HP OpenView Extensible SNMP Agent Administrator's Reference

Sun Systems

Manual Part Number: B1038-90002
Printed in U.S.A., February 1994

Notice

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

The specific warranty information for the product as to which this material relates is available in Hewlett-Packard's purchasing and licensing terms and conditions.

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

Hewlett-Packard Company
3404 East Harmony Road
Fort Collins, CO 80525 U.S.A.

Printing History

First Edition February 1994

Contents

1

Introduction and Operational Concepts

This manual is for the person who installs configures, administers, and troubleshoots the HP OpenView Extensible SNMP Agent.

This manual assumes that you are a knowledgeable Sun system and network administrator and troubleshooter. You should know how to do the following:

- Update your system with new software.
- Kill processes.
- Write shell scripts.
- View, search, and edit files.

In addition, you also need to understand SNMP MIBs and traps.

Documentation Guide and References

Note Throughout this manual, references to SunOS refer to SunOS 4.1.x. References to Solaris refer to SunOS 5.2 and SunOS 5.3.

The man pages for system administration commands are contained in section 8 on SunOS systems, but in section 1M on Solaris systems. Throughout this manual, man pages for system administration commands are denoted as ***command*** **(1M|8)**; for example, **snmpd(1M|8)**.

Sun System Administration Manuals

- *SunOS System and Network Administration*
- *SunOS Reference Manual volumes 1, 2, and 3*
- *Solaris System Configuration and Installation Guide* (for Solaris only)

Network Management Manuals

Refer to the Administrator's Reference manual that was shipped with your HP OpenView product.

TCP/IP and SNMP Concepts

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

Rose, Marshall T. *The Simple Book: An Introduction to Management of TCP/IP-based internets*. Englewood Cliffs, New Jersey: Prentice-Hall, 1989. To order, reference ISBN 0-13-812611-9.

Stallings, William *SNMP, SNMPv2, and CMIP: The Practical Guide to Network-Management Standards*. Addison-Wesley Publishing Company Inc, 1993. To order, reference ISBN 0-201-63331-0.

Internet Request for Comments Documents

Copies of these documents are shipped on the HP OpenView Extensible SNMP Agent product installation media. They are in the **EAGENT-MAN** fileset and are installed in the **/usr/OV/doc** directory.

To find out about...	See...	Filename
how MIB objects are defined	RFC1155: Structure and Identification of Management Information for TCP/IP-based Internets	**rfc1155.txt**
the protocol used to manage the MIB objects	RFC1157: A Simple Network Management Protocol (SNMP)	**rfc1157.txt**
the format for creating MIB object files	RFC1212: Concise MIB Definitions	**rfc1212.txt**
the objects contained in the Internet-standard MIB	RFC1213: Management Information Base for Network Management of TCP/IP-based Internets: MIB-II	**rfc1213.txt**

Definitions and Concepts

This section describes the definitions and the concepts that you need to successfully configure and manage your HP OpenView Extensible 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 and system management operations which monitor and control agent systems. The implementation of these network and system 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 **agent** responsible for performing the network and system management operations requested by the manager.

A manager and agent may exist on the same system.

The Simple Network Management Protocol (**SNMP**) communicates management information between a manager and an agent. SNMP allows

- 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 information requested.
- 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 system to send traps to through a configurable trap destination.

Requests for information on an agent are accompanied by a **community name**, which is a password that allows the manager access to that information.

MIBs

Conceptually, the information on the agent is known as the Management Information Base (**MIB**). The MIB is not a physically distinct database, but rather, a concept that encompasses configuration and status values normally available on the agent system. MIB values conform to an Internet-standard structure of management information and compose a virtual data store on the agent system. This structure is defined by *RFC 1155: Structure and Identification of Management Information for TCP/IP-based Internets.* Agents contain the “intelligence” required to access MIB values.

The MIB objects are defined using the Internet-standard structure of management information (SMI) and compose a virtual data store on the agent system. This structure is defined by *RFC 1155: Structure and Identification of Management Information for TCP/IP-based Internets* and amended by *RFC 1212: Concise MIB Definitions*. Together RFCs 1155 and 1212 define the structure of management information for SNMP-based management. SNMP agents contain the "intelligence" required to access MIB values.

MIBs are organized into **MIB modules**. A MIB module is a file defining all the MIB objects under a subtree. The foundation module is the standards-based MIB-II module defined by *RFC 1213: 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. The MIBs defined by these various hardware vendors are referred to as enterprise-specific MIBs in this manual.

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 just described.

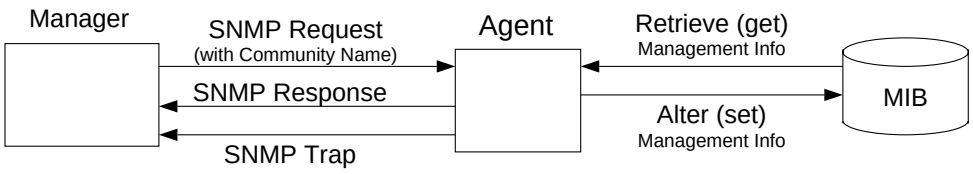


Figure 1-1. Manager-Agent Communication through SNMP

How MIBs are Organized

Conceptually the **MIB objects** are organized in a hierarchical tree structure. Figure 1-2 illustrates the tree structure. The numbers in parenthesis are the numerical representation of the nodes.

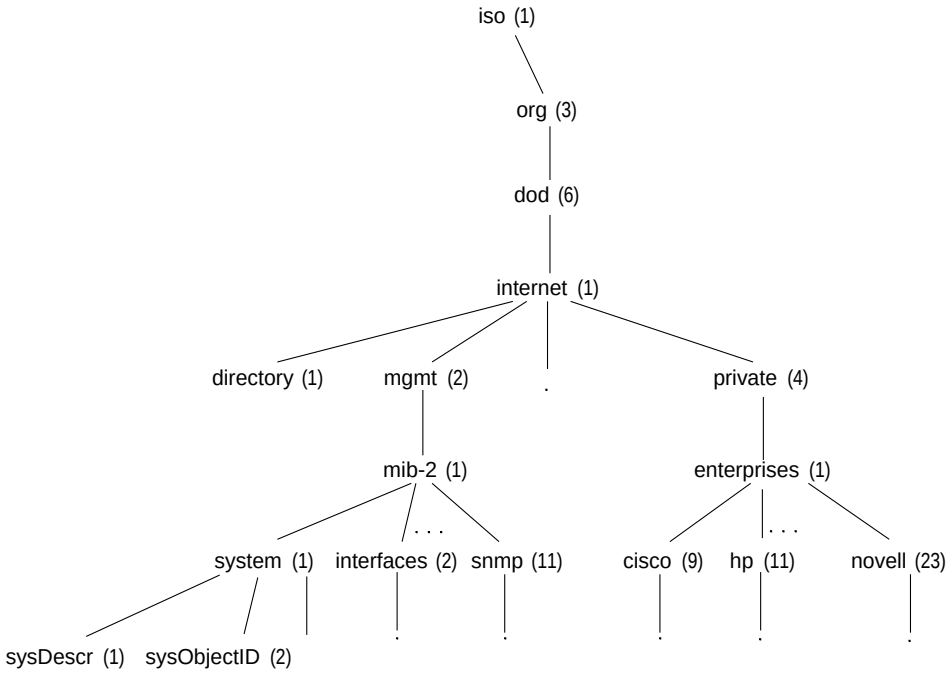


Figure 1-2. MIB Organization

Each branch of the tree consists of logical groupings used to generate unique object IDs. 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 its place in the tree. 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`** subtree is **`iso.org.dod.internet.mgmt.mib`** 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 MIBs are registered under the **`enterprises`** subtree. The Internet-standard MIB-II is registered under the **`mib`** subtree. The **`mib`** subtree is primarily used to manage TCP/IP-based networks through SNMP.

Traps

A trap is a message sent from a remote system (an agent) to a manager without an explicit request from the manager. Agents send traps to the manager to indicate that a particular condition exists on the agent system such as an error has occurred or an event has happened.

The traps are defined in *RFC 1157: A Simple Network Management Protocol (SNMP)*. Each SNMP trap contains a generic trap number and a specific trap number. The generic trap numbers range from 0 to 6.

ColdStart(0)
WarmStart(1)
LinkDown(2)
LinkUp(3)
AuthenticationFailure(4)
EgpNeighborLoss(5)
EnterpriseSpecific(6)

The generic traps 0 through 5 are defined by SNMP. You can define your own trap by combining the generic enterprise-specific trap 6 with your own specific trap number.

Information the Manager Can Receive

Typically, the information the manager can receive from the agent is limited to the set of MIB objects defined on the agent. Using the tandard HP OpenView SNMP Agent you can get information about Internet-standard MIB-II objects defined in RFC 1213 and HP enterprise-specific MIB objects. However, sometimes you have information on a device that you cannot get through these predefined MIB objects. For example, you may want to know if

- An LP spooler is up or down.
- A certain piece of software is still running or not.

The HP OpenView Extensible SNMP Agent enables you to retrieve this type of information.

The HP OpenView Extensible SNMP Agent

The **HP OpenView Extensible SNMP Agent** is a superset of the HP OpenView SNMP Agent. The HP OpenView Extensible SNMP Agent still supports MIB-II and HP enterprise-specific objects *plus* it supports any MIB objects you add to the agent -- you are no longer limited to just retrieve management information about a given set of objects defined by the various hardware vendors and standards organizations. Using the HP OpenView Extensible SNMP agent, you can

- Add your own objects to the agent.
- Configure the agent to notify the manager when something is wrong on the agent.

You add your own objects by defining the objects 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. You can add MIB objects using either a command or a file.

For each object you define, you must either specify 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. It is by executing the specified command that the **extensible agent** can respond to the SNMP request. 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.

Applying this concept, you can, for example, create a new MIB module that defines objects that will help you

- Retrieve client data not available from your current standard agent through an SNMP GetRequest. For example, you can
  - 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 can change the number of parts produced per hour.

You configure the agent 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. You can use the **`snmptrap`** 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.

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 do system management, peripheral management, and application management. It is a tool that enables you, through SNMP, to extend your agent to essentially get or set any MIB objects associated with a system, device, or application. The objects you manage do not have to be TCP/IP devices running SNMP as long as the agent system can communicate with the device.

Figure 1-3 illustrates the extensible agent concepts just described. For example, if a critical software process goes down, the agent can send a trap to the manager indicating the process went down. You can then use the set capability from the manager to re-start the process.

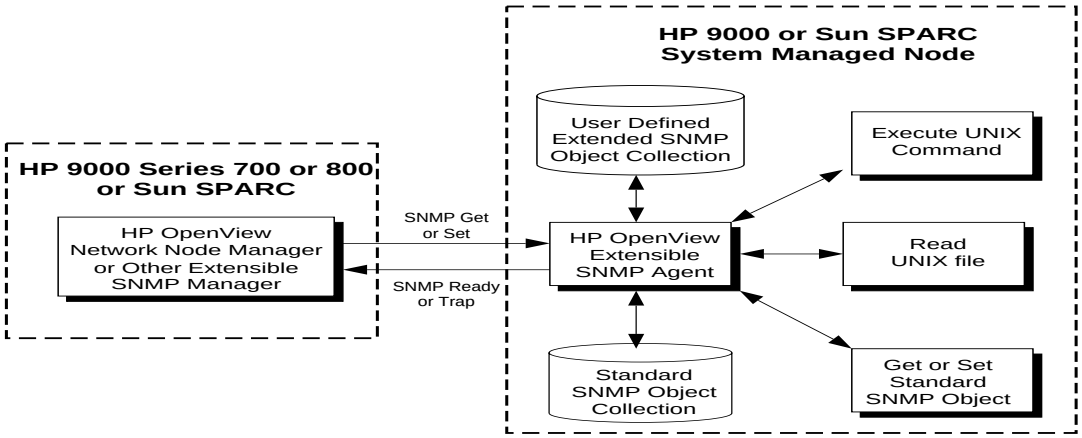


Figure 1-3. Extensible SNMP Agent Concepts

The HP OpenView Network Node Manager

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

Note While this manual 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 HP OpenView Extensible SNMP 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 agent.
- Manage critical software processes that are running on unattended systems in dispersed locations.

Processes and Files

This section defines the processes and files that are key to the operation of the HP OpenView Extensible SNMP Agent. Included are explanations of the interactions and relationships among these processes and files. 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 Extensible SNMP Agent is **`/etc/snmpd.ea`**, the extensible SNMP daemon. The **`/etc/netnmrc`** (for SunOS) (**`/etc/init.d/netmgt`** for Solaris) script automatically starts the **`snmpd.ea`** background process. This process should run on the agent system at all times to respond to requests from the manager system. To ensure that this process is invoked, the **`/etc/rc.local`** script starts the **`netnmrc`** script during system reboot. (For Solaris, the **`/etc/rc.local`** script does not start the **`netmgt`** script.)

- On invocation, **`snmpd.ea`** reads the following files to obtain its configuration: **`/etc/snmpd.conf`**

  The **`/etc/snmpd.conf`** file contains the agent's trap destinations, community names, and certain MIB values.

- **`/etc/snmpd.extend`**

  The **`/etc/snmpd.extend`** file defines the extended agent MIB objects.

Operational Behavior

Once the HP OpenView Extensible SNMP Agent is operational, the **snmpd.ea** background process continues to run. When the manager sends an SNMP request to the agent, **snmpd.ea** processes the request, takes the appropriate action, such as executing a command or reading a file, and sends an SNMP reply to the manager.

Agents such as **snmpd.ea** 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.

snmpd.ea (agent) errors are logged to the **/usr/adm/snmpd.log** (for SunOS) file (**/var/adm/snmpd.log** for Solaris).

Figure 1-4 shows the operational interactions and relationships among these processes and files.

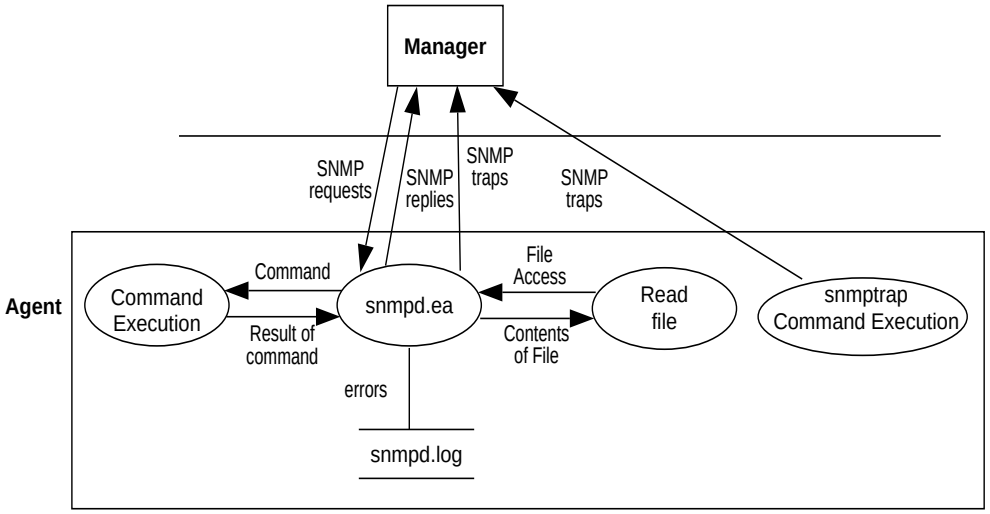


Figure 1-4. Agent Processes and Files During Operation

2

Installing Verifying, Stopping, and Restarting the Agent Software

This chapter gives hardware and software requirements for the HP OpenView Extensible SNMP Agent. This chapter also explains how to

- Install the HP OpenView SNMP Agent.
- Install the HP OpenView SNMP Agent remotely.
- Verify that the agent software is running correctly.
- Stop and restart the agent software.

Prerequisite Extensible Agent Hardware and Software

The following software must be installed, configured, and running properly prior to installing the HP OpenView Extensible SNMP Agent:

- SPARCstation 2 running SunOS 4.1.x
- SPARCstation 10 running SunOS 4.1.3 or Solaris 2.2 or 2.3
- SPARCstation IPX running SunOS 4.1.x
- SPARCclassic running SunOS 4.1.3 or Solaris 2.2 or 2.3
- SPARCstation LX running 4.1.3 or Solaris 2.2 or 2.3
- Minimum of 2 megabytes free local disk space (DO NOT USE NFS)
- SunOS System V Optional Software (SunOS only)
- SunOS Networking Services (Ethernet and TCP/IP) (SunOS only)

Note QIC-24 compatible tape drive or CD-ROM is required for installation with SunOS. CD-ROM is required for installation with Solaris.

Installing the Agent Software

This section describes how to install the HP OpenView Extensible SNMP Agent software. There are three steps to complete:

- Step 1: Checking disk space availability.
- Step 2: Installing the OVIC fileset (which contains the **`ovinstall`** utility that helps install the individual HP OpenView products on your system). This step has two procedures: one to use if your product is on tape media, and the other if your product is on CD-ROM media.
- Step 3: Installing the HP OpenView Extensible SNMP Agent product.

Installation Step 1: Checking Disk Space Availability

1. Log in as **root**.

2. Verify that you have at least 2 megabytes of free disk space in the correct directory:

   For SunOS -- **/etc**
   For Solaris -- **/usr/sbin**

   To check your disk space, issue the following command:

   ```
   df
   ```

3. Create a directory named **OV** to hold the **OVIC** fileset by executing:

   ```
   mkdir /extra/OV
   ```

   where ***/extra*** is the name of the file system that has sufficient free disk space.

4. Make **/usr/OV** a symbolic link to ***/extra*****/OV** by executing:

   ```
   ln  -s  /extra/OV   /usr/OV
   ```

   where ***/extra*** is the name of the directory on which the file system with sufficient space is mounted.

Note The next section contains two procedures:
-- "Tape Media Installation (SunOS Only)"
-- "CD-ROM Media Installation"

Go directly to the OVIC installation procedure for the type of media on which your product was shipped.

Installation Step 2: Installing the OVIC Fileset

This section has two OVIC installation procedures: one for products shipped on tape media and the other for products shipped on CD-ROM media. Go directly to the procedure that matches the type of media you have.

Tape Media Installation (SunOS Only)

You will use the **`extract_unbundled`** program to load the HP OpenView installation utilities fileset, **`OVIC`**. Do the following steps:

1. Insert the product media into the appropriate drive.

2. Install the **`OVIC`** fileset (which contains the **`ovinstall`** program) by executing **`extract_unbundled`**. (See your SunOS system administration manual for prerequisites and instructions on using the **`extract_unbundled`** program.)

3. Follow the instructions given by **`extract_unbundled`**. You must supply the device name of the tape drive (for example, **`st0`**)

   `extract_unbundled` will then execute, listing all the OVIC files as they are being installed.

For information about OVIC's features, skip to the "What OVIC Does" section.

CD-ROM Media Installation

Do the following steps:

1. Mount the CD-ROM drive by doing the following steps:

   a. If necessary, create a directory called

      ```
      /cdrom
      ```

   b. Mount the CD-ROM drive by typing

      ```
      /usr/etc/mount  -rt  hsfs /dev/sr0  /cdrom
      ```

   c. Load the OVIC fileset by typing

      ```
      /cdrom/INSTALL  -d  /cdrom/OPENVIEW
      ```

What OVIC Does

The **`OVIC`** fileset you just installed contains the **`ovinstall`** utility. By default, **`ovinstall`**:

- Installs all the filesets and does consistency checking to make sure that the HP OpenView filesets are compatible with any existing HP OpenView products that may already be installed.
- Starts up the daemon.

Note The next step, "Installation Step 3: Installing the HP OpenView Extensible SNMP Agent Product," documents the procedure for using **`ovinstall`** to perform a default installation. This default is recommended for almost all installations.

Installation Step 3: Installing the HP OpenView Extensible SNMP Agent Product

To install the HP OpenVIew SNMP Agent, you will use the **ovinstall** command. You will need to include a **-d *device_filename*** argument in the **ovinstall** command. The syntax for the **ovinstall** command appears below:

/usr/OV/bin/ovinstall -p *component* -- -d *device_filename*

-p *component* the component you are installing (that is, **EAGENT**)

-d *device_filename* the source of the installation media you are copying the product software from

The following table shows examples for using **ovinstall** with several different installation media:

To Install From:	Syntax/Example
1/4" Cartridge Tape	Syntax: **/usr/OV/bin/ovinstall -p *component* -- -d *device_filename*** Example: **/usr/OV/bin/ovinstall -p EAGENT -- -d /dev/rst0**
CD-ROM	Syntax: **/usr/OV/bin/ovinstall -p *component* -- -d *device_filename*** Example: **/usr/OV/bin/ovinstall -p EAGENT -- -d /dev/cdrom/OPENVIEW** Note: When using CD-ROM, the ***device_filename*** must be the file **OPENVIEW** under the directory at which the CD-ROM is mounted (the location you specified in "Installation Step 2: Installing the OVIC Fileset").

The ``**--**" signifies the end of **ovinstall** options. When **ovinstall** later calls **ovupdate** to perform the actual software installation, all arguments following the ``**--**" are passed as command line arguments to **ovupdate**.

Installation Procedure

1. Insert the product media in the appropriate drive.

2. Install the product, using the **`ovinstall`** command.

   If **`ovinstall`** executes

   - With no errors, it will output the message **`NOTE: Installation completed successfully`** to standard output and to **`/tmp/update.log`**. The **`ovinstall`** command then calls the **`ovconfigure`** program to do the configuration of the component's filesets. Skip to **step 4**.
   - With errors, it will output messages describing the problems to standard output and to **`/tmp/update.log`**. Go on to **step 3**.

3. Read the file **`/tmp/update.log`** for messages about the results of the installation or any installation or configuration errors. Type

   ```
   more /tmp/update.log
   ```

   - If there were installation errors, you should fix the problems and then run the **`ovinstall`** command again to re-install the software. After **`ovinstall`** successfully installs the product, it will call the **`ovconfigure`** program to do the configuration.
   - If there were configuration errors, you should fix those problems and then run the **`ovconfigure`** command again. (Since you have already successfully installed the software, do not re-run **`ovinstall`** before running **`ovconfigure`** the second time.) Specify the **`ovconfigure`** command as shown below:

     ```
     /usr/OV/bin/ovconfigure -p EAGENT
     ```

   - If there were no errors at all, go on to **step 4.**

4. If you are using NIS to serve the services database, add the following entry to the NIS database.

   ```
   snmp        161/udp     #  Simple Network Management Protocol
   ```

   This entry is automatically added to the local **`/etc/services`** file, but for the NIS server to know about these entries, the entry must also be added to the NIS database.

5. The default installation/configuration is done. (If this default installation does not meet your needs, you can modify it later. See chapter 3 for more information.)

Note It is highly recommended that you verify the installation of the HP OpenView SNMP Agent software. See the "Verifying Agent Software" section for this procedure.

Installing the Agent Remotely

If you do not have a local tape of CD-ROM drive to use with the installation media that came with your product, you can do an installation from a remote drive.

This section contains procedures for installing HP OpenView products remotely:

- On a SunOS system, from a remote tape drive
- On a SunOS or Solaris system, from a remote CD-ROM drive

On a SunOS System from a Remote Tape Drive

The remote system with the tape drive must be a SunOS or Solaris system.

Note You must be super-user to do the following installation procedure.

1. Arrange that root on the agent system has **`rsh`** access to the system that has the tape drive. Do this by editing root's **`.rhosts`** file on the system with the tape drive. (See the **`hosts.equiv(4)`** man page for more information.)

2. On the agent station, run **`/usr/etc/extract_unbundled`**, as described below in **steps a-d**. You will interactively supply the host name of the remote system that has the tape drive, and the name of the tape drive device file on the remote system.

   For example, **`fred`** is the remote system's hostname and **`/dev/rst0`** is the remote system's tape drive device file name.

   a. Type:

      `/usr/etc/extract_unbundled`

   b. At the **`Enter media drive location [local | remote]:`** prompt, type:

      `remote`

   c. At the **`Enter hostname of remote drive:`** prompt, type:

      `fred`

d. At the **`Enter Device Name (e.g., rst0, rmt0, rfd0c):/dev/r`** prompt, type:

```
st0
```

This will install the HP OpenView installation software, **`OVIC`**, on the management station.

3. Proceed to install the HP OpenView Extensible SNMP Agent product, indicating the remote host and remote tape drive with arguments following the **`--`** argument in the **`ovinstall`** command.

For example, to use the system **`fred`** from **step 2** above, type:

```
/usr/OV/bin/ovinstall -p EAGENT -- -r fred -d /dev/rst0
```

4. After installation has been completed, if you wish, you can eliminate the **`rsh`** access you set up in **step 1** above.

On a SunOS or Solaris System from a Remote CD-ROM Drive

The remote system with the CD-ROM drive can be an HP-UX, SunOS, or Solaris system.

Note You must be super-user to do the following installation procedure.

---

1. Insert the product CD-ROM in the CD-ROM drive.

2. Mount the CD-ROM on the system with the CD-ROM drive (for example, at **`/cdrom`**).

– On HP-UX (a typical CD-ROM drive device file name might be **`/dev/dsk/c201d2s0`**), type:

```
mkdir /cdrom
mount -rt cdfs /dev/dsk/c201d2s0 /cdrom
```

– On SunOS or Solaris (a typical CD-ROM drive device file name might be **`/dev/sr0`**), type:

```
mkdir /cdrom
mount -rt hsfs /dev/sr0 /cdrom
```

3. On the system with the CD-ROM drive, export the CD-ROM file system so that the management station can NFS mount it. For example, **`marvin`** is the name of the management station.

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

```
/cdrom -ro marvin
```

b. Export the file system with the command

```
/usr/etc/exportfs /cdrom
```

4. On the management station, NFS mount the CD-ROM directory (at **`/cdrom`**, for example). Type:

```
mkdir  /cdrom
mount  fred:/cdrom  /cdrom
```

5. On the management station, proceed with the installation as if the CD-ROM drive were locally mounted:

```
/cdrom/INSTALL   /cdrom/OPENVIEW
/usr/OV/bin/ovinstall  -p  EAGENT  --  -d  /cdrom/OPENVIEW
```

6. After installation has been completed, you can clean up as follows:

a. On the management station, unmount the NFS-mounted CD-ROM directory. Type:

```
umount  /cdrom
```

b. On the system with the CD-ROM drive, stop exporting the CD-ROM file system.

i. Remove the line you added to **`/etc/exports`** (done in **step 3** above), and unexport the directory. Type:

```
/usr/etc/exportfs  -u  /cdrom
```

ii. Unmount the CD-ROM drive. Type:

```
umount  /cdrom
```

7. Remove the installation media from the CD-ROM drive.

Verifying Agent Software Operation

To verify that the agent software is running correctly, enter the command

For SunOS -- **/etc/chksnmpd**
For Solaris -- **/usr/sbin/chksnmpd**

If you run **chksnmpd** as root, and the agent software is running correctly, the results look similar to the following.

```
chksnmpd: hostname, Fri Nov 12 12:43:06 1993
          SU privileges
          Reading file    : /etc/snmpd.conf
          Sending request :
Success: snmpd responded to request
```

If you receive errors during your verification refer to the "Troubleshooting" chapter.

Stopping and Restarting the Agent Software

If you want to stop the agent software, you must kill the **snmpd** process. To do so, enter the command

```
snmpd -k
```

To restart the HP OpenView SNMP Agent software, you must be root. Execute the startup script with the command

For SunOS -- **/etc/netnmrc**
For Solaris -- **/etc/init.d/netmgt**

3

Configuration

This chapter discusses how to

- Configure the HP OpenView Extensible SNMP Agent to support new objects.
- Configure HP OpenView Extensible SNMP Agents to execute the **`snmptrap`** command.
- Optionally configure system contact and location, community name, and trap destinations on the HP OpenView Extensible SNMP Agent.

Configuring Extensible SNMP Agents

Configuring the HP OpenView Extensible SNMP Agent so 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 3-1 illustrates the configuration process.

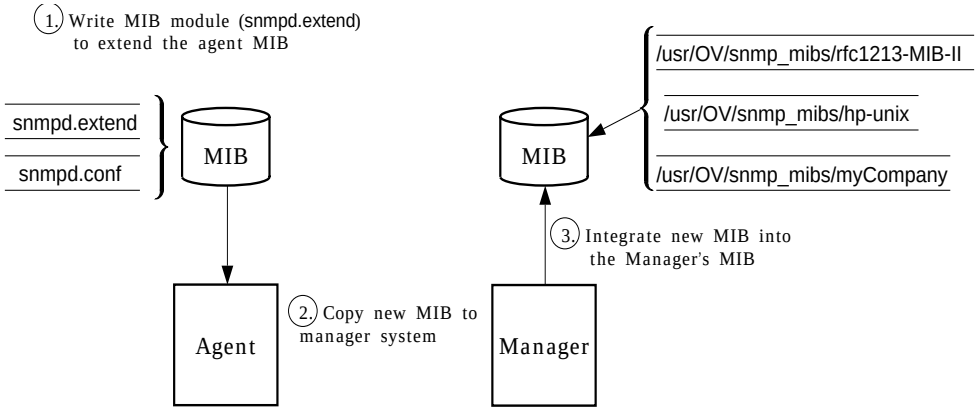


Figure 3-1. Extensible SNMP Agent Configuration

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 4, “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 or the MIB modules in the **`/usr/OV/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 counters with the 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. above.
      - 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 in chapter 1 for an illustration of the MIB tree.

If you do not have an enterprise ID assignment, you can get one by registering your enterprise with

Internet Assigned Numbers Authority
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695 USA

Telephone: (213) 822-1511
iana@isi.edu

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 **`/usr/OV/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 the “Objects Definition” section in appendix A. MIB-II is documented in RFC 1213. You can also access these MIBs online. The respective files are **`/usr/OV/snmp_mibs/hp-unix`** and **`/usr/OV/snmp_mibs/rfc1213-MIB-II`**.

Refer to the "Creating Your MIB Module" chapter for detailed information about defining MIB objects for the MIB module.

4. Reconfigure the HP OpenView Extensible SNMP Agent (**`snmpd.ea`**) **using the `/etc/snmpd.ea -c` command.**

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 **`/usr/adm/snmpd.log`** (for SunOS). For Solaris (**`/var/adm/snmpd.log`**). To see the errors on standard error, kill **`/etc/snmpd.ea`** 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 the HP OpenView Network Node Manager, the SNMP commands are **`snmpget`**, **`snmpnext`**, **`snmpnext`**, and **`snmpwalk`**. For information about the SNMP commands, see the **`snmpget(1)`**, **`snmpnext(1)`**, **`snmpset(1)`**, and **`snmpwalk(1)`** man pages. 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 5 “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 using the **`/etc/snmpd.ea -c`** command.

– Create separate **`snmpd.extend`** files for each agent.

If you want to manage different objects on the different agents, you can create separate **`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.

Once 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 the HP OpenView Network Node Manager, the default directory is **`/usr/OV/snmp_mibs`**.

Copy the **`snmpd.extend`** file from the agent to **`/usr/OV/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 the 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 the 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

- 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 **snmptrap(1)** man page

How to Define Traps

To define your own trap, you need to uniquely identify your trap. You do so by combining the generic EnterpriseSpecific 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 **snmptrap(1)** man page. See appendix B “Reference.”

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.

```
#! /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 -ax | grep lpd | 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
```

Optional Agent Software Configuration

This section defines and covers the following HP OpenView Extensible SNMP configuration values:

- system contact and location
- community name
- trap destinations

This section explains how these configuration values are implemented on the HP OpenView Extensible SNMP and tells how to optionally set them.

System Contact and Location

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

Configuring System Contact and Location

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

- Edit the **/etc/snmpd.conf** file.
- Execute the following command:

```
snmpd.ea  -C  "contact"  -L  "location"
```

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)
```

5. Reconfigure **snmpd.ea** using the **snmpd.ea -c** command.

Note The system contact and system location values set through the command **snmpd.ea -C "contact" -L "location"** take precedence over the system contact and system location values in the **snmp.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 SNMP access to MIB values on an agent. Community names are not highly secure; they go unencrypted across the network. Also, currently accessible MIB values are not normally considered sensitive information.

The HP OpenView Extensible 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 HP OpenView Extensible SNMP Agent (**`snmpd.ea`**) responds to GetRequests with any community name. No community name is initially configured for the agent in the **`snmpd.conf`** file. You can configure the agent to respond to more than one get-community-name. The associated lines from the file are

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

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 are

```
set-community-name:                    #enter community name
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 how the manager interacts with the agent, see your manager documentation.

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 Extensible SNMP Agent does not send authentication failure traps because it responds to any community name. If you want HP OpenView Extensible SNMP Agents to send authentication failure traps, you need to configure a community name in the agent's **`snmpd.conf`** file.

Configuring Community Name

Caution HP does not recommend that you change the HP OpenView Extensible SNMP Agent default community name configuration. Changing the default community name configuration results in the likelihood of inadvertently making agents inaccessible to the manager and its operations.

You may want to configure a community name on your HP OpenView Extensible SNMP Agents for the following reasons:

- To cause the agents to send **`Incorrect Community Name (authenticationFailure trap)`** events to their manager systems.
- To allow the agents to respond to a non-HP manager that sends a different community name with its SNMP requests.
- To restrict managers from accessing object values.

Entering 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 Extensible SNMP Agent's community name, edit the **`snmpd.conf`** file on the agent system.

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

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

2. Delete the comment (preceded by the # sign).

3. After the **`get-community-name:`** label, add the agent's community name.

   EXAMPLE:

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

4. Reconfigure the HP OpenView Extensible SNMP Agent software with the **`snmpd.ea -c`** command.

Note Once an HP OpenView Extensible SNMP Agent is configured with a community name, if you want to stop the agent from sending authentication failure traps, kill **`snmpd.ea`** and restart it with the **`-a`** option: **`/etc/snmpd.ea -a`**. For more information about options, see the **`snmpd.ea(1M|8)`** man page.

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 is to receive the agent's traps. An agent may have multiple trap destinations if multiple managers are managing the agent.

If you are using the HP OpenView Network Node Manager to manage your network, you do not need to set HP OpenView Extensible SNMP Agents' trap destinations manually. When an HP OpenView Network Node Manager "discovers" an HP OpenView Extensible SNMP Agent within its management region, the manager tells the agent to send SNMP traps to the manager's system. In other words, the manager's **`netmon`** process automatically sets the agent's trap destination in the **`snmpd.conf`** file. Trap destinations set in this manner are followed by the comment

```
# automatically added by network manager
```

When you unmanage a node, or delete a node from the map, the manager's **`netmon`** process automatically removes the agent's trap destination in the **`snmpd.conf`** file.

Configuring Trap Destinations

If you want an HP OpenView Extensible SNMP Agent to send traps to a manager system other than one running the HP OpenView Network Node Manager, you must manually set the agent's trap destination. To set an HP OpenView Extensible SNMP Agent's trap destination, edit the **`/etc/snmpd.conf`** file on the agent system.

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

   ```
   trap-dest:       #enter name of management 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.

   EXAMPLE:

   ```
   trap-dest: 15.2.113.223
   ```

4. If you want to specify additional trap destinations, add more **`trap-dest:`** lines to the file.

5. Reconfigure the HP OpenView Extensible SNMPAgent software with the **`snmpd.ea -c`** command.

4

Creating your MIB Module

This chapter discusses how to

- Determine what 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.

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.

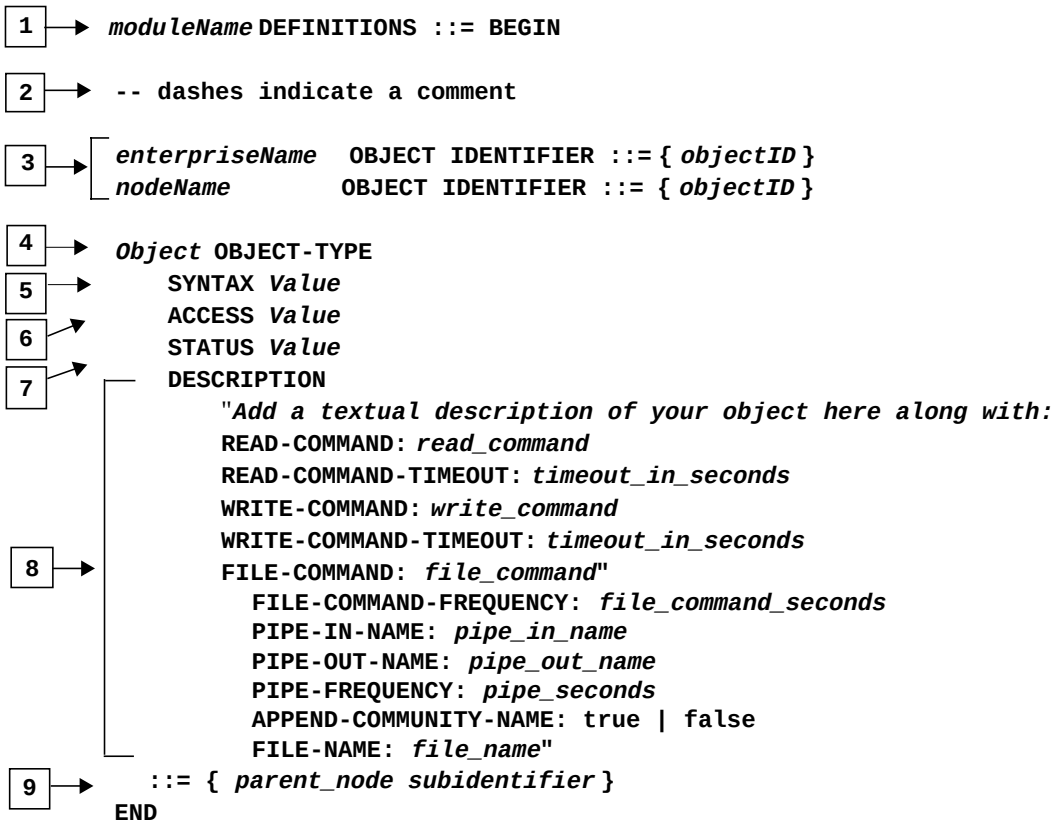
Once you have determined how to define your MIB object, use the macro template described in the next section to help you define the MIB object.

Using the Macro Template

Here is an illustration of the macro template you will use. You will use this template for defining both command and file MIB objects. You need to 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.



Here is an explanation of each callout number in the **snmpd.extend** file. For an example, see the section "Sample MIB Solution" later in this chapter.

1 - ***moduleName***	The name of your MIB module.
2 - insert two dashes (--)	Adds documentation in front of your comments.
3 - ***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.
4 - ***Object***	The textual label of your object.
5 - **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.
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).

6 - **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

7 - **STATUS** Defines the implementation support required. Valid values for **STATUS** are **mandatory**, **optional**, **obsolete**, and **deprecated**. It is recommended that you use **mandatory**.

8 - Use the **DESCRIPTION** clause to describe 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 a general error.

snmpd.ea processes one command at a time and waits for an answer before processing the next command.

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 10 seconds, the agent returns a **genErr**. 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(1)** command.

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.

snmpd.ea 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.

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.

9 - ***parent_node*** is the 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 to

- 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 4-1.

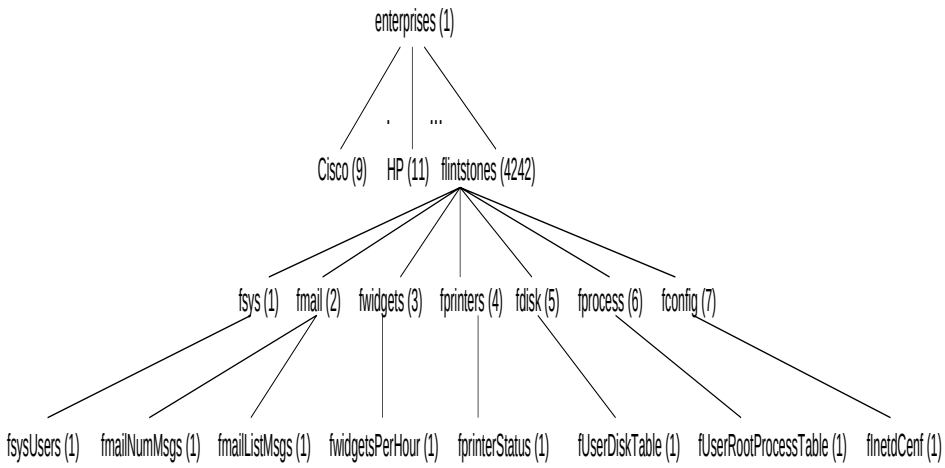


Figure 4-1. MIB Tree Structure for Sample MIB Solution

The object identifiers for the leaf nodes, that is, the MIB objects, are:

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 **/usr/OV/prg_samples/eagent/snmpd.extend to /etc/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
        STATUS  mandatory
        DESCRIPTION
                "List of users on the flintstones machine.
                READ-COMMAND: /usr/ucb/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/lib/sendmail -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/lib/sendmail -b
                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: /usr/OV/prg_samples/eagent/num_widgets $i $c $o $s
           READ-COMMAND-TIMEOUT: 2
          WRITE-COMMAND: /usr/OV/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 -ax | grep lpd | 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 will wait three seconds for a response. If the command does not finish before the timeout, the agent kills the command and returns a general error.

4. Reconfigure the HP OpenView Extensible SNMP Agent (**snmpd.ea**) using the **/etc/snmpd.ea -c** command.

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 **/usr/adm/snmpd.log** (for SunOS). (For Solaris the file is **/var/adm/snmpd.log**). To see the errors on standard error, kill **/etc/snmpd.ea** 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 **snmpget(1)**, **snmpnext(1)**, **snmpset(1)**, and **snmpwalk(1)** man pages.

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/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**, or **SEQUENCE OF**. These objects can have **ACCESS** of **read-only** or **read-write**.

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 hexidecimal. For example, if the file contained **0x0800093519D0**, the agent would return the hexidecimal 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(4)** 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 **snmpd.ea** 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** (for SunOS). (For Solaris the error is logged to **/var/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 email address of the user. The data on the remote machine is located in a file **`/usr/OV/prg_samples/eagent/user_disk_space`**. The contents of the file look like this.

```
# User ID     User Name    Disk Space      Email Address
   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@bobby
```

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(4)**.

```
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 }

fUserDiskTable          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: /usr/OV/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       isplayString
           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 **/usr/OV/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@bobby
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 **/usr/OV/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 email 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@bobby"
```

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 email 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 **/usr/OV/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@bobby"
```

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(1M|8)`** command to update the file's contents. For example, a UNIX script could be run every night that creates **`/usr/OV/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** field is executed before the agent reads the **snmpd.extend** 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: /usr/OV/prg_samples/eagent/get_processes
                  FILE-NAME: /usr/OV/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

```
/usr/OV/prg_samples/eagent/get_processes
```

This command creates the file

```
/usr/OV/prg_samples/eagent/root_processes
```

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

The agent then reads **`/usr/OV/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.

/**`usr/OV/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 **`genErr`** 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(1M|8)**, 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: /usr/OV/prg_samples/eagent/update_inetd $r
             FILE-COMMAND-FREQUENCY: 7200
             FILE-NAME: /etc/inetd.conf"
      :: = { fconfig 1 }
```

If the agent receives a SNMP SetRequest for **flintstones.fconfig.fIntdConf.0** with the contents of a **inetd.conf(4)** configuration file, the agent would execute the command

```
/usr/OV/prg_samples/eagent/update_inetd SetRequest
```

The agent would then write the contents to **/etc/inetd.conf**. The agent then executes

```
/usr/OV/prg_samples/eagent/update_inetd PostSetReques
```
t

The command checks the value of the first argument. If the value is "**PostSetRequest**," the command executes **/etc/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: /usr/OV/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

`/usr/OV/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 **`genErr`** is returned to the management station and a message is logged to **`snmpd.log`**.

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

An example background process could look like this:

```
#! /bin/sh
 if [ ! -p  /tmp/fifo_out ] ;then
    mkfifo /tmp/fifo_out
fi
if [ ! -p  /tmp/fifo_in ] ;then
    mkfifo /tmp/fifo_in

fi
while [ 1 ]; do
     read addr cname oid oid_type pdu instance < /tmp/fifo_out
     ps -u root | grep -v PID | awk '{print $1 " " $4}' \
             > /usr/OV/prg_samples/eagent/root_processes
     echo "\00" >>/tmp/fifo_in
done
exit 0
```

The files **/tmp/fifo_out** and **/tmp/fifo_in** can be created using the **mkfifo(1)** 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 10 seconds for data from the ***pipe_in_name*** pipe.

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:

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

/usr/OV/prg_samples/eagent/memory.larry contains the value **32** for 32 megabytes.
/usr/OV/prg_samples/eagent/memory.curly contains **128** and
/usr/OV/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: /usr/OV/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

/usr/OV/prg_samples/eagent/memory.communityName.

If the agent receives an SNMP GetRequest for **flintstones.fsys.fmemory.0** with community name "**moe**," the agent reads **/usr/OV/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 **/usr/OV/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**:

```
/usr/OV/prg_samples/eagent/sysDescr.larry
/usr/OV/prg_samples/eagent/sysDescr.curly
/usr/OV/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: /usr/OV/prg_samples/eagent/sysDescr"
     ::= { system 1 }
```

If the agent receives an SNMP GetRequest for **system.sysDescr.0** with community name "**moe**," the agent reads **/usr/OV/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 **/usr/OV/prg_samples/eagent/num_widgets** or **/usr/OV/prg_samples/eagent/change_num_widgets**.

   Arguments

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

   By default, **snmpd.ea** 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 **snmpd.ea** 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 **snmpd.ea** to pass in the IP address of the management station, the community name, and the object ID when executing the **/usr/OV/prg_samples/eagent/change_num_widgets** command, specify the following in the **DESCRIPTION** clause in the **snmpd.extend** file:

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

When the command executes, **snmpd.ea** 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 **sh(1)** man page. Shell commands do not have a path. See the **snmpd.extend(4)** man page 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 command must exit with the following exit codes (These exit codes are defined by RFC 1157.):

read_command exit codes are:

Exit Code	What the code means
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 a general 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**, a general error will be returned to the management station.
5	General error, that is, the command is unsuccessful.

write_command exit codes are:

Exit Code	What the code means
0	No error, that is, your command is successful. No data should be echoed to standard out or standard error.
3	Bad value, that is, the value passed into the command is invalid. For example, if the command accepts values from 1 to 9, and a user tries to pass in a value of 132, the shell command exits with 3.
5	General error, 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 general error is returned to the management station.

Any errors the system encounters while trying to execute your shell command, are returned as general errors. For example, if your program does not exist, the system returns a general error. If the agent returns a general error, an error message is logged to **/usr/adm/snmpd.log** (for SunOS). For Solaris, the message is logged to **/var/adm/snmpd.log**.

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/lib/sendmail -bp
```

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 below. The **num_widgets** command is defined in the example **snmpd.extend** file as

   ```
   DESCRIPTION "
         READ-COMMAND: /usr/OV/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(1M|8)).
#
# 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
```

5

Troubleshooting

This chapter focuses on troubleshooting the HP OpenView Extensible SNMP Agent product itself and the **`snmpd.extend`** file. It does not cover how to do network troubleshooting.

This chapter discusses the following topics:

- recommended practices for problem prevention, isolation, and recovery
- characterizing the problem
- general product troubleshooting considerations
- troubleshooting by component

Recommended Practices

Following these recommended practices helps you to prevent problems, isolate problems, and recover from problems.

- Ensure that the agent system meets the hardware, software and configuration prerequisites, and recommendations discussed in previous chapters.
- Do not modify HP OpenView Extensible SNMP Agent product files, such as **`snmpd.conf`** and **`snmpd.extend`**, 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 **`snmpd(1M|8)`** man page.

The next table describes the logging options for the HP OpenView Extensible SNMP Agent background process.

Caution Logging is normally used only by support personnel. Logging can cause the log file to grow extremely large (multiple megabytes in size) fairly quickly. If you use logging, remember to frequently monitor the size of the log file and turn logging off as soon as you are finished.

The following information explains the variable ***logmask*** in the next table. Log masks specify the type of output listed in **/usr/adm/snmpd.log** or in ***logfilename***. To select multiple output types, add the individual ***logmask*** values together and enter that number.

0	Turn off logging.
1	Log authentication failure traps.
2	Log errors.
4	Log configuration requests.
8	Log requests and replies.
16	Log requests and replies for objects that have been added.
32	Log hexdumps of packets received and sent by the **snmpd.ea**.
64	Log trace messages.

The default log mask is 3, that is, **snmpd.ea** logs authentication failure traps and errors.

EXAMPLES:

To turn logging off when the agent is running, enter the following command:

```
snmpd.e -M 0
```

To log errors and requests and replies, add their ***logmask*** value (2 + 8) to determine the ***logmask*** value to enter.

```
snmpd.ea -M 10
```

The default log file is **/usr/adm/snmpd.log** (for SunOS) (for Solaris, **/var/adm/snmpd.log**) and logging is automatically set to ON (log mask 3). See the following table if you want to make changes to the logging defaults.

Logging of the Agent Process `/etc/snmpd.ea/etc/snmpd` (for SunOS) `/usr/sbin/snmpd` (for Solaris)	
To...	**Use...**
Set initial logging mask	`-m` *`logmask`*
Change default log file	`-l` *`logfilename`*
Change logging mask	`-M` *`logmask`*
Turn off logging	`-M 0`

When the object you are trying to get returns an error, look in the **`snmpd.log`** to find out what the error is. Also, 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 that follows.

Scope: What is Affected?

Distinguish Agent vs. 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 the "Troubleshooting by Component" section.

Distinguish Agent vs. snmpd.extend File Problem

See the "Troubleshooting by Component" section.

Affected Parts of the HP OpenView Extensible SNMP Agent

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

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, or UDP, which 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 in

- the operational overview in the “Introduction and Operational Concepts” chapter
- the “Supported MIB Objects” appendix
- the “Reference” appendix
- the Request for Comment (RFC) documents referenced in the “Introduction and Operational Concepts” chapter

If a problem does not appear to be with the HP OpenView Extensible SNMP Agent itself or with the **`snmpd.extend`** file, see to your

- networking documentation for network troubleshooting procedures
- 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
- **`snmpd.extend`** file

For detailed information on a command or process mentioned in this section, see its associated man page in the “Reference” appendix.

Runtime Components

If you are having problems running the HP OpenView Extensible SNMP Agent, check software versions, file permissions, and startup scripts.

Agent File Permissions

By default, the HP OpenView Extensible SNMP Agent (**`snmpd.ea`**) is executable only by root; however, the **`/etc/chksnmpd`** (for SunOS) command is executable by anyone. For Solaris, the command is **`/usr/sbin/chksnmpd`**.

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 startup execution sequence for the HP OpenView Extensible SNMP Agent is shown below.

For SunOS:
`/etc/rc.local --> /etc/netnmrc --> /etc/snmpd`

For SunOS:
`/etc/init.d/netmgt --> /usr/sbin/snmpd`

SNMP Subsystem

Use the following methods to troubleshoot the SNMP subsystem.

- Verify that community names are correctly configured on both the manager and agent system. See the "Configuration" chapter for detailed information on this topic.
- Use the **`chksnmpd`** command to verify operation of the **`snmpd.ea`** (agent) on an HP OpenView Extensible SNMP Agent system. See the **`chksnmpd(1)`** man page.
- Check the **`snmpd.log`** file on an HP OpenView Extensible SNMP Agent for **`snmpd.ea`** errors (if you have logging turned on).
- Verify that the agent system's trap destination is set properly. See the "Configuration" chapter for more information on this topic.
- If you use the HP OpenView Network Node Manager, you can also use the **`Diagnose: Network Connectivity -> IP / TCP / SNMP`** operation to verify SNMP operation on a remote SNMP node.

Agent MIB

Use the following 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 the 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 the 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 **`/etc/snmpd.conf`** file.
- If you use the HP OpenView Network Node Manager or the HP OpenView SNMP Platform Management product, you can also use the **`Browse MIB`** operation to verify that the information retrieved is accurate.

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 find out what commands are executed with each object, set the following logging mask

```
snmpd.ea -M 19
```

- 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 supported by HP OpenView Extensible SNMP Agents for Sun
- non-standard HP extended MIB objects, that is, Hewlett-Packard's enterprise-specific MIB module These are located in **`/usr/OV/snmp_mibs/hp-unix`**

Standard MIB-II Objects Supported by Extensible SNMP Agents

This section lists the standard MIB-II
objects that

- the HP OpenViewExtensible SNMP Agent allows you to change
- return the value 0 (zero)
- return **`noSuchName`** errors (SunOS 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, enter a **`set-community-name`** to the **`/etc/snmpd.conf`** file on the agent.

```
sysContact
sysLocation
sysName

snmpEnableAuthTraps
```

Objects That Return Null Values (SunOS only)

```
ifInNUcastPkts
ifInDiscards
ifOutNUcastPkts
ifOutDiscards
```

Objects That Return noSuchName Errors (SunOS 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
```

Non-Standard HP Extended MIB Objects

This section defines HP OpenView Extensible SNMP Agent-specific MIB objects. The HP OpenView SNMP Agent MIB is consistent with the Internet Activity Board's (IAB's) network management strategy defined in RFC1109, RFC1155, RFC1157, and RFC1212. For a listing of the non-standard HP extended MIB objects, see the **`/usr/OV/snmp_mibs/hp-unix`** file on the HP OpenView network management system.

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 multi-vendor environments. As such it is vital that object types have consistent meaning across all machines.
::=	The OBJECT IDENTIFIER corresponding to the object type.

B

Reference

This appendix contains the Reference Pages (man pages) for the HP OpenView Extensible SNMP Agent product.

Note For Solaris system administration commands (1M), refer to the section 8 man pages in this appendix.

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

A superset of the HP OpenView TCP/IP Agent that extends an agent's 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.

HP OpenView TCP/IP Agent

HP's TCP/IP agent that provides the capabilities necessary to manage HP 9000 and Sun SPARCstation computers through SNMP.

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.

snmpd.ea

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.

SNMP table

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

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

Symbols

A

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

We Welcome Your Comments

HP OpenView
Extensible SNMP Agent Administrator's Reference

Please circle the number that best describes your opinion about the following items

	Strongly Agree	Agree	No Opinion	Disagree	Strongly Disagree
The information you need is:					
Easy to find	**1**	**2**	**3**	**4**	**5**
Easy to understand	**1**	**2**	**3**	**4**	**5**
Complete	**1**	**2**	**3**	**4**	**5**
Accurate	**1**	**2**	**3**	**4**	**5**
The illustrations are:					
Useful	**1**	**2**	**3**	**4**	**5**
Easy to understand	**1**	**2**	**3**	**4**	**5**
The examples are:					
Useful	**1**	**2**	**3**	**4**	**5**
Easy to understand	**1**	**2**	**3**	**4**	**5**

Comments: _____

_____

_____

_____

Name and Job Title: _____

Company: _____

Address: _____

City & State: _____

Country: _____ **Zip Code:** _____

How often do you use this manual? _____

Are you a Network Administrator? _____ **Programmer?** _____

Other (please specify) _____

How long have you worked with computer networks? _____

Which computers, operating systems, and networks, have you worked with? How long? _____

_____

_____

_____

Part Number: B1038-90002 E0294

HEWLETT PACKARD

Do not Staple - Please fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 37 LOVELAND, CO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD COMPANY
Colorado Networks Division
3404 East Harmony Road
Fort Collins, CO 80525

Manual Part No.
B1038-90002

B1038-90002