HP OpenView Extensible SNMP Agent

Administrator's Guide

HP-UX and Solaris

Manufacturing Part Number: J1183-90003

March 2004

Legal Notices

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices.

Trademark Notices

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Windows NT® is a U.S. registered trademark of Microsoft Corporation.

Windows® and MS Windows® are U.S. registered trademarks of Microsoft Corporation.

Netscape™ and Netscape Navigator™ are U.S. trademarks of Netscape Communications Corporation.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

Oracle7™ is a trademark of Oracle Corporation, Redwood City, California.

Contents

Contents

6. Creating your MIB Module

7. Troubleshooting

Contents

1 Introduction and Operational Concepts

This chapter provides an introduction to the HP OpenView SNMP Extensible Agent. It includes the following:

- A guide to related documentation and references.
- A list of RFCs relating to SNMPv1, SNMPv2 and SNMPv3.
- A discussion of how to access and print manpages.
- Definitions and concepts that are key to understanding how the HP OpenView SNMP Extensible Agent operates, including discussions of:
  - — Manager and agent systems.
  - — MIBs.
  - — Traps.
  - — The Structure of Management Information (SMI) for SNMPv1 and SNMPv2.
  - — SNMP Agent architecture.
  - — Operational behavior of the HP OpenView SNMP Extensible Agent, and examples of how it can be used.

Documentation Guide and References

The manuals listed in this section provide system and network management information that supplements use of the HP OpenView Extensible SNMP Agent.

Network Management Manuals

Refer to the document set shipped with your HP OpenView product.

TCP/IP and SNMP Concepts

SNMPv1

Comer, Douglas. *Internetworking With TCP/IP: Principles, Protocols, and Architecture*. Englewood Cliffs, New Jersey: Prentice-Hall, 1988. To order, reference ISBN 0-13-470154-2.

Rose, Marshall T. *The Simple Book: Management of TCP/IP-based Internets.* Englewood Cliffs, New Jersey: Prentice-Hall, 1990. To order, reference ISBN 0-13-812611-9.

SNMPv2

If you are using HP OpenView Network Node Manager, these RFCs are supplied with the product. They are located under `$OV_DOC`.

Table 1-1 Outside Reading

Document	Description
RFC 1155: Structure and Identification of Management Information for TCP/IP-based Internets.	K. McCloghrie and M. T. Rose, (May 1990). Contains MIB object definitions. (Obsoletes RFC 1065).
RFC 1157: A Simple Network Management Protocol.	J. D. Case, M. Fedor, M. L. Schoffstall, and C. Davin, (May 1990). Defines SNMP. (Obsoletes RFC 1098).
RFC 1187: Bulk Table Retrieval with the SNMP.	K. McCloghrie, M. T. Rose, and C. Davin, (October 1990).
RFC 1212: Concise MIB Definitions.	K. McCloghrie and M. T. Rose, (March 1991). Describes the format for creating MIB object files.

Table 1-1 Outside Reading (Continued)

Document	Description
RFC 1213: Management Information Base Network Management of TCP/IP based internets: MIB-II.	K. McCloghrie and M. T. Rose, eds., (March 1991). Defines MIB-II. (Obsoletes *RFC 1158*; most current edition as of the printing of this guide.)
RFC 1215: Convention for Defining Traps for Use with the SNMP.	M. T. Rose, ed. (March 1991).
RFC 1901: Introduction to Community-based SNMPv2.	SNMPv2 WG, J.Case, K. McCloghrie, M.T. Rose, S. Waldbusser, (January 1996). Defines "Community-based SNMPv2." (Experimental. Obsoletes *RFC 1441*).
RFC 1902: Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2).	SNMPv2 WG, J.Case, K. McCloghrie, M.T. Rose, S. Waldbusser, (January 1996). MIB Definition language for Managed Objects (Draft Standard. Obsoletes *RFC 1442*).
RFC 1903: Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2).	SNMPv2 WG, J.Case, K. McCloghrie, M.T. Rose, S. Waldbusser, (January 1996). MIB Definition language for refined data types. (Draft Standard. Obsoletes *RFC 1443*).
RFC 1904: Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2).	SNMPv2 WG, J.Case, K. McCloghrie, M.T. Rose, S. Waldbusser, (January 1996). MIB Definition language for Conformance and Capability definitions. (Draft Standard. Obsoletes *RFC 1444*).
RFC 1905: Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2).	SNMPv2 WG, J.Case, K. McCloghrie, M.T. Rose, S. Waldbusser, (January 1996). Defines SNMPv2 protocol. (Draft Standard. Obsoletes *RFC 1448*).
RFC 1906: Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2).	SNMPv2 WG, J.Case, K. McCloghrie, M.T. Rose, S. Waldbusser, (January 1996). Defines SNMPv2 transport mappings for IP, IPX, DDP. (Draft Standard. Obsoletes *RFC 1449*).

Table 1-1 Outside Reading (Continued)

Document	Description
RFC 1907: Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2).	SNMPv2 WG, J.Case, K. McCloghrie, M.T. Rose, S. Waldbusser, (January 1996). Defines MIB objects that are mandatory for SNMP agents that support SNMPv2. (Draft Standard. Obsoletes *RFC 1450*).
RFC 1908: Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework	SNMPv2 WG, J.Case, K. McCloghrie, M.T. Rose, S. Waldbusser, (January 1996). Coexistence guidelines for SNMPv1 and SNMPv2. (Draft Standard. Obsoletes *RFC 1452*)

SNMPv3

The SNMPv3 Management Framework, as described in RFCs 2570, 2571, 2572, 2573, 2574, and 2575, addresses the deficiencies in SNMPv2 relating to security and administration. Coexistence issues relating to SNMPv1, SNMPv2, and SNMPv3 can be found in RFC 2576.

Table 1-2 Outside Reading

Document	Description
RFC 3410 (Informational): Introduction and Applicability Statements for Internet Standard Management Framework (December 2002).	R. Mundy D. Partain B. Stewart December 2002.
RFC 3414: User-based Security Model (USM) for version 3 of the Simple Network Management (SNMPv3).	U. Blumenthal B. Wijnen December 2002. Describes the User-based Security Model (USM) for SNMP version 3.
RFC 3415: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP).	B. Wijnen R. Presuhn K. McCloghrie December 2002. Describes the View-based Access Control Model (VACM) for use in the Simple Network Management Protocol (SNMP) architecture.

Table 1-2 Outside Reading (Continued)

Document	Description
RFC 2576: Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework.	R. Frye D. Levi S. Routhier B. Wijnen March 2000, describes coexistence between version 3 (SNMPv3), version 2 (SNMPv2), and the original Internet-standard Network Management Framework (SNMPv1).
RFC 2570: Introduction to Version 3 of the Internet-standard Network Management Framework.	J. Case R. Mundy D. Partain B. Stewart April 1999, provides an overview of the third version of the Internet-standard Management Framework, termed the SNMP version 3 Framework (SNMPv3).
RFC 2571: An Architecture for Describing SNMP Management Frameworks.	D. Harrington R. Presuhn B. Wijnen April 1999, describes an architecture for describing SNMP Management Frameworks.
RFC 2572: Message Processing and Dispatching for the SNMP.	J. Case D. Harrington R. Presuhn B. Wijnen April 1999, describes the Message Processing and Dispatching for SNMP messages within the SNMP architecture *[RFC2571]*
RFC 2573: SNMP Applications.	D. Levi P. Meyer B. Stewart April 1999 Describes five types of SNMP applications which make use of an SNMP engine as described in *[RFC2571]*.

General Operating System Knowledge

In addition to understanding networking concepts, it is critical that you have a good working knowledge of the operating system, or systems, that are present on your network. This book does not attempt to explain any concepts of the UNIX®[1] operating system (HP-UX, HP's UNIX operating system implementation or Solaris, Sun's UNIX operating system implementation). This book frequently refers to commands and functions of these operating systems, without further explanation. Refer to the documentation that came with your operating system for more information.

1. The term *UNIX* in this manual is referring to HP-UX and Solaris systems only.

Manpages

The manpages for system administration commands are contained in section (1M) on Solaris and HP-UX systems. Throughout this guide, manpages for system administration commands are denoted as *command* (1M); for example, *snmpd* (1M). The manpages for HP OpenView Extensible Agent are *snmpd.ea* and *snmpd.extend.*

Accessing Manpages

On HP-UX systems, the following procedure is just one suggestion for displaying or printing manpages. This procedure requires that you have manpages installed locally on your system. (If your network provides manpages remotely instead (for example, from a central server), then check with your system administrator about how to access them.)

1. Determine where on your system the manpage files are kept. Type

   `echo $MANPATH`

   You should see a list with one or more directories. Multiple directories will be separated by colons (for example, `/usr/local/man:/usr/man`). It is recommended that you check the contents of each directory to make sure it actually has manpage files in it.

2. Determine if the `MANPATH` list the directory `/opt/OV/man`. If the directory does not exist, add the directory to the `MANPATH` variable and export `MANPATH` variable.

   `export MANPATH=$MANPATH:/opt/OV/man`

Definitions and Concepts

This section describes definitions and concepts you need to configure and manage HP OpenView SNMP agents.

A manageable network consists of one or more manager systems or network management stations, and a collection of agent systems or network elements.

- A **manager system** executes network management operations that monitor and control agent systems. The implementation of these network management operations is called the **manager**.
- An **agent system** is a device, such as a host, gateway, terminal server, hub, or bridge, that has an **SNMP agent** responsible for performing the network management operations requested by the manager. The HP OpenView SNMP agent resides on a host running the UNIX operating system and facilitates the management of the host.

A typical manageable network may have one manager system and 1000 agent systems. A manager and agent may exist on the same system.

The Simple Network Management protocols communicate management information between a manager and an agent. SNMP enables the following:

- A manager to retrieve (**get**) management information from an agent. The manager sends requests for information to the agent, and the agent sends back replies containing the requested information.
- A manager to alter (**set**) management information on an agent.
- An agent to send information to the manager without an explicit request from the manager. Such an operation is called a **trap**. Traps alert the manager of changes that occur on the agent system, such as a reboot. The agent knows which manager systems to send traps to via a configurable trap destination table.

SNMP requests for information on an agent are accompanied by a **community name**; a community name is a password that allows the manager access to that information.

MIBs

The information on the agent is known as the Management Information Base (**MIB**). The MIB is not a physically distinct database; rather, it is a concept that encompasses configuration and status values normally available on the agent system. For example, MIB-II information that the HP OpenView SNMP agent supplies actually resides in the UNIX

kernel, not in a traditional database. MIB values conform to an Internet-standard structure of management information and compose a virtual data store on the agent system. Agents contain the “intelligence” required to access MIB values.

NOTE In this guide, MIB-II refers to standards; mib-2 refers to subtree organization.

MIBs are organized into **MIB modules**. A MIB module is described in a file that defines all the MIB objects under a subtree. The foundation module is the standards-based MIB-II module defined by *RFC1213: Management Information Base of Network Management of TCP/IP internets: MIB-II*. In addition to the Internet-standard MIB-II objects defined in *RFC 1213*, many hardware vendors, such as Hewlett-Packard, Cisco Systems, Wellfleet, and Novell (Excelan), have developed MIB extensions for their own products. In this guide, the MIBs defined by hardware vendors are referred to as enterprise-specific MIBs.

MIBs are defined using an Internet-standard language called the **Structure of Management Information**, or **SMI**. Your agent supports both the original SNMP version 1 SMI and the newer SNMP version 2 SMI. Key differences between the two SMI definitions are noted in Table 1-3.

Table 1-3 Comparison of SNMPv1 and SNMPv2 SMI

SNMPv1 SMI	SNMPv2c SMI
Forms the basis for all existing SNMPv1-based MIBS. Defined in *RFC 1155: Structure and Identification of Management Information for TCP/IP-based Internets.* Amended in *RFC 1212: Concise MIB Definitions*.	Defined as a superset of the SNMPv1 SMI. Extends the original SNMPv1 SMI to support these new data types: • `Counter64`--(64-bit unsigned integers)[a] • `Unsigned32`--(32-bit unsigned integers) • `Bits`--(allows the enumeration of bits) Allows custom data types to be built by constraining the range, size, or possible values of existing data types. For example, a new enumeration data type could be constructed by defining a small set of possible integer values. The SMI refers to this form of definition as a "Textual Convention." Defined in *RFC 1902: Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2).*

a. `extsubagt` does not support `counter64` and some SNMPv2 SMI constructs.

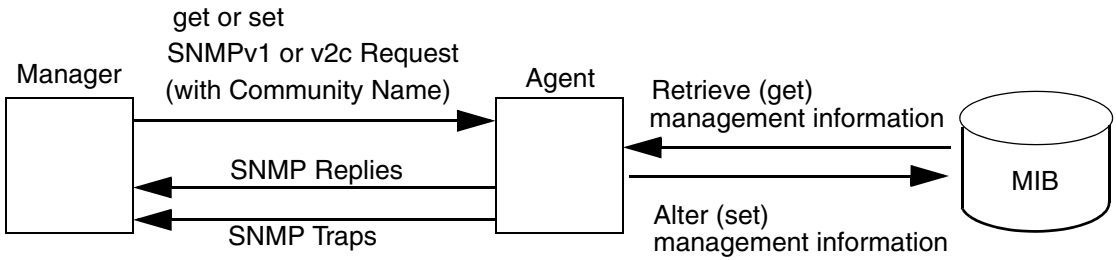
The MIB vendor is responsible for defining a MIB and describing its use.

See the `snmp_mibs` file and directory paths in Appendix B, "Platform Equivalents," for the exact location of the standard MIBs on your system.

How to Access the MIB

The manager accesses the agent's MIB using SNMP's get and set operations. Figure 1-1 shows a simplified diagram of the manager-agent interactions.

Figure 1-1 Manager-Agent Communication through SNMP

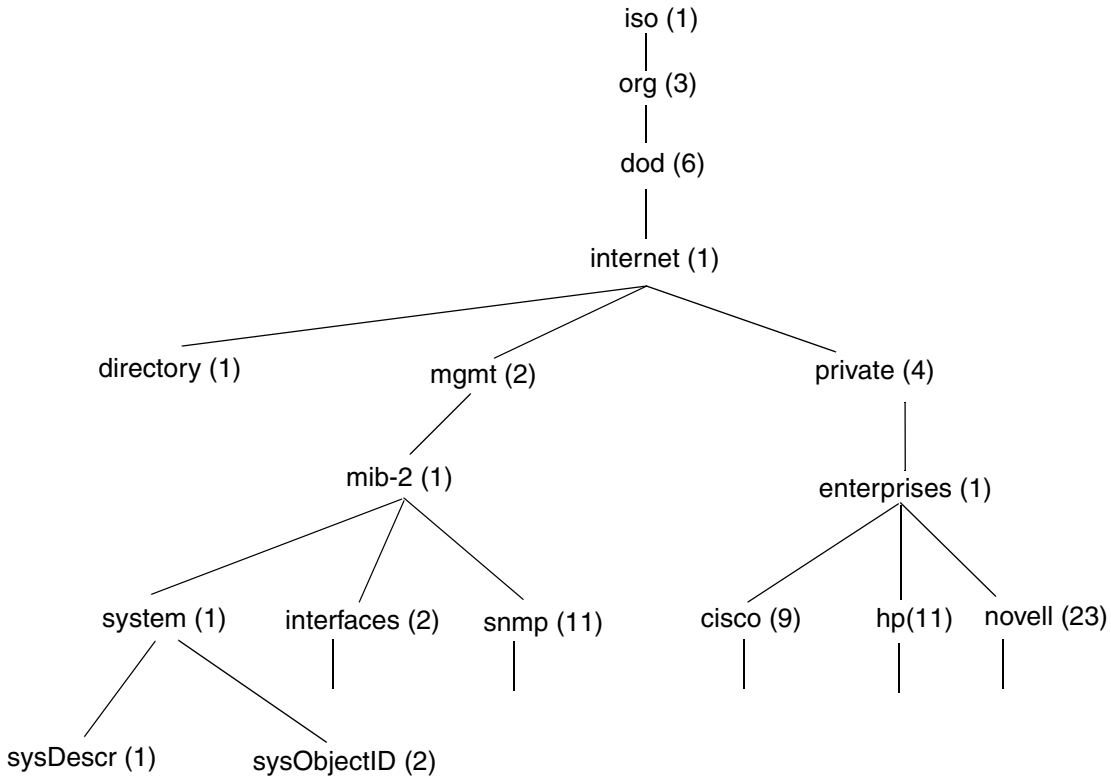


How MIBs are Organized

MIB objects are organized conceptually in a hierarchical tree structure. Figure 1-2 illustrates the tree structure. The numbers in parentheses are the numerical representation of the nodes. These numbers are called "object sub-identifiers."

Notice that SNMP messages contain only the object sub-IDs. The associated names are used for documentation purposes only.

Figure 1-2 Part of the Top of the MIB Naming Tree



Each branch of the tree consists of logical groupings used to generate unique object ID's. The branch is referred to as a **node**. A node can have both parents and children. A node that does not have children is referred to as a **leaf node**. The leaf node is the actual object. Only leaf nodes return MIB values from agents or have their MIB values altered. A **subtree** is used to refer to all nodes and children under a branch of the tree.

A MIB object is named by concatenating (linking together in a series) the numerical names of each node when traversing the MIB tree from `iso(1)` to the particular node. A full object ID name contains all the nodes, including the leaf nodes. The nodes are concatenated and separated by periods. For example, the `mib-2` subtree is `iso.org.dod.internet.mgmt.mib-2`, which is concisely written as `1.3.6.1.2.1`. This MIB object notation follows the standard notation defined in Abstract Syntax Notation One (ASN.1).

To avoid conflicts of object IDs, each branch of the tree must be registered, that is, defined through a designated organization. Enterprise- specific MIB's are registered under the `enterprises` subtree. The Internet- standard MIB-II is registered under the `mib-2` subtree. The `mib-2` subtree is primarily used to manage TCP/IP-based networks through SNMP.

Traps

A trap is a message sent from a remote system (as an agent) to a manager without an explicit request from the manager. Agents send traps to managers to indicate that an error has occurred or an event has taken place. Traps are also known as notifications.

The SNMP agent emits five types of notifications:

- SNMPv1 trap. This is the original type of notification defined in the original SNMP (version 1) protocol definition. The message is unconfirmed. In other words, the system that emits the trap never receives confirmation that the trap was received. The receiving system must support the SNMPv1 protocol.
- SNMPv2c trap. The trap serves the same function as for SNMPv1, but it has a slightly different format. The SNMPv2 trap is unconfirmed. The receiving system must support the SNMPv2c protocol.
- SNMPv2C inform. This message serves a similar purpose as a SNMPv2c trap, except that the message is confirmed. The sending system receives confirmation, via the protocol itself, that the message was indeed received. The receiving system must support the SNMPv2c protocol.
- SNMPv3 trap. SNMPv3 TRAPs use the engineID of the local application along with community name sending the trap rather than the engineID of the remote application. The receiving system must support SNMPv3 protocol. Refer *RFCs 2570 - 2575* for the detailed description about the terms used in SNMPv3 trap and inform.
- SNMPv3 inform. This uses the remote engineID along with community name when sending the message and the securityName and engineID must exist as a pair in the remote user table. The receiving system must support SNMPv3 protocol.

The format and style of a notification can vary, depending on several factors. For more information about notifications, see the *snmptrap* (1), *snmpv2trap* (1), or *snmpinform* (1) manpages.

Extensible SNMP Agent Architecture

The HP OpenView Network Node Manager and the Extensible SNMP Agent software each include a master Emanate Agent and two subagents:

- The HP-UNIX (`hp_unixagt`) subagent.
- The MIB-II (`mib2agt`) subagent.

The HP OpenView Extensible SNMP Agent product includes a third subagent: the Extensible Agent (`extsubagt`) subagent.

The relationships among the master agent and the three subagents in the Extensible Agent product are illustrated in Figure 1-3.

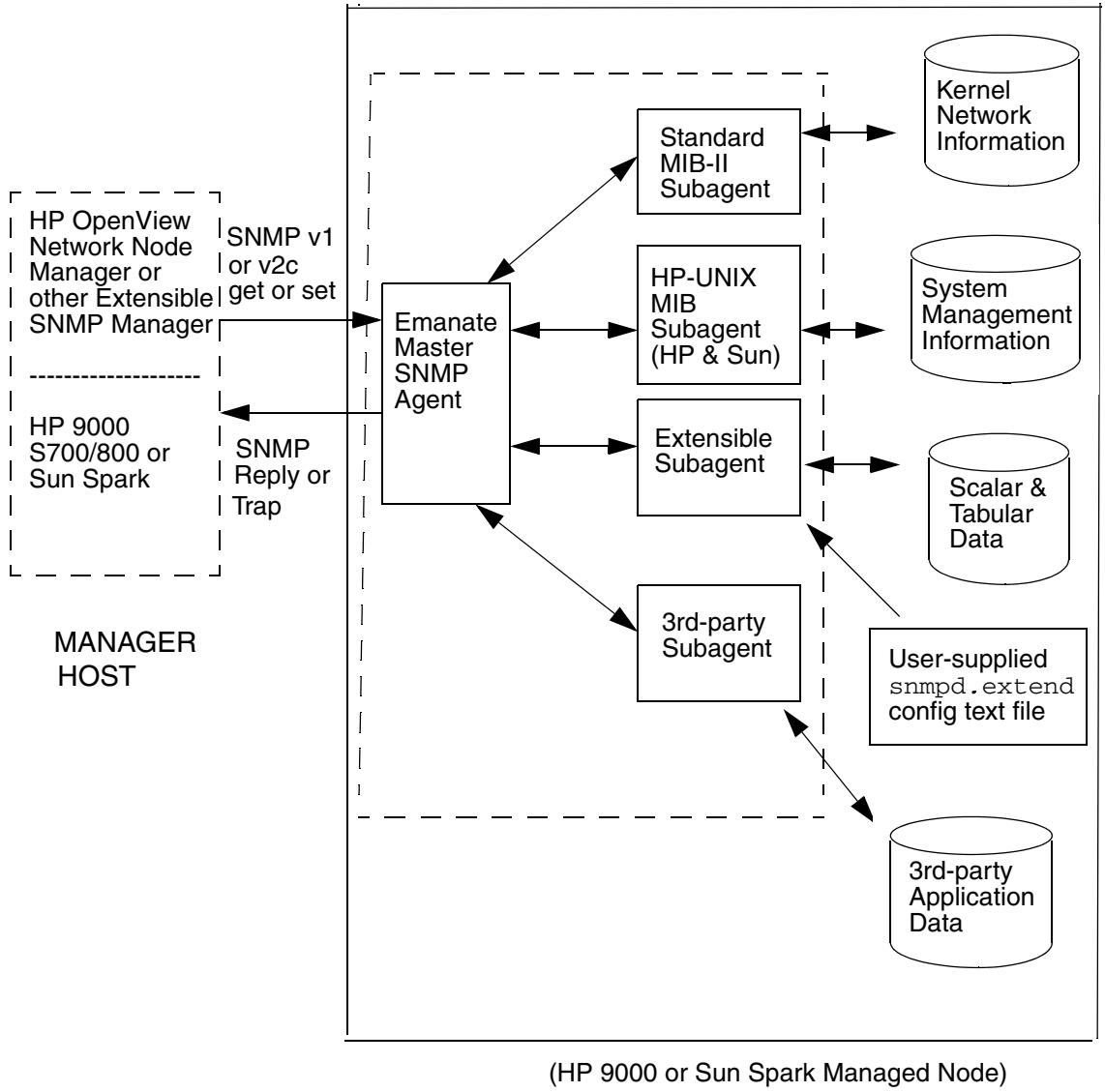
The master agent contains the SNMP communications stack, which facilitates communication with the management station and the SNMP agent.

The MIB-II subagent implements the MIB-2 standard, which contains networking statistics for the agent host. This instrumentation of the MIB-2 standard by the subagent facilitates management of your network.

The HP-UNIX subagent provides system information such as the number of UNIX processes currently running, or the percent of utilization of each file system. The instrumentation of system information by the subagent facilitates management of your HP-UX or Solaris system.

The third subagent (`extsubagt`), which is included only in the Extensible Agent product, enables you to rapidly develop an instrumentation to manage proprietary data without programming.

Figure 1-3 Extensible SNMP Agent Concepts



The HP OpenView Extensible SNMP Subagent

The **HP OpenView Extensible SNMP Agent** includes the Extensible Subagent (`extsubagt`). The Extensible Subagent extends the functionality provided by the master agent (`snmpdm`), the MIB-II subagent (`mib2agt`), and the HP-UNIX (`hp_unixagt`) subagent.

The HP OpenView Extensible SNMP subagent is used for the following:

- Adding your own objects to the agent.
- Configuring the agent to notify the manager when something is wrong on the agent.

You add your own objects by defining them in a new MIB subtree. This MIB subtree is contained in a new MIB module that extends the MIB that already exists on the agent.

For each object that you define, you must specify either a command that you want the agent to execute, or a file that you want the agent to read when the agent receives an SNMP request. The **extensible subagent** responds to an SNMP request by executing the specified command. When a manager sends an SNMP GetRequest or SetRequest, the agent executes the command associated with that object and returns the results of the command in the SNMP reply.

Using the extensible subagent, you can create a new MIB module that defines objects that will help you do the following:

- Use SNMP GetRequest to retrieve client data that is not available from your current standard agent through an SNMP GetRequest. For example, you can configure the extensible subagent to:
  - — List the users logged into a remote system.
  - — List the size of mail queues on a remote system.
  - — Check the status of the printers on your network.
  - — Monitor critical processes on a remote system.
- Execute an application or script on a remote system through an SNMP SetRequest. For example, in a manufacturing environment you could change the number of parts produced per hour.

You configure the system to notify the manager when something is wrong by executing the `snmptrap` command. The `snmptrap` command is included with the HP OpenView Extensible SNMP Agent. Use this command to define your own enterprise-specific trap and notify the

manager of events or conditions detected by applications. For example, you can configure the agent to notify the manager when a process on a remote system stops running. You can then use the set capability from the manager to restart the process.

By adding your own objects and configuring traps on the agent, the HP OpenView Extensible SNMP Agent becomes a tool that you can use to perform system management, peripheral management, and application management. Through SNMP, it enables you to extend your agent to get or set any MIB objects associated with a system, device, or application. The objects you manage do not have to be network devices. As long as the agent system can communicate with the device, the agent can be extended to manage the device.

Example Uses for the Extensible Subagent

The HP OpenView Extensible SNMP Agent can help you manage your network more proactively than before. The examples in this section use the **HP OpenView Network Node Manager** software as the management station.

NOTE While this guide uses the HP OpenView Network Node Manager to illustrate examples of how to apply the functionality of the HP OpenView Extensible SNMP Agent, you can use any SNMP management station to manage your Extensible Agents.

After adding objects to the agent, you can:

- Load the new MIB module (that is, the file defining the new objects) into the HP OpenView Network Node Manager MIB. Once you have loaded the new MIB module on the management station you can manage any of the MIB objects you have defined on the network.
- Get and set values of your new objects using a point-and-click MIB Browser.
- Without programming, build new MIB applications in a matter of minutes for your new objects. Once you have built your MIB applications, you can monitor the objects through the HP OpenView Network Node Manager menu bar.
- Collect historical MIB information about your new object and display collected data.
- Define event thresholds for the new objects. This helps you, for example, to find out when a printer goes down.
- Define actions to be taken upon receipt of an SNMP trap coming from a system running the extensible subagent.
- Manage critical software processes that are running on unattended systems in dispersed locations.

2 Before You Install

This chapter provides preinstallation information for the HP OpenView Extensible SNMP Agent. It includes:

- A brief description of the role of the HP Software Distributor (SD) in the installation of the agent.
- Hardware and software requirements for installation.
- Descriptions of the files and processes involved both when the Extensible Agent is invoked and when it is operational.
- A description of how to manually stop and restart the agent software.

Installation Mechanism

This release of the HP OpenView Extensible SNMP Agent uses HP Software Distributor (SD) for its installation mechanism. Several components of SD are included on the CD-ROM that you purchased. They are not installed on your system and will not affect any current installation of SD on your system.

You use a command line interface when installing the Extensible Agent. If you know that you already have SD installed on your system and are familiar with its use, you may use its graphical user interface instead of the command line instructions provided in this guide.

If you choose to use the SD graphical user interface, be sure to complete the preinstallation steps in this chapter first.

Hardware and Software Prerequisites

You need the following hardware and software to run the HP OpenView SNMP Extensible Agent. Consult the product data sheets for additional supported software and hardware.

Hardware

You need one of the following workstations:

- HP 9000 Series 700 or 800
- Sun SPARCstation

and

- a CD-ROM drive

Software

You need to be running one of the following operating systems:

- HP-UX version 11.X
- Solaris 2.8 or later

Disk Space Requirements

At a minimum, you need the disk space listed below to install the HP OpenView Extensible SNMP Agent product. You will need additional space for your databases and application files.

HP-UX 11.X	10 Mbytes
Solaris 2.8 or later	10 Mbytes

On all platforms, most of this disk space should be in `/usr` and `/lib`.

To provide enough disk space on your workstation:

1. Log in as `root`.
2. To check your disk space, issue the following command:

HP-UX	`bdf`
Solaris	`df`

   If the directory

- Has sufficient free disk space, proceed to Chapter 3, “Installing the HP OpenView Extensible SNMP Agent.”
- Does not have sufficient free disk space, continue with step 3.

3. Locate a file system that has sufficient free disk space, or mount a dedicated volume under the appropriate directory.

Processes and Files

This section defines the processes and files used during invocation and operation of the HP OpenView SNMP Extensible Agent. It includes explanations of the interactions and relationships among these processes and files. Note that the interactions among processes and files at invocation time and during regular operation are different.

Invocation Behavior

The process that runs as the HP OpenView SNMP agent is platform dependent. Network Node Manager (NNM) contains the Emanate SNMP agent for installation on Solaris and HP-UX.

On invocation, the SNMP agent reads the `snmpd.conf` file to obtain its configuration. This file contains the agent's trap destinations, community names, and certain MIB values.

The interactions and relationships among these processes and files at invocation time is as follows:

Table 2-1

HP-UX 11.X and later		
`/sbin/rc`	invokes	`/sbin/rc2.d/S560SnmpMaster`
`/sbin/rc2.d/S560SnmpMaster`	invokes	`/sbin/init.d/SnmpMaster`
`/sbin/init.d/SnmpMaster`	invokes	`/usr/sbin/snmpdm`
`/usr/sbin/snmpdm`	reads	`/etc/SnmpAgent.d/snmpd.conf`
`/sbin/rc`	invokes	`/sbin/rc2.d/S565SnmpMib2`
`/sbin/rc2.d/S565SnmpMib2`	invokes	`/sbin/init.d/SnmpMib2`
`/sbin/init.d/SnmpMib2`	invokes	`/usr/sbin/mib2agt`
`/sbin/rc`	invokes	`/sbin/rc2.d/S565SnmpHpunix`
/sbin/rc2.d/S565SnmpHpunix	invokes	`/sbin/init.d/SnmpHpunix`

Table 2-1 (Continued)

HP-UX 11.X and later		
/sbin/init.d/SnmpHpunix	invokes	/usr/sbin/hp_unixagt
/sbin/rc	invokes	/sbin/rc2.d/S566SnmpExtAgt
/sbin/rc2.d/S566SnmpExtAgt	invokes	/sbin/init.d/SnmpExtAgt
/sbin/init.d/SnmpExtAgt	invokes	/usr/sbin/extsubagt
/usr/sbin/extsubagt	reads	/etc/SnmpAgent.d/snmpd.extend

Table 2-2

Solaris		
/etc/rc2	invokes	/etc/rc2.d/S96SnmpMaster
/etc/rc2.d/S96SnmpMaster	invokes	/sbin/init.d/SnmpMaster
/sbin/init.d/SnmpMaster	invokes	/usr/sbin/snmpdm
/usr/sbin/snmpdm	reads	/etc/SnmpAgent.d/snmpd.conf
/etc/rc2	invokes	/etc/rc2.d/S97SnmpMib2
/etc/rc2.d/S97SnmpMib2	invokes	/sbin/init.d/SnmpMib2
/sbin/init.d/SnmpMib2	invokes	/usr/sbin/mib2agt
/etc/rc2	invokes	/etc/rc2.d/S97SnmpHpunix
/etc/rc2.d/S97SnmpHpunix	invokes	/sbin/init.d/SnmpHpunix
/sbin/init.d/SnmpHpunix	invokes	/usr/sbin/hp_unixagt
/etc/rc2	invokes	/etc/rc2.d/S98SnmpExtAgt

Table 2-2 (Continued)

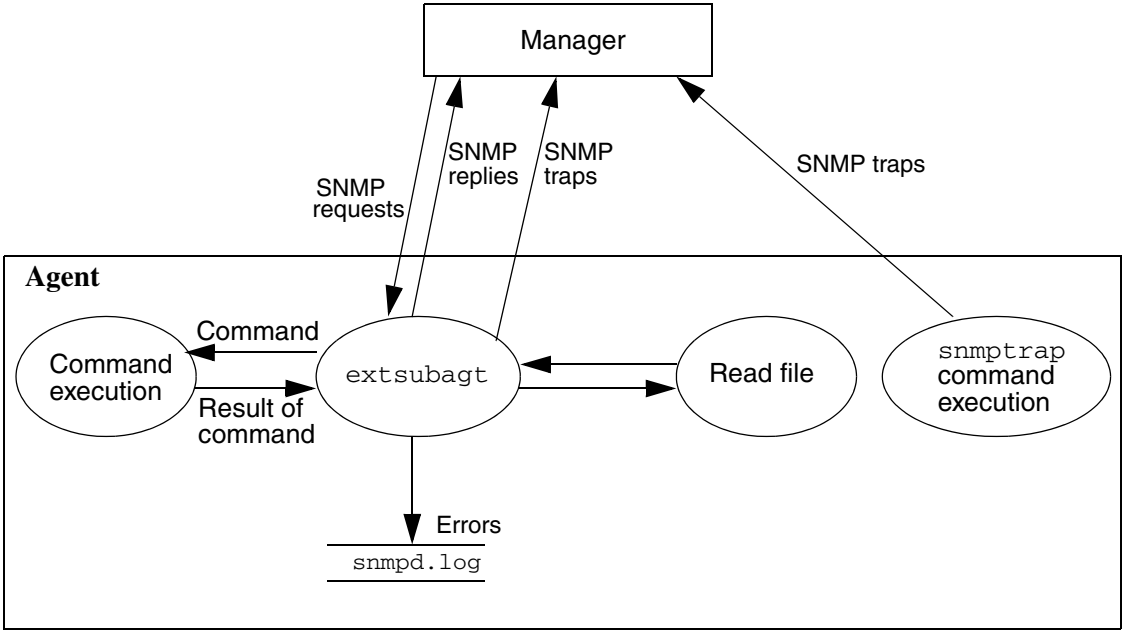
Solaris		
/etc/rc2.d/S98SnmpExtAgt	invokes	/sbin/init.d/SnmpExtAgt
/sbin/init.d/SnmpExtAgt	invokes	/usr/sbin/extsubagt
/usr/sbin/extsubagt	reads	/etc/SnmpAgent.d/snmpd.extend

Operational Behavior

After the Extensible SNMP Agent is operational, the agent's background process continues to run. When the manager sends an SNMP request to the agent, the Extensible SNMP Agent processes the request, takes the appropriate action, such as executing a command or reading a file, and sends an SNMP reply to the manager.

SNMP agents send standard SNMP traps such as cold start, warm start, link down, link up, and authentication failure traps to the manager. Optionally, you can configure the HP OpenView Extensible SNMP Agent to execute `snmptrap` commands. The `snmptrap` command also sends traps to the manager. Extensible SNMP Agent errors are logged to the `snmpd.log` file. Figure 2-1 shows the operational interactions and relationships among these processes and files.

Figure 2-1 Agent Processes and Files during Operation



Manually Stopping and Restarting the Agent Software

If you want to stop the agent software, you must kill the SNMP agent process. Use these commands:

```
ps -ef | grep snmpdm
kill -9 <snmpdm's process number>
kill -9 <all subagent process numbers of interest>
```

NOTE All shared library agents die when the master agent is killed. All separate process subagents receive an event when the master agent dies; they may or may not die. The subagents delivered with this product are configured to die by default when the master agent dies.

To restart the HP OpenView SNMP Agent software, you must be root. The agent starts automatically the next time your workstation is rebooted. You can start it manually by executing the following startup script as root:

```
snmpd
```

Table 2-3 lists the executable command for each agent or subagent.

Table 2-3 Extensible SNMP Agent Executables

Agent or Subagent	Executable
Master agent	/usr/sbin/snmpdm
Mib2 subagent	/usr/sbin/mib2agt
HP UNIX subagent	/usr/sbin/hp_unixagt
Extensible subagent	/usr/sbin/extsubagt

3 Installing the HP OpenView Extensible SNMP Agent

This chapter describes how to install the HP OpenView Extensible SNMP Agent. Three types of installation are described:

- Installing on a system that does not have any HP OpenView software installed.
- Installing on an NFS diskless system.
- Installing remotely. Use this procedure if you do not have a local CD-ROM drive to use with the installation media that came with your product.

NOTE Remote installation is not supported on NFS diskless systems.

Before you begin installing, check the hardware and software prerequisites listed in Chapter 2, “Before You Install.”

Installing on a System with No Other HP OpenView Software Installed

1. Log in as `root` to the workstation on which you want to install the HP OpenView Extensible SNMP Agent.
2. Insert the HP OpenView Extensible SNMP Agent CD-ROM into the CD-ROM drive.
3. Determine the device filename of your CD-ROM drive.

NOTE Be sure that you choose a block drive device and not a raw drive device.

4. Mount the CD-ROM drive by typing the appropriate command, specifying the appropriate drive device filename and the directory to which you want to mount the file system. The directory can be any existing directory. For example:

   On an HP-UX 11.X system the command syntax is:

   `/etc/mount` *`device_filename directory_name`*

   So you would type something similar to:

   `/etc/mount /dev/dsk/c201d3s0 /cdrom`

   The volume management performed by Solaris systems causes the CD-ROM to be mounted automatically.

5. Change to the directory where you mounted the CD-ROM file system. For example:

   `cd /cdrom`

6. Use the following command to install HP OpenView Extensible SNMP Agent.
   - On HP-UX 11.00/11.11 PA_RISC system
     `./HPUX11.X_PA/install`
   - On HP-UX 11.23 IA system
     `./HPUX11.X_IA/install`
   - On Solaris system
     `./install`

7. Answer the questions that appear on the screen. The questions are the following:

   - `Do you want to install the manpages?`

     Enter **`Y`** or **`N`**.

     If you choose not to install the manpages now, may install them at a later time.

   - `Do you want to continue with the installation?`

     A list appears showing what will be installed.

     Enter **`Y`** to continue or **`N`** to cancel the installation.

   The installation begins. The installation program goes through two phases, Analysis and Execution. Messages reflecting which stage the installation procedure is in are written to the screen. When installation is complete you will see a status message. It will begin with one of the following:

   - `Your installation was successful`

     If you receive this message, go to the next step.

   - `Errors have been encountered in your installation.`

     Go to the section "If Errors Occurred."

8. To verify that your installation was successful, check the end of the log file `/var/adm/sw/swagent.log`. Messages are appended to this file. Therefore the status messages for this installation will be the final entries in the file. Each set of status messages includes the date and the time the installation occurred. Look for the date and time that corresponds to the most recent installation.

   If the installation program completed without fatal errors, each fileset that was installed should have the word `Configured` next to it. Even though you may see the status message `ERROR`, if `Configured` appears next to each fileset, the installation was successful.

   You may also want to scan the `NOTE` and `WARNING` sections to see if there are any other steps you might need to take.

If Errors Occurred

If errors occurred during installation, look at the end of the log file `/var/adm/sw/swagent.log`. Each set of status messages includes the date and the time the installation occurred. Look for the date and time that corresponds to the most recent installation.

There are several kinds of errors that could have occurred. Some filesets may have been installed but not configured. These filesets are noted in the log file with the word `ERROR` instead of the word `Configured` next to them.

Installing on an NFS Diskless Cluster

This section describes how to install the HP OpenView Extensible SNMP Agent on an NFS diskless cluster. It also includes procedures for:

- Starting the Extensible Agent on an NFS diskless system.
- Removing an NFS diskless cluster.

HP OpenView Extensible SNMP Agent supports installation of NFS diskless clusters on HP Series 700/800 systems with HP-UX 11.X. Remote installation of NFS diskless clusters is not supported.

Before installing on an NFS diskless cluster, you must configure the diskless cluster according to the instructions in your operating system manual.

NOTE Before you begin this procedure, please consult the *Release Notes* that were shipped with your product for any possible modifications to these steps.

Procedure

After the diskless cluster is configured, follow these steps on the NFS diskless server:

1. Log in to the NFS diskless server as `root`.
2. Insert the CD-ROM for HP OpenView Extensible SNMP Agent into the CD-ROM drive.
3. Determine the device filename of your CD-ROM drive.
4. Mount the CD-ROM drive by typing the appropriate command, specifying the appropriate drive device filename and the directory to which you want to mount the file system. The directory can be any existing directory.

   On an HP-UX system the command syntax is:

mount /dev/rmt *device_filename directory_name*

   So you would type something similar to:

   mount /dev/rmt/0m /cdrom

5. Start HP SAM. This is usually done by typing `sam &`.
6. Double-click on the `Software Management` icon.
7. Double-click on the `Install Software to Cluster` icon.

8. In the `Select Alternate Root Path` dialog box, use the default that is in the entry field. It should look similar to:

   `/export/shared_roots/OS_700 -> /`

9. In the `Specify Source` dialog box, specify the following:

   In the `Source Host Name` field, enter the name of the host where the software depot is located. This should be the local server hostname

   In the `Source Depot Path` field, enter the path to CD-ROM file system and the depot name. For example:

   `/cdrom/OVDEPOT`

   In the `Change Software View` field, select the default of `All Bundles`.

10. In the `Software Selection` dialog box, highlight the bundles that you want to install. You can select more than one at a time by holding down the **CTRL** key while clicking on the bundle names with the mouse.

11. When you have selected all the bundles you want to install, select `Action:Mark for Install`.

    You may receive an error message stating that difficulties were encountered while marking some items that your selections depends on. The installation procedure has a dependency on some filesets that should already be installed on your system as part of the HP-UX 11.X operating system.

    Click `[OK]` to close this error dialog box. The installation continues. Later in the installation process, your system will be checked again.

12. After marking the items for installation, select `Action:Install`.The installation process begins.

13. When the Analysis phase of installation is complete, click `[OK]` to continue the installation.

14. You will be asked if you want to continue the installation. Click `[Yes]` to continue or `[No]` to cancel.

    The installation dialog box closes, and installation messages are written to the screen of the window from which you started HP SAM. The software will be installed and configured on the server, and configured on each node in the cluster.

15. To verify the installation, look at the end of the log file `/var/adm/sw/swagent.log`. Messages are appended to this file. Therefore, go to the end of the file to see the status messages for this installation. Each set of status messages includes the date and the time the installation occurred. Look for the date and time that corresponds to the most recent installation.

    If the installation program completed without errors, each fileset that was installed should have the word `Configured` next to it. Even though you may see the status message `ERROR`, if `Configured` appears next to each fileset, the installation was successful.You may also want to scan the `NOTE` and `WARNING` sections to see if there are any other steps you might need to take.

If Errors Occurred

If errors occurred during installation, look at the end of the log file `/var/adm/sw/swagent.log`. Each set of status messages includes the date and the time the installation occurred. Look for the date and time that corresponds to the most recent installation.

Several kinds of errors may have occurred. Some filesets may have been installed but not configured. These are noted in the log file with the word `ERROR` instead of the word `Configured` next to them.

After fixing the errors, reinstall the product.

Starting the Extensible Agent on an NFS Diskless System

To start the Extensible Agent on a diskless system, follow these instructions:

1. On the NFS diskless server *only*, run **ovstart**.
2. On the NFS diskless server *only*, run **ovstatus**.

   If any of the processes listed reports an error or a status of `NOT_RUNNING`, refer to the troubleshooting chapter of the *Managing Your Network with HP OpenView Network Node Manager* book.
3. On either the NFS diskless server or the NFS diskless client, run **ovw**.

Removing an NFS Diskless Cluster

1. Log in to the NFS server as `root`.
2. Start HP SAM. This is usually done by typing `sam &`.
3. Double-click on the `Software Management` icon.

4. Double-click on the `Remove Cluster Software` icon.
5. In the `Select Alternate Root Path` dialog box, click on `[OK]` to use the default that is in the entry field. It should look similar to:

   ```
   /export/shared_roots/OS_700 -> /
   ```

6. In the `SD Remove-Software Selection` dialog box, highlight the bundles that you want to remove. To select more than one at a time, hold down the **CTRL** key while clicking on the bundle names with the mouse.
7. When you have selected all the bundles you want to remove, select `Action:Mark for Remove`.
8. After marking the items for installation, select `Action:Remove`. The removal process begins.
9. When the Analysis phase of removal is complete, click `[OK]` to continue the removal.
10. You will be asked if you want to continue the removal process. Click `[Yes]` to continue or `[No]` to cancel.

Installing from a Remote CD-ROM

If you do not have a local CD-ROM drive to use with the installation media that came with your product, you may install the HP OpenView Extensible SNMP Agent from a remote drive.

This section contains the procedure for installing the Extensible SNMP Agent product remotely.

NOTE Remote installation is not supported on NFS diskless systems.

The workstation with the CD-ROM drive (source workstation) can be an HP-UX or Solaris system. The workstation where you want to install HP OpenView Extensible SNMP Agent (target workstation) can be any of the supported workstations mentioned in Chapter 2, “Before You Install.”

On the Source Workstation

On the workstation where the CD-ROM drive is located:

1. Log in as `root`.
2. Insert the HP OpenView Network Node Manager CD-ROM into the CD-ROM drive.
3. Determine the device filename of your CD-ROM drive.

NOTE Be sure that you choose a block drive device and not a raw drive device.

4. Mount the CD-ROM drive by typing the appropriate command, specifying the appropriate drive device filename and the directory to which you want to mount the file system. The directory can be any existing directory. For example:

   On an HP-UX system the command syntax is:

`/etc/mount` *`device_filename directory_name`*

   So you would type something similar to:

   `/etc/mount /dev/dsk/c201d3s0 /cdrom`

5. Export the CD-ROM file system so that the target workstation can NFS mount it.

   In these examples, `marvin` is the name of the target workstation and `marion` is the name of the source workstation where the CD-ROM is physically mounted.

On a HP-UX 11.X Source Workstation

a. Add the following line to the file `/etc/exports`:

```
/cdrom -ro,root=marvin
```

b. Export the file system with the following command:

HP-UX 11.X `/usr/sbin/exportfs -a`

On a Solaris 2.X Source Workstation

a. Add the following line to the file `/etc/dfs/dfstab`:

```
share -F nfs -o ro,root=marvin
```

b. Check to see if the NFS daemon `nfsd` is running. For example:

```
ps -ef | grep nfsd
```

c. If the `nfsd` daemon is running, execute:

```
/usr/sbin/shareall
```

d. If the `nfsd` daemon is not running, execute:

```
/etc/init.d/nfs.server start
```

On the Target Workstation

On the workstation where you want to install the HP OpenView Extensible SNMP Agent product:

1. NFS mount the CD-ROM file system (at `/cdrom`, for example). Type:

```
mkdir /cdrom

mount marion:/cdrom /cdrom
```

2. Change to the directory where you mounted the CD-ROM file system. For example:

```
cd /cdrom
```

3. Install the product with the following command:
   - On HP-UX 11.00/11.11 PA_RISC system
     `./HPUX11.X_PA/install`
   - On HP-UX 11.23 IA system
     `./HPUX11.X_IA/install`
   - On Solaris system
     `./install`

4. To verify your installation, check the end of the log file /var/adm/sw/swagent.log. Messages are appended to this log file. Therefore the status messages for this installation will be the final entries in the file. Each set of status messages includes the date and the time the installation occurred. Look for the date and time that corresponds to the most recent installation.

   If the installation program completed without errors, each fileset that was installed should have the word `Configured` next to it. Even though you may see the status message `ERROR`, if `Configured` appears next to each fileset, the installation was successful.

   You may also want to scan the `NOTE` and `WARNING` sections to see if there are any other steps you might need to take.

If Errors Occurred

If errors occurred during installation, look at the end of the log file `/var/adm/sw/swagent.log` on the target workstation. Each set of status messages includes the date and the time the installation occurred. Look for the date and time that corresponds to the most recent installation.

Several kinds of errors may have occurred. Some filesets may have been installed but not configured. These filesets are noted in the log file with the word `ERROR` instead of the word `Configured` next to them.

After fixing the errors, reinstall the product.

Post-Installation Steps

After installation has completed, you can clean up as follows. Before you start these steps, be sure that you are not in (or beneath) the mounted directory on the either target or the source workstation:

On the source workstation:

1. Remove the line you added to `/etc/exports` on HP-UX 11.X and to `/etc/dfs/dfstab` on Solaris 2.X.
2. Unexport the directory. Type:

   HP-UX 11.X **`/usr/sbin/exportfs -u /cdrom`**

   Solaris 2.X **`unshare /cdrom`**
3. Unmount the CD-ROM drive. Type:

   `umount /cdrom`

On the target workstation, complete this step:

1. Unmount the CD-ROM drive. Type:

   ```
   umount /cdrom
   ```

4 Configuring the Master SNMP Agent

The information describing the configuration of the HP OpenView Extensible SNMP Agent has been divided into two chapters. This chapter describes the configuration of three *optional* values in the Master portion of the Extensible SNMP Agent. These values are:

- System contact and location.
- Community name.
- Trap destinations.

If you do not wish to configure any of these values, you may proceed to Chapter 5, "Configuring the HP OpenView Extensible SNMP Subagent."

Configuration information for an SNMP agent can exist in the `snmpd.conf` configuration file, on the agent command line, or in both places.

When configuring the agent on the command line, the command line options should be placed in the appropriate startup script or startup configuration file for the particular platform. The configuration files for the startup scripts are as follows:

Master	`/etc/rc.config.d/SnmpMaster`
MIB2SubAgt	`/etc/rc.config.d/SnmpMib2`
HPUnix	`/etc/rc.config.d/SnmpHpunix`
ExtSubAgt	`/etc/rc.config.d/SnmpExtAgt`

Most of the platform dependency complexity can be avoided by configuring `snmpd.conf` when possible. The procedure is described on the next page.

System Contact and Location

The **system contact** consists of the name of the system's administrative contact and information on how to contact this person. The **system location** is a description of the physical location of the system.

The agent software operates correctly without any configuration. Optionally, you may want to set the agent's system contact and system location so that the manager can request these values remotely.

The system and SNMP MIB2 groups are implemented by the master agent rather than the MIB2 subagent. This is because the SNMP protocol stack, which contains the SNMP group statistics, resides in the Master Agent. When the Master Agent reads `snmpd.conf`, it configures the stack and the system group.

Configuring System Contact and Location

You can set the system contact and system location for the HP OpenView SNMP agent software in one of two ways:

- The first way is the recommended method. Set the system contact and system location by editing the `snmpd.conf` file. Use the following procedure:

  1. At the end of the `snmpd.conf` file, find these two lines.

     ```
     location:#enter location of agent
     contact:#enter contact person and how to contact this person
     ```

  2. Delete the comments (preceded by the # sign).
  3. After the `location:` label, add the system's physical location. The maximum length of the ASCII system location string is 256 characters.
  4. After the `contact:` label, add the name of the system's administrative contact and information on how to contact that person. The maximum length of the ASCII system contact string is 256 characters.

     EXAMPLE:

     ```
     location: 1st floor, south of post P2
     contact: Bob Jones (Phone: Ext. 2815, Mail: jones@host2)
     ```

- The second method is to use the command line.

  1. Add the following command line option to the appropriate startup script configuration file:

```
-C "contact" -L "location"
```

2. Reconfigure the SNMP agent using the `-C` or `-L` options. For example, on HP-UX 10.01 and later or Solaris, edit `/etc/rc.config.d/SnmpMaster` and add

```
SNMP_MASTER_OPTIONS="-c contact -l location"
```

NOTE The system contact and system location values set through the command line take precedence over the system contact and system location values in the `snmpd.conf` file. That is, if the values are set both ways, the system uses the values specified in the command-line invocation instead of those specified in the `snmpd.conf` file.

Community Name

A **community name** is a password that enables SNMPv1 and SNMPv2c access to MIB values on an agent. Community names are not highly secure; they go unencrypted across the network. Currently accessible MIB values are not normally considered sensitive information.

The HP OpenView SNMP agent's community name implementation has the following characteristics:

- You configure the community name in the agent's `snmpd.conf` file.
- The community name is associated with all of the agent's MIB values, not with subsets of MIB values.
- You can enter multiple community names for GetRequests and SetRequests.

GetRequests

By default, the agent only responds to configured community names.

- If it is not present at installation time, the community string, `public`, is added to the `snmpd.conf` file.
- If you configure a different community name, the SNMP agent responds only to that community name.
- If no community name is configured, the agent does not respond.

You can configure the agent to respond to more than one get-community- name. The associated line from the file is

```
get-community-name:#enter community name
```

Authentication Failure

An **authentication failure** results when a manager system sends an incorrect or an invalid community name to an agent. When an agent receives an invalid community name, it can send an authentication failure trap to the manager system.

By default, the HP OpenView SNMP agent sends authentication failure traps. To prevent the sending of authentication failure traps, start the agent with the `-a` option.

Configuring an Agent's Community Name

If you change the default community name configuration, HP recommends that you use the same community name on all agents. To enter an HP OpenView SNMP agent's community name, edit the `snmpd.conf` file on the agent system.

NOTE No duplicate community strings are allowed in the `snmpd.conf` file. Entering one results in an error message in the log file. Also, the `set` community string is valid for both `get` and `set` operations. There is no need to configure a `get` community string separately.

1. Near the end of the file, find the line:

   ```
   get-community-name:#enter community name
   ```

2. Delete the comment (it is preceded by the # sign).
3. After the `get-community-name:` label, add the agent's community name. For example:

   ```
   get-community-name: private
   ```

   ```
   get-community-name: mark
   ```

4. Restart the agent.

NOTE Authentication failure traps can be inhibited by starting the agent with the `-a` command line option (see the manpage for *snmpd* (1M)).

SetRequests

By default, the agent does not respond to SetRequests. To enable managers to set MIB values you must configure the agent to respond to SNMP SetRequests. To do so, enter a community name in the `snmpd.conf` file. You can configure the agent to respond to more than one set-community- name. The associated line from the file is

```
set-community-name:#enter community name
```

Managers can only set MIB values using the set-community-name entered.

Manager-Agent Community Name Relationship

To learn about the interaction between manager and agent, see your manager documentation. If you are using Network Node Manager as your manager, see the *snmpconf* (4) manpage.

Trap Destinations

Trap destinations exist on agents to tell the agents where to send their SNMP traps. A **trap destination** identifies a manager system that will receive the agent's traps. An agent may have multiple trap destinations if multiple managers are managing the agent.

Configuring Trap Destinations

If you want an HP OpenView SNMP agent to send traps to a manager system, you must manually set the agent's trap destination. To do so, edit the `snmpd.conf` file on the agent system as follows:

1. Near the end of the file, find the line:

   ```
   trap-dest:  #enter the IP Address of the system where traps will be sent
   ```

2. Delete the comment (preceded by the `"#"` sign).
3. After the `trap-dest:` label, add the host name or IP address of the management system that you want the agent to send its traps to. For example:

   ```
   trap-dest: 15.2.113.223
   ```

4. To specify more trap destinations, add more `trap-dest:` lines to the file.
5. Restart the agent.

5 Configuring the HP OpenView Extensible SNMP Subagent

This chapter describes how to configure the HP OpenView Extensible SNMP Subagent. It includes procedures for:

- Configuring the HP OpenView Extensible SNMP Subagent to support new objects.
- Configuring HP OpenView Extensible SNMP Subagents to execute the `snmptrap` command.

To configure the optional values of system contact and location, community name, and trap destinations, see Chapter 4, “Configuring the Master SNMP Agent.”

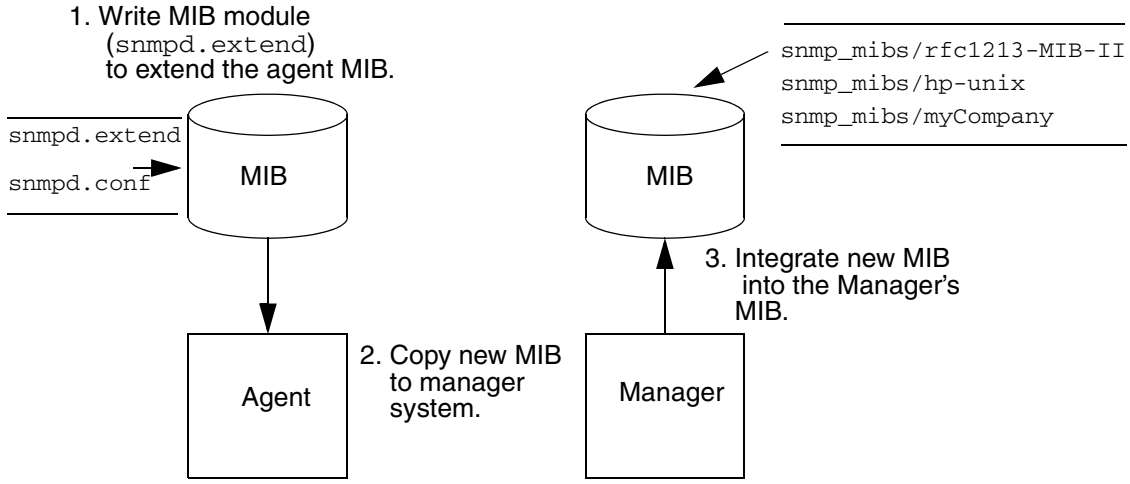
Configuring Extensible SNMP Agents

Configuring the Extensible SNMP Agent so that the management station can manage the new objects you add to the extensible agent is a three step process.

1. Write a MIB module, `snmpd.extend`, that extends the HP OpenView Extensible SNMP Agent to support new MIB objects. See "Step 1. Write MIB Module."
2. Copy the MIB module to the manager system. See "Step 2. Copy New MIB to the Manager System."
3. Integrate the new MIB objects into the manager's MIB. See "Step 3. Integrate New MIB into the Manager's MIB."

Figure 5-1 illustrates the configuration process.

Figure 5-1 Extensible SNMP Agent Configuration



Before You Begin

To write a MIB module, you need to understand what a MIB is and how MIBs are organized. For the manager to access the MIB objects you define on an agent, the MIB module you write must follow the conventions specified by *RFC 1155: Structure and Identification of Management Information for TCP/IP-based Internets* and *RFC 1212: Concise MIB Definitions*. If you are not familiar with these concepts, see

- The Request for Comments (RFC) documents. For a listing, see Chapter 1, “Introduction and Operational Concepts.”
- The sections “MIBs” and “How MIBs are Organized” in Chapter 1, “Introduction and Operational Concepts.”
- The “Sample MIB Solution” section in Chapter 6, “Creating your MIB Module.”

Step 1. Write MIB Module

To write a MIB module that extends the MIB on the HP OpenView Extensible SNMP Agent to support new objects, follow these steps:

1. Define your MIB objects.

   You can define one or more MIB objects, and you can group and define the MIB objects in one or more subtrees. The size of a subtree is limited to 200 nodes.

   To define your MIB objects, follow these steps:

   a. List all the objects that you want to add to the agent.

   b. Determine if objects can be grouped together into subtrees.

      Organize your MIB objects into logical groups. For example, the following MIB-II objects are all members of the `systems` group: `sysDescr`, `sysObjectID`, `sysUpTime`, `sysContact`, `sysName`, `sysLocation`, and `sysServices`. For a list of supported MIB objects, see Appendix A, Supported MIB Objects, or the MIB modules in the `snmp_mibs` directory.

   c. Define all the nodes in each subtree.

      Keep in mind that nodes can be children of other nodes.

      When defining the node, follow these rules defined by ASN.1:

      - Use an arbitrary number of letters, digits, and hyphens.
      - Begin with a lowercase letter.
      - Do not end with a hyphen.
      - Do not follow a hyphen with another hyphen.
      - Do not use underscores.

      You may also want to follow these conventions

      - End counter names with letters.
      - Give each node a unique name, although the name is not required to be unique.

d. Define the actual objects, that is, the leaf nodes in the subtree.

When you define the actual object, determine a unique name for the MIB objects.

Common conventions when defining the object names are to:

- Start all object names in a group with a prefix that can be derived from the group name. For example, the objects in the `systems` group all begin with the prefix `sys`. See the listing in step b.
- Capitalize the word after the prefix. For example, `Contact` is capitalized in the object name `sysContact`.

e. Determine where to place the object in the MIB tree.

To ensure that your object IDs are unique, add your MIBs under your own company (enterprise) name in the `enterprises` subtree. See Figure 1-2 on page 30 for an illustration of a MIB tree.

If you do not have an enterprise ID assignment, access the following web site for information about registering your enterprise:

`http://www.iana.org`

The benefit of registering your enterprise with the Internet Assigned Numbers Authority (IANA) is that you can control your own MIB subtree and avoid conflict with other MIBs.

2. Log on as root user to the agent system.
3. Create the MIB module, `snmpd.extend`.

Note that an example `snmpd.extend` file is provided with the HP OpenView Extensible SNMP Agent product in the `prg_samples/eagent` directory.

The `snmpd.extend` file is the MIB module that extends the MIB on the agent to include the objects you define. The `snmpd.extend` file is designed to use the macro template defined in *RFC 1212: Concise MIB Definitions*. Therefore, when you create the `snmpd.extend` file, follow the Abstract Syntax Notation One (ASN.1) format described in *RFC 1212*. Use the Hewlett-Packard enterprise-specific MIB or the Internet- standard MIB-II as a model. The Hewlett-Packard MIB is documented in Appendix A, “Supported MIB Objects.” MIB-II is documented in *RFC 1213*. You can also access these MIBs online. The respective files are `snmp_mibs/hp-unix` and `snmp_mibs/rfc1213-MIB-II`. After you create the `snmpd.extend` file, move it to the `/etc/snmpAgent.d` directory.

Refer to Chapter 6, “Creating your MIB Module,” for detailed information about defining MIB objects for the MIB module.

4. Reconfigure the HP OpenView Extensible SNMP Agent by killing `extsubagt` and then restarting it.

When you reconfigure the agent, the system checks the syntax and the structure of the `snmpd.extend` file and logs any errors or success to `snmpd.log`. To see the errors on standard error, kill `extsubagt` and restart it.

5. From the manager, verify that the agent responds to the objects you have added.

   To verify that the agent responds, use any of the SNMP commands provided with your SNMP manager product.

   If you use HP OpenView Network Node Manager, the SNMP commands are `snmpget`, `snmpnext`, `snmpset`, and `snmpwalk`. For information about the SNMP commands, see the manpages for `snmpget`, `snmpnext`, `snmpset`, and `snmpwalk`. After loading the MIB (`snmpd.extend`), you can also use MIB operations such as the MIB Browser. As an example of how to use an SNMP command, here is the syntax of the `snmpget` command

   `snmpget` *`hostname .objectID`*

   Note that when you specify the object ID in an SNMP command, the object ID must start with a dot (`.`).

   To troubleshoot any problems, see Chapter 7, “Troubleshooting.”

6. Configure all of your HP OpenView Extensible SNMP Agent systems.

   To configure additional HP OpenView Extensible SNMP Agent systems, do one of the following:

   - Copy the `snmpd.extend` file to all of your HP OpenView Extensible SNMP Agent systems.

     When you copy the `snmpd.extend` file to all your agents, all your agents are the same.

     After you copy the `snmpd.extend` file, you must reconfigure the agent by killing and then restarting `extsubagt`.

   - Create separate `snmpd.extend` files for each agent.

     If you want to manage different objects on the different agents, you can create separate `/etc/SnmpAgent.d/snmpd.extend` files for each agent. If you do this, make sure that the object IDs you use are unique. That is, each object ID should have the same description associated with it across all agents. To create separate `snmpd.extend` files for each agent, repeat steps 1 through 5 on each agent.

After you have configured all of your HP OpenView Extensible SNMP Agent systems, you are ready to configure the manager. See the next section “Step 2. Copy New MIB to the Manager System.”

Step 2. Copy New MIB to the Manager System

Before the manager can access the new MIB objects you add to the agent, you need to copy the MIB module, `snmpd.extend`, to the manager system.

You can copy the `snmpd.extend` file into any directory on the manager system. However, to make it easier to keep track of the MIB module files, you may want to copy your MIB module file to the default MIB module directory. If you use HP OpenView Network Node Manager, the default directory is `snmp_mibs`.

Copy the `snmpd.extend` file from the agent to `snmp_mibs/`*`myCompany`* where *`myCompany`* is a name that uniquely identifies your MIB.

If you create separate MIB object files for different agents, name your MIB module files `myCompanyAgent1`, `myCompanyAgent2`, and so forth.

Step 3. Integrate New MIB into the Manager's MIB

Once you have copied the MIB module to the manager system, integrate the new objects into the manager's MIB.

If you use HP OpenView Network Node Manager to manage the agent, the steps to integrate the new MIB into the manager's MIB are as follows:

1. Run HP OpenView Network Node Manager.
2. Load the new MIB, *`myCompany`*, into the manager's MIB.

   For HP OpenView Network Node Manager to access your new MIB objects, the MIB module defining those objects must be loaded into the manager's MIB. Use the `Load/Unload MIBs` operation to do so.

You are now ready to manage your HP OpenView Extensible SNMP Agents. For example, if you use the HP OpenView Network Node Manager you can use the `Browse MIB`, `MIB Data Collection`, `MIB Application Builder`, and the applications built by the `MIB Application Builder` to manage the new objects.

Configuring Traps

This section explains how to use the `snmptrap` command to send SNMP traps from agents to managers. It discusses the following:

- How to define traps.
- How traps are sent.
- When to use `snmptrap`.
- How to use the `snmptrap` command.
- Sample solution.

Before You Begin

To configure your agent to send traps, you need to understand what traps are. If you are not familiar with traps, see:

- The “Traps” section in Chapter 1, “Introduction and Operational Concepts.”
- *RFC 1157: A Simple Network Management Protocol (SNMP).*
- The manpage for `snmptrap`.

How to Define Traps

To define your own trap, you need to uniquely identify your trap. You do so by combining the generic Enterprise Specific trap 6 with your own specific trap number. The maximum specific trap number is 2^{32-1}.This combination tells the manager what kind of trap it is. For example, in the sample trap solution shown at the end of this section, the trap is a combination of 6 and the specific trap 2.

Optionally, you can pass along data.

How Traps Are Sent

The agent sends the traps using the `snmptrap` command. For example, you can configure your agent to send traps by executing the `snmptrap` command from a shell script.

When to Use snmptrap

As a manager, you have two alternatives when monitoring the status of an agent. You can:

- Continuously poll the agent from the manager to get information.
- Send a trap from the agent to the manager.

Polling creates a lot of traffic on the network and, if an event occurs shortly after polling has taken place, the manager may not find out about an event for an extended period of time. The key benefits of using the `snmptrap` command are that you can decrease the amount of SNMP traffic on the network and that you can find out about an event sooner.

If you have HP OpenView Network Node Manager, you can customize your environment by using the `snmptrap` command in conjunction with the `Event Configuration` operation. For example, assume that you have written a script on an agent that executes the `snmptrap` command when a particular process on the agent goes down. You can then use the `Event Configuration` operation from the HP OpenView Network Node Manager station to take an action when the manager receives that particular trap from the agent.

Using snmptrap

The `snmptrap` command is documented in the manpage for `snmptrap`.

Sample Trap Solution

Assume that you are responsible for managing the printers on your network. Your goal is to write a script that executes the `snmptrap` command when the printer scheduler goes down. Here is an example script for an HP-UX system.

```
#! /bin/sh
#
#
# This script checks to see if the printer scheduler (lpsched) is
# running. This check is performed every hour. If the scheduler is not
# running, the agent sends an SNMP trap to the management station.

#
# If a management station receives a trap from a system with enterprise
# equal to .1.3.6.1.4.1.4242, generic-trap equal to 6, and the specific
# trap equal to 2, the management station knows that the printer
# scheduler for that agent-addr is down.
#
# The agent sends how many hours the lp scheduler has been down with
# the trap.
#
#
AGENT_ADDRESS=15.6.71.223

MGMT_STATION=flcndmak

hours=0
```

```
while true
do
     sleep 3600
     pid=`ps -ef | grep lpsched| grep -v grep | wc -l`
     if [ $pid -eq 0 ]
     then
            hours=`expr $hours + 1`
            snmptrap $MGMT_STATION .1.3.6.1.4.1.4242 $AGENT_ADDRESS 6 2 0 \
            1.3.6.1.4.1.4242.4.2.0 integer $hours
     else
            hours=0
     fi
done
```

6 Creating your MIB Module

This chapter describes how to do the following:

- Determine the type of MIB object you want to define.

- Define MIB objects using commands.
- Define MIB objects using files. This includes simple objects and table objects.

This information provides the detailed instructions needed to complete the high-level procedure for writing a MIB module that was mentioned in Chapter 5, “Configuring the HP OpenView Extensible SNMP Subagent.”

Determining the Type of MIB Object to Define

The MIB objects that you define may be of two types. The objects may be associated with commands or they may be associated with a file. If the object is associated with a command and the agent receives an SNMP request for that object, the command is executed and the output of the command is returned in the SNMP reply. If the object is associated with a file and the agent receives an SNMP request for that object, the file is read and the contents of the file are returned in the SNMP reply.

When choosing whether to define objects as command objects or file objects you may want to ask the following questions:

Is a command required to obtain the object's value? If so, you may want to define the object associated with a command. If the value of the object does not require a command to be executed every time the object's value is retrieved, you may want to define the object associated with a file.

Are the objects arranged in a table (that is, do the objects have rows and columns)? If so, you must define the objects associated with a file.

Are you concerned with the performance of executing a command each time the object is retrieved? If so, you may want to avoid executing a command and define the objects associated with a file.

Once you have determined how to define your MIB object, use the macro template described in the next section to help you define the MIB object.

Using the Macro Template

Here is an illustration of the macro template you will use. Use this template to define both command and file MIB objects. You must fill in the fields shown in italics.

Note that the `snmpd.extend` file differs from the RFCs in the following areas:

- The `IMPORTS` and `EXPORTS` clauses are not required in the `snmpd.extend` file and will be ignored if added.
- The `DESCRIPTION` clause is required. Use this field to define the commands you want to execute.

```
moduleName DEFINITIONS ::= BEGIN

-- dashes indicate a comment

enterpriseName     OBJECT IDENTIFIER ::= { objectID }
```

```
nodeName            OBJECT IDENTIFIER ::=  { objectID }

Object  OBJECT-TYPE
   SYNTAX  Value
   ACCESS  Value
   STATUS  Value
   DESCRIPTION
   "Add a textual description of your object here along with:
          READ-COMMAND: read_command
          READ-COMMAND-TIMEOUT: timeout_in_seconds
          WRITE-COMMAND: write_command
          WRITE-COMMAND-TIMEOUT: timeout_in_seconds
          FILE-COMMAND: file_command
           FILE-COMMAND-FREQUENCY: file_command_seconds
           PIPE-IN-NAME: pipe_in_name
           PIPE-OUT-NAME: pipe_out_name
           PIPE-FREQUENCY: pipe_seconds
           APPEND-COMMUNITY-NAME: true | false
           FILE-NAME: file_name"

   ::= { parent_node subidentifier }
END
```

Here is an explanation of each command in the `snmpd.extend` file. For an example, see the section “Sample MIB Solution” later in this chapter.

Table 6-1

moduleName	The name of your MIB module.
insert two dashes (--)	Adds documentation in front of your comments.
enterpriseName	The enterprise ID you have registered with the Internet Assigned Numbers Authority. For example, Hewlett-Packard’s enterprise ID is `hp` and the corresponding `objectID` is `{ enterprises 11 }`.
nodeName	The name of the node under which you want to organize your MIB objects. You can have multiple node name entries and a node can be a child of another node.
Object	The textual label of your object.

Table 6-1 (Continued)

SYNTAX	Defines the data structure corresponding to the object type. The HP OpenView Extensible SNMP Agent supports the following values:
	INTEGER — A simple type consisting of positive and negative whole numbers, including zero. Do not use the value zero as an enumerated type.
	OCTET STRING — A simple type taking zero or more octets, each octet being an ordered sequence of eight bits.
	OBJECT IDENTIFIER — A type denoting an authoritatively named object. An example is 1.3.6.1.2.1.1.
	NULL — A simple type consisting of a single value, also called null. This type can only be used with defining an object associated with a command.
	NetworkAddress — A type representing an IP address.
	IpAddress — A type representing an IP address.
	Counter — A type representing a non-negative integer which calculates change and increases until it reaches a maximum value. When it reaches the maximum value, it wraps around and starts increasing again from zero.
	Gauge — A type representing a non-negative integer which may increase or decrease, but which latches at a maximum value.
	TimeTicks — A type representing a numeric value which counts the time since an event.
	Opaque — A type representing an arbitrary encoding.
	DisplayString — A type representing textual information taken from the NVT ASCII character set.
	PhysAddress — A type representing a media address. For many types of media, this will be a binary representation. For example, an ethernet address is represented as a string of six octets.

Table 6-2

SYNTAX	SEQUENCE OF	A type representing a table. This type represents objects that have rows and columns. This type can only be used when defining objects associated with a file.
	SEQUENCE	A type representing the entry of a table. This type is a child of an object with type SEQUENCE OF. This type may optionally have an INDEX clause. The INDEX clause identifies the column that uniquely defines the row. The default INDEX clause is the first column of the table. This type can only be used when defining objects associated with a file.
	The maximum value for INTEGER, Counter, Gauge, and TimeTicks is 2^{32-1} (4294967295 decimal).	
ACCESS	Defines the level of access allowed. Valid values are	
	read-only	Means you can perform GetRequests but not SetRequests on the object.
	read-write	Means you can perform both GetRequests and SetRequests on the object.
STATUS	Defines the implementation support required. Valid values for STATUS are mandatory, optional, obsolete, and deprecated. It is recommended that you use mandatory.	

Table 6-2 (Continued)

DESCRIPTION	Describes your object. This is also where you define the commands or files associated with the object. If your object is associated with a command, enter the following fields after the textual description: READ-COMMAND If your ACCESS *Value* is read-only or read-write, you must enter the READ-COMMAND. When the agent receives an SNMP GetRequest or a GetNextRequest, the agent executes the *read_command* and returns the results of that command in the SNMP reply. The output can be either standard out or standard error. READ-COMMAND-TIMEOUT Specifies the time in seconds you want the agent to wait for the *read_command* to finish. The maximum value is 90 seconds. The READ-COMMAND-TIMEOUT is optional. If you do not specify a *timeout_in_seconds* value, the agent will wait three seconds. WRITE-COMMAND If your ACCESS *Value* is read-write, you must enter the WRITE-COMMAND. The agent executes the *write_command* when it receives an SNMP SetRequest. The *write_command* should not generate output to standard out or standard error. WRITE-COMMAND-TIMEOUT Specifies the time in seconds you want the agent to wait for the *write_command* to finish. The maximum value is 90 seconds. The WRITE-COMMAND-TIMEOUT is optional. If you do not specify a *timeout_in_seconds* value, the agent will wait three seconds. If you specify a *timeout_in_seconds* value of -1, the *write_command* is executed and the agent responds without waiting for the command to finish. If the commands do not finish before the *timeout_in_seconds*, the agent kills the commands and returns an error. extsubagt processes one command at a time and waits for an answer before processing the next command.

Table 6-2 (Continued)

DESCRIPTION	The DESCRIPTION clause may contain the READ-COMMAND, READ-TIMEOUT, WRITE-COMMAND, and WRITE-TIMEOUT fields if the SYNTAX is one of the following: INTEGER, OCTET STRING, OBJECT IDENTIFIER, NULL, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks, Opaque, DisplayString, or PhysAddress. Refer to the next section for the syntax of the DESCRIPTION clause. If your object is associated with a file, enter the following fields after the textual description: FILE-COMMAND When the agent receives an SNMP GetRequest, GetNextRequest, or SetRequest, the agent executes the *file_command* before either reading or creating the *file_name*. When the agent receives an SNMP SetRequest, the agent also executes the *file_command* after the *file_name* has been created. The *file_command* must complete within 10 seconds. FILE-COMMAND-FREQUENCY When the agent receives an SNMP GetRequest or GetNextRequest, the agent executes the *file_command* if the agent last executed the *file_command* more than *file_command_seconds* ago. By default, the *file_command* will get executed at most every 10 seconds. The agent executes the *file_command* both before and after the file has been created for every SNMP SetRequest regardless of when it was last executed. PIPE-IN-NAME After the agent writes to the *pipe_out_name*, the agent reads the *pipe_in_name* waiting for a message. If the agent reads a 0, the agent continues processing and either reads or creates a *file_name*. If the agent reads something other that 0, or if the agent does not receive a message within DEFAULT_PIPE_TIMEOUT (20 seconds), the agent returns an error. The PIPE-IN-NAME is required if the PIPE-OUT-NAME is present. Both the *pipe_in_name* and *pipe_out_name* can be created using the mkfifo command.

Table 6-3

DESCRIPTION	PIPE-OUT-NAME	When the agent receives an SNMP GetRequest, GetNextRequest, or SetRequest, the agent writes to the *pipe_out_name*. The *pipe_out_name* must be a FIFO (named pipe). The management station's IP address, the community name used in the request, the OBJECT IDENTIFIER used in the request, the SYNTAX of the object, the request issued by the management station, and the instance are written to the pipe. Each value is separated by white space and the message ends with the \0 character. When the agent receives an SNMP SetRequest, the agent also writes to *pipe_out_name* after the *file_name* has been created.
	PIPE-FREQUENCY	When the agent receives an SNMP GetRequest or GetNextRequest, the agent writes to the *pipe_out_name* if the agent last wrote to the *pipe_out_name* more than *pipe_seconds* ago. By default, the *pipe_out_name* will get written to at most every 10 seconds. The agent writes to the *pipe_out_name* both before and after the file has been created for every SNMP SetRequest, regardless of when it was last written to.
	APPEND-COMMUNITY-NAME	If APPEND-COMMUNITY-NAME is true, the agent reads or creates *file_name.comm*, where *comm* is the community name sent in the request. If *file_name.comm* is not present, the agent returns an error. The value for APPEND-COMMUNITY-NAME must be either true or false.
	FILE-NAME	When the agent receives an SNMP GetRequest or GetNextRequest, the agent reads the *file_name* and returns the contents of that file in the SNMP Reply. When the agent receives an SNMP SetRequest, the agent creates *file_name* containing the value that the object is set to. The *file_name* will no longer contain comments after the agent creates it.
	extsubagt processes one command at a time and waits for an answer before processing the next command.	
	Note that the read and write access values do not have anything to do with the read and write file permissions.	

Table 6-3 (Continued)

DESCRIPTION	The FILE-COMMAND, FILE-COMMAND-FREQUENCY, PIPE-IN-NAME, PIPE-OUT-NAME, PIPE-FREQUENCY, and APPEND-COMMUNITY-NAME clauses are optional. The DESCRIPTION clause may contain the FILE-NAME field if the SYNTAX is one of the following: INTEGER, OCTET STRING, OBJECT IDENTIFIER, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks, Opaque, DisplayString, PhysAddress, SEQUENCE, or SEQUENCE OF. Refer to the next section for the syntax of the DESCRIPTION clause.
parent_node	Name of the node you have already identified. *subidentifier* is the number that is appended to the parent node to make the object unique.

The DESCRIPTION Clause

Valid syntax and ordering for Extensible SNMP Agent commands in DESCRIPTION clause are:

Simple Objects

```
READ-COMMAND: command-line
[ READ-COMMAND-TIMEOUT: seconds-to-timeout ]
WRITE-COMMAND: command-line
[ WRITE-COMMAND-TIMEOUT: seconds-to-timeout ]
```

EXAMPLE:

```
READ-COMMAND: /usr/bin/users
READ-COMMAND-TIMEOUT: 10
```

Simple Objects and Table Objects

```
[ FILE-COMMAND: command-line  [ FILE-COMMAND-FREQUENCY: seconds ] ]
[ PIPE-IN-NAME: pipe-name
PIPE-OUT-NAME: pipe-name
[ PIPE-FREQUENCY: seconds ] ]
[ APPEND-COMMUNITY-NAME: true | false ]
FILE-NAME: filename
```

EXAMPLE:

```
FILE-COMMAND: /usr/bin/filecmd
FILE-COMMAND-FREQUENCY: 30
APPEND-COMMUNITY-NAME: false
FILE-NAME: /tmp/myfile
```

Ordering is important. For example, FILE-COMMAND must be before FILE-NAME.

Using Commands to Define your MIB Object

If you use commands to define your MIB object, the command can be either an existing UNIX command, or a command that you have written. If you write your own commands, see "Writing Shell Commands" later in this section.

- You specify these commands in the `DESCRIPTION` clause in the `snmpd.extend` file.
- The maximum command size is 5120 characters.
- A command can span multiple lines. Optionally, end each line with a backslash (`\`).

Sample MIB Solution

To illustrate how you can configure your HP OpenView Extensible SNMP Agent to support any object you define, this section has a step-by-step sample solution.

Assume you work for the Flintstones Company. Your goal is to write MIB objects which will:

- List users who are using a system.
- Manage mail queues.
- Manage the number of widgets produced per hour on an unattended system.
- Keep track of the LP scheduler.

Your agent system is `flintagent` with the default community name `public`. The set community name is `secret`.

The manager system is running the HP OpenView Network Node Manager software.

Here are the steps.

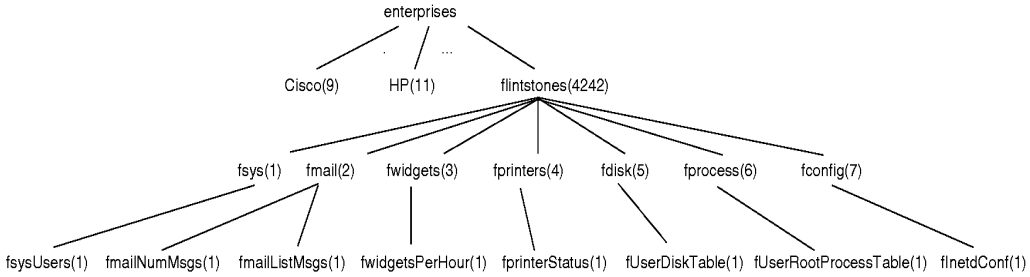
1. Define your MIB objects.

   To ensure that your object IDs are unique, you decide to define your MIB objects in the `flintstones (4242)` subtree. The MIB tree hierarchy is:

```
internet     OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises  OBJECT IDENTIFIER ::= { internet private (4) 1 }
flintstones  OBJECT IDENTIFIER ::= { enterprises 4242 }
fsys         OBJECT IDENTIFIER ::= { flintstones 1 }
fmail        OBJECT IDENTIFIER ::= { flintstones 2 }
fwidgets     OBJECT IDENTIFIER ::= { flintstones 3 }
fprinters    OBJECT IDENTIFIER ::= { flintstones 4 }
fdisk        OBJECT IDENTIFIER ::= { flintstones 5 }
fprocess     OBJECT IDENTIFIER ::= { flintstones 6 }
fconfig       OBJECT IDENTIFIER ::= { flintstones 7 }
```

   The MIB objects are defined and organized into the logical groups shown in Figure 6-1.

Figure 6-1 MIB Tree Structure for Sample MIB Solution



The object identifiers for the leaf nodes (the MIB objects) are shown in Table 6-4:

Table 6-4 Leaf Node Object Identifiers

Leaf Node	Object Identifier
`fsysUsers`	`1.3.6.1.4.1.4242.1.1.0`
`fmailNumMsgs`	`1.3.6.1.4.1.4242.2.1.0`
`fmailListMsgs`	`1.3.6.1.4.1.4242.2.2.0`
`fwidgetsPerHour`	`1.3.6.1.4.1.4242.3.1.0`
`fprintersStatus`	`1.3.6.1.4.1.4242.4.1.0`

Note that the suffix `0` indicates that this is a leaf node (instance `0`).

2. Log on as root user to the `flintagent` system.
3. Create the MIB module, `snmpd.extend`.

   You may decide to use the example `snmpd.extend` file provided with the HP OpenView Extensible SNMP Agent as a model. To do so, copy `prg_samples/eagent/snmpd.extend` to `snmpd.extend`. This is the example `snmpd.extend` file.

```
FLINTSTONES DEFINITIONS ::= BEGIN

-- This MIB module, snmpd.extend, defines the MIB objects for the
-- Flintstones Company.

internet       OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises    OBJECT IDENTIFIER ::= { internet private(4) 1 }
```

```
flintstones     OBJECT IDENTIFIER ::= { enterprises 4242 }
fsys            OBJECT IDENTIFIER ::= { flintstones 1 }
fmail           OBJECT IDENTIFIER ::= { flintstones 2 }
fwidgets        OBJECT IDENTIFIER ::= { flintstones 3 }
fprinters       OBJECT IDENTIFIER ::= { flintstones 4 }
fdisk           OBJECT IDENTIFIER ::= { flintstones 5 }
fprocess        OBJECT IDENTIFIER ::= { flintstones 6 }
fconfig         OBJECT IDENTIFIER ::= { flintstones 7 }

-- The fsys Group

fsysUsers       OBJECT-TYPE
    SYNTAX      DisplayString
    ACCESS      read-only
    STATUSmandatory
    DESCRIPTION
            "List of users on the flintstones machine.
             READ-COMMAND: /usr/bin/users; exit 0
             READ-COMMAND-TIMEOUT: 5"
    ::= { fsys 1 }

-- The fmail Group

fmailNumMsgs    OBJECT-TYPE
    SYNTAX      Gauge
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
           "Message count on the mail queue.
            READ-COMMAND: /usr/bin/mailq -bp | fgrep -v Mail | wc -l"
    ::= { fmail 1 }

fmailListMsgs     OBJECT-TYPE
    SYNTAX        DisplayString
    ACCESS        read-only
    STATUS        mandatory
    DESCRIPTION
            "List of mail messages on the mail queue.
             READ-COMMAND: /usr/bin/mailq
             READ-COMMAND-TIMEOUT: 10"
    ::= { fmail 2 }



-- The fwidgets Group

fwidgetsPerHour     OBJECT-TYPE
    SYNTAX          Gauge
    ACCESS          read-write
    STATUS          mandatory
```

```
    DESCRIPTION
            "Number of widgets produced per hour.
             READ-COMMAND: prg_samples/eagent/num_widgets $i $c $o $s
             READ-COMMAND-TIMEOUT: 2
             WRITE-COMMAND: prg_samples/eagent/change_num_widgets $*
             WRITE-COMMAND-TIMEOUT: 10"
    ::= { fwidgets 1 }

-- The fprinters Group

fprintersStatus    OBJECT-TYPE
    SYNTAX         Integer {
                      up(1),
                      down(2)
                   }
    ACCESS         read-only
    STATUS         mandatory
    DESCRIPTION
            "Status of the printer scheduler.
             READ-COMMAND: ps -ef | grep lpsched | grep -v grep | wc |
             awk '{ if ($1 == 0) print 2; else print 1 }' "
    ::= { fprinters 1 }

END
```

Note the following:

- The commands you are executing are defined in the `DESCRIPTION` clause in the `snmpd.extend` file. One benefit of adding the commands to the `DESCRIPTION` clause is that you can see what commands you are executing when you ask for description from the manager.
- The `READ-COMMAND-TIMEOUT` and the `WRITE-COMMAND-TIMEOUT` lines are optional. It indicates the time in seconds the agent will wait for a response. If a timeout is not specified, the agent by default waits three seconds for a response. If the command does not finish before the timeout, the agent kills the command and returns an error.

4. Reconfigure the HP OpenView Extensible SNMP Agent (`extsubagt`) by killing it and restarting it.

   When you reconfigure the agent, the system checks the syntax and the structure of the `snmpd.extend` file and logs any errors or success to `snmpd.log`. To see the errors on standard error, kill `extsubagt` and restart it.

5. From the manager, verify that `flintagent` responds to the objects you have added.

   Use any of the SNMP commands or use the MIB operations in HP OpenView Network Node Manger such as the MIB Browser. For information about the SNMP commands, see the manpages for `snmpget`, `snmpnext`, `snmpset`, and `snmpwalk`.

For example, to verify that the `fsysUsers` object returns a list of users with the `snmpget` command, type:

```
snmpget flintagent .1.3.6.1.4.1.4242.1.1.0
```

Note that when you specify the object ID in an SNMP command, the object ID must start with a dot (`.`).

The `snmpget` command executes the /`usr/bin/ucb/users` command and returns a list of users logged into the `flintagent` system.

To change the `fwidgetsPerHour` object to produce 15 widgets per hour with the `snmpset` command, type:

```
snmpset -c secret flintagent .1.3.6.1.4.1.4242.3.1.0 Gauge 15
```

Your sample configuration is done.

Refer to the “Writing Shell Commands” section later in this chapter if you need information on writing shell commands.

Using Files to Define Your MIB Object

Both simple objects and tables can be added to the MIB module. The file based method requires a new set of keywords in the `DESCRIPTION` clause of the `OBJECT-TYPE` macro definition. The agent recognizes the `FILE-NAME`: field in the `DESCRIPTION` clause. The file associated with the `FILE-NAME`: field is read to retrieve the values of the object and created to modify the values associated with the object.

The following objects can be added using the file based technique:

```
INTEGER
OCTET STRING
OBJECT IDENTIFIER
NetworkAddress
IpAddress
Counter
Gauge
TimeTicks
Opaque
DisplayString
PhysAddress
```

```
SEQUENCE

SEQUENCE OF
```

These objects can have ACCESS of read-write or read-only.

Simple Objects

These are objects that have one value. For example, most systems have only one CPU model number. This object would be a simple object. If a system had several CPUs, a table of CPU model numbers would be required. Tables are discussed later.

If the SYNTAX of the object is OCTET STRING, Opaque, or DisplayString, the agent reads the entire file for the value. The maximum size of the file contents is 4096 (4K). If the string is prefixed with 0x, the value of the string is converted to hexadecimal. For example, if the file contained 0x0800093519D0, the agent would return the hexadecimal representation of this value rather than the ASCII representation.

Other objects defined are read from the file looking for the first value separated by white space. You can use the # character in the first column for comments.

EXAMPLE:

You want the agent to respond with the system's default printer. The default printer is stored in /usr/spool/lp/default. You also want to modify the default printer, so you want to update /usr/spool/lp/default remotely using SNMP.

The following entries are entered into snmpd.extend to add this object to the agent.

```
internet         OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises      OBJECT IDENTIFIER ::= { internet private(4) 1 }
flintstones      OBJECT IDENTIFIER ::= { enterprises 4242 }
fprinters        OBJECT IDENTIFIER ::= { flintstones 4 }

fdefaultPrinter  OBJECT-TYPE
    SYNTAX       DisplayString
    ACCESS       read-write
    STATUS       mandatory
    DESCRIPTION
            "Default printer
             FILE-NAME: /usr/spool/lp/default"
    ::= { fprinters 2 }
```

Note that this object is a string and has read-write access.

If the file /usr/spool/lp/default contains ljetp4 and the agent receives an SNMP GetRequest for flintstones.fprinters.fdefaultPrinter.0, the extsubagt reads/usr/spool/lp/default and returns ljetp4 to the management station.

The agent caches this value in memory for future possible requests. If the file contents do not change (the modification time of the file does not change) and the agent receives another SNMP GetRequest for `flintstones.fprinters.fdefaultPrinter.0`, the agent returns `ljetp4` from its cache rather than re-read the file contents.

The same procedures are used for GetNextRequests. If the agent receives an SNMP GetNextRequest for `flintstones.fprinters.fdefaultPrinter` the agent reads `/usr/spool/lp/default` and returns `ljetp4` for the object `flintstones.fprinters.fdefaultPrinter.0` to the management station.

If the agent receives a SNMP SetRequest for `flintstones.fprinters.fdefaultPrinter.0` with value `laserjet-l4` the agent will create `/usr/spool/lp/default` with the contents `laserjet-l4`. The contents of the file before the SetRequest are gone. If the agent cannot write the file, a `genError` is returned and the error is logged to `/usr/adm/snmpd.log`.

Table Objects

Table objects are useful for objects that have several values. An example of table objects found in MIB-II are the TCP connection table and the interface table.

EXAMPLE:

You want to retrieve the list of users on a remote machine, their user id, the disk space associated with the user, and the e-mail address of the user. The data on the remote machine is located in a file `prg_samples/eagent/user_disk_space`. The contents of the file look like the following:

```
100     zach      120       zach@server1
201     alice     65        alice@server2
320     john      2         john@server3
119     craig     500       root@server1
217     steve     75        steve@server1
83      bob       111       bob@harrumph
```

This table has four columns and six rows. Every table defined using the Extensible SNMP Agent must have a column or a set of columns that uniquely define the row. In some models, this column would be called a key. In this example, the first column is unique. The `User ID` is unique on this system. If the `User Names` are unique, the second column could be used as the key.

To configure the agent to respond to these objects, you need to make the following entries into the agent configuration file `snmpd.extend`.

```
internet     OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises  OBJECT IDENTIFIER ::= { internet private(4) 1 }
flintstones  OBJECT IDENTIFIER ::= { enterprises 4242 }
fdisk        OBJECT IDENTIFIER ::= { flintstones 5 }
```

```
UserDiskTable         OBJECT-TYPE
    SYNTAX            SEQUENCE OF FUserDiskEntry
    ACCESS            not-accessible
    STATUS            mandatory
    DESCRIPTION
            "List of users and the number of kilobytes in their home directory.
             FILE-NAME: prg_samples/eagent/user_disk_space"
    ::= { fdisk 1 }

fUserDiskEntry        OBJECT-TYPE
    SYNTAX            FUserDiskEntry
    ACCESS            not-accessible
    STATUS            mandatory
    DESCRIPTION
            "This macro documents the column that uniquely describes each row."
    INDEX { fUID }
    ::= { fUserDiskTable 1 }

FUserDiskEntry ::=
    SEQUENCE {
       fUID INTEGER,
       fUserName DisplayString,
       fUserDiskSpace INTEGER,
       fUserEmailAddress DisplayString
    }

fUID                  OBJECT-TYPE
    SYNTAX            INTEGER
    ACCESS            read-only
    STATUS            mandatory
    DESCRIPTION
            "User's unique ID"
    ::=  { fUserDiskEntry 1 }

fUserName         OBJECT-TYPE
    SYNTAX        DisplayString
    ACCESS        read-only
    STATUS        mandatory
    DESCRIPTION
           "User login name"
    ::= { fUserDiskEntry 2 }

fUserDiskSpace    OBJECT-TYPE
    SYNTAX        INTEGER
    ACCESS        read-only
    STATUS        mandatory
    DESCRIPTION
            "Amount of disk space (in kilobytes) used by the user."
    ::= { fUserDiskEntry 3 }
```

```
fUserEmailAddress  OBJECT-TYPE
    SYNTAX         DisplayString
    ACCESS         read-write
    STATUS         mandatory
    DESCRIPTION
            "Email address for the user."
    ::= { fUserDiskEntry 4 }
```

The first `OBJECT-TYPE` macro, `fUserDiskTable`, describes the file name associated with the object. The second `OBJECT-TYPE` macro, `fUserDiskEntry`, describes the column that uniquely identifies a row. The next entry, `FUserDiskEntry`, is for documentation purposes. This entry lists the columns in the table. This entry is optional. The last four `OBJECT-TYPE` macros, `fUID`, `fUserName`, `fUserDiskSpace`, and `fUserEmailAddress` define the columns in the table.

If the agent receives a GetNextRequest for `fUserDiskTable.fUserDiskEntry.fUID`, the agent will read the entire file `prg_samples/eagent/user_disk_space`. The agent then sorts the table based on the object specified in the `INDEX` clause. The sorted table will then look like this:

```
83      bob          111    bob@harrumph
100     zach         120    zach@server1
119     craig        500    root@server1
201     alice        65     alice@server2
217     steve        75     steve@server1
320     john         2      john@server3
```

The file contents have not been changed, the cached values in the agent are sorted.

The agent would then return the first value in the table. This value would be the first column of the first row. The management station would receive the value `83` for the MIB object `flintstones.fdisk.fUserDiskTable.fUserDiskEntry.fUID.83`. The `fUID` has `83` appended to the object identifier.

The user ID `83` uniquely identifies the value in the row.

If the agent receives an SNMP GetNextRequest for `flintstones.fdisk.fUserDiskTable.fUserDiskEntry.fUID.83`, the agent would check to see if the file has been modified. If the file has been modified, the agent would re-read `prg_samples/eagent/user_disk_space`. It would then return the next user ID after the user ID for `bob`. It would return the value `100` for `flintstones.fdisk.fUserDiskTable.fUserDiskEntry.fUID.100`.

If the agent receives an SNMP GetNextRequest for `flintstones.fdisk.fUserDiskTable.fUID.320`, which is the user ID for `john`, the agent would then notice that there are not any more user names and it would return the first value in the second column. It would return the value `bob` for MIB object `fUserDiskEntry.fUserName.83`.

If the agent receives an SNMP GetRequest for `fUserDiskEntry.fUserEmailAddress.217` the agent would look for an e-mail address associated with the row `217`. `217` is the key for `steve`, so the agent would return the value `steve@server1` for the MIB object f`UserDiskEntry.fUserEmailAddress.217`.

If the user would like to print the entire table, the user would issue an SNMP GetNextRequest for `fUserDiskTable.fUserDiskEntry.fUID`, `fUserDiskTable.fUserDiskEntry.fUserName`, `fUserDiskTable.fUserDiskEntry.fUserDiskSpace`, and `fUserDiskTable.fUserDiskEntry.fUserEmailAddress`.

The agent would return the first row back to you. The agent would return the following MIB objects value pairs:

```
fUserDiskTable.fUserDiskEntry.fUID.83, 83
fUserDiskTable.fUserDiskEntry.fUserName.83, "bob"
fUserDiskTable.fUserDiskEntry.fUserDiskSpace.83, 111
fUserDiskTable.fUserDiskEntry.fUserEmailAddress.83, "bob@harrumph"
```

The user would then issue another SNMP GetNextRequest for

```
fUserDiskTable.fUserDiskEntry.fUID.83
fUserDiskTable.fUserDiskEntry.fUserName.83
fUserDiskTable.fUserDiskEntry.fUserDiskSpace.83
fUserDiskTable.fUserDiskEntry.fUserEmailAddress.83
```

The agent would reply with the next row in the table. The following MIB object, value pairs would be returned.

```
fUserDiskTable.fUserDiskEntry.fUID.100, 100
fUserDiskTable.fUserDiskEntry.fUserName.100, "zach"
fUserDiskTable.fUserDiskEntry.fUserDiskSpace.100, 120
fUserDiskTable.fUserDiskEntry.fUserEmailAddress.100, "zach@server1"
```

This would continue until the last row was retrieved.

If you would want to modify the e-mail address for "`alice`", the user would issue an SNMP SetRequest for MIB object `fUserDiskTable.fUserDiskEntry.fUserEmailAddress.201` with value `"alice@mailer"`. The agent would write this value, along with all the other entries in the table to `prg_samples/eagent/user_disk_space`. The file would then contain:

```
100     "zach"     120      "zach@server1"
201     "alice"    65       "alice@mailer"
320     "john"     2        "john@server3"
119     "craig"    500      "root@server1"
217     "steve"    75       "steve@server1"
83      "bob"      111      "bob@harrumph"
```

The following rules apply to creating the file that contains a table:

- Each row of the table ends in a new-line. A row can continue over a new- line by adding the character \ at the end of the line. For example, if the file contains

  ```
  Column1 "Column # 2" \
  "Column 3" Column4 Column5
  ```

  The agent would consider this as one row with five columns.

- Each column is separated by white space. A column may be enclosed in double quotes. The second column of the above example is enclosed in double quotes and is equal to `Column # 2`. The agent converts `\"` to `"`, and `\\"` to `\"`.

  For example, if the file contains

  ```
  "This is an \"example\" of a column with \\" style quotes"
  ```

  The agent would return the following to the management station.

  ```
  This is an "example" of a column with \" style quotes
  ```

- A column's value may not extend over a new-line.

Filling the File with Values

The file contents can be updated using the usual conventions available with UNIX. For example, you can enter static configuration information using an editor. For information that changes every five minutes, you can execute a `cron` command to update the file's contents. For example, a UNIX script could be run every night that creates `prg_samples/eagent/user_disk_space` containing the disk space for every user.

A UNIX process may elect to update the file with up-to-date data. If you want a UNIX command to be executed before the file is read, you can optionally enter the field `FILE-COMMAND` in the description clause found in `snmpd.extend`.

If you want a UNIX process to receive a message before the file is read, you can optionally enter the filed `PIPE-IN-NAME` and `PIPE-OUT-NAME` in the `DESCRIPTION` clause found in `snmpd.extend`.

The FILE-COMMAND

The *`file_command`* specified after the `FILE-COMMAND` keyword is executed before the agent reads the data file. The command may update the contents of the file.

For example, if you want to remotely retrieve the list of processes that are owned by root, you would define the following MIB:

```
internet        OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises     OBJECT IDENTIFIER ::= { internet private(4) 1 }
flintstones     OBJECT IDENTIFIER ::= { enterprises 4242 }
fprocess        OBJECT IDENTIFIER ::= { flintstones 6 }
```

```
fUserRootProcessTable    OBJECT-TYPE
    SYNTAX               SEQUENCE OF FUserRootProcessEntry
    ACCESS               not-accessible
    STATUS               mandatory
    DESCRIPTION
            "List of root processes.
             FILE-COMMAND: prg_samples/eagent/get_processes
             FILE-NAME: prg_samples/eagent/root_processes"
       ::= { fprocess 1 }

fUserRootProcessEntry     OBJECT-TYPE
    SYNTAX                FUserRootProcessEntry
    ACCESS                not-accessible
    STATUS                mandatory
    DESCRIPTION
            "This macro documents the column that uniquely describes each row."
             INDEX { fProcessID }
    ::= { fUserRootProcessTable 1 }

FUserRootProcessEntry     ::=
    SEQUENCE    {
    fProcessID            INTEGER,
    fProcessName          DisplayString
    }

fProcessID                OBJECT-TYPE
    SYNTAX                INTEGER
    ACCESS                read-only
    STATUS                mandatory
    DESCRIPTION
            "Process ID"
    ::= { fUserRootProcessEntry 1 }

fProcessName              OBJECT-TYPE
    SYNTAX                DisplayString
    ACCESS                read-write
    STATUS                mandatory
    DESCRIPTION
            "Name of process"
    ::= { fUserRootProcessEntry 2 }
```

If the agent receives an SNMP GetNextRequest for

```
fprocess.fUserRootProcessTable.fUserRootProcessEntry.fProcessID
fprocess.fUserRootProcessTable.fUserRootProcessEntry.fProcessName
```

the agent would execute the following command

```
prg_samples/eagent/get_processes
```

This command creates the file

```
prg_samples/eagent/root_processes
```

which contains the process ids and process names run by root.

The agent then reads `prg_samples/eagent/root_processes`, sorts the contents, and returns the first row to the management station.

By default, the command is run at most once every 10 seconds. For example, if the agent receives another SNMP GetNextRequest for

```
fprocess.fUserRootProcessTable.fUserRootProcessEntry.fProcessID
fprocess.fUserRootProcessTable.fUserRootProcessEntry.fProcessName
```

the agent would only execute the following command if it had been more than 10 seconds since the agent last executed the command.

```
prg_samples/eagent/get_processes
```

If you only want the command to be run at most every hour, you can change the default frequency by entering the following in the `DESCRIPTION` clause:

```
FILE-COMMAND-FREQUENCY: 3600
```

The command must exit with value `0`, or a `noSuchName` error is returned to the management station. The agent will kill the command if it does not return in 10 seconds.

Using the FILE-COMMAND with Set Requests

The *`file_command`* is executed before and after a set request happens. This occurs every time a set operation is requested regardless of when the command was last executed.

For example, if you want to be able to update the configuration file for `inetd` the following object would be defined.

```
internet        OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises     OBJECT IDENTIFIER ::= { internet private(4) 1 }
flintstones     OBJECT IDENTIFIER ::= { enterprises 4242 }
fconfig          OBJECT IDENTIFIER ::= { flintstones 7 }

fInetdConf      OBJECT-TYPE
    SYNTAX      DisplayString
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
            "The configuration file for inetd
         FILE-COMMAND: prg_samples/eagent/update_inetd $r
         FILE-COMMAND-FREQUENCY: 7200
         FILE-NAME: inetd.conf"
    :: = { fconfig 1 }
```

If the agent receives a SNMP SetRequest for `flintstones.fconfig.fIntdConf.0` with the contents of a `inetd.conf` configuration file, the agent would execute the command

```
prg_samples/eagent/update_inetd SetRequest
```

The agent would then write the contents to `inetd.conf`. The agent then executes

```
prg_samples/eagent/update_inetd PostSetRequest
```

The command checks the value of the first argument. If the value is `PostSetRequest`, the command executes `inetd -c` to tell `inetd` to re- read its configuration file.

Using the PIPE-IN-NAME and PIPE-OUT-NAME Clauses

The `PIPE-IN-NAME` and `PIPE-OUT-NAME` clauses are used for interprocess communication between the agent and another UNIX process.

The *`pipe_out_name`* file name specified after the `PIPE-OUT-NAME` receives data before the agent reads the file. After a process reads the data, the process may update the contents of the file. After the process is finished updating the file, it must notify the agent that it is done by writing to the *`pipe_in_name`* file specified after `PIPE-IN-NAME`.

Lets use the same example for `FILE-COMMAND`. Rather than execute a command to fill the contents of the file, a process will be running in the background waiting to fill the contents of the file.

```
internet       OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises    OBJECT IDENTIFIER ::= { internet private(4) 1 }
flintstones    OBJECT IDENTIFIER ::= { enterprises 4242 }
fprocess       OBJECT IDENTIFIER ::= { flintstones 6 }

fUserRootProcessTable      OBJECT-TYPE
    SYNTAX                 SEQUENCE OF FUserRootProcessEntry
    ACCESS                 not-accessible
    STATUS                 mandatory
    DESCRIPTION
    "List of root processes.
     PIPE-IN-NAME: /tmp/fifo_in
     PIPE-OUT-NAME: /tmp/fifo_out
     FILE-NAME:prg_samples/eagent/root_processes"
        ::= { fprocess 1 }

fUserRootProcessEntry       OBJECT-TYPE
    SYNTAX                  FUserRootProcessEntry
    ACCESS                  not-accessible
    STATUS                  mandatory
    DESCRIPTION
            "This macro documents the column that uniquely describes each row."
    INDEX { fProcessID }
    ::= { fUserRootProcessTable 1 }

FUserRootProcessEntry ::=
    SEQUENCE {
```

```
    fProcessID INTEGER,
    fProcessName  DisplayString
    }

fProcessID                  OBJECT-TYPE
    SYNTAX                  INTEGER
    ACCESS                  read-only
    STATUS                  mandatory
    DESCRIPTION
            "Process ID"
    ::= { fUserRootProcessEntry 1 }

fProcessName                OBJECT-TYPE
    SYNTAX                  DisplayString
    ACCESS                  read-write
    STATUS                  mandatory
    DESCRIPTION
            "Name of process"
    ::= { fUserRootProcessEntry 2 }
```

If the agent receives an SNMP GetNextRequest for

```
fprocess.fUserRootProcessTable.fUserRootProcessEntry.fProcessID
fprocess.fUserRootProcessTable.fUserRootProcessEntry.fProcessName
```

the agent would send a message to `/tmp/fifo_out`.

The message contents include:

- The manager's IP address
- Community name
- Object id
- Object type
- PDU type
- Instance

The end of the message is denoted by a `\0` character.

The background process would then receive this message, create the file

`prg_samples/eagent/root_processes`

containing the process ids and process names run by root.

The background process then writes a `0` to `/tmp/fifo_in` to indicate that the file creation was successful. If the agent receives data other than a `0`, a `noSuchName` error is returned to the management station and a message is logged to `snmpd.log`.

The agent then reads `prg_samples/eagent/root_processes`, sorts the contents, and returns the first row to the management station.

An example background process might look like this:

```
#! /bin/ksh

set -u
#set -x

PIPE_IN="$PWD/fifo_in"
PIPE_OUT="$PWD/fifo_out"

if [ -p "PIPE_OUT" ] ; then
  ! rm -f "PIPE_OUT"
fi

! mkfifo "PIPE_OUT"
! chmod 777 "PIPE_OUT"

if [ -p "$PIPE_IN" ] ; then
  ! rm -f "PIPE_IN"
fi

! mkfifo "PIPE_IN"
! chmod 777 "PIPE_IN"

while true ; do
  echo "\n-----------------------------------------------------"
  read ipAddr comm oid oidType pduKeywd instance theRest < "PIPE_OUT"
  echo "         ipAddr           '$ipAddr'"
  echo "         comm             '$comm'"
  echo "         oid              '$oid'"
  echo "         oidType          '$oidType'"
  echo "         pduKeywd         '$pduKeywd'"
  echo "         instance         '$instance'"
  echo "         theRest          '$theRest'"
  echo
  echo "\00" >> "PIPE_IN"       # Handshake with snmpd.
done < "PIPE_OUT"       # Extra pipe reader to keep pipe open so snmpd read
```

The files `/tmp/fifo_out` and `/tmp/fifo_in` can be created using the `mkfifo` command.

By default, data is written to the *`pipe_out_name`* pipe no more often than every 10 seconds. To change this frequency, you can specify a new value using the `PIPE-FREQUENCY` clause.

Set operations using the `PIPE` clauses work similarly as the `FILE-COMMAND`. Data is written to the *`pipe_out_name`* pipe before and after the pipe is read. After the process receives data from the *`pipe_out_name`* pipe the process must write data to the *`pipe_in_name`* pipe.

The agent will wait at the most, `DEFAULT_PIPE_TIMEOUT` (20 seconds) for data from the *`pipe_in_name`* pipe.

For a set request, extsubagt does the following actions:

- Sends a SetRequest to change value of object. The external process responds with "\0" in PIPE_IN.
- Queries external process if the set was a success by sending PostSetRequest into PIPE_OUT. The external process now responds with "\0" in PIPE_IN.

NOTE If the external process is unable to set the value, it will not respond with a "\0" into PIPE_IN.

If the variables of multi-Varbind requests belong to the same family:

- For SET/GET requests, only the first variable is put into the PIPE file.
- For SET requests, only the last variable PostSetRequest is put into the PIPE file.

extsubagt assumes that all objects that belong to a particular family are handled by a single external process. So, when you request a GET:

- The external process updates the data of all the objects in the family.
- extsubagt reads the data for all objects and updates its internal cache.

Creating Proxies Using the Extensible SNMP Agent

You can use the Extensible SNMP Agent to create a proxy. The agent can respond to objects on behalf of another system, device, or application.

For example, if you want the agent to respond with the amount of memory for three systems that do not support SNMP, the agent can act as a proxy for those other systems. The three systems that do not support SNMP are named larry, curly, and moe. The following three files contain the amount of memory on each system:

```
prg_samples/eagent/memory.larry
prg_samples/eagent/memory.curly
prg_samples/eagent/memory.moe
```

prg_samples/eagent/memory.larry contains the value 32 for 32 megabytes. prg_samples/eagent/memory.curly contains 128 and prg_samples/eagent/memory.moe contains 64.

Larry has IP address 15.2.112.244, curly has IP address 15.2.114.237, and moe has IP address 15.2.113.223.

To implement this proxy, you would define the following object:

```
internet        OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises     OBJECT IDENTIFIER ::= { internet private(4) 1 }
flintstones     OBJECT IDENTIFIER ::= { enterprises 4242 }
fsys            OBJECT IDENTIFIER ::= { flintstones 1 }

fmemory             OBJECT-TYPE
    SYNTAX          INTEGER
    ACCESS          read-only
    STATUS          mandatory
    DESCRIPTION
            "Amount of memory (in megabytes) on system
    APPEND-COMMUNITY-NAME:true
    FILE-NAME: prg_samples/eagent/memory"
    ::= { fsys 2 }
```

The agent will respond on behalf of `larry`, `curly`, and `moe` for the object `flintstones.fsys.fmemory`. The community name in the request indicates the system of interest. If the community name is `larry`, the amount of memory for 15.2.112.244 will be returned. If the community name is `curly`, the amount of memory for 15.2.114.237 will be returned, and if the community name is `moe`, the amount of memory for 15.2.113.223 will be returned.

The new field, `APPEND-COMMUNITY-NAME`, tells the agent to read the file named

`prg_samples/eagent/memory.communityName`

If the agent receives an SNMP GetRequest for `flintstones.fsys.fmemory.0` with community name `moe`, the agent reads `prg_samples/eagent/memory.moe` and returns `64` to the management station.

If the agent receives an SNMP GetRequest for `flintstones.fsys.fmemory.0` with community name `larry`, the agent reads `prg_samples/eagent/memory.larry` and returns `32` to the management station.

Using Proxy for Objects that are Built into the Agent

This proxy can be used to proxy for MIB-II objects or HP-UNIX objects.

For example, if you want to return different `sysDescr` values for the proxied systems `larry`, `curly` and `moe`, the following files would contain a value for `sysDescr`:

```
prg_samples/eagent/sysDescr.larry
prg_samples/eagent/sysDescr.curly
prg_samples/eagent/sysDescr.moe
```

To implement this proxy, you would define the following object:

```
internet        OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) }
enterprises     OBJECT IDENTIFIER ::= { internet private(4) 1 }
flintstones     OBJECT IDENTIFIER ::= { enterprises 4242 }
```

```
mgmt            OBJECT IDENTIFIER ::= { internet 2 }
mib-2           OBJECT IDENTIFIER ::= { mgmt 1 }
system          OBJECT IDENTIFIER ::= { mib-2 1 }

sysDescr        OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
            "A textual description of the entity.  This value
             should include the full name and version
             identification of the system's hardware type,
             software operating-system, and networking
             software.  It is mandatory that this only contain
             printable ASCII characters.
    APPEND-COMMUNITY-NAME: true
    FILE-NAME: prg_samples/eagent/sysDescr"
    ::= { system 1 }
```

If the agent receives an SNMP GetRequest for `system.sysDescr.0` with community name `moe`, the agent reads `prg_samples/eagent/sysDescr.moe` and returns the string contained in the file to the management station.

If the agent receives an SNMP GetRequest for `system.sysDescr.0` with community name `public`, the agent returns its internal value for `sysDescr`.

Writing Shell Commands

This section discusses the steps for writing your own shell commands and the conventions that you need to follow for the commands to work with SNMP. The shell commands can be either UNIX shell scripts or programs.

Here are the steps.

1. Log on to the agent system where you want the command to execute.
2. Write the script or the program.

   For an example, see the example script at the end of these steps, or `prg_samples/eagent/num_widgets` or `prg_samples/eagent/change_num_widgets`.

   Arguments

   By default, `extsubagt` does not pass any arguments to the *`read_command`* or *`file_command`*. If you want `extsubagt` to pass arguments, use the arguments listed below.

   By default, `extsubagt` passes an argument to the *`write_command`* -- the value you want to set the object to. For example, if you want to set the object value to 2, the agent passes that value to the command.

Optionally, if you want `extsubagt` to pass arguments to the *`read_command`*, *`write_command`* and *`file_command`*, use the following arguments. You can specify the arguments in any order.

Argument	Value passed in
`$i`	The management station's IP address. The address is in internet dot notation.
`$c`	The community name used in the request.
`$o`	The OBJECT IDENTIFIER used in the request. The OBJECT IDENTIFIER is in dot notation.
`$r`	The request issued by the management station. One of the following values will be passed: GetRequest, GetNextRequest, SetRequest, or PostSetRequest. The PostSetRequest will be passed to the *`file_command`* after the *`file_name`* has been created.
`$I`	The instance used in the request.
`$s`	The `SYNTAX` of the object. One of the following values will be passed: `INTEGER`, `OCTET STRING`, `OBJECT IDENTIFIER`, `NULL`, `NetworkAddress`, `IpAddress`, `Counter`, `Gauge`, `TimeTicks`, `Opaque`, `DisplayString`, or `PhysAddress`.
`$*`	This is the same as specifying `$i $c $o $s`.
`$$`	Substitute `$`.

You add these arguments when you specify the command in the `snmpd.extend` file. For example, if you want `extsubagt` to pass in the IP address of the management station, the community name, and the object ID when executing the `prg_samples/eagent/change_num_widgets` command, specify the following in the `DESCRIPTION` clause in the `snmpd.extend` file:

```
READ-COMMAND:prg_samples/eagent/change_num_widgets $i $c $o
```

When the command executes, `extsubagt` substitutes the `$` arguments with the real IP address of the management station, the community name, and the object ID.

Search path

HP recommends that you use the full path when specifying the command. However, this is not a requirement. You inherit the path of the calling process.

Return values

The return values for the *`read_command`* should be printed to standard out or standard error.

Execution

The *read_command*, *write_command*, and *file_command* are executed as if executed by /bin/sh.

Shell commands are supported. You can specify shell commands such as exit, read, if, and for in the *read_command*, *write_command*, and *file_command*. For a description of the shell commands, see the manpage for sh. Shell commands do not have a path. See the manpage for snmpd.extend for examples.

Exit codes

Make sure that the shell command exits with the correct exit code. An invalid exit code results in an error returned to the management station. The following tables are the errors returned to the management station for different exit codes:

read_command exit codes are:

Table 6-5

Exit Code	What it Means to the Management System
0	No error, that is, your command is successful. The data echoed to standard out or standard error will be returned to the management station in the SNMP reply. Only data necessary for the reply should be echoed to standard out or standard error. Too much data will return an error. The data echoed must be of the same SYNTAX as specified in the snmpd.extend file. If the data is not the same SYNTAX, an error will be returned to the management station.
Non-zero	noSuchName error, that is, the command is unsuccessful.

write_command exit codes are:

Table 6-6

Exit Code	What it Means to the Management System
0	No error, that is, your command is successful. No data should be echoed to standard out or standard error.

Table 6-6 (Continued)

Exit Code	What it Means to the Management System
Non-zero	General error (genErr), that is, the command is unsuccessful. If anything is echoed to standard out or standard error on the *write_command*, a general error is returned.

file_command should exit with 0, if not, a noSuchName error is returned to the management station.

Any errors the system encounters while trying to execute your shell command, are returned as noSuchName error for the READ-COMMAND and FILE-COMMAND and genErr for WRITE-COMMAND. The error is logged in the snmpd.log file.

3. Verify that the shell command executes successfully.

   To verify that your shell command executes successfully, execute the command. For example, to verify that the command associated with fmailListMsgs in the example snmpd.extend file executes successfully, type

   /usr/bin/mailq

   The command should return a list of mail messages on the mail queue.

4. Check the exit code by typing:

   echo $?

   If the value is 0, the shell command is successful.

5. If your shell command has arguments, verify the arguments.

   For example, assume you want to verify the num_widgets command shown in the "Sample Shell Command" section. The num_widgets command is defined in the example snmpd.extend file as:

```
DESCRIPTION "
READ-COMMAND: prg_samples/eagent/num_widgets $i $c $o $s"
```

   To verify the arguments, the command you type may look something like:

   num_widgets 15.2.3.149 public 1.3.6.1.4.4242.3.1 Gauge

The steps for writing your own shell commands are done.

Sample Shell Command

```
#! /bin/sh
## @(#) HP OpenView Extensible SNMP Agent Release 3.0
# num_widgets $Date: 94/01/19 15:35:33 $ $Revision: 3.3 $
#
# This shell script is an example for the SNMP extensible agent
# (snmpd.ea manpage).
#
# This script is called by the agent when a management station requests
# the object
#
# iso.org.dod.internet.private.enterprises.flintstones.fwidgets.
# fwidgetsPerHour.
#
# This script is registered in the snmpd.extend(4) file.
#
# This program will return the number of fwidgetsPerHour.  The number of
# widgets per hour is stored in /tmp/widgets_per_hour.
#
# A general error will be returned if
#    a) the request is made for an object identifier other than the one
#       listed above.
#    b) the request is made with the community string not equal to "public"
#    c) the request is made for an object with SYNTAX not equal to "Gauge"
#
WIDGETS_FILE=/tmp/widgets_per_hour

echo $* >$0.out 2> $0.err

case $2 in
       public) break;;
       secret) break;;
       *) echo "Invalid Community Name"; exit 5;
esac

case $3 in
       1.3.6.1.4.1.4242.3.1) break;;
       *) echo "Invalid Object Identifier"; exit 5;
esac

case $4 in
       Gauge) break;;
       *) echo "Invalid syntax"; exit 5;;
esac

if [ -r $WIDGETS_FILE ]
then
       cat $WIDGETS_FILE
else
       echo 3 >$WIDGETS_FILE
```

```
        cat $WIDGETS_FILE
fi
exit 0
```

7 Troubleshooting

This chapter focuses on troubleshooting the HP OpenView Extensible SNMP agent product. The following topics are discussed:

- Recommended practices for problem prevention, isolation, and recovery.

- Characterizing the problem.
- General product troubleshooting considerations.
- Troubleshooting by component.

General network troubleshooting is not discussed.

Recommended Practices

Following these recommended practices helps prevent and isolate problems, and recover from them:

- Ensure that the agent system meets the hardware, software, and configuration recommendations discussed in previous chapters.
- Do not modify HP OpenView SNMP agent product files, such as `snmpd.conf`, without retaining the original files. The original files provide a way of restoring a known good operational configuration. If you correct a problem by restoring the original files, you can isolate the problem to the changes you made to these files.
- Use logging of the agent background processes to help isolate problems, but be sure to clean up log files regularly. For information about logging, see the next page and check the manpage for `snmpd`.

CAUTION Logging is usually used by support personnel only. It can cause the log file to quickly grow very large (multiple megabytes in size). If you use logging, remember to monitor the size of the log file often, and to turn logging off as soon as you are finished debugging.

Logging Options

The logging options for the HP OpenView SNMP agent background process are described in Table 7-1. To set the initial log mask, use the following option:

`-m` *`logmask`*

The default log mask is 3, which means the HP OpenView SNMP agent logs authentication failure traps and errors.

Table 7-1 Command Line Options for SNMP Agent Logging

Task	Hexadecimal Value	String
Turn off logging.	0	`LOGGING_OFF`
Log errors.	0x10000000	`FACTORY_ERROR`

Table 7-1 Command Line Options for SNMP Agent Logging (Continued)

Task	Hexadecimal Value	String
Log trace messages.	0x00800000	FACTORY_TRACE
Log warning messages.	0x20000000	FACTORY_WARN

NOTE Log masks specify the type of output listed in `snmpd.log`. To select multiple output types, add the individual *`logmask`* values together and enter that number.

When the object you are trying to get returns an error, look in the `snmpd.log` to find out what the error is. If you have just defined new objects, look in the `snmpd.log` for syntax errors.

Characterizing the Problem

Symptoms are visible circumstances that indicate a problem. Whenever you encounter a symptom, collect the basic information as described in this section.

Scope: What is Affected?

Is This an Agent or Manager Problem?

A problem with the agent often shows up as a symptom of a problem on the manager system. When a manager depends on SNMP for data, the problem is usually with the agent. For example, if your manager provides information about a particular node on the network and that information is incorrect, the problem may be that the agent sends incorrect information. To isolate the problem, see “Troubleshooting by Component” on page 109.

Affected Parts of the HP OpenView SNMP Agent

What part of the HP OpenView SNMP agent is affected? All operations, or just some?

Is This a Master or Subagent Problem?

Any general protocol error is probably a problem with the master. However, you are much more likely to encounter a failed request for a MIB object — an operation handled by subagents. When a failed request occurs, try the following:

- Do any other MIB objects respond? A good one to try is `sysDescr.0`. This MIB object is supported by the master. If you receive a successful response, it is likely that the master is operating properly.

  Next, try querying other objects. Try some objects from the subagent that appears to be the problem, as well as other subagents. If only one subagent is failing, the problem is likely with that subagent.

- If you get no response from the agent at all, the problem is likely with the master or the configuration. To find out:

  First, confirm that the agent is running. Execute:

  ps -ef | grep snmpd

  You should see something similar to the following:

  ```
  root 18511 1 0 14:56:56 ?     0:00 /usr/sbin/snmpdm
  ```

If the master is *not* running, restart it by executing **snmpd**.

At this point, the master is up and responding. You probably have a configuration problem. Verify that the configuration file, including community strings, is correct. Then restart the agent.

- At this point, you know the master is up and responding, but the agent is still not responding properly. Contact Hewlett-Packard. Please have the following information ready at hand:

  1. A complete description of the problem. Include the type of hardware and operating system that is running. If possible, include a description of how to reproduce the problem.
  2. A copy of `/etc/SnmpAgent.d/snmpd.conf`
  3. A complete list of all subagents on the system
  4. The names and IP addresses of the management stations that interact with this agent.
  5. The output from a `what` command run on:

     ```
     /usr/sbin/snmpdm
     /usr/sbin/hp_unixagt
     /usr/sbin/mib2agt
     ```

     and all other subagents.
  6. If the problem can be reproduced, kill the master.

Context: What Changed?

Determine what may have changed on your network or product configuration: hardware, software, files, security, utilization, and so forth.

Duration: How Long or How Often?

Is the problem consistent (fails every time) or intermittent (fails sometimes)?

Context: What Action Was Performed?

When the problem occurred, what was happening? For instance,

- What operation was selected?
- What command was executed?

- What data was requested or sent?

General Product Troubleshooting Considerations

When troubleshooting the HP OpenView Extensible SNMP agent product, consider the fact that SNMP is based on User Datagram Protocol. UDP is an unreliable protocol (no error checking and no guarantee of message receipt). This may cause occasional failures of manager-agent communication.

When You Need More Information

Other information that may assist your troubleshooting of the HP OpenView Extensible SNMP agent product is available, as follows:

- Chapter 1, “Introduction and Operational Concepts.”
- Appendix A, “Supported MIB Objects.”
- Manpages.

If a problem does not appear to be with the HP OpenView SNMP agent itself, refer to the following:

- Your networking documentation for network troubleshooting procedures.
- Your system documentation for system troubleshooting procedures.

Troubleshooting by Component

This section suggests actions to take if you suspect a problem with a component of the agent system. This section also discusses the sequence of interactions and/or the flow of data associated with a component's operation when such information may help you to isolate the problem. This section covers the following components of the network management system:

- Runtime components
- SNMP subsystem
- Agent MIB

For detailed information on a command or process mentioned in this section, see its associated manpage.

Runtime Components

If you are having problems running the HP OpenView SNMP agent, check software versions, file permissions, and start-up scripts. Make sure the communication strings match on the manager and agent.

NOTE No duplicate community strings are allowed in the `snmpd.conf` file. Entering one results in an error message in the log file. Also, the `set` community string is valid for both `get` and `set` operations. There is no need to configure a `get` community string separately.

Agent File Permissions

By default, the HP OpenView SNMP agent (`snmpd` or `snmpdm`) is executable only by root.

Startup Scripts

Check to see if some component in the execution sequence is “broken,” such as a syntax error in one of the product’s startup scripts. The start-up script for each platform during invocation of the HP OpenView SNMP agent is listed in Chapter 2, “Before You Install.”

SNMP Subsystem

Use these methods to troubleshoot the SNMP subsystem:

- Verify that community names are correctly configured on both the manager and agent system. See Chapter 4, “Configuring the Master SNMP Agent,” for more information on this topic.
- Use the command to verify operation of the SNMP agent on an HP OpenView SNMP agent system. See the *snmpget* manpage.
- Use the `snmpget` command to verify operation of the SNMP agent on an HP OpenView SNMP agent system.
- Check the `snmpd.log` file on an HP OpenView SNMP agent for errors (if you have logging turned on).
- Verify that the agent system’s trap destination is set properly. See Chapter 4, “Configuring the Master SNMP Agent,” for more information.
- If you use HP OpenView Network Node Manager, you can also use the `Fault:Test IP/TCP/SNMP` operation to verify SNMP operation on a remote SNMP node.
- Enable all logging features so more detail is seen. (See the logging list in the *snmpd* manpage.)
- Ensure you have the right agent. If you are running the right agent, ensure the configuration has been internalized by “kill;” then restart the SNMP agent.

Agent MIB

Use these methods to troubleshoot the agent MIB.

- Make sure that the agent and the manager can communicate. The problem may be with the network configuration on the agent system. To test network connectivity, execute the `ping` command.
- Use an SNMP `get` command from the manager system to inspect an individual agent MIB value. If you use HP OpenView Network Node Manager, the command is `snmpget`.
- Use an SNMP `walk` command from the manager system to dump part or all of the agent’s MIB for inspection. If you use HP OpenView Network Node Manager, the command is `snmpwalk`.
- Check that the object ID configured on the agent system is the same as the object ID configured on the manager system.
- If you are trying to do an SNMP SetRequest, verify that the agent is configured to respond to SNMP SetRequests. (By default, the agent does not allow managers to alter MIB values.)

To configure the agent to respond to SNMP SetRequests, add a `set-community-name` to the agent's `snmpd.conf` file. If you use HP OpenView Network Node Manager, see *snmpconf* (4) to configure the manager.

- If you use the HP OpenView Network Node Manager product, you can also use the `Browse MIB` operation to verify that the information retrieved is accurate.

Troubleshooting the snmpd.extend File

To troubleshoot the `snmpd.extend` file, first troubleshoot locally, then troubleshoot over the network.

Use the following methods to troubleshoot the `snmpd.extend` file.

Locally

- Execute each command in the `snmpd.extend` file by itself from the operating system command line to see if the command returns the expected response.
- Verify that the command executed correctly by typing:

  `echo $?`
- If the command has arguments, verify the arguments. To do so, execute the commands independently supplying all necessary parameters.
- Check the `snmpd.log` for syntax errors.
- If the agent finds an error when reading in the `snmpd.extend` file, the agent will display the line where the error occurred and the correct syntax.
- Check that the command defined in the `snmpd.extend` file is executable.
- Verify that execute permission is set on the command.
- Check that the object ID on which you are trying to get information executes the proper command. To turn on maximum logging start `extsubagt` with the `APALL` logging option:

  `/usr/sbin/extsubagt -m APALL`
- Check that the output for the command you are executing corresponds to the proper data type.
- Check that the command in the `snmpd.extend` file is spelled correctly. For example, a common error is to type `/usr` as `/user`.

From the Manager

If, after troubleshooting the problem on the agent system you still have a problem, check the following:

- Do an SNMP request to each object in the `snmpd.extend` file from the management station to make sure the file works correctly.

- After you have done an SNMP SetRequest, do an SNMP GetRequest to verify that the value was set correctly.

A Supported MIB Objects

This appendix lists the MIB objects supported by the HP OpenView Extensible SNMP Agent. Included in the list are:

- Standard MIB-II objects which are implemented by the HP OpenView SNMP MIB-II subagent. This MIB definition is in file `rfc1213-MIB-II` beneath the `snmp_mibs` directory on the HP OpenView manager system.
- Hewlett-Packard's enterprise-specific hp-unix MIB objects which are implemented by the HP UNIX subagent. This MIB definition is in file `hp-unix` beneath the directory `snmp_mibs` on the HP OpenView manager system.

Standard MIB-II Objects Supported by the MIB2 Subagent

The HP OpenView Extensible SNMP Agent running on your HP OpenView network management system supports all standard MIB-II objects, except for the EGP group. Refer to *RFC 1213: Management Information Base for Network Management of TCP/IP-based internets: MIB-II* for details.

This section lists the standard MIB-II objects that:

- The HP OpenView Extensible SNMP Agent allows you to change.
- Return the value 0 (zero).
- Return `noSuchName` errors (Solaris only).

Refer to *RFC 1213: Management Information Base for Network Management of TCP/IP-based internets: MIB-II* for details on the MIB-II objects. Solaris supports all of MIB-II.

Objects That Agents Allow You to Change

The following list shows the objects that the HP OpenView Extensible SNMP Agent allows you to change through an SNMP SetRequest.

NOTE Before you can change agent MIB values, you must configure the agent to respond to SNMP SetRequests. To do so, add a `set-community-name` to the `snmpd.conf` file on the agent.

```
sysContact
sysName
sysLocation
ifAdminStatus
atPhysAddress
ipRouteNextHop
ipRouteType
ipRouteAge
ipNetToMediaPhysAddress
ipNetToMediaTypesysName
snmpEnableAuthTraps
```

Objects That Return Null Values (Solaris only)

```
ifInNUcastPkts
ifInDiscards
ifOutNUcastPkts
ifOutDiscards
```

Objects That Return noSuchName Errors (Solaris only)

```
ifLastChange
ifInOctets
ifInUnknownProtos
ifOutOctets

ipInReceives
ipInAddrErrors
ipForwDatagrams
ipInUnknownProtos
ipInDiscards
ipInDelivers
ipOutRequests
ipOutDiscards
ipOutNoRoutes
ipReasmTimeout
ipReasmReqds
ipReasmOKs
ipReasmFails
ipFragOKs
ipFragFails
ipFragCreates

ipAdEntReasmMaxSize

ipRouteAge

ipRoutingDiscards

tcpActiveOpens
tcpPassiveOpens
tcpAttemptFails
tcpEstabResets
tcpInSegs
tcpOutSegs
tcpRetransSegs
tcpInErrs
tcpOutRsts
```

```
udpInDatagrams
udpNoPorts
udpOutDatagrams

egp group
```

MIBs Supported by the HP UNIX Subagent

This section describes which MIB objects are supported by the HP UNIX subagent (`hp_unixagt`). This MIB resides in the file `hp-unix` on the HP OpenView network management system.

HP-UX:

All objects described in `hp-unix` are implemented on HP-UX.

Solaris:

```
nm.snmp.snmpdConf.snmpdConfRespond
nm.snmp.snmpdConf.snmpdFlag
nm.snmp.snmpdConf.snmpdLogMask
nm.snmp.snmpdConf.snmpdReConfigure
nm.snmp.snmpdConf.snmpdSize
nm.snmp.snmpdConf.snmpdStatus
nm.snmp.snmpdConf.snmpdVersion
nm.snmp.snmpdConf.snmpdWhatString
nm.system.general.computerSystem.computerSystemAvgJobs1
nm.system.general.computerSystem.computerSystemAvgJobs15
nm.system.general.computerSystem.computerSystemAvgJobs5
nm.system.general.computerSystem.computerSystemUpTime
nm.system.general.computerSystem.computerSystemUsers
nm.system.general.fileSystem.fileSystemMounted
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemBavail
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemBfree
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemBlock
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemBsize
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemFfree
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemFiles
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemID1
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemID2
general.fileSystem.fileSystemTable.fileSystemEntry.fileSystemName
```

Format of Definitions

The next section contains the specification of all HP object types contained in the MIB. Following the conventions of the *RFC 1212*, the object types are defined using the following fields:

OBJECT-TYPE A textual name, termed the OBJECT DESCRIPTOR, for the object type.

SYNTAX	The abstract syntax for the object type, presented using ASN.1. This must resolve to an instance of the ASN.1 type ObjectSyntax defined in the Structure of Management Information (SMI). SMI identifies the rules used to define the objects that can be accessed through a network management protocol.
ACCESS	A keyword, one of read-only, read-write, write-only, or not-accessible.
STATUS	A field describing the status of the object type.
DESCRIPTION	textual description of the semantics of the object type. Implementations should ensure that their interpretation of the object type fulfills this definition since this MIB is intended for use in multivendor environments. As such it is vital that object types have consistent meaning across all machines.
::=	The OBJECT IDENTIFIER corresponding to the object type.

B Platform Equivalents

Table C-1 lists the directory paths used in this guide and their equivalents on each supported platform. It also provides the name for the `snmp_mib` file.

File Path Names

Table B-1 File and Directory Paths

Platform	Agent Path	`snmp.log` Path	`snmpd.conf` Path
HP-UX 11.X	`/usr/sbin/snmpd`	`/var/adm/snmpd.log`	`/etc/SnmpAgent.d/snmpd.conf`
Solaris 2.8 or later	`/usr/sbin/snmpd`	`/var/adm/snmpd.log`	`/etc/SnmpAgent.d/snmpd.conf`

Table B-2 File and Directory Paths

Platform	MIB Path	`prg_samples` Path	Startup Config Path
HP-UX 11.X	`/var/opt/OV/share/snmp_mibs/*`	`/opt/OV/prg_samples`	`/etc/rc.config.d/SnmpMaster`
Solaris 2.8 or later	`/var/opt/OV/share/snmp_mibs/*`	`/opt/OV/prg_samples`	`/etc/rc.config.d/snmp*`

Glossary

A

Agent 1- A process running on a device that responds to SNMP requests and sends SNMP traps.
2 - A device that responds to SNMP requests and sends SNMP traps.

Agent system A device, such as a host, gateway, terminal server, hub, or bridge, that has an agent responsible for performing the network management operations requested by the manager.

C

Community name A password that allows a manager to access MIB values on an agent.

E

Enterprise-specific MIB MIBs developed by individual vendors for their specific product lines. Vendors register their private MIBs under the enterprises object identifier subtree.

Extensible Agent An agent that has been configured to support any MIB object defined on the agent.

G

get An action that enables the manager to retrieve management information from an agent.

H

HP OpenView Extensible SNMP Agent

An agent that extends the existing Management Information Base (MIB) to support new objects.

HP OpenView Network Node Manager

An application that provides fault, configuration, and performance management of multivendor TCP/IP networks.

L

Leaf node A node in the MIB tree that does not have “children.” The leaf node is the actual object.

M

Manager A system that is executing network management software.

Manager system A system that executes network management operations and control agent systems.

MIB Management Information Base. A collection of objects (management information) that can be accessed through a network management protocol.

MIB module A file defining all the MIB objects under a subtree. For example, the Internet-standard MIB-II and the HP enterprise-specific MIB are MIB modules.

MIB object Managed object defined according to the RFC 1155: Structure and Identification of Management Information for TCP/IP-based Internets or RFC 1212: Concise MIB Definitions.

MIB tree A concept used to illustrate the organization of MIB objects.

MIB variable A pairing of a MIB object and associated MIB value or values.

N

Node A branch of the MIB tree.

Network A group of data communication objects, with varying degrees of intelligence, that are interconnected through a common transmission medium. A network has both logical and physical characteristics.

O

Object ID The name used to uniquely identify a MIB object. The object ID is based on an object's place in the MIB tree.

S

set An action that enables the manager to alter management information on an agent.

Simple Network Management Protocol

The protocol used to retrieve network information from nodes; defined by RFC 1157: A Simple Network Management Protocol (SNMP).

SNMP See the Simple Network Management Protocol.

SNMP table A table defined using the SYNTAX SEQUENCE OF and the INDEX clause.

snmpd The background process on the extensible agent system that processes requests from the manager.

snmpd.extend The default MIB module that extends the MIB on the agent to include new user-defined objects.

Subtree All nodes and children under a branch of the MIB tree.

T

Trap Information sent from a node that supports SNMP (an agent) to the manager without an explicit request from the manager. Traps inform the manager of changes that occur on these nodes (for example, reboot).

Index

Index

W