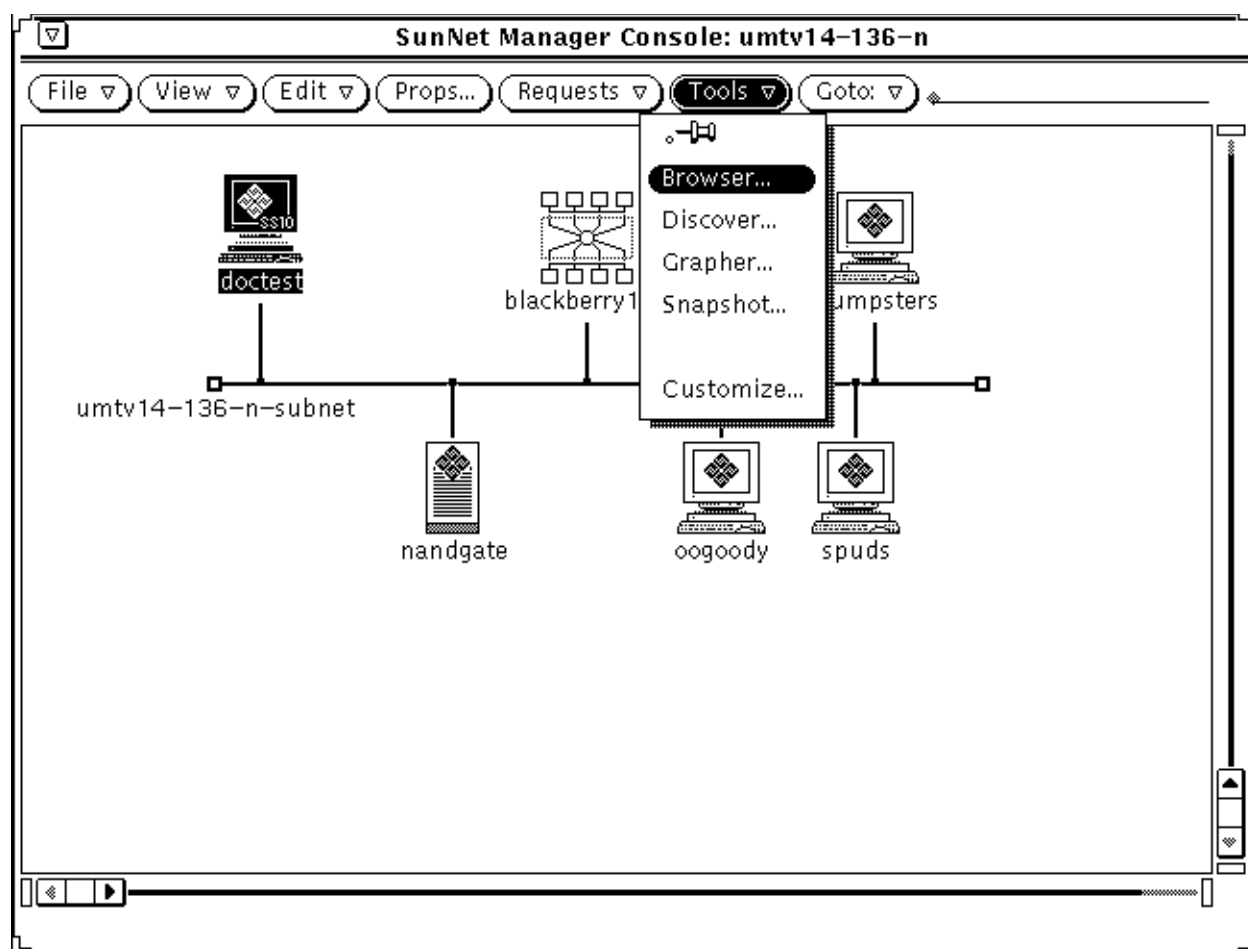


Figure 6-1 Invoking the Browser from the Console

The Browser can also be invoked from the command line by using a command of the following form:

```
mgrhost% <tools-path>/bin/snm_br [-b] [<filename1>] [<filename2>] ..
```

- *<tools-path>* is `/usr/snm` for Solaris 1.x installations.
- *<tools-path>* is `/opt/SUNWconn/snm` for Solaris 2.x installations.

When you start the Browser from a command line, you can optionally specify files to be automatically loaded into the Results Browser. File names can include wildcards. When invoked without any file names specified, the Results Browser displays the window shown in Figure 6-2.

If the Browser is started from a command line with the `-b` option, this causes graphs to be plotted on a white background, rather than the default black background, if the Results Grapher is invoked from the Browser.

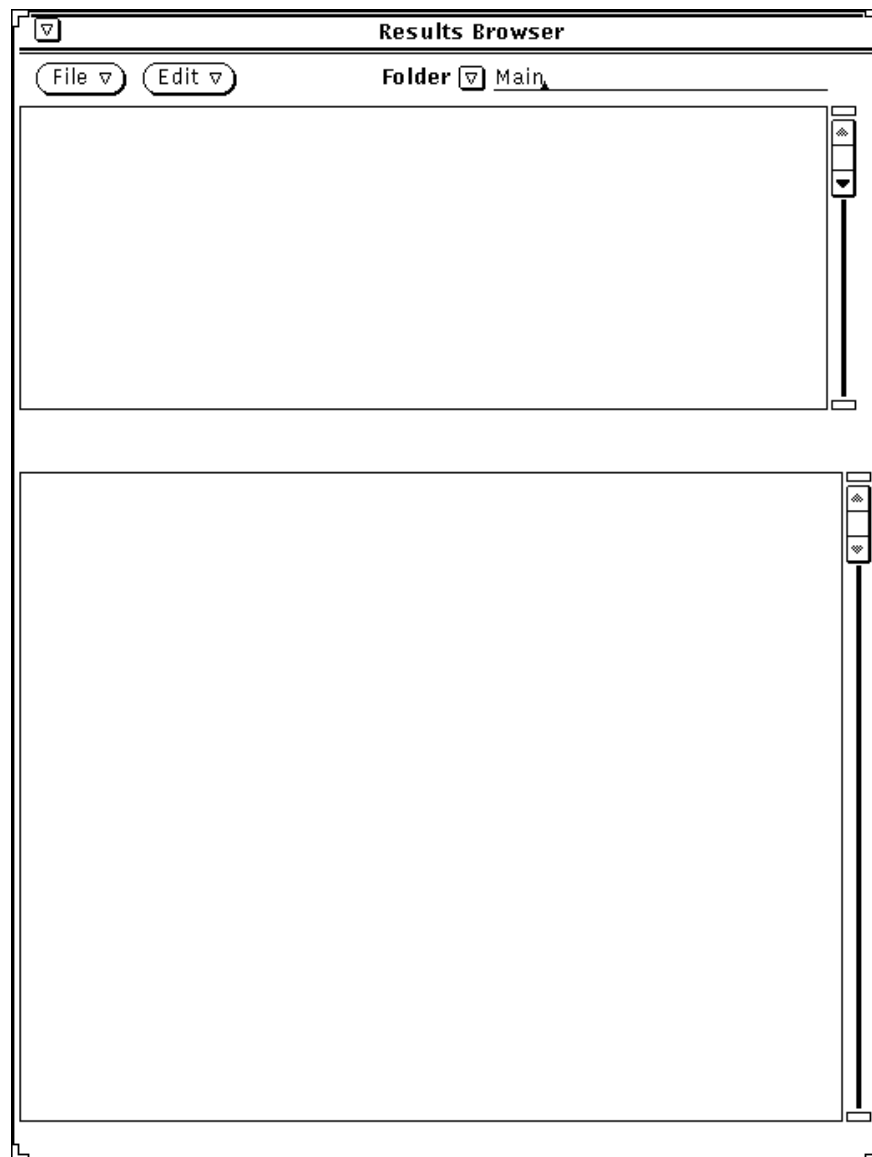


Figure 6-2 Results Browser Window

6.2 Loading Files

To load a file (which must be in the prescribed format) into the Browser, press MENU on the File button and release MENU on the Load option. A pop-up window appears, prompting you for the name of a file to be loaded. See the example in Figure 6-3.

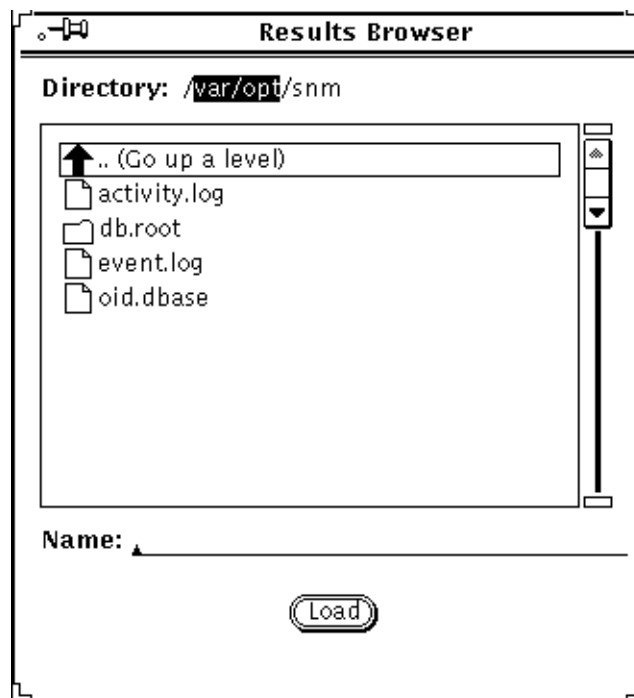


Figure 6-3 Load Window

To load a file, double-click SELECT on the file name or its icon or type in the file name at the Name prompt and then click SELECT on the Load button. You can load multiple files into the Browser, one at a time; however, you cannot specify wildcards in any file name.

Note – Because the Browser keeps open every file it reads, there is a limit to the number of files you can load in a Browser session. This limit is determined by the number of file descriptors allowed by the C-shell to a single process; the

default is 64. You can increase this number up to 256 by using the `limit` command in the C-shell. You can also run multiple Browser sessions at the same time.

6.3 Report Streams

When you load a file into the Browser, the data in the file is automatically organized into report streams, which appear in the scrolling list in the upper pane of the Browser window. See Figure 6-4.

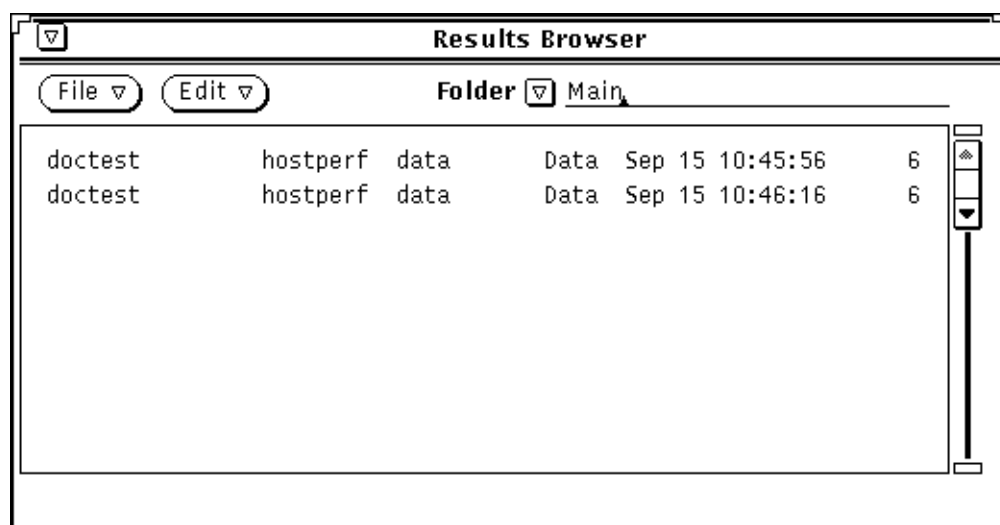


Figure 6-4 Results Browser Report Streams

Report streams are logical groupings of reports from agents. Each stream contains the results of a request from a particular requester address at a particular time. The Browser also distinguishes streams by report type (data, event, error, etc.). For example, if a file contains both data reports and error reports resulting from an individual request, the data reports are listed in a separate stream from the error reports.

Each report stream is defined in the following format in the upper portion of the Browser window:

<code><system></code>	<code><agent></code>	<code><group></code>	<code><type></code>	<code><date></code>	<code><number_of_reports></code>
-----------------------------	----------------------------	----------------------------	---------------------------	---------------------------	--

where:

`<system>` is the name (up to 15 characters) of the target system (the host where the managed object resides).

`<agent>` is the name of the agent.

`<group>` identifies the attribute group.

`<type>` identifies the type of report (data, event, trap, error).

`<date>` is the time stamp of the original request from the manager.

`<number_of_reports>` indicates the number of reports in the stream.

To view the contents (individual reports) of a particular stream, double-click SELECT on the stream you want to view. Individual agent reports from the selected stream are displayed in the lower portion of the Browser window, as shown in Figure 6-5.

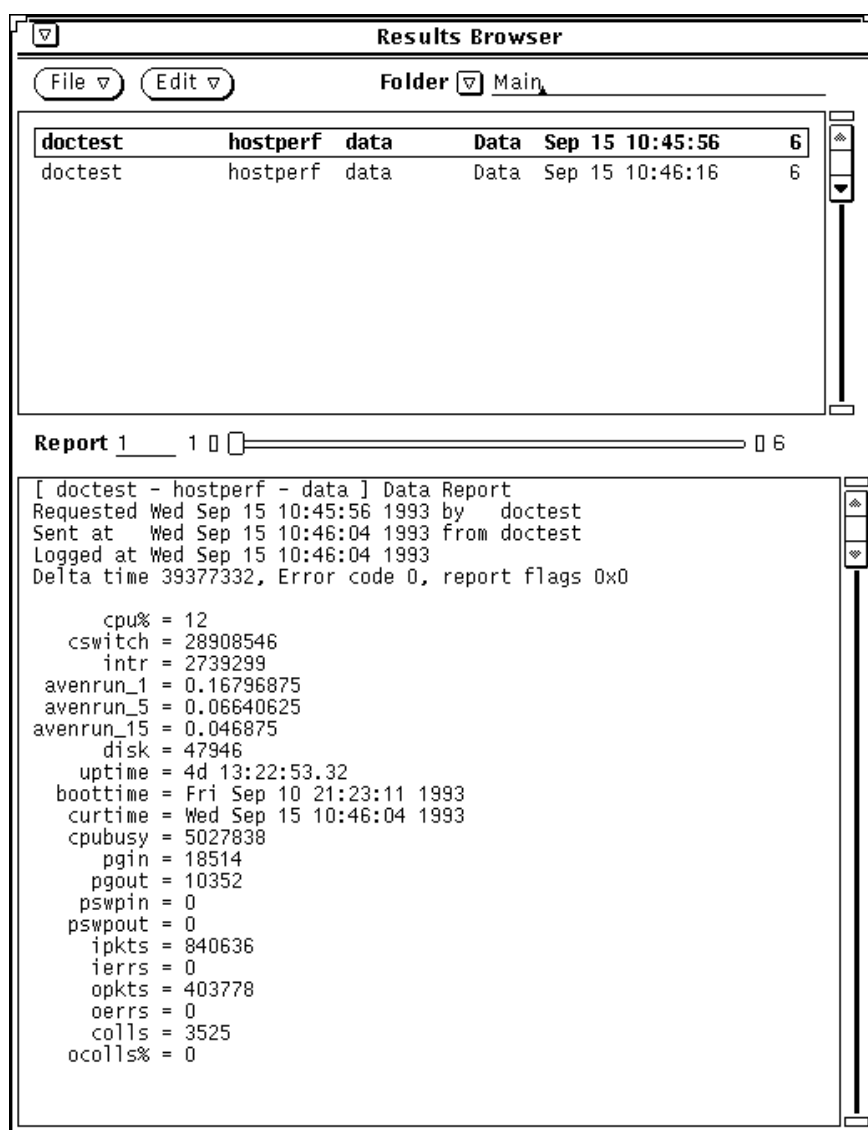


Figure 6-5 Agent Reports from Selected Stream

If more than one report exists in the selected stream, a slider bar appears. To view a specific report, type in the report number on the Report line to the left of the slider bar and press Return. You can also view reports by pressing

SELECT on the drag area (the small rectangular area between the ends of the slider bar) and dragging the mouse pointer to the right or the left until the desired report number is reached.

You can clone, delete, or print reports from the selected stream by pressing MENU in the Report window to bring up the Report menu. The Report menu is shown in Figure 6-6.

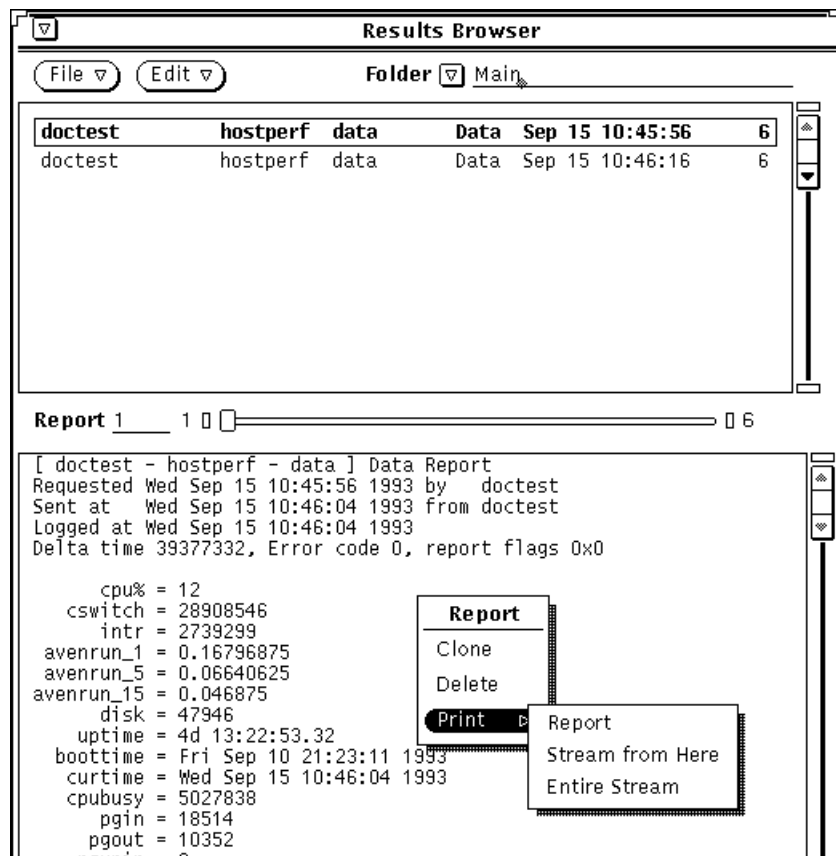


Figure 6-6 Report Menu

Report Menu functions are described as follows:

- To clone a report, select the Clone option. This causes a copy of the current report to be displayed in a pop-up window so that its contents can be compared with other reports. To make the pop-up window disappear, unpin it by clicking SELECT on the pushpin.
- To delete a report, select the Delete option.
- To print a report, press MENU in the Report window and move the pointer to one of the three Print Options:
 - Report—prints the currently displayed report.
 - Stream From Here—prints all reports in the stream, beginning with the currently displayed report.
 - Entire Stream—prints all reports in the stream.

By default, reports are printed to `lpr` on Solaris 1.1, or `lp` if SNM has been installed on a Solaris 2.x machine. You can choose a different printer by using the Tool Properties option of the Browser's Edit menu. Refer to Section 6.7, "Customizing the Browser," on page 6-17 for more information.

If the reports to be printed exceed a set number of kilobytes, you are asked to confirm that you want the reports printed before printing commences. By default, this limit is set at 100 kilobytes. You can adjust this limit by using the Tools Properties option of the Browser's Edit menu. Refer to Section 6.7, "Customizing the Browser," on page 6-17 for more information.

6.4 Streams Menu

To perform operations on one or more streams, invoke the pull-down Streams menu by pressing MENU in the top panel of the Browser window. You receive:

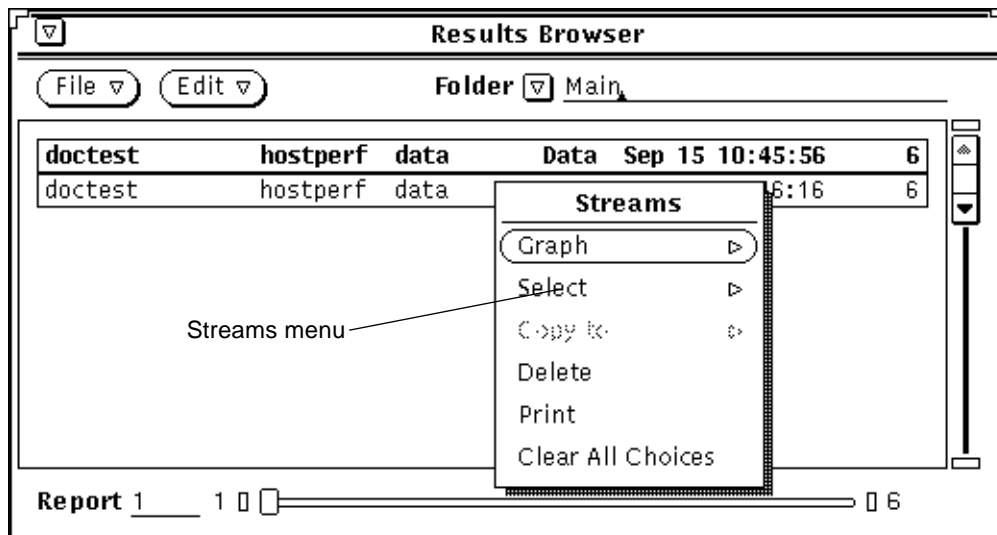


Figure 6-7 Results Browser Streams Menu

The following briefly describes Streams menu functions.

- Graph allows you to send selected streams to the Results Grapher. This function is described in detail in the section Section 6.5.1, “Sending Data to the Results Grapher.”
- Select allows you to select streams based on certain properties. This function is described in detail in the section Section 6.5, “Selecting Streams,” below.
- Copy to copies selected streams to a folder. The use of folders is described later in this chapter.
- Delete deletes selected streams.
- Print prints selected streams to your configured printer.
- Clear All Choices deselects any selected streams.

6.5 Selecting Streams

The Browser provides mechanisms for selecting streams based on certain user-specified properties. This function can be invoked by pressing MENU in the scrolling list and pulling right on the Select option of the Streams menu. You

can select streams By System, By Agent.Group, or By Report Type. When you have made your selections, the names of the report streams in the scrolling list to which the specified properties apply are highlighted. Figure 6-8 illustrates selection by system.

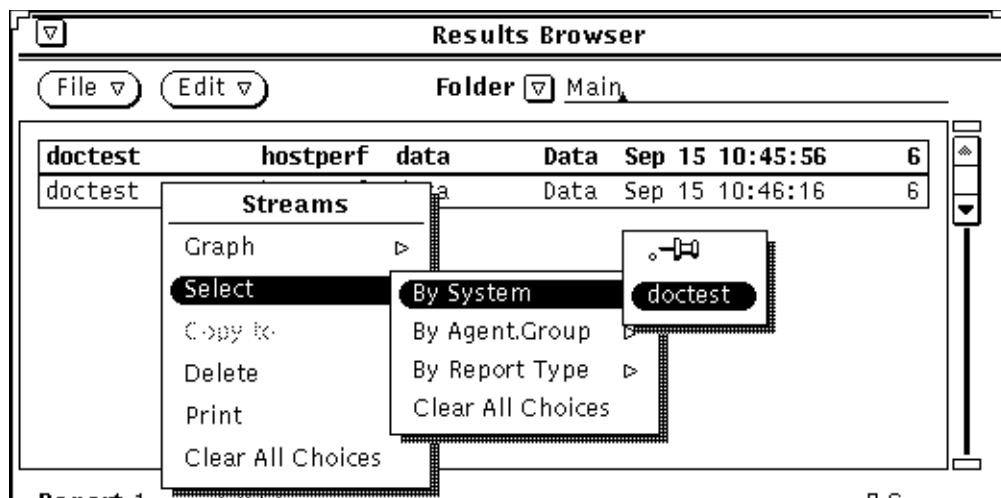


Figure 6-8 Streams Selection by System

6.5.1 Sending Data to the Results Grapher

You can have data in a report stream plotted using the Results Grapher. To send data to the Grapher, first select the desired streams. Press MENU in the scrolling list in the upper portion of the Browser window to invoke the Streams menu, pull right on Graph and release on one of the listed attribute names (shown in an alphabetized list). Figure 6-9 below illustrates how this is done:

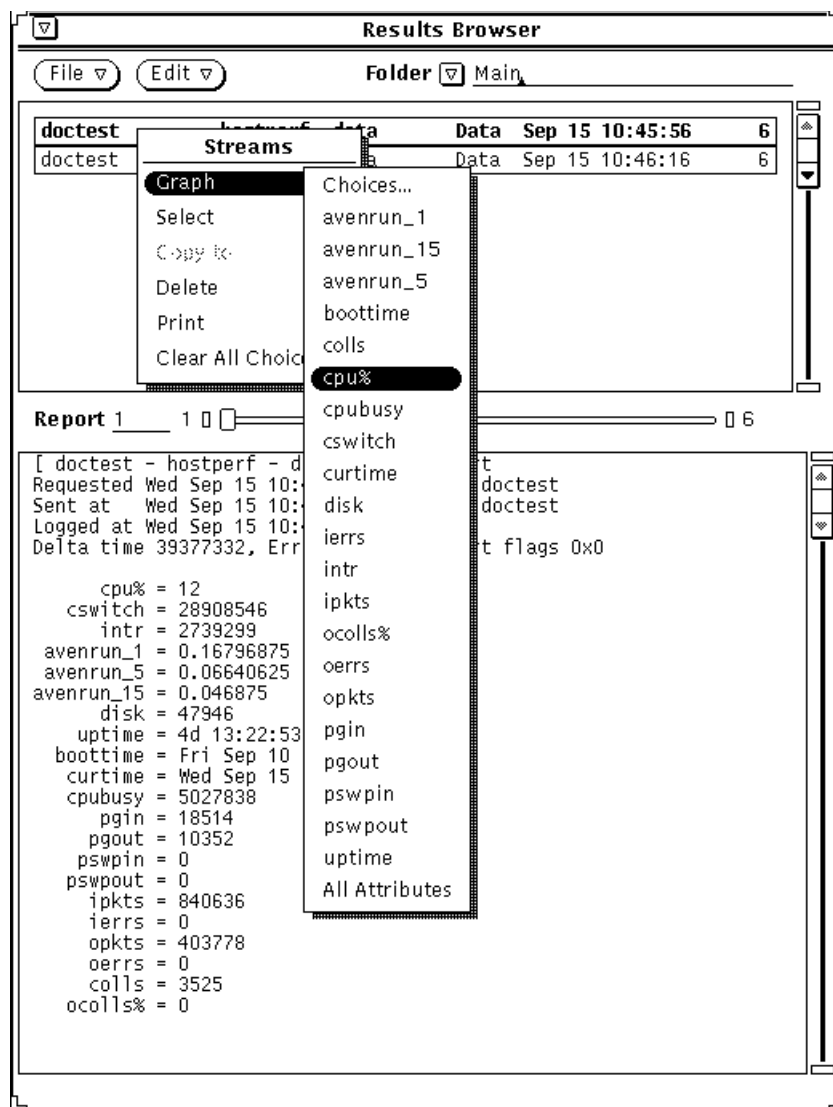


Figure 6-9 Sending Data to the Grapher

When you make this selection, the Browser sends a set of attribute and corresponding time values to the Results Grapher for plotting. Only integer, float, counter, gauge, timestamp, and UNIX time data may be plotted. Therefore, only the names of attributes of these data types are included in the menu list.

You can select more than one report stream and have each plotted on the same graph. For example, you can graph comparative data for multiple systems onto a single graph.

You can also graph multiple attributes from one or more selected report streams. To do this, pull right on either the All Attributes or the Choices option. The All Attributes option plots every integer and float attribute listed in the report stream. The Choices option displays a pop-up window with a list from which you can select the attributes to be graphed. Note that the All Attributes option is only available if there is more than one attribute that can be plotted.

If the Browser was started from a command line with the `-b` option, graphs will be plotted on a white background when the Results Grapher is invoked. Otherwise, if the `-b` option was not specified, the Results Grapher will display graphs on a black background

Once information is sent to the Grapher, you can view the data in a Grapher session. See Chapter 7, “Results Grapher,” for information on using the Results Grapher.

6.6 Folders

When you load a file into the Browser, the file is automatically loaded into the default folder `Main`. A folder is a *temporary* place to logically group report streams and store them for reference during a Browser session. You can create other folders to hold specified groups of report streams by using the Edit menu.

To create new folders, press MENU on Edit, drag the pointer to New Folder, and then release the MENU button. A folder named “New Folder” is created, which you can rename by typing the desired name on the line where “New Folder” is displayed, and pressing Return. The “Main” folder *cannot* be renamed.

To load files into a folder, press MENU on File and drag the pointer to the Load option. A pop-up window appears, prompting you for the name of the directory and file name of the file to be loaded. To load the file, click SELECT the Load button. You can load multiple files into a single folder, but duplicate reports are ignored.

You can also copy selected streams into a folder by using the Copy to option in the Streams menu. To do so, first select one or more streams by using the Select option of the Streams menu or by clicking SELECT on the desired stream in the scrolling list. Press MENU in the scrolling list and drag the pointer to Copy to. Continue to pull right to see a list of folder names; release the pointer on the desired folder. Figure 6-10 illustrates copying streams to a folder.

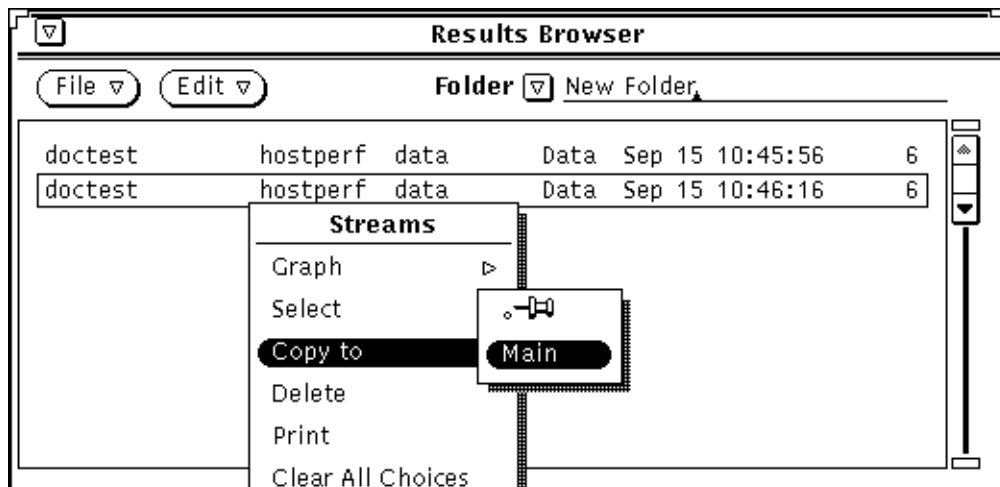


Figure 6-10 Copying Streams to a Folder

To empty a folder of all its streams, press MENU on Edit and drag the pointer to Empty Folder.

To delete a folder, press MENU on Edit and drag the pointer to Delete Folder. If you try to delete the Main folder, it will be emptied but not deleted.

As mentioned previously, folders are temporary places to keep related streams together. Once you exit the Browser, the folder names and their contents disappear. You can save the contents of a folder to a file for future access. To do so, press MENU on File and drag the pointer to the Save Folder option. In the

pop-up window, specify a path and file name and click the Save button. In a future Browser session, you can load this file to view and manipulate the same set of streams.

Caution – Do *not* attempt to save the contents of a folder into a file that contains reports that are currently loaded into the Browser, as this will corrupt the file.

Because log files may be extremely large, individual reports are not kept in memory. Instead, pointers to the files are maintained. Therefore, if a file is overwritten while it is open, the file would be destroyed and the pointers to the file would be corrupted.

6.7 Customizing the Browser

You can customize the properties of the Browser by using the Tool Properties option of the Edit menu. Press the mouse MENU button on Edit and drag and release on the Tool Properties option. The Properties window appears, as in Figure 6-11.

Results Browser – Properties

Window Position: X: Y:

Window Size: X: Y:

Print Option:

Verify printing more than **KBytes**

Report Format:

- ☐ Stream Name
- ☐ Time Requested
- ☐ Time Sent
- ☐ Time Logged
- ☐ Status Codes
- ☐ Attributes

Justify:

Figure 6-11 Tool Properties Window

The following fields in the Properties window affect the operation of the Browser. Any changes you make take effect when you press the Apply or Save buttons.

Window Position:

Defines the location (in pixels) of where the Browser window is displayed. Default values are 300 and 100 for x and y locations, respectively.

Window Size:

Defines the dimensions (in pixels) of the Browser window. Default values are 512 and 640 for x and y dimensions, respectively.

Print Option:

Specifies the printer name. `lpr` is the default printer for Solaris 1.1 installations. `lp` is the default printer for Solaris 2.x machines.

Verify printing more than (100) KBytes

Defines a print output size for printing report streams. For any output that exceeds the specified size, you will be asked to confirm whether you want to print the reports. Either enter in a new value on the line or click SELECT on the up- or down-arrow glyphs to increase or decrease the specified size.

Report Format:

Defines the appearance and contents of stream reports in the lower portion of the Browser window. All of the listed choices are selected by default, which causes all report information to be displayed in the report window. Click SELECT on any of the fields to deselect or select a choice. You can also choose to have attribute names displayed left- or right-justified on the screen. By default, attribute names are displayed right-justified.

The four buttons at the bottom of the Properties window operate as described below.

- **Apply** applies any changes that have been made to the fields in the Tool Properties window.
- **Save** saves window parameter changes to the OpenWindows `.xdefaults` file and applies any changes you made to the Report Format and Print Option parameters.
- **Load** loads window parameters from the X Resource Manager. You can then click the Apply button to apply these parameters to the Browser window.
- **Reset** sets the values in each of the fields to the last applied changes (the last time you clicked the Apply button) or to the values when the Properties window was first displayed.

Results Grapher



<i>Starting the Results Grapher</i>	<i>page 7-2</i>
<i>Results Grapher Window</i>	<i>page 7-3</i>
<i>Graph Properties Window</i>	<i>page 7-4</i>
<i>Displaying Graphs</i>	<i>page 7-8</i>
<i>Merging Graphs</i>	<i>page 7-11</i>

This chapter describes the Results Grapher Tool. The Results Grapher is a graphing utility that you can use to visualize attribute values returned from Data requests or stored data sent from the Results Browser.

In a Data request, you can choose to have reported attribute values sent to the Results Grapher for display. (This is described in Chapter 3, “Data and Event Requests.”) From the Results Browser, you can send data to the Results Grapher by selecting a report stream and pulling down a menu to graph selected attribute values from the report stream. (This is described in Chapter 6, “Results Browser.”)

Each instance of the Grapher can only work with other SNM tools that use the same user name.

7.1 Starting the Results Grapher

The Grapher can be invoked from either the Tools menu in the SunNet Manager Console or from a command line. Either way, the Results Grapher runs as a separate process from the SunNet Manager Console and remains running when you quit the Console. To start the Grapher from the Console, press MENU on the Tools button and drag and release on the Grapher menu option (see Figure 7-1).

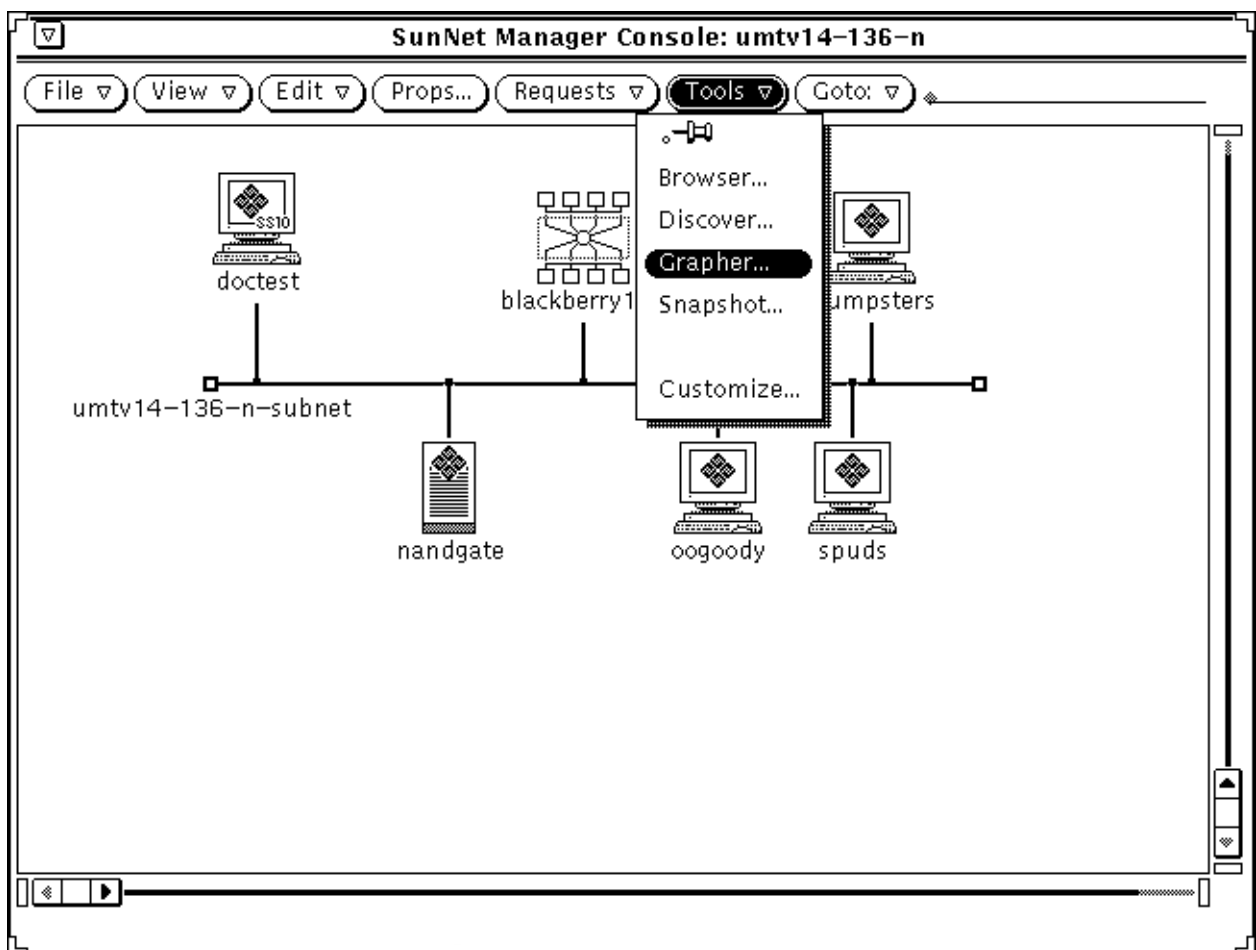


Figure 7-1 Invoking the Grapher from the Console

You can also invoke the Grapher from the command line by entering.

```
mgrhost% <tools-path>/snm_gr [-b]
```

where *<tools-path>* is:

- /usr/snm/bin for Solaris 1.x installations
- /opt/SUNWconn/snm/bin for Solaris 2.x installations.

If the *-b* option is not specified in the command line, graphs will be plotted on a black background. If the *-b* option is specified, graphs will be plotted on a white background. When you invoke the Grapher from the Tools menu, graphs are by default plotted on a black background.

To cause graphs to be displayed on a white background when you invoke the Grapher from the Tools menu, you must add the *-b* option to the *snm_gr* *elementCommand* instance in the *elements.schema* file. (Refer to Chapter 10, “Management Database,” for more information about modifying the *elements.schema* file.)

To cause graphs to be displayed on a white background when you invoke the Grapher from the Results Browser, start the Browser with the *-b* command-line option.

When the Grapher is invoked, the Results Grapher window appears. When you send data to the Grapher from the Browser, a graph window appears. The following describes these windows.

Note – When the Grapher starts, it writes its temporary RPC program number to */tmp/snm_gr.rpcid.<user>* (*<user>* is either the name of the user or the value of the environment variable *SNM_USER*). Other applications, such as the Console, the Browser, or *snm_cmd*, use this file to send data to the Grapher.

7.2 Results Grapher Window

If you previously sent data to the Grapher, the names of the graphs appear in the scrolling list; if not, the list is empty. The Results Grapher window contains a scrolling list and four buttons, as shown in Figure 7-2.

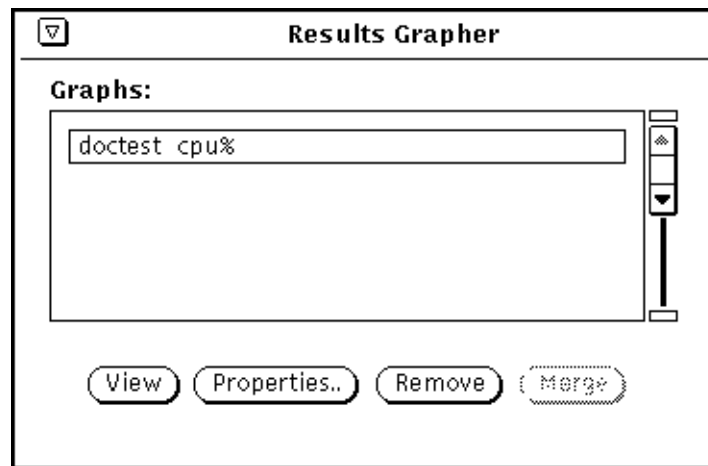



Figure 7-2 Results Grapher Window

The scrolling list contains the name of graphs that you have created. Click SELECT to select one or more of the graph names in the scrolling list. By using one of the four buttons in the window, you can perform the following operations on one or more selected graphs:

- *View* displays the selected graph in a Graph pop-up window.
- *Properties* displays the selected graph properties in a Graph Properties pop-up window.
- *Remove* deletes the selected graph, along with its properties.
- *Merge* merges the selected graphs into a single pop-up Graph window. This button is only active if there are two or more selected graphs.

7.3 Graph Properties Window

The Graph Properties window, as shown in Figure 7-3, contains parameters for the display of each graph.



Graph Properties

Name: doctest cpu%

Frame Color ▾

Base Color ▾

Data Scale:

Absolute

Relative

Time Scale:

Absolute

Relative

Order:

Normal

Reverse

Fill:

On

Off

Draw:

3D

2D

Data Sets

doctest cpu%

Data Set:

Show

Hide

Plot Value:

Absolute

Cumulative

Delta

Line Color ▾

Line Style ▾

Solid

Apply

Reset

Figure 7-3 Graph Properties

The following describes the Properties parameters:

Name:

Specifies the name of the graph that will appear in the scrolling list in the Results Grapher window, and as the label on the Graph pop-up window. By default, this field contains the host name, followed by the attribute selected to be graphed. If you are graphing data from multiple systems (for example, you select multiple Browser report streams to be plotted on the same graph), the name would contain multiple host names instead of a single name.

Frame Color

Specifies the color of the frame of the graph. By default, this value is Cyan on color monitors. To change the frame color, press MENU over the Frame Color abbreviated menu button and release the pointer on the desired color in the palette. Note that on monochrome monitors, this parameter and any other fields that specify color choices are dimmed.

Base Color

Specifies the color of the base of the graph. By default, this value is Dark Orchid on color monitors. To change the base color, press MENU over the Base Color abbreviated menu button and release the pointer on the desired color in the palette.

Data Scale:

Specifies whether the data sets (vertical axis) are plotted using Absolute or Relative values. This is useful only if more than one data set is to be plotted. If Absolute is chosen, the maximum and minimum values of the graph assume the greatest and least values among all the data sets. If Relative is chosen, the maximum and minimum values of the graph vary according to the maximum and minimum values of each data set. Absolute data scale should be used when displaying multiple data sets whose data values are of the same units—for example, CPU percentage for multiple systems. Relative data scale should be used when displaying multiple data sets whose data values are of different units—for example, CPU percentage, interrupts, and collisions for a single system. By default, Absolute is selected.

Time Scale:

Specifies whether the time stamps of the data sets are plotted in Absolute or Relative scale. If Absolute is chosen, the start and end times of the graph equal the earliest and latest times, respectively, of the data sets. If Relative is chosen, the start and end times of the graph vary according to each data set. Absolute time scale is useful when displaying data sets from reports whose time spans overlap—for example, yesterday's CPU percentage for three different systems. Data from a very short time period that is graphed with

data from a different time (for example, 30-second time slices taken six months apart) can appear almost invisible when graphed in Absolute scale. Relative time scale is useful in comparing data sets whose time stamps differ greatly—for example, CPU percentages for a single system on three different days. With Relative time scale, the time stamps at the bottom of the graph display only the relative time from the first data value. By default, Absolute is selected.

Order:

Specifies the order in which the data sets are drawn if there is more than one data set in the graph. If Normal is chosen, the graph in the foreground corresponds to the first data set in the list of the Properties window. If Reverse is chosen, the graphs are drawn in reverse order, that is, the graph in the foreground corresponds to the last data set in the list of the Properties window. By default, Normal is selected.

Fill:

Specifies the style in which the graph is to be drawn. If On is chosen, each graph is drawn in a solid fashion with the base of the graph extending up to the apex. If Off is chosen, each graph is drawn in a ribbonlike or line fashion. Setting Fill to Off allows you to view graphs that might otherwise be hidden by the graph in the foreground. By default, On is selected.

Draw:

Specifies whether the graph is drawn in a two-dimensional (2-D) or three-dimensional (3-D) manner. By default, 3-D is selected.

Data Sets

A scrolling list containing all the data sets plotted in the graph. Each item in the list has a color glyph and the host name and attribute graphed. The color glyph corresponds to the color (or line-style for monochrome monitors) used to draw the graph and may be used as a legend for the graph. This list is an exclusive list; only one item may be selected at any time.

The following four properties affect only the selected item in the Data Sets scrolling list:

Data Set:

Controls the visibility of the selected data set's graph. Show makes the data set's graph visible, while Hide makes the data set's graph invisible. By default, Show is selected.

Plot Value:

Specifies the data sets to be plotted in one of three modes: Absolute, which plots the values of the data sets as is; Cumulative, which sums up each successive data value and plots them; and Delta, which takes the difference between each pair of data value and plots them. By default, Absolute is selected.

Line Color:

Sets the line color of the graph. The Grapher automatically selects an initial color based on the number of data sets plotted.

Line Style:

Sets the line-style of the graph. This option is only active when using a monochrome monitor. One of four line-styles may be chosen: Solid, Dotted, Dashed, Dot-Dashed. By default, Solid is chosen.

To establish or change the attributes of the graph, click the Apply button. To get back the original settings of the properties just after the last Apply, click the Reset button.

Note – Changes to the default values of the Graph Properties fields *cannot* be saved from one Grapher session to another. When you exit a Results Grapher session, all changes you have made to the default values of the Graph Properties fields are lost.

7.4 Displaying Graphs

To display a graph, select it from the scrolling list in the Results Grapher window and click SELECT on the View button. A pop-up window appears as shown in Figure 7-4.

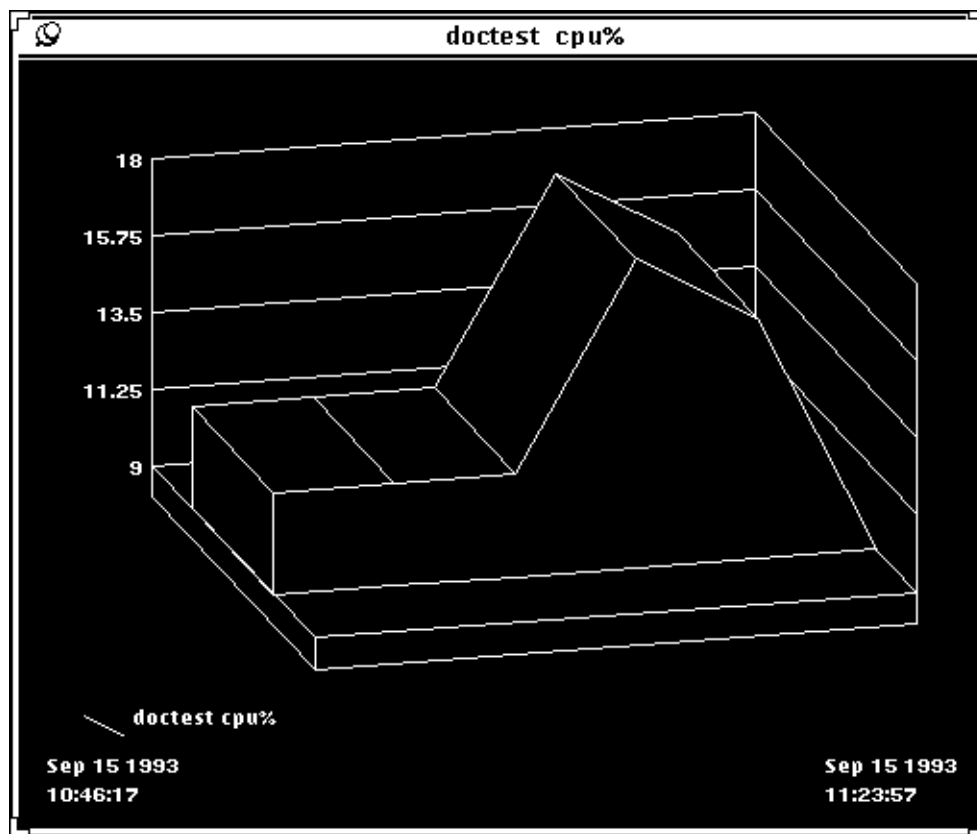


Figure 7-4 Graph

By default, graphs are plotted on a black background. To change the background to white, see the explanation on invoking the Results Grapher at the beginning of Section 7.1, "Starting the Results Grapher."

If Data Scale is set to Absolute, labels appear on the left vertical axis indicating the scales of the data sets. No labels are displayed if Data Scale is set to Relative. The start and end times of the graph are displayed at the bottom two corners of the graph; the start time is displayed at the bottom left corner and the end time is displayed at the bottom right corner.

If Time Scale is set to Absolute, the exact start and end times are displayed. If Time Scale is set to Relative, the start time is labeled 00:00:00 (<hours>:<minutes>:<seconds>) and the end time is the time elapsed since start time (<hours>:<minutes>:<seconds>).

You can zoom into a graph to get finer readings. Press SELECT at a point in the graph, then drag the mouse to another point in the graph; this must be done along the X- (time) axis. Two boundary lines mark the points that you have selected. At the same time, two time labels appear in the middle at the bottom of the graph. The time label on the left displays the start boundary time, and the label on the right displays the end boundary time. You can use this feature to inspect the times of particular points on the graph.

To replot the graph, press MENU and move the pointer to the “Replot graph with new times” option. The graph is then replotted based on the boundary times between the two points you selected. You can zoom repeatedly to get successively finer readings. To replot the original graph, press MENU and move the pointer to the “Show entire graph” option.

You can change the viewing angles of the graph in the three-dimensional view. Press MENU and move the pointer to Controls. A pop-up window containing two slider controls appears, as shown in Figure 7-5.

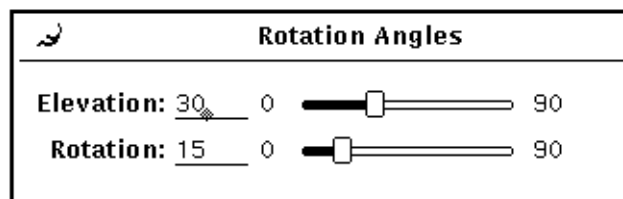


Figure 7-5 Controlling Grapher Rotation Angles

The Elevation slider controls the rotation of the graph on the vertical plane. The Rotation slider controls the rotation of the graph on the horizontal plane. You can interactively control the viewing angles of the graph by shifting the values of the sliders.

7.5 Merging Graphs

You can superimpose graphs on one another. Select the graphs to be merged from the scrolling list in the Results Grapher window, then click SELECT on the Merge button. A pop-up window appears with the selected graphs plotted together.

After graphs are merged, common parameters such as Data Scale or Time Scale take on default settings. For example, if one of the graphs being merged has Time Scale set to Relative, its Time Scale is reset to the default value of Absolute. Data set-specific parameters, such as Plot Value, retain their individual settings. Data sets are always displayed; a new line color may be assigned if there are two or more data sets using the same line color.

Set Tool



<i>Set Tool Window</i>	<i>page 8-4</i>
<i>Set Information List</i>	<i>page 8-5</i>
<i>Invoking Set Tool from the Command Line</i>	<i>page 8-6</i>

The Set Tool allows you to change attribute values. You invoke the Set Tool with the Set Request option of the element's Glyph menu. An example of selecting the Set Request option for an SNMP agent with a group type of `ifStatus` is shown in Figure 8-1.

Note – In SunNet Manager 2.2.3, the only agent that supports setting attributes is the SNMP proxy agent. See Chapter 11, “SNMP Support,” for information about the SNMP proxy agent.

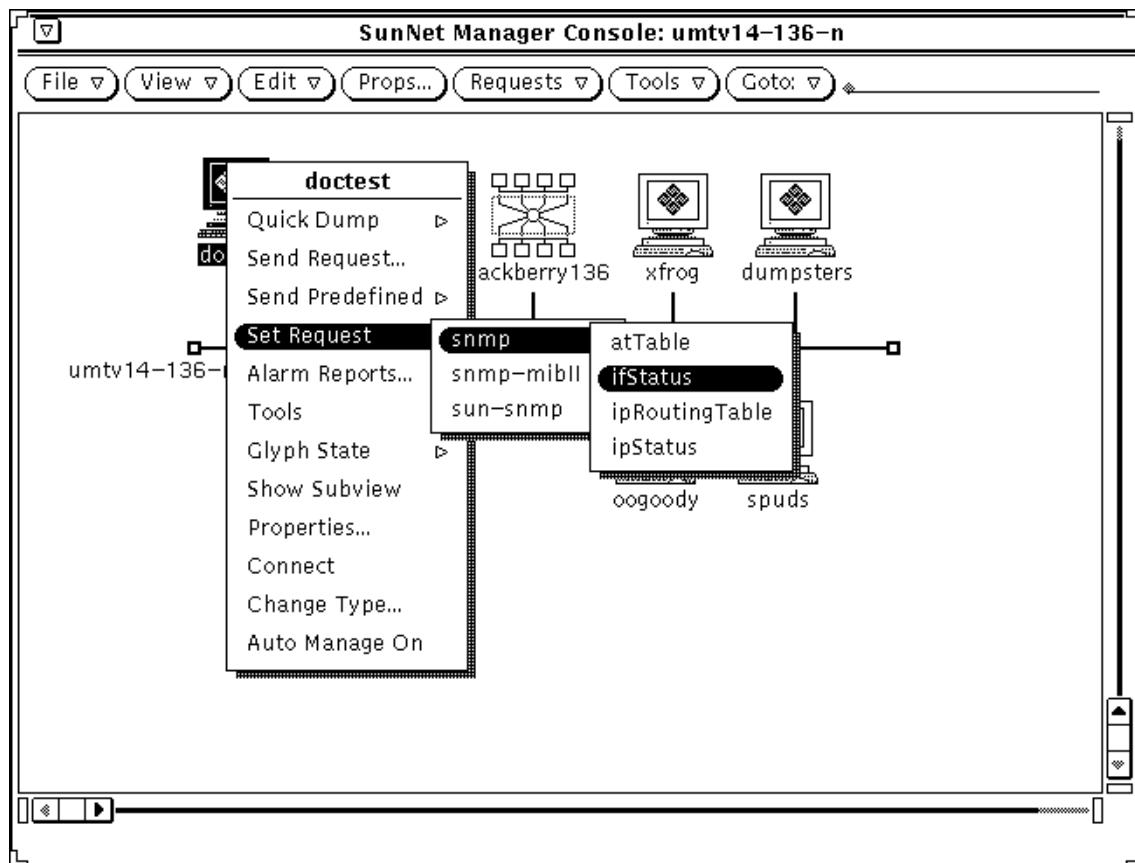


Figure 8-1 Set Request Menus

When the Set Tool is invoked, a pop-up window is displayed as shown in Figure 8-2.

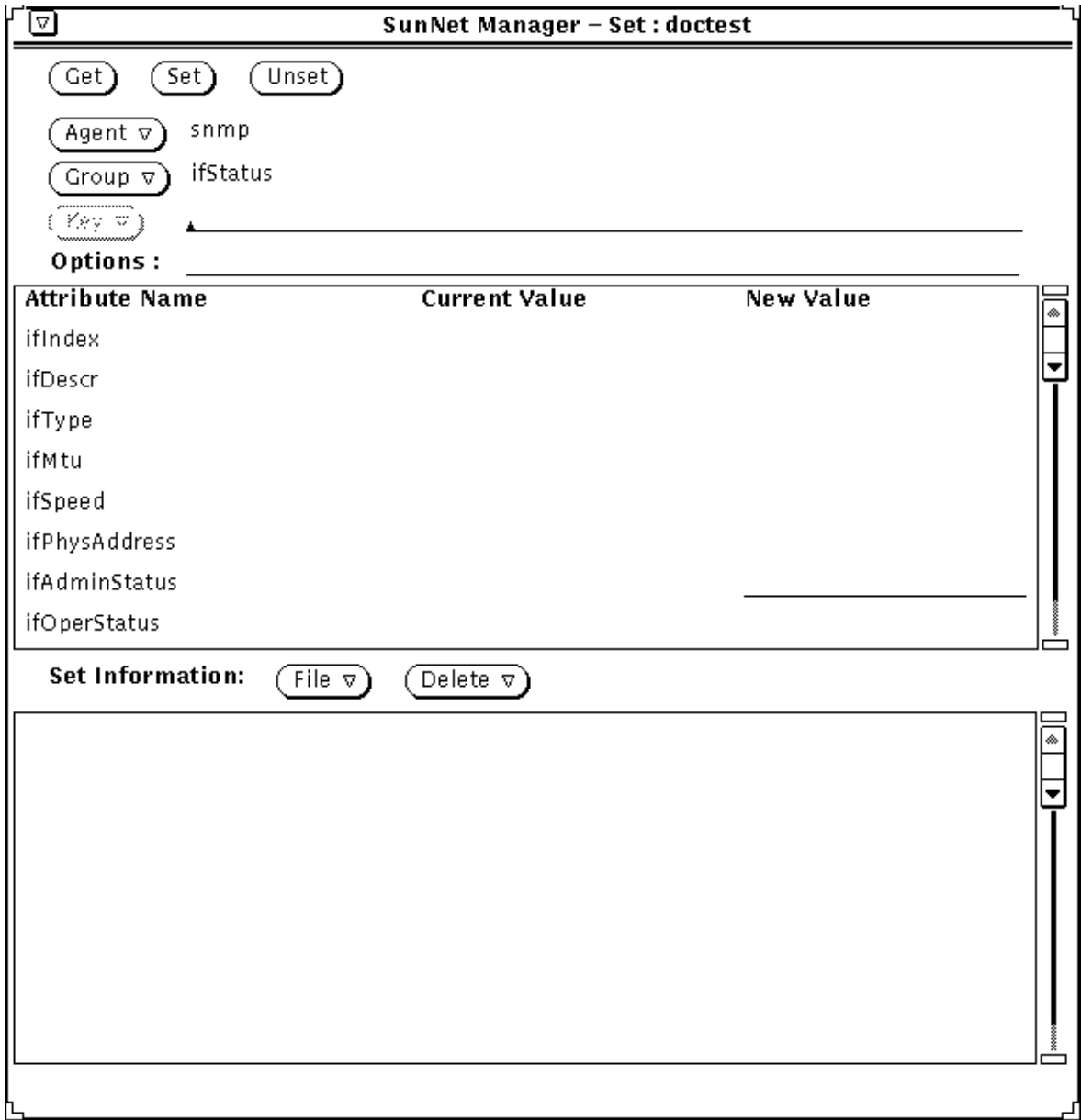


Figure 8-2 Set Tool Window

8.1 *Set Tool Window*

This section briefly describes the functions of the Set Tool window. Using the window functions to perform Get and Set requests are described later.

The system to be affected is shown to the right of Set in the title bar of the window. The Set Tool window is divided into three portions:

- The top portion is a control panel that contains control buttons and text item fields. The control panel fields determine what is displayed in the middle portion of the window. Control panel buttons and fields are described in more detail below.
- The middle portion is an attribute list that displays a group of attributes and their values. The attributes that are displayed in the attribute list are specified in the control panel.
- The bottom portion is a Set Information list. The Set Information list displays any new values that are entered into the attribute list. The Set Information list allows you to collect changes to multiple attributes (from one or more agents or groups) to be set in one operation.

The buttons in the control panel have the following functions:

- Get displays the current values for a group of attributes, or in the case of a table, a row of attributes.
- Set sends a Set request with the current set information.
- Unset undoes, if possible, the last successfully completed Set request.

The text item fields in the control panel allow you to specify the characteristics of the attributes displayed in the attribute list. These fields have the following functions:

- The Agent and Group fields are menus that allow you to select other settable agents and groups for the same system. To choose a different agent or group, press MENU on the abbreviated menu button.
- The Key field is only displayed if the group is a table and allows you to specify a key to identify a row (an instance) of attributes in the table. If you know the key of the row you want, you can simply type in the value on the line. If you do not specify a key value when making a get request, the first key in the table is shown and the Key menu becomes available with the abbreviated menu button. The Key menu consists of all the keys of the rows in a table at the moment that you make the Get request.

- The Options field allows you to specify the community name that the SNMP proxy agent should use when making a Get or Set request. The community name specified in this field overrides the community name that is defined in the Properties window for the target element.

8.2 Set Information List

The Set Information list allows you to set the values of multiple attributes—not necessarily from the same group or agent—in a single operation of the Set Tool. As individual new values are entered, they are appended to the Set Information list. When you select the Set button, the Set Tool makes the appropriate number of agent requests to change the selected values.

To change the value of an attribute displayed in the Set Information list, double-click the SELECT button on the desired attribute. The control panel and the attribute list are updated with the information about the selected attribute. You can then change the New Value column. To delete a line from the Set Information list, first click SELECT on the line to highlight it, then press MENU on the Delete abbreviated menu button and release on Delete Selection. You can also delete multiple lines at a time by selecting the lines and then using the Delete Selection option. The Delete All option from the Delete menu clears the entire Set Information list.

The File button menu allows you to store the current Set Information list to a specified file, or load previously stored Set Information data. The format of the file should be exactly the same as the contents of the Set Information list. Each line in the file (and in the Set Information list) contains the set information for a single attribute. It has the following format:

```
<agent>/<group>/<attribute>(<key>) New Value = <newvalue>
```

where:

<agent>
is the agent name of the attribute to be set.

<group>
is the group name of the attribute to be set.

<attribute>
is the attribute name to be set.

(*<key>*)

is the key of the attribute to be set, if the attribute is in a row in a table.

<newvalue>

is the new value for the attribute to be set to.

The above variables are delimited by the slash (/) character. One or more blanks must precede the keyword `New Value`.

If the file to be loaded has any agent or group data which does not apply to the current target system, the data is not loaded. Data loaded from a file is appended to any entries in the Set Information list.

While a Get or Set request is being performed, a busy pointer is displayed and the title bar of the Set Tool window is dimmed. The Set Tool window does not accept any input until a response to the request is received from the agent. If the request is taking an inordinate amount of time, perhaps due to an agent error, use the Quit option of the window menu to exit the Set Tool.

8.3 Invoking Set Tool from the Command Line

The Set Tool is implemented as a separate process and can be invoked from the command line (whether or not the SunNet Manager Console is running). The `snm_set` command line tool attempts to use whatever runtime database is available on the machine where the command is entered. If the SunNet Manager Console is running (or was running at some time), the Console's runtime database is used. If no database is found, `snm_set` returns an error message. The format for invoking the Set Tool, is:

```
host% <tools-path>/snm_set -t <target_system> -a <default_agent_name> -g <default_group_name>
```

where:

<tools-path> is:

- /usr/snm/bin for Solaris 1.1 installations
- /opt/SUNWconn/snm/bin for Solaris 2.x installations.

`-t <target_system>`

is the name of the system you want to change attribute values on.

Note – If the name *<target_system>* contains spaces, then it must be surrounded by quote marks — for example, “jaks machine”.

-a *<default_agent_name>*

is the name of the default agent you want to change attribute values for.

-g *<default_group_name>*

is the name of the default group you want to change attribute values for.

Console Properties



<i>SunNet Manager Console Properties</i>	<i>page 9-2</i>
<i>Windows</i>	<i>page 9-5</i>
<i>Requests</i>	<i>page 9-6</i>
<i>Automatic Management</i>	<i>page 9-7</i>
<i>Events and Traps</i>	<i>page 9-12</i>
<i>Errors</i>	<i>page 9-18</i>
<i>Locations</i>	<i>page 9-21</i>
<i>Miscellaneous</i>	<i>page 9-22</i>
<i>Other Configuration</i>	<i>page 9-24</i>

This chapter describes the configuration of the SunNet Manager (SNM) Console. The Console Properties window allows you to configure global preferences for the operation of the Console through a window interface. In addition, there are a couple of special Console operations you can define in a Console resource file—see Section 9.9, “Other Configuration,” on page 9-24 for more information.

9.1 SunNet Manager Console Properties

The SunNet Manager Console provides a set of global preferences that you set from the Console Props button in the Console window. You display the Console properties window by making sure no glyphs are selected in the Console window (click SELECT anywhere in the Console window that is not a glyph), then clicking SELECT on the Props button as illustrated in Figure 9-1.

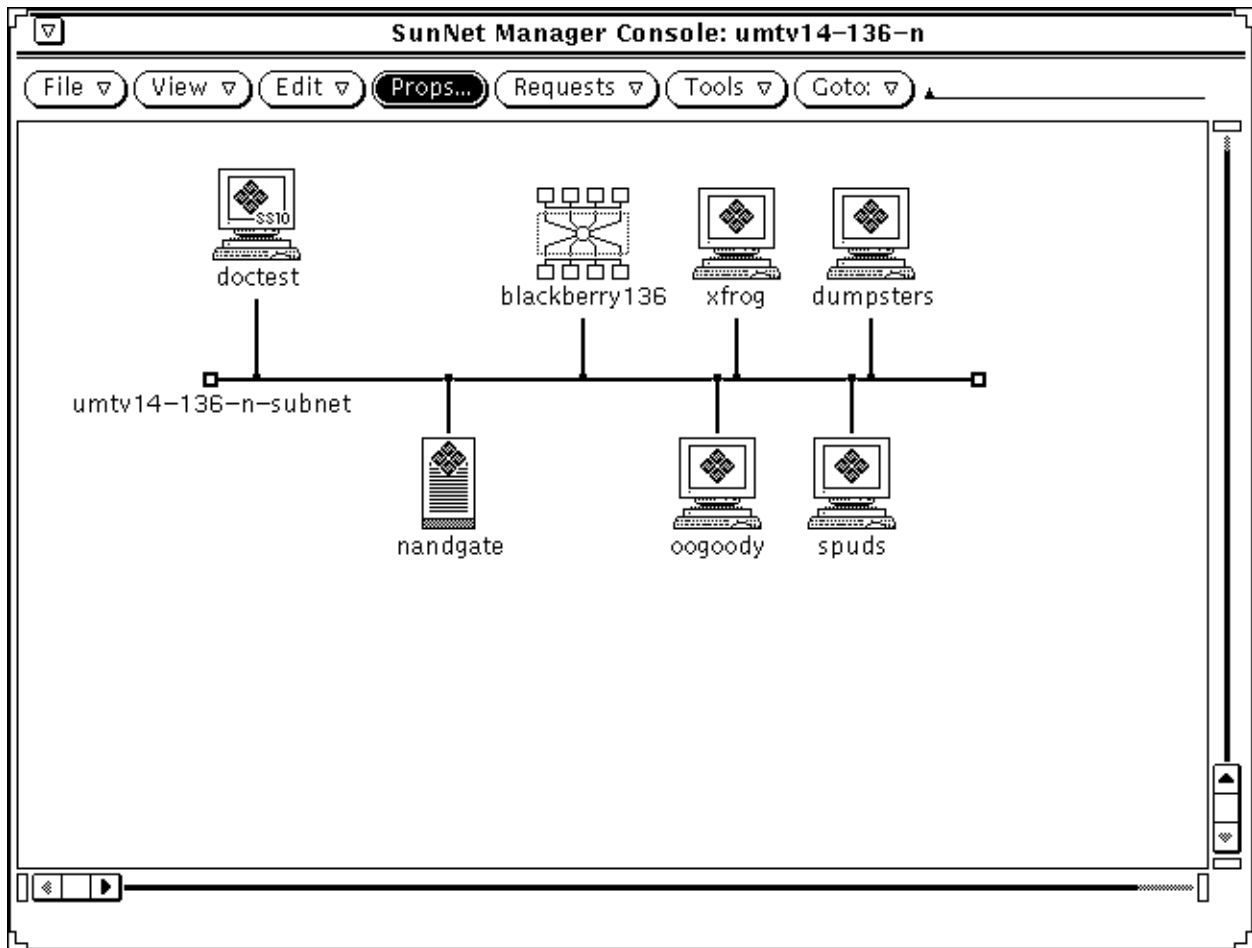
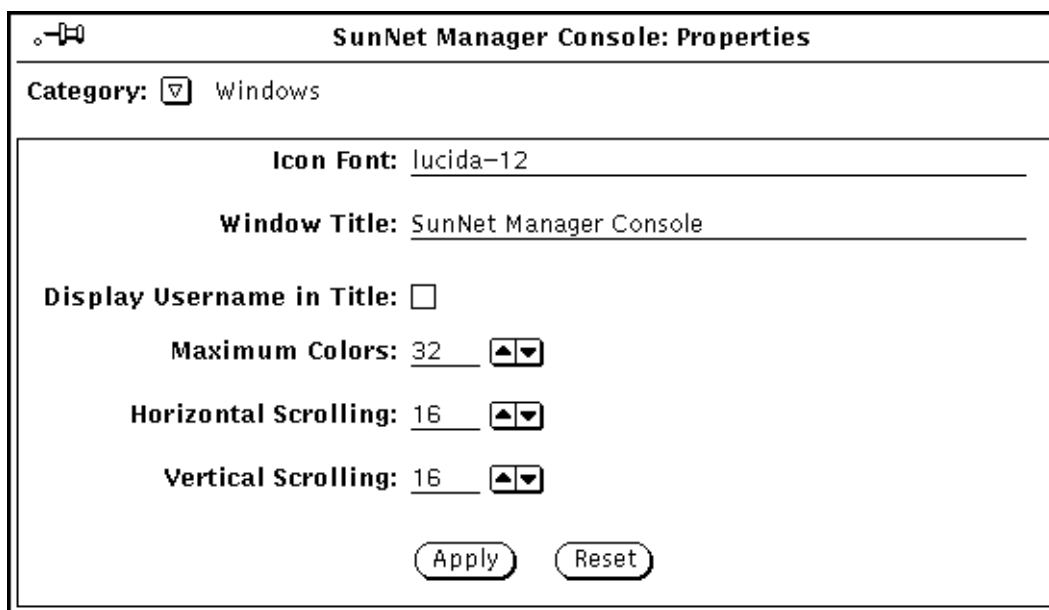


Figure 9-1 Selecting Props Button

After clicking on the Props button, the window shown in Figure 9-2 will be displayed.



The image shows a window titled "SunNet Manager Console: Properties". At the top, there is a "Category:" label followed by a dropdown menu showing "Windows". Below this, there are several settings: "Icon Font:" with a text field containing "lucida-12", "Window Title:" with a text field containing "SunNet Manager Console", "Display Username in Title:" with an unchecked checkbox, "Maximum Colors:" with a text field containing "32" and a spinner button, "Horizontal Scrolling:" with a text field containing "16" and a spinner button, and "Vertical Scrolling:" with a text field containing "16" and a spinner button. At the bottom, there are "Apply" and "Reset" buttons.

Figure 9-2 Console Properties Window

The SNM Console Properties window has a Category abbreviated menu button that provides the window categories shown in Figure 9-3.

Note – No on-line help is available for the Console Properties window.

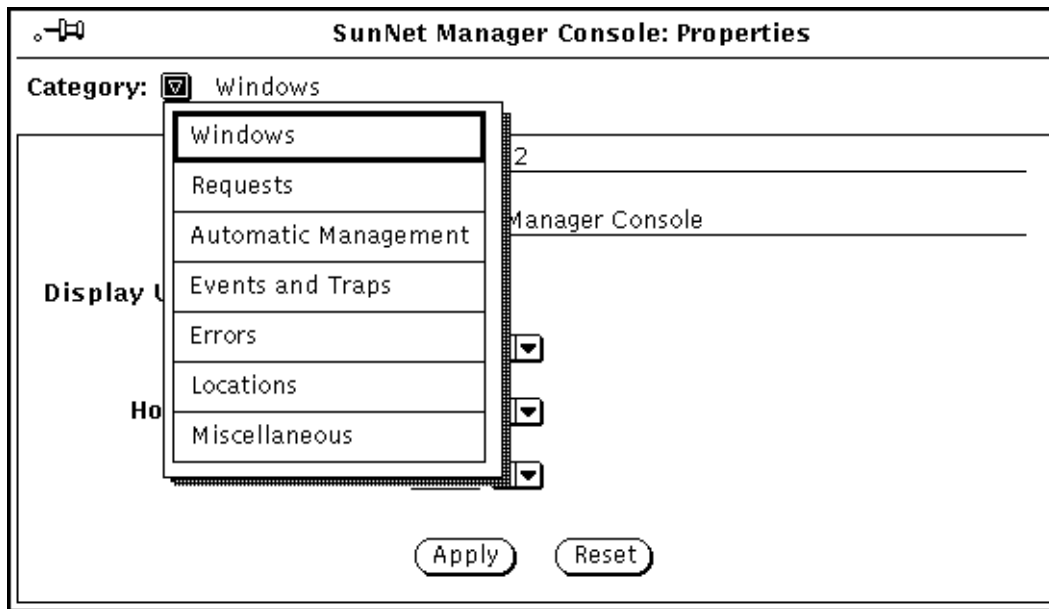


Figure 9-3 Console Properties Window Categories

These categories are described in the following subsections. When you choose a category from the menu, the name to the right of the abbreviated menu button changes to reflect the new category, and the property window displays a pane with the settings for the selected category.

Once you have entered in all the changes in a window, click **SELECT** on the Apply button to apply the new settings and dismiss the Console Properties window. Click **SELECT** on the Reset button to return to the previous settings. Click **SELECT** on the pushpin in the upper right corner to unpin, and thereby dismiss, the Console Properties window.

Most changes to window settings take effect during your current Console session, although you may not see the change immediately. For instance, if you change the window title, you will not see the change until you change views or the currently-displayed Console window is changed. A few changes to window settings do not take effect until you restart the Console—these settings are noted in the descriptions below.

9.2 Windows

The settings in the Windows category define the properties of the main windows of the SNM Console and Console tools. The Console Properties window for the Windows category is the window you receive when you first select the Props button in the Console window and is shown in Figure 9-2 on page 9-3.

The items in the Console Properties window for the Window category are as follows:

Icon Font:

Specifies the font used by the Console to display the labels under the glyphs. The default is “lucida-10.” You can enter a font that is available on your server—use the `xlsfonts` command to see available fonts.

Note – You must restart the Console in order for changes to the Icon Font setting to take effect.

Window Title:

Specifies the text that appears in the title bar of *all* of the Console's windows. The default is “SunNet Manager Console.” The text can be up to 80 characters (no quotation marks are required). If the text is too long for the size of a window, only a view name is displayed.

Display Username in Title:

Specifies whether or not the value of the environment variable `SNM_USER` (or `USER` if `SNM_USER` is not set) is displayed in the title bar in parentheses just after the value of Window Title. A check mark in the box indicates that the user name is to be displayed in the title bar—click **SELECT** on the box to toggle the check mark on or off. By default, this is off (no user name is displayed).

Maximum Colors:

Specifies the number of user colormap entries the Console uses. This value is the maximum number of unique colors which can be supported in a given view. The default is 32. To change the value, either type in a new number or click **SELECT** on the down arrow. The range of possible values is 16 to 255.

Note – You must restart the Console in order for changes to the Maximum Colors setting to take effect.

Horizontal Scrolling:

Specifies the number of pixels to move whenever you click SELECT on the left or right arrows of the horizontal scroll bar of a Console window. The default is 16. To change the value, either type in a new number or click SELECT on the up or down arrows. The range of possible values is 1 to 99.

Vertical Scrolling:

Specifies the number of pixels to move whenever you click SELECT on up or down arrows of the vertical scroll bar of a Console window. The default is 16. To change the value, either type in a new number or click SELECT on the up or down arrows. The range of possible values is 1 to 99.

9.3 Requests

The settings in the Requests category define certain Data and Event Request properties. The Requests category is shown in Figure 9-4.

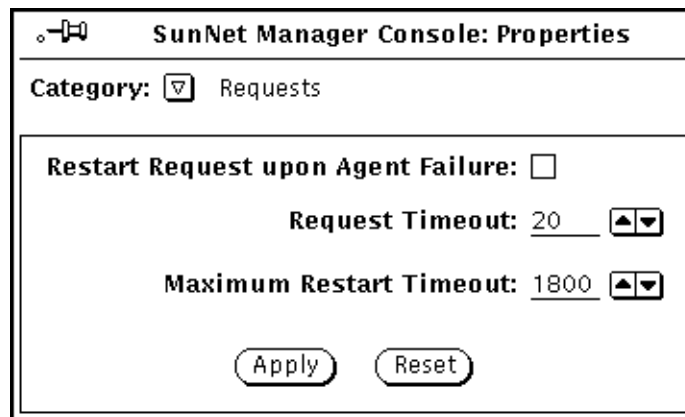


Figure 9-4 Console Properties Requests Category

Restart Request upon Agent Failure:

Specifies the default value of the Restart field in the Request Properties window. A check mark in the box specifies that whenever a new Request Properties window is displayed, the Restart field in the Request Properties

window is on (a check mark appears next to the Restart field). The default value can be overwritten in the Request Properties window for individual requests. Click SELECT on the box to toggle the check mark on or off. By default, this field is off (the Restart field is off in the Request Properties window).

Request Timeout:

Specifies the time (in seconds) that the Console waits for a response from an agent when starting or stopping requests. The default value is 20. The range of possible values is 10 to 300 (5 minutes).

Maximum Restart Timeout:

Specifies the maximum interval between attempts to start a failed request. If a request fails to start, the Console retries the request after 60 seconds. If the request fails again, it is retried again after 120 seconds. The interval between failed requests and the re-sending of the request is doubled for each restart attempt. If the maximum restart timeout value is reached before the agent responds to the request, the maximum restart timeout value becomes the interval between resent requests. The default value is 1800 seconds (30 minutes). The range of possible values is 120 to 3600 (1 hour).

9.4 Automatic Management

The settings in the Automatic Management category allow you to enable the SunNet Manager automatic node management feature. This feature allows predefined Event Requests to start automatically for elements that are added to the management database. The Automatic Management category in the Console Properties windows is shown in Figure 9-5.

The image shows a screenshot of the 'SunNet Manager Console: Properties' dialog box. The title bar includes a small icon and the text 'SunNet Manager Console: Properties'. Below the title bar, there is a 'Category:' label followed by a dropdown menu showing 'Automatic Management'. The main content area is titled 'Automatic Management:' with a checked checkbox. Below this, there are several settings: 'Polling Interval:' with a text box containing '600' and a spin button; 'Event Priority:' with three buttons labeled 'Low', 'Medium', and 'High'; 'Glyph Effect:' with four buttons labeled 'Blink', 'Dim', 'Color by Priority', and 'None'; 'Audio Effect:' with three buttons labeled 'None', 'Ring Bell', and 'Play Audio File'; 'Audio File:' with a text box containing a dotted line; 'Send Mail To:' with a text box containing 'jesse.brown@debs.Eng'; and 'Send To Program:' with an empty text box. At the bottom right, there are two buttons labeled 'Apply' and 'Reset'.

Figure 9-5 Console Properties Automatic Management Category

By default, the automatic node management feature is disabled. If you enable this feature, you can also define the reporting characteristics and signal options of automatic requests. Before you enable automatic node management, read the following section, which describes how it works.

An element can be created using the Console's Edit>Create function, by the Discover Tool, or by an application that uses the SNM database API functions. If automatic node management is enabled, a predefined Event Request is automatically started for new elements that support at least *one* of the following proxy agents:

- SNMP
- hostperf
- ping

Note – If you enable automatic node management, you can specify that predefined Event Requests *not* be sent for elements that are created in certain views or buses. This is done with the Auto Manage Off option in the bus or view’s Glyph menu. Refer to Section 2.2.12, “Auto Manage Off,” on page 2-42 for more information about this option.

If the SNMP, `hostperf`, or `ping` proxy agent is *not* specified for the element, no automatic requests are started. Note that only *one* predefined Event Request is sent, even if an element supports more than one of the above proxy agents. The event specification for each agent is described below.

Note – If you enable automatic node management and then load an ASCII database file, the Console will attempt to start a predefined Event Request for every element defined in the database that supports one of the proxy agents listed above. You can disable automatic node management for certain views. This is done with the Auto Manage Off option of the element’s Glyph menu (refer to Section 2.2.12, “Auto Manage Off,” on page 2-42 for more information). You can kill launched Event Requests at the same time by using the Requests Summary window (refer to Section 2.3, “Viewing and Modifying Requests,” on page 2-44 for more information). Once you have killed these requests, re-save the database to an ASCII file. The next time the file is loaded, no predefined Event Requests will be started.

If the element supports more than one of the SNMP, `hostperf`, or `ping` proxy agents listed above, the predefined Event Request that is actually sent is determined in the following order:

1. If you have specified that the new element can be managed with a Sun-supplied SNMP schema, the following is defined as the event for the element:

`ifOperStatus Not Equal To 1`

which specifies the condition where a device is not up.

If the element can be managed with multiple Sun-supplied SNMP schemas, the Console uses the following order of schema files:

- a. `snmp.schema` (MIB I)
- b. `snmp-mibII.schema` (MIB II)

- c. `sun-smmp.schema` (MIB II with Sun enterprise-specific extensions for Sun workstation support)

For example, if an element supports MIB I as well as the Sun workstation MIB, the Console uses the schema file associated with MIB I (`snmp.schema`) for the auto request. Note that if the element supports SNMP but does not support the `ifOperStatus` attribute, an error is returned to the Console. If this happens, the request is stopped. If the element supports the `hostperf` proxy agent, the predefined Event Request that is associated with the `hostperf` proxy agent is sent. Otherwise, if the element supports the `ping` proxy agent, the predefined Event Request that is associated with the `ping` proxy agent is sent.

2. If you have specified that the new element can be managed with the `hostperf` agent, the following is defined as the event for the element:

`uptime increased by less than <number>`

where `<number>` is the value of the Polling Interval setting in the Console Properties Automatic Management category minus 2 minutes. By default, the value of `<number>` is 8 minutes.

3. If you have specified that the new element can be managed with the `ping` agent, the following is automatically defined as the event for the element:

`reachable equal to 'false'`

Once started, an automatic Event Request can be modified like any other request. The window for the Automatic Management category is shown in Figure 9-5 on page 9-8, with the automatic node management feature enabled (checked off). If you disable the automatic node management feature, the rest of the items in the window are greyed-out.

The items in the Automatic Management category window are described as follows:

Automatic Management:

Specifies whether the automatic node management feature is enabled. A check mark in the box indicates that the feature is enabled—click SELECT on the box to toggle the check mark on or off. When this field is off (automatic node management is disabled), the remaining items in the window are greyed-out. If automatic node management is not enabled (check box is empty and you click SELECT on Apply), the Auto Manage

option in the element Glyph menus is not available. Note that if you enable automatic node management then later disable it, all active automatic requests are killed.

Polling Interval:

Specifies the interval (in seconds) for the agent to send reports. By default, the reporting frequency for automatic requests is set to '600' (10 minutes). To change the value, either type in a number on the line or click SELECT on the up or down arrows. The range of possible values is 1 to 9999 (seconds).

Caution – If you decrease this number substantially, you could cause network traffic problems.

Event Priority:

Specifies the priority of an event from an automatic node management request. The default is High. To change the setting, click SELECT on the appropriate rectangle.

Glyph Effect:

Specifies a visible indicator when an event is reported from an automatic node management request. Only one glyph effect option can be chosen. The default is Color by Priority. To change the setting, click SELECT on the appropriate rectangle.

Audio Effect:

Specifies an audible indicator when an event is reported from an automatic node management request. Only one audio effect option can be chosen. The default is None (no audible indicator). To change the setting, click SELECT on the appropriate rectangle.

Audio File:

Defines the audio file to be played when an event is reported from an automatic node management request. (The option Play Audio File must be specified in the Audio Effect field for this setting to be enabled.) Type in the path name of the audio file. The audio file will be played at the currently-set audio level.

Note – The Console must be running (not merely displaying) on a machine with an audio port. If you are running the Console on a server but displaying the Console windows on a local workstation, the Console will attempt to play the audio file on the server.

Send Mail To:

Defines one or more mail recipients to whom an Event Report is sent when an event is reported from an automatic node management request. Type in a list of mail recipients. If there is more than one recipient, use a space or comma between each entry.

Send To Program:


Defines a program or shell script to be run when an event is reported from an automatic node management request. Include the directory path, if necessary. A new copy of the program or shell script is forked for each event report. The event report is passed to the standard input of the program or shell script. For example, if the value were `cat > /tmp/trap.rpt`, it would cause the `cat` program to receive the event report and write it to the file `/tmp/trap.rpt`.


9.5 *Events and Traps*

The settings in the Events and Traps category allow you to define Console operations for events and traps. You can also specify signal options for trap reports.


Note – Trap report signal options are effective for all trap reports that the Console receives.

The window for the Events and Traps category is shown in Figure 9-6.


 **SunNet Manager Console: Properties**

Category:  Events and Traps

Open on Event or Trap: ☐

Upon Opening:  Stay in Current View


Propagate Event Effect: ☒

Event Effect Overlap Time: 10 

Trap Information

Glyph Effect:

Propagate Effect: ☒

Effect Decay Time: 0 

Audio Effect:

Audio File:

Send Mail To:

Send To Program:

Ignore Database Traps: ☒

Figure 9-6 Console Properties Events and Traps Category

Open on Event or Trap:

Specifies whether the Console window should be opened if it is closed to an icon when an event or trap report is received. A check mark in the box indicates that Console should be opened—click SELECT on the box to toggle the check mark on or off. By default, this is off (Console window is not opened).

Upon Opening:

The Open on Event or Trap setting described above *must* be checked for this setting to be enabled. Upon Opening specifies the view to be displayed when the Console window is opened and an event or trap report has been received. Press MENU in the Upon Opening abbreviated menu button to receive the following menu:

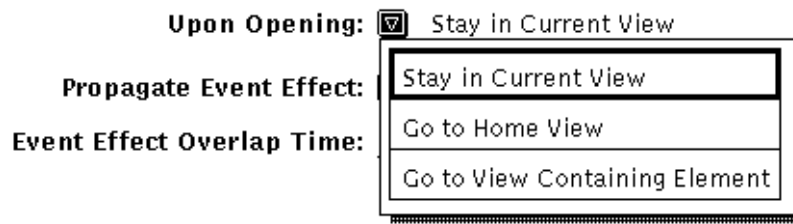


Figure 9-7 Events and Traps—Upon Opening Menu

The default is Stay in Current View (the Console displays the same view it displayed before the Console window was closed to an icon). To change the setting, press MENU on the abbreviated menu button and release on the desired option. Go to Home View causes the Console to always display the Home view while Go to View Containing Element causes the Console to display the view that contains the element for which an event or trap report is received.

Propagate Event Effect:

Specifies whether the glyph effect options of events are propagated to all parent views of the affected element. A check mark in the box indicates that propagation is enabled—click SELECT on the box to toggle the check mark on or off. By default, this is on (event's effects are propagated through the view hierarchy).

Note – You can disable glyph state propagation for a view by setting the Glyph State field in the view’s Property window to “Not Inherited.”

If a glyph is contained in more than one view, its state is propagated through its multiple view hierarchies. When the Console is closed to an icon, glyph state changes reflected in the Home view cause the SNM Console icon to display question marks.

If there are different glyph states propagating to the same view glyph, the view glyph assumes the state of the highest-priority member glyph state. The following list defines glyph-state priority from highest to lowest:

- a. Color set by the priority of the event. By default, high, medium, and low priority events are red, orange, and yellow, respectively.
- b. Dimmed.
- c. Blinking.

You can explicitly change the state of a glyph by using the Glyph State option in the Glyph menu for the element. There are two important points to remember when using this option to change glyph state:

- When you change the state of a glyph, the new glyph state is propagated as though an event had occurred to change the glyph state. For example, if you change the glyph state of an element from normal to blinking, the glyph state (blinking) propagates as if an event had occurred.
- If you have multiple glyph states propagating into a single view, and you change the glyph state for the view to normal, the member glyph states are also reset to normal. For example, if you change the glyph state of a view from blinking to normal, the effects of the event that caused a member element to blink are cleared. This allows you to clear multiple events by resetting the glyph state of a view.

Event Effect Overlap Time:

Specifies the amount of time (in seconds) beyond the reporting interval that an event’s glyph effect options remain in effect. If another event does not occur within this overlap time, the Console cancels the current signal option. The default is 10 seconds. To change the value, either type in a number on the line or click SELECT on the up or down arrows. The range of possible values is 0 (no overlap) to 86400 (24 hours).

Note – If the reporting interval is not specified in the event report, the Console assumes the reporting interval to be 60 seconds.

The following fields pertain to traps only.

Glyph Effect:

Specifies a visible indicator when a trap is reported. Only one glyph effect option can be chosen. The default is Color by Priority. To change the setting, click SELECT on the appropriate rectangle. Note that the Console can receive both high and low priority traps. (Low priority traps are generated when the SNM database API functions are used to add, change, or delete elements in the database.) If you keep the default Color by Priority option, high priority traps cause the element's glyph to turn red and low priority traps cause the glyph to turn yellow. If both high and low priority traps are received for an element, high priority is used. In association with the Priority by Color option is the decay-to-blue feature. The decay-to-blue feature pertains to traps that have been reported and for which the Effect Decay Time period has expired. In response to these conditions, the glyph for the affected element turns blue. Once a glyph has decayed to blue, the trap report must be acknowledged in order for the glyph to return to its original color. To get a glyph to return to its original color, you can either:

- i. Select the Console's View►Event Summary option
- ii. Highlight the name of the glyph
- iii. Press the Drop from List button.

Or

From the Glyph Menu for the specific element, pull right over the Glyph State►Normal option.

Propagate Effect:

Specifies whether the glyph effect options of traps are propagated to all parent views of the affected element. A check mark in the box indicates that propagation is enabled—click SELECT on the box to toggle the check mark on or off. By default, this is on (trap's effects are propagated through the view hierarchy).

Glyph state changes that result from traps propagate in the Console's view hierarchy in the same manner as with glyph state changes that result from events. See the description of the Propagate Event Effect setting for more information about propagation of glyph state changes.

Effect Decay Time:

Specifies the amount of time (in seconds) during which a trap's glyph effect option remains in effect. After the specified time, the effect caused by the trap is automatically cleared and the glyph's state is returned to "Normal" or decayed to blue. The latter only occurs when the Color by Priority option is in effect as described above under Glyph Effect. The default is 0 (glyph effects caused by a trap are not automatically cleared). To change the value, either type in a number on the line or click SELECT on the up or down arrows. The range of possible values is 0 to 86400 (24 hours).

Audio Effect:

Specifies an audible indicator when a trap is reported. Only one audio effect option can be chosen. The default is None (no audible indicator). To change the setting, click SELECT on the appropriate rectangle.

Audio File:

Defines the audio file to be played when a trap is reported. (The option Play Audio File must be specified in the Audio Effect field for this setting to be enabled.) Type in the path name of the audio file. The audio file will be played at the currently-set audio level.

Note – The Console must be running (not merely displaying) on a machine with an audio port. If you are running the Console on a server but displaying the Console windows on a local workstation, the Console will attempt to play the audio file on the server.

Send Mail To:

Defines one or more mail recipients to whom a trap report is sent when a trap is reported. Type in a list of mail recipients. If there is more than one recipient, use a space or comma between each entry.

Send To Program:

Defines a program or shell script to be run when a trap is reported. Include the directory path, if necessary. A new copy of the program or shell script is forked for each trap report. The trap report is passed to the standard input of the program or shell script. For example, if the value were `cat > /tmp/trap.rpt`, it would cause the `cat` program to receive the trap report and write it to the file `/tmp/trap.rpt`.

Ignore Database Traps:

Defines whether the traps generated when changes are made in the database are displayed by the Console. Low priority traps are generated when elements are added, changed, or deleted in the database, or if a new database file is loaded. A check mark in the box indicates that these traps are ignored by the Console — click SELECT on the box to toggle the check mark on or off.

Note – With the Ignore Database Traps option turned off, when you create an element, the trap report for that element says that the element was "created". An analogous operation using the database API returns the word "added" rather than "created".

9.6 Errors

The settings in the Errors category allow you to specify signal options for errors that the Console receives from agents. Note that the only type of errors that cause glyph state changes are errors received from agents; glyph state changes do not occur as a result of local Console errors. The window for the Errors category is shown in Figure 9-8.

The screenshot shows a window titled "SunNet Manager Console: Properties". Inside, the "Category:" dropdown is set to "Errors". The "Signal On:" section has two buttons: "All Errors" and "Fatal Errors", with "Fatal Errors" being the selected option. The "Glyph Effect:" section has four buttons: "Blink", "Dim", "Color by Priority", and "None", with "Dim" being the selected option. The "Propagate Effect:" checkbox is unchecked. The "Effect Decay Time:" is set to 0 with a spin button. The "Audio Effect:" section has three buttons: "None", "Ring Bell", and "Play Audio File", with "None" being the selected option. Below this is an "Audio File:" text field. The "Send Mail To:" and "Send To Program:" fields are empty. At the bottom are "Apply" and "Reset" buttons.

Figure 9-8 Console Properties Errors Category

Signal On:

Specifies whether signal options are in effect for all errors received from agents or only fatal errors. The default is Fatal Errors. To change the setting, click SELECT on All Errors.

Glyph Effect:

Specifies a visible indicator when an agent error is reported. Only one glyph effect option can be chosen. The default is Dim. To change the setting, click SELECT on the appropriate rectangle. Note that agent errors can be either high or low priority errors. If you choose the Color by Priority option, high priority errors (generic errors and agent-specific fatal errors) cause the element's glyph to turn red and low priority errors (agent-specific warnings) cause the glyph to turn yellow. If both high and low priority errors are received for an element, high priority is used.

Propagate Effect:

Specifies whether the glyph effect options of errors are propagated to all parent views of the affected element. A check mark in the box indicates that propagation is enabled—click SELECT on the box to toggle the check mark on or off. By default, this is off (error's effects are not propagated).

Glyph state changes that result from errors propagate in the Console's view hierarchy in the same manner as with glyph state changes that result from events. See the description of the Propagate Event Effect setting in the Events and Traps category for more information about propagation of glyph state changes.

Effect Decay Time:

Specifies the amount of time (in seconds) during which an error's glyph effect option remains in effect. After the specified time, the effect caused by the error is automatically cleared and the glyph's state is returned to "Normal." The default is 0 (glyph effects caused by an error are not automatically cleared). To change the value, either type in a number on the line or click SELECT on the up or down arrows. The range of possible values is 0 to 86400 (24 hours).

Audio Effect:

Specifies an audible indicator when an error is reported. Only one audio effect option can be chosen. The default is None (no audible indicator). To change the setting, click SELECT on the appropriate rectangle.

Audio File:

Defines the audio file to be played when an error is reported. (The option Play Audio File must be specified in the Audio Effect field for this setting to be enabled.) Type in the path name of the audio file. The audio file will be played at the currently-set audio level.

Note – The Console must be running (not merely displaying) on a machine with an audio port. If you are running the Console on a server but displaying the Console windows on a local workstation, the Console will attempt to play the audio file on the server.

Send Mail To:

Defines one or more mail recipients to whom an error report is sent when a error is reported. Type in a list of mail recipients. If there is more than one recipient, use a space or comma between each entry.

Send To Program:

Defines a program or shell script to be run when an error is reported. Include the directory path, if necessary. A new copy of the program or shell script is forked for each error report. The error report is passed to the standard input of the program or shell script. For example, if the value were `cat > /tmp/trap.rpt`, it would cause the `cat` program to receive the error report and write it to the file `/tmp/trap.rpt`.

9.7 Locations

The settings in the Locations category allow you to define the directories where the schema and icon files used by the Console are found. The Locations category is shown in Figure 9-9.)

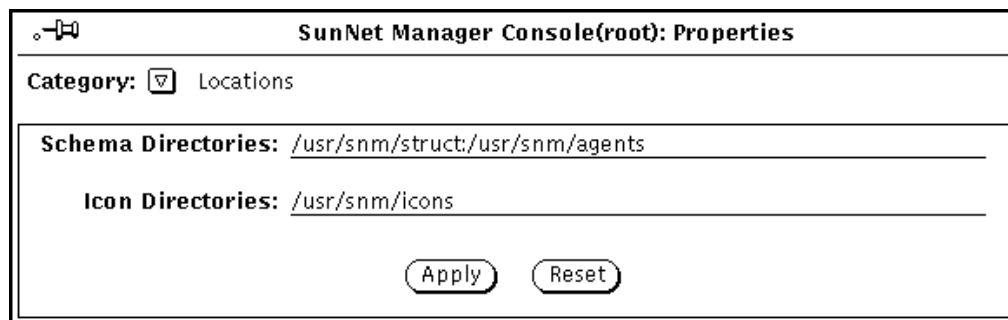


Figure 9-9 Console Properties Locations Category

Note – You must restart the Console in order for changes to the Locations settings to take effect.

Schema Directories:

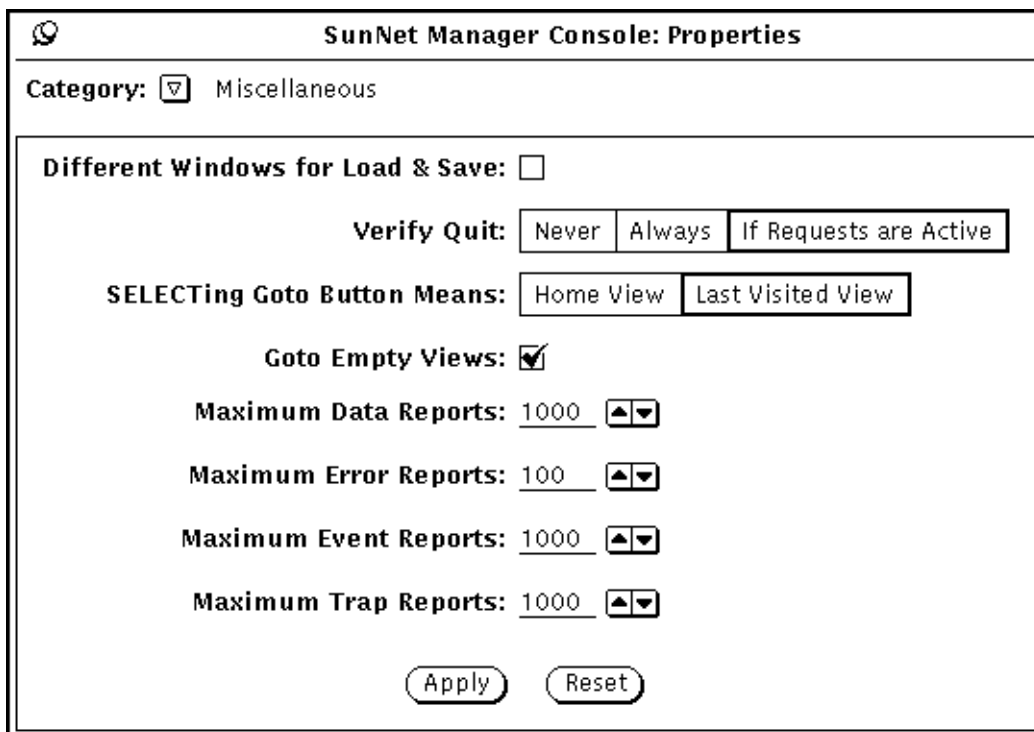
Specifies one or more directories that contain agent and element schema files. The Console searches these directories from left to right, loading any files ending in `.schema`. Multiple directories are separated by colons (:). The directories must be absolute path names (start with `/`) and cannot contain environment variables.

Icon Directories:

Specifies one or more directories that contain icon and icon mask files. The Console searches these directories from left to right, loading any files ending in `.icon` and `.iconmask`. Multiple directories are separated by colons (:). The directories must be absolute path names (start with /) and cannot contain environment variables.

9.8 Miscellaneous

The settings in the Miscellaneous category allow you to define Console parameters not covered in the other categories. The window for the Miscellaneous category is shown in Figure 9-10.



The screenshot shows a window titled "SunNet Manager Console: Properties". Below the title bar, there is a "Category:" label followed by a dropdown menu set to "Miscellaneous". The main content area contains several settings:

- Different Windows for Load & Save:** An unchecked checkbox.
- Verify Quit:** A group box containing three buttons: "Never", "Always", and "If Requests are Active".
- SELECTing Goto Button Means:** A group box containing two buttons: "Home View" and "Last Visited View".
- Goto Empty Views:** A checked checkbox.
- Maximum Data Reports:** A text field with "1000" and up/down arrow buttons.
- Maximum Error Reports:** A text field with "100" and up/down arrow buttons.
- Maximum Event Reports:** A text field with "1000" and up/down arrow buttons.
- Maximum Trap Reports:** A text field with "1000" and up/down arrow buttons.

At the bottom of the window are two buttons: "Apply" and "Reset".

Figure 9-10 Console Properties Miscellaneous Category

Different Windows for Load & Save:

Specifies whether different windows are displayed for Load and Save operations. A check mark in the box indicates that different windows are displayed—click SELECT on the box to toggle the check mark on or off. By default, this is off (the same window is used for Load or Save operations).

Verify Quit:

Specifies the verification prompt, if any, displayed by the Console whenever you choose to quit a Console session. The default is If Requests are Active (a verification prompt is displayed only if there are active requests; the verification prompt asks if you want the active requests automatically killed before the Console is exited). To change the setting, click SELECT on the appropriate rectangle: Never causes no verification prompt to be displayed, Always causes a verification prompt to be displayed even if there are no active requests.

SELECTing Goto Button Means:

Specifies the view that is displayed whenever you click SELECT on the Goto menu button. The default is Last Visited View. To change the setting, click SELECT on Home View (the Home view is displayed whenever you click SELECT on the Goto menu button).

Goto Empty Views:

Specifies whether or not the Console displays an empty view. A check mark in the box indicates that the Console can display empty views. Click SELECT on the box to toggle the check mark on or off. By default, this is on (Console can display empty views). If the box is not checked, the Console does not display empty views and an error message appears in the Console's footer. However, an empty view can still be selected if the user clicks SELECT twice on the empty view.

Maximum Data Reports:

Defines the maximum number of data reports that can be displayed in the Data Reports window. The default is 1000. The range of possible values is 10 to 99999.

Maximum Error Reports:

Defines the maximum number of error reports that can be displayed in the Error Reports window. The default is 100. The range of possible values is 10 to 99999.

Maximum Event Reports:

Defines the maximum number of event reports that can be displayed in the Event/Trap Reports window. The default is 1000. The range of possible values is 10 to 99999.

Maximum Trap Reports:

Defines the maximum number of trap reports that can be displayed in the Event/Trap Reports window. The default is 1000. The range of possible values is 10 to 99999.

9.9 Other Configuration

When you apply a setting in the SunNet Manager Console Properties window (under Props, when no glyphs are selected), the information is stored in the `$HOME/.SNMdefaults` file. The `$HOME/.SNMdefaults` file is intended to contain per-user configuration information.

Per-product defaults can be placed in `$SNMHOME/defaults`. This allows you to set up site-wide Console resources and definitions so that everyone who uses the product uses those defaults. If the user has their own `$HOME/.SNMdefaults` file, the information in the `$HOME/.SNMdefaults` overrides the information in `$SNMHOME`. Users can also set the environment variable `SNMDEFAULTS` to have the Console read another file instead of `$HOME/.SNMdefaults`. If `SNMDEFAULTS` is not set, the Console reads `$HOME/.SNMdefaults`.

Note – The Console also supports X11 resources configurations, however settings in `$HOME/.SNMdefaults` take priority.

There are a few optional definitions you can specify to further customize the operation of the Console. You can add these definitions into one of the files described above.

9.9.1 Forking Programs on Element Creation or Modification

You can specify a program that is forked for an element type when an instance of the element type is created or modified. To do this, use the following definitions:

```
snm.console.<element-category>.<element-type>.createProg: <program1>  
snm.consolecomponent.<element-category>.<element-type>.modifyProg: <program2>
```

where *<program1>* and *<program2>* are forked whenever the *<element-type>* is created or modified, respectively. Note that you can specify '*' for *<element-category>* or *<element-type>*. If you want programs to be forked for any creation or modification of an element, use the following definitions:

```
snm.console.*.*.createProg: <program1>  
snm.console.*.*.modifyProg: <program2>
```

If you want a program to be forked whenever any view is created, use the following definition:

```
snm.console.view*.createProg: <program>
```

If you want a program to be forked whenever an ethernet element type (bus.ethernet, view.ethernet, etc.) is created, use the following definition:

```
snm.console*ethernet.createProg: <program>
```

The process that is forked when an element is created has the following parameters:

```
<program1> -c -v <viewname> -t <object_type>
```

where *<viewname>* is the current view, and *<object_type>* is the type of the object created.

The process that is forked when an element is modified has the following parameters:

```
<program2> -m -v <viewname> -n <object_name>
```

where *<viewname>* is the current view, and *<object_name>* is the name of the object being modified.

9.9.2 Redirecting SNMP Requests

You can have SNMP requests from the Console redirected to another vendor's SNMP proxy. To do this, use the following definition:

```
snm.console.snmpRedirect: true
snm.console.snmpRedirectItem: <label>
snm.console.snmpRedirectAgent: <agent>
```

where *<label>* appears in the bottom left panel of the Data and Event Request Properties window and *<agent>* is the name of the SNMP proxy agent to which the requests will be redirected. In the Request Properties window, a box appears next to *<label>*—click SELECT in the box to toggle a check mark on or off. A check mark causes the request to be redirected.

Note – The Console does not kill redirected requests when it starts or exits. You must explicitly kill these requests with the `snm_kill(1)` command. This feature is intended for vendor-supplied SNMP proxies that do not return reports to the Console.

9.9.3 Activating Console Database Manager Traps for File Loads

The Console can be configured to issue a trap to inform other applications when a new runtime database is loaded. Such a trap contains the names of the elements that have been loaded.

To activate this feature, the `.SNMdefaults` file needs to be modified to set the value of `snm.console.DBMgrTrapAlways` to true. By default, this property is set to false.

Management Database

<i>Element Type Definition</i>	<i>page 10-3</i>
<i>Element Instance Definition</i>	<i>page 10-9</i>
<i>Connection Definition</i>	<i>page 10-12</i>
<i>Background Definition</i>	<i>page 10-12</i>
<i>Tools Menu Definition</i>	<i>page 10-13</i>
<i>Definition of Requests</i>	<i>page 10-14</i>

The runtime database represents the data model of a site's environment. It contains information on the agents that are supported and the objects that are managed. The information in the management database determines how the SunNet Manager Console displays the layout of the site's network, machine icons (glyphs in OPEN LOOK parlance), characteristics, and associated menus.

Certain kinds of changes to your management database can *only* be made by editing a saved MDB file with a text editor. These kinds of changes include:

- Adding or modifying element types.
- Specifying a new glyph for an element type.
- Adding or modifying tools in an element's Glyph menu.

Chapter 8 of the *SunNet Manager 2.2.3 User's Guide* describes how to make these changes. This chapter describes the format of the definitions contained in the database files. Specifically, these definitions include:

- Element type, including definition of the glyph and user commands associated with an element type.
- Element instances.
- Connections between elements.
- Background images for view instances.
- Tools available from the Console's Tools menu.
- Data requests.
- Event requests.

Input to the runtime database comes from two sets of ASCII files, *structure* files and *instance* files.

Structure files provide a set of definitions to the Console. These include:

- The `snm.glue` file that contains a starting set of definitions for the Console. The Console depends on these definitions, so do **not** modify them.
- The agent schemas that define the attributes each agent manages. (See the *SunNet Manager 2.2.3 Programmer's Guide* for a discussion of agent schema files.)
- The element schema file(s) that define the element types in a site's configuration. Element schema also specify the glyphs and commands for each element type. SunNet Manager provides definitions of numerous element types in the file, `elements.schema`. See Section 10.1, "Element Type Definition," below, for more information.

Note – Do *not* change the `elements.schema` file. If you need to add element types, create a file with the `.schema` extension in the same directory as the `elements.schema` file.

Instance files provide definitions of element and request instances.

The runtime database can also be modified by the Console through its graphical editing capability and through the Discover Tool. As you move glyphs and start requests, you are modifying the runtime management database.

10.1 Element Type Definition

Although many element types are defined in the file, `elements.schema`, additional types may be added or current types modified to customize the SunNet Manager Console to meet a particular site's needs. This section discusses element type definition.

Note – Element types are constructed from data types specified in the enumerated type `snmdb_type`, which is defined in the header file `netmgt_db.h`. These data types are listed and described in Chapter 11 of the *SunNet Manager 2.2.3 Programmer's Guide*.

There are four categories of elements. New element categories cannot be added by a user. The element categories are:

- **Components.** Components represent elements such as workstations, printers, and routers. Components appear as a glyph (a discussion of how to specify the glyph for a component follows below).
- **Views.** A view is a collection of elements. A view may contain other views. Like a component, a view also appears as a glyph.
- **Buses.** Buses represent elements such as an Ethernet LAN segment. A bus appears as a line with movable endpoints.
- **Connections.** A connection is an element that connects two other elements. A connection appears as a line connecting the two elements. Examples are a leased internetwork link and an RS-232-C line.

Element types are defined using a *record*. The record specifies the fields stored in the database for each instance of that element type. The format is:

```
record <category>.<name> (  
    <type field-name>  
    ...  
)
```

where:

<category>

one of the element categories discussed above (component, view, bus, or connection).

<name>

provides a name for the element type. Each element category is a separate namespace. For instance, it is legal to define both a `view.ethernet` and a `bus.ethernet`. The maximum length of *name* is 64 bytes.

<type field-name>

defines one or more fields that define the properties of the element. The maximum length of *<type field-name>* is 64 bytes. Certain fields must be defined, while other fields have particular functions—these are described in the following section.

- All element type definitions must have a field called `Name` which names the element instance.
- Connections must have two additional fields, `Object1` and `Object2` that name the endpoints of the connection.
- `Glyph_State` allows you to enable or disable glyph state propagation to the element.
- Components may be SNMP devices. The following fields provide SNMP information that is included with requests to the SNMP proxy agent (see Chapter 11, “SNMP Support,” for more information):
 - `SNMP_RdCommunity` specifies the SNMP community name to use when reading attribute values.
 - `SNMP_WrCommunity` specifies the SNMP community name to use when writing attribute values.
 - `SNMP_Vendor_Proxy` specifies a non-SNM SNMP proxy agent. This should only be specified when a vendor has supplied an SNMP proxy agent that communicates with the SunNet Manager SNMP proxy agent.
 - `SNMP_Timeout` specifies the number of seconds that the SNMP proxy agent is to wait for a response to requests sent to the target element.
- For `bus.ethernet` elements, `Default_Proxy` specifies the default proxy system for any elements that are discovered in the subnet. When the Discover Tool is run for the subnet, the name specified in this field is the default proxy system for each element in the subnet.
- `Label` is an optional field you can add that allows a label that is different from `Name` to appear under the glyph that corresponds to an element instance. To use the `Label` field, enter in a string in the Element Properties

sheet. If no value is entered, the value in the `Name` field is displayed. If a single hyphen (-) is entered in the `Label` field, neither the `Label` nor the `Name` value is displayed under the element's glyph.

Other fields are optional and provide information used when displaying the element's properties.

The following rules are used to display an element's properties:

- The properties are the list of field names and values in an element's record.
- Any field whose name begins with underscore is not displayed in an element's properties.
- Embedded underscores in field names are converted to blanks in the Properties list. For instance, the field `User_Name` will be displayed in the Properties under *<User Name>*.

The following example defines some element types:

```
record component.sun4 (
    string[64]    Name
    string[40]    IP_Address
    string[40]    User
    string[40]    Location
    string[80]    Description
    string[40]    SNMP_RdCommunity
    string[40]    SNMP_WrCommunity
    string[40]    SNMP_Vendor_Proxy
    int           SNMP_Timeout
)

record view.subnet(
    string[64]    Name
    string[80]    Description
    enum stateProp Glyph_State
)

record connection.rs232 (
    string[64]    Name
    string[64]    Object1
    string[64]    Object2
    string[80]    Description
)

record bus.ethernet (
    string[64]    Name
    string[40]    IP_Network_Number
    string[80]    Description
    enum stateProp Glyph_State
    string[64]    Default_Proxy
)
```

10.1.1 *Glyph Definition (for Component or View Element Type)*

Elements that are either a component or view are displayed as a glyph. An `elementGlyph` instance in the element's schema specifies the glyph for a component or a view.

An `elementGlyph` instance has the following format:

```
instance elementGlyph (  
    ( <element> "<icon-path-name>" )  
    ...  
)
```

Note that the `elementGlyph` instance must be placed *after* the element type component record. If the `<icon-path-name>` begins with a slash (/), it is treated as an absolute path. Otherwise, the path name is relative to the directories specified by the Icon Directories setting in the Console Properties Locations category. See Chapter 9, “Console Properties,” for more information about Console Properties settings.

The following example shows an `elementGlyph` instance that specifies several glyph/component mappings:

```
instance elementGlyph (  
    ( component.sun3                sun3.icon )  
    ( component.sun4                sun4.icon )  
    ( component.sun386              sun386.icon )  
    ( component.laserwriter          laserwriter.icon )  
    ( component.pc                  pc.icon )  
    ( component.genhost              mainframe.icon )  
    ( component.genws                generic-ws.icon )  
    ( component.router              router.icon )  
    ( component.bridge              bridge.icon )  
    ( component.lanbox              lanbox.icon )  
)
```

Note that you can have up to 1024 icons or raster files defined in the runtime database; this includes background images as well as element glyphs.

10.1.2 User Command Definition for an Element Type

A UNIX command may be defined to appear in the Tools menu of a component, view, bus, or connection by including an `elementCommand` instance in the element schema. This provides the ability to quickly execute any UNIX command by pointing to a glyph and selecting the command. The `elementCommand` instance has the following syntax:

```
instance elementCommand (
    ( <element> "<menu-name>" "<cmd-string>" )
    ...
)
```

where:

<element>

specifies an element type. All elements of this type will have the command in their glyph menu. The maximum length of **<element>** is 64 bytes.

<menu-name>

specifies the name of the command to appear in the menu. The maximum length of **<menu-name>** is 32 bytes.

<cmd-string>

specifies the UNIX command string to execute. Any word in the command string of the format **%field** will be replaced by the value of **field** for that element. The maximum length of **<cmd-string>** is 1024 bytes.

Note that the `elementCommand` instance must be placed *after* the element type component record.

The following example defines two commands that can be run from elements of type `component.sun4`.

```
instance elementCommand (
    (component.sun4 "Rlogin..." "$SNMHOME/bin/snm_cmdtool
$SNMHOME/bin/snm_exec rlogin %Name")
    (component.sun4 "Telnet..." "$SNMHOME/bin/snm_cmdtool
$SNMHOME/bin/snm_exec telnet %Name")
)
```


As shown in the example above, the environment variables `$SNMHOME` or `$SNMDBDIR` can be used, even if you have not explicitly set them.

If you have not set `SNMHOME` or `SNMDBDIR`, then

- `/usr/snm` and `/var/adm/snm` are assumed for Solaris 1.x environments
- `/opt/SUNWconn/snm` and `/var/opt/SUNWconn/snm` are assumed for Solaris 2.x environments.

Thus, for example, if SNM is installed on a SunOS 5.x machine, the string `"$SNMHOME/bin/snm_cmdtool"` is expanded to `"/opt/SUNWconn/snm/bin/snm_cmdtool"` if `SNMHOME` is not otherwise defined.

10.2 *Element Instance Definition*

Element and request instances are defined using cluster records. A cluster record provides a means for collecting multiple records to make up a single instance. A cluster record contains an initial record defining the instance's type followed by one or more records providing additional information about the element or request.

For example, the following cluster record defines a component `.sun4`:

```
cluster(  
    component.sun4 ( SunEng14 1.2.2.388 Engineer23  
                    "Building 2, Room 146"  
                    "Sun 4/110 - monochrome,  
                    diskfull" )  
    membership ( Subnet2 100 230 0 )  
    agent ( etherif )  
    agent ( hostif )  
    agent ( hostmem )  
    proxy ( hostperf SunEngServer )  
    agent ( layers )  
    proxy ( ping SunEngServer )  
    agent ( rpcnfs )  
    glyphColor ( 220 220 250 )  
    connect ( SunLAN )  
)
```

The first record in the example, `component.sun4`, defines the instance's type. It must be a record defined in your element's schema file. The first field in the record is normally the name of the element, in this case, `SunEng14`. The string "Building 2, Room 146" is a single field with a maximum length of 64 bytes. When a field contains an ASCII string with embedded blanks, surround the string with double quotes.

Note – Element names are not case-sensitive. `SunEng14` is the same as `sunengl4`.

The other fields in the first record correspond to the `component.sun4` record definition in the elements schema. See Section 10.1, "Element Type Definition," on page 10-3 for more information. Except for when the instance is a connection, these other fields are optional. Connections have two additional required fields that specify the endpoints of the connection.

The second record in the example, the `membership` record, places the instance in a view. Views represent a logical grouping of elements. A view may contain other views. The way you choose to arrange instances in views is up to you. Some examples are:

- Lay out your network elements in a "logical" (network) view as well as in a "physical" (machines in a building) view. For instance, display all the routers in the `Routers` view and also place them in views corresponding to their location within the network.
- Arrange views in a circular (or loop) arrangement where a view that is a member of the `Home` view can also contain the `Home` view.

The first field of the `membership` record indicates the name of the view—in this case, `Subnet2`. A special view, `Home`, refers to the view that is displayed when the Console is started. This is also the default view if no `membership` records are specified.

The remaining fields in the `membership` record provide positioning coordinates for the instance and are optional. For bus and connection elements, five coordinates are specified: X coordinate, Y coordinate, Z value (used when glyphs are stacked on top of each other in a view), and X and Y coordinates of the second point of the element. For other types of elements, only the X, Y, and Z coordinates are specified. If the coordinates are not specified, the instance will be automatically positioned in the view by the Console.

Although the example shows only a single `membership` record in the cluster, an instance may be placed into several views by including multiple `membership` records in the cluster.

The third through ninth records in the example—the `agent` and `proxy` records—provide the list of available agents and proxies for the instance. The first field in the record provides the name of the agent or proxy. `proxy` records contain a second field that supplies the name of the system providing the proxy.

The tenth record in the example is a `glyphColor` record. This record provides the color for the glyph. It is optional and, if omitted, the glyph is transparent. The three fields provide the red, green, and blue intensities, respectively. Each intensity can vary between 0 and 255. Transparent is all zeros. White is all 255s.

The final record in the example is a `connect` record. It is optional and indicates that the element is connected to another instance. This will cause a line to be drawn to the connected instance if it appears in the view. `connect` records are valid only in an instance that is a component, bus, or view. The example connects `SunEng14` to `SunLAN`. Connections made via `connect` records are referred to as simple connections since they have no name.

Cluster records can also contain a `glyphState` record, which should not be modified. If you are creating a new cluster record, you do not need to define the `glyphState` record; glyph state is assumed to be normal.

10.3 *Connection Definition*

A connection is another element category supported by the SunNet Manager Console. Connections are like any other element and can be selected and named, and can contain subviews containing other elements. A connection is created by creating a cluster whose first record is of type `connection.<type>`. The endpoints of the connection are named by the second and third fields in the `connection.<type>` record which correspond to the fields `Object1` and `Object2`.

The following example shows a definition of a link type of connection:

```
cluster (
    connection.link ( ExtLink BigHost SomeNet )
    membership ( Home 400 320 32766 401 88 )
    glyphColor ( 255 0 255 )
)
```

In this case, `BigHost` and `SomeNet` are the elements that are the endpoints of the connection named `ExtLink`. The connection will only appear in the view `Home` if the elements `BigHost` and `SomeNet` both exist in that view.

10.4 Background Definition

Background images may be associated with a view instance by including a `viewBackground` instance record in an instance file. The image should be in raster file format. Icon format is supported if the file ends with the extension `.icon`. However, large backgrounds in icon format load much more slowly than raster format files.

Before you specify a background, you need to create the view instance that will be associated with the background. If the icon path name begins with a slash (`/`), it is treated as an absolute path. Otherwise, the path name is relative to the directories specified by the `Icon Directories` setting in the `Console Properties` `Locations` category. See Chapter 9, “`Console Properties`,” for more information about `Console Properties` settings. The following example shows a definition of a background image file called `USmap.iml` with the `Home` view:

```
instance viewBackground (
    ( Home USmap.iml )
)
```

The background image file can be a maximum of 64 bytes. Note that the background definitions must be placed *after* the view instance definition. You can have up to 1024 icon or raster files defined in the runtime database; this includes background images as well as element glyphs.

Note – The preferred method of adding a background image to the current view is to invoke the View►Add Background option in the Console window. Refer to Chapter 8 of the *SunNet Manager 2.2.3 User's Guide* for more information.

10.5 Tools Menu Definition

A UNIX command may be defined to appear in the Tools menu of the Console, allowing you to launch the command as a separate process. This definition is accomplished by an `elementCommand` instance, similar to the `elementCommand` definition for an element type. The `elementCommand` instance has the following syntax:

```
instance elementCommand (
    ( SNM_Console "<cmd-name>" "<cmd-string>" )
    ...
)
```

where:

`SNM_Console` must be specified to add the command to the Console's Tools menu.

`<cmd-name>` specifies the name of the command to appear in the menu. The maximum length of `<cmd-name>` is 32 bytes.

`<cmd-string>` specifies the UNIX command string to execute. The maximum length of `<cmd-string>` is 1024 bytes.

The following example defines tools that can be invoked from the Console's Tools menu.

```
instance elementCommand (
    (SNM_Console    "Browser..."    "$SNMHOME/bin/snm_br")
    (SNM_Console    "Grapher..."    "$SNMHOME/bin/snm_gr")
    (SNM_Console    "Snapshot..."    "snapshot")
)
```

As shown in the example above, any string in the form `$name` is replaced by the expanded `<name>` environment variable. The string `"$SNMHOME/bin/snm_br"` is expanded to `"/usr/snm/bin/snm_br"` if you have the `SNMHOME` environment variable set to `/usr/snm`. (If you have not set the `SNMHOME` environment variable, `/usr/snm` is assumed for Solaris 1.1 installations; `/opt/SUNWconn/snm` is the default for Solaris 2.x installations.)

10.6 Definition of Requests

This section discusses the format of requests. Most users will want to create requests from the Console rather than manually building requests as described below. However, this information is included as reference information on how requests are stored in the management database.

Every request, whether active or held, is defined by a cluster record. The definitions of the records used in these clusters can be found in the Console definitions file. This file is located at:

- `/usr/snm/struct/snm.glue`, on Solaris 1.1 installations
- `/opt/SUNWconn/snm/struct/snm.glue`, for Solaris 2.x installations.

Like other instances, the type of request (data or event) is determined by the first record in its cluster. Requests for data begin with a `dataRequest` record, while event requests begin with an `eventRequest` record.

10.6.1 Data Requests

The following example is a cluster record representing a Data Request:

```
cluster(
  dataRequest ( hostperf.data.0 0 SunEng14
                SunEngServer hostperf data
                "'Interval' 'Count' Times"
                10 5 "" "" "" undefined True
                False "Save Request" undefined
                hostperf.data SunEng14 )
  dataAttribute ( cpu% True 1 True 1473696 None
                  0 )
  dataAttribute ( intr False 0 False 0
                  "Delta Values" 1515136 )
  rqstState ( Idle 0 0 )
  membership ( SunEng14 460 260 0 )
)
```

The first record is the dataRequest record. See Table 10-1.

Table 10-1 Data Request — dataRequest Record

Field #	Field Name	Description
1	Name	Name of the request. Displayed under the request's glyph and in messages about the request. Not sent to the agent. Maximum length is 64 bytes.
2	_serialnumber	Serial number of the request. Set serial number to 0 for new requests.
3	_targetsystem	Name of the target system. Passed to the agent as the <i><system></i> name. If the next field is NULL, this field is also used to determine the location of the agent. Maximum length is 64 bytes.
4	Proxy_System	Name of the proxy system. Used to determine the location of the agent. If null, the value of the previous field is used to determine the location of the agent. Maximum length is 64 bytes.
5	_agent	Name of the agent. Maximum length is 64 bytes.
6	_group	Name of the group/table. Maximum length is 64 bytes.
7	Interval	Interval field passed to agent.
8	Count	Count field passed to agent.
9	Key	Key field passed to agent. Maximum length is 128 bytes.

Table 10-1 Data Request — dataRequest Record

Field #	Field Name	Description
10	Restart	Boolean indicating whether to automatically restart the request if it dies or the system on which it is running dies. Value must be either <code>True</code> or <code>False</code> .
11	Defer_Reports	Boolean indicating whether sending responses should be deferred until requested by the Console. Value must be either <code>True</code> or <code>False</code> .
12	On_Completion	Disposition of request when it terminates. Value must be either <code>Delete Request</code> or <code>Save Request</code> .
13	Options	Options field passed to agent as an argument with the name of the string in <code>netmgt_optstring</code> . Maximum length is 128 bytes.
14	Log_to_File	File name for data reports. If <code>NULL</code> , reports are sent to the data log window. Maximum length is 128 bytes.
15	_fileport	Internal variable for the Console. Set to undefined for new requests. Do not modify for existing requests.
16	To_Program	Sends reports to a program. This is the name of the program.
17	_prmpport	Internal variable for the Console. Set to undefined for new requests. Do not modify for existing requests.
18	_timestamp	Internal variable for the Console. Set to undefined for new requests. Do not modify for existing requests.
19	_groupId	<code><Agent>.<group></code> name (should match the values of parameters 5 and 6).
20	_targetobject	Object associated with the request. Used to determine the name that should be put in error messages, which glyph should be blinked, etc. Must be the name of an element in the database. Maximum length is 64 bytes.
21	SnmpRedirect	Boolean indicating whether an SNMP request is to be redirected to another (vendor's) SNMP proxy agent.

Following the `dataRequest` record are zero or more `dataAttribute` records. These records control how data is displayed by the Console. If no `dataAttribute` records are in a request element, all data is displayed in the Data Reports Log (or specified file). See Table 10-2 for a description of the `dataAttribute` record.

Table 10-2 Data Request—dataAttribute Record

Field #	Field Name	Description
1	Attribute	Name of the attribute. Maximum length is 64 bytes.
2	Data_Log	Boolean indicating whether attribute should be displayed in the Data Log. Value must be either True or False.
3	_asciipoint	Internal variable for the Console. Set value to 0 for new requests. Do not modify for existing requests.
4	Indicator	Boolean indicating whether this attribute should be displayed in an indicator. Value must be either True or False.
5	_bargraphpoint	Internal variable for the Console. Set value to 0 for new requests. Do not modify for existing requests.
6	Strip_Chart	If and how the attribute should be displayed in a strip chart. Value must be either None, Absolute Values, or Delta Values. None specifies no strip chart, while Absolute Values and Delta Values specify a strip chart based on <absolute> or <delta> (current - previous) values.
7	_stripchartpoint	Internal variable for the Console. Set value to 0 for new requests. Do not modify for existing requests.
8	Graph_Tool	If and how the attribute should be displayed in the Grapher. Value must be either None, Absolute Values, or Delta Values. None specifies no graph, while Absolute Values and Delta Values specify a grapher based on <absolute> or <delta> (current - previous) values.
9	_grapherhandle	Internal variable for the Console. Set value to 0 for new requests. Do not modify for existing requests.

Following the dataAttribute records is the rqstState record. See Table 10-3 for a description of the rqstState record.

Table 10-3 Data Request — rqstState Record

Field #	Field Name	Description
1	state	State of the request. If you wish the request to be automatically activated when the Console is started, set this field to Active and set the Restart field in the dataRequest record to True. Otherwise set this field to Idle. Refer to the rqstmgr_state enumeration in snm glue for other possible values.
2	reqflags	Internal variable. Always set this to 0.
3	timestamp	Internal variable. Always set this to 0.

The final record is a membership record. This record indicates in which element's subview the request glyph will appear. See Table 10-4 for a description of the membership record.

Table 10-4 Data Request — membership Record

Field #	Field Name	Description
1	view	Name of the element in whose subview the request glyph appears. Normally, this would be the same as the final value of the dataRequest record. Maximum length is 64 bytes.

The remaining fields in this record specify X and Y positioning. These fields should be left blank and the Console used to position the request glyph.

10.6.2 Event Requests

The following shows a cluster record representing an Event Request:

```
cluster(
    eventRequest ( hostperf.data.1 1 SunEng14
                  SunEngServer hostperf data Once
                  0 1 "" "" False False False
                  "Delete Request" undefined
                  hostperf.data SunEng14 )
    eventAttribute ( cpu% "Greater Than" 80
                    "Threshold Not Set" ""
                    "Blink Glyph" "" Low )
    rqstState ( Idle 0 0 )
    glyphState ( 32 )
    membership ( SunEng14 )
)
```

The first record is an `eventRequest` record. See Table 10-5 for a description of the `eventRequest` record.

Table 10-5 Event Request — `eventRequest` Record

Field #	Field Name	Description
1	Name	Name of the request. Displayed under the request glyph and in messages about the request. Not sent to the agent. Maximum length is 64 bytes.
2	_serialnumber	Serial number of the request. Internal serial number used by the tool. Set this field to 0 for new requests.
3	_targetsystem	Name of the target system. Passed to the agent as the <system> name. If the next field is NULL, this field is also used to determine the location of the agent. Maximum length is 64 bytes.
4	Proxy_System	Name of the proxy system. Used to determine the location of the agent. If NULL, the value of the previous field is used to determine the location of the agent. Maximum length is 64 bytes.
5	_agent	Name of the agent. Maximum length is 64 bytes.
6	_group	Name of the group or table. Maximum length is 64 bytes.
7	Interval	Interval field passed to agent.
8	Count	Count field passed to agent.
9	Key	Key field passed to agent. Maximum length is 128 bytes.
10	Restart	Boolean indicating whether to automatically restart the request if it dies or the system on which it is running dies. Value must be either <code>True</code> or <code>False</code> .
11	Send_Once	Boolean indicating whether the request terminates after the first report. Value must be either <code>True</code> or <code>False</code> .
12	Defer_Reports	Boolean indicating whether sending responses should be deferred until requested by the Console. Value must be either <code>True</code> or <code>False</code> .
13	On_Completion	Disposition of the request when it terminates. Value must be either <code>Delete Request</code> or <code>Save Request</code> .
14	Options	Options field passed to agent. Maximum length is 128 bytes.
15	_timestamp	Internal variable for the Console. Set to <code>undefined</code> for new requests. Do not modify for existing requests.
16	_groupId	<Agent>.<group> name (should match values of parameters 5 and 6).
17	_targetobject	Object associated with the request. Used to determine the name that should be put in error messages, which glyph should be blinked, etc. Must be the name of an element in the database. Maximum length is 64 bytes.
18	SnmpRedirect	Boolean indicating whether an SNMP request is to be redirected to another (vendor's) SNMP proxy agent.

Following the `eventRequest` record are one or more `eventAttribute` records. These records set thresholds that determine when an event should be reported. Each record can set up to two thresholds for a particular attribute. See Table 10-6 for a description of the `eventAttribute` record.

Note – Only one record can be defined for a particular attribute. If you define multiple `eventAttribute` records for a request, each `eventAttribute` record must be for a different attribute.

Table 10-6 Event Request — eventAttribute Record

Field #	Field Name	Description
1	Attribute	Name of the attribute. Maximum length is 64 bytes.
2	Relation1	Relation for first threshold. Refer to the enumeration <code>threshold</code> in <code>snm.glue</code> for a list of possible relations.
3	Threshold1	Value for the first threshold. In the example, an event report is generated if <code>cpu%</code> was greater than 80. Maximum length is 16 bytes.
4	Relation2	Relation for the second threshold. <code>Threshold Not Set</code> indicates there is no second relation.
5	Threshold2	Value for the second threshold. Maximum length is 16 byte.
6	Priority	Defines the priority of the event. Must be either <code>Low</code> , <code>Medium</code> , or <code>High</code> .
7	Glyph_Effect	Action when event is reported. Refer to the enumeration <code>visualopt</code> in <code>snm.glue</code> for a list of possible actions.
8	Audio_Effect	Action when event is reported. Refer to the enumeration <code>audioopt</code> in <code>snm.glue</code> for a list of possible actions.
9	Audio_File	Name of the audio file. Maximum length is 128 bytes.
10	Mail_To	Mail address. Maximum length is 128 bytes.
11	To_Program	Shell script or program. Maximum length is 128 bytes.
12	_relop	Internal value. Leave blank or set to undefined.

Following the `eventAttribute` records is a `rqstState` record. See Table 10-7 for a description of the `rqstState` record.

Table 10-7 Event Request — `rqstState` Record

Field #	Field Name	Description
1	state	State of the request. Set this field to <code>Active</code> to automatically activate the request when the Console is started. Maximum length is 64 bytes.
2	timeout	Internal value. Always set this to zero.
3	reqflags	Internal value. Always set this to zero.

The final record is a `membership` record. This record indicates in which element's subview the request glyph will appear. See Table 10-8 for a description of the `membership` record.

Table 10-8 Event Request — `membership` Record

Field #	Field Name	Description
1	view	Name of the element in whose subview the request glyph appears. Normally, this would be the same as the final value of the <code>dataRequest</code> record.

SNMP Support

<i>SNMP Proxy Agent Operation</i>	<i>page 11-2</i>
<i>Creating a Schema File</i>	<i>page 11-7</i>
<i>Creating an SNMP Host File</i>	<i>page 11-9</i>
<i>Asynchronous Event Reports (Traps)</i>	<i>page 11-11</i>
<i>Mapping SNMP Object IDs to Strings</i>	<i>page 11-25</i>

SunNet Manager provides a proxy agent that supports the Simple Network Management Protocol (SNMP), the most widely-implemented network management protocol for managing TCP/IP networks. The SNMP proxy agent allows you to get data and event information from and set attribute values for devices that are managed via SNMP.

Note – SunNet Manager also provides an SNMP agent for Sun workstations called the `snmpd` daemon. The SunNet Manager Console communicates with the `snmpd` daemon through the SNMP proxy agent. The `snmpd` daemon also allows Sun workstations to be managed by other SNMP management stations. For more information about the `snmpd` daemon, see the `snmpd(8)` manual page.

This chapter discusses the operation of the SunNet Manager SNMP proxy agent and trap daemon. Using the `build_oid` utility to map SNMP object identifiers to strings is also described. For step-by-step information on how to add SNMP devices to your management database and set up the SNMP proxy agent and trap daemon, refer to the *SunNet Manager 2.2.3 User's Guide*.

Note – This chapter assumes that you are familiar with SNMP concepts. If you will be creating your own enterprise-specific schemas, you should also understand SNMP group, table, and attribute definitions.

11.1 SNMP Proxy Agent Operation

The proxy agent runs on Sun workstations—either the one on which the management application is running or another workstation on the network. SunNet Manager management applications, such as the Console, communicate with the SNMP proxy agent (`na.snmp`) using the same RPC-based protocol as with other SunNet Manager agents. The SNMP proxy agent communicates with other devices using the SNMP protocol defined in RFC 1157, as shown in Figure 11-1.

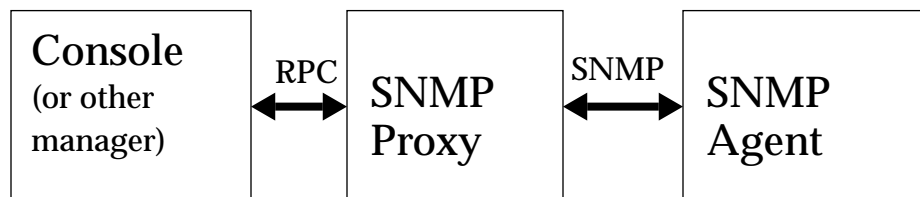


Figure 11-1 SNMP Proxy Agent

The proxy agent allows you to manage any number of management information bases (MIBs), in which you can define either standard SNMP MIB objects or enterprise-specific objects. The proxy agent uses a schema file to map objects described in a MIB and SunNet Manager attributes. A schema file is the representation of a MIB used by SunNet Manager. How to generate schema files from MIBs is discussed later in this chapter. To ensure successful operation, a schema file must have a set of object definitions that are identical to those in the MIB—see Figure 11-2.

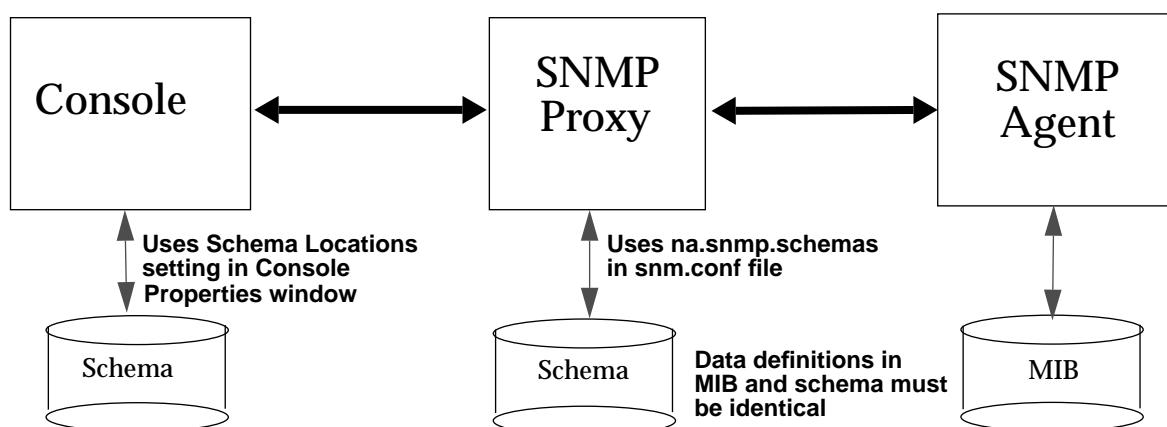


Figure 11-2 MIB and Schema Definitions

Four SNMP schemas are supplied with SunNet Manager:

- `snmp.schema` describes MIB I, as defined by RFC 1156.
- `snmp-mibII.schema` describes MIB II, as defined by RFC 1213.
- `snmpv2-mibII.schema` describes MIB II, as used by SNMP version 2. See Chapter 12, “SNMP Version 2 Support” for a description of SNMPv2 support in the current release of SNM.
- `sun-snmp.schema` describes the MIB associated with the SNMP agent (`snmpd`) for Sun workstations. This schema file provides MIB II support with Sun enterprise-specific extensions. For more information about the `sun-snmp.schema`, refer to the `snmpd(8)` manual page.

Except for the two MIB II files (which differ only in the RPC number specified), each of the schema files listed above is a subset of the file that follows it. That is, `snmp.schema` is a subset of the two MIB II files, which are, in turn, are a subset of `sun-snmp.schema`.

The SNMP proxy agent can simultaneously access any of the above-mentioned schemas, as well as other enterprise-specific schemas that you might create. Both the Console and the SNMP proxy agent need to access the same schema information. If the SNMP proxy agent and the Console are running on different systems, identical schema files must be accessible to both systems. The Console uses the Schema Locations setting of the Console Properties Locations category

to locate the directories where the schema files reside. SNMP schema files on the Console are treated like any other schema files; they define groups, tables, attributes, etc., for the user interface. The SNMP proxy agent uses the keyword `na.snmp.schemas` in the `snm.conf` file to locate the directories where the SNMP schema files reside.

The following section describes in detail how the SNMP proxy agent works. Note that many of the operations of the proxy agent are defined by arguments passed in the request or with keywords in the `snm.conf` file on the proxy system. You should refer to the `snm.conf(5)` manual pages for the keywords that are related to the SNMP proxy agent.

When the SNMP proxy agent starts up (normally via `inetd`), it loads all the SNMP schemas located in the directories specified by the keyword `na.snmp.schemas` in the `snm.conf` file on its host system. Only SNMP-related schemas (schemas that contain an `rpcid` keyword value of '100122') are loaded.

When the SunNet Manager SNMP proxy agent receives a request for an SNMP agent on a particular device, it performs the following sequence of operations:

1. It checks whether there are any new or modified SNMP related schema files since the last request. If the proxy agent finds a new or modified schemas in any of the directories specified by the `na.snmp.schemas` keyword in the `snm.conf` file on the proxy's system, it loads the schema file.
2. It passes the request to an existing agent subprocess or forks a new subprocess, if needed, to handle the request. A single subprocess can handle multiple SNMP requests from an instance of a management application. The maximum number of subprocesses that the SNMP proxy agent can fork is set by the keyword `na.snmp.max-subprocs` in the `snm.conf` file. At installation, this value is set to 20. The maximum number of requests that a subprocess can handle is set by the keyword `na.snmp.max-requests` in the `snm.conf` file. At installation, this value is set to 50.
3. It checks whether the request contained any optional arguments. Requests sent by the SNM version 2.2.3 management applications may include arguments in an SNMP request. For example, the SNM 2.2.3 Console sends SNMP-related arguments that are defined in the target element's Properties window. (Refer to Section 2.2.9, "Properties," on page 2-34 for information about the element's Properties window.) These arguments can include:

- a. the name of the schema to be used with the request. If, for some reason, the specified schema does not contain the attribute group specified in the request, the proxy agent attempts to use the schema specified by the keyword `na.snmp.default-schema` in the `snm.conf` file on its host system. At installation, the default schema is set to be:
 - `/usr/snm/agents/snmp-mibII.schema` for Solaris 1.x installations
 - `/opt/SUNWconn/snm/agent/snmp-mibII.schema` for Solaris 2.x installations.

This schema supports the MIB II definition.

- b. a community name that specifies the SNMP community name the proxy agent is to use when reading or writing attribute values. If no community name is specified, `public` is used for both Get and Set requests.
 - c. a request timeout that specifies the number of seconds the proxy agent is to wait for a response to a request sent to the target system. If no request timeout is specified, the proxy agent uses the value specified by the keyword `na.snmp.request_timeout` in the `snm.conf` file on its system. At installation, the value is set to 5 (seconds).
4. This step only applies if the proxy agent receives a request from a management application that is not an SNM version 2.2 or later management application. If you are using the SNM version 2.2.3 proxy agent with the version 2.2.3 Console, skip to paragraph #5 below.

If no schema name is specified in the request, the proxy agent searches various schemas for the attribute group specified in the request. Schemas are searched in the following order:

- a. the schema(s) specified for the target device in the SNMP host file on the proxy system. The SNMP host file also specifies the community names and request timeout to be used for the request.

Note – Versions of the SNMP proxy agent prior to SunNet Manager 2.2 *required* that you define an SNMP host file so that the proxy agent could determine the correct schema file to use for each request. When used with the current (SNM 2.2.3) version of the Console or other management application, the 2.2.3 SNMP proxy agent does not *require* the SNMP host file to determine which schema file to use for each request. However, an SNMP host file will need to be defined in the following cases:

1. If you want to define the handling of traps on a per-device basis.
2. If the management application you are using with the 2.2.3 SNMP proxy

does *not* supply the schema name in a request.

The location of the SNMP host file is specified by the keyword `na.snmp.hostfile` in the `snm.conf` file on the proxy system. See the `na.snmp.hostfile(5)` manual page for more information about the contents of the SNMP host file.

- b. the schema specified by the keyword `na.snmp.default-schema`. If this schema is used, community names are assumed to be `public` and the request timeout is assumed to be 5.
- 5. The proxy agent then sends an SNMP message to the device and waits for a response.

If the proxy agent is sending a Get request, the proxy sends up to three SNMP requests per reporting interval. (The maximum number of SNMP requests sent is specified by the keyword `na.snmp.max_attempts` in the `snm.conf` file—by default the value is set to 3.) For each PDU sent, the proxy waits for the specified request timeout for a response from the device. As mentioned previously, the request timeout can be an optional argument in the request. If it is not specified in the request, request timeout is either the request timeout value specified in the SNMP host file for the device or the value of the keyword `na.snmp.request_timeout` in the `snm.conf` file.

If the proxy agent does not receive a response after sending three SNMP requests, it sends a “No response from system” report to the Console. (The keyword `na.snmp.trap-if-no-response` in the `snm.conf` on the proxy system determines whether the proxy agent sends a trap or an error report. At installation, the keyword’s value is `true`—send a trap report.) The proxy agent then waits until the next reporting interval to send out another set of SNMP requests. If no reporting interval has been specified in the request, the proxy agent sends out SNMP requests every 30 seconds. If the proxy agent does not receive a response when the last report is due, it sends both an error report and a trap report to the Console if `na.snmp.trap-if-no-response` is `true`.

If the proxy agent is sending a Set request, the proxy waits for the specified request timeout for a response before timing out. There is no attempt to re-send the request. The reason for this is as follows: Because UDP is the transport mechanism, there is no guarantee of message delivery, thus there is no way to determine whether the request or the response to the request

was lost. If you do not receive a response from your initial Set request, you should perform a Get request to see whether or not the Set operation was successful.

6. When the proxy agent receives a response from the target device, it sends a report to the SNM rendezvous.

If the proxy agent does not receive an acknowledgment from the rendezvous within a specified time, the proxy agent terminates the request. The specified time that the proxy waits for the rendezvous to acknowledge the report is specified by the `na.snmp.report_timeout` keyword in the `snm.conf` file. At installation, the keyword's value is set to 5 (seconds).

Normally, if the SNMP proxy agent is not performing any requests, it will exit. The keyword `na.snmp.exit-if-no-requests` in the `snm.conf` file allows you to specify otherwise.

Asynchronous or unexpected reports (traps) from SNMP agents are handled by the trap daemon `na.snmp-trap`, which may run on one or more machines on the network. The daemon listens for incoming traps on the SNMP trap port and translates them to SunNet Manager traps. The trap daemon uses an SNMP trap file, which contains information on enterprise-specific traps and attribution of events to pseudo-devices. Refer to Section 11.3, "Creating an SNMP Host File," on page 11-9 for more information about SNMP traps.

11.2 Creating a Schema File

If you do not already have a schema file for the device you want to manage, use the `SNM mib2schema` utility to convert an existing MIB file for the device.

Note – Nested groups or tables are *not* supported in SNM schema files.

You may need to manually edit the resulting schema file produced by `mib2schema`. The areas that are likely to require changes are:

- When `mib2schema` encounters an OCTET STRING, it inserts `-C ???` in place of a format string. If you want to format octet strings in a particular way, search the schema file for occurrences of `-C ???` to replace `???` with the required format string. If a format string is specified, the SNMP proxy agent formats each octet of the attribute value it receives from an SNMP

agent before sending the attribute value to a SunNet Manager rendezvous. You may, however, choose *not* to enter any format string. In this case, the contents of the OCTET STRING will be printed as is.

The format string is the same as the `sprintf(3S)` format argument. Up to 16 octets can be formatted; each byte is sent to `sprintf` as a separate, unsigned character. For example, the format string:

```
%02.2X:%02.2X:%02.2X:%02.2X:%02.2X:%02.2X
```

causes an OCTET STRING containing a 48-bit Ethernet address to be formatted in standard colon notation (for example, 08:00:20:07:8F:93).

Note – The format string and the length of the OCTET STRING to be formatted must match. All bytes specified in the format string are displayed. If the OCTET STRING is smaller than the format string, unexpected characters may be displayed in the formatted output.

Note that the `-C` format parameter is only used if the parameter `-T STRING` is specified for the attribute. If the parameter `-T STRING` is specified and `-C` format is not specified, the attribute is displayed as either octets or as a string, depending upon whether the attribute is an octet or display string.

An example of the characteristics string for the `ifPhysAddress` attribute in the `ifStatus` table is shown below:

```
"-N ifPhysAddress -O 1.3.6.1.2.1.2.1.6 -T STRING -A RO
-C %2.2X:%2.2X:%2.2X:%2.2X:%2.2X:%2.2X -X equal -F 0"
```

This results in the display:

```
ifPhysAddress=08:00:20:09:A0:D5
```

- Some SNMP devices cannot return groups or tables with a large number of attributes; this is due to local space limitations. When this happens, the SNMP proxy agent returns an error message that the response is “too big”. This means that very large groups or tables need to be split into smaller groups or tables to be received by the SNMP proxy. `mib2schema` does not

automatically split groups or tables. Generally, if a group has more than 15 fields, it is a good idea to split the fields up into smaller groups. You can choose your own name for subgroups.

In addition to the schema file, the `mib2schema` utility produces an object identifier file (with the `.oid` suffix) that contains a table of object identifiers and names. Run the `build_oid` utility to add the contents of the resulting object identifier file to the SNM Object Identifier Database. See the `build_oid(1)` manual page for more information.

`mib2schema` may also produce a trap definition file (with the `.traps` suffix), depending upon whether traps were specified in the MIB. Refer to Section 11.4, “Asynchronous Event Reports (Traps),” on page 11-11 for more information about using the trap definition file.

If `mib2schema` cannot determine the key for a table characteristics field in the schema file, it inserts `-K ???` into the schema file.

Note – SNM 2.2.3 does not *require* the `.oid` or `.trap` files.

11.3 Creating an SNMP Host File

The SNMP host file maps a request with a schema file. It also maps devices with trap files (refer to section Section 11.4, “Asynchronous Event Reports (Traps),” on page 11-11 for more information). Create an SNMP host file *only* for the following conditions:

- You want to specify the handling of traps on a per-device basis. Refer to “Asynchronous Event Reports (Traps)” on page 11-11 in this chapter for more information.
- You are using the 2.2.3 version of the SNMP proxy agent with an SNM 1.x version management application.
- You are using a 1.x version of the SNMP proxy agent.

Note – If you do not meet any of the above conditions, you do not need to create an SNMP host file.

The name of the SNMP host file is defined by the `na.snmp.hostfile` keyword in the `snm.conf` file. The default location for the hosts file is:

- `/var/adm/snm/snm.hosts` for Solaris 1.x environments
- `/var/opt/SUNWconn/snm/snm.hosts` for Solaris 2.x environments.

Each line in the SNMP host file specifies information for a single device. The line contains five to seven fields, defined below (the first five fields are mandatory):

<code><host-name></code>	<code><read-community></code>	<code><write-community></code>	<code><request-timeout></code>	<code><schema-file></code>
<code><trap-file></code>	<code><vendor-proxy></code>			

where:

`<host-name>`

is the name of the SNMP device. This name must match the element instance name (system name) of the device in a Console view. The name is not case sensitive.

`<read-community>`

is the SNMP community name the proxy agent uses when reading attribute values from `<host-name>` and when sending traps.

`<write-community>`

is the SNMP community name the proxy agent uses when writing attribute values to `<host-name>`.

`<request-timeout>`

is the number of seconds the proxy agent waits for an SNMP response from `<host-name>`.

`<schema-file>`

is a colon-separated list of the names of the agent schema files that contain the group and attribute definitions for `<host-name>`. Each name must be an absolute path name that begins with a 'slash' (/). Note that each schema file must reside in a directory that is specified by the keyword `na.snm.schemas` in the `snm.conf` file on the proxy system.

`<trap-file>` (optional)

is the name of the trap file that contains the trap definitions for `<host-name>`. This must be either an absolute path name that begins with a slash (/) or "-" to specify the default trap file. Note that the directory in which the trap file resides must be a directory that is specified by the keyword

na.snmp.schemas in the snm.conf file on the proxy system. Refer to Section 11.4, “Asynchronous Event Reports (Traps),” below, for more information about traps and the trap file.

<vendor-proxy> (optional)

is an optional field that specifies the name of a proxy system. If this field is set, the SNMP request is not sent to <host-name>, but will be sent instead to <vendor-proxy>. This parameter should be specified only when a vendor has supplied an SNMP proxy agent to manage a particular device or set of devices. In this situation, the vendor’s SNMP proxy agent communicates with the SunNet Manager SNMP proxy agent via SNMP, but communicates with the target device using either SNMP or a *different* protocol. If you specify this field, you must specify a value for <trap-file>.

An example of an SNMP host file with three device entries is shown below:

```
sunbox public private 5 /opt/SUNWconn/snm/agents/sun-snm.schema
device1 admin admin 10 /snm/vendor/device1.schema /snm/vendor/device1trap.schema
device2 public administrator 10 /snm/vendor/device2.schema - proximo
```

11.4 Asynchronous Event Reports (Traps)

SNMP agents may return unsolicited or unexpected reports called traps. SNMP defines six generic trap types for the following conditions:

- 0 coldStart
- 1 warmStart
- 2 linkDown
- 3 linkUp
- 4 authenticationFailure
- 5 egpNeighborLoss

In addition, trap type 6 is used for enterprise-specific traps. Enterprise-specific traps contain an enterprise object identifier (OID) and an enterprise-specific trap number.

11.4.1 Trap Daemon Operation

The SNMP trap daemon (`na.snmp-trap`) is installed with SunNet Manager agents and daemons, normally in the `agents` directory. The trap daemon translates received SNMP traps into SunNet Manager traps and forwards them to the Event Dispatcher on one or more management stations.

When the SunNet Manager SNMP trap daemon receives a trap, it performs the following sequence of operations:

1. It first determines what to do with the received trap. The SNMP trap daemon uses an SNMP trap file to carry out the following tasks:
 - translate trap type numbers to ASCII strings
 - determine if a trap should be discarded
 - determine the database element (glyph) to which a trap should be attributed (in cases where elements represent pseudo-devices — components that do not have their own IP address but share the network address of a device they are parts of, or attached to)

Refer to Section 11.4.2, “SNMP Trap File,” on page 11-14 for more information about creating trap files.

To locate the appropriate trap file, the trap daemon first checks the SNMP host file to see if there is an entry for the device. The SNMP host file allows you to map specific devices to trap files. Refer to Section 11.3, “Creating an SNMP Host File,” on page 11-9 for more information about the SNMP host file.

If the trap daemon cannot find a trap file for the device in the SNMP host file, it then searches a default trap file. The default trap file is specified by the `na.snmp-trap.default-trapfile` keyword in the `snm.conf` file on the system where the trap daemon is running. Normally, the file name is:

- `/var/adm/snm/snm.traps` for Solaris 1.x installations
- `/var/opt/SUNWconn/snm/snm.traps` for Solaris 2.x installations.

The default trap file allows you to specify the handling of traps on a per-enterprise basis.

If the trap daemon cannot find an entry for a trap type in a trap file defined in the SNMP host file *or* in the default trap file, it forwards the trap to the Event Dispatcher on the management station(s). If the trap daemon encounters any kind of syntax error while reading a trap file, it sends an error message indicating the line of the error and forwards the trap.

2. If the trap is not discarded, the trap daemon then searches through the loaded schema files for an OID match.

A trap may include a list of variables and their values that provide further information about the cause of the trap. Normally, the variable is in the form of a raw object identifier (OID). If the trap includes such variables, the SNMP trap daemon can translate the raw OIDs into ASCII text. If the variable is an enumerator data type, the trap daemon translates the variable's integer value into an equivalent ASCII string. The translation for each variable can be specified in a SNMP schema file that contains the OID.

When the trap daemon is started up, it loads the following files:

- a. schema files in the directories specified by the keyword `na.snmp.schemas` in the `snm.conf` file
- b. the schema file specified by the keyword `na.snmp.default-schema` in the `snm.conf` file. This file is normally `snmp-mibII.schema` in the `agents` directory.
- c. `snmp-mibII.schema` (in the `agents` directory), if it is *not* specified as the value of the keyword `na.snmp.default-schema`.

Note – The SNMP trap daemon does not load duplicate schema files. A particular OID can only be defined in one schema file.

If an OID match is found in a schema file, the trap daemon performs the translation before sending the trap to the Event Dispatcher. Since enumeration types for traps are not supported by SunNet Manager, the trap daemon translates enumerations to a string if it can find the enumeration definition in a schema file. Otherwise, the enumeration is returned as an integer. If the trap daemon is unable to find an OID match in the loaded schema files, the raw OID is forwarded to the Event Dispatcher.

Normally, if the trap daemon finds an OID match in a schema file, it only forwards translated attribute name/value pairs to the Event Dispatcher. The keyword `na.snmp-trap.raw` in the `snm.conf` file allows you to specify

that the SNMP trap daemon return raw OIDs and their values without translation, in addition to the translated attribute names and values—see the `snm.conf(5)` manual page for more information.

3. The trap daemon then forwards the trap to the Event Dispatcher. By default, the trap daemon `na.snmp-trap` sends trap reports to the Event Dispatcher on the machine on which the trap daemon is installed. The keyword `na.snmp-trap.rendez` in the `snm.conf` file allows you to specify the names of one or more manager systems that should receive the traps processed by the SNMP trap daemon — see the `snm.conf(5)` man page for more information.

11.4.2 SNMP Trap File

As mentioned previously, the SNMP trap daemon uses a trap file to:

- translate enterprise-specific trap type numbers to ASCII strings
- determine if a trap should be discarded
- determine which database element (glyph) a trap should be attributed to (in cases where elements represent pseudo-devices — components that do not have their own IP address but share the network address of a device they are parts of, or attached to)

If you used the `mib2schema` utility to create your schema file and traps were specified in the MIB, `mib2schema` generates a trap file. Otherwise, you need to create a trap file for any of the above-mentioned purposes.

There are two types of trap file entry which are differentiated by a keyword that begins the entry in the file:

- Entries that define traps on a per-enterprise basis. Each such entry begins with the keyword `enterprise`
- Entries that associate traps with database elements (glyphs). Each such entry begins with the keyword `glyph`.

Note – If you use `glyph` entries in the trap file, the associated SNMP trap daemon must run on the host which has the SNM database where these elements reside. The trap daemon cannot be distributed to machines where the SNM Console is not installed because the trap daemon must access the SNM database to retrieve the properties of the specified elements.

11.4.2.1 Trap File Entries for Enterprise-Specific Traps

The trap file may contain one or more lists of generic or enterprise-specific traps. Each list begins with a line containing the keyword `enterprise` followed by the enterprise object identifier. Subsequent lines contain a trap number, trap name, and, optionally, the keyword `discard`. Comments may be added by beginning a line with a pound sign (`#`). The syntax for the trap file is shown below:

```
enterprise <enterprise-object-ID>  
<trap-number>      <trap-name>      [discard]
```

<enterprise-object-ID>

is the enterprise object identifier. If you want to discard generic traps, specify zero for the *<enterprise-object-ID>*.

<trap-number>

is an integer that represents the trap number of an enterprise-specific trap. If you want to discard a generic trap, specify the trap type number: 0, 1, 2, 3, 4, or 5.

<trap-name>

is the ASCII string associated with the *<trap-number>*. *<trap-name>* can be up to 127 characters in length; blanks are not allowed.

The keyword `discard` specifies that the trap daemon should not forward the trap to the Event Dispatcher. As mentioned previously, traps are treated as high priority events. You can specify the `discard` keyword for traps that you do not wish to be notified about, such as system reboots.

For example, the following lines define two enterprise-specific traps for the Sun enterprise and specify the discarding of generic trap type 0:

```
# generic traps
enterprise 0
    0      cold-start      discard

# Sun Microsystems traps
enterprise 1.3.6.1.4.1.42
    1      CPU_Failure
    2      Power_Supply_Failure
    3      Network_Connection_Failure
    4      Over_Heating      discard
    5      RealTimeClock_Failure      discard
```

When searching the SNMP trap file for matches, the SNMP trap daemon only compares as much of the enterprise identifier as specified in the SNMP trap file. For example, a file with the line:

```
enterprise 1.3
```

would match *all* enterprises beginning with 1.3. Since the searching is performed from the top of the file, you should place more restrictive enterprise identifiers before less restrictive ones.

If you define a single trap file that applies to all or most of your SNMP devices, you should make this file the default trap file. For example, suppose that you do not want to see any generic traps for most of the SNMP devices in your management domain. You would define a trap file that specifies that generic trap types 0 through 5 be discarded and make this the default trap file. If there are a couple of critical devices for which you want to see generic trap types, define another trap file that does not specify that the generic trap types be discarded. Create an entry for each device in the SNMP host file and specify the second trap file. Because the trap daemon searches the SNMP host file first to match a trap to a device name, generic traps will be returned for only those devices specified in the SNMP host file.

You create the SNMP host file on the system where the trap daemon resides. Refer to Section 11.3, “Creating an SNMP Host File,” on page 11-9 for information on creating the host file. If you have only one trap file that applies to all your SNMP devices, you can specify the trap file as the default trap file—you would not need to create an SNMP host file.

11.4.2.2 *Attributing Traps to Glyphs*

Glyph entries in the trap file allow the user to attribute traps to database elements that represent pseudo-devices — network components that do not have their own IP address but share the network address of a device that they are components of, or to which they are attached.

An example would be glyphs that represent the ports on a hub, as shown in Figure 11-3. In this case the trap-attribution feature would allow the SNMP trap daemon to determine which port is the source of the event. When a trap is generated (for example, due to a hub port failure), the appropriate Console effect — such as blinking or color by priority — would then be applied to the glyph for that port rather than to the hub as a whole, facilitating fault-isolation.

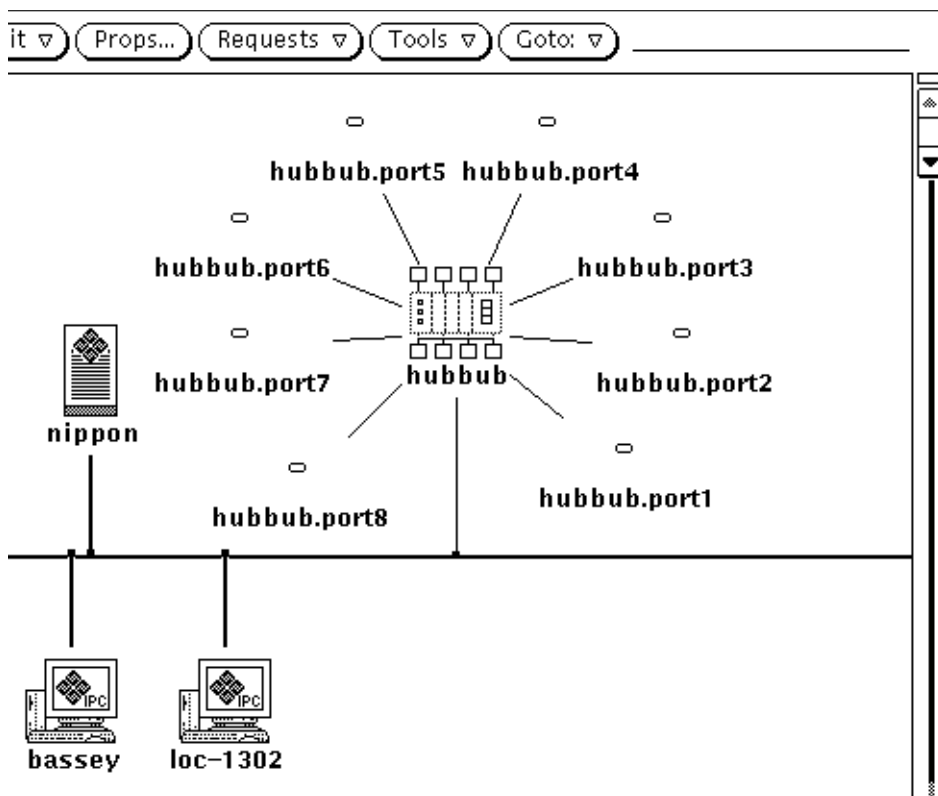


Figure 11-3 Example of Glyphs for Pseudo-devices

If you have vendor software whose MIB supports this trap-distribution feature, there are five tasks that must be accomplished to enable it for your local SNM Console:

- The agent schema file that supports the trap-distribution feature — either provided by the vendor or created from the vendor MIB using the `mib2schema` utility — must be installed in the `agents` directory.
- An appropriate `glyph` entry must be added to an SNMP trap file on the system where the SNMP trap daemon runs (and this must be the system on which the SNM Console runs).
- The `snmp.hosts` file must also exist on the system where the trap daemon runs and must contain an entry that identifies the SNMP trap file to be used for interpreting traps from the originating device.
- An element definition must be provided for the pseudo-device in the `elements.schema` file (or a separate element schema file in the `agents` directory) and this definition must contain a field used to correlate this pseudo-device with a particular glyph. This field must match the appropriate field used by the enterprise-specific agent schema.
- An icon and iconmask to represent the pseudo-device must exist in one of the directories specified for icon locations in the Console properties sheet.

For information on creating an agent schema from a vendor MIB, refer to Section 11.2, “Creating a Schema File.” How to point the SNM Console to the location of the icon and iconmask is discussed in Section 9.7, “Locations.”

The other tasks can be accomplished as follows:

1. Trap File Glyph Entries

The glyph entry in the trap file must begin with the keyword **glyph** and have the following syntax:

```
glyph <trap_attribute> <database_user_name> <component_type> <component_property>
```

<trap_attribute>

is the attribute whose value will be matched against the value of the property in the component property sheet to determine the glyph to be associated with the trap.

<database_user_name>

is the name of the runtime SNM database without the “db.” prefix. The user name must be provided because the trap daemon could have been started by the `inetd` process; in that case, the user owning the trap daemon will not be the same as the user running the Console. This field is used to create the `SNM_NAME` environment variable required to access the the SNM database.

<component_type>

is the type of the component whose property is to be compared (for example, `component.pseudo`).

<component_property>

is the property of the component whose value will be compared against the trap attribute value.

A sample glyph entry would be the following:

glyph	ifPortKey	alfred	component.hubport	PortKey
-------	-----------	--------	-------------------	---------

In this example, `ifPortKey` is in the variable binding list of the trap Protocol Data Unit (PDU). To determine if this trap should be attributed to a particular glyph, the trap daemon is to look at the value of the property `PortKey` for an element. The trap is to be attributed to that glyph if the glyph’s `PortKey` value matches the value passed in the trap’s `ifPortKey` variable. `alfred` is the name of the SNM database user and `component.hubport` is the element type.

2. SNMP Host File Entry

An entry must be added to the SNMP hosts file only if a schema file other than the trap daemon’s default SNMP schema file is to be used to enable the trap-distribution feature. The hosts file resides in the same directory as the trap file. The syntax of host file entries is as follows:

<host-name>	<read-community>	<write-community>	<request-timeout>	<schema-file>
<trap-file>	<vendor-proxy>			

For other types of host file entry, **<host-name>** is the name of the SNMP device. However, for the glyph-distribution feature, **<host-name>** is to be the name of the host where the proxy agent resides. In a standard host file

entry, the proxy system name is specified in the *<vendor-proxy>* field. However, for purposes of the pseudo-device trap-attribution feature, the optional *<vendor-proxy>* field is never used since this information is specified in the *<host-name>* field. For more information about the SNMP host file, refer to Section 11.3, “Creating an SNMP Host File.” An example of a host file entry to implement the trap-distribution feature would be the following:

```
bigguy    public  private  20 /opt/SUNWconn/snm/agents/snm-  
pseudodev.schema /var/opt/SUNWconn/snm/snm.traps
```

In this example, “bigguy” is the name of the host where the SNMP proxy agent resides. `/opt/SUNWconn/snm/agents/snm-pseudodev.schema` is the vendor agent schema; the only reason the host file is necessary is to point to this file.

3. Adding an Element Definition for the Pseudo-device

You will need to add a component definition for the glyph that represents the pseudo-device. The component definition is created in the form of a record definition as in the following example:

```
record component.hubport (                                # generic sun 4
    string[64]      Name
    string[40]      User
    string[40]      Location
    string[80]      Description
    netaddress      Port_Key
    string[40]      SNMP_RdCommunity
    string[40]      SNMP_WrCommunity
    string[64]      SNMP_Vendor_Proxy
    int             SNMP_Timeout
)

# Glyphs for components

instance elementGlyph(
    ( component.hubport                                hubport.icon)
)
```

The fields in this element definition will be displayed in the properties sheet for glyphs of this type, as shown in Figure 11-4.

SunNet Manager Console: hubbub.port1 (component.hubport)

Name: hubbub.port1

User: Jak Netman

Location: Room 232

Description: port 1 on hub

Port Key: 10.12.8.1

SNMP RdCommunity: public

SNMP WrCommunity: private

<input type="checkbox"/> hostperf	host performance data
<input type="checkbox"/> iostat	I/O statistics (SunOS 4.x)
<input type="checkbox"/> iostat2	I/O statistics (Solaris 2.x)
<input type="checkbox"/> ippath	IP path information
<input type="checkbox"/> iproutes	IP route table and statistics
<input type="checkbox"/> layers	protocol layer statistics
<input type="checkbox"/> layers2	protocol layer statistics (for SunOS 5)
<input type="checkbox"/> lpstat	line printer status and queue info
<input type="checkbox"/> ping	IP connectivity info
<input type="checkbox"/> rpcnfs	RPC and NFS stats

Red: 0

Figure 11-4 Sample Properties Sheet for Pseudo-Device

Note that underscores in the element definition are displayed as spaces in the field names. In this example, the Port Key field gives the attribute — a user-defined port number — that is used to match traps to glyphs. (This “Port Key” value is in the same format as an IP address though it is not an IP address.) For information on how to create element definitions, see Section 10.2, “Element Instance Definition.” This definition can be added to the `elements.schema` file or to another element schema file in the `agents` directory.

This component definition must include a field that will have the same value as the SNMP trap attribute that will be matched with it. The following restrictions apply:

- The trap attribute to be matched with the component property must be in the variable binding list in the trap PDU. This follows the regular members of a trap message, which are the following:
 - version
 - community
 - enterprise
 - agent address
 - generic trap type
 - specific trape type for enterprise-specific trap
 - time stamp
- For purposes of matching the element property to the SNMP attribute, only the SNMP types listed in Table 11-1 are supported.

Table 11-1 Supported SNMP and Schema Types

SNMP Type	SNMP Attribute Type	Element Schema Attribute Type
INTEGER	int	int
INTEGER	enum	enum
OCTET STRING	string	string
IPAddress	netaddress	netaddress
NetworkAddress	netaddress	netaddress
OBJECT IDENTIFIER	objectid	string

11.4.3 Trap Reports on the Console

On a Console system, trap reports are logged with other event and error reports and are displayed by selecting the Console's View►Event/Trap Reports menu item. See Section 4.2, "Event Reports," on page 4-6," for more information about viewing event/trap reports from the Console.

Changes in a glyph's state that results from a trap are propagated through the Console's view hierarchy as if an event had occurred. SNMP traps are treated as high-priority events. The type of signal that the Console uses to indicate received traps is specified in the Events and Traps category of the Console's Properties window. By default, the Console blinks the glyph of the target element when a trap is received.

11.4.4 Interpreting a Trap Report

A typical trap report logged in the Event/Trap Reports window looks like the following:

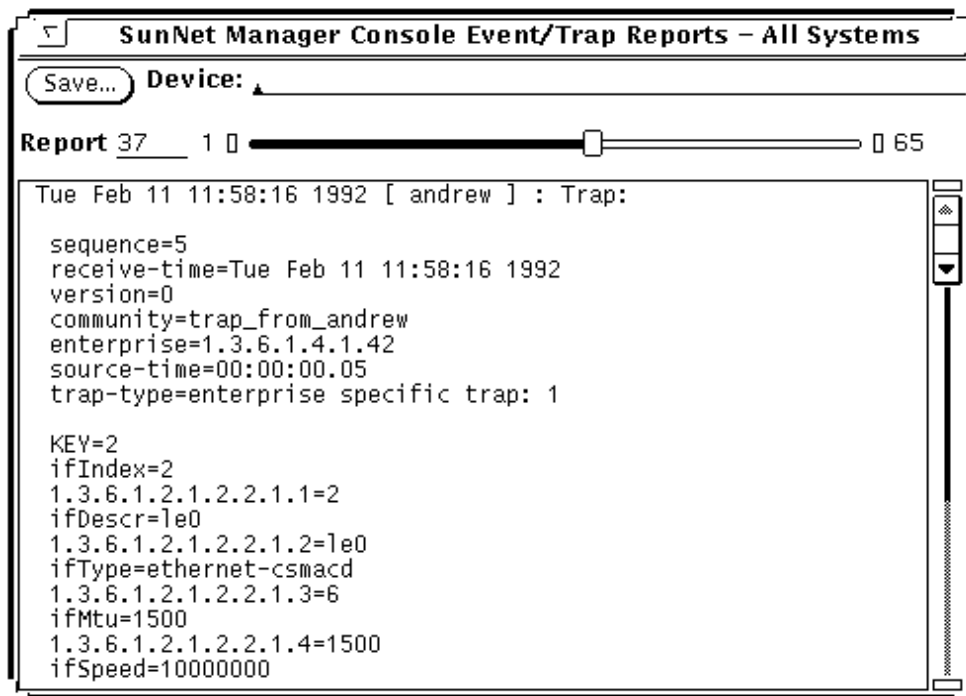


Figure 11-5 Trap Report

The following describes the lines in the trap report:

At the top of the display is the header line giving the data, time and hostname of the device which generated the trap.

sequence

is the sequence number of the trap. This counter keeps track of the number of traps generated since the device was last booted.

receive-time

is the time when the trap was received by the Console.

version

is returned by the device which generated the trap.

community

is the SNMP trap community name sent by the device which generated the trap.

enterprise

is the enterprise name of the device which generated the trap.

source-time

is the length of time between when the device was booted to when the trap was generated.

trap-type

is the type name of the trap generated. In this example, it is warmStart.

The remaining fields, if any, are variables and their values returned by the trap. Some traps do not return any variables. The values of these variables are usually relevant to the reason why the trap was generated. When the trap daemon receives a trap from any device, it will attempt to translate these variables from OIDs into ASCII text before forwarding it. See Section 11.4.1, “Trap Daemon Operation,” on page 11-12, for more information.

11.5 Mapping SNMP Object IDs to Strings

One of the data types supported by SunNet Manager is an object identifier. Since object identifiers are numeric (for example, 1.2.3.4.5), a database is provided so the Console can display the object identifier using a more meaningful string. The Object Identifier Database provides the information required for this mapping. The `build_oid` utility builds the Object Identifier Database. The Database is used by the Console to interpret object identifiers.

`build_oid` should be run anytime the Object Identifier Database needs to be built. Such instances include installation of the Console (when `build_oid` is run automatically by the installation script) and whenever new object identifiers need to be added to the database.

Input files for `build_oid` are ASCII files with the suffix `.oid`. Each file consists of lines containing a descriptive name followed by its object identifier. If the descriptive name contains special characters, it must be enclosed in

double quotes. Comment lines may be included by beginning them with a pound sign (#). When `mib2schema` is used to convert a MIB to a schema file, it also generates a `.oid` file.

To locate input files, `build_oid` uses the directories specified by `SNMHOME/agents` and `SNMHOME/schemas`. If you have not defined the `SNMHOME` environment variable, these directories are:

- `/usr/snm/agents` and `/usr/snm/struct` for Solaris 1.x installations
- `/opt/SUNWconn/snm/agents` and `/opt/SUNWconn/snm/struct` for Solaris 2.x installations

You can also specify other directories as input arguments—see the `build_oid(1)` manual page for more information. There are three input files supplied with the SunNet Manager product, and normally found in the `/usr/snm/agents` directory:

- `enterprises.oid` contains the mappings for a variety of enterprise and organization identifiers.
- `snmp.oid` contains the mappings for object identifiers for MIB I of the SNMP protocol.
- `sun-snmp.oid` contains the mappings for object identifiers referenced by the Sun SNMP agent, `snmpd`. The Sun agent implements a superset of MIB II, so this file also serves as a `snmp-mibII.oid` file.

The environment variable `SNMDBDIR` specifies the directory where the output database `oid.dbase` should be written. If you have not specified the `SNMDBDIR` environment variable, then

- `/var/adm/snm` directory is assumed for Solaris 1.x installations
- `/var/opt/SUNWconn/snm` directory is assumed for Solaris 2.x installations.

SNMP Version 2 Support

<i>SNMP Version 2 Support</i>	<i>page 12-1</i>
<i>SNMPv2Files</i>	<i>page 12-4</i>
<i>Using the v2mib2schema Program</i>	<i>page 12-4</i>

12.1 SNMP Version 2 Support

This chapter assumes that you are familiar with SNMP and SNMPv2 concepts. If you will be creating your own enterprise-specific schemas, you should also understand SNMP group, table, and attribute definitions. Refer to Chapter 11, “SNMP Support,” for detailed information pertaining to SNMP.

Note – Instructions for installing and de-installing SNMPv2 can be found in Chapter 5 of the *SunNet Manager 2.2.3 Installation Guide*.

SunNet Manager provides a proxy agent that supports the Simple Network Management Protocol Version 2 (SNMPv2). The SNMPv2 proxy agent allows you to get data and event information from and set attribute values for devices that are managed via SNMPv2.

Note – SunNet Manager 2.2.3 also provides an SNMP agent for Sun workstations called the `snmpv2d` daemon. The SunNet Manager 2.2.3 Console communicates with the `snmpv2d` daemon through the SNMP proxy agent. The

`snmpv2d` daemon also allows Sun workstations to be managed by other SNMPv2 and SNMP stations. For more information about the `snmpv2d` daemon, see the `snmpv2d` (8) manual page.

The following sections discuss the differences between SNMP and SNMPv2. For details on using the `build_oid` utility and creating new schema files, see Chapter 11, “SNMP Support,” for detailed information. For information pertaining to the SNMPv2 configuration files, see the following manual pages: `v2install`(1), `acl.pty`(5), `agt.pty`(5), `context.pty`(5), `mgr.cnf`(5), `mgr.pty`(5), `snmpv2d.conf`(5), and `view.pty`(5).

Note – This release of SNMPv2 does not contain a full implementation of SNMPv2 as described in RFCs 1441-1452. Items which are not included are noted below.

Note – When the Discover Tool is used to locate SNMP devices on your network, it cannot determine whether these devices support functionality specific to SNMPv2.

12.1.1 *SNMPv2 Enhancements*

The key enhancements from SNMP to SNMPv2 are in the following categories:

- Structure of Management Information (SMI)
- Protocol operations
- Manager-to-manager capability
- Security

12.1.1.1 *Structure of Management Information*

The SMI for SNMPv2 is based on the SMI for SNMP. The SNMPv2 SMI provides more extensive specification and documentation of managed objects and MIBs.

Several new data types were created for SNMPv2. These include a 64 bit-counter (`Counter64`) and the `UInteger32` type which allows you to represent integers in the range 0 to $2^{32} - 1$.

The SNMPv2 `OBJECT-TYPE` macro includes an optional `UNITS` clause, which contains a textual definition of the units associated with an object. This clause is useful for any object that represents a measurement in some kind of units (ex. “seconds”). The `OBJECT-TYPE` macro for SNMPv2 also includes a `MAX-ACCESS` clause which allows you to specify the maximal level of access allowed.

SNMPv2 also enhances conventions to facilitate row creation, deletion, and access.

12.1.1.2 Protocol Operations

Three new protocol data units (PDUs) were added in SNMPv2. The SNMPv2 trap PDU works much the same way as the SNMP trap PDU, but it uses the same format as most other SNMPv2 PDUs. This eases the processing task of the receiver.

One of the major enhancements in SNMPv2 is the `GetBulkRequest` PDU. This PDU can significantly minimize the number of protocol exchanges required to retrieve a large amount of management information.

The third additional PDU is the `InformRequest` PDU. This is sent by an SNMPv2 manager, on behalf of an application, to another SNMPv2 manager, providing management information to an application using the latter manager.

12.1.1.3 Manager-to-Manager Capability

Manager-to-manager operations are supported through the use of the manager-to-manager MIB. It consists of a set of objects which describe the behavior of an SNMPv2 entity that acts in a manager role. For more information on this MIB see RFC 1451. This capability is currently not fully supported in the SunNet Manager2.2.3 product.

12.1.1.4 Security

SNMPv2 uses the party concept for security which was developed in S-SNMP (Secure SNMP). The most significant improvements from S-SNMP are the elimination of the ordered delivery mechanism and the simplification of the clock synchronization algorithm. In addition, SNMPv2 introduces the *context* concept. The use of contexts provides for more efficient storage of access

control and MIB view information. SNMPv2 uses both DES and MD5 for message security and authentication. Due to export restrictions, Data-Encryption Standard (DES) and Message Digest 5 (MD5) are not currently being used in this release.

12.2 *SNMPv2Files*

SNMPv2 can be installed as an agent (`snmpv2d`), a manager (`na.snmpv2`), or both. The files necessary for SNMPv2 installation are installed as part of the SunNet Manager 2.2.3 product. The installation steps are the same for both agents and managers. Before installing SNMPv2, you must first install the SunNet Manager 2.2.3 product (see the *SunNet Manager 2.2.3 Installation Guide* for details). After the SNM 2.2.3 product has been successfully installed, you must create the three configuration files needed by the `v2install` script. These configuration files are:

- `agents` - contains the names of the hosts on which the `snmpv2d` agent will be installed.
- `mgrs.v1` - contains the names of the hosts that will be running SNMPv1 managers (`na.snmp`).
- `mgrs.v2` - contains the names of the hosts that will be running SNMPv2 managers (`na.snmpv2`).

See the `v2install(1)` manual page for detailed information about these files.

Procedures for installing (or removing) SNMPv2 software are to be found in the *SunNet Manager 2.2.3 Installation Guide*.

12.3 *Using the v2mib2schema Program*

A program, `v2mib2schema`, has been included with SunNet Manager 2.2.3 to allow users to translate their own SNMPv2 MIBs to SNM schema files.

It is very important to note that SunNet Manager schemas do not have the flexibility of SNMPv2 MIBs, so changes to the MIB may be necessary before `v2mib2schema` can successfully parse it.

Although `v2mib2schema` parses `TEXTUAL-CONVENTIONS` clauses, it currently ignores them, so later references to the newly defined types will result in syntax errors. See the `v2mib2schema (5)` man page for more details.

<i>Authentication</i>	<i>page 13-2</i>
<i>Access Control</i>	<i>page 13-2</i>
<i>The Security Algorithm</i>	<i>page 13-3</i>
<i>Conferring Right-of-Access</i>	<i>page 13-4</i>

SunNet Manager provides an optional security mechanism for restricting access to agent services. As delivered, any manager may request data and/or event reporting from any SunNet Manager agent. For each agent, you can specify read-security for data and event report requests and write security for Set requests. This section describes how to customize security for your SunNet Manager environment.

SunNet Manager security uses and builds upon the “secure networking features” introduced in SunOS 4.x. These security features are based on DES encryption and public key cryptography. If you are installing SNM 2.2.3 on Solaris 1.x, we recommend that you first read the “Secure Networking” chapter in the *Security Features Guide* before reading the rest of this chapter. If you are installing SNM 2.2.3 for Solaris 2.x, you should first read *SunOS 5.2 Administering Security, Performance, and Accounting* before continuing with this chapter.

SunNet Manager implements security by giving an agent the option to authenticate each network management request before acting on it. The two elements of security are:

- Authentication—is the requestor really who he claims to be?
- Access control—does the requestor have sufficient right-of-access to make this request?

13.1 Authentication

SunNet Manager uses the RPC protocol DES authentication feature to prove the identity of the user making the request. This guarantees that the request is coming from a genuine source and makes it computationally infeasible for anyone to impersonate that source.

13.2 Access Control

Once the requestor's identity is authenticated, SunNet Manager verifies that the requestor has the required right-of-access. SunNet Manager confers right-of-access based on the "netgroups" feature of Sun OS 4.x, which was carried forward to SunOS 5.x (Solaris 2.x). Consult the following:

- the SunOS 5.2 manual *Administering TCP/IP and UUCP* (for Solaris 2.x installations),
- the netgroup(5) man page, and
- "Setting up Network Software" in the *System and Network Administration Guide* (for Solaris 1.1 installations).

Access rights are administered by granting membership in one of the five network security groups. These groups are named:

- netmgt_security_one
- netmgt_security_two
- netmgt_security_three
- netmgt_security_four
- netmgt_security_five

Access control for members of this group works as follows:

The Agent Library has a global variable, Network Management Security Level, that can have a value between 0 and 5. The value 5 is the most secure and will allow only members of the network group netmgt_security_five access to network management functions served by this agent library. All other requestors will be refused. The value 4 will allow access to members of groups

`netmgt_security_four` and `netmgt_security_five` and so on. Lower values of the Network Management Security Level variable allow access to members of more and more network groups and the value 0 provides no access control at all.

For the security scheme to work, both the manager's and the agent's machine (and their respective NIS/NIS+ master server machines) must be running under SunOS 4.0 or later, and the `keyserver` daemon must have been started at boot-time in one of the following ways:

- by the `/etc/rc2.d/S71rpc` script, if the installation is in a Solaris 2.x environment
- from the `/etc/rc.local` file, if installed in a SunOS 4.x environment.

13.3 The Security Algorithm

The following algorithm is used by the agent library to verify the authenticity and the access rights associated with every incoming request.

1. Since all security and authentication depends upon NIS/NIS+, first check if NIS/NIS+ is up and running. If NIS/NIS+ is down, reject the request indicating that NIS/NIS+ is not running.
2. Check Network Management Security Level to ascertain the level of security being requested. If this value is zero, bypass the following checks and allow this request through without authentication.

Note – UNIX-style authentication is not supported.

3. Verify that the request has a DES style authenticator. If not, the request is refused, indicating too weak an authentication.
4. Verify that a domain specific user-id and password table entry exists for the network name supplied in the DES authenticator contained in the request. If not, the request is refused.
5. Finally, confirm membership of the appropriate network groups based on the value of Network Management Security Level (for a value of 1, check groups 1 through 5; for a value of 2, check groups 2 through 5; etc). Deny the request if membership is not confirmed.

If the request passes the above tests, it is considered successfully authenticated, and processing of the request proceeds as normal.

13.4 *Conferring Right-of-Access*

Now that you understand how security works, let's look at how to confer right-of-access for your agent.

1. Create up to five network groups for the network named `netmgt_security_one`, `netmgt_security_two`, `netmgt_security_three`, `netmgt_security_four`, and `netmgt_security_five`.
2. Include in these groups the user-names of the system administrators who have permission to run SunNet Manager. This grouping defines a hierarchy of abilities for the administrators. Those administrators who are members of the `netmgt_security_five` group have maximum privilege and can send requests to any agent. Members of the `netmgt_security_four` group can send requests to agents with security level 4, 3, 2, 1 or 0. Members of the last group, `netmgt_security_one`, can send requests to agents with security level 1 or 0.
3. For each administrator, create a new public key for the administrator's user-name using `newkey(8)`.
4. For each host on which you have agents installed, create a new public key for the super-user at the host using `newkey(8)`.
5. Decide the level of read-security and write-security you wish to assign to your agents and set the security levels associated with the agent name in the SunNet Manager configuration file `snm.conf` to values between 0 (no security) and 5 (maximum security) on each system where agents are installed.

An example entry is shown below:

<code>na.snmp</code>	<code>2</code>	<code>4</code>
----------------------	----------------	----------------

This sets the SNMP proxy agent read–security level to 2 and write–security level to 4. A user must be in at least the `netmgt_security_two` network group to request SNMP data or event reports and in at least the `netmgt_security_four` group to invoke SNMP Set requests.

If the `snm.conf` file does not contain an entry for an agent, the agent’s read *and* write security levels default to 0—no security checking. If the entry for an agent contains only one number, the agent’s read *and* write security levels are set to that number.

Error Messages



<i>If Agents or Daemons Don't Start</i>	<i>page A-2</i>
<i>NetISAM Messages</i>	<i>page A-3</i>
<i>Types of SNM Errors</i>	<i>page A-4</i>
<i>SNM Error Messages</i>	<i>page A-5</i>

Error messages can be generated from many different pieces of the SunNet Manager package. Error messages can be sent to a number of places: to `stdout`, `stderr`, the system console, the system's `messages` file, and in the case of the Console, they may go to the Console's Error Reports window or the Console's window footer.

This appendix discusses:

- Methods for getting agents to start.
- NetISAM error messages.
- Errors that can be generated by the SunNet Manager services and Console (in alphabetical order).

Note – For descriptions of errors specific to agents, see the `man` page for each agent.

A.1 *If Agents or Daemons Don't Start*

Occasionally you may find that an agent does not start when you send a request to it. It could be due to a remote procedure call error, listed in Section A.4, “SNM Error Messages,” on page A-5. If you notice the following message in the `messages` file or the system console log:

```
<host> inetd[nnn]: <agent>/rpc/udp server failing (looping), service terminated
```

it's because every time `inetd(8c)` starts the agent, the agent dies immediately. `inetd(8c)` tries a number of times to start the agent, and gives up when it's clear the agent doesn't want to be started. In this case, start the agent manually with debugging enabled by entering,

```
host# <agents-path>/agents/na.<agent> -d 3
```

where `<agents-path>` is typically `/usr/snm` for Solaris 1.1 installations or `/opt/SUNWconn/snm` for Solaris 2.x installations.

You then should get some indication of why the agent is failing. Maybe it can't find `libnetmgt.so`. Maybe `ldconfig(8)` needs to be run to get the runtime linker-editor to know about the shared library. It's possible the agent is failing for some other reason. Maybe some resource it needs isn't available, and it is unable to send an error message back to the manager. Once you know what the error means, you should stop the agent (if it hasn't stopped itself), correct the error, and restart `inetd(8c)`, rather than sending it a HUP signal. The commands to do this are shown below:

On Solaris 1.x:

```
host# ps -ax | grep inetd
host# kill <processid>
host# /usr/etc/inetd
```

On Solaris 2.x:

```
host# ps -ef | grep inetd
host# kill <processid>
host# /usr/sbin/inetd
```

If an agent dumps core, the core file is stored in the `/usr/tmp` directory. If you cannot resolve the problem with the agent, send a copy of the core file in with your problem report.

The above commands can also be used to stop and restart the SNM activity daemon. You should do this if you see the following message:

```
Cannot contact na.activity to determine ID
```

A.2 NetISAM Messages

If the SNM Console fails to start or crashes, you may see messages that are returned by NetISAM. These messages have the following format:

```
netisam[xxx]:<error>:unix errno <n>
```

where `errno <n>` messages can be found in the header file `errno.h`.

The following is a list of the more common NetISAM message numbers and descriptions of the problem.

Table A-1 Common NetISAM Errors

errno	Description
2	There is no such file or directory. The database directory specified by the SNMDBDIR environment variable does not exist. Or, the runtime MDB files <code>nc.rec</code> , <code>nc.ind</code> , <code>events.rec</code> , and <code>events.ind</code> (normally in <code>./db.<username></code> directory) do not exist.
12	There is not enough memory. You have run out of virtual memory and need to increase the amount of swap space.
13	You do not have the proper read/write access permission to the database directory specified by the SNMDBDIR environment variable or to the database files in the MDB directory (normally in <code>./db.<username></code>).
28	The file system on which the database exists is full.
30	The database directory that you are trying to write to is a read-only file system.
70	The database you are using is on an NFS-mounted file system and the machine from which you are mounting may have been rebooted. Reboot your machine if this is the case.
71	The database is on a file system that is mounted from more than one host.

A.3 Types of SNM Errors

Error messages fall into the following categories:

Information Only:

For user information only. No action is required.

Warning:

If possible, the application will recover as described.

Error:

Remedial action from the operator is required to correct the problem.

API Error:

The Manager/Agent Services library detected an error caused by the application program. These are often due to programming errors in the application.

Internal Error:

An unexpected error occurred. It's possible the Console's runtime database may have become corrupted. If you suspect that is the cause of the error, try saving the runtime database to the MDB—use the File button's Save►Management Database option— and correct the invalid entry. After the correction is made, reinitialize the runtime database specifying the `-i` flag in the `snm` command.

It's also possible this internal error does *not* indicate an error in the Console's runtime database. It could be an internal error generated by another application or by the Manager/Services library. If an internal error persists, contact the appropriate Sun service entity and provide the error message, circumstances that caused the error, and any relevant files.

A.4 SNM Error Messages

The actual error message text may be prefixed with “Error:” or the name of the application that is returning the error (e.g. “snm:”). The error messages in this section are generally listed in alphabetical order according to the text of the error message.

Abnormal termination

Internal Error: The SNMP trap daemon has encountered a fatal error and has terminated.

Agent error:<why>

Warning/Error: An agent returned an agent-specific error. Refer to the agent documentation for more information. (You can determine the agent from the request name, which is also returned. The default request name is in the format `<agent>.<group>.<number>`.)

Agent has over <number> groups

Warning: An agent has too many groups or tables. Not all groups and tables will be displayed in the menus.

`(alloc_buffer):error, no more space`

Error: The Console could not write a message to the data or event log because it ran out of virtual memory. Either increase the amount of swap space on the system or decrease the number of applications running concurrently.

`(API):request failed, <why>`

Error: A request failed for the reason given. The request will not be retried.

`(API):request failed, <why>, retrying`

Error: A request failed for the reason given. The request will be retried.

`Argument name is too big`

API Error: The specified name string is too long for the name buffer.

`Attempt(s) in which an element was no longer in the MDB`

Internal Error: A race condition in the database has occurred.

`Attempt(s) to Drop or Paste a request unknown to an element`

Error: The target element is not associated with an agent of the type specified in the request being dropped or pasted.

`Attempt(s) to Drop or Paste into a request`

Error: An attempt was made to put elements into the subview of a request. Requests have no subviews.

`Attempt(s) to Drop or Paste to an element accepting no requests`

Error: The target element has no associated agents and therefore accepts no requests.

`Attempt(s) to inappropriately Drop or Paste into Home`

Error: An attempt was made to put a request into the Home view.

Attribute not supported, not using SNMP for auto-management.

Warning: An SNMP autorequest resulted in an “attribute not supported” error.

Attribute type does not match: group ‘(<group>)’ attribute ‘(<attr>)’: schema type ‘(<type1>)’, returned type ‘(<type2>)’

Error: The agent and schema disagree on the attribute’s data type.

Attribute unavailable for set operation: variable #*n*

Error: The set request was not performed because the *n*th attribute to be set was not available for set operations. This variable cannot be set.

Attribute <agent>/<group>/<attribute> is not writable

Error: The attribute does not have write permission, it is not set-able.

Bad index <index>

Internal Error: An invalid runtime database index was passed to a database routine.

Bad rasterfile:<file>

Error: The raster file <file> could not be loaded because its format is not that of a raster file.

Bad security credentials

Error: The requester’s security credentials are bad. Ask your system administrator to add you to the appropriate security netgroup. Refer to Chapter 13, “Network Management Security,” for more information about security.

Cannot acquire database lock, try again later

Error: Check to make sure that the database directory is writable.

Cannot add agent to system:<name>

Warning: Failure adding an agent to the element called <name>.

Cannot add index to event database files `<name>.:<why>`

Error: The Console could not add an event report to its runtime database. Check to see if the directory exists and is writable. If it is, remove the runtime database files in the directory and restart the Console specifying `-i`.

Cannot allocate memory

Error: An error occurred allocating dynamic memory. The `malloc(3)` library function failed. Try increasing the swap space on your system or running fewer applications.

Cannot allocate memory for request

Error: There is not enough memory for the Console to proceed. Increase swap space on the system.

Cannot allocate memory to view request.

Error: A new request has been started, but the Console cannot allocate the memory required to display the request information in the View Requests window. Increase the swap space.

Cannot allocate request

Error: The system does not have enough memory to allocate resources for another request.

Cannot allocate resources to update request list.

Warning: Due to a memory allocation failure, the state of a request could not be updated in the request list. The system will recover by itself, but the user must make sure the system has enough memory.

Cannot backup existing file - no saving done.

Error: `snm_set` is unable to make a backup copy of the existing file before saving the newer version. This might be because there is no more space or no write permission on the file system.

Cannot bind socket

Error: An error occurred binding the socket name. The `bind(2)` system call failed.

Cannot cache request

Internal Error: An error occurred when the activity daemon tried to cache the request.

Cannot contact na.activity to determine id.

Error: The Console or Browser or Grapher tools attempted to contact the daemon to obtain an ID using SNM_NAME or USER. Make sure that na.activity is installed correctly and is in /etc/inetd.conf.

Cannot create agent request log file

Error: The agent request log file does not exist and cannot be created. Ensure the directory exists and is writable.

Cannot create base frame! Try removing some windows

Error: The operator has too many windows and the Console windows cannot be created. Try deleting some existing windows.

Cannot create colormap, (mono display?)

Error: The Console attempted to create a colormap for a color background, but the system appears to have a monochrome display. You must run the Console on a color display to load color background images.

Cannot create database directory:

The directory specified by the database keyword in the /etc/snm.conf file does not appear to be writable. Set the protection on the directory to 777 or change the database keyword to point to a directory that is writable by the user running SNM.

Cannot create event database files <name>.:<why>

Error: The Console could not create its runtime database. Check to see if the directory exists and is writable. If it is, remove the runtime database files in the directory and restart the Console specifying -i.

Cannot create key menu, not enough memory!

Internal Error: There is not enough memory to create the key menu. Check if the table has too many entries.

Cannot create resources for selecting requests; please remove some windows and try again.

Error: The pop-up window to allow you to select requests cannot be displayed because there are too many windows already open.

Cannot create resources for viewing requests; please remove some windows and try again.

Error: The Console is unable to display the View Requests window because there are too many windows already open.

Cannot create runtime database files *<name>.<why>*

Error: The Console could not create its runtime database. Check to see if the directory exists and is writable. If it is, remove the runtime database files in the directory and restart the Console specifying *-i*.

Cannot create RPC client:program = *<RPC program number>*;
version = *<RPC version number>*; RPC:Program not registered

Error: An error occurred creating an RPC client handle for the agent with the specified *<RPC program number>* and *<RPC version number>*. The agent (an RPC program) is not registered with the portmapper on the agent system. The error is generally caused by one of the following:

The agent is not installed on the host where you are sending the request. Install the agents on the remote system, as described in Chapter 4 of the *SunNet Manager 2.2.3 Installation Guide*.

The agent was started from the shell command line and subsequently killed. Follow the procedures listed under the error message "Remote procedure call failed: Can't send request: RPC: Timed out".

Cannot create TCP transport

Internal Error: An error occurred creating a TCP/IP RPC transport handle.

Cannot create UDP transport

Internal Error: An error occurred creating a UDP/IP RPC transport handle.

Cannot create window!

This is a window system problem. Check to make sure that the `DISPLAY` environment variable is set correctly, the user has permission to create a window on the requested `DISPLAY`, or the user does not have too many windows.

Cannot decode argument list

Internal Error: An error occurred during XDR decoding and de-serializing a message argument list.

Cannot decode message header

Internal Error: An error occurred during XDR decoding and de-serializing a message header.

Cannot delete activity in cache

Internal Error: The activity daemon failed to delete the activity in its activity cache.

Cannot delete request

Internal Error: An error occurred deleting a request from the agent request queue.

Cannot determine id. Check for existence of directory *<id-directory>*.

For the Solaris 2.x version of SNM, *<id-directory>* is

`/etc/opt/SUNWconn/snm/snm.id`.

For the Solaris 1.1 version, *<id-directory>* is `/etc/snm.id`.

Error: Check that *<id-directory>* exists and contains the user ID that you are using. If the directory exists, kill the activity daemon.

Cannot determine name. Neither `SNM_NAME` or `USER` set.

Error: One of these environment variables must be set for the Grapher and Browser or Console and Grapher to communicate; it must be the same for both tools.

Cannot display more than 1024 graphic objects.

Error: The Console cannot display more than 1024 different graphic objects (icons and raster files). You need to reduce the number of graphic objects in the runtime database.

Cannot encode argument list

Internal Error: An error occurred during XDR encoding and serializing a message argument list.

Cannot encode message header

Internal Error: An error occurred during XDR encoding and serializing a message header.

Cannot execute command:<cmd>

Error: The Console could not send an event to the program <cmd> because the command failed. Verify that the program exists and is executable.

Cannot execute mail command:<cmd>

Error: The Console could not mail an event to a user because the mail command <cmd> failed. Verify that the mail program exists and is executable.

Cannot fetch stats on database directory:

Error: The database db.<username> could not be found. Make sure that the directory specified by the SNMDBDIR environment variable is a valid directory that is writable. If you did not set SNMDBDIR, make sure that you are starting the Console from the same directory where the Console was last started.

Cannot find activity in cache

Internal Error: The activity daemon could not find the specified activity in its activity cache.

Cannot find 'activity-log' entry in configuration file, using '<file>'

Warning: The snm.conf file does not contain an entry for 'activity-log'.

Cannot find agent *<name>*

Internal Error: While launching a request, the agent name could not be found in the runtime database.

Cannot find agent name, *<name>*

Internal error: While making a request, the agent name specified could not be found in the runtime database.

Cannot find agent name, serial# = *<number>*, rid = *<number>*

Internal Error: Could not find the runtime database record describing the agent for the incoming data or event report.

Cannot find agent *<defaultagent>* to set

Error: The default agent you specified does not run on the target system or does not have any set-able attributes. The default will be ignored.

Cannot find attribute record for *<agentname>/<groupname>*.

Internal Error: snm_set is unable to retrieve attributes under group *<groupname>* for the agent *<agentname>* in the database. This might be an internal database error or the database is corrupted.

Cannot find cluster record, deferred request not sent.

Error: While trying to send a request to dump the deferred statistics, the request could not be found in the runtime database.

Cannot find cluster record, request not killed.

Error: While trying to send a request, the request could not be found in the runtime database.

Cannot find element *<name>* in database – no automatic request started.

Error: Element was deleted before the request could be started. Create the element and try again.

Cannot find enum text for *<attribute>*, number displayed.

Error: There is no database record to describe the text of the attribute which is an enumeration data type. Check the schema file for the attribute to see if the enumeration record is defined properly and all possible values for the attribute are specified.

Cannot find enumeration text for *<agent>/<group>/<attribute>*.

Internal Error: *snm_set* cannot find the enumeration record in the database for the attribute specified.

Cannot find enumeration value for *<attribute>*.

Internal Error: *snm_set* is unable to get the enumeration record for *<attribute>* due to a database error.

Cannot find font **

Error: The font specified in the Icon Font setting in the Console Properties Window category is not available on your OpenWindows server. Use the *xlsfonts* command to see available fonts.

Cannot find glue file *<filename>*.

Error: The *snm.glue* file cannot be found. Make sure that the *SNMHOME* environment variable is set correctly, and its contents are valid.

Cannot find group *<name>*

Internal Error: While launching a request, the group name could not be found in the runtime database.

Cannot find group *<defaultgroup>* to set.

Error: The default group you specified does not have any set-able attributes. The default will be ignored.

Cannot find group record, serial# = *<number>*, rid = *<number>*

Internal Error: Could not find the record describing the group/table for the incoming data or event report.

Cannot find home directory for *<userid>*

Error: The user name specified (with “~” or “~user”) is unknown or is known but has no associated home directory. Make sure that the user name is known (in the /etc/passwd file or in the NIS/NIS+ passwd map/table) and has an associated home directory.

Cannot find icon/rasterfile *<file>*

Error: Could not load a raster or icon file. Verify the file exists and is readable. Verify the viewBackground record in the instance file or elementIcon record in the elements schema is correct.

Cannot find key field:*<field>*.

Internal Error: The field called *<field>* used as the runtime database key for the record was not found in the record as expected. If the record is a component, view, connection or bus definition, be sure the record contains a field called Name.

Cannot find product; please set SNMHOME environment variable

Error: The SNMHOME environment variable needs to be set to the correct path for the SNM directory. The default of /usr/snm does not apply in this case.

Cannot find record for group *<group>*.

Internal Error: snm_set cannot locate the group record for *<group>* due to a database error.

Cannot find record of attribute *<attribute>*, request not sent.

Internal Error: The Console could not find the threshold for *<attribute>* in its runtime database.

Cannot find request in database

Internal Error: The Console couldn't find a request in its runtime database.

Cannot find request to terminate

API Error: The agent could not find the requested activity to terminate.

Cannot find request to update

Internal Error: An attempt to update the runtime database with a requests updated status failed. Ensure the file system containing the runtime database is not full.

Cannot find rpc id for *<agent>*, request not sent.

Error: While trying to send a request, the RPC program number for the *<agent>* could not be found. Be sure the RPC program number for this *<agent>* has been added to the RPC database.

Cannot find rpc id for redirection, request not sent

Error: This error may occur only when SNMP Redirect is in effect. The request could not be sent as the RPC id for the target agent was not specified correctly.

Cannot find system/schema name in *<hostfile>* on line *<lineno>*, ignored

Error: An invalid schema name is found in the SNMP host file on line *<lineno>*. Make sure the file has the right path and permission, and the line has the proper syntax.

Cannot find target object record *<system>*

Error: While processing an incoming report, the Console could not find the element representing the *<system>*. The report has been discarded.

Cannot find your home directory

Error: The user name specified (with “~” or “~user”) is unknown or is known but has no associated home directory. Make sure that the user name is known (in the */etc/passwd* file or in the NIS/NIS+ *passwd* map/table) and has an associated home directory.

Cannot fork subprocess

Internal Error: An error occurred creating a subprocess. The *fork(2)* system call failed.

Cannot get agent ID: *<why>*

Error: A call to the agent to get its ID failed for the reason shown.

Cannot get current working directory.

Error: snm_set is unable to get the current working directory for 'File' pop-up window. This might be because read or search permission is denied for a component of the path name.

Cannot get database record for <element> - unable to retrieve the element in the database due to database error.

Internal Error

Cannot get DES authentication handle

Internal Error: An error occurred getting a DES authentication handle.

Cannot get local host name

Internal Error: A error occurred getting the local hostname. The sysinfo(2) system call failed.

Cannot get report message information:<reason>

Error: While processing an incoming report, the Console could not obtain the information about the report. The report has been discarded.

Cannot get request info, deferred request not sent.

Error: While trying to send a request to dump deferred reports, the information about the request could not be extracted from the runtime database record.

Cannot get request info, request not killed.

Error: While trying to kill a request, the request could not be found in the runtime database. The request was not killed.

Cannot get RPC# for agent <agentname>.

Error: snm_set is unable to get the rpc number for <agentname>. This might be a mismatch of the agent's RPC number or the agent is not registered.

Cannot get RPC program number from rpc database for agent *<agentname>*.

Error: The RPC database does not have an entry for the specified agent. Make sure that are specifying a valid agent name. If you are running NIS/NIS+, you must manually add the agent entries to */etc/rpc* and */etc/services* on the NIS/NIS+ master.

Cannot get socket

Internal Error: An error occurred creating a socket. The *socket(2)* system call failed.

Cannot get socket name

Internal Error: An error occurred getting the socket name. The *getsockname(2)* system call failed.

Cannot get time of day

Error: The system did not return the time of day. (The *gettimeofday(2)* system call failed.)

Cannot get user netname

Internal Error: An error occurred converting from a domain-specific hostname to an operating-system independent netname.

Cannot initialize database manager traps – exiting.

Internal Error: Contact Sun support.

Cannot initialize field value *<value>*

Error: The field value *<value>* could not be placed in the record. Either there are not enough fields in the record or the value is not the correct type for the field.

Cannot insert activity in cache

Internal Error: The activity daemon failed to insert the activity in its activity cache.

Cannot load color rasterfile on B&W monitor:*<file>*

Error: The raster file *<file>* is a color raster file and the Console is running on a B&W system. Color raster file can only be loaded on color systems.

Cannot load file *<file>*

Error: The named *<file>* does not exist or is not readable. Verify that the *<file>* exists and is readable.

Cannot load more than 1024 icons/rasterfiles.

Error: No more than 1024 icons/raster files can be loaded into the Console. Reduce the number of icons/raster files in the runtime database.

Cannot open *<file>* - load failed

Error: The named *<file>* does not exist or is not readable. Verify that the *<file>* exists and is readable.

Cannot open activity log:*<file>*

Warning: The Console could not open the activity log. No checking will be done for requests that have terminated. Verify that the activity daemon has started, and that the user has access to the activity log file.

Cannot open event database files *<name>.:<why>*

Error: The Console could not open its runtime database. Check to see if the directory exists and is readable. If it is, first delete the file `/usr/tmp/NETISAM.LOCKTABLE`, then remove the runtime database files in the directory. Restart the Console with the `-i` option.

Cannot open file *<file>*

Error: The requested *<file>* could not be opened. Make sure the directory exists, and has the appropriate permissions for access.

Cannot open *<file>* for reading.

Error: The *<file>* to load does not have read permission.

Cannot open *<file>* for writing.

Error: The *<file>* to save does not have write permission.

Cannot open *<file>* - save failed.

Error: The named path to the *<file>* might not exist or is not writable. Verify that the path to the *<file>* exists and is writable.

Cannot open icon directory *<directory>*

Error: The Console could not open the icon directory *<directory>* as specified in the Locations category of the Console's Property window. Check to verify that the *<directory>* exists and that you have read and execute (list) access.

Cannot open lock file: *<file>*: *<why>*

Error: The lock file *<file>* could not be opened for the stated reason. Set the protection on the lock file so that the *<file>* is readable.

Cannot open runtime database files *<name>*..*<why>*

Error: The Console could not open its runtime database. Check to see if the directory exists and is readable. If it is, first delete the file `/usr/tmp/NETISAM.LOCKTABLE`, then remove the runtime database files in the directory. Restart the Console with the `-i` option.

Cannot open schema directory *<directory>*

Error: The agent/element schema directory could not be found. Verify that the *<directory>* is readable and that path name specified in the Locations category of the Console's Properties window points to the correct *<directory>*.

Cannot open temporary print file *<file>*

Error: The temporary print file could not be created. Verify that `/tmp` is writable.

Cannot read kernel memory

Error: An error occurred reading from kernel memory. This could happen if the application doesn't have permission to read kernel statistics.

Cannot register callback (*<why>*)

Internal Error: The Console could not register itself with the port mapper.

Cannot register rendezvous with event dispatcher (*<why>*)

Internal Error: An error occurred registering a rendezvous with the event dispatcher.

Cannot register RPC service

Internal Error: An error occurred registering the RPC service with the portmapper.

Cannot register with event dispatcher (<why>)

Error: The Console could not register itself with the event dispatcher, na.event. Verify na.event has been correctly installed and is either started manually or from inetd.

Cannot request data report for group '(<groupname>)' (<reason>)

Error: The request to the agent failed for the reason shown.

Cannot save enumeration <name>

Error: The enumeration could not be saved in the runtime database.

Cannot set argument (<error>)

Error: While trying to send a request, the netmgt_set_argument(3n) call failed and returned <error>.

Cannot set first threshold (<error>).

Internal Error: While trying to set a threshold, the netmgt_set_threshold(3n) call failed and returned <error>.

Cannot set instance (<error>)

Error: While trying to send a request, the call to netmgt_set_instance(3n) failed and returned <error>.

Cannot set second threshold (<error>)

Error: While trying to set a threshold, the call to netmgt_set_threshold(3n) failed and returned <error>.

Cannot set socket SO_REUSEADDR

Internal Error: An error occurred setting the socket option. The setsockopt() system call failed.

Cannot start more requests

Error: The agent could not start the request because it would exceed the maximum number of requests the agent is configured to perform. You must either terminate some of the requests the agent is performing or reconfigure the agent to perform more requests. See the `snm.conf(5)` manual page.

Cannot start request '<reason>'

Error: The agent could not start the request because of the specified '<reason>'.

Cannot terminate request

API Error: An error occurred terminating the requested operation. The `kill(2)` system call failed.

Cannot unregister rendezvous with event dispatcher

Internal Error: An error occurred unregistering a rendezvous with the event dispatcher.

Cannot unregister RPC service

Internal Error: An error occurred unregistering the RPC service with the port mapper.

Child process ended unexpectedly

Internal Error: The child process has encountered a fatal error and exited.

Cluster record overflow.

Error: Too much data in an element record.

Console must be restarted to get correct color images

Error: The Console was unable to obtain the colors required for the icons or backgrounds specified in the schema file being loaded. To obtain the correct colors, stop and restart the Console.

Could not change type for <system> due to memory allocation failure

Error: There was not enough memory or swap space available to make the change. Free up space by killing unnecessary processes etc., and retry.

Could not open schema file : (<filename>)

Error: The listed file could not be opened for reading.

Creating a database in <directory>.

Information Only: Tells the user where database was created.

Database directory is not a directory: <name>

Error: db.username was found, but is not a directory (it could be a regular file, named pipe, etc.). Verify that the value associated with the SNMDBDIR environment variable is a valid directory.

Delete from view <view> failed.

Warning: In attempting to delete an element from the view called <view> the operation failed.

Detected *n* errors between schema file and agent'(<agentname>)' on '(<hostname>)'

Error: This is a summary of errors found. Correct the agent schema file and/or agent code.

Duplicate agent name <name>

Error: The agent/proxy name matched an agent/proxy already in the runtime database. Change the conflicting agent name.

Duplicate agent/proxy:<name>

Error: An element was defined that contained two or more agent or proxy records with the same name. Fix the schema file and reload it.

Duplicate attribute name:<name>

Error: The agent has a group with two attributes with the same name. Change the name of one of the attributes and then reload the schema file.

Duplicate definition of group (<groupname>)

Error: The named group/table has been previously defined in this agent schema. Correct the agent schema file

Duplicate enum name *<name>*

Error: The enumeration name is a duplicate of an existing enumeration name.

Duplicate error code *<number>*

Error: The agent error code *<number>* appears for multiple messages. Correct the agent schema file.

Duplicate field name *<name>*

Error: The field name matched a field already in the runtime database. Change the conflicting group/table name.

Duplicate group/table name *<name>*

Error: The group/table name matched a group/table already in the runtime database. Change the conflicting group/table name.

Duplicate id: *<id>*

Error: When adding a request record to the runtime database, the serial number *<id>* of the request matched an existing request. Make sure all requests have unique serial numbers.

Duplicate name: *<name>*

Error: When adding a component, view, connection or bus record to the runtime database, the name for the record matched an existing record. All elements must have unique names. Change the name of the element.

Duplicate record name *<name>*

Error: A record name matched that of an existing record. Change the name of one of the record definitions.

Duplicate request id #*<number>* ignored

Warning: While reading in a MDB file, two requests with the same request identification were found. The duplicate request is ignored.

Empty queue

Internal Error: The queue specified is empty.

Error creating new stream, file load aborted.

Internal Error: Contact the appropriate Sun service entity and submit the log file.

Error report message too long

API Error: The error report message string is too long.

Error while asking for portmapper list

Warning: The remote portmapper was not reachable.

Error while connecting to portmapper

Warning: The remote portmapper was not reachable.

Error while creating client to portmapper

Warning: The remote portmapper was not reachable.

Error while creating ICMP socket

Internal Error

Error while creating pipe

Internal Error: An error occurred when creating the pipe to a subprocess.

Error while creating portmapper socket

Error while creating raw socket

Internal Error

Error while getting interface flags

Internal Error

Error while receiving response

Warning: Discover noticed that an unexpected error occurred while waiting for a ping response and will retry the ping request.

Event occurred, however could not get event data

Internal Error

Event report data too big

API Error: The event report is larger than 6KB.

Failed to fetch netmgt error

Internal Error: An incoming error message was received, but the Set tool could not fetch the error information.

Failed to fetch Set reply: *<reason>*

Internal Error: snm_set cannot get the confirmation message of the set request.

Fatal error

The requested operation experienced a fatal agent specific error. The global variable `netmgt_error.agent_error` (if nonzero) is an index to an error string in the agent schema file. This string describes the cause of the error. The global variable `netmgt_error.message` (if non-null) is a pointer to a string containing additional error information.

File *name* load aborted after *<number>* reports read.

Error: Loading of the file was aborted, as too many errors were detected in the log file. Make sure that the log file is in the correct log file format; see the `snm.logfile(5)` manual page.

File not found: *<file>*.

Error: The *<file>* to load does not exist. Make sure that the *<file>* exists in the specified directory.

File *<file>* is not a regular file.

Error: The Browser can only read regular UNIX files and not directories or special (device) files. Supply the name of a regular file.

Grapher not running

Error: A request was launched from the Console or the Browser that specified sending results to the Grapher but the Grapher is not running. Start the Grapher and then restart the request.

HOME environment variable is not set

Error: Set the HOME environment variable to point to the home directory.

Invalid agent security level

API Error: An invalid agent security level was specified.

Invalid message type

API Error: The requested operation is invalid for this message type.

Invalid group name

API Error: An invalid group name was specified. This error occurs when: (1) there is no entry for the target device in the SNMP host file so the default SNMP schema (na.snmp.default-schema) was used, but the default schema does not contain the specified group, or (2) the schema defined in the SNMP host file does not contain the specified group.

Invalid input (date type is *<type>*).

Error: There is an error in the input data. Check if the input is of the proper data type.

Invalid key name

API Error: An invalid key was specified.

Invalid object id *<id>* found in *<filename>*

Error: Invalid input was found in the SNMP schema file while the SNMP trap daemon was loading data. This might be a syntax error in or an invalid object identifier in the file.

Invalid relational operator

API Error: An invalid threshold relational operator was specified.

Invalid reporting count

API Error: An invalid data or event reporting count was specified.

Invalid reporting interval

API Error: An invalid data or event reporting interval was specified.

Invalid request timestamp

API Error: An invalid request timestamp was specified.

Invalid RPC timeout

API Error: An invalid RPC timeout was specified.

Invalid system name

API Error: An invalid system name was specified.

Invalid trap-signal type *<type>*

Error: The trap-signal type specified is invalid.

Invalid type *<type>*

Error: The type is unknown.

Load failed, invalid input data/format

Error: The file to load does not have a valid input format. Refer to Chapter 8, "Set Tool," for more information about using Set Tool.

Load failed, no writeable attribute for target system in file.

Warning: There is no applicable data for the current target system to load from the file. Nothing will be loaded.

Load failed, not enough memory!

Internal Error: Not enough internal memory is available to load the data.

Load of icon file failed

Error: Could not load a raster or icon file. Verify the file exists and is readable. Verify the `viewBackground` record in the instance file or `elementIcon` record in the elements schema is correct.

Memory allocation failure – no automatic request started for *<name>*

Error: System ran out of memory. Maybe there is not enough swap space. Try again.

`nc_rebuildIcons:add` failed.

Internal Error: when rebuilding the glyph list, the Console could not insert a glyph record.

NIS not running

Error: The Network Information Services (NIS) are not running. NIS/NIS+ is required if you are using secure RPC.

No agent host name

API Error: No agent host name was specified when sending the request.

No agent name

API Error: No agent name was specified.

No agent to set.

Error: There is no agent that has any set-able (that is, writable) attributes running on the current target system.

No argument buffer

API Error: No argument buffer was specified.

No argument length

API Error: No argument length was specified.

No argument name

API Error: No argument name was specified.

No argument type

API Error: No argument name was specified.

No argument value

API Error: No argument value was specified.

No attribute information saved to Unset.

Error: snm_set is unable to undo the previous set request. This might be because the previous set request failed.

No attribute information supplied to Set.

Error: No attribute set information has been supplied to do the set request.

No callback dispatch function

API Error: No callback dispatch function was specified.

No Console

Information Only: The Console was not running when the database record was added by Discover.

No data buffer
API Error: No data buffer was specified.

No error report buffer
API Error: No error report buffer was specified.

No event report buffer
API Error: No event report buffer was specified.

No event threshold buffer
API Error: No event threshold buffer specified.

No group name
API Error: No group name was specified.

No group to set.
Error: There is no group that has any set-able (that is, writable) attributes under the current agent.

No groups in this agent?
Error: There were no groups or tables listed in the schema file. Check the agent schema file.

No manager host name
API Error: No manager host name was specified when sending the request.

No message information buffer
API Error: No message information buffer was specified.

No more space to do Unset (malloc failed when building backup list).
Internal Error: Not enough memory is available.

No optional argument buffer
API Error: No optional argument buffer specified.

No relational operator
API Error: No threshold relational operator was specified.

No rendezvous host name
Error: No rendezvous host name was specified when sending the request. You need to specify a rendezvous name in your request.

No report to print

Warning: There was no report to print.

No request information buffer

API Error: No request information buffer specified.

No Streams Selected

Warning: Attempted to send data from the Browser to the Grapher without having stream(s) selected first.

No such file: <file>

Error: The image file specified in the database does not exist. Check the database for the correct file name and directory path.

No such rasterfile: <file>

Error: The raster file <file> could not be opened. Check that the file exists and that the permissions allow reading.

No suitable network interfaces are available

Error: None of the interfaces found on the system were available for Discover watch mode. Discover does not use the loopback interface, a point-to-point interface, or an interface that is marked “down” when using watch mode. If no other interfaces are available, watch mode cannot be used. Use Discover with “ping” mode instead.

No system name

Error: No system name was specified. You need to specify a system name in your request.

No table key

Error: The set request for a tabular attribute did not specify a table key. Specify a table key and re-send the request.

No threshold buffer

API Error: No threshold buffer was specified.

No threshold is set for attribute <name>.

Internal Error: While trying to request event reports, an attribute was found that did not have any thresholds set.

No threshold is set, request is not sent.

Error: A request to start event reporting has no thresholds. The request is ignored.

No threshold value

API Error: No threshold value was specified.

No value buffer

API Error: No value buffer specified.

No value saved to unset attribute *<attribute>*

Error: `snm_set` is unable to undo the previous, successful set for *<attribute>*. This might be because the previous set request failed, or the attribute is not in the list of items set in the previous request.

Non-fatal error

Warning: An agent-defined warning message is being returned. The global variable `netmgt_error.agent_error` (if nonzero) is the number of the error string in the agent schema file. This string describes the cause of the error. The global variable `netmgt_error.message` (if non-null) is a pointer to a string containing additional error information.

Not a cluster record

Internal Error: A request was found in the runtime database that was not a cluster record.

Not enough memory for additional records

Error: The Console has run out of virtual memory for additional record definitions. Either allocate more swap space for the system or run fewer applications concurrently.

Not enough memory for clone window!

Error: The Browser could not allocate enough memory to clone the requested window. Try increasing the size of the system's swap space.

Not Interesting

Information Only: The `-v` and `-e` options were specified for Discover, and a system was found that did not meet the "interesting system" criteria.

<file name>: Not Readable by SunNet Manager...

An attempt was made to load *<file name>* as a runtime database or data/events request file but *<file name>* is not in ASCII format.

Not regular file: *<file name>*

Error: The file *<file name>* is not a regular file. Check to make sure the file is of a valid type.

Note - attribute is 'writeonly': group '*(<group>)*',
attribute '*(<attr>)*'

Error: Since the named attribute is write-only, the agent doesn't return it; it cannot be checked with `snm_cmd -v`.

NULL callback socket pointer

Error: A NULL callback socket pointer was specified. Specify the address of a socket descriptor in your request.

Null cluster record

Error: An empty cluster record was found in the MDB.

NULL queue node data pointer

Internal Error: A null queue node data pointer was specified.

NULL queue node pointer

Internal Error: A null queue node pointer was specified.

NULL queue pointer

Internal Error: A null queue pointer was specified.

Number of bits in host part must be >0 and <=24

Error: A negative, zero-length or large (more than three bytes) host part is not legal under TCP/IP. Specify a legal number of host bits between 1 and 24 inclusive.

Only agent child can call this function

API Error: Only an agent child process can call this function.

Only agent parent can call this function

API Error: Only an agent parent process can call this function.

Only manager can call this function

API Error: Only a manager application can call this function.

Only rendezvous can call this function

API Error: Only a rendezvous application can call this function.

Opening event log

Error: The Console couldn't allocate enough memory for the event log.

Opening event log state

Error: The Console couldn't allocate enough memory for the event log state.

Out of memory (<*program*>)

Error: The program <*program*> was not able to allocate enough memory for its operation. Try to reduce memory utilization by killing unused processes or by increasing swap space. If this error persists, contact the appropriate Sun service entity and specify the name of the program.

Out of memory...cannot mail to <*user*>

Error: The Console ran out of virtual memory while trying to mail an event to <*user*>. Increase the amount of swap space for the system or run fewer applications concurrently.

Out of memory...cannot send to <*cmd*>

Error: The Console ran out of virtual memory while trying to send an event to the program <*cmd*>. Increase the amount of swap space for the system or run fewer applications concurrently.

Over <*number*> agents defined

Warning: Too many agents defined in the system. Reduce the number of agents.

Over <*number*> agents on system: <*name*>

Warning: Too many agents were added to the element called <*name*>. Not all agents will be displayed in the menus.

Over <*number*> elements in view.

Warning: A view contains too many elements. Not all the elements in the view will be displayed. Break the view up into two or more subviews.

Over <*number*> record types defined.

Warning: Too many record types defined.

Over <*number*> types defined.

Warning: Too many components, views, connections or buses defined. Not all will be listed in the menus.

Parsing error, file not saved

Error: An error occurred when the SNMP trap daemon was parsing the SNMP schema file. Make sure that the file has the proper syntax, as specified by the agent schema syntax definitions.

Reading MDB: *<error>*

Error: An error occurred when reading from the MDB.

Record *<#id>* is not an attribute record, request not sent.

Internal Error: A record in the runtime database that was expected to be an attribute record was not. The request has not been started.

Record does not contain 'Name' field

Error: A element was defined whose record did not contain a 'Name' field. All elements definitions must contain a 'Name' field. Fix the element schema file and then restart the Console and load in the new files.

Remote procedure call failed: *<message>*

Error: A remote procedure call error was detected. The error *<message>* indicates the reason.

Remote procedure call failed: Can't send request:RPC:
Timed out

Error: The request failed because the remote procedure call timed out before a request confirmation was received by requesting processes. This error is generally caused by one of the following:

The network connection between the manager and agent cannot be established because of network congestion. The Console continues to re-send the request until the request succeeds or the user kills the request.

The host where the Console and/or the agent is installed is slow because of high CPU utilization, disk I/O, etc. The Console continues to re-send the request until the request succeeds or the user kills the request.

A new copy of the agent was installed on the system and inetd(8c) was not initialized. You should:

1. Kill any requests for the agent from the Console. You only need to kill requests for the agent on the system where the new version of the agent is installed.

2. Log into the system where the new version of the agent is installed.
3. Reinitialize `inetd(8c)` by sending `kill -HUP` to the `inetd(8c)` process.
4. Re-send the request.

Report is too big: 52438 byte maximum for event reports.
Error: The maximum event report size that can be returned is 52438 bytes. You cannot request event reports where the returned report size would exceed this limit (for example, where every row in a very large table is returned).

Request `<req>`: Agent error: `<error>`: `<error_info>`
Warning: Recoverable operational error encountered; the request will be retried.
Error: Unrecoverable operational error encountered; the request will not be retried.

Request `<req>`: API error: `<error>`
Error: Unrecoverable operational error encountered; the request will not be retried.

Request `<req>`: API error: `<error>`: Retrying
Error: A recoverable operational error was encountered. The request will be retried.

Request `<req>`, cannot get confirm message
Internal Error: An incoming error message was received but the Console could not fetch the error specific information.

Request must be performed by a new subprocess
API Error: Request cannot be performed by an existing subprocess. You must create a new subprocess to handle this request.

Request too large
Internal Error: a request was found that was too large.

Request without request state encountered
Internal Error: A runtime database entry for a request was found that did not contain a request state entry.

Requesting deferred data reports (`<error>`)
Error: While requesting deferred data reports, the API error specified occurred.

Request(s) failed

Error: The request could not be started.

Request has ended unexpectedly

Warning: A request ended without notifying the Console that it was ending. Some possible reasons are the agent has crashed, the remote machine has crashed, etc. If the request has “Restart” set to true, the Console will attempt to restart the request and will indicate this by appending the text “— restarting” to the error message.

Request(s) invalid

Error: The request’s data fields were not valid for submission to the helper process.

Request(s) submitted

Information Only: Requests were successfully submitted to the helper process.

Row/instance does not exist with key <key>

Error: snm_set is unable to retrieve the row identified by <key>.

Runtime database does not contain <name> record definition

Error: Restart the Console with the -i option.

Save failed, calloc(): not enough memory!

Internal Error: Not enough memory is available.

Save failed, fwrite error!

Internal Error: The file is not able to be saved due to an I/O error.

Save failed, unable to process line <linenumber>

Internal Error: snm_set is unable to save the line numbered <linenumber> in the Set Information window.

Saw bad file entry '<entry>...' at line <line>

Saw bad logfile entry at offset <number>

Saw bad microseconds value '<seconds>' at line <line>

Saw bad received time '<time>' at line <line>

Saw bad RPC number '<number>' at line <line>

Saw bad seconds value '<seconds>' at line <line>

Error: The log file has an incorrect entry at the indicated <line>. Correct the entry according to the format specified in snm.logfile(5). If the log file entry was programmatically generated, the application that wrote the entry should be fixed.

Schema serial number mismatch: agent wants *<n>*, schema is *<m>*.

No attribute validation done.

Error: The agent was initialized with a particular serial number coded in the agent software. The agent schema serial number given by the 'serial' keyword should match, otherwise the agent and schema are out of sync. If no 'serial' keyword is given, serial 1 is assumed. This is a fatal error; *snm_cmd* will not continue.

scroll.c: Cannot acquire selection!
Internal Error.

Security credentials too weak

Internal Error: The requester's security credentials are too weak.

SizeField: Need to give size for type *<type>*

Internal Error: the type for a field did not match a type known to the Console.

snm already running: started *<date>* on *<host>* (*pid <pid>*) by *<user>* on display *<displayname>*

Error: The Console is already being run under the user name specified in the message. Change *SNM_NAME* and rerun the Console. (Existing Browser and Grapher instances would need to be restarted with the same name.)

Specified database directory is not a directory: *<path>*

Error: The database directory path must be a directory.

Success

The requested operation completed successfully.

Symbol 'rthost' is not in the kernel's namelist

Symbol 'rtnet' is not in the kernel's namelist

Symbol 'rthashsize' is not in the kernel's namelist

Internal Error: A required symbol (related to the routing table) was not found in the kernel's name list.

Syntax error in *<trapfile>* on line *<lineno>*

Error: Invalid input found in the trap file on the line indicated.

The receiving application cannot handle
Error: The Browser cannot send data to the specified receiver because the receiver cannot accept the indicated data type. For example, the Results Grapher can only accept integer, float, gauge, and counter data types.

These attribute(s) not returned by agent '*agentname*' group '*group*'

attribute '*attr1*' (type (*type1*))

attribute '*attr2*' (type (*type2*))

Error: The given list of attributes was named in the agent schema but the agent didn't return any of them. Check the agent schema.

This folder cannot be renamed

Warning: An attempt was made to rename the Browser Main folder. To create a new folder, use the New Folder option of the Edit menu.

Time must be > 0

Error: The pause or wait times for Discover must be positive numbers.

Too many fields

Error: An attempt was made to store into a field that did not exist in the record.

Too many panel items – <*n*> items drawn

Internal Error: Contact Sun support.

Too many requests

Warning: No more requests can be added. Out of virtual memory.

Two agent definitions in schema file

Error: Only one agent definition is allowed per schema file. Correct the agent schema file.

Type field. Need to give size for type <*type*>

Internal Error: The type for a field did not match a type known to the Console.

Unable to connect to Graph Tool: <*reason*>

Error: The Console or Browser could not contact the Grapher for the reason shown. If the Grapher is not running, start the Grapher and restart the request. If the Grapher is running, restart the Grapher and then restart the request.

Unable to extract RPC id number from *<file>*

Error: The Console or Browser could not determine the RPC program number of the Grapher from the file *<file>*. Restart the Grapher and then restart the request.

Unable to load requested color, try increasing colormap-size.

Error: The raster file file has too many colors to be loaded. Either reduce the number of colors in the raster file or increase the number allocated to the Console by modifying the Maximum Colors setting in the Console Properties Windows category.

Unable to obtain Grapher port number

Error: The Console or Browser could not locate the port number for the Grapher. You may have attempted to send data to the Grapher, but the Grapher is not running. Start/Restart the Grapher and then either restart the request or re-send the data to be graphed.

Undefined attribute '*(<attr>)*' (type (*<type>*)) returned by agent

Error: An attribute with the given name and type was returned by the agent but is not listed in the agent schema. Correct the agent schema.

Undefined enumeration name *<name>*

Error: The enumeration *<name>* is undefined.

Unexpected NIT read error

Internal Error

Unknown agent name *<name>*

Error: The agent name *<name>* is unknown.

Unknown argument name

API Error: An unknown argument name was specified.

Unknown argument type

API Error: An unknown argument type was specified.

Unknown data type (*<type>*) returned for request.

Error: Agent error: The incoming monitor or event report contained the unknown data type *<type>*.

Unknown data type: attribute *<attribute>*, type *<type>*

Error: The data type of the attribute to be set is not valid.

Unknown field *<name>*

Error: The field *<name>* cannot be found in the record.

Unknown field (*<name>*) returned for request.

Error: Probable agent/schema file mismatch: an incoming monitor or event report contained a field that was not defined in the agent schema.

Unknown host address

API Error: The specified host address could not be translated to a host name.

Unknown host *<name>*

API Error: The specified host name could not be translated to a host address.

Unknown name *<name>*

Error: The element *<name>* is unknown.

Unknown net/host *<name>*

Error: Either the host named was not found in the `/etc/hosts` file or the NIS/NIS+ host map/table, or the network named was not found in the file `/etc/networks` or the NIS/NIS+ host map/table.

Unknown protocol requested

Error: The Console or Browser attempted to contact the Grapher using an unknown protocol id. Restart the Console and try the request again.

Unknown record *<name>*

Error: Record *<name>* is undefined.

Unknown record name *<name>*

Error: The record name *<name>* is unknown.

Unknown request type

API Error: An unknown request type was specified.

Unknown system name *<name>*

Error: The element called *<name>* could not be found.

Unknown transport protocol

API Error: An invalid transport protocol was specified.

Unrecoverable error while opening ICMP socket

Internal Error.

Username too long: *<name>*

Error: The supplied user name is illegal; it has too many characters.
Supply a legal user name.

Using monochrome mode.

Information Only: Static visuals are treated as monochrome—all
glyphs are black and white.

Value is too big

Internal Error: The agent cannot return a value because it would
overflow an internal buffer.

Warning: field *<field>* not present in component.*<type>*

Warning: When the element type of an element is changed, not all
properties of the original element type can be converted to the new
element type.

Would exceed queue length limit

Internal Error: A node cannot be appended to a queue because the
queue length limit would be exceeded.

Writing MDB: *<error>*

Error: An error occurred when writing to the MDB.

Man Page Summary



<i>User Commands</i>	<i>page B-2</i>
<i>File Formats</i>	<i>page B-4</i>
<i>Agents and Daemons</i>	<i>page B-5</i>

This appendix lists on-line man pages available for user commands, file formats, and maintenance commands. The *SunNet Manager 2.2.3 Programmer's Guide* lists man pages that describe the programmatic interface to the product.

To use the man pages on-line, add the SunNet Manager man pages directory to MANPATH environment variable statement in your `.cshrc` or `.profile` file.

B.1 Setting the MANPATH on Solaris 1.x

If you installed the SunNet Manager software in `/usr/snm`, enter one of the lines below on a command line or in your shell startup file.

For a C-shell, in your `$HOME/.cshrc` file enter:

```
setenv MANPATH ${MANPATH}:/usr/man
```

For a Bourne or Korn shell, in your `$HOME/.profile` file enter:

```
set MANPATH=${MANPATH}:/usr/man
export MANPATH
```

B.2 Setting the MANPATH on Solaris 2.x

If you installed the SunNet Manager software in `/opt/SUNWconn/snm`, enter one of the lines below on a command line or in your shell startup file.

For a C-shell, in your `$HOME/.cshrc` file enter:

```
setenv MANPATH ${MANPATH}:/opt/SUNWconn/man
```

For a Bourne or Korn shell, in your `$HOME/.profile` file enter:

```
set MANPATH=${MANPATH}:/opt/SUNWconn/man
export MANPATH
```

B.3 User Commands

The man pages for user commands are as follows:

build_oid

Builds an object identifier database, which the Console uses to interpret object identifiers. Object identifiers are numeric; the database allows the Console to display the identifier as a string.

mib2schema

Translates MIB files for SNMP agents to the format required by SunNet Manager's agent schema files.

snm

The command that invokes the SunNet Manager Console. The Console enables you to model and visualize an arbitrary collection of network elements and to coordinate agent activity to gather network information.

`snm_br`

The SunNet Manager Results Browser is an OpenLook application that processes log files generated by various components of the SunNet Manager package.

`snm_cmd`

The SunNet Manager command-line manager application. It provides a way to send requests to agents without having to run the Console or write dedicated test programs.

`snm_cmdtool`

Uses `snm_exec` (1) to bring up a Command Tool in which a specified command line is invoked.

`snm_exec`

Executes a command and pause following command completion. The program then prompts you to press Return. Usually used in SunNet Manager schema files to cause the Console to create a new Command Tool.

`snm_gr`

The SunNet Manager Results Grapher is an OpenLook application that displays in various graph forms the information sent to it by the Console or Browser.

`snm_kill`

Stop one or more agent requests according to options you specify, such as the agent host, agent name, and protocol version of the agent.

`snm_parser`

An example application for parsing SunNet Manager schema files. Provided in source form so that you can build applications that can read schema files.

`snm_set`

SunNet Manager Set Tool, which allows setting the writable attributes for a system.

`snm_version`

Displays the SNM binaries version information.

`getagents`

Copies agents from manager station to agent machine. Must be run on the agent machine. Supports installation of agents onto both SunOS 4.x and 5.x machines.

`snm_discover`

Dynamically creates SunNet Manager runtime database records for devices it finds on a network. When an element is discovered, some analysis is done on it to determine its hostname, type, and capabilities. The element is then added to the database.

B.4 File Formats

The man pages for file formats are as follows:

`discover.conf`

A file used to customize Discover Tool to find agents not shipped with SNM, to utilize system descriptions returned by SNMP agents, to customize the glyphs representing network elements, or to make use of customized views.

`linkmap`

A file that contains entries defining link management information for links of connection type objects (`connection.link`) *not* simple connections. This file gets populated by the SNM Discover program. The format of the `linkmap` file is `<hostname> <ifIndex> <linkname>`.

`na.snmp.hostfile`

Contains per-host information for the SNMP proxy agent and the SNMP trap proxy.

`na.snmp.trapfile`

Contains mappings from SNMP trap numbers to names, on a per-enterprise basis. Also allows for the discarding of specified trap types.

`snm.conf`

SunNet Manager configuration file for agents and daemons. Also specifies per-host security levels and can be used for third-party SunNet Manager applications.

`snm.logfile`

Defines the format of the log file written by `na.event` and `na.logger`. This is the format that is read by the Results Browser.

`snm_schema`

An ASCII file used to build a management database. Consists of one or more definition types among agents, records, instances, clusters, and enumerations.

`snmp.schema`

Contains data definitions for MIB-I objects defined in RFC 1156. The SNMP proxy agent can use this schema to return information from SNMP agents that use the MIB-I data definitions.

`snmp-mibII.schema`

Contains data definitions for MIB-II objects defined in RFC 1213. The SNMP proxy agent can use this schema to return information from SNMP agents that use the MIB-II data definitions

B.5 Agents and Daemons

Agents and daemons include:

`na.activity`

Management activity daemon. Maintains a record of currently active data- and event-reporting requests.

`na.diskinfo`

Agent that reports the total amount of disk space occupied by a filesystem, the amount of used and available disk space and the amount of capacity used.

`na.etherif`

Returns Ethernet interface information for SunOS 4.x systems.

`na.etherif2`

Returns Ethernet interface information for SunOS 5.x systems.

`na.event`

Receives agent event reports and forwards them to all processes that have registered to receive such reports. Also records each event report it receives.

`na.hostif`

Returns information about the interfaces that send IP packets on the host where the agent runs.

`na.hostmem`

Returns statistics for memory buffer pool usage of network routines for SunOS 4.x. These statistics include such items as streams allocation and mbuf usage.

`na.hostmem2`

Returns statistics for memory buffer pool usage of network routines for SunOS 5.x systems. For 5.x, `na.hostmem2` has a single attribute group, streams.

`na.hostperf`

A proxy agent that returns statistics from the kernel, including CPU usage, paging/swapping, and disk and interface statistics.

`na.iostat`

Returns CPU, disk, and TTY statistics for SunOS 4.x systems.

`na.iostat2`

Returns CPU, disk, and TTY statistics for SunOS 5.x systems.

`na.ippath`

A proxy agent that tries to trace an IP packet's path between the proxy system and a destination host.

`na.iproutes`

Returns routing table statistics, including the number of bad routing redirects, the number of dynamically created routes, the number of routes changed due to redirects, the number of destinations found unreachable, and the number of uses of a wildcard route. Can also return routing table data such as is returned by `netstat -r (1m)`.

`na.layers`

Returns network layers (TCP, UDP, ICMP, and IP) information for 4.x systems.

`na.layers2`

Returns network layers (TCP, UDP, ICMP, and IP) information for 5.x systems.

`na.logger`

Receives data, event, and error reports from agents and records them in a log file.

`na.lpstat`

A proxy agent that allows you to examine printer status and the print job queue.

`na.ping`

A proxy agent that reports machine reachability information, as determined through the use of ICMP echo request packets.

`na.rpcnfs`

Reports RPC and NFS statistics for a host.

`na.snmp`

The SNMP proxy agent that allows a SunNet Manager user to manage SNMP devices.

`na.snmp-trap`

The SNMP trap daemon receives SNMP traps, translates them to SunNet Manager traps, and forwards them to the event dispatcher on one or more management stations.

`na.snmpv2`

The SNMP proxy agent that allows a SunNet Manager user to manage SNMP Version 1 and SNMP Version 2 devices.

`na.sync`

Returns a table of information about the synchronous serial interfaces on the host running the agent.

`na.traffic`

Gathers statistics about Ethernet traffic in a given interval and reports the information at the end of the interval.

`snmpd`

An RFC 1157-compliant SNMP agent for use on SunOS 5.x systems.

`snmpv2d`

A mostly RFC 1448-compliant agent for managing SNMP devices on SunOS 5.x systems.

Glossary

This is a glossary of terms used in the SunNet Manager documentation. Wherever possible, terms have meanings similar to standard usage within the network management community. Some definitions have been slightly modified to better illustrate unique SunNet Manager concepts.

activity daemon

Process that keeps a list of active agent requests that have been sent from the host where the daemon is running. The daemon periodically queries the agents to make sure they are still servicing their requests.

agent

A process, usually corresponding to a particular managed object, that carries out requests from a manager.

agent schema

A formatted description of the groups and attributes available from an agent, with an enumerated list of possible agent errors. This description enables manager processes to make requests for specific groups and attributes.

agent services

The function library that allows agents and managers to communicate, as viewed by the agent.

API

Application Programming Interface. The function library interface an application can call to perform a particular service.

attribute	A named data item corresponding to a property of a managed object.
attribute name	The name of an attribute as defined by the agent schema.
bus	A non point-to-point network type. Used by the MDB to represent a network link to which many other elements may connect.
cluster	A collection of records in the MDB which together represent an element. Cluster records are defined in instance files. Also called cluster record or instance record.
component	A type of network element corresponding to devices such as computers, operating systems, gateways, etc.
connection	A point-to-point network link connecting two other elements. Connections come in two types: regular and simple. Regular connections are full-fledged elements that can have properties, can be managed by agents, etc. Simple connections are a user-convenience feature that have a graphical representation but no database representation. (They don't have properties, can't be managed by agents, etc.)
Console	The window-based manager application that comes with SunNet Manager.
control	The ability to modify attribute values to change the characteristics of managed objects.
data report	Attributes routinely reported for analysis, as opposed to an event report.
decay to blue	The decay-to-blue feature pertains to events and traps that have reached a threshold and then fallen below it, or that have had an event or trap stop being reported. In response to these conditions, the glyph for the affected element turns blue.

discover function	Using this function, from the machine on which the SNM Console is running, the Discover Tool finds hosts, routers, networks, and Simple Network Management Protocol (SNMP) devices reachable from the Console machine.
dispatch function	Agent routine that calls an appropriate agent function to service a request.
element	An individual object subject to management. Examples: “the machine called mgrhost,” “ethernet interface le0,” “building 14,” and so on.
element type	A structural definition of a particular kind of element. Examples: <code>component.sun3</code> , <code>bus.ethernet</code> , etc.
element category	A family of element types. The SunNet Manager Console defines four element categories: bus, component, connection and view.
event	The occurrence of a particular change in the state of an attribute.
event dispatcher	Process that acts as a rendezvous for event reports. Management applications can ask this daemon to send them selected copies of reports based on particular criteria.
event report	The notification an agent sends when an event is detected on a managed object.
glyph	A pictorial representation. Similar to icon (except OPEN LOOK distinguishes between the two). The Console provides three forms of glyphs: the fixed-sized image (components, views and connections without both end elements in the current view), the variable length vector with grab handles (buses) and the variable length vector (connections with both end elements in the current view).
group	A collection of related attributes. Defined by the agent writer and specified in the agent schema.

instance	The data representing a particular element. As used by the MDB, definitions of instances are called instance records.
instance file	A collection of agent request and element instance records.
managed object	A particular resource (computer system, network interface, etc.) subject to management. In the Console this concept is called an element.
MDB	Management database. The collection of all managed object attribute definitions, along with data definitions specific to the SunNet Manager Console.
manager	The collection of programs or processes that work on behalf of a user to help manage a particular set of networks and devices. Usually sends requests to agents, accepts collected data from them, and displays the data for the user. Also called manager application or manager process.
manager services	The function library that allows agents and managers to communicate, as viewed by the manager.
manager station	The workstation where a human manages a particular set of networks and devices, usually running a manager application such <code>snm(1)</code> or <code>snm_cmd(1)</code> .
MIB	Management Information Base. The logical representation of network devices and their components as managed objects.
monitor function	Using this function, the Discover Tool compares the elements stored in the runtime database with the elements it finds at a specified interval or specified time. If new elements are detected, the monitor function stores the elements in a “holding area” view and records these elements in a log file. Through the same log file, the monitor function can also notify you if a previously discovered host is down or was down within a given period.

predefined request	Provides the user with an easy and quick method of making network management calls from a manager to an agent process, usually to collect and return specific attribute values.
proxy agent	An agent that manages on behalf of the manager. Used to manage objects not directly manageable due to different protocol suites or other incompatibilities. Sometimes shortened to proxy.
proxy system	The host where a proxy agent runs.
rendezvous	Process where an agent sends data and event reports. Also known as rendezvous process.
report	Answer (usually from an agent to a manager) containing attribute values or error messages. Sometimes called response.
request	Network management call from a manager to an agent process, usually to collect and return specific attribute values.
RPC	Remote Procedure Call.
schema	A description of element structures and instances. While technically every file (except icons and rasterfiles) that the Console loads into the runtime database is a schema, in practice, records that define the structure of elements are called structure definitions, and records that define the actual elements are called instance records. The remaining schema records, used by agents to define their capabilities, are called agent schema.
schema file	A UNIX file containing one or more agent schema. Also called agent schema file.
SNMP	Simple Network Management Protocol, the standard network management protocol used in TCP/IP networks.

SNMPv2

Simple Network Management Protocol Version 2, an enhanced version of the standard network management protocol used in TCP/IP networks.

stream

A logical grouping of reports created according to the name of the requesting agent, type of report, and the request timestamp.

structure file

A collection of agent and element structure definitions.

system

An instance of a component. Sometimes called host or device.

target system

The host where the managed object resides.

verify function

Agent routine that checks the validity of a request before the request is dispatched.

view

A collection of components in the Console's MDB. Views have members, called elements. Since views are elements, they may be members of other views. Elements may have multiple memberships. Memberships may be circular.

Index

Symbols

\$HOME/.SNMdefaults file, 1-16
\$HOME/.SNMpredefined, 3-5
\$HOME/.SNMpredefined file, 1-16
\$SNMHOME/defaults file, 9-24
.SNMdefaults file, 9-24
.SNMpredefined file, 1-16
/opt/SUNWconn/snm/agents
 directory, 11-26
/opt/SUNWconn/snm/struct
 directory, 11-26
/tmp/snm_gr.rpcid. file, 7-3
/usr/snm/agents directory, 11-26
/usr/snm/struct directory, 11-26
/usr/snm/struct/.SNMpredefined
 file, 3-5
/var/adm/snm directory, 1-13, 11-26
/var/adm/snm/event.log file, 6-1
/var/adm/snm/snmp.traps, 11-12
/var/opt/snm directory, 1-13, 11-26
/var/opt/snm/snmp.traps, 11-12
~/.SNMpredefined file, 3-5

Numerics

5.x directory, 1-14

A

About option, 2-7
absolute values
 graphing data sets, 7-6
 graphing time stamps, 7-6
active request state, 2-46
activity daemon, 1-4, 1-12, 1-15
 definition of, Glossary-1
 occurrences of, 1-13
activity.log file, 1-15
Add button
 data report request, 3-15, 3-22
adding element types, 10-1
adding user commands, 10-1
agent, 1-2, 1-4
 definition of, Glossary-1
 security, 13-4
agent configuration file, 1-13
agent files, 1-14
agent header files, 1-14
agent processes, 1-8
agent record, 10-11
agent reports, 6-8
agent schema, 1-5
 definition of, Glossary-1
agent schema files, 10-2

agent services
 definition of, Glossary-1

agents, 1-6
 using Discover for custom ..., 5-26
 writing your own, 1-12

agents directory, 1-14, 11-12

agents for Sun products, 1-12

alarm report
 definition of, 4-9
 filtering, 4-16
 finding agent-specific, 4-19
 printing device-specific, 4-20
 saving device-specific, 4-19
 sorting, 4-17
 viewing device-specific, 4-11, 4-14

Alarm Reports Summary window, 4-11

Alarms Reports
 viewing, 2-10

Alias button, 2-34, 2-37

Alias name, 2-38

Aliases List, 2-38

API
 definition of, Glossary-1

APIs, 1-8

application programming interface,
 database, 1-5

Apply button
 Console properties, 9-4
 data report request, 3-31
 Request Builder template, 3-24, 3-26

architecture overview, 1-2

asynchronous event reports (traps), 11-7,
 11-11

attribute
 data report request, 3-14, 3-30
 definition of, Glossary-2

attribute name
 definition of, Glossary-2

attribute values
 data report request, 3-13, 3-28
 event report request, 3-18, 3-36

attributes, 1-5

Audio Effect, 9-11

Audio File, 9-11

audio file, 3-21

audio signal for trap, 9-17

audio signal options
 event report request, 3-21
 play audio file, 3-21
 ring bell, 3-21

Automatic Management, 9-10

automatic management, 9-7
 audio effect, 9-11
 audio file, 9-11
 event priority, 9-11
 glyph effect, 9-11
 polling interval, 9-11
 send mail, 9-12
 send to program, 9-12

Automatic Management category of
 Console Properties, 9-7

automatic node management, 9-7
 enabling, 9-10

available attributes
 event report request, 3-18

awaiting activation request state, 2-46

awaiting stop request state, 2-46

B

background image — definition, 10-12

background images — definition, 10-2

banner
 Console, 1-4

being activated request state, 2-46

bin directory, 1-14

binary files, 1-14

blink glyph signal option, 3-20

blink system glyph, 2-31

blinking glyph
 priority, 9-15

Browser, 1-6, 2-24
 agent reports, 6-8
 customizing, 6-17
 file data format, 6-1
 folders, 6-15

- loading a file, 6-6
- report streams, 6-7
- sending data to Grapher, 6-13
- starting, 6-3
- build
 - Predefined Data Request, 2-22
 - Predefined Event Request, 2-22
- build.oid utility, 11-25
- build_oid utility, 11-25, 11-26, 12-2
- building
 - predefined data and event requests, 3-7
- bus
 - definition of, Glossary-2
- buses, 10-3
 - Ethernet LAN, 10-3
- buttons
 - Edit, 2-12
 - File, 2-2
 - Goto, 2-24
 - Props, 2-18
 - Tools, 2-23
 - View, 2-8

C

- Change button
 - data report request, 3-15, 3-22
- changing element type, 2-41
- clipboard, 2-11, 2-13
- cluster
 - definition of, Glossary-2
- cluster record, 10-14
- cluster records, 10-9
- color
 - element, 2-37
 - priority, 9-15
- color of graph, 7-6
- colormap entries, 9-5
- colors
 - specifying .. for discovered components, 5-23
- command-line interface, 1-20
- community string, 8-5
- completion of request, 3-13, 3-28
- component
 - changing element type, 2-41
 - definition of, Glossary-2
- component element type definition, 10-6
- components, 10-3
 - printers, 10-3
 - routers, 10-3
 - workstations, 10-3
- conferring right-of-access, 13-4
- configuration
 - Console, 9-1
- configuring colormap entries, 9-5
- configuring error effect, 9-19
- configuring error signals, 9-19
- configuring event effect, 9-14
- configuring Goto button operation, 9-23
- configuring horizontal scrolling, 9-6
- configuring icon fonts, 9-5
- configuring location of icon
 - directories, 9-22
- configuring request timeout, 9-7
- configuring Restart field default, 9-6
- configuring schema file locations, 9-21
- configuring trap effects, 9-16
- configuring username display, 9-5
- configuring vertical scrolling, 9-6
- configuring window title, 9-5
- connect record, 10-11
- connecting elements, 2-39
- connecting two glyphs, 2-39
- connection
 - definition of, Glossary-2
 - simple vs. manageable distinction, 5-10
- connection— definition, 10-12
- connection element type definition, 10-4
- connections, 10-3
 - leased internetwork link, 10-3
 - RS-232-C line, 10-3
- connections — definition, 10-2

- Console
 - definition of, Glossary-2
 - features, 1-4
 - multiple instances, 1-4
 - user name, 1-4
- Console configuration, 9-1, 9-24
- Console customization, 9-24
- Console lock file, 1-15
- Console operation
 - Edit button, 2-12
 - Glyph menu, 2-26
 - Goto button, 2-24
 - Props button, 2-18
 - Requests button, 2-21
 - Tools button, 2-23
 - View button, 2-8
- console operation
 - File button, 2-2
- Console Properties, 9-3
 - Category menu, 9-3
 - Events and Traps, 9-12
 - fonts for glyph labels, 9-5
 - Locations, 9-21
 - Miscellaneous, 9-22
 - Requests, 9-6
 - Windows
 - Display Username in Title, 9-5
 - Horizontal Scrolling, 9-6
 - Icon Font, 9-5
 - Maximum Colors, 9-5
 - Vertical Scrolling, 9-6
 - Window Title, 9-5
- Console properties, 2-18
 - Apply button, 9-4
 - Automatic Management, 9-7
 - effects, 9-4
 - Errors, 9-18
 - Reset button, 9-4
 - Windows, 9-5
- Console properties categories, 9-4
- control
 - definition of, Glossary-2
- control panel of Set Tool, 8-4

- Copy
 - editing, 2-13
- copying elements, 2-13
- count, 3-11, 3-27
- Create
 - editing, 2-14
- creating elements, 2-14
- customizing Console, 9-24
- Cut
 - editing, 2-13
- cutting elements, 2-13

D

- daemon configuration file, 1-13
- daemons, 1-4, 1-12
- data log
 - data report request, 3-14, 3-30
 - examining, 4-1
- data report, 1-4
 - definition of, Glossary-2
 - indicator, 4-3
 - security, 13-1
 - Strip Chart, 4-4
- Data Report entry format, 4-2
- data report request
 - Add button, 3-15, 3-22
 - Agent Schema, 3-11
 - Apply button, 3-31
 - attribute values, 3-13, 3-28
 - Change button, 3-15, 3-22
 - count, 3-11, 3-27
 - data log, 3-14, 3-30
 - defer reports, 3-12, 3-28
 - Define button, 3-16, 3-22
 - Delete button, 3-15, 3-22, 3-31
 - disposition of request, 3-13, 3-28
 - file name, destination of ASCII text, 3-13, 3-29
 - Grapher Tool, 3-15, 3-31
 - group/table, 3-11
 - Hold button, 3-31
 - indicator, 3-14, 3-30
 - interval, 3-27

- key, tabular information, 3-11, 3-27
- naming a request, 3-10, 3-27
- option string, 3-12, 3-29
- polling interval, 3-11
- proxy system, 3-27
- Request Name, 3-10
- Reset button, 3-16, 3-22, 3-31
- restart, 3-12, 3-27
- specify a particular attribute, 3-14, 3-30
- Start button, 3-31
- strip chart, 3-15, 3-30
- Undefine button, 3-16, 3-23
- data reporting, 1-5
- Data Reports
 - viewing, 2-10
- data reports
 - maximum number, 9-23
- Data Reports window, 4-1
 - maximum number, 9-23
- data request — definition, 10-15
- Data Request template, 3-26
- data requests — definition, 10-2
- data sets
 - graphing, 7-6
 - order in graphing, 7-7
- database, 1-5
 - changes, 10-1
 - definitions, 10-1
- database API, 1-5
- database API lock file, 1-15
- database change traps, 9-18
- database directory, 1-13, 1-15
- dataRequest record, 10-14
- Default file locations, summary of, 1-19
- default request timeout, 9-7
- default value of Restart field, 9-6
- defer reports, 3-12, 3-28
 - sending, 2-51
- Define button
 - data report request, 3-16, 3-22
- definition
 - glyph, 10-6
- Delete
 - editing, 2-14
- Delete button
 - data report request, 3-15, 3-22, 3-31
- deleting elements, 2-14
- dim glyph signal option, 3-20
- dim system glyph, 2-31
- dimmed glyph
 - priority, 9-15
- directories
 - log file, 1-13
- directories created at installation, 1-13
- Discover, 2-24
- Discover configuration window, 5-4
- discover function, 5-1
 - full discover mode, 5-1
- discover function configuration
 - parameters, 5-7
- Discover Properties
 - Add Object Connections, 5-10
 - Add To All Views, 5-10
 - Default Proxy, 5-11
 - ICMP Retries, 5-9
 - ICMP Timeout, 5-9, 5-16
 - Maximum Hops, 5-8
 - Net Name/Number, 5-7
 - Netmask, 5-8
 - Objects to Discover, 5-12
 - SNMP Community, 5-9
 - SNMP Retries, 5-9
 - SNMP Timeout, 5-9
 - Verbose Mode, 5-9, 5-17
 - Viewname, 5-10
- Discover Tool, 1-4, 1-6
 - invoking, 5-2
- Monitor
 - Cycle Time, 5-16
 - Holding Area Viewname, 5-16
 - Ignore Host File Name, 5-17
 - Log Filename, 5-16
 - Log History, 5-16
 - Mail Logfile To, 5-17
 - Response Timeout, 5-16
 - Start Date, 5-17

- Start Time, 5-17
- Stop Date, 5-18
- Stop Time, 5-17
- monitor function, 5-1, 5-14
- using with large subnetworks, 5-21
- discover.conf file
 - location and description of, 5-21
- Display username, 9-5
- displaying empty views, 9-23
- displaying request properties, 2-48
- disposition of request upon
 - completion, 3-13, 3-28
- distributing proxy agents, 1-6

E

- Edit button, 2-12
- Edit menu
 - Copy, 2-13
 - Create, 2-14
 - cut, 2-13
 - Delete, 2-14
 - Paste, 2-13
 - Select all, 2-13
- effect of event, 9-15
- effect propagation, 9-14
- effects of Console properties changes, 9-4
- element
 - definition of, Glossary-3
- element category
 - definition of, Glossary-3
- element Glyph menu
 - Change Type, 2-41
 - Connect, 2-39
 - Glyph State, 2-31
 - Properties, 2-34
 - Quick Dump, 2-27
 - Send Predefined, 2-28
 - Send Request, 2-28
 - Set request, 2-28
 - Show Subview, 2-32
 - Tools, 2-30
- element instance —definition, 10-2, 10-9
- element instance definition
 - agent record, 10-11
 - connect record, 10-11
 - glyphState record, 10-12
 - instance type record, 10-10
 - membership record, 10-10
 - proxy record, 10-11
- element instance names, 10-10
- element schema files, 10-2
- element type
 - changing, 2-41
 - definition of, Glossary-3
- element type — definition, 10-2, 10-3
- element type categories, 10-3
- element type definition
 - connections, 10-4
 - Glyph_State, 10-4
 - Label, 10-5
 - Name field, 10-4
 - user command, 10-8
- element type definition file, 1-14
- element type definition record, 10-3
- element types, 10-1
- elementCommand instance, 10-8
- elementGlyph instance, 10-6
- elements
 - copying, 2-13, 2-14
 - creating, 2-14
 - cutting, 2-13
 - deleting, 2-14
 - pasting, 2-13
 - selecting, 2-13
- elements.schema file, 1-14, 7-3, 10-3
- empty views, 9-23
- enabling automatic node
 - management, 9-10
- enabling propagation of effect, 9-14
- enterprise object identifier, 11-15
- enterprise object identifier (OID), 11-11
- enterprises.oid file, 11-26
- enterprise-specific trap, 11-15
- enterprise-specific traps, 11-11, 11-14

- environment variables, 1-17
 - HELPPATH, 1-17
 - HOME, 3-36
 - MANPATH, 1-17
 - SNM_NAME, 1-15
 - SNMDBDIR, 1-13, 1-15
 - SNMDISCOVERMAP, 1-18
 - SNMHOME, 1-13
 - SNMLINKMAP, 1-18
- error effect, 9-19, 9-20
 - propagating, 9-20
- error messages
 - advisory, A-4
 - API error, A-4
 - error, A-4
 - internal error, A-4
 - warning, A-4
- Error Reports
 - maximum number, 9-23
 - viewing, 2-10
- Error Reports window
 - maximum number, 9-23
- error signals, 9-19
- Errors category of Console
 - Properties, 9-18
- event
 - definition of, Glossary-3
- Event Dispatcher, 11-15
- event dispatcher, 1-4, 1-12, 1-15
 - definition of, Glossary-3
 - occurrences of, 1-13
- event effect, 9-15
- event notification
 - audio signal options, 3-21
 - mail options, 3-21
 - program options, 3-22
- Event Priority, 9-11
- event report
 - definition of, Glossary-3
 - security, 13-1
- event report request
 - agent schema, 3-11
 - attribute values, 3-18, 3-36
 - audio effect, 3-21
 - available attributes, 3-18
 - count, 3-11, 3-27
 - defer reports, 3-12, 3-28
 - disposition of request, 3-13, 3-28
 - file name, destination of ASCII
 - text, 3-13, 3-29
 - glyph effect, 3-20
 - group/table, 3-11
 - interval, 3-27
 - key, tabular information, 3-11, 3-27
 - mail options, 3-21
 - naming a request, 3-10, 3-27
 - option string, 3-12, 3-29
 - polling interval, 3-11
 - priority, 3-20
 - program options, 3-22
 - proxy system, 3-27
 - relation, 3-18
 - Request Name, 3-10
 - restart, 3-12, 3-27
 - threshold, 3-20
 - threshold conditions, 3-19
 - threshold value, 3-20
 - visual signal options, 3-20
- event report requests
 - automatic, 9-7
- event reporting, 1-4, 1-5
 - propagation, 1-5
- event reports
 - maximum number, 9-24
- event request — definition, 10-18
- event requests — definition, 10-2
- event.log file, 1-15
- event/trap Log
 - interpreting traps, 11-24
- event/trap log
 - examining, 4-7
- Event/Trap Reports
 - viewing, 2-10
- Event/Trap Reports window, 4-6, 11-24
 - maximum number, 9-24
- eventRequest record, 10-14
- events
 - decay to blue, 3-20, 9-16

- Events and Traps category of Console
 - Properties, 9-12
- events.ind file, 1-15
- events.rec file, 1-15
- examining a request, 2-44
- examining data reports log, 4-1
- examining event/trap log, 4-7
- example database, 1-15
- example.db file, 1-15

F

- fatal errors, 9-19
- File button, 2-2
- file formats, 6-1
- File locations, summary of, 1-19
- files
 - loading into Console, 2-5
 - saving from Console, 2-5
- files created at installation, 1-13
- finding elements, 2-10
- folders, 6-15
 - creating, 6-15
 - deleting, 6-16
 - emptying, 6-16
- forking programs for element creation or modification, 9-25

G

- Get button
 - Set Tool, 8-4
- glyph
 - definition of, Glossary-3
 - returning to normal color, 3-21, 9-16
- glyph definition, 10-6
- Glyph Effect, 9-11, 9-16
- glyph effect
 - event report request, 3-20
- Glyph Menu
 - Quick Dump, 2-27
- Glyph menu, 2-26
 - Change Type, 2-41

- Connect, 2-39
- Glyph State, 2-31
- Properties, 2-34
- Send Predefined, 2-28, 3-23, 3-36
- Send Request, 2-28, 3-23, 3-24
- Set Request, 8-1
- Set request, 2-28
- Show Subview, 2-32
- Tools, 2-30
- glyph menu
 - Glyph State, 9-15
- glyph state
 - blinking, 9-15
 - changing, 2-31, 9-15
 - color, 9-15
 - dimmed, 9-15
 - priority, 9-15
 - resetting, 9-15
 - view, 9-15
- Glyph State option, 2-31, 9-15
- Glyph_State, 10-4
- glyphs, 10-1
 - positioning, 2-17
 - request, 2-48
- glyphState record, 10-12
- Goto button, 2-24, 9-23
- graph
 - data report request, 3-15, 3-31
- Graph Properties window, 7-4
- Graph Tool, 4-6
- Grapher, 1-6, 2-24
 - b option, 7-3
 - displaying graphs, 7-8
 - invoking from the command line, 7-3
 - Merge button, 7-4
 - properties, 7-4
 - Properties button, 7-4
 - Remove button, 7-4
 - starting, 7-2
 - temporary RPC program number, 7-3
 - View button, 7-4
- Grapher Window, 7-3
- graphical editing, 2-12

- graphical editor
 - copying elements, 2-13
 - creating elements, 2-14
 - cutting elements, 2-13
 - deleting elements, 2-14
 - pasting elements, 2-13
 - selecting all elements, 2-13

- graphs
 - background color, 7-3, 7-9
 - base color, 7-6
 - data scale, 7-6, 7-9
 - data set order, 7-7
 - dimension specification, 7-7
 - displaying, 7-8
 - drawing style, 7-7
 - fill, 7-7
 - frame color, 7-6
 - hide, 7-7
 - line color, 7-8
 - line style, 7-8
 - merging, 7-11
 - name, 7-6
 - plotting values, 7-8
 - replotting, 7-10
 - show, 7-7
 - start and end times, 7-6
 - time scale, 7-6, 7-10
 - viewing angles, 7-10
 - zooming, 7-10

- group
 - definition of, Glossary-3

H

- header files, 1-14
- help file, 1-17
- HELPPATH environment variable, 1-17
- hiding graphs, 7-7
- Hold button
 - data report request, 3-31
- Horizontal Scrolling, 9-6

I

- icon directories, 9-22

- icon files, 1-14

- Icon Font, 9-5

- icons directory, 1-14

- include directory, 1-14

- Indicator

- data report, 4-3

- indicator

- data report request, 3-14, 3-30

- installation directory, 1-13

- instance

- definition of, Glossary-4

- instance file

- definition of, Glossary-4

- instance files, 10-2

- instance type record, 10-10

- interpreting traps, 11-24

- interval

- request properties, 3-27

- invoking Discover Tool, 5-2

- Console Tools window, 5-2

- SunOS command line, 5-2

K

- key, 3-11, 3-13, 3-27, 3-29, 8-4

- killing a request, 2-50

L

- Label, 10-5

- launch

- Predefined data requests, 2-26

- Quick Dump requests, 2-26

- standard requests, 2-26

- lib directory, 1-14

- library files, 1-14

- line color of graphs, 7-8

- line graphs, 7-7

- line style of graphs, 7-8

- link management

- interaction with Discover Tool, 5-10

- linkmap file, 1-16, 1-18, 5-10

- list of supplied Predefined Data requests, 3-6
- list of supplied Predefined Event requests, 3-7
- Load menu
 - Predefined Requests option, 2-6, 2-7
- Load option
 - ASCII-format files only, 2-6
 - Predefined Requests, 2-7
- Load window, 2-5, 9-23
- localhost, 2-36
- locations
 - icons directories, 9-22
 - schema files, 9-21
- Locations category of Console Properties, 9-21
- lock file, 1-15
- log file directory, 1-13
- log file locations, 1-14
- log files
 - directory, 1-14
 - installation, 1-14

M

- mail options
 - event report request, 3-21
- mailing trap report, 9-18
- man directory, 1-14, 1-17
- man pages directory, 1-14
- managed object
 - definition of, Glossary-4
- management database, 1-5
 - changes, 10-1
 - definitions, 10-1
- manager, 1-2
 - definition of, Glossary-4
- manager ID, 1-8
- manager IP address, 1-8
- manager services
 - definition of, Glossary-4
- manager station
 - definition of, Glossary-4

- manager-agent communications, 1-8
- manager-agent model, 1-2
- MANPATH environment variable, 1-17
- Maximum Colors, 9-5
- maximum data reports, 9-23
- maximum error reports, 9-23
- maximum event reports, 9-24
- maximum number of requests, 1-8
- maximum number of subprocesses, 1-8
- maximum trap reports, 9-24
- MDB, 1-5
 - definition of, Glossary-4
- MDB files, 1-5
- membership record, 10-10
 - positioning coordinates, 10-11
 - view name, 10-11
- menu
 - Edit, 2-12
 - file, 2-2
 - Glyph, 2-26
 - Goto, 2-24
 - Requests, 2-21
 - Tools, 2-23
 - View, 2-8
- merging graphs, 7-11
- MIB
 - definition of, Glossary-4
- Miscellaneous category of Console Properties, 9-22
- modifying a request, 2-44, 2-50
- modifying element type glyphs, 10-1
- modifying element types, 10-1
- modifying user commands, 10-1
- monitor function, 5-1
 - time parameters for, 5-18
- monitor.log file, 1-15
- multiple Console instances, 1-4

N

- na.activity, 1-13
- na.event, 1-13

- na.hostif, 1-10
- na.logger agent, 1-15
- na.snmp proxy agent, 11-2
- na.snmp-trap daemon, 11-12
- name
 - data report request, 3-10, 3-27
- Name field of element type
 - definition, 10-4
- name of graph, 7-6
- nc.ind file, 1-15
- nc.rec file, 1-15
- netmgt_db.h file, 10-3
- New Folder, 6-15
- number of requests, 2-46

O

- object identifier, 11-25
- Object Identifier Database, 11-25
- object IDs, 11-25
- Object1 field of element type
 - definition, 10-4
- Object2 field of element type
 - definition, 10-4
- occurrences of activity daemon, 1-13
- occurrences of event dispatcher, 1-13
- OID
 - mapping to element types in Discover, 5-23
- oid.dbase database, 11-26
- one-time data report, 2-22, 3-4
- opening Console window, 9-14
- OpenWindows, 1-4
- option string, 3-12, 3-29, 8-5
- Options field
 - Set Tool, 8-5
- order of data sets
 - graphing, 7-7

P

- Paste
 - editing, 2-13
- pasting elements, 2-13
- PATH environment variable, 1-17
- per-product configuration, 9-24
- per-user configuration, 9-24
- play audio file signal option, 3-21
- plotting values in graphs, 7-8
- Polling Interval, 9-11
- polling interval
 - request properties, 3-11
- position of glyphs, 2-17
- positioning coordinates in membership record, 10-11
- predefined data and event requests, 1-16, 3-1
- predefined event report requests, 9-7
- predefined request, Glossary-5
- predefined requests, 1-16, 2-21, 2-22
- printing Console screens, 2-24
- priority
 - blinking, 9-15
 - color, 9-15
 - dimmed, 9-15
 - event report request, 3-20
 - glyph state, 9-15
- priority sets color signal option, 3-20
- processes
 - agent, 1-8
 - request, 1-8
- program directories, 1-13
- program files, 1-13
- program options
 - event report request, 3-22
- propagating error effect, 9-20
- propagation
 - blinking priority, 9-15
 - color priority, 9-15
 - dimmed priority, 9-15
 - glyph state priority, 9-15
- propagation of event effect, 9-14

propagation of events, 1-5
propagation of trap effect, 9-17
properties
 Console, 2-18
 element glyph, 2-34
properties window
 agent schema list, 2-36
 color, 2-37
 component data, 2-35
Props button, 2-18
proxy agent, 1-4
 default location, 2-36
 definition of, Glossary-5
proxy agents, 1-6
 distributing, 1-6
proxy record, 10-11
proxy system, 1-6
 definition of, Glossary-5
 request properties, 3-27
pseudo-device
 element definition for, 11-21
pseudo-devices, 11-17

Q

Quick Dump pop-up window, 3-4
Quick Dump request, 1-8, 3-3
Quick Dump results, 3-4
quit Console verification, 9-23

R

RCP/XDR, 1-8
read-security, 13-1
record
 element type definition, 10-3
redirecting SNMP requests, 9-26
relation
 event report request, 3-18
relative values
 graphing data sets, 7-6
 graphing time stamps, 7-6
Remote Procedure Call (RPC), 1-8

rendezvous
 definition of, Glossary-5
rendezvous IP address, 1-8
rendezvous process, 1-12
rendezvous RPC program number, 1-8
rendezvous RPC version number, 1-8
replotting graphs, 7-10
report
 definition of, Glossary-5
Report menu, 6-10
report streams, 6-7
request
 definition of, Glossary-5
Request Builder template
 Apply button, 3-24, 3-26
request glyph, 2-48
request name, 2-46
request properties window, 2-48
request restart interval, 9-7
request state, 2-46
request timeout, 9-7
request timestamp, 2-46
request type, 2-45
Request Viewer window, 2-44
 Props button, 2-48
 selecting requests, 3-28
 Sort menu, 2-48
request.log file, 1-15
requests, 1-4
 numbers of, 2-46
requests — definition, 10-14
Requests button, 2-21
Requests category of Console
 Properties, 9-6
Requests menu, 2-21
 Create Predefined, 2-22, 3-23, 3-36
 Create Predefined option, 3-8
 Quick Dump, 2-22, 3-2
 Send Request, 2-22, 3-23, 3-25
 Summary, 2-22
Reset button
 data report request, 3-16, 3-22, 3-31

Rest button
 Console properties, 9-4
 Restart, 9-6
 restart interval, 9-7
 restart requests, 1-15, 3-12, 3-27
 restarting failed requests, 9-7
 Results Browser, 1-6, 2-24
 agent reports, 6-8
 customizing, 6-17
 features, 1-4
 folders, 6-15
 loading a file, 6-6
 report streams, 6-7
 Results Grapher, 1-6, 2-24, 4-6
 -b option, 7-3
 displaying graphs, 7-8
 features, 1-4
 invoking from the command line, 7-3
 Merge button, 7-4
 properties, 7-4
 Properties button, 7-4
 Remove button, 7-4
 starting, 7-2
 View button, 7-4
 Results Grapher Window, 7-3
 retry-interval for SNMP proxy, 11-6
 reverse order
 graphing datasets, 7-7
 ribbon graphs, 7-7
 right-of-access, 13-4
 ring bell signal option, 3-21
 rlogin, 2-30
 RPC
 definition of, Glossary-5
 RPC program number (for Grapher), 7-3
 runtime database, 1-5, 10-2
 runtime database API, 1-5
 runtime database files, 1-15

S
 Save menu
 Management Database option, 2-6
 Predefined Requests option, 2-7
 Save window, 9-23
 schema
 definition of, Glossary-5
 schema file
 definition of, Glossary-5
 schema file locations, 9-21
 schema files, 1-14
 agent, 10-2
 element, 10-2
 relationship among .. shipped with
 SNM, 11-3
 schema list, 2-36
 schemas
 SNMP, 11-3
 scrolling configuration, 9-6
 security, 13-1
 security algorithm, 13-3
 security levels, 13-2, 13-4
 Select all
 editing, 2-13
 selecting all elements, 2-13
 selecting requests, 3-28
 selecting settable agents
 Set Tool, 8-4
 selecting settable groups
 Set Tool, 8-4
 send
 Data request, 2-22, 2-28, 3-23
 deferred reports, 2-51
 Event request, 2-28, 3-23
 Predefined Data request, 2-28
 Predefined Event request, 2-28
 Quick Dump, 3-2
 Quick Dump request, 2-27
 Send Mail signal option, 3-21
 Send Mail to, 9-12
 Send to Program, 9-12
 Send to Program signal option, 3-22
 sending deferred reports, 2-51
 sending mail
 error report, 9-20

-
- sending reports
 - deferred, 3-12, 3-28
 - sending to program
 - error report, 9-21
 - sending trap report to program, 9-18
 - Set button
 - Set Tool, 8-4
 - Set Information list, 8-5
 - Set request
 - security, 13-1
 - setting attribute values, 2-28
 - Set Tool
 - Agent field, 8-4
 - attribute list, 8-4
 - control panel, 8-4
 - control panel buttons, 8-4
 - control panel fields, 8-4
 - Get button, 8-4
 - Group field, 8-4
 - how to invoke, 8-1, 8-6
 - invoking from command line, 8-6
 - invoking from Console, 8-1
 - Key field, 8-4
 - Options field, 8-5
 - selecting settable agents, 8-4
 - selecting settable groups, 8-4
 - Set button, 8-4
 - Set Information list, 8-4, 8-5
 - Unset button, 8-4
 - window functions, 8-4
 - Show Subview option, 2-32
 - showing graphs, 7-7
 - signal options
 - audio, 3-21
 - blink glyph, 3-20
 - dim glyph, 3-20
 - mail options, 3-21
 - play audio file, 3-21
 - priority sets color, 3-20
 - program, 3-22
 - ring bell, 3-21
 - Send Mail, 3-21
 - Send to Program, 3-22
 - Simple Network Management Protocol (SNMP), 1-5
 - SNMP hosts file
 - use for trap attribution, 11-19
 - Snapshot, 2-24
 - snm directory, 1-13
 - SNM libraries, 1-14
 - SNM log files
 - installation, 1-14
 - snm+lock file, 1-15
 - snm.conf file, 1-8, 1-13, 1-14, 1-15, 11-4, 11-5, 11-6, 11-7, 11-9, 11-11, 11-12, 11-13
 - snm.glue file, 1-15, 10-2
 - snm/bin directory, 1-17
 - snm_cmd command, 1-15, 1-20
 - snm_discover command, 5-2
 - SNM_NAME environment variable, 1-15, 1-17
 - snmdb+lock file, 1-15
 - SNMDBDIR environment variable, 1-13, 1-15, 1-18, 11-26
 - SNMDEFAULTS environment variable, 9-24
 - SNMDISCOVERMAP environment variable, 1-18
 - SNMHOME environment variable, 1-13, 1-17
 - SNMLINKMAP environment variable, 1-18
 - SNMP, 1-5
 - community string, 8-5
 - definition of, Glossary-5
 - SNMP devices
 - specifying color for, 5-24
 - SNMP proxy
 - enterprise-specific traps, 11-14
 - receiving responses, 11-6
 - retry-interval, 11-6
 - trap handling, 11-11
 - SNMP proxy agent, 11-1, 12-2
 - SNMP proxy agents, 1-8

SNMP requests
 redirecting to another proxy, 9-26

SNMP schemas, 11-3

SNMP system description
 mapping to element type, 5-25

SNMP trap daemon, 11-12, 11-16

SNMP trap file, 11-14, 11-16
 glyph entries in, 11-18

SNMP traps, 11-11
 interpreting, 11-24

snmp.hosts, 11-10

snmp.hosts file, 1-16
 when required, 11-5, 11-9

snmp.oid file, 11-26

snmp.schema, 11-3

snmp.trapfile file, 1-16

snmp-mibII.oid file, 11-26

snmp-mibII.schema, 11-3

snmp-mibII.schema file, 11-13

SNMPv2 agent (snmpv2d) process, 12-4

SNMPv2 configuration file
 agents, 12-4
 mgrs.v1, 12-4
 mgrs.v2, 12-4

SNMPv2 data type
 Counter64, 12-3
 UInteger32, 12-3

SNMPv2 enhancements
 Manager-to-manager capability, 12-2
 Protocol operations, 12-2
 Security, 12-2
 Structure of Management Information (SMI), 12-2

SNMPv2 GetBulkRequest PDU, 12-3

SNMPv2 InformRequest PDU, 12-3

SNMPv2 installation, 12-4

SNMPv2 installation script
 (v2install), 12-4

SNMPv2 managed objects, 12-2

SNMPv2 manager (na.snmpv2)
 process, 12-4

SNMPv2 manager-to-manager MIB, 12-3

SNMPv2 mibs, 12-5

SNMPv2 new PDUs, 12-3

SNMPv2 OBJECT-TYPE macro, 12-3

SNMPv2 proxy agent, 12-1

SNMPv2 security context concept, 12-4

SNMPv2 security party concept, 12-4

SNMPv2 trap PDU, 12-3

snmpv2d daemon, 12-2

solid graphs, 7-7

sorting requests, 2-48

source code for selected agents, 1-14

specifying key, 8-4

specifying options string, 8-5

src directory, 1-14

Start button
 data report request, 3-31

stopped request state, 2-46

stopping request state, 2-46

stream
 definition of, Glossary-6

streams
 copying to folder, 6-16
 selecting, 6-12

Streams menu, 6-11

Strip Chart
 data report, 4-4
 data report request, 3-15, 3-30

Strip Chart Properties window, 4-6

struct directory, 1-14

structure file
 definition of, Glossary-6

structure files, 10-2

style of graphs, 7-7

subviews
 changing, 2-32

summary
 Data and Event reports, 2-22

Sun products with agents, 1-12

SunNet Manager Bulletin Board, 2-7

sun-snmp.oid file, 11-26

- supplied predefined data and event requests, 3-5
- switching views
 - through Alarm Reports window, 4-13
- system
 - definition of, Glossary-6

T

- target host, 2-46
- target system
 - definition of, Glossary-6
- TCP/IP, 1-2
- Telnet, 2-30
- three-dimensional graphs, 7-7
- threshold
 - event report request, 3-20
- threshold conditions
 - event report request, 3-19
- threshold value
 - event report request, 3-20
- time stamps
 - graphing, 7-6
- timeout for requests, 9-7
- title of Console windows, 9-5
- Tool Properties option, 6-17
- Tools
 - Glyph menu, 2-30
 - rlogin to target system, 2-30
 - Telnet, 2-30
- Tools button, 2-23
- Tools menu
 - Discover, 5-2
 - Grapher, 7-2
- Tools menu — definition, 10-2, 10-13
- trap audio signal, 9-17
- trap effect, 9-17
- trap effect propagation, 9-17
- trap file, 11-14
 - glyph entries, 11-18
- trap report
 - mailing, 9-18
 - sending to program, 9-18

- trap reports
 - maximum number, 9-24
- trap type
 - authenticationFailure, 11-11
 - coldStart, 11-11
 - egpNeighborLoss, 11-11
 - enterprise-specific, 11-11
 - linkDown, 11-11
 - linkUp, 11-11
 - warmStart, 11-11, 11-25
- Traps
 - when database file loaded, 9-26
- traps, 1-16, 11-11
 - attributing to pseudo-devices, 11-17
 - changes to database, 9-18
 - interpreting, 11-24
- two-dimensional graphs, 7-7
- type of request, 2-45

U

- Undefine button
 - data report request, 3-16, 3-23
- Unset button
 - Set Tool, 8-4
- user command definition, 10-8
- user commands, 10-1
- User Commands menu, 10-8
- username display, 9-5

V

- v2mib2schema program, 12-4
- verification of Console quit, 9-23
- verify function
 - definition of, Glossary-6
- Vertical Scrolling, 9-6
- view
 - Data and Event reports, 4-1
 - definition of, Glossary-6
 - Event/Trap reports, 4-1
 - glyph state, 9-15
- View button, 2-8
- view displayed, 9-14

view element type definition, 10-6

View menu

- Add Background, 2-10

- Alarms Reports, 2-10

- Clipboard, 2-11

- Data Reports, 2-10, 4-1

- Error Reports, 2-10

- Event/Trap Reports, 2-10, 11-23

- Events, 2-10

- Find, 2-10

- Remove Background, 2-10

- Requests, 2-44

view name in membership record, 10-11

viewBackground instance record, 10-12

viewing requests, 2-44

views, 10-3, 10-10

- background images, 10-12

- changing, 2-32

- glyph, 10-3

- navigating, 2-24

- using monitor function for non-
standard ..., 5-26

views, switching

- through Alarm Reports window, 4-13

visual signal options

- blink glyph, 3-20

- dim glyph, 3-20

- event report request, 3-20

- priority sets color, 3-20

Z

Window Title, 9-5

Windows category of Console

- Properties, 9-5

write-security, 13-1

X

X resource configuration, 9-24

X terminal, 1-4

X11 protocol, 1-4

X11 server, 1-4

Z

zooming graphs, 7-10

Copyright 1995 Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, Californie 94043-1100 USA.

Tous droits réservés. Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie et la décompilation. Aucune partie de ce produit ou de sa documentation associée ne peuvent être reproduits 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 en a.

Des parties de ce produit pourront être dérivées du système UNIX®, licencié par UNIX Systems Laboratories Inc., filiale entièrement détenue par Novell, Inc. ainsi que par le système 4.3. de Berkeley, licencié par l'Université de Californie. 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.

LEGENDE RELATIVE AUX DROITS RESTREINTS : l'utilisation, la duplication ou la divulgation par l'administration américaine sont soumises aux restrictions visées à l'alinéa (c)(1)(ii) de la clause relative aux droits des données techniques et aux logiciels informatiques du DFAR 252.227- 7013 et FAR 52.227-19.

Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevet(s) américain(s), étranger(s) ou par des demandes en cours d'enregistrement.

MARQUES

Sun, Sun Microsystems, le logo Sun, Solaris, SunSoft, le logo SunSoft, SunOS, OpenWindows, DeskSet, ONC, SNM, and NFS sont des marques déposées ou enregistrées par Sun Microsystems, Inc. aux Etats-Unis et dans certains autres pays. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays, et exclusivement licenciée par X/Open Company Ltd. OPEN LOOK est une marque enregistrée de Novell, Inc., PostScript et Display PostScript sont des marques d'Adobe Systems, Inc.

Toutes les marques SPARC sont des marques déposées ou enregistrées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. SPARCcenter, SPARCcluster, SPARCcompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC II et UltraSPARC sont exclusivement licenciées à Sun Microsystems, Inc. Les produits portant les marques sont basés sur une architecture développée par Sun Microsystems, Inc.

Les utilisateurs d'interfaces graphiques OPEN LOOK® et Sun™ ont été développés 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, cette licence couvrant aussi les licences de Sun qui mettent en place OPEN LOOK GUIs 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 REPENDRE A UNE UTILISATION PARTICULIERE OU LE FAIT QU'ILS NE SOIENT PAS CONTREFAISANTS DE PRODUITS DE TIERS.

CETTE PUBLICATION PEUT CONTENIR DES MENTIONS TECHNIQUES ERRONEES OU DES ERREURS TYPOGRAPHIQUES. DES CHANGEMENTS SONT PERIODIQUEMENT APPORTES AUX INFORMATIONS CONTENUES AUX PRESENTES, CES CHANGEMENTS SERONT INCORPORES AUX NOUVELLES EDITIONS DE LA PUBLICATION. SUN MICROSYSTEMS INC. PEUT REALISER DES AMELIORATIONS ET/OU DES CHANGEMENTS DANS LE(S) PRODUIT(S) ET/OU LE(S) PROGRAMME(S) DECRITS DANS CETTE PUBLICATION A TOUS MOMENTS.

