Provision Administrator's Guide

Document Number 007-3273-001

CONTRIBUTORS

Written by Jeffrey B. Zurschmeide
Production by Ruth Christian
Engineering contributions by Vic Mitnick, Ed Mascarenhas, Anurag Narula
St Peter's Basilica image courtesy of ENEL SpA and InfoByte SpA. Disk Thrower image courtesy of Xavier Berenguer, Animatica.

Provision Administrator's Guide
Document Number 007-3273-001

Contents

List of Figures

List of Tables

About This Guide

"About This Guide" includes brief descriptions of the contents of this guide and an explanation of typographical conventions used, and refers you to additional sources of information you might find helpful.

This guide explains how to perform general operation tasks of the *provision* distributed system monitoring application, used with Silicon Graphics® workstations and servers.

What This Guide Contains

This guide contains the following chapters:

- Chapter 1, "Distributed System Monitoring With provision"
  Provides an overview of *provision*.
- Appendix A, "The MIB Browser"
  Provides documentation for the MIB browser software tool.

How to Use This Guide

This guide is written for administrators who are responsible for configuring and maintaining the *provision* implementation at their site.

Target Audience of This Guide

This guide is intended for administrators who are responsible for one or more systems running the *provision* distributed system monitor.

Additional Resources

For easy reference, here is a list of the guides and resources provided with your system and the specific focus and scope of each:

IRIX Admin Manual Set

This guide is an additional resource to the *IRIX Admin* manual set.

The *IRIX Admin* suite is intended for administrators: those who are responsible for servers, multiple systems, and file structures outside the user's home directory and immediate working directories. If you find yourself in the position of maintaining systems for others or if you require more information about IRIX than is in the end-user manuals, these guides are for you. The *IRIX Admin* guides are available through the IRIS InSight™ online viewing system. The set comprises these volumes:

- *IRIX Admin: Software Installation and Licensing*—Explains how to install and license software that runs under IRIX™, the Silicon Graphics implementation of the UNIX® operating system. Contains instructions for performing miniroot and live installations using Inst, the command line interface to the IRIX installation utility. Identifies the licensing products that control access to restricted applications running under IRIX and refers readers to licensing product documentation.
- *IRIX Admin: System Configuration and Operation*—Lists good general system administration practices and describes system administration tasks, including configuring the operating system; managing user accounts, user processes, and disk resources; interacting with the system while in the PROM monitor; and tuning system performance.
- *IRIX Admin: Disks and Filesystems*—Describes how to add, maintain, and use disks and filesystems. Discusses how they work, their organization, and how to optimize their performance.
- *IRIX Admin: Networking and Mail*—Describes how to plan, set up, use, and maintain the networking and mail systems, including discussions of sendmail, UUCP, SLIP, and PPP.
- *IRIX Admin: Backup, Security, and Accounting*—Describes how to back up and restore files, how to protect your system's and network's security, and how to track system usage on a per-user basis.

- *IRIX Admin: Peripheral Devices*—Describes how to set up and maintain the software for peripheral devices such as terminals, modems, printers, and CD-ROM and tape drives. Also includes specifications for the associated cables for these devices.
- *IRIX Admin: Selected Reference Pages* (not available in InSight)—Provides concise reference page (manual page) information on the use of commands that may be needed while the system is down. Generally, each reference page covers one command, although some reference pages cover several closely related commands. Reference pages are available online through the man(1) command.

Reference Pages

The IRIX reference pages (often called “man” or “manual” pages) provide concise reference information on the use of IRIX commands, subroutines, and other elements that make up the IRIX operating system. This collection of entries is one of the most important references for an administrator. Generally, each reference page covers one command, although some reference pages cover several closely related commands.

The IRIX reference pages are available online through the *man* command. To view a reference page, use the *man* command at the shell prompt. For example, to see the reference page for *diff*, enter:

```
man diff
```

It is a good practice to print those reference pages you consistently use for reference and those you are likely to need before major administrative operations and keep them in a notebook of some kind.

Each command, system file, or other system object is described on a separate page. The reference pages are divided into seven sections, as shown in Table i. When referring to reference pages, this document follows a standard UNIX convention: the name of the command is followed by its section number in parentheses. For example, *cc*(1) refers to the *cc* reference page in Section 1.

Table i shows the reference page sections and the types of reference pages that they contain.

Table i Outline of Reference Page Organization

Type of Reference Page	Section Number
General Commands	(1)
System Calls and Error Numbers	(2)
Library Subroutines	(3)
File Formats	(4)
Miscellaneous	(5)
Demos and Games	(6)
Special Files	(7)

Release Notes

Provide specific information about the current release. Exceptions to the administration guides are found in this document. Release Notes are available online through the *relnotes* command. Each optional product or application has its own set of release notes. The *grelnotes* command provides a graphical interface to the release notes of all products installed on your system.

The IRIX Help System

Your IRIX system comes with a help system. This system provides help cards for commonly-asked questions about basic system setup and usage. The command to initiate a help session is *desktophelp*.

The Silicon Graphics World Wide Web Site

The Silicon Graphics World Wide Web (WWW) presence has been established to provide current information of interest to Silicon Graphics customers. The following URLs are accessible to most commercially available Web browsers on the Internet:

http://www.sgi.com
: The Silicon Graphics general Web site, Silicon Surf

http://www.mips.com
: The Silicon Graphics mips division site

http://www.studio.sgi.com
: The Silicon Studio site

http://www.alias.com
: The Alias site

http://www.sgi.com/Technology/TechPubs
: The Silicon Graphics Technical Publications Library

From these sites you can find all the Silicon Graphics Web-published information, including the Technical Publications Library.

Conventions Used in This Guide

These type conventions and symbols are used in this guide:

Bold
: User account names.

Italics
: Filenames, glossary entries (online, these show up as underlined), IRIX commands, manual/book titles, new terms, onscreen button names, program variables, variable command-line arguments, and variables to be supplied by the user in examples, code, and syntax statements

`Fixed-width type`
: Onscreen text and MIB specifications

`Bold fixed-width type`
: User input, including keyboard keys (printing and nonprinting); literals supplied by the user in examples, code, and syntax statements (*see also* <>)

“”	(Double quotation marks) Onscreen menu items and references in text to document section titles
()	(Parentheses) Following IRIX commands—surround reference page (man page) section number
<>	(Angle brackets) Surrounding nonprinting keyboard keys, for example, <**Esc**>, <**Ctrl-D**>

This guide uses the standard UNIX convention for citing reference pages in the IRIX documentation. The page name is followed by the section number in parentheses. For example, *rcp*(1C) refers to the *rcp* online reference page.

User Interface Terminology

Figure i and Figure ii show examples of windows, with the window terms used in this guide noted.

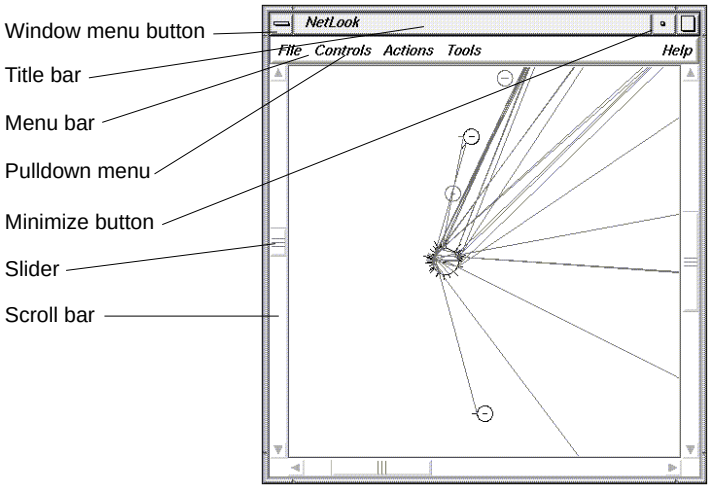

Figure i Window Terms

The mouse buttons have these functions:

left	Perform most basic tasks: click buttons, select an entry field to type into, select menu choices, select items in a display, select text to modify, and so on.
middle	Reposition windows and icons.
right	Access popup menus. Popup menus appear when you press the right mouse button in certain locations on the screen.

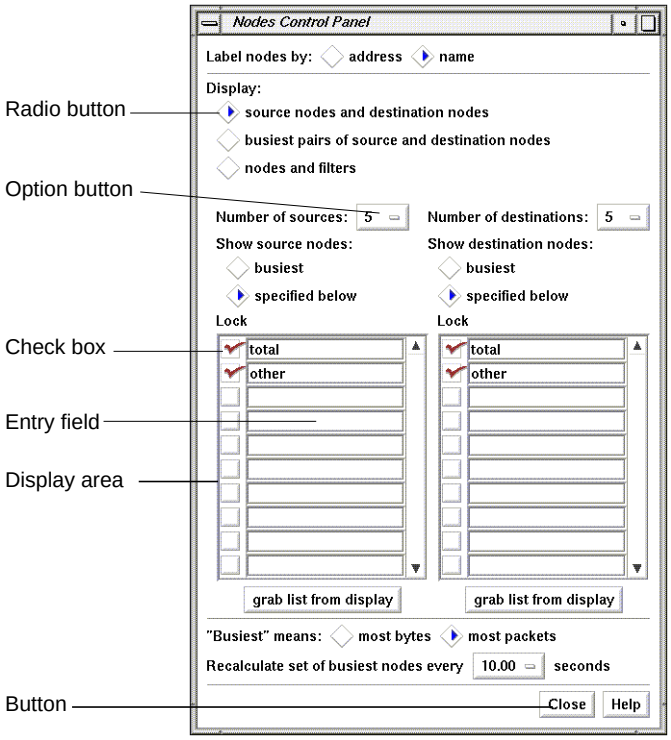

Figure ii More Window Terms

This guide uses the following terms to describe the use of the mouse:

press	Hold down a mouse button.
drag	Move the mouse while a mouse button is pressed.

click	Press a mouse button and immediately release it without moving the mouse.
double-click	Press and release a button twice in quick succession without moving the mouse.
select	The term “select” is used in the following ways: • Click the left mouse button on an item to highlight it. • Press the left mouse button in an entry field, drag the cursor across some or all of the text, and release the mouse button. The text becomes highlighted. • Press the left mouse button on a menu title in a menu bar, move the cursor to a menu choice, and release the mouse button while a menu choice is highlighted.
deselect	Click a highlighted item to turn off the highlighting.

Common User Interface Operations

This guide assumes that you are familiar with using the mouse, working with windows, and using pulldown and rollover menus. These operations are described in the *IRIS Essentials* guide.

The sections below explain how to use additional components of the user interface that are common to several of the tools.

Using Scroll Bars

You can use scroll bars (see Figure iii) to change the area and scale of a viewing area and to display different lines or portions of lines in a display area. The size of the slider is proportional to the amount of the total that you are viewing. You operate scroll bars by pressing the left or middle mouse button when the cursor is in the scroll bar. There are several ways to operate the scroll bar:

- Press the left mouse button on the slider, drag the cursor to a new slider position, and release the button.
- Move the slider incrementally by clicking the triangles at each end of the scroll bar.

- Move the slider up or down by positioning the cursor in the trough above or below the slider and clicking the left mouse button.
- Move the slider to a specific position by positioning the cursor at that position and clicking the middle mouse button.

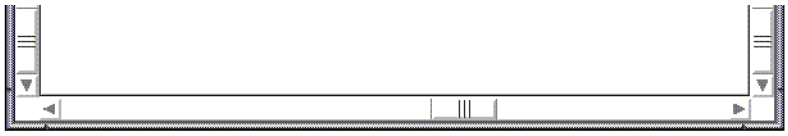

Figure iii A Horizontal Scroll Bar

Entering and Removing Text in a Field

Editing text in the entry fields (see Figure iv) is the same as editing text in the entry fields of other applications:

- Position the text insertion point by moving the mouse to the entry field and clicking the left mouse button.
- Select (highlight) text by pressing the left mouse button at one end of the text that you want to select and dragging to the other end.
- Select a word including a space or punctuation-delimited characters by moving the cursor to the word and double-clicking the left mouse button.
- Select the entire contents of an entry field by moving the cursor over the entry field and triple-clicking the left mouse button.
- Delete selected (highlighted) text by pressing the **<Backspace>** key.
- Delete the character to the left of the insertion point by pressing the **<Backspace>** key.

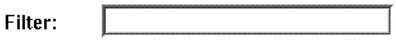

Figure iv An Entry Field

Using Option Buttons

Option buttons (on the left in Figure v) let you select a numeric value from among a predefined set of choices. To use an option button, first press the option button with the left mouse button. A menu pops up (on the right in Figure v). Move the cursor to your selection and release the mouse button.

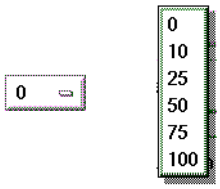

Figure v An Option Button and an Option Button Menu

Using a File Prompter

File prompter windows (like the one in Figure vi) are used to specify filenames. You can choose a filename by double-clicking a name in the display area. You can also type the name into the filename entry field and press **<Enter>** or click the *Accept* button to complete your filename selection. You can change directories to the parent of the current directory by clicking the *Up* button, or return to the directory where you started the tool by clicking the *Original* button.

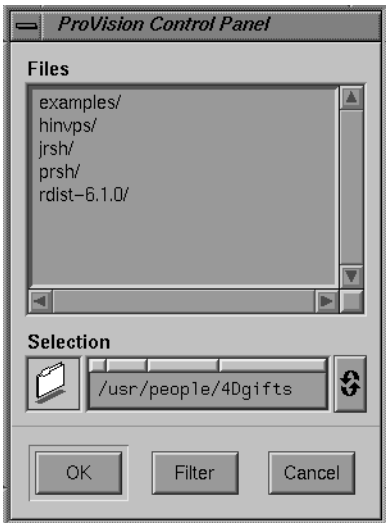

Figure vi A File Prompter Window

Using Online Help

The *provision* tool provides many online help files to help you as you learn to use the tools. You access these files from the Help menu in the menu bar of many *provision* windows (shown in Figure vii on top) and from the *Help* button that appears in some *provision* windows (on the bottom in Figure vii).

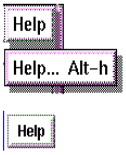

Figure vii A Help Menu and a Help Button

When you choose "Help..." from a menu or click a *Help* button, a Showcase window appears and displays the first help card.

Some help files contain several cards. Page through these cards using the **<Page Up>** and **<Page Down>** keys in the cluster of six keys just to the right of the **<Backspace>** key or

click the left mouse button on the arrows at the bottom of the pages. Make sure the cursor is in the Help window when you press these keys.

When you're finished reading a help file, you can close the Help window just as you close any other window, for instance, by double-clicking the Window menu button in the upper left corner of the window or by selecting Quit from the Window menu.

Chapter 1

Distributed System Monitoring With provision

This chapter describes the *provision* application and its use in monitoring the status of the host systems on your network. The following sections are provided:

- "The provision Monitoring System" provides an overview of the nature and purposes of the *provision* application.
- "Installing and Configuring provision" provides detailed instructions on installing *provision* and configuring it to your specific needs.
- "Using provision" provides general information on the interfaces and different programs that make up *provision*.
- "Using pvcontrolpanel" provides detailed information on the *pvcontrolpanel* logging and notification tool.
- "Using pvcontrol" provides detailed information on the *pvcontrol* text-based notification and logging tool.
- "Using pvgraph" provides detailed information on the dynamic *pvgraph* graphing tool.

The provision Monitoring System

The *provision* application allows you (the administrator) to keep track of the running statistics of each system in your heterogeneous network from a single location. This location may be a host workstation or server console, or even a text-based terminal. The data provided about each system on the network can be displayed graphically, if the administrator's host system allows it, or in text, or data can be stored for later analysis. Error messages from each system can be displayed immediately on the administrator's console.

The *provision* application provides three basic utilities:

- *pvcontrolpanel* is a graphical notification and logging utility for use on systems with graphics capabilities, such as graphical workstations and X-terminals.
- *pvcontrol* is a text-based utility for use on non-graphics servers and ASCII terminals.
- *pvgraph* is a graphical tool to dynamically graph system performance and view logs of statistics made with *pvcontrol* and *pvcontrolpanel*.

The graphical user interface provides the full power of *provision* to the administrator. Information is updated in real time, and you can add or delete variables as you wish.

The text-based user interface is a subset of the graphical interface, and is provided for those administrators without access to graphics capability. The text-based interface does not provide the real-time updating of information that is featured in the graphical interface, but an interactive mode is available to change the collection instructions.

You may also want to coordinate the use of the standard IRIX features *sysmon*(1M), and *syserrpanel*(1M) with *provision*. These standard IRIX utilities use the system log daemon (*syslogd*) to monitor the system status. Complete information on *sysmon* and *syserrpanel* is available through the IRIX reference pages.

The *provision* application collects its information according to programs provided as part of the standard distribution, but you can write your own instruction sets in the programming language of your choice to customize *provision*. The *provision* application uses SNMP to collect information over the network.

SNMP stands for Simple Network Management Protocol. SNMP is used to communicate with other systems that also run SNMP. The other system can be a workstation, a router, a bridge, a hub, or a gateway—any host that has an IP address and implements the SNMP protocol and agent. SNMP implements an "agent." An agent is an SNMP program that exchanges information with a remote host. *snmpd*(1M) is the Silicon Graphics SNMP agent. Agents for other types of nodes may be implemented in software or firmware and are vendor-specific. There is a reference book for SNMP called *The Simple Book, An Introduction to Management of TCP/IP-Based Internets*, by Marshall T. Rose. The book was published in 1991 by Prentice-Hall, of Englewood Cliffs, New Jersey, USA 07632. The ISBN number of this book is 0-13-812611-9.

SNMP relays basic system information about each host to the other hosts, on request. The information relayed comes from the Management Information Bases (MIBs) for that host. An MIB is the specification for the virtual store of the information supported by an agent.

The standard provision MIBs are the *hp-ux_sgi* MIB and the *mib2* MIB, both distributed with provision in the */usr/lib/netvis/mibs* directory. Further information about MIBs and the MIB browser is available in Appendix A, "The MIB Browser." The browser is designed to be used by network managers experienced in managing various devices on the network.

You can create and add your own MIBs to your systems, or you can use MIBs obtained from other vendors with provision. MIB textual descriptions should be placed in the directory */usr/lib/netvis/mibs* to be accessed through provision.

The reason for creating and developing this software is to allow the system administrator of a large site with many different brands of hardware an extensible system to monitor many heterogeneous hosts from a single station.

Installing and Configuring provision

To use *provision* on your network, you must first propagate the *snmp* daemon to all systems that may be monitored.

On the monitoring system, the following requirements must be met before *provision* can run successfully. These requirements assume that you also wish to monitor the monitoring system itself:

- provision must be correctly installed.
- The *provisiond* daemon must be running. Place the following lines in the following files to cause *provisiond* to run automatically on the monitoring system:

  To */etc/services*, add this line:

  ```
  provisiond    5299/udp   # provision daemon
  ```

  To */etc/inetd.conf*, add this entry:

```
provisiond dgram udp wait root /usr/provision/bin/provisiond provisiond
```

  Then enter the command

  ```
  killall -HUP inetd
  ```

  to cause *inetd* to restart and run the *provision* daemon.

- The SNMP daemon (*snmpd*) must be running. Enter the following commands as **root** to cause *snmpd* to run automatically on the monitoring system:

  ```
  chkconfig network on
  /etc/init.d/network start
  chkconfig snmpd on
  /etc/init.d/snmp start
  ```

- The hp-ux_sgi MIB must be installed. This HP-UX support file is installed by default with the *snmpd* package of provision in */usr/lib/netvis/mibs/hp-ux_sgi.mib*.

On all Silicon Graphics systems to be monitored, the following requirements must be met before *provision* will successfully monitor their status:

- The SNMP daemon (*snmpd*) must be running. First, install the following package from your provision distribution on each Silicon Graphics system to be monitored:

  ```
  snmpd          01/04/95 SNMP Daemon with HP MIB Support
  ```

  Next, enter the following commands as **root** on the monitored system to cause *snmpd* to run automatically:

  ```
  chkconfig snmpd on
  /etc/init.d/snmp start
  chkconfig network on
  /etc/init.d/network start
  ```

- The hp-ux_sgi MIB is installed by default with the *snmpd* package of provision. This MIB must be installed in order for the system to be monitored.

On all systems not manufactured by Silicon Graphics, the following requirements must be met before *provision* will operate correctly. Note that if another manufacturer's system MIB and SNMP daemon are not fully compatible with the distributed MIB and SNMP daemon, some scripts and MIB variables distributed with *provision* may not function for those systems. However, new MIBs and variables may be created for any or all systems:

- An SNMP agent (daemon) must be running on the system.
- An MIB must be installed on the system.

Consult your system manufacturer's documentation for information on fulfilling these requirements.

Running snmpd on Your Network

In order to use many of the utilities and features of provision, each system on your network should be running the *snmpd* daemon. This daemon provides support for SNMP (Simple Network Management Protocol) and allows other systems to query the host for information about its configuration. This daemon is described in Appendix A, "The MIB Browser."

To obtain SNMP support, the provision distribution packages to install on each station are:

```
snmpd          01/07/97 SNMP Daemon with HP MIB Support
snmpd.sw       01/07/97 SNMP Daemon with HP MIB Support
snmpd.sw.hp    01/07/97 SNMP Daemon with HP MIB Support
```

To configure a workstation so that *snmpd* is started automatically when the system is rebooted, install the packages listed above, and enter this command on the system while logged in as **root**:

```
chkconfig -f snmpd on
```

To check to see if the daemon is already running, enter this command:

```
ps -e | grep snmpd
```

If there is no output from this command, *snmpd* is not running. Become **root** and enter this command to start *snmpd*:

```
snmpd &
```

Using provision

There are two interfaces provided for *provision*, the graphical and the text interface. The graphical interface is the primary interface, since *provision* is designed to provide graphical information about your systems.

The provision Graphical Interface

When you first invoke *provision*, you see the window shown in Figure 1-1, in the standard Silicon Graphics desktop format. (See the section titled "Managing Windows" in the *IRIS Essentials* guide for complete information on the facilities of desktop windows.)

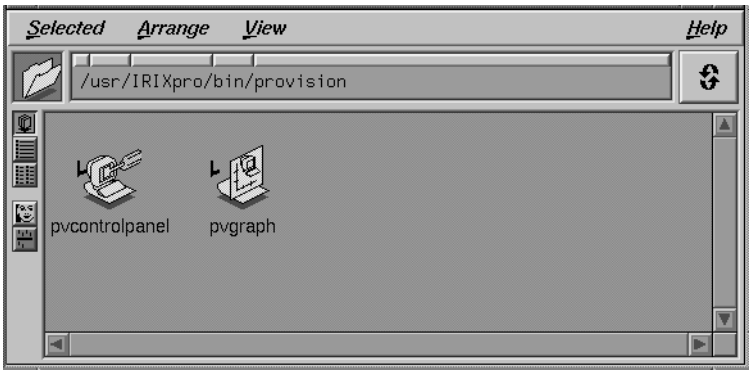

Figure 1-1 The provision Window

There are two main tools you can select from this window, *pvcontrolpanel* and *pvgraph*. These tools and their subordinate tools are discussed in the sections titled "Using pvcontrolpanel," "Using pvcontrol," and Appendix A, "The MIB Browser." To invoke a tool, place the cursor over the desired icon and double-click the left mouse button. The icon changes color when you select it, and the "carpet" underneath the icon moves up to show that the invocation was successful and a new tool window appears on your screen. Each of these tools is detailed in its own section below.

Using pvcontrolpanel

This tool is the main controlling panel for *provision*. From this panel, you can set up monitors on all systems on your network, and receive error messages and notifications. When you invoke *pvcontrolpanel*, the window shown in Figure 1-2 appears on your screen.

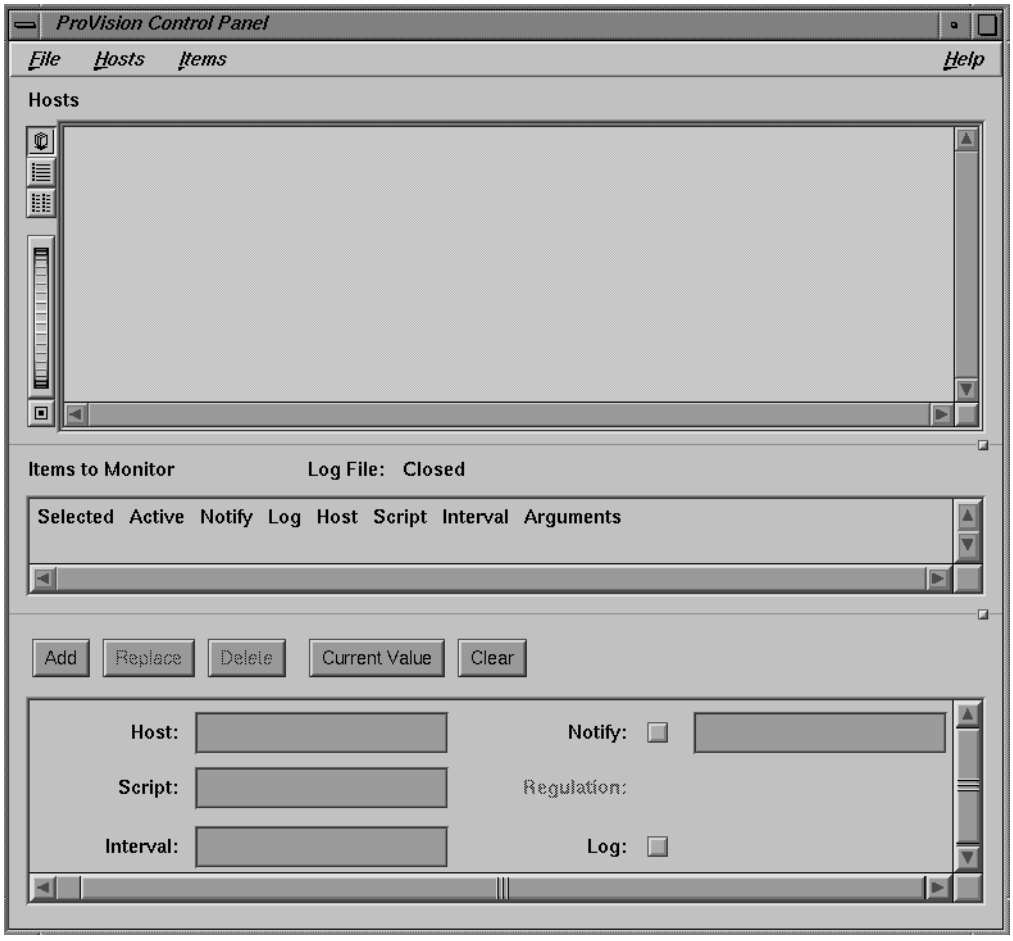

Figure 1-2 The pvcontrolpanel Window

There are four main sections of the *pvcontrolpanel* window. From the top of the window to the bottom, these sections are:

Menu Bar — The top bar, with the File, Hosts, Items, and Help menus. This is discussed in the section titled "Using the pvcontrolpanel Menu Bar."

Hosts — All the hosts and collections currently monitored by this instance of *provision* and any other icons you may have added are shown in this area. This is discussed in the section titled "The pvcontrolpanel Hosts Area."

Items to Monitor — All the items currently being monitored on any host are listed in this area. This is discussed in the section titled "The pvcontrolpanel Items to Monitor Area."

Script Configuration — This area is where you enter information about the scripts and hosts to be monitored. This is discussed in the section titled "The pvcontrolpanel Script Configuration Area."

Using the pvcontrolpanel Menu Bar

There are four menus available on the *pvcontrolpanel* menu bar. The menus and their choices are listed below.

The File menu contains options dealing with the *pvcontrolpanel* configuration files and contains the options to restart and quit the session. The following choices are provided:

- Read Config

  This option reads a previously stored monitoring configuration from a file. You can also drag an icon representing a previously stored config file from the directory view onto the *pvcontrolpanel* icon to start *pvcontrolpanel* with that configuration. For more information on config files, see "The provision Configuration File."

- Save Config

  This option saves the current monitoring configuration in a file.

- Save Config as...

  This option saves the current configuration to a different filename.

- Close Log File...

  This option closes the current log file and opens a new one.

- Show Log Files...

  This option brings up the pvgraph window (see "Using pvgraph to View a Log") to display the contents of a log file recording of your *pvcontrolpanel* activity. A log file simply contains a series of values for a script or variable accumulated over a period of time.

- Quit

  This option ends the *pvcontrolpanel* session.

The Hosts menu allows you to control the arrangement of the hosts in your *pvcontrolpanel* window. The following choices are available:

- View as Icons

  This option tells *pvcontrolpanel* to represent the hosts in your window with large icons, arranged alphabetically left to right.

- View as List

  This option tells *pvcontrolpanel* to represent the hosts in your window with smaller icons in a single column alphabetized list.

- View in Columns

  This option tells *pvcontrolpanel* to represent the hosts in your window with smaller icons, in an evenly columnized, vertical, alphabetized list.

- Add Icon

  This option adds an icon for a named host to your hosts area. You must first enter the hostname in the script configuration area.

- Remove Icon

  This option removes the selected icon from your hosts area.

Icons in the Hosts section represent each host that is currently communicating in some way with provision. A new icon is not added for each addition item monitored on a listed host unless it is specifically requested with the "Add Icon" menu choice.

The Items menu contains options dealing with the operation of the *pvcontrolpanel* activity. An *Item* is any configured monitoring unit, for example, monitoring a script on a particular host at a particular interval. The following choices are provided:

- Start All Items

  This option starts all currently configured monitoring.

- Stop All Items

  This option stops all monitoring activity.

- Show Available Variables and Scripts

  This option brings up a window with a list of all available variables for monitoring, and all available scripts. To select a variable or script, place the mouse cursor over the desired list item and double-click with the left mouse button. This window is discussed further in "The Available Variable and Script Window."

- MIB Browser

  This option invokes the MIB browser. For more information, see Appendix A, "The MIB Browser."

- Add

  This option takes the information entered in the script configuration area and adds the entry to the Items area, and the specified host to the Hosts area.

- Delete

  This option deletes the selected item from the Items area.

- Delete All

  This option deletes all items and monitoring instructions.

- Replace

  This option changes the selected item by replacing it with a new item according to the current entries in the script configuration area.

- Current Value

  This option runs the selected script once and returns the current value. The script will be run locally, although scripts can be written that execute other scripts on remote systems.

The Help menu invokes the online help utility to provide help on all aspects of using *provision*.

The Available Variable and Script Window

When you select Show Available Variables and Scripts from the Variables menu (or from the *Configure One Graph* window in *pvgraph*), you see the window shown in Figure 1-3.

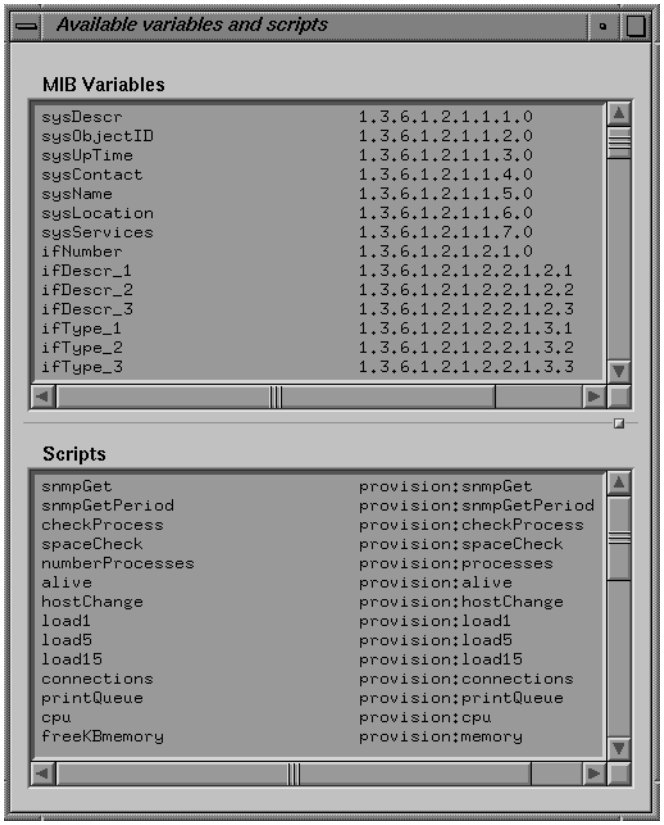

Figure 1-3 The Available Variables and Scripts Window

This window lists available MIB variables that can be monitored in the upper half, and all available monitoring and notification scripts in the lower half. You can monitor MIB variables not listed in this window, but they must be specified by their full numeric Object ID (for example, the *sysServices* variable has an Object ID of 1.3.6.1.2.1.1.7.0). You can also monitor any script you have created that is not represented in this window, but it must be specified with its full name. For example, the *snmpGet* script's full name is *provision:snmpGet*.

The MIB variables are described in the MIB file. To see a variable's description, select the MIB browser from the *Add One Graph* window (in *pvgraph*) or select "MIB Browser" from the Variables menu in *pvcontrolpanel*.

Once the browser is up, press the *Variable...* button and enter the name of the variable you wish to have described in the Name field. Then, select Description from the Help menu of the Variable window. A new window appears, showing the description text. For example, Figure 1-4 shows the description window of the *sysDescr* variable.

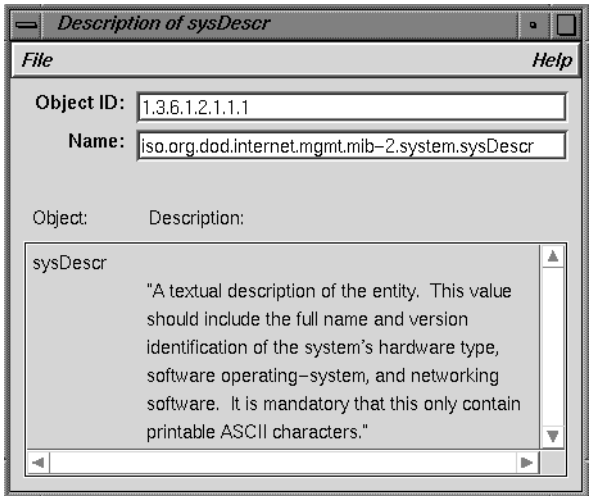

Figure 1-4 Description Window for the sysDescr Variable

This procedure is also described in the section titled "Obtaining Descriptions of Variables" in Appendix A.

The Default provision Scripts

You can obtain a list of currently available scripts at any time by selecting the Show Available Variables and Scripts option from the from the Variables menu (or from the *Configure One Graph* window in *pvgraph*). The scripts shipped with *provision* are as follows:

alive This script simply sends an ICMP ECHO (*ping*(1M)) request to the named remote system, and returns an error if it fails to get a response within a reasonable time. The arguments to this script are a test interval (in seconds) and a list of hosts to check. The script returns *true* or *false* for each system, and a status message if the script fails to fetch the data.

checkProcess This script reads the process table from a remote system and checks for the existence of a particular process name. The arguments for this scripts are a test interval (in seconds), a list of hosts, and a process name. The script returns *true* or *false*, and a status message if the process does not exist, or if the script fails to get a response.

connections This script returns the number of open network connections to a system, and an error if it is above a limit. The arguments are a test interval (in seconds), a list of hosts to check, and an upper bounds.

contextSwitchRstat
This procedure returns the raw number of process switches that have occurred on a remote system since the last boot, or an error if the script does not receive the information. The argument is a list of hosts.

contextSwitchRstatPeriod
This procedure returns the number of context switches that have occurred on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit, and a lower limit.

cpu This script returns the average percentage of CPU utilization on a system, or a status message if the number is out of bounds or if the script fails to retrieve the data. The arguments to this script are a test interval (in seconds), a list of hosts, a lower bound, and an upper bound.

fileSystemBavail
This script returns the number of free blocks in the specified file system available to non-superuser.

fileSystemBfree This script returns the number of free blocks in the specified file system.

fileSystemBlock	This script returns the total number of blocks in the specified file system.
fileSystemBsize	This script returns the fundamental block size for the specified file system.
fileSystemDir	This script returns path prefix for the specified file system. This script is useful with "get current value" to check file system identity, but is not useful for monitoring.
fileSystemFfree	This script returns the number of free file nodes in the specified file system.
fileSystemFiles	This script returns the total number of file nodes in the specified file system.
fileSystemName	This script returns the name of the specified file system. This script is useful with "get current value" to check file system identity, but is not useful for monitoring.
freeKBmemory	This script returns the amount of free memory in KB.
freeKBswap	This script returns the amount of free swap space in KB.
hostChanged	This script watches for a change in the availability of a host. The argument to this script is a test interval (in seconds) and a list of hosts to watch. The script returns *true* or *false* for each host, and a status message if the status of a host changes.
ifAdminStatus	This script gets the administrative status of the specified network interface.
ifCollisionsRstat	This procedure returns the raw network collisions that have occurred on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.
ifCollisionsRstatPeriod	This procedure returns the raw number of network collisions that have occurred on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit and a lower limit.
ifDescr	This script gets the description of the specified network interface.

ifInDiscards This script gets the number of inbound packets on the specified network interface, which were chosen to be discarded even though no errors had been detected, to prevent their being deliverable to a higher-layer protocol since the last sample.

ifinErrors This script gets the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol on the specified interface since the last sample.

ifInErrorsRstat This procedure returns the raw number of input errors that have occurred on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.

ifInErrorsRstatPeriod
This procedure returns the raw number of network read errors that have occurred on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts and an upper limit.

ifInNUcastPkts This script gets the number of non-unicast (for example, subnetwork-broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol on the specified network interface since the last sample.

ifInOctets This script gets the total number of octets received on the specified network interface since the last sample.

ifInPacketsRstat
This procedure returns the raw number of packets that have been received on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.

ifInPacketsRstatPeriod
This procedure returns the raw number of network packets that have been read in on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit, and a lower limit.

ifInUcastPkts This script gets the of subnetwork-unicast packets delivered to a higher-layer protocol on the specified interface since the last sample.

ifInUnknownProtos
This script gets the number of packets received through the specified network interface which were discarded because of an unknown or unsupported protocol since the last sample.

ifOperStatus This script gets the administrative status of the specified network interface.

ifOutDiscards This script gets the number of outbound packets on the specified network interface which were chosen to be discarded even though no errors had been detected to prevent their being transmitted since the last sample.

ifOutErrors This script gets the number of outbound packets on the specified network interface that could not be transmitted because of errors since the last sample.

ifOutErrorsRstat
This procedure returns the raw number of output errors that have occurred on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.

ifOutErrorsRstatPeriod
This procedure returns the raw number of network write errors that have occurred on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts and an upper limit.

ifOutNUcastPkts
This script gets the total number of packets that higher-level protocols requested be transmitted on the specified network interface to a non-unicast (for example, a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent since the last sample.

ifOutOctets This script gets the total number of octets transmitted out of the specified network interface since the last sample.

ifOutPacketsRstat
This procedure returns the raw number of packets that have been sent from a remote system, or an error if the script does not receive the information. The argument is a list of hosts.

ifOutPacketsRstatPeriod
This procedure returns the raw number of network packets that have been written out on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit, and a lower limit.

ifOutQLen	This script gets the length of the output packet queue (in packets) of the specified network interface.
ifOutUcastPkts	This script gets the total number of packets that higher-level protocols requested be transmitted on the specified network interface to a subnetwork-unicast address, including those that were discarded or not sent since the last sample.
ifPhysAddress	This script gets the hardware address of the specified network interface.
ifSpecific	This script gets a reference to MIB definitions specific to the particular media being used to realize the network interface. For example, if the interface is realized by an ethernet, then the value of this object refers to a document defining objects specific to ethernet.
ifSpeed	This script gets the speed (estimated bandwidth in bits per second) of the specified network interface.
ifType	This script gets the type of the specified network interface. For example, ethernet, FDDI, or HIPPI.
interruptsRstat	This procedure returns the raw number of interrupts that have been received on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.
interruptsRstatPeriod	This procedure returns the raw number of interrupts that have occurred on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit, and a lower limit.
load1	This procedure returns the current load average of a system over the previous second, and a status if the load is out of bounds or the script does not receive the data. The arguments are a list of hosts to check, a low boundary, and a high boundary.
load5	This procedure returns the current load average of a system over the previous 5 seconds, and a status if the load is out of bounds or the script does not receive the data. The arguments are a list of hosts to check, a low boundary, and a high boundary.
load15	This procedure returns the current load average of a system over the previous 15 seconds, and a status if the load is out of bounds or the script does not receive the data. The arguments are a list of hosts to check, a low boundary, and a high boundary.

memory This script returns the amount of free memory in kilobytes, and a status message if the number is out of bounds or the script fails to get the data. The arguments are a test interval (in seconds), a list of hosts, a lower bound, and an upper bound.

nfsChanged This script performs essentially the same function as *nfsCheck*, but returns a status message only when the state of a remote server changes, that is, if a host that was formerly responding correctly ceases, or a host that was not responding begins to respond. The argument is a list of hosts.

nfsCheck This script checks NFS server remote systems for correct response. The script returns a true or false value for each host. A true value indicates a correct response, and a false value indicates that the NFS server is not functioning correctly. A status message is also displayed if the server is not responding correctly or if the script fails to get the information. The argument is a list of hosts.

pageInRstat This procedure returns the raw number of pages that have been paged in on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.

pageInRstatPeriod
This procedure returns the raw number of pages that have been paged in on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit and a lower limit.

pageOutRstat This procedure returns the raw number of pages that have been paged out on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.

pageOutRstatPeriod
This procedure returns the raw number of pages that have been paged out on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit, and a lower limit.

printQueue This script checks the status of a remote printer queue. The arguments to this script are a test interval (in seconds), a list of hosts, and the remote printer name. The script returns *true* or *false* for the named printer, and a status message if the printer is down, or if the script fails to get the information.

processChanged
: This script reads the process table from a system and checks for the existence of the specified name. The script returns a data field of true or false, and a status message if the process used to exist but has exited, or if the process did not exist before but has now started, or if the script fails to retrieve the information. The arguments for this script are a list of hosts, and a process name.

processCPU
: This script returns the processor utilization for scheduling of the specified process.

processCPUticks
: This script returns the ticks of cpu time for the specified process.

processCPUticksTotal
: This script returns the total ticks of cpu time for the specified process, for the life of the process.

processCmd
: This script returns the name of the command the specified process is running This script is useful with “get current value” to check process identity, but is not useful for monitoring.

processes
: This script checks the number of processes on a remote system and notifies you if the number is not in the specified bounds, and a status message if the number is out of script bounds, or if the script fails to get the data. The arguments for this script are a test interval (in seconds), a list of hosts, a low bound, and a high bound.

processPctCPU
: This script returns the percentage of CPU time used by the specified process.

processPrio
: This script returns the *nice*(1) priority of the specified process.

processRssize
: This script returns the resident set size of the specified process.

processStatusInt
: This script returns the status of the specified process as an integer. Values are: sleep(1), wait(2), run(3), idle (4), zombie(5), and stop(6).

processStatusString
: This script returns the status of the specified process as a text string. Values are: sleep, wait, run, idle, zombie, and stop.

processStime
: This script returns the system time spent executing the specified process.

processUtime
: This script returns the user time spent executing the specified process.

processWchan	This script returns, for the specified process, the value it is sleeping on if its processStatus script value is sleep(1).
random	This script invokes a random number generator. The arguments are a test interval (in seconds), a list of hosts, a lower bound, and an upper bound. This script is used for testing purposes or demonstration.
snmpGet	This is a very simple script to query a system (or a collection of systems) for an *snmp* variable and return the value of the variable. The arguments for this script are a test interval (in seconds), a list of hosts, and a list of variables to be queried.
snmpGetPeriod	This is a very simple routine to query a system (or a collection of systems) for an *snmp* variable and return the change in it since the last query. The arguments for this script are a list of hosts and a list of variables.
spaceCheck	This script checks the available space on a given file system, and verifies that it is between the specified bounds. The arguments for this script are a test interval (in seconds), a list of hosts, a file system name, a low bound, and a high bound. The script returns a data field of the available space, and a status message if the check fails the bounds check or the script cannot get the data.
swap	This script returns the amount of free swap space on a host. The arguments are a test interval (in seconds), a list of hosts, a lower bound, and an upper bound. The script returns the amount of free space in kilobytes, and a status message if the number is out of bounds or the script fails to get the data.
swapInRstat	This procedure returns the raw number of processes that have been swapped in on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.
swapInRstatPeriod	This procedure returns the raw number of pages that have been swapped in on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit, and a lower limit.
swapOutRstat	This procedure returns the raw number of processes that have been swapped out on a remote system, or an error if the script does not receive the information. The argument is a list of hosts.

swapOutRstat Period

This procedure returns the raw number of pages that have been swapped out on a remote system since the last check, or an error if the script does not receive the information. The arguments are a list of hosts, an upper limit, and a lower limit.

Adding Custom Scripts to provision

The *provision* application retrieves data from remote systems through commonly used protocols including *rstat* and SNMP. The *provisiond* daemon has an embedded SGITCL interpreter and uses SGITCL scripts to retrieve remote information.

When a request is made for a new SGITCL script from *pvgraph* or *pvcontrolpanel*, the *provisiond* daemon creates a new, private copy of the SGITCL interpreter. The daemon then calls a predefined script called *provision:wrapper* to gather the script information and return it. If the requested script is a valid SNMP Object ID, the routine *provision:snmpGet* is called to do the retrieval. If the script is a custom file you have created, *provision:wrapper* executes the file to retrieve the script data. Finally, the wrapper calls the script as an internal SGITCL routine which can be in any SGITCL tlib library.

You must create a file with a name ending in *.tlib* in the */usr/provision/lib* directory to hold your SGITCL script in order for *provision:wrapper* to locate the new script and call it as an sgitcl routine.

Your scripts are not required to be written in SGITCL. If your new script is not an SGITCL script, simply place the full pathname of the executable program or script in the */usr/provision/scriptDefs* file.

A description of each script must be placed in the file */usr/provision/scriptDefs*. This description is used to determine the type of any arguments, and the type of the return data from the script. If a script is customized or a new script is added, then this file must be updated. A description of each new MIB variable must be placed in the file */usr/provision/varDefs*.

Custom Script Reply Format

All custom scripts must report their data back in a specific format. The format is that of an SGITCL list of lists. There is a list for each host containing three elements:

- the hostname

- the data
- a status string

The hostname and status string are optional. The status message is a script-generated error message that, if it exists, is sent to the selected *provision* notifier.

Custom Script Argument List Format

All scripts are called with a command line of the format:

```
host-list [argument]. . .
```

The host list is a space-separated list of hostnames or addresses. All arguments are defined in the individual script. Common arguments are low and high bounds on the data. When data comes in that is out of bounds, a status message is returned.

Custom Script SGITCL Routine Locations

All of the scripts provided with *provision* are found in the *tlib* file */usr/provision/lib/provision.tlib*. If a provided script does not meet your needs then the script file can be copied and edited to create a custom script. When you have edited the new script, restart the *provisiond* daemon. When the daemon restarts, all tlib files are searched for unknown procedures, so custom scripts should be kept in a custom script tlib file, which can include routines from any other tlib library files as well.

Custom Script SGITCL Extensions

The SGITCL programming language provides several extensions to fetch information. These include *rstat*, SNMP, and the Silicon Graphics object management system. There are SGITCL help pages for each call in these three extensions and a reference page on each library.

The pvcontrolpanel Hosts Area

This area of the *pvcontrolpanel* main window lists all hosts and collections currently being monitored by or otherwise known to your *provision* session. An icon appears with each host's name. You can double-click a host icon and a dialogue window appears showing the items currently being monitored and any alarms received for the host or collection.

When an alarm comes in on a monitored host or collection, the object icon turns red to show you that an alarm has been received. When you double click the icon to view the alarm, the icon turns orange to show the alarm has been noted. If you then click the *Clear Alarms* button on the dialogue window, the icon returns to its default color.

The pvcontrolpanel Items to Monitor Area

All the items currently being monitored on any host are listed in this area. You can select which item is displayed in the Script Configuration Area and start and stop any item by clicking the provided buttons for each item being monitored.

The pvcontrolpanel Script Configuration Area

The script configuration section of the *pvcontrolpanel* window is functionally identical to the script configuration window used with *pvgraph*. This window is discussed in the section titled "The New Graph Window." Some key differences are:

- A button is provided for you to specify logging for the script or variable.
- A button and command window are provided for you to specify notification and a notification command.
- A regulation time selector is provided if you choose notification. This controls the frequency with which you will be notified if the specified limit is reached. For example, if you have set a notification alarm if the free disk space is under 10000 blocks and you have specified a monitoring interval of 30 seconds, you can specify a regulation time of 10 minutes and you will only be notified at that time interval, rather than every 30 seconds.

Creating a Log File With pvcontrolpanel

To create a log file with *pvcontrolpanel* click the button labeled *log* when you configure a variable or script. The name of the log file used appears at the top of the Items to Monitor section.

When you wish to review the log you must select the Close Log File menu option from the File menu or stop the actual logging. In order to stop all logging, close the log file, and not open a new log file, you must stop monitoring all items currently configured, delete all the items currently configured, and select the Close Log File menu option.

Alternately, you can select Close Log File from the File menu, and the log file for the selected item will be closed and a new one opened. You can then review the log that was closed. This method is recommended.

Note: If you change the host or parameter being logged with a *modify* command, the log file will not be restarted, nor will it register this change in any way. Thus, when the log is viewed it will be presented as if the parameters had not changed, and any information collected after the change is attributed to the initial configuration.

Using pvcontrol

The *provision* package offers a text-based interface that replicates the functions of the *pvcontrolpanel* graphical logging and notification tool. The text-based interface to *provision* can be run on any shell window, X-terminal, or character-based terminal. As **root**, enter the command:

```
pvcontrol
```

When you enter the command, you see the following prompt:

```
pvcontrol>
```

To see a list of commands, type **`pvhelp`** (or simply **`h`**) and press **`<Enter>`** at the *pvcontrol* prompt. You see the following list:

```
Provision commands:

list [log | notify]     - list currently monitored items
listAlarms hostName     - list alarms reported for specified
                          host
clearAlarms hostName    - clear alarms for specified host
getCurrentValue hostName scriptName args
                        - get the value of the specified
                          script or variable
add hostName scriptName interval notifyCommand regulationTime
notify|nonotify log|nolog args
                        - add item to monitor
modify itemID hostName scriptName interval notifyCommand
regulationTime notify|nonotify log|nolog args
                        - modify an item that is being
                          monitored
delete all              - delete all items
delete itemID           - delete item with specified itemID
```

```
start all               - start monitoring all items
stop all                - stop monitoring all items
start itemID            - start monitoring specified item
stop itemID             - stop monitoring specified item

showAvail               - list all available variables and
                          scripts
browser                 - start the snmp browser
closeLog                - close the log file
logStatus               - check the status of the log file
readConfig fileName     - read specified configuration file
saveConfig              - save configuration file
saveConfigAs fileName   - save configuration file to new name
pvhelp                  - display this help
quit                    - quit

pvcontrol>
```

The commands have the following meanings:

`list [log | notify]`

This command prints a list of items currently being logged or monitored for notification along with the itemID numbers. The itemID number is provided to allow more convenient manipulation of each specific item.

`listAlarms hostName`

This command directs *pvcontrol* to list any alarms reported for the specified host.

`clearAlarms hostName`

This command directs *pvcontrol* to clear all received alarms for the specified host.

`getCurrentValue hostName scriptName args`

This command directs *pvcontrol* to get the current value of the specified script or variable.

`add host scriptName interval notifyCommand regulationTime notify|nonotify log|nolog args`

This command adds a new item to the list. You must supply an entry for each argument shown. When your new item is accepted, the itemID is displayed along with the parameters you used. For example, the command

`add myhost interrupts 1 “mail dhhill” 1 nonotify log`

produces this response:

```
4 off myhost interrupts 1 off on - mail dhhill 1
```

The itemID in the displayed response is 4.

In this command and in the *modify* command:

- The interval is the frequency with which the script is run or the variable is checked.
- The Notify Command is the shell command to run to notify you if the limit (set on a per-script basis in the arguments) is reached.
- The regulation time specifies how frequently you are notified if the script is chronically past the limits you have specified.
- The notify and log switches specify notification and logging.
- The arguments required vary based on the script or variable you select.

Note that the script is not actually being monitored or logged until you enter the command *start all* or *start itemID*.

```
modify itemID host scriptName interval notifyCommand
regulationTime notify|nonotify log|nolog args
```

This command modifies an item being monitored. You provide the itemID of the item, and the new values for the item. For example, to change the item used above, you might enter the command:

`modify 4 myhost connections 5 “mail dhhill” 1 nonotify log 50`

With this command you have changed the script to *connections*, the interval to 5, and changed the argument to 50. The new parameters of the item are displayed for you.

In this command and in the *add* command:

- The interval is the frequency with which the script is run or the variable is checked.
- The Notify Command is the shell command to run to notify you if the limit (set on a per-script basis in the arguments) is reached.
- The regulation time specifies how frequently you are notified if the script is chronically past the limits you have specified.

- The notify and log switches specify notification and logging.
- The arguments required vary based on the script or variable you select.

Using flags to the modify command, you can modify individual parameters of an item. The following flags are recognized:

`-h` [**`hostname`**] – indicates new host name

`-r` [**`regulation time`**] – specifies a regulation time

`-s` [**`scriptname`**] – indicates new script

`-i` [**`interval`**] – indicates new interval

`-n` [**`on`** | **`off`**] – turns notification on or off

`-c` [**`notify command`**] – specifies a notification command

`-l` [**`on`** | **`off`**] – turns logging on or off

`-a` [**`arguments`**] – indicates new arguments

Use the following command syntax with flags:

```
modify itemID flag flag
```

`delete all` This command deletes all items currently configured.

`delete itemID` This command deletes only the item with the specified itemID.

`start all` This command starts monitoring all currently configured items.

`stop all` This command stops monitoring all currently monitored items.

`start itemID` This command starts monitoring the specified item.

`stop itemID` This command stops monitoring the specified item.

`showAvail` This command lists all available variables and scripts in text. The list of variables is quite long, and definitions of the variables can be obtained only through the SNMP Browser on a graphics system. Descriptions of the available scripts are in the section titled “The Available Variable and Script Window.”

`browser` This command starts the SNMP browser. The browser is a graphical-only tool, and so cannot display on a non-graphics system. The browser is described in the section titled “Obtaining Descriptions of Variables” in Appendix A.

`closeLog` This command directs *pvcontrol* to close the log file. A new log file is opened immediately.

`logStatus` This command checks and reports the status of the log file.

`readConfig fileName`
This command directs *pvcontrol* to read the specified configuration file and use the monitoring and logging settings found in it.

`saveConfig` This command saves the current configuration in the default configuration file.

`saveConfigAs fileName`
This command saves the current configuration to a new configuration file.

`help` This command displays the list of available commands.

`quit` This command quits *pvcontrol*.

The commands available through *pvcontrol* are substantially similar to those available through the graphical *pvcontrolpanel*, and the description of that utility provides further helpful information.

Creating a Log File With pvcontrol

To create a log file with *pvcontrol* you must select logging as a command line option when you use the *add* or *modify* commands to select a variable or script. The name of the log file used is displayed in the following manner:

```
The log file is:/usr/provision/Logs/0.950201-22:34:26
```

When you wish to review the log you can stop all monitoring action by deleting all items and entering the *closeLog* command at the *pvcontrol* prompt to stop all monitoring and logging, or you can simply enter the *closeLog* command and the current log file will be closed and a new one opened.

Note: If you change the host or parameter being logged with a *modify* command, the log file will not be restarted, nor will it register this change in any way. Thus, when the log is viewed it will be presented as if the parameters had not changed, and any information collected after the change is attributed to the initial configuration.

Using pvgraph

The second tool available directly from the *provision* window is *pvgraph*. This tool allows you to select command scripts and graph the values of certain variables and system statistics in a window. When you first bring up *pvgraph*, you see the following window (shown in Figure 1-5):

Figure 1-5 The pvgraph Window

The *pvgraph* window is blank when it comes up on your screen, and you create graphs by selecting options from the menu bar. The menu bar has three menus: File, Graphs, and Help. The options available in these menus are listed below.

The pvgraph File Menu

The File menu has the following choices:

Read Config...
: This option reads a configuration file that specifies a set of graphs to run. You can also drag an icon representing a previously stored config file from the directory view onto the *pvgraph* icon to start *pvgraph* with that configuration.

Save Config
: This option saves the current graphing configuration to a file.

Save Config As...
: This option saves the current graphing configuration to a new filename.

Show Log Files...
: This option brings up the pvgraph window (see "Using pvgraph to View a Log") to display the contents of a log file recording of your *pvcontrolpanel* activity. A log file simply contains a series of values for a script or variable accumulated over a period of time.

Quit
: Quits *pvgraph* and ends all graphing.

The pvgraph Graphs Menu

The Graphs menu has the following choices:

Add A Graph
: Use this choice to add a new graph. This choice brings up the New Graph window, described in the section titled "The New Graph Window."

Modify Selected Graph
: Use this choice to change an existing graph. This choice brings up the Edit Selected Graph window with the parameters of the selected graph displayed in the fields for modification.

Delete Selected Graph
: This choice deletes the selected graph.

Change Style of Selected Graph
: This choice brings up the Graph Styles window. This window is described completely in the section titled "The Graph Style Window."

Change Parameters of All Graphs
: This choice brings up the Graph Parameters window. This window is described completely in the section titled "The Graph Parameters Window."

Show Alarms This choice shows all received *provision* alarms for the graphed items. See "Working With Graph Alarms" for more information.

Clear Alarms This choice clears all received *provision* notification alarms. See "Working With Graph Alarms" for more information.

Start Selected Graph
This choice starts a previously stopped graph.

Stop Selected Graph
This choice stops a selected graph.

Start All Graphs
This choice starts all previously stopped graphs.

Stop All Graphs
This choice stops all graphing.

The pvgraph Help Menu

The Help menu offers online help with *pvgraph*.

The New Graph Window

When you select the menu choice to add a graph to your *pvgraph* window, you see the new window shown in Figure 1-6.

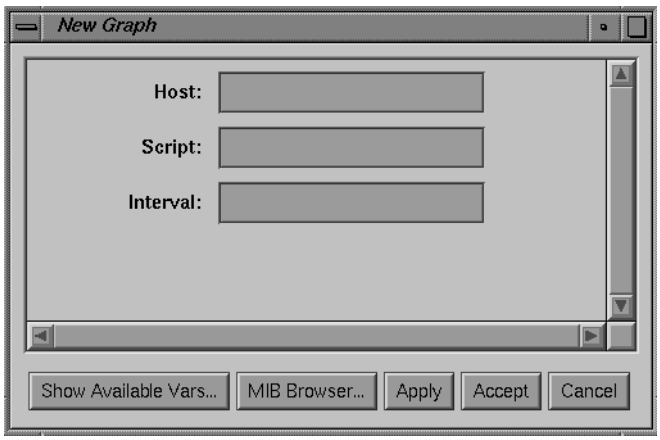

Figure 1-6 The New Graph Window

This window has several fields for you to fill in the parameters of the graph you wish to make. There are also other fields that may appear as you enter information. Certain scripts require more parameters than others, and if you enter the name of such a script in the Script field, additional fields appear below the basic fields. The fields require the following kinds of information:

Host Field — This field takes the name of a host. The host must be connected with the local system by the network, and the host must be running the *snmpd* daemon.

Script Field — This field takes a script or variable name. If you do not know the name of the script or variable you wish to use, press the *Show Available Vars* button at the bottom of the window and the Available Variable and Script window will appear. This window is described in the section titled "The Available Variable and Script Window." All distributed scripts are described in that section of this chapter.

Interval Field — This field is where you specify the time interval (in seconds) at which the script will run and the results will be displayed. For example, if you enter **1**, the script will run and the graph will be updated every second.

Arguments Fields — These fields are where any necessary arguments to the script are specified. When you enter a variable or script, appropriate fields appear for each needed argument. If the script or variable is not known to *provision*, a field titled *arguments* appears to receive any arguments

required. To make a new script or variable known to *provision*, an entry must be placed in the */usr/provision/scriptDefs* or */usr/provision/varDefs* file.

At the bottom of the New Graph window, there are five buttons, labeled *Show Available Vars*, *MIB Browser*, *Apply*, *Accept*, and *Cancel*. The *Show Available Vars* button brings up the Available Variable and Script window, described in the section titled "The Available Variable and Script Window." The SNMP Browser button brings up the Browser, described in Appendix A, "The MIB Browser." Use the *Apply* button to add your graph and leave the New Graph window on the screen, or the *Accept* button to add the new graph and remove the New Graph window. The *Cancel* button removes the New Graph window without applying your changes.

If you add additional graphs, the window is subdivided for each graph. When you have more graphs than can fit on the window, you must enlarge the window to accommodate the new graphs.

Working With Graphs

When you have applied your graph to the *pvgraph* main window, the center of the window looks something like that shown in Figure 1-7.

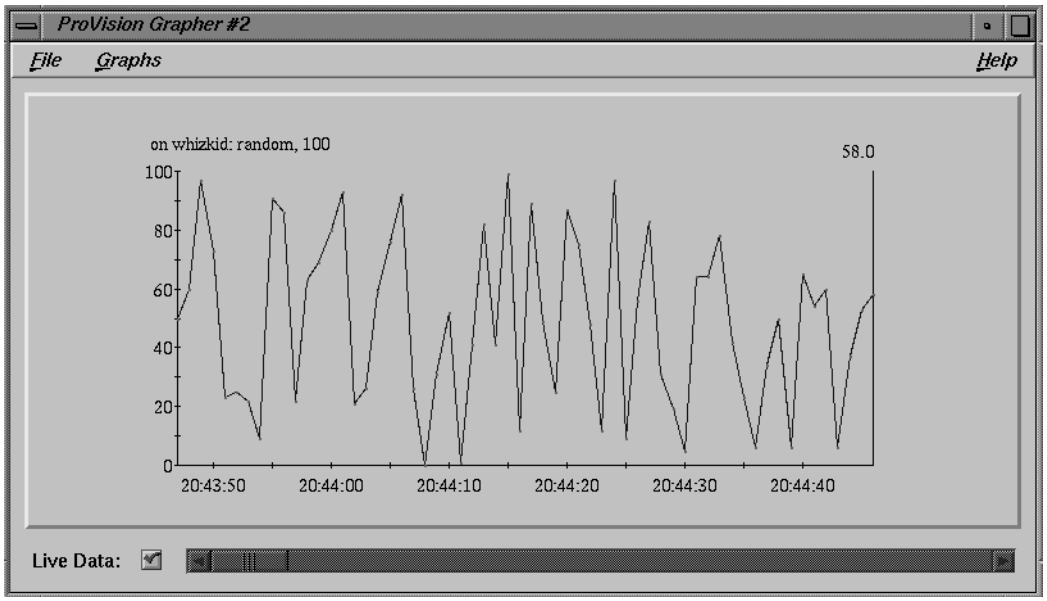

Figure 1-7 The pvgraph Window With One Graph

Note that there is a check box and a slider present at the bottom of the *pvgraph* window. When you begin your graph, the check box has a check mark, indicating that the graph being made is using data as it is collected in real time. The slider is grayed-out and inoperable. At any time you can click on this check box and the entire history of the graph is made available to you. The slider bar becomes active and you can use it to review your graph. When you wish to return to live graphing, simply click the check box again and the graph is updated. No data is lost during your review operation.

Working With Graph Alarms

When the script results or variable values being graphed exceed the low and high limits you specified when you added or modified the graph, an alarm is set off for you. This alarm is a visual cue to check the item being graphed. When the value of the script or variable has gone out of bounds, the graph turns red, as shown in Figure 1-8.

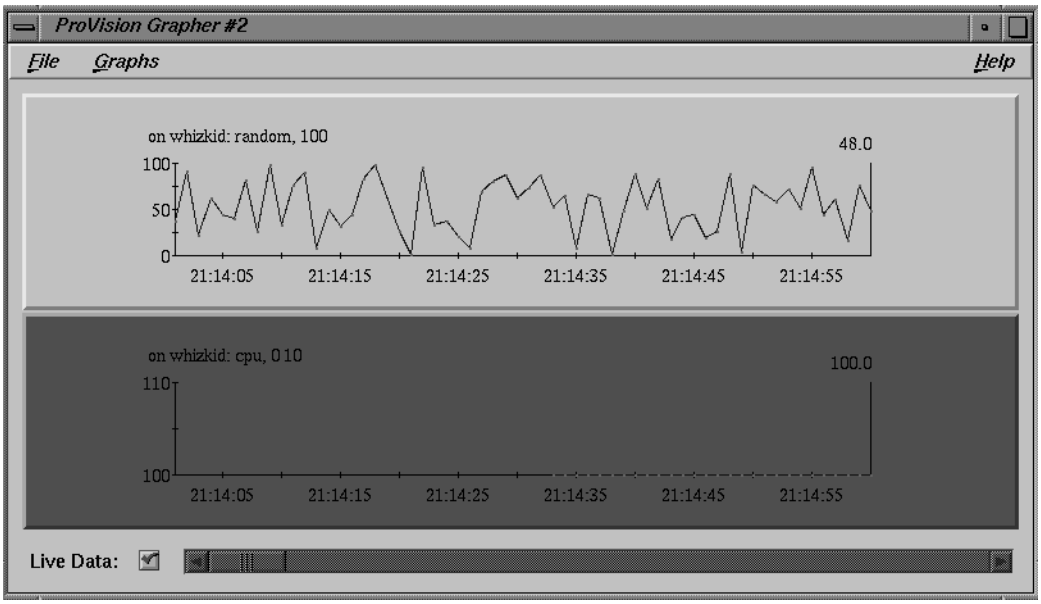

Figure 1-8 A Graph With an Alarm Showing

To clear alarms, select the Clear Alarms menu option from the Graphs menu. If you select the Show Alarms option, a window appears with a log of all alarms received since the last Clear Alarms command, or since the beginning of the *pvgraph* session.

Selecting a Graph

You can select a graph for further operations by placing the mouse cursor in the window section of the graph and clicking the left mouse button. The background of the selected graph turns yellow. Only one graph may be selected at a time. You may perform the following operations from the Graphs menu on selected graphs:

Modify Selected Graph
: This choice brings up the Modify Selected Graph window with the parameters of the selected graph displayed in the fields for modification. This window is identical to the New Graph window except for the title.

Delete Selected Graph
: This choice deletes the selected graph.

Change Style of Selected Graph
: This choice brings up the Graph Style window. This window is described completely in the section titled "The Graph Style Window."

Stop Selected Graph
: This choice stops the selected graph.

Start Selected Graph
: This choice starts the selected graph.

The Graph Parameters Window

When you select the "Change Parameters of All Graphs" menu item from the *pvgraph* Graphs menu, you see the new window shown in Figure 1-9.

Figure 1-9 The Graph Parameters Window

What you are changing is the period of graph-time that is displayed in the window at any given moment. The parameters you can change are the graph width value and the time unit. The width value is simply the number of increments of the selected time unit. In the above example, the width value is 1 and the time unit is *minutes* for a width of 1 minute. You may select *1*, *2*, *5*, *10*, *20*, or *30* for the width value, and one of *seconds*, *minutes*, *hours*, *days*, or *weeks* for the time unit.

Once you have made your selections, you may press the *Apply* button to apply the change and leave the Graph Parameters window on the screen, or the *Accept* button to apply the changes and remove the Graph Parameters window. The *Cancel* button removes the window without applying your changes. The *Help* button invokes *provision*'s online help utility.

The Graph Style Window

If you select Change Style of Selected Graph from the Graphs menu in *pvgraph*, you see a new window on your screen, as shown in Figure 1-10.

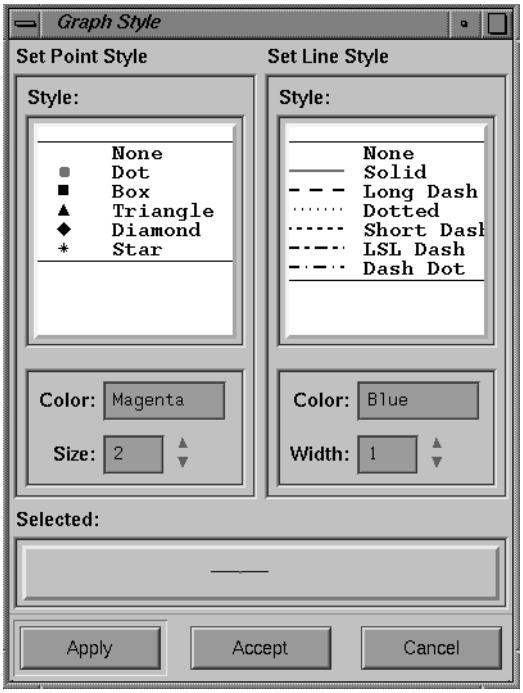

Figure 1-10 The Graph Style Window

This window allows you to select and modify the way the selected graph is presented in *pvgraph*. When the polling interval arrives on a graph, the new value of a variable (or the value of the output of the script being graphed) is placed on the graph as a point, and a line is drawn between the new point and the previous point. You may select the shape of the point marker, its size and color, and the style, width, and color of the connecting line. Click the style of marker and line you prefer. Any valid X color or value may be named in the Color field, and you can use the arrow buttons to increase or decrease the size of the line or marker.

When you have made your selection, press the *Apply* button to apply the new format to your graph, or the *Cancel* button to discard your unapplied changes.

The provision Configuration File

At any time during your *pvcontrolpanel* or *pvgraph* session, you can save the current graphing and/or monitoring selections in a configuration file. The options are in the File menu in both utilities:

```
Read Config...
Save Config
Save Config as...
```

When you first save your current state, use the “Save Config as...” option. When you select this option, you see a file selection window for your current working directory, such as that shown in Figure 1-11.

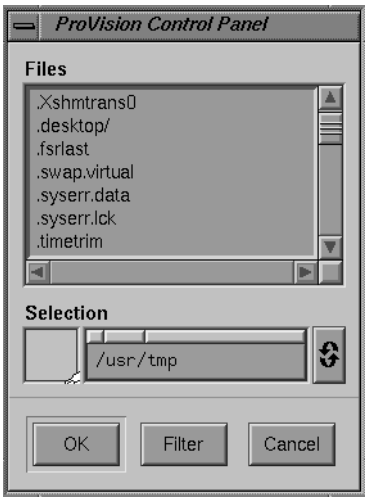

Figure 1-11 A File Selection Window

Select a new filename for your configuration file and click the *OK* button when you are satisfied with your selection. Your current state is now saved. If you wish to save your current selections again later, using Save Config option from the Files menu of *pvcontrolpanel* or *pvgraph* will automatically bring up the file selection window with the most recently used config file specified. You can, however, change the name so as not to overwrite the existing config file. Using config files, you can create templates for commonly used monitoring and graphing scenarios. For example, you can have preset configuration files to monitor all systems’ network traffic or the swap rates on your servers.

Each *provision* configuration file is written in clear text and looks similar to the following example:

```
# provision config file written Mon Feb  6 14:29:52 PST 1995 by
pvcontrolpanel
off on off  random provision:random  wookie 1 {}  -1 0 {200}
off on off  ifInOctets_1 1.3.6.1.2.1.2.2.1.10.1  wookie 1 {}  10 0 {2}
{3}
off on off  ifOutOctets_1 1.3.6.1.2.1.2.2.1.16.1  wookie 1 {}  10 0
{2} {3}
off on off  random provision:random  5 {}  600 0 {100}
```

Using pvgraph to View a Log

You can view log files created with *pvcontrol* or *pvcontrolpanel* using *pvgraph*. For more information on log files and how they are created, see “Creating a Log File With pvcontrolpanel” on page 23 or “Creating a Log File With pvcontrol” on page 28 in this chapter. A log file is simply a file containing a series of values for a script or variable accumulated over a period of time. The *provision* application stores the log files in the log directory */usr/provision/Logs.* A log file is actually contained in two filenames in that directory. For example, a log might be placed in filenames similar to *0.950201-21:18:33.Desc*, and *0.950201-21:18:33.Data*. Filenames ending in *.Desc* contain information about what was logged, and filenames ending in *.Data* contain the actual log information.

To view a log as a graph, use the command syntax

`pvgraph` *filename*

to invoke *pvgraph* in log file mode. The *filename* argument can be any of the three filenames that refer to the desired log. For example, using the example filenames as shown above, you could invoke *pvgraph* in these ways:

```
pvgraph 0.950201-21:18:33
```

```
pvgraph 0.950201-21:18:33.Desc
```

```
pvgraph 0.950201-21:18:33.Data
```

Each of the above commands results in the same action by *pvgraph*. By default, *pvgraph* looks for the given filename in */usr/provision/Logs*, but you can specify any log file in any directory by issuing the pathname of the file on the command line.

When you invoke *pvgraph* with a log file name as a command line argument, *pvgraph* does not connect with the *provisiond* daemon as usual. Instead, the named log file is loaded. The log is not displayed as a graph, though, until you use Add A Graph from the Graphs menu. When you use this command, you see a different New Graph window, similar to Figure 1-12.

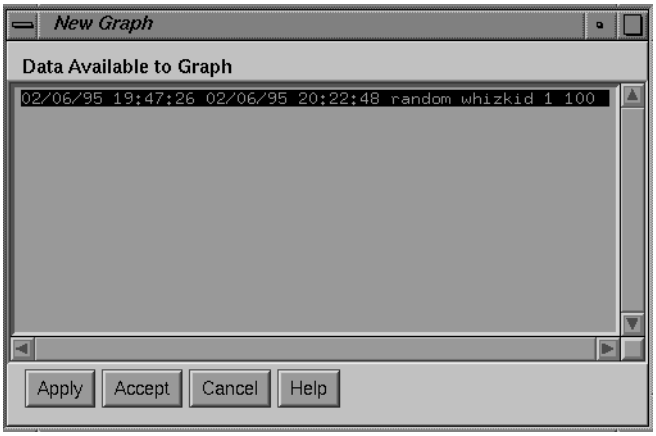

Figure 1-12 The New Graph Window with Log File Information

When you view a log file, it does not scroll by as usual; you must use the slider at the bottom of the window to move forward and back throughout the log.

If you are invoking *pvgraph* from the desktop or directory view rather than as a shell command, you can drag the file icon for the log you wish to view and drop it on the *pvgraph* icon and *pvgraph* will come up with the log loaded.

Appendix A

The MIB Browser

The MIB Browser is available through either *pvcontrolpanel* or *pvgraph*. It lets you select a node on your network and view and change the contents of one or more Management Information Bases (MIBs) for that node. Browser communicates with a node that you select using Simple Network Management Protocol (SNMP). The node can be a workstation, router, bridge, hub, or gateway—any device that has an IP address and implements the SNMP protocol and agent.

Several MIB specifications are provided with Browser. The supplied MIB specifications are:

- `hp-ux_sgi`
- `mib-2`
- `armon`

Browser is designed to be used by network managers experienced in managing various devices on the network. This section assumes that you are familiar with SNMP management terminology and technology, especially the MIBs for different devices. If you are not familiar with this terminology, the section titled “SNMP Management Glossary” on page 62 defines the basic terms.

This section explains how to

- start Browser
- use the Browser File menu
- use the Browser main window to specify the node you want to browse and begin navigating the SNMP Containment Tree
- navigate the SNMP Containment Tree to view subtrees, tables, and variables
- get descriptions of variables
- get and set the values of variables

In addition, an example of using Browser is provided. For complete information on Browser command line options, see the *browser*(1M) reference page.

Note: To enable Browser to get and set MIB variables on a Silicon Graphics workstation, that workstation must be running the SNMP daemon *snmpd*(1M), and your Display Station must be authorized in the file */etc/snmpd.auth* on that workstation. See "Authorizing Browsing" for details.

Caution: With proper authorization, Browser lets you change some MIB variable values on devices you browse. Because MIB variable values can be critical to the operation of a device and your network, do not change values unless you understand the effects of your changes.

SNMP Agents

Browser uses Simple Network Management Protocol (SNMP) agents to obtain information. The SNMP agent for Silicon Graphics workstations is *snmpd*(1M). Vendor-specific SNMP agents are used to obtain information about other types of nodes (see Figure A-1). For more information on SNMP see the section titled "SNMP Management Glossary." For information on enabling SNMP agents, see "Enabling SNMP Agents."

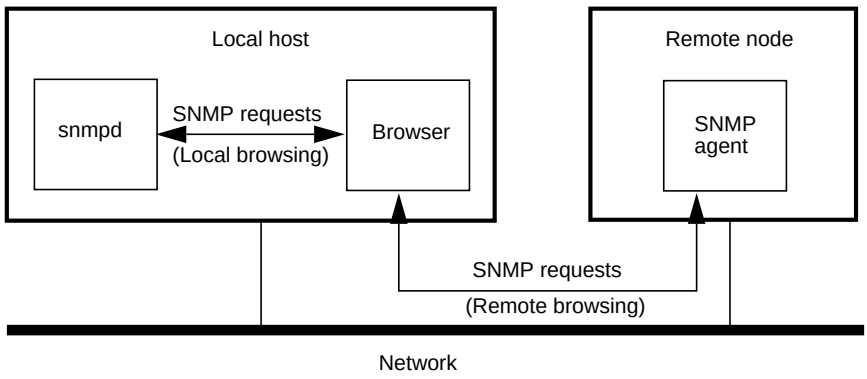

Figure A-1 SNMP Agents and Browser

Enabling SNMP Agents

Browser must communicate with the SNMP agent on each node you wish to browse. SNMP agents and the procedures for enabling them are vendor-specific. The procedure for enabling the SNMP agent on Silicon Graphics workstations is described below. For other types of nodes, contact the system administrator for that node for help in enabling SNMP on that node.

The Silicon Graphics SNMP agent is *snmpd*(1M). The SNMP agent software is distributed with IRIX, and the `hp-ux_sgi` MIB and agent are distributed with provision. To configure a workstation so that *snmpd* is started automatically when the system is rebooted, copy the snmpd executable file to the workstation, and enter this command on the workstation as **root**:

```
chkconfig -f snmpd on
```

To see if the daemon is already running, enter this command:

```
ps -e | grep snmpd
```

If there is no output from this command, *snmpd* is not running. Enter this command as **root** to start *snmpd*:

```
snmpd
```

Authorizing Browsing

No special authorization other than a valid community string is required to browse on nodes other than Silicon Graphics workstations. (See "Browser Main Window" and "SNMP Management Glossary" for more information about community strings.)

If you want to get and set MIB variables on a Silicon Graphics workstation using Browser, you must perform several setup steps in addition to providing a valid community string while using Browser: confirm that the workstation you are browsing has SNMP agent software running; start it if necessary (see "Enabling SNMP Agents," in this chapter); and authorize your Display Station to browse on that workstation.

To be authorized to browse a Silicon Graphics workstation, the Display Station's host name must be specified in the file */etc/snmpd.auth* on the workstation you are browsing. You must be superuser (**root**) to read or write */etc/snmpd.auth*. For security reasons, the owner and permissions of this file should not be changed.

As an example, suppose that you want to browse a Silicon Graphics workstation named `tahoe`. Your workstation's name is `sequoia`. First, confirm that *snmpd* is running on `tahoe`:

```
rsh guest@tahoe 'ps -e | grep snmpd'
```

Assuming that it is running, log onto `tahoe` as superuser and add this line to */etc/snmpd.auth*:

```
accept sequoia:*
```

This line authorizes anyone using your workstation to browse the workstation `tahoe` when they give any community string. These users can perform both `get` and `set` operations.

By default, */etc/snmpd.auth* contains this authorization line:

```
accept  *:public/get
```

This line authorizes any user from any host who provides the community `public` to get variable values for this workstation.

See the *snmpd*(1M) reference page and the file for more information about the syntax used in this file.

About the Browser

Browser enables you to walk the tree of information represented by the MIBs and the SNMP Containment Tree, and to get the values of MIB variables. While you are using Browser, you can save the variable values that you receive to a file. You can set MIB variables if the SNMP and MIB implementations on the node you are browsing allow it and the community string you provide authorizes it.

Starting Browser

To start Browser, click the *browser* button from a *provision* window.

The Browser main window appears. An example is shown in Figure A-2.

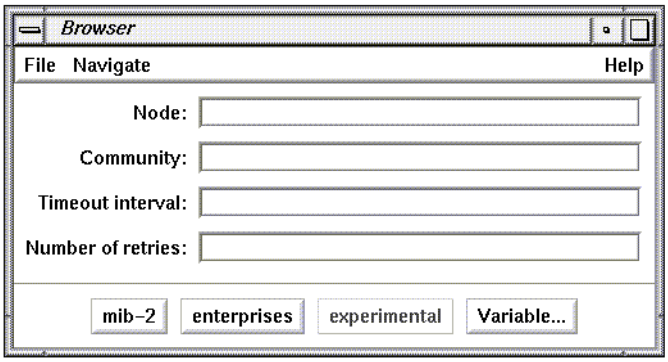

Figure A-2 Browser Main Window

Browser Main Window

The entry fields in the Browser main window enable you to specify the node you wish to browse, a community string, a time-out value for accessing the SNMP agent on the node, and the number of retries to make when attempting to access a remote node.

When you invoke Browser, the entry field shown in Figure A-3 contains the name of your workstation. You can replace it with the name or address of the node you want to browse. A blank entry field is the same as specifying the name of your workstation.

Figure A-3 Node Entry Field

The Community entry field shown in Figure A-4 contains the community string that is to be used in the SNMP packets sent to the node. The community string is an authorization password for the node you browse on. On Silicon Graphics workstations, valid community strings and other authorization information is specified in the file */etc/snmpd.auth*. The default community string on Silicon Graphics workstations is "public." For other types of nodes, such as routers and bridges, the community string is specified for each device by a system administrator. A valid community string must be supplied in order to use Browser to view MIB information.

Figure A-4 Community Entry Field

If Browser doesn't receive a reply from the SNMP agent on the specified node within the time-out value, it will try again. The default time-out value, shown in Figure A-5, is 5 seconds.

Figure A-5 Time-out Interval Entry Field

The Number of retries entry field, shown in Figure A-6, specifies the number of retries when there has been no reply from the node. The default is 3.

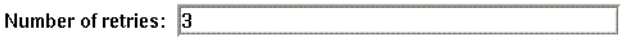

Figure A-6 Number of Retries Entry Field

The *mib-2*, *enterprises*, and *experimental* buttons in the Browser main window, shown in Figure A-7, provide quick ways to specify what you want to browse: the `mib-2` MIB, or the `enterprises` or `experimental` nodes in the SNMP Containment Tree, respectively. When you click these buttons, a Subtree window appears. Subtree windows and Table windows are described in the next section. These buttons are grayed out if no MIB specifications in that portion of the SNMP Containment Tree are available to Browser. The term "grayed out" means that the button title is gray at that moment, rather than black. This indicates that the function or service represented by that button is unavailable.

Figure A-7 mib-2, enterprises, and experimental Buttons

When you click the *Variable...* button, shown in Figure A-8, a Variable window appears. This window is used to get and set the values of specific MIB variables. It is explained in detail in "Obtaining and Setting Values Using the Variable Window" in this chapter.

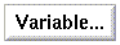

Figure A-8 Variable... Button

Browser Subtree and Table Windows

To display MIB information, Browser uses two types of windows: Subtree windows and Table windows. For every nonleaf node in the SNMP Containment Tree, Browser displays one of these types of windows:

- a Subtree window showing the subtrees of that node
- a Subtree window showing the variables and/or tables of that node
- a Table window showing the array of table variables in that table

The remainder of this section discusses examples of these windows.

Subtree Windows That Show Subtrees

Figure A-9 shows the Subtree window for `mib-2`. It is an example of a Subtree window for a subtree that contains other subtrees.

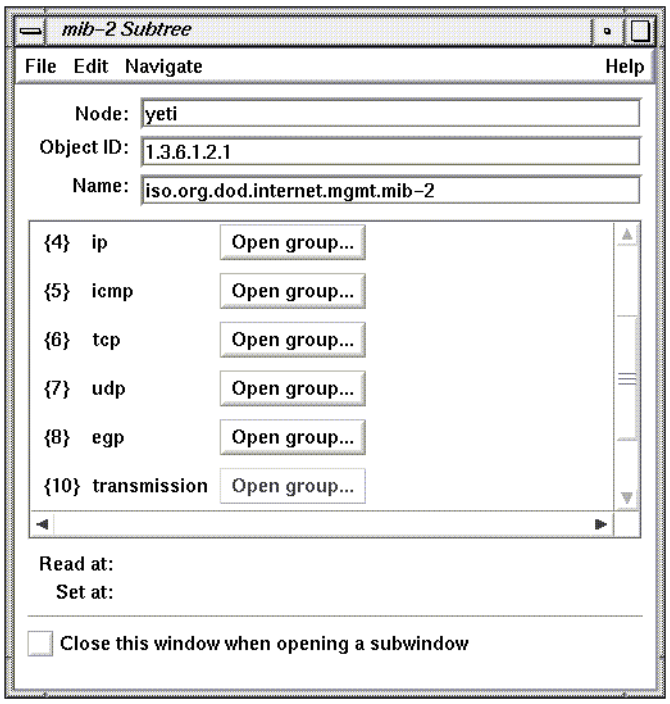

Figure A-9 Subtree Window Showing Subtree Objects

The Node entry field, shown in Figure A-10, contains the node name or address you specified in the Browser main window.

Figure A-10 Node Entry Field

The Object ID and Name entry fields, shown in Figure A-11, contain two different representations of the name of the subtree displayed in the window. The Object ID entry

field contains the numeric representation of the name (dot separated object numbers) and the Name entry field contains the text string representation (dot-separated object names).

Figure A-11 Object ID and Name Entry Fields

The scrolling display area in the center of the window contains one line for each object in the subtree, such as the `udp` line shown in Figure A-12. The line begins with the object's number in curly braces followed by its object name. Clicking the *Open group...* button brings up a Subtree window for the object on this line. Its use is described more fully in "Navigation Using Buttons in the Subtree and Table Windows" in this chapter. A grayed-out button means there are no variables under this object in the MIB.

Figure A-12 Object in a Display Area

The Read At line provides status information during a "`get`" operation (see "Obtaining, Setting, and Saving Variable Values" in this chapter), which is replaced by the current time after the operation is completed. An example is shown in Figure A-13.

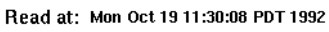

Figure A-13 Read At Line

The current time is displayed on the Set At line after a "`set`" operation (see "Obtaining, Setting, and Saving Variable Values" in this chapter). An example is shown in Figure A-14.

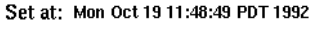

Figure A-14 Set At Line

Checking the "Close this window when opening a subwindow" check box, shown in Figure A-15, specifies that you want this Subtree window to be closed when a new Subtree or Table window for a node in this subtree is opened. By default, each of the

Subtree or Table windows you open for subtrees or tables within this subtree will have the same setting.

Figure A-15 "Close This Window When Opening a Subwindow" Check Box

Subtree Windows That Show Variables and Tables

Figure A-16 shows the Subtree window for `mib-2.udp`. It is an example of a Subtree window that shows the variables and/or tables of that subtree (in MIB terminology, this type of subtree is called a *group*).

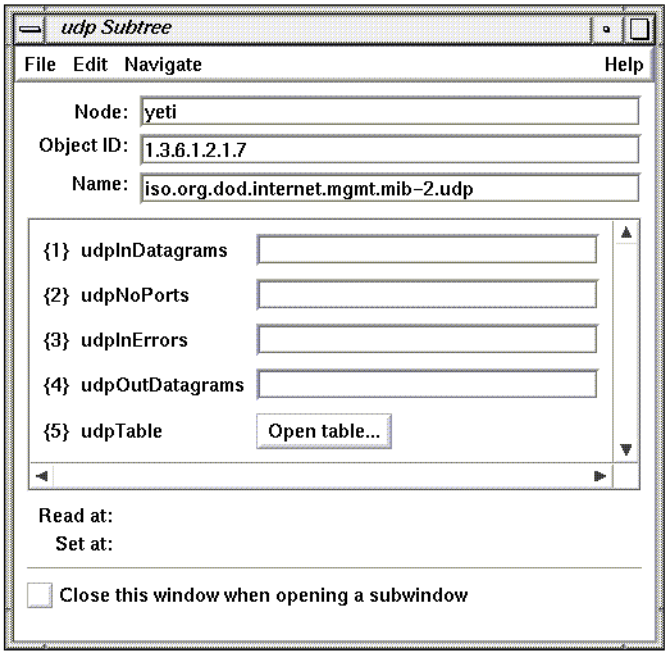

Figure A-16 Subtree Window Showing Variables and a Table

Most portions of this type of Subtree window are the same as the Subtree windows described in "Subtree Windows That Show Subtrees" in this chapter. However, the

display area of this type of Subtree window contains entry fields and *Open table...* buttons rather than *Open group...* buttons.

Variables in the subtree are shown in the display area as an object number, an object name, and an entry field, as shown in Figure A-17. When the object number (in braces) is appended to the object ID at the top of the window, it forms the complete object ID for the object. The entry field is gray for variables whose values are defined as read-only in the MIB and pink for variables that are defined as read-write or write-only in the MIB. If the entry field is pink, you can set the value of that variable (see "Obtaining and Setting Values Using the Edit Menu of a Subtree Window," in this chapter).

Figure A-17 Variable Line in a Subtree Display Area

Tables in the subtree have lines that include their object number, their name, and the *Open table...* button, as shown in Figure A-18. When you click an *Open table...* button, a Table window, described in the next section, appears.

Figure A-18 Table Line in a Subtree Display Area

Browser Table Windows

Figure A-19 shows the default Table window for `mib-2.udp.udpTable`. When a Table window appears, the display area contains only the names of the table variables. Entry fields appear for the variables as you retrieve their values with "Get next row" in the Edit menu. (See "Obtaining and Setting Values Using the Edit Menu of a Table Window," in this chapter.)

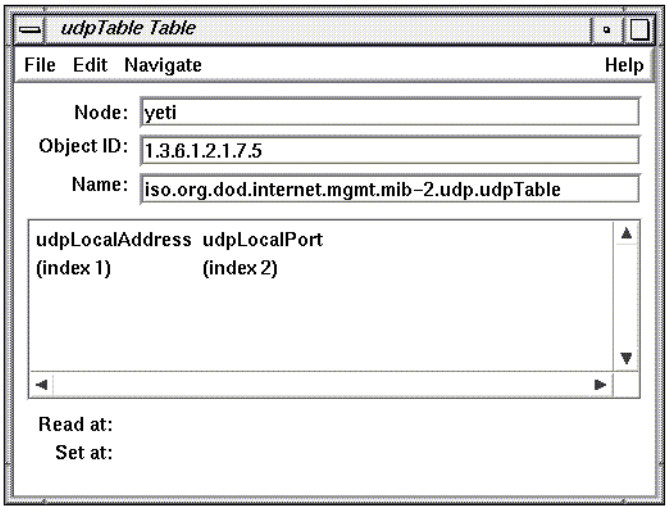

Figure A-19 Browser Table Window

Navigating the SNMP Containment Tree

With Browser you can open a Subtree window for each subtree in an MIB that you want to browse and a Table window for each table in an MIB that you want to browse. Browser buttons and menus enable you to specify the subtree or table you want to view. When you use these buttons and menus, you are "navigating" the MIB Tree. The three navigation methods are described in the following sections.

Navigation Using the Buttons in the Main Window

The *mib-2*, *enterprises*, and *experimental* buttons in the Browser main window provide three starting points for browsing the SNMP Containment Tree. Clicking the *mib-2* button brings up a Subtree window for the MIB-II MIB. Clicking the *enterprises* and *experimental* buttons brings up Subtree windows for the Enterprises and Experimental subtrees, respectively.

Navigation Using the Navigate Menu

To use the Navigate menu from any window, follow these steps:

1. Press the left mouse button on Navigate in the menu bar.

   In the menu that appears, each choice except the last is the name of a subtree or table that is an object in the subtree in the window. Choices are highlighted as you move the cursor on them; if they have a rollover menu, it appears automatically. Figure A-20 shows an example of the Navigate menu at the `mib-2` subtree with the cursor on `udp`.

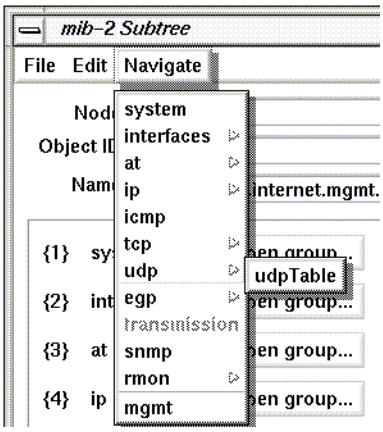

Figure A-20 Navigate Menu

2. To view one of the subtrees of the subtree in the window, select one of the choices on the Navigate menu (not on a rollover menu).

   If you make the subtree selection shown in Figure A-20, the window shown in Figure A-16 appears.

3. To view subtrees or tables farther down in the hierarchy, move the cursor to a choice on a rollover menu and release the mouse button. In this way you can traverse the entire width and length of the subtree in the window.

4. To view the parent of the current subtree, select the last choice on the Navigate menu. It is the name of the parent of the subtree in the window.

Navigation Using Buttons in the Subtree and Table Windows

Figure A-9 shows an example of *Open group...* buttons in the display area of the `mib-2` Subtree window. When you click one of these buttons, a new Subtree window appears for this object. It is equivalent to choosing this subtree from the Navigate menu.

In Figure A-16, `udpTable` is a table and has an *Open table...* button. Clicking an *Open table...* button is equivalent to choosing the table from the Navigate menu. Figure A-19 shows the Table window for `udpTable` that appears when you click this button.

Obtaining Descriptions of Variables

To get a description of each of the objects in a subtree or each of the variables in a table, select "Description" from the Help menu of the Subtree or Table window. A Description window appears. Figure A-21 shows an example.

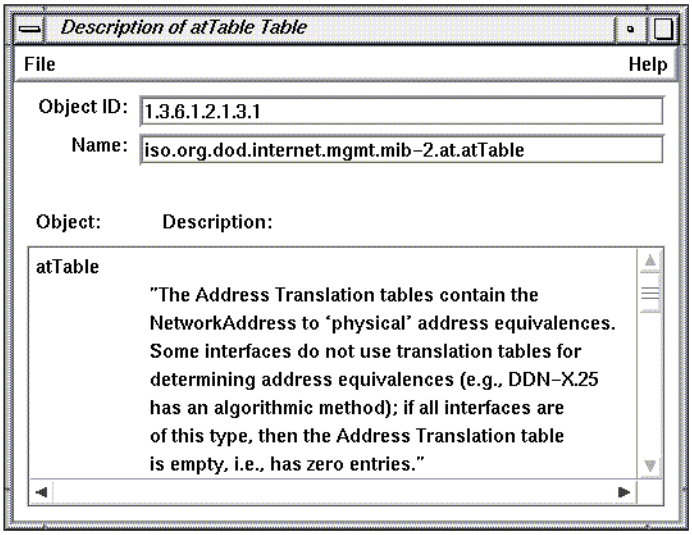

Figure A-21 Description Window

Obtaining, Setting, and Saving Variable Values

Browser enables you to obtain the values of MIB variables and set them if you have write access. (Write access is determined by the type of the variable and by your community. See "Authorizing Browsing" and "SNMP Management Glossary.") Three types of Browser windows can be used to get and set variables:

- The Variable window enables you to get and set individual variables. The Variable window is described in "Obtaining and Setting Values Using the Variable Window," in this chapter.
- Subtree windows enable you to get and set variables that aren't part of tables. "Obtaining and Setting Values Using the Edit Menu of a Subtree Window" in this chapter describes how to do this.
- Table windows enable you to get and set variables that are part of tables. "Obtaining and Setting Values Using the Edit Menu of a Table Window," in this chapter, describes how to do this.

Obtaining and Setting Values Using the Variable Window

Follow the steps below to use the Variable window to get and set variable values.

1. Click the *Variable...* button in the Browser main window. The window shown in Figure A-22 appears.

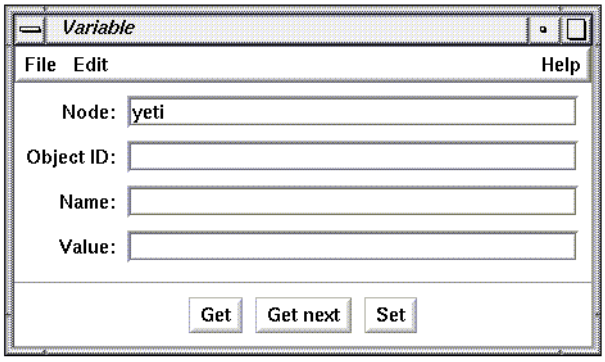

Figure A-22 Variable Window

2. If you want to specify a variable by object identifier, fill in the Object ID entry field with the object identifier and append the following:

   `.0`

   The "dot zero" specifies that you want the value of the object; if you forget to use `.0`, Browser adds it automatically. For example, to specify `mib-2.ip.ipForwarding` (`1.3.6.1.2.1.4.1`), the Object ID entry field should look like the one shown in Figure A-23.

Figure A-23 Object ID Entry Field

3. To specify a variable in a table, enter its object identifier in the Object ID entry field. To construct its object identifier, you can use the object identifier of the table and append:

   `.1.x.y`

   The column number is represented by *x* (beginning with 1), and *y* is the value of `index` for the row you want. For example, the object identifier for the `ifDescr` variable (column 2) in the first row (index value of 1) of the `mib-2.interfaces.ifTable` (`1.3.6.1.2.1.2.2`) table is `1.3.6.1.2.1.2.2.1.2.1`. If the table you are using has more than one `index` column, create *y* by specifying each `index` value in order and separating them with periods. For example, if the value of `index1` is `127.1.9` and the value of `index2` is `7`, *y* is `127.1.9.7`.

4. If you want to specify the variable by name, fill in the Name entry field. You need not type in the complete hierarchical name, just the last component. Adding `.0` to the name is optional. If the Object ID and the name you fill in don't match, the Object ID is used.

5. To obtain the value of the variable, click the *Get* button. The value of the variable appears in the Value entry field. The Name entry field is automatically modified so that it contains the complete hierarchical name.

6. To set the value of a variable, enter the value in the Value entry field and click the *Set* button.

7. To obtain the value of the next variable, click the *Get next* button. Depth-first search is used to determine the next variable, so the right-most component of the object identifier varies fastest as the tree is traversed with *Get next*.
8. Continue obtaining and setting variables as necessary by modifying the Object ID, Name, and/or Value entry fields and using the *Get*, *Get next*, and *Set* buttons.

Obtaining and Setting Values Using the Edit Menu of a Subtree Window

You can obtain and set the values of variables from a Subtree window using the Edit menu:

1. Bring up the Subtree window that contains the variable whose value you want to obtain or set (see "Navigating the SNMP Containment Tree" in this chapter).
2. Select Get from the Edit menu to obtain the values of all of the variables. The current time is displayed on the Read At line.
3. Make changes in the entry fields for any variables whose values you want to change. Only variables whose entry fields are pink may be changed.
4. Select Set from the Edit menu to change variable values. The current time is displayed on the Set At line.

Obtaining and Setting Values Using the Edit Menu of a Table Window

You can obtain and set the values of variables in a table from its Table window using the Edit menu:

1. Bring up the Table window for the table you are interested in (see "Navigation Using the Navigate Menu" in this chapter).
2. To obtain the first row of variables in the table, select "Get next row" from the Edit menu. The current time is displayed on the Read At line.
3. To obtain other rows for the table, select "Get next row" from the Edit menu as many times as necessary.
4. Make changes in the entry fields for any variables whose values you want to change.
5. Select Set from the Edit menu to change variable values. The current time is displayed on the Set At line.

Browser File Menu

The File menu in each window gives you one or more of the window management and quitting choices listed below.

Save MIB Values
: Save MIB values for this subtree for all nodes in the subtree that have open windows to a file. The values are appended to the file that you last specified with Save MIB Values As... .

Save MIB Values As...
: Save MIB values for this subtree for all nodes in the subtree that have open windows to a file. When you select this choice, a file prompter appears. Use it to specify a filename. The filenames are sorted by object identifier.

Pop Main Window
: Display the Browser main window. This is useful if you have many windows open and want to locate the Browser main window quickly.

Close Lower Level Windows
: Close the windows for all subtrees and tables below the subtree in this window. (You can close windows automatically as new windows are opened, using the check box "Close this window when opening a subwindow." See "Subtree Windows That Show Subtrees," in this chapter.)

Close
: Close this window.

Quit
: Quit Browser (available only from the File menu in the Browser main window).

Browser Example

This section contains an example Browser session on a network that contains a Cisco® router with IP address `192.26.51.27`.

To begin this session, invoke Browser through *provision*.

The Browser main window is placed on the screen. To browse the MIB for the Cisco router, first fill in the entry fields in the Browser main window:

Node — Enter the Cisco router's IP address, `192.26.51.27`.

Community — Enter the community string your workstation is authorized to use. In this example, the string `public` is used.

Time-out interval — Use the default value, 5 seconds, for the time-out interval. If the Browser doesn't receive a reply from the Cisco SNMP agent in 5 seconds, it will try again.

Number of retries — Use the default number of retries, which is 3. This entry field specifies the number of times that Browser tries again if no reply is received from the Cisco agent within the time-out interval.

Figure A-24 shows the Browser main window after you've filled in the entry fields.

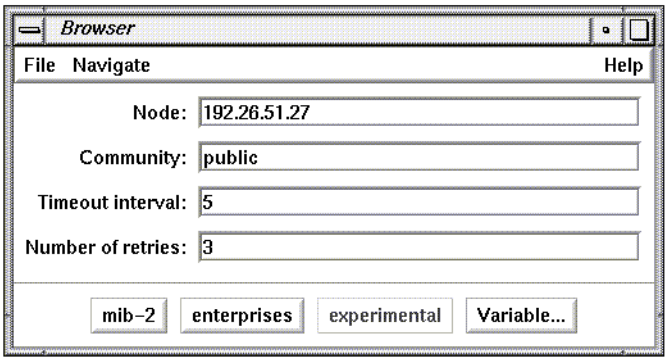

Figure A-24 Example Browser Main Window

Suppose you want to get the values for the `lsystem` group in the Cisco MIB. To display the variables in this group, use the Navigate rollover menus to navigate through the MIB hierarchy to `lsystem`, as shown in Figure A-25.

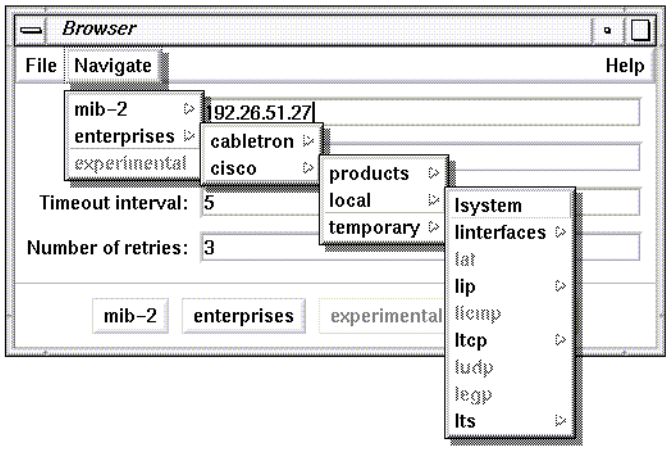

Figure A-25 Navigate Rollover Menus for cisco.local.lsystem

When you release the mouse button, the Cisco Subtree window shown in Figure A-26 appears:

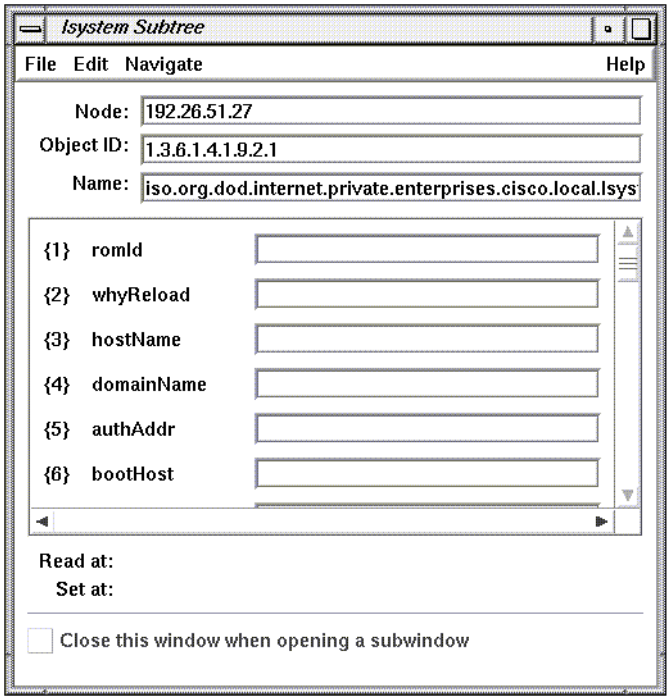

Figure A-26 Subtree Window for cisco.local.lsystem

To get the values for these variables, select Get from the Edit menu. The entry fields for the variables are filled in with their current values, and the current time is indicated at the bottom of the window. Figure A-27 shows an example:

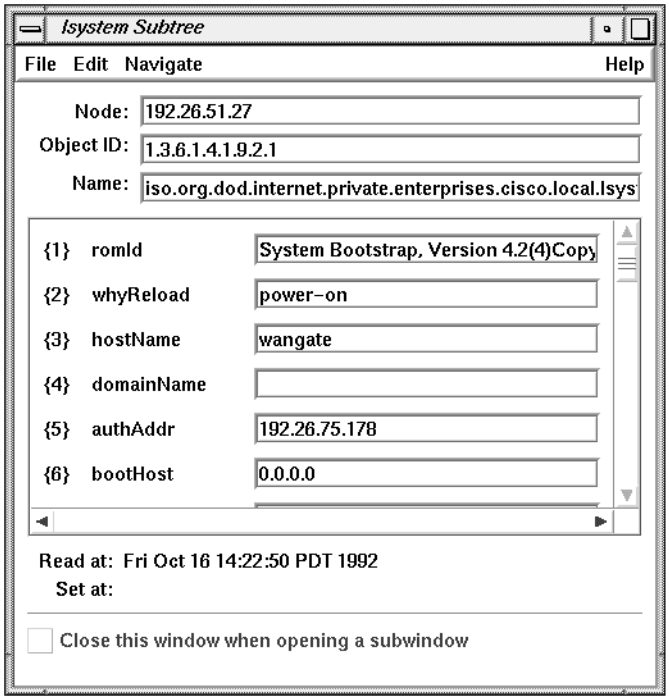

Figure A-27 Subtree Window With Values for cisco.local.lsystem

Use the right scroll bar to adjust the display area so that you can examine the values of the variables that don't fit in the default-size display area.

SNMP Management Glossary

This section describes some basic terms used in the SNMP management framework. It begins with basic concepts. Later definitions build on terms defined previously. Terms in *italics* are defined elsewhere in this section.

Agent

Software or firmware that gathers the information important to the device on which it resides. It also implements a protocol that exchanges that information with a *network management station. snmpd*(1M) is an example of an SNMP agent.

Network Management Protocol

The protocol used to convey management information between an *agent* and a *network management application*. The protocol used by Browser to query information from various agents is *SNMP*.

Simple Network Management Protocol (SNMP)

The *network management protocol* used by Browser to talk to *agents* on remote *managed nodes*. For Silicon Graphics workstations, the SNMP agent is *snmpd*(1M). Agents for other types of nodes may be implemented in software or firmware and are vendor-specific.

Management Information Base (MIB)

The specification for the virtual store of the information supported by an *agent*. Some MIBs have *RFC* status, which means they have been approved by the *IETF*.

MIBs are defined in *SMI* format. For instance, a router MIB is a collection of important information about a router defined in *SMI* format. An SNMP *agent* typically implements two MIBs, *MIB-II* and a device-specific MIB (an *Enterprise MIB*). However, the *agent* may not implement all of the objects defined in each MIB.

SNMP Containment Tree

A hierarchical tree of information gathered by agents about devices.

Subtree

A node with children in the *SNMP Containment Tree*.

MIB-II

The Internet-standard *MIB (RFC* 1213). This MIB has *managed objects* that are important for managing the TCP/IP suite of protocols.

Managed Object

A managed object is also known as a variable.

Variable

A leaf node in the *SNMP Containment Tree* is called a variable. In the *SMI* definition for each *MIB*, each variable is defined to be read-write, read-only, or write-only.

Object Identifier (Object ID)

An object identifier is a name in a dot notation that uniquely identifies an object (*subtree* or *managed object*) in the *MIB*. For example, the object ID for `sysContact` is `1.3.6.1.2.1.1.4`.

Enterprise MIBs

MIBs defined by different vendors for managing their devices. They can be specific to one device or vendor. For example, the CISCO *MIB* has objects for the Cisco router.

Community String

A password used by an *SNMP agent* for authentication purposes. The default string for Silicon Graphics workstations is “public.” Community strings are usually defined by system administrators.

Index

S

T

U

W

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-3273-001.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 415-965-0964
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389