HP SNMP/XL User's Guide

HP 3000 MPE/iX Computer Systems

Edition 5

HEWLETT® PACKARD

Manufacturing Part Number: 36922-90036
E0494

U.S.A. April 1994

Notice

Restricted Rights Legend

Acknowledgments

UNIX is a registered trademark of The Open Group.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Contents

Figures

Tables

Preface

HP SNMP/XL is bundled with the links and comes with the LAN link, token ring link, X.25 link, Fiber Distributed Data Interface link, and the point-to-point link. HP SNMP/XL is started automatically at system startup.

This manual describes how to use HP SNMP/XL to allow HP 3000 Series 900 systems to actively participate in the management of the network in which they are installed.

Special Note

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers.

In HP documentation and in talking with HP 3000 users, you will encounter references to MPE XL, the direct predecessor of MPE/iX. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX. You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories).

Finally, you may encounter references to MPE V, which is the operating system for HP 3000s, not based on the PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as compatibility mode.

Audience

This manual is intended for network managers whose responsibility it is either to manage the HP 3000 from another network management platform or to use the HP 3000 to manage other objects in the network.

Organization of This Manual

Chapter 1 , “Introduction to HP SNMP/XL,” provides an overview of HP SNMP/XL and its product structure.

Chapter 2 , “Configuring HP SNMP/XL,” provides configuration information, if you choose to change parameters.

Chapter 3 , “Using SNMPUTIL,” provides information on control of the HP SNMP/XL subsystem.

Chapter 4 , “Troubleshooting HP SNMP/XL,” includes tracing, logging, and subsystem information.

Appendix A , “SNMP Variable Description,” describes the format of the variable-name parameter for the SNMP commands.

Appendix B , “Supported MIB Objects,” contains a list of the MIB objects that HP SNMP/XL supports.

Appendix C , “SNMP Command Examples,” provides examples of SNMP commands and the use of instance specifications.

Appendix D , “Time Zones,” contains a list of commonly used time zones and their corresponding environment variable strings.

Related Publications

The following networking manuals are referenced in this manual or may be of use to you in managing your network.

Using the Node Management Services (NMS) Utilities

HP 3000/iX Network Planning and Configuration Guide

NS 3000/iX NMMGR Screens Reference Manual

NS 3000/iX Operations and Maintenance Reference Manual

NS 3000/iX Error Messages Reference Manual

1 Introduction to HP SNMP/XL

Simple Network Management Protocol (SNMP) is a group of internet protocols that is a de facto standard for managing TCP/IP networks. The architectural model for SNMP contains three components:

1. At least one network management station with manager software. A network management station is a host running the network management protocol and network management applications that manage the network. The network management applications request information from the agents, put the information into a database, and translate messages into a readable format.
2. Network Agents. Network agents are managed devices containing agent software. They may be hosts, gateway systems, or media devices (such as bridges, hubs, or multiplexors). These agents are responsible for performing the network management functions requested by the network management stations. They are also responsible for monitoring information such as the number of connections and the speed of transmission at their location.
3. The network management protocol (SNMP). SNMP allows network management stations to manage (monitor and control) network devices. SNMP is used by the stations and agents to exchange management information. SNMP describes how agents and managers communicate and defines the types of information that can be exchanged.

The SNMP architecture provides a framework for managing internets and is defined in the following documents:

- RFC 1157 (The SNMP).
- RFC 1213 (The MIB — Management Information Base).
- RFC 1155 (The SMI — Structure of Management Information).

This chapter consists of the following major topics:

- Product overview.
- Concepts and definitions.
- Product structure.
- Hardware and software requirements.

Product Overview

HP SNMP/XL provides an industry standard network management agent on the HP 3000/XL. This agent allows the HP 3000 to be managed or to manage other objects in a multivendor environment.

HP SNMP/XL provides the HP 3000 with the Network Management Stack and Object Management Services. These services allow the HP 3000 to integrate network management applications that provide fault, performance, and configuration solutions. By providing the Stack and Object Management Services, the HP 3000 integrates into the HP Network Management Architecture.

The Network Management Stack is the Simple Network Management Protocol (SNMP) as defined in RFC 1157. The SNMP stack implements the Simple Network Management Protocol to allow internetworked managers and agents to exchange network management information. The purpose of the SNMP standard is to provide a protocol for management of the Internet protocols such as TCP, IP, ICMP, and UDP.

The Object Management Services supports the Management Information Base (MIB) as defined in RFC 1213, and various HP private extensions to the MIB. See the following section titled “Concepts and Definitions” as well as Appendix B , “Supported MIB Objects,” for more information on the MIB.

Concepts and Definitions

Following is information on some concepts related to SNMP.

Community Name

Requests for information on an agent are accompanied by a community name. A community name is similar to a password preventing unauthorized network managers from accessing agent information (MIB values). The community name sent with a request must match the community name expected by the target SNMP node (agent) and is case-sensitive. Some SNMP nodes support multiple levels of access based on the community name (e.g., read-only versus read-write). Most nodes support the community name “public,” generally considered the default.

Traps

An agent can send information to the manager without a request from the manager. Such an operation is called a trap (or an event). Traps inform the manager of changes that occur on the agent system, such as a reboot, without being asked. The agent knows which manager system to send traps to via its trap destination. A trap destination identifies a manager system that is to receive the agent’s traps.

Protocol Data Units (PDUs)

The Protocol Data Unit (PDU) is the format of data exchanged between the SNMP manager and agent. This data object is exchanged by peers and contains protocol control information and user data.

SMI

The Structure of Management Information (SMI) is a document where the SNMP network management information structure and language for organizing that information is defined.

MIB

The information on the agent that the manager requests resides in the Management Information Base (MIB). The MIB is a virtual database of managed objects contained within the SNMP agent. The MIB is not a physically distinct database, but rather it is a concept that includes configuration and status values normally available on the agent system.

The MIB defines the set of objects to be managed by SNMP and defines the syntax by which the data is represented. It allows you to get statistics and tabular data relevant to an internet protocol layer such as

TCP. It also allows you to reset some of the internet protocol values. The MIB actually outlines the set of objects (statistics and tables) that are meaningful for each internet protocol it covers.

The MIB stores information needed to manage devices on a network. It contains a list of network objects and their attributes, such as the number of packets sent to a network interface, routing table entries, and protocol-specific variables for IP routing. MIB I includes objects dealing with IP internetworking routing variables. MIB II, now an Internet standard, adds new objects to the MIB I groups and also adds two new groups. The new groups add media devices and network devices to the SNMP capabilities. See Appendix B , “Supported MIB Objects,” for a list of supported MIB objects.

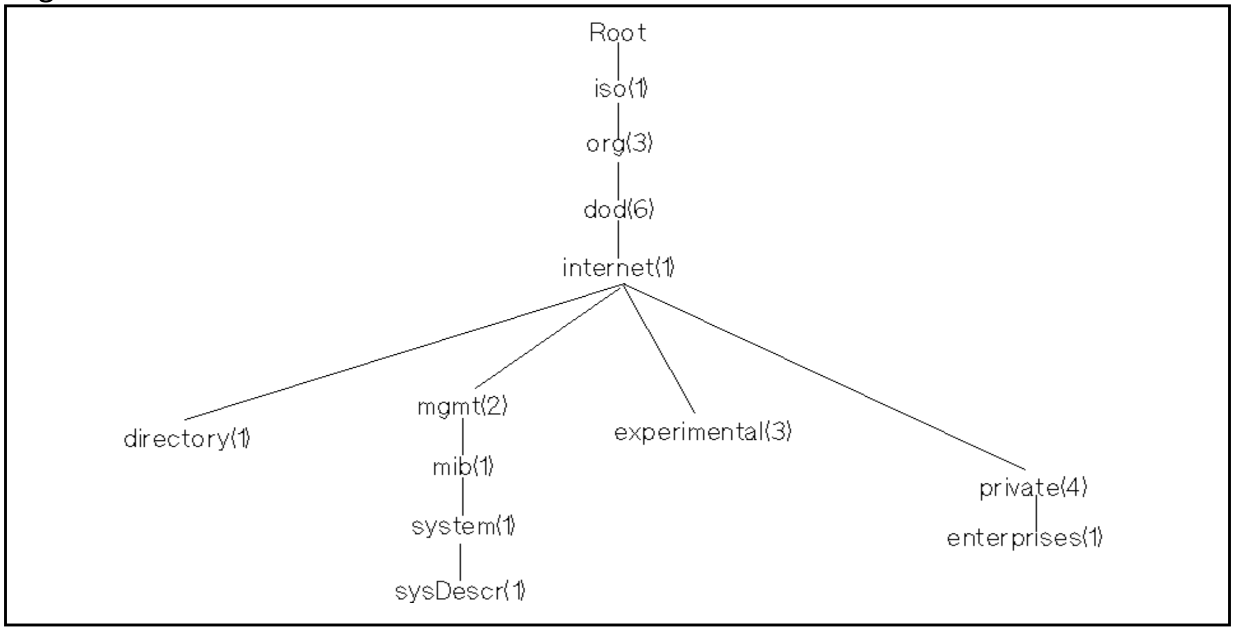
The MIB conforms to the encoding rules determined by the American National Standards Institute. These rules are called Abstract Syntax Notation One (ASN.1). ASN.1 is used:

- To define the formats of the PDUs (Protocol Data Unit) exchanged by the management protocol.
- As a means of defining the managed objects.

These objects (Object Types) are given an object identifier (name), syntax, definition, and access information.

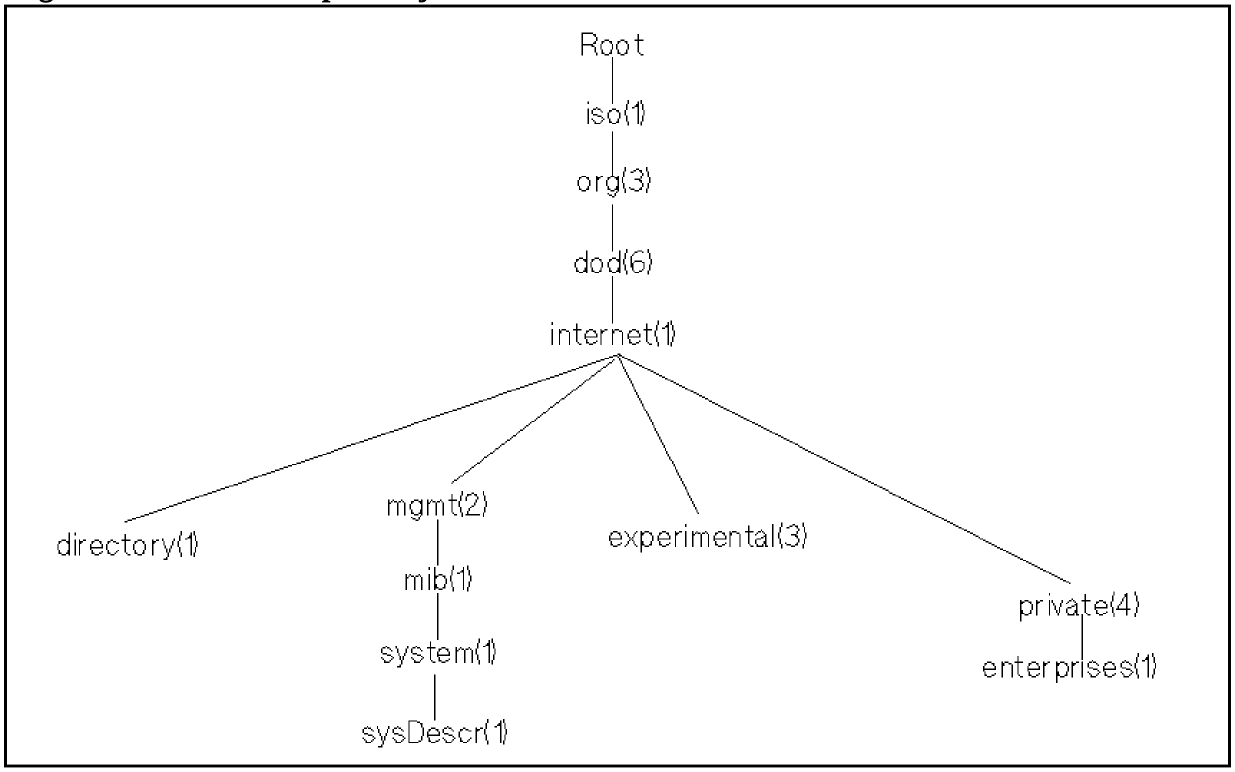
Object Identifier (Name) Each object type has a unique name which is called an Object Identifier. An Object Identifier is a sequence of integers that lead you to a certain node in the MIB architecture. Object Identifiers are organized in a hierarchical tree-like structure. Figure 1-1 shows the MIB architecture with some of the defined Object Identifiers.

Figure 1-1 MIB Architecture



For example the **sysDescr** Object Identifier is represented as `1.3.6.1.2.1.1.1.0`. (The `0` added at the end is an instance sub-identifier. This means the one and only instance of **sysDescr**.) This Object Identifier structure is shown in Figure 1-2.

Figure 1-2 Simple Object Identifier



Under the internet subtree there are four subtrees.

1. The **directory** subtree(1) is reserved for the OSI directory in the Internet.
2. The **mgmt** subtree(2) is used to identify objects which are defined in documents approved by the Internet Activities Board (IAB). At present, this includes only one subtree, **mib**.
3. The **experimental** subtree(3) is used to identify objects used in Internet experiments. If an experiment is successful, it will be moved over into the Internet-standard MIB.
4. The **private** subtree(4) is used to identify objects that are defined privately and not standardized. At present, this includes only one subtree, **enterprises**, where vendor-specific objects are registered.

Syntax The syntax of an object defines the abstract data structure for that object type. It determines how a value is displayed according to ASN.1.

Definition The definition describes the meaning of the object type.

Access Information Once SNMP acknowledges the SNMP entity as a member of a community, the managed node determines the level of access that is allowed. Access modes can be: Read-only, Read-Write, Write-only, or Not accessible.

MIB Groups

The MIB is divided into groups allowing the SNMP manager to poll the SNMP agents for information. MIB I divides the objects into the following eight groups:

1. **System Group**. Contains generic configuration information.
2. **Interface Group**. Contains generic information on the entities at the interface layer.
3. **Address Translation Group (AT)**. Contains address resolution information.
4. **Internet Protocol Group (IP)**. Contains information used to keep track of the IP layer on the managed node.
5. **Internet Control Message Protocol Group (ICMP)**. Contains 26 counters, counting how many times this message type was generated by the local IP entity and how many times this message type was received by the local IP entity. It also counts the total number of ICMP messages received, sent, received in error, or not sent due to error.
6. **Transmission Control Protocol Group (TCP)**. Contains information used to keep track of the application entities using TCP.
7. **User Datagram Protocol Group (UDP)**. Contains information used to keep track of the application entities using UDP.
8. **Exterior Gateway Protocol Group (EGP)**. Contains information about the EGP, if it is implemented.

MIB II adds the following two groups to the list.

1. **Transmission Group**. Holds a place for media-specific MIBS. They start out in the experimental branch and may eventually be placed in the Internet-standard MIB.
2. **SNMP Group**. Contains information used to keep track of SNMP application entities. It provides statistical information about the SNMP protocol entity and tracks the amount of management traffic that a device responds to.

Product Structure

The two major components of HP SNMP/XL that you will use are:

- The SNMP stack
- SNMPUTIL

The SNMP Stack

The SNMP stack runs as a system process and is the backbone of HP SNMP/XL. Its main functions are:

- To provide the interface to the network via the UDP (User Datagram Protocol) interface.
- To perform the ASN.1 encoding/decoding for both incoming and outgoing PDUs (Protocol Data Units).
- To provide event message services.
- To access MIB data for the agent.

The SNMP stack interfaces to the connectionless mode transport service (UDP subsystem) via NetIPC. Other subsystems that are used include the MPE/XL ports subsystem and the MPE/XL process management subsystem.

SNMPUTIL

The SNMPUTIL is a program file that provides the following functions:

- Manually startup and shutdown the SNMP stack.
- Generate SNMP requests and format responses back to the user.
- Generate SNMP traps.
- Access diagnostics information.
    - — Enable tracing.
    - — Enable logging.
    - — Obtain version reporting.
    - — Obtain subsystem status.

You may use SNMPUTIL to manually startup or shutdown SNMP. (Normally, SNMP is started and stopped automatically when the transport is started or stopped.) SNMPUTIL is then responsible for creating the SNMP process on startup and sending a shutdown message to the stack process for shutdown, in order to start and stop the SNMP stack. See Chapter 3 , “Using SNMPUTIL,” for more information on using SNMPUTIL.

Hardware and Software Requirements

The following hardware is required for HP SNMP/XL:

- Any HP 3000 Series 9xx.
- Any link product (NIO, CIO, PSI).
- Any protocol (SDLC, LAP-B, X.25).

The following software is required for HP SNMP/XL:

- MPE/XL Release 3.1 or later.
- One or more NS Links products.

2 Configuring HP SNMP/XL

SNMP/XL requires no configuration; however it provides a mechanism of configuring several parameters including: system contact, system location, and IP destination addresses of the Network Management Station(s) configured to receive traps.

To configure SNMP/XL, you must modify a special configuration file using any standard ASCII editor. If you want to modify the default logging configuration for SNMP/XL, you must use the NMMGR configuration utility. This chapter includes information of the following topics:

- SNMPSAMP and SNMPCONF.
- The HOSTS file.
- Modifying the SNMP/XL logging configuration.

SNMPSAMP and SNMPCONF

The SNNP/XL product is shipped with a skeleton configuration file, SNMPSAMP.NET.SYS. When the system is started, SNMP is automatically launched. The first time this is done, the product copies the SNMPSAMP file to the SNMPCONF file and initializes the configuration from the new SNMPCONF file. This file contains the configuration information for the HP SNMP/XL subsystem. Initially, SNMPSAMP looks like this:

```
File Name: SNMPSAMP.NET.SYS
#
# $Revision: 5.1.1.2 $
#
# snmp/xl reads this configuration information from this file upon
# execution.
#
# When the SNMP/XL subsystem starts up, if the file SNMPCONF.NET.SYS does not
# exist,this file SNMPSAMP.NET.SYS, is copied to SNMPCONF.NET.SYS so that the
# SNMP/XL subsystem may successfully start.
#
# Enter appropriate values after the following keywords.
#
#      agent-community-name:
#      trap-dest:
#      location:
#      contact:
#
# agent-community-name: enter community name for the snmp agent. The
# agent will respond to any SNMP requests using this community name.
# If a community name is not entered, the agent will respond to get
# requests using any community name.
#
#      example: to restrict the agent to only respond to get requests
#      using the community name “secret,” enter:
#
#      agent-community-name: secret
#
# trap-dest: enter the system name where traps will be sent. This system
# name is usually the hostname or IP address of the management station.
#
#      example: to send the traps to system names “manager1,” and
#      “15.2.113.223,” enter:
#
#      trap-dest: manager1
#      trap-dest: 15.2.113.223
#
# location: enter the physical location of the agent/system.
#
# example:if the agent is on the first floor near the west elevator, enter:
#
```

```
#       location: 1st Floor near west elevator
#
# contact: enter the person responsible for this agent, together with
# information on how to contact this person.
#
#       example: if Bob Jones is responsible for this agent, and his
#       phone number is 555-2000, enter
#
#       contact: Bob Jones (Phone 555-2000)
#
##
agent-community-name:   # enter a community name
trap-dest: # enter a system name where traps will be sent
location:  # enter location of agent
contact:   # enter contact person and how to contact this person
```

The SNMPCONF file should be the primary configuration file. This file can be modified with any standard editor and reloaded via the `UPDATECONFIG` command described in Chapter 3 , “Using SNMPUTIL.” You should add the **agent-community-name**, **trap-dests**, **location**, and **contact** information to this file.

NOTE It is recommended that you modify these parameters in a consistent manner with the SNMP network management station and applications. For example, both the agent and manager should have the same **agent-community-name**. The IP address of the **trap-dest** should be that of the management station.

If you do not add any information to this skeleton file before starting up the SNMP agent, then the **agent-community-name**, **trap-dests**, **location**, and **contact** remain unconfigured. This means that:

- The **agent-community-name** is NULL. No inbound packets are excluded on the basis of community name.
- The **location** and **contact** is NULL. An attempt to retrieve the MIB object for location or contact results in an SNMP response being returned, indicating that they are NULL.
- No traps are sent from this agent as long as no **trap-destinations** are configured.

If you want to add any new information to the configuration file, once the product has been started, you must edit the SNMPCONF file and execute the command `SNMPCONTROL UPDATECONFIG`. Or, you can shut down the product, edit the SNMPCONF file, and restart the product so that the new SNMPCONF file is read.

The HOSTS File

The SNMP/XL agent uses the Name Service Routines to resolve node names into IP addresses. These routines are used for node names that appear in the SNMPCONF file and node names that are specified in the snmp commands (for example, SNMPGET, SNMPNEXT, SNMPSET, SNMPTRAP, and SNMPWALK).

The Name Service Routines use the following two methods of resolving node names into IP addresses:

1. The file `RESLVCNF.NET.SYS` is checked to determine if a node in the network is acting as the domain name server. If a node is listed in this file, a domain name request is sent to that node to resolve the name into an IP address. If the request fails or the file does not exist, the following method is used.
2. The host database, located in the file `HOSTS.NET.SYS` is checked. This flat-ascii file contains entries for all of the hosts in the company network. An example of an entry is:

   ```
   192.1.27.63 mpenode mpenode.cup.hp.com
   ```

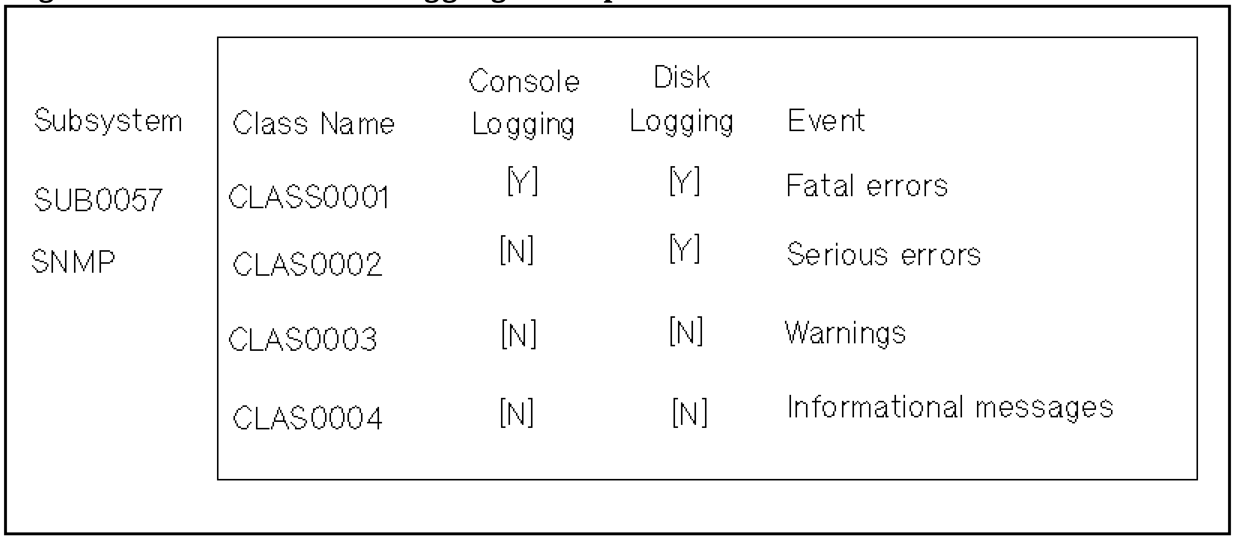
   When this method is used, the Name Service Routine sequentially searches the host database for the node name, and if found, returns the proper IP address.

If both methods fail, the subsystem is not able to generate an IP address for the node name.

Modifying the SNMP/XL Logging Configuration

When SNMP/XL first comes up, it reads the `NMCONFIG.PUB.SYS` file maintained by NMMGR and checks to see if logging is configured for HP SNMP/XL. If logging has not been configured, HP SNMP/XL automatically configures its own logging information using the set of defaults shown in Figure 2-1.

Figure 2-1 Default Logging Example



Subsystem	Class Name	Console Logging	Disk Logging	Event
SUB0057	CLASS0001	[Y]	[Y]	Fatal errors
SNMP	CLAS0002	[N]	[Y]	Serious errors
	CLAS0003	[N]	[N]	Warnings
	CLAS0004	[N]	[N]	Informational messages

If you choose to do so, you can modify the default logging settings for HP SNMP/XL using the NMMGR configuration utility. See *Using the Node Management Services (NMS) Utilities* for information on using NMMGR.

3 Using SNMPUTIL

Control of the HP SNMP/XL subsystem is provided through a User Defined Command (UDC) and a set of MPE/iX command files that invoke the SNMPUTIL program. The SNMPCONTROL UDC file (SNMPUDC) and command files are located in the NET group of the SYS account.

The SNMPUTIL is responsible for creating the SNMP process on startup and sending a shutdown message to the stack process, in order to start and stop the SNMP stack. It also can be used to generate SNMP requests, format responses back to the user, and generate SNMP traps.

It is necessary to have NM capability in order to use either the command line interface or the `SNMPCONTROL` command. When the command line interface is used, the SNMPUTIL process is responsible for outputting the input/output packets.

This chapter consists of the following major topics:

- SNMPCONTROL User Defined Command.
- SNMP command files.

SNMPCONTROL User Defined Command

The SNMPCONTROL command allows you to control the HP SNMP/XL subsystem. This command allows you to start and stop the subsystem manually, enable the internal trace facility, gather version information, obtain the current status of the subsystem, enter the help facility, dump the internal data structures to a flat-ascii file, and update the subsystem configuration.

To facilitate access to the SNMPCONTROL command, the SNMPUDC.NET.SYS must be enabled for those users who require access. This is accomplished by using the SETCATALOG command as follows:

```
SETCATALOG SNMPUDC.NET.SYS;SYSTEM.
```

Before the initial SETCATALOG you must make sure the TZ variable is properly set for your local time zone by adding the following set of lines to SNMPUDC.NET.SYS after the `setvar opt ups(“!cmd”)`:

```
if (not(bound(TZ))) then
  setvar TZ “timezonevariable”
endif
```

See Appendix D , “Time Zones,” for a list of commonly used time zones and the TZ environment variable strings that correspond to them.

You must also have NM capability to issue the SNMPCONTROL command.

NOTE Three clock parameters must be set to the correct value for the SNMPCONTROL STATUS command to display the correct time. They are the “hardware” clock, set vial CLKUTIL, the system start up clock, and the SNMPCONTROL time zone variable TZ. CLKUTIL must be set at the ISL prompt using Greenwich Mean Time (GMT). SNMP uses GMT to print out the time, therefore if the “hardware” clock is not set or set inaccurately the SNMPCONTROL STATUS will report the incorrect time.

It should be noted that the CLKUTIL is linked to the system start up clock, and when the CLKUTIL clock is changed the system start up clock will change accordingly. The system start up clock is set by the user when prompted by the console at start up. However, if the system start up clock is changed during bookup, the CLKUTIL clock is not affected. To verify that the correct time is set, ensure that the time shown by SHOWTIME (displays system start up clock) and SNMPCONTROL STATUS display match whenever a time change is made.

The following are the `SNMPCONTROL` commands:

UDC	Action
`dump`	Causes the HP SNMP/XL subsystem to dump its internal data structures.
`help`	Provides online help for the HP SNMP/XL subsystem.
`start`	Activates the HP SNMP/XL subsystem.
`status`	Reports the status of the HP SNMP/XL subsystem.
`stop`	Deactivates the HP SNMP/XL subsystem.
`trace`	Enables/disables the HP SNMP/XL internal tracing.
`updateconfig`	Provides dynamic update of the HP SNMP/XL configuration information.
`version`	Reports internal version levels of the various HP SNMP/XL modules.

SNMPCONTROL DUMP

The `SNMPCONTROL DUMP` command causes the HP SNMP/XL subsystem to dump its internal data structures to a flat-ascii file.

```
SNMPCONTROL DUMP
```

Parameters

None.

Description

This command results in a `SNMPDMxx.NET.SYS` dump file to be created and all of the internal data structures of the HP SNMP/XL subsystem to be dumped to it. The data structures are dumped by the symbolic formatting feature of the MPE/iX debugger.

Example

```
MPEXL: snmpcontrol dump
The SNMP/XL subsystem has dumped its internal data structures to
SNMPDM01.NET.SYS (SNMPINFO 1704)
MPEXL:
```

SNMPCONTROL HELP

The `SNMPCONTROL HELP` command enters the HP SNMP/XL online help facility. Once inside the online help facility, you can access help information on the HP SNMP/XL subsystem. The help facility is identical to the MPE/iX help subsystem.

SNMPCONTROL HELP

Parameters

None.

Description

This command provides an online help facility for the HP SNMP/XL subsystem.

Example

```
MPEXL: snmpcontrol help

 Welcome to the HP SNMP/XL HELP Facility

The HP SNMP/XL Help Facility has the same syntax and usage as the MPE/iX HELP
Facility. To receive help on either a SNMPCONTROL command, or one of the
other SNMP commands (e.g., SNMPGET, SNMPNEXT), enter the command, or the
option for the SNMPCONTROL command. You may also enter an optional help
parameter indicating the category of help you wish to review. Help is also
available for various HP SNMP/XL errors, warnings, and informational messages
that are returned to the user. You can get help on a SNMP/XL error, warning,
or informational message by entering the following type of command:

 help SNMPERRxxxx

Other parameters available on most help commands are:

            ALL: Explains everything about the command.
          PARMS: Explains parameter(s) for the command.
       EXAMPLES: Typical uses of that command.>

KEYWORDS: COMMAND,SYNTAX,AVAIL
MPEXL:
```

SNMPCONTROL START

The `SNMPCONTROL START` command allows you to start the HP SNMP/XL subsystem.

```
SNMPCONTROL START
                        [;TRACE]
                        [;TFILESIZE=file_size]
                        [;TFILE=file_name]
                        [;OVERRIDE]
                        [;SYSFAIL]
```

Parameters

TRACE

This parameter is optional and specifies that you want to start the subsystem with the internal trace facility enabled. When this parameter is used, the tracing remains active until the HP SNMP/XL subsystem is stopped or the trace command is used to turn off tracing. The tracing information is recorded in a file named `NMTCnnnn.PUB.SYS`, where `nnnn` represents numerically sequential trace files. This file name is the default and is used unless the `TFILE` option is specified.

The default is no tracing.

TFILE-SIZE=filez_size

This parameter is optional and is ignored unless the *TRACE* parameter is specified. It allows you to specify the number of records in the SNMP/XL trace file as the value of *file_size*. The value should be a decimal number specifying the trace file size in number of 128 word records.

The default is 1024.

TFILE=file_name

This parameter is optional and is ignored unless the *TRACE* parameter is specified. It allows you to specify the name of the trace file, instead of using the default trace file name as the value of *file_name*. *File_name* must be a string of characters that define a valid MPE file name.

OVERRIDE

This parameter is optional and allows you to override HP SNMP/XL version inconsistencies which would normally prevent successful activation of the HP SNMP/XL subsystem. Use of this parameter requires PM and DI capabilities in addition to the NM capability required to issue the `SNMPCONTROL` command.

CAUTION The *OVERRIDE* parameter is intended for use by Hewlett-Packard support engineers. Hewlett-Packard is not responsible for damages resulting from unauthorized use.

SYSFAIL

This parameter is optional and allows you to produce a system halt if an irrecoverable internal error occurs within the SNMP/XL subsystem.

The default is to trap and handle all errors but you are given the option to fail the system by specifying this parameter.

Description

This command is used to activate the HP SNMP/XL subsystem. It is only needed if you stop the subsystem and wish to restart it, or if the subsystem terminated abnormally.

Example

```
MPEXL: snmpcontrol start;trace
The HP SNMP/XL subsystem was successfully started. (SNMPINFO 1705)
MPEXL:
```

SNMPCONTROL STATUS

The SNMPCONTROL STATUS command allows you to query the status of the SNMP/XL subsystem. This status includes if the SNMP protocol is active, if the internal trace facility is active, if the sysfail option is enabled, and the status of the UDP connection.

```
SNMPCONTROL STATUS
```

Parameters

None.

Description

This command reports the status of the HP SNMP/XL subsystem.

Example

```
MPEXL: snmpcontrol status
SNMP/XL Subsystem Status (As of)     : Tues Aug 20 10:54:51  1991
SNMP/XL Subsystem Started            : Fri Aug 16 18:32:34 1991
SNMP Protocol                        : Enabled
SNMP Portid                          : $ffffdfea
Subsystem Tracing                    : Disabled
Sysfail Start Option                 : Disabled
UDP Connection                       : Enabled
Port 161 Socket Descriptor           : $0000000d
Port 162 Socket Descriptor           : $0000000e
MPEXL:
```

NOTE Make sure that the “hardware” clock, set via CLKUTIL, the system start up clock, and the SNMPCONTROL time zone variable (TZ) are set properly. See the information under the heading “SNMPCONTROL User Defined Command” at the beginning of this chapter. See Appendix D , “Time Zones,” for a list of time zone values.

SNMPCONTROL STOP

The `SNMPCONTROL STOP` command allows you to shut down the SNMP/XL subsystem. Use of this command gracefully terminates the SNMP/XL subsystem.

```
SNMPCONTROL STOP
```

Parameters

None.

Description

This command gracefully terminates the HP SNMP/XL subsystem.

Example

```
MPEXL: snmpcontrol stop
The stop request has been successfully sent to the SNMP/XL stack.
(SNMPINFO 1702)
MPEXL:
```

SNMPCONTROL TRACE

The `SNMPCONTROL TRACE` command allows you to enable or disable the internal trace facility of the SNMP/XL subsystem.

```
SNMPCONTROL TRACE
                          [;ON|OFF]
                          [;TFILE=file_name]
                          [TFILESIZE=file_size]
```

Parameters

ON/OFF

This parameter is optional. If it is not present, the current state of the internal trace is toggled. For example, if tracing is enabled, the internal trace is disabled if the parameter is not present.

The *ON* parameter enables the internal trace facility. If tracing is currently active, the current trace file is closed.

The *OFF* parameter disables the internal trace facility. If tracing is currently not active, then the command is ignored.

TFILE=file_name

This parameter is optional and is ignored unless the *TRACE* parameter is specified. It allows you to specify the name of the trace file instead of using the default trace file name as the value of *file_name*. *File_name* must be a string of characters that define a valid MPE file name.

TFILE-SIZE=file_size

This parameter is optional and is ignored unless the *TRACE* parameter is specified. It allows you to specify the number of records in the SNMP/XL trace file as the value of *file_size*. The value should be a decimal number specifying the trace file size in number of 128 word records.

The default is 1024.

Description

This command controls the HP SNMP/XL internal trace facility. When tracing is enabled the default trace file, created by the Node Management Subsystem, is named `NMTCnnnn.PUB.SYS`, where `nnnn` reflects numerically sequential trace files. If the *TFILE* parameter is specified, then the *file_name* specified is the name of the trace file.

Example

One of the examples below (`SNMPINFO 1724`) starts the internal trace facility, specifying the number of trace records at 2048.

```
MPEXL: snmpcontrol trace;on; TFILESIZE=2048
The SNMP/XL subsystem trace facility has been successfully activated
on trace file. (SNMPINFO 1724)
Tracing stopped on file. (SNMPINFO 1726)
MPEXL:
```

SNMPCONTROL UPDATECONFIG

The `SNMPCONTROL UPDATECONFIG` command allows you to dynamically update the configuration information.

```
SNMPCONTROL UPDATECONFIG
```

Parameters

None.

Description

This command provides the ability to update the SNMP configuration information without shutting down the HP SNMP/XL subsystem. The configuration information that is updated is contained in the `SNMPCONF.NET.SYS` file.

If an error is detected, the update does not take effect and the former configuration is retained.

Before any configuration information is updated, the format of the configuration file is validated.

Example

```
MPEXL: snmpcontrol updateconfig
The SNMP/XL subsystem has successfully updated its configuration information.
(SNMPINFO 1709)
MPEXL:
```

SNMPCONTROL VERSION

The `SNMPCONTROL VERSION` command allows you to obtain the version information for the SNMP/XL subsystem.

```
SNMPCONTROL VERSION
```

Parameters

None.

Description

This command reports the version level of the HP SNMP/XL subsystem. Each module comprising the HP SNMP/XL subsystem is reported. You are informed if module versions do not match or modules are missing.

Example

```
MPEXL: snmpcontrol version
HP SNMP/XL SNMP module versions:<

NM program file:     SNMP.NET.SYS           Version: A000000
NM program file:     SNMPUTIL.NET.SYS       Version: A0000000
NM program file:     SNMP_OMI_VERS          Version: A0000000
XL procedure:        SNMP_API_VERS          Version: A0000000
XL procedure:        SNMP_TRAP_VERS         Version: A0000000
XL procedure:        SNMP_UTIL_VERS         Version: A0000000
XL procedure:        SNMP_CMDS_VERS         Version: A0000000
XL procedure:        SNMP_INIT_VERS         Version: A0000000
NL procedure:        SNMP_IOCOMP_VERS       Version: A0000000

HP SNMP/XL   SNMP overall version = A.00.00
(c) Copyright 1991 Hewlett-Packard Company
(c) Copyright 1988 by Carnegie Mellon University
(c) Copyright 1988,1989, Massachusetts Institute of Technology

MPEXL:
```

SNMP Command Files

The following is a list of the command files provided for the HP SNMP/XL subsystem that perform the following operations:

Command File	Operation
`snmpget`	Sends an SNMP get request to the specified node to retrieve specific management information.
`snmpnext`	Sends an SNMP get next request to the next specified node.
`snmpset`	Sends an SNMP set request to the specified node.
`snmptrap`	Generates an SNMP trap and sends it to the configured management stations. This is used to report extraordinary events.
`snmpwalk`	Sends an SNMP get next request from the specified starting point (usually a group name), through the MIB names until the end of the MIB group is reached.

The format of the variable-name parameters for these commands are described in Appendix A , "SNMP Variable Description."

In order to execute the SNMP commands, either include NET.SYS in the HPPATH variable (see setvar in *MPE/iX Commands Manual*.)

```
setvar HPPATH "!hppath,net.sys"
```

or execute the command with full path name.

```
SNMPxxxx.NET.SYS
```

NOTE Although SNMP starts up and shuts down automatically when the network is started or shut down, you can manually start or stop SNMP using SNMPUTIL. The user interface for these commands changes if the network is down and you are trying to execute one of these commands to query the local SNMP agent. (This is true of all the CI commands listed above except SNMPTRAP.) If the network is down, the second field in these commands must have a value of `0`. This sends the request to the agent on the machine that you are running the command on. For example if the network is down the following requests are rejected:

```
snmpget perseus public system.sysDescr.0
snmpget 15.13.104.133 public system.sysDescr.0
snmpget wabbit public system.sysDescr.0
```

An example of an accepted value might look like this:

```
snmpget 0 public system.sysDescr.0
```

SNMPGET

The SNMPGET command is used to query a node using SNMP GetRequests.

```
SNMPGET NODE
                  COMMUNITY
                  VARIABLE
                  [VARIABLE...]
```

Parameters

NODE	This is a required parameter. It specifies the network element that is to be the target of the SNMP GetRequest. This parameter can either be a fully qualified IP address or a valid host name that is found in the hosts database. If `0` is specified, the request is sent to the local host.
COMMUNITY	This is a required parameter. It defines the SNMP community name that is used in the SNMP GetRequest. If `0` is specified, the community name defaults to `public`.
VARIABLE	This is a required parameter. This command accepts 1–20 fully qualified object identifiers as arguments. Each variable has the format of A.B.C.D..., where A, B, C, and D are subidentifiers in decimal notation.

Description

SNMPGET is an SNMP application that uses the GetRequest to query for information on a node. The response to the SNMP GetRequest is then formatted and returned.

Examples

The following example retrieves the system description object from the node called MPENODE.

```
MPEXL: snmpget mpenode public 1.1.0
Name: system.sysDescr.0
Octet String: HP3000 SERIES 925, MPE XL version B.31.00
NS Transport version B.
04.00

MPE XL:
```

SNMPNEXT

The SNMPNEXT command is used to query a node using SNMP GetNextRequests.

```
SNMPNEXT NODE
                    COMMUNITY
                    VARIABLE
                    [VARIABLE...]
```

Parameters

NODE	This is a required parameter. It specifies the network element that is to be the target of the SNMP GetNextRequest. This parameter can either be a fully qualified IP address or a valid host name that is found in the hosts database. If `0` is specified, the request is sent to the local host.
COMMUNITY	This is a required parameter. It defines the SNMP community name that is used in the SNMP GetNextRequest. If `0` is specified, the community name defaults to `public`.
VARIABLE	This is a required parameter. This command accepts 1 to 20 fully qualified object identifiers as arguments. Each variable has the format of A.B.C.D..., where A, B, C, and D are subidentifiers in decimal notation.

Description

SNMPNEXT is an SNMP application that uses the GetNetRequest to query for information on a node. The response to the SNMP get request is then formatted and returned.

Examples

The first example retrieves the next object after the object described by 1.1.0.

The second example retrieves the same object but uses the symbolic name for the variable parameter.

```
MPEXL: snmpnext mpenode public 1.1.0
Name: system.sysObjectID.0
Object Identifier:
.iso.org.dod.internet.private.enterprises.hp.nm.system.mpe_xl.2

MPEXL:

MPEXL: SNMPNEXT SLNODE1 PUBLIC SYSTEM.SYSCONTACT.0
system.sysName.0 : DISPLAY STRING- : xlnode1.domain.org
MPEXL:
```

SNMPSET

The SNMPSET command is used to send an SNMP SetRequest to a node.

```
SNMPSET NODE
                COMMUNITY
                [VARIABLE
                 TYPE
                 VALUE...]
```

Parameters

NODE	This is a required parameter. It specifies the network element that is to be the target of the SNMP SetRequest. This parameter can either be a fully qualified IP address or a valid host name that is found in the hosts database. If `0` is specified, the request is sent to the local host.
COMMUNITY	This is a required parameter. It defines the SNMP community name that is used in the SNMP SetRequest. If `0` is specified, the community name defaults to `public`.
VARIABLE	This is a required parameter. This command accepts 1 to 10 fully qualified object identifiers as arguments. Each variable has the format of A.B.C.D..., where A, B, C, and D are subidentifiers in decimal notation.
TYPE	This is a required parameter which is paired up with a variable and value. The type parameter must be one of the following: `Integer`, `OctetString`, `ObjectIdentifier`, `Null`, `IpAddress`, `Counter`, `Gauge`, `TimeTicks`, or `Opaque`.
VALUE	This is a required parameter. It defines the new value for the variable specified. The value must be valid for the type specified.

Description

SNMPSET is an SNMP command that is used to send a SetRequest to an SNMP agent and allows for certain MIB variables to be modified. It is used to update the configuration file and IP address. This means that a trap is sent to this managed node.

Example

The following example sets the system contact to “`Bob Jones`” for the node `xlnode` using the SNMP community name `public`.

```
MPEXL: snmpset xlnode public system.syscontact.0
               Octetstring “ 'Bob Jones' ”
               Name: system.sysContact.0
Display String: Bob Jones
MPEXL:
```

SNMPWALK

The `SNMPWALK` command is used to retrieve particular groups of the MIB from a remote node or possibly the entire MIB. This command uses the SNMP GetNextRequest to “walk” through the MIB.

```
SNMPWALK NODE
                      COMMUNITY
                      [VARIABLE]
```

Parameters

NODE	This is a required parameter. It specifies the network element that is to be the target of the SNMP Walk request. This parameter can either be a fully qualified IP address or a valid host name that is found in the hosts database. If `0` is specified, the request is sent to the local host.
COMMUNITY	This is a required parameter. It defines the SNMP community name that is used in the SNMP Walk request. If `0` is specified, the community name defaults to `public`.
VARIABLE	This is an optional parameter. If it is not specified, the walk operation begins at the top of the internet management MIB (`.iso.org.dod.internet.mgmt.mib`). If the parameter is specified, it must be a fully qualified object identifier. The variable has the format of A.B.C.D..., where A, B, C, and D are subidentifiers in decimal notation.

Description

`SNMPWALK` is an SNMP application that uses the GetNextRequest to query for a subtree of information about a node. All variables in the subtree below the given mib-group-name are queried and their values presented.

Example

The following example walks through the system group of the MIB.

```
MPEXL: snmpwalk mpenode public system
Name: system.sysDescr.0
Octet String: HP3000 SERIES 925, MPE XL version B.31.00 NS Transport version B.
04.00

Name: system.sysObjectID.0
Object Identifier:
.iso.org.dod.internet.private.enterprises.hp.nm.system.mpe_xl.2

Name: system.sysUpTime.0
Timeticks: (319338799) 3 days, 16:43:07

Name: system.sysContact.0
Display String: John Smith

Name: system.sysName.0
Display String: mpenode.domain.organization

Name: system.sysLocation.0
Display String: in Cupertino, Building 43U Pillar N4

Name: system.sysServices.0
Integer: 72

MPEXL:
```

SNMPTRAP

The SNMPTRAP command is used to send a SNMP Trap to a remote node which in most cases is a management node.

```
SNMPTRAP NODE
                     COMMUNITY
                     ENTERPRISE
                     AGENT_ADDR
                     GENERIC_TRAP
                     SPECIFIC_TRAP
                     TIME_STAMP
                     [VARIABLE
                      TYPE
                      VALUE..]
```

Parameters

NODE	This is a required parameter. It specifies the network element that is to be the target of the SNMP Trap. This parameter can either be a fully qualified IP address or a valid host name that is found in the hosts database. If you specify `0` the trap is broadcast to all registered managers.
COMMUNITY	This is a required parameter. It defines the SNMP community name that is used in the SNMP Trap

request. If `0` is specified, the community name defaults to `public`.

ENTERPRISE	This is a required parameter. The enterprise name is an object identifier that defines the entity generating the SNMP trap. If you wish, this parameter can be set to `0`, and the command inserts the enterprise value for the system where the trap is generated.
AGENT_ADDR	This is a required parameter. It defines the IP address of the node where the trap actually originated. This parameter can be an IP address in dot notation or a host name that is in the hosts database. If you wish, this parameter can be set to `0`, and the IP address of the current node is used.
GENERIC_TRAP	This is a required parameter. The `generic_trap` must be a number between 1 and 6 where:

SPECIFIC_TRAP

```
1 — warmStart
2 — linkDown
3 — linkUp
4 — authentificationFailure
5 — egpNeighborLoss
6 — enterpriseSpecific
```

This is a required parameter. When the `generic_trap` value is `6`, this value is used to further define the trap that has occurred. This parameter must be a positive number.

TIME_STAMP	This is a required parameter. This parameter defines the number of TimeTicks since the network entity has been restarted. If you set this value to `0`, the `time_stamp` is computed by the `SNMPTRAP` command.
VARIABLE	This is an optional parameter. This command accepts 0 to 5 fully qualified object identifiers as arguments. Each variable has the format of A.B.C.D..., where A, B, C, and D are subidentifiers in decimal notation.
TYPE	This is an optional parameter which is paired up with a variable and value. The type parameter must be one of the following: `Integer`, `OctetString`, `ObjectIdentifier`, `Null`, `IpAddress`, `Counter`, `Gauge`, `TimeTicks`, or `Opaque`. For a complete description of each type, refer to RFC 1155.
VALUE	This is an optional parameter. It defines the new value for the variable specified. The value must be valid for the type specified.

Description

SNMPTRAP allows you to generate a trap to a single node in the network or a trap that is routed to each of the management stations that are listed in the trap-destination list in the SNMP configuration file.

This command requires PM and DI capabilities, in addition to the normal NM capability required for all of the SNMP command set.

Example

The following example sends a linkUp trap to node `xlnode2` from `xlnode1` with the default enterprise name and `time_stamp`.

```
MPEXL: snmptrap 0 public 0 192.1.27.63 6 1234 0
The snmp trap was successfully sent (SNMPINFO 1813)
MPEXL:
```

4 Troubleshooting HP SNMP/XL

This chapter contains information on the following topics which are the diagnostic features for HP SNMP/XL:

- Tracing and logging.
- Subsystem information.

Tracing and Logging

HP SNMP/XL uses the NMS utility NMDUMP for tracing and logging. HP SNMP/XL also follows the general format of NetXport, NetIPC, and NS services in tracing and logging. This includes initiation and formatting options.

Tracing

A trace initiation is able to trace the subsystem startup and allows you to specify the trace file name and file size. There is a single command for turning on/off tracing which is `SNMPCONTROL TRACE`.

The following filters are available for formatting the trace file:

- Time range selection.
- Subsystem ID.
- Data display options (octal, hex decimal, ASCII).
- Exception reporting.

System Logging

The logging destination for messages with various logging classes is provided through NMMGR guided configuration. The general message format looks like:

```
Message text (SNMPFATAL nnnn)
             (SNMPINFO nnnn)
             (SNMPWARN nnnn)
             (SNMP nnnn)
```

where `nnnn` is the message number.

Message Content

The following information is included in the message record to allow for formatting options based on these criteria:

- Subsystem ID.
- Logging class.
- System time stamp.
- Log all SNMPCONTROL commands issued.

Logging Class

The general classification of the logging classes consists of the following classes:

- CLAS0001 Fatal error.
- CLAS0002 Serious error.
- CLAS0003 Warning.
- CLAS0004 Informational message.

Log Formatting Options

Formatting of the log files is by the following criteria:

- Time range.
- Subsystem ID.
- Data display option (octal, hex decimal, ASCII).

Subsystem Information

The subsystem version, status, and configuration information of the HP SNMP/XL subsystem is provided through the `SNMPCONTROL` commands. This information helps identify problems.

The data that needs to be collected for problem resolution are:

- OS version.
- SNMP version.
- UDP version.
- Output of `SNMPCONTROL DUMP` (a dumpfile).
- Output of `SNMPCONTROL STATUS`.
- Log file.
- Trace file (if trace is on).
- SNMPCONF file information.

  — Community name.

  — Trap destinations.

  — Location.

  — System contact.

A SNMP Variable Description

The following text describes the format of the variable-name parameter of the `snmpget`, `snmpnext`, `snmpset`, and `snmpwalk` commands.

SNMP variable names are specified in the format of Abstract Syntax Notation One (ASN.1) Object Identifiers. Each variable represents an object specification in the Management Information Base (MIB), which is organized in a hierarchical tree-like manner.

Variables may be represented in several ways. Each variable name is given in the format of A.B.C.D..., where A, B, C, and D are subidentifiers in one of two forms of notation. Each subidentifier may be encoded as a decimal integer or as a symbolic name matching those listed in the following section. The symbolic names are not case-sensitive.

Fully specified variables begin with a dot(.).

If there is no leading dot (.) in the variable name, and if the first subidentifier is the symbolic name hp, then the name is formed as if it was preceded with `iso.org.dod.internet.private.enterprises`.

If there is no leading dot (.) in the variable name, and if the first subidentifier is not the symbolic name `hp`, the name is formed as if it was preceded with `iso.org.dod.internet.mgmt.mib`.

Variables contain two portions. The first portion is the standardized specification of the variable, (that is, `system.sysDescr`). The second portion is an instance specification used to distinguish different instances of the object in the same MIB. If there can only be one instance, this part of the variable name is the subidentifier zero `0`, (that is, `system.sysDescr.0`). See RFC 1213 for more information on object instances. Refer to Appendix C , “SNMP Command Examples,” for command examples.

SNMP Variables

The RFC 1213 specifies the following variables. These variables are in the `.iso.org.dod.internet.mgmt.mib` subtree.

```
system.sysDescr
system.sysObjectID
system.sysUpTime
system.sysContact
system.sysName
system.sysLocation
system.sysServices
```

```
interfaces.ifNumber
```

```
interfaces.ifTable.ifEntry.ifIndex
interfaces.ifTable.ifEntry.ifDescr
interfaces.ifTable.ifEntry.ifType
interfaces.ifTable.ifEntry.ifMtu
interfaces.ifTable.ifEntry.ifSpeed
interfaces.ifTable.ifEntry.ifAdminStatus
interfaces.ifTable.ifEntry.ifOperStatus
interfaces.ifTable.ifEntry.ifLastChange
interfaces.ifTable.ifEntry.ifPhysAddress
interfaces.ifTable.ifEntry.ifInOctets
interfaces.ifTable.ifEntry.ifInUcastPkts
interfaces.ifTable.ifEntry.ifInNUcastPkts
interfaces.ifTable.ifEntry.ifInDiscards
interfaces.ifTable.ifEntry.ifInErrors
interfaces.ifTable.ifEntry.ifInUnknownProtos
interfaces.ifTable.ifEntry.ifOutOctets
interfaces.ifTable.ifEntry.ifOutUcastPkts
interfaces.ifTable.ifEntry.ifOutNUcastPkts
interfaces.ifTable.ifEntry.ifOutDiscards
interfaces.ifTable.ifEntry.ifOutErrors
interfaces.ifTable.ifEntry.ifOutQLen
interfaces.ifTable.ifEntry.ifSpecific
```

```
at.atTable.atEntry.atIndex
at.atTable.atEntry.atPhysAddress
at.atTable.atEntry.atNetAddress
```

```
ip.ipForwarding
ip.ipDefaultTTL
ip.ipInReceives
ip.ipInHdrErrors
ip.ipInAddrErrors
ip.ipForwDatagrams
ip.ipInUnknownProtos
ip.ipInDiscards
ip.ipInDelivers
ip.ipOutRequests
ip.ipOutDiscards
ip.ipOutNoRoutes
ip.ipReasmTimeout
ip.ipReasmReqds
ip.ipReasmOKs
ip.ipReasmFails
ip.ipFragOKs
ip.ipFragFails
ip.ipFragCreates

ip.ipAddrTable.ipAddrEntry.ipAdEntAddr
ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex
ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask
ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr
ip.ipAddrTable.ipAddrEntry.ipAdEntReasmMaxSize

ip.ipRoutingTable.ipRouteEntry.ipRouteMetric1
ip.ipRoutingTable.ipRouteEntry.ipRouteMetric2
ip.ipRoutingTable.ipRouteEntry.ipRouteMetric3
ip.ipRoutingTable.ipRouteEntry.ipRouteMetric4
ip.ipRoutingTable.ipRouteEntry.ipRouteNetxHop
ip.ipRoutingTable.ipRouteEntry.ipRouteType
ip.ipRoutingTable.ipRouteEntry.ipRouteProto
ip.ipRoutingTable.ipRouteEntry.ipRouteAge
ip.ipRoutingTable.ipRouteEntry.ipRouteMask
ip.ipRoutingTable.ipRouteEntry.ipRouteMetric5
ip.ipRoutingTable.ipRouteEntry.ipRouteInfo

ip.ipNetToMediaTable.ipNetToMediaEntry
.ipNetToMediaIfIndex
ip.ipNetToMediaTable.ipNetToMediaEntry
.ipNetToMediaPhysAddress
ip.ipNetToMediaTable.ipNetToMediaEntry
.ipNetToMediaNetAddress
ip.ipNetToMediaTable.ipNetToMediaEntry
.ipNetToMediaType
ip.ipRoutingDiscards

icmp.icmpInMsgs
icmp.icmpInErrors
icmp.icmpInDestUnreachs
icmp.icmpInTimeExcds
icmp.icmpInParmProbs
icmp.icmpInSrcQuenchs
icmp.icmpInRedirects
icmp.icmpInEchos
icmp.icmpInEchoReps
icmp.icmpInTimestamps
icmp.icmpInTimestampReps
icmp.icmpInAddrMasks
icmp.icmpInAddrMaskReps
icmp.icmpicmpOutMsgs
icmp.icmpOutErrors
icmp.icmpOutDestUnreachs
icmp.icmpOutTimeExcds
icmp.icmpOutParmProbs
icmp.icmpOutSrcQuenchs
icmp.icmpOutRedirects
icmp.icmpOutEchos
icmp.icmpOutEchoReps
icmp.icmpOutTimestamps
icmp.icmpOutTimestampReps
icmp.icmpOutAddrMasks
icmp.icmpOutAddrMaskReps

tcp.tcpRtoAlgorithm
tcp.tcpRtoMin
tcp.tcpRtoMax
tcp.tcpMaxConn
tcp.tcpActiveOpens
tcp.tcpPassiveOpens
tcp.tcpAttemptFails
tcp.tcpEstabResets
tcp.tcpCurrEstab
tcp.tcpInSegs
tcp.tcpOutSegs
tcp.tcpRetransSegs

tcp.tcpConnTable.tcpConnEntry.tcpConnState
tcp.tcpConnTable.tcpConnEntry.tcpConnLocalAddress
tcp.tcpConnTable.tcpConnEntry.tcpConnLocalPort
tcp.tcpConnTable.tcpConnEntry.tcpConnRemAddress
tcp.tcpConnTable.tcpConnEntry.tcpConnRemPort

tcp.tcpInErrs
tcp.tcpOutRsts

udp.udpInDatagrams
udp.udpNoPorts
udp.udpInErrors
udp.udpOutDatagrams
udp.udpTable.udpEntry.udpLocalAddress
udp.udpTable.udpEntry.udpLocalPort

egp.egpInMsgs
egp.egpInErrors
egp.egpOutMsgs
egp.egpOutErrors
egp.egpAs

egp.egpNeighTable.egpNeighEntry.egpNeighState
egp.egpNeighTable.egpNeighEntry.egpNeighAddr
egp.egpNeighTable.egpNeighEntry.egpNeighAs
egp.egpNeighTable.egpNeighEntry.egpNeighInMsgs
egp.egpNeighTable.egpNeighEntry.egpNeighInErrs
egp.egpNeighTable.egpNeighEntry.egpNeighOutMsgs
egp.egpNeighTable.egpNeighEntry.egpNeighOutErrs
egp.egpNeighTable.egpNeighEntry.egpNeighInErrMsgs
egp.egpNeighTable.egpNeighEntry.egpNeighOutErrMsgs
egp.egpNeighTable.egpNeighEntry.egpNeighStateUps
egp.egpNeighTable.egpNeighEntry.egpNeighStateDowns
egp.egpNeighTable.egpNeighEntry.egpNeighIntervalHello
egp.egpNeighTable.egpNeighEntry.egpNeighIntervalPoll
egp.egpNeighTable.egpNeighEntry.egpNeighMode
egp.egpNeighTable.egpNeighEntry.egpNeighEventTrigger
```

```
snmp.snmpInPkts
snmp.snmpOutPkts
snmp.snmpInBadVersions
snmp.snmpInBadCommunityNames
snmp.snmpInBadCommunityUses
snmp.snmpInASNParseErrs
snmp.snmpInBadTypes
snmp.snmpInTooBigs
snmp.snmpInNoSuchNames
snmp.snmpInBadValues
snmp.snmpInReadOnlys
snmp.snmpInGenErrs
snmp.snmpInTotalReqVars
snmp.snmpInTotalSetVars
snmp.snmpInGetRequests
snmp.snmpInGetNexts
snmp.snmpInSetRequests
snmp.snmpInGetResponses
snmp.snmpInTraps
snmp.snmpOutTooBigs
snmp.snmpOutNoSuchNames
snmp.snmpOutBadValues
snmp.snmpOutReadOnlys
snmp.snmpOutGenErrs
snmp.snmpOutGetRequests
snmp.snmpOutGetNexts
snmp.snmpOutSetRequests
snmp.snmpOutGetResponses
snmp.snmpOutTraps
snmp.snmpEnableAuthenTraps
```

HP-UX Specific SNMP Variables

The following variables are HP-UX specific SNMP variables and are located in the `.iso.org.dod.internet.private.enterprises` subtree.

```
hp.nm.system.general.computerSystem
.computerSystemUpTime
hp.nm.system.general.computerSystem
.computerSystemUsers
hp.nm.system.general.computerSystem
.computerSystemAvgJobs1
hp.nm.system.general.computerSystem
.computerSystemAvgJobs5
hp.nm.system.general.computerSystem
.computerSystemAvgJobs15
```

```
hp.nm.system.general.fileSystem.fileSystemMounted
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemID1
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemID2
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemName
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemBlock
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemBfree
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemBavail
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemBsize
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemFiles
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemFfree
hp.nm.system.general.fileSystem.fileSystemTable
.fileSystemEntry.fileSystemDir
```

```
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacIndex
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacTransmitted
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacNotTransmitted
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacDeferred
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacCollisions
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacSingleCollisions
hp.nm.interface.ieee8023Mac.ieee8023MacTable
```

```
.ieee8023MacEntry.ieee8023MacMultipleCollisions
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacExcessCollisions
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacLateCollisions
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacCarrierLostErrors
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.eee8023MacEntry.ieee8023MacNoHeartBeatErrors
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacFramesReceived
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacUndeliverableFramesReceived
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacCRCErrors
```

```
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacAlignmentErrors
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacResourceErrors
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacControlFieldErrors
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry
.ieee8023MacUnknownProtocolErrors
hp.nm.interface.ieee8023Mac.ieee8023MacTable
.ieee8023MacEntry.ieee8023MacMulticastsAccepted
```

Traps

```
trapDestinationNum
trapDestinationTable.trapDestinationEntry
.trapDestination
```

MPE/iX Specific SNMP Variables

The following variables are MPE/iX specific SNMP variables.

```
hp.nm.system.general.mpeXLSystem.volume
.volumeMounted
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeLDEV
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeName
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeDriveType
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeSectorSize
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeType
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeCapacity
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeMPEOverhead
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeMPETransOverhead
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeMPEConfigMaxTrans
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeDirSpaceOverhead
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeFileLabelOverhead
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.TransactionMgmtOverhead
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeSpoolFileDiscUsage
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumePermFiles
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeTempFiles

hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeTotalFreeSpace
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumeLargestContigFree
hp.nm.system.general.mpeXLSystem.volume.volumeTable
.volumeEntry.volumePercentUtilized

hp.nm.system.general.mpe_XLSystem.processor
.numActive
hp.nm.system.general.mpeXLSystem.processor
.numPresent
hp.nm.system.general.mpeXLSystem.processor
.processorMIstate
hp.nm.system.general.mpeXLSystem.processor
.cpuUtilization
```

Example

All of the following examples refer to the same variable.

```
1.1.0
system.sysDescr.0
1.sysDescr..0
.1.3.6.1.2.1.1.1.0
.iso.org.dod.internet.mgmt.mib.system.sysDescr.0
.1.3.6.1.2.1.1.sysDescr.0
```

B Supported MIB Objects

The following are MIB objects supported by HP SNMP/XL.

Included in the list are:

- Standard MIB-II objects supported by MPE/XL 4.0 agents.
- Non-standard HP extended MIB objects.

Standard MIB-II Objects Supported by MPE/iX Agents

This section lists the MIB values supported by the HP SNMP/XL Agent running on the MPE/XL 4.0. Depending on the object requested, the HP SNMP/XL agent may reply with the following:

- The object's value.
- A null value (0).
- A **noSuchName** error.

Only objects that return their value are listed here.

Objects That Return Their Value

The HP SNMP/XL Agent responds with the object's value for the following objects.

System Group

```
sysDescr
sysObjectID
sysUpTime
sysContact
sysName
sysLocation
sysServices
```

Interface Group

Not all of the objects in the interface group are supported on all links.

```
ifNumber
ifTable
  ifEntry
    ifIndex
    ifDescr
    ifType
    ifMtu
    ifSpeed
    ifPhysAddress
    ifAdminStatus
    ifOperStatus
    ifInOctets
    ifInUcastPkts
    ifInNUcastPkts
    ifInDiscards
    ifInErrors
    ifOutOctets
    ifOutUcastPkts
    ifOutNUcastPkts
    ifOutDiscards
    ifOutErrors
    ifOutQLen
    ifSpecific
```

Address Translation Group

```
atTable
  atEntry
    atIfIndex
    atPhysAddress
    atNetAddress
```

IP Group

```
ipForwarding
ipInReceives
ipInHdrErrors
ipInAddrErrors
ipForwDatagrams
ipInUnknownProtos
ipInDiscards
ipInDelivers
ipOutRequests
ipOutDiscards
ipOutNoRoutes
ipReasmTimeout
ipReasmReqds
ipReasmOKs
ipReasmFails
ipFragOKs
ipFragFails
ipFragCreates
ipAddrTable
  ipAddrEntry
    ipAdEntAddr
    ipAdEntIndex
    ipAdEntNetMask
    ipAdEntBcastAddr
ipRoutingTable
  ipRouteEntry
    ipRouteDest
    ipRouteIfIndex
    ipRouteMetric1
    ipRouteMetric2
    ipRouteMetric3
    ipRouteMetric4
    ipRouteNextHop
    ipRouteType
    ipRouteProto
    ipRouteMetric5
    ipRouteInfo
```

ICMP Group

```
icmpInMsgs
icmpInErrors
icmpInDestUnreachs
icmpInTimeExcds
icmpInParmProbs
icmpInScrQuenches
icmpInRedirects
icmpInEchos
icmpInEchoReps
icmpInTimestamps
icmpInTimestampReps
icmpInAddrMasks
icmpInAddrMaskReps
icmpOutMsgs
icmpOutErrors
icmpOutDestUnreachs
icmpOutTimeExcds
icmpOutParmProbs
icmpOutScrQuenches
icmpOutRedirects
icmpOutEchoReps
icmpOutTimestampReps
icmpOutAddrMaskReps
```

TCP Group

```
tcpRtoAlgorithm
tcpRtoMin
tcpRtoMax
tcpMaxConn
tcpActiveOpens
tcpPassiveOpens
tcpAttemptFails
tcpEstabResets
tcpCurrEstab
tcpInSegs
tcpOutSegs
tcpRetransSegs
tcpConnTable
  tcpConnEntry
    tcpConnState
    tcpConnLocalAddress
    tcpConnLocalPort
    tcpConnRemAddress
    tcpConnRemPort
tcpInErrs
tcpOutRsts
```

UDP Group

```
udpInDatagrams
udpNoPorts
udpInErrors
udpOutDatagrams
udpListenerTable
  udpListenerEntry
    udpLocalAddress
    udpLocalPort
```

SNMP Group

```
snmpInPkts
snmpOutPkts
snmpBadVersions
snmpBadCommunityNames
snmpBadCommunityUses
snmpInASNParseErrors
snmpInBadTypes
snmpInTooBigs
snmpInNoSuchNames
snmpInBadValues
snmpInReadOnlys
snmpInGenErrs
snmpTotalReqVars
snmpTotalSetVars
snmpInGetRequests
snmpInGetNexts
snmpInSetRequests
snmpInGetResponses
snmpInTraps
snmpOutTooBigs
snmpOutNoSuchNames
snmpOutBadValues
snmpOutReadOnlys
snmpOutGenErrs
snmpOutGetRequests
snmpOutGetNexts
snmpOutSetRequests
snmpOutGetResponses
snmpOutTraps
snmpEnableAuthenTraps
```

ieee802MacTable

```
ieee802MacEntry
...Index
...Transmitted
...NotTransmitted
...Deferred
...Collisions
...SingleCollisions
...MultipleCollisions
```

```
...ExcessCollisions
...LateCollisions
...CarrierLostErrors
...NoHeartBeatErrors
...FramesReceived
...UndeliverableFramesReceived
...CRCErrors
...AlignmentErrors
```

Trap

```
trapDestinationNum
trapDestinationTable
  trapDestinationEntry
    trapDestination
```

Volume

```
volumeMounted
volumeTable
  volumeEntry
    volumeLDEV
    volumeName
    volumeDrivetype
    volumeSectorSize
    volumetype
    volumeCapacity
    volumeMPETransOverhead
    volumeMPEConfigMaxTrans
    TransactionMgmtOverhead
    volumeTotalFreeSpace
    volumeLargestConfigFree
    volumePercentUtilized
processor
  numActive
  numPresent
  processorMIstate
  cpuUtilization
```

Non-Standard HP Extended MIB Objects

The MIB defined in this section is consistent with the Internet Activity Board’s (IAB’s) network management strategy.

Format of Definitions

The next section contains the specification of all HP object types contained in the MIB. 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 Object Syntax defined in the SMI.

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

A 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.

Object Definitions

HP-MIB{iso org(3) dod(6) internet(1) private(4) enterprises(1) 11}

DEFINITIONS ::= BEGIN

IMPORTS

private, enterprises, OBJECT-TYPE, Counter, NetworkAddress, Gauge, FROM RFC 1155-SMI;

hp	OBJECT IDENTIFIER ::= { enterprises 11 }
nm	OBJECT IDENTIFIER ::= { hp 2 }
system	OBJECT IDENTIFIER ::= { nm 3 }
interface	OBJECT IDENTIFIER ::= { nm 4 }
snmp	OBJECT IDENTIFIER ::= { nm 13 }
general	OBJECT IDENTIFIER ::= { system 1 }
mpe_xlSystem	OBJECT IDENTIFIER ::= { general 3 }
volume	OBJECT IDENTIFIER ::= { mpe_xlSystem 1 }
processor	OBJECT IDENTIFIER ::= { mpe_xlSystem 2 }
ieee8023Mac	OBJECT IDENTIFIER ::= { interface 1 }
trap	OBJECT IDENTIFIER ::= { snmp 1 }

END

The Ieee8023Mac Group

ieee8023MacTable OBJECT-TYPE

SYNTAX: SEQUENCE OF Ieee8023MacEntry

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “A list of IEEE 802.3 Interface entries.”

::= { ieee8023Mac 1 }: +

ieee8023MacEntry OBJECT-TYPE

```
SYNTAX:Ieee8023MacEntry ::= SEQUENCE {
ieee8023MacIndex,

INTEGER, ieee8023MacTransmitted,
Counter,
ieee8023MacNotTransmitted,
Counter,
ieee8023MacDeferred,
Counter,
ieee8023MacCollisions,
Counter,
ieee8023MacSingleCollisions,
Counter,
ieee8023MacMultipleCollisions,
Counter,
ieee8023MacExcessCollisions,
Counter,
ieee8023MacLateCollisions,
Counter,
ieee8023MacCarrierLostErrors,
Counter,
ieee8023MacNoHeartBeatErrors,
Counter,
ieee8023MacFramesReceived,
Counter,
ieee8023MacUndeliverableFramesReceived,
Counter,
ieee8023MacCRCErrors,
Counter,
ieee8023MacAlignmentErrors,
Counter,
ieee8023MacResourceErrors,
Counter,
ieee8023MacControlFieldErrors,
Counter,
ieee8023MacUnknownProtocolErrors,
Counter
ieee8023MacMulticastsAccepted,
Counter,
}
```

ACCESS: read-only

STATUS: mandatory

DESCRIPTION, “An interface entry containing objects for the IEEE 802.3 networking layer.”

::= { ieee8023MacTable 1 }: +

ieee8023MacIndex OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The index value that uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of **ifIndex**.”

::= { ieee8023MacEntry 1 }: +

ieee8023MacTransmitted OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames successfully transmitted.”

::= { ieee8023MacEntry 2 }: +

ieee8023MacNotTransmitted OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames not transmitted.”

::= { ieee8023MacEntry 3 }: +

ieee8023MacDeferred OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames deferred because the medium was busy.”

::= { ieee8023MacEntry 4 }: +

ieee8023MacCollisions OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Total number of transmit attempts that were retransmitted due to collisions.”

::= { ieee8023MacEntry 5}: +

ieee8023MacSingleCollisions OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of transmit attempts that are involved in a single collision and are subsequently transmitted successfully.”

::= { ieee8023MacEntry 6}: +

ieee8023MacMultipleCollisions OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of transmit attempts that are involved in between 2 and 15 collision attempts and are subsequently transmitted successfully.”

::= { ieee8023MacEntry 7}: +

ieee8023MacExcessCollisions OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of transmit attempts that are involved in more than 15 collision attempts and are subsequently transmitted successfully.”

::= { ieee8023MacEntry 8}: +

ieee8023MacLateCollisions OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of transmit attempts aborted because a collision occurred after the allotted channel time had elapsed.”

::= { ieee8023MacEntry 9 }: +

ieee8023MacCarrierLostErrors OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of times that carrier sense was lost when attempting to transmit.”

::= { ieee8023MacEntry 10 }: +

ieee8023MacNoHeartBeatErrors OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of times no heart beat was indicated after a transmission.”

::= { ieee8023MacEntry 11 }: +

ieee8023MacFramesReceived OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames successfully received.”

::= { ieee8023MacEntry 12 }: +

ieee8023MacUndeliverableFramesReceived OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames received that were not delivered because the software buffer was overrun when frames were sent faster than they could be received.”

::= { ieee8023MacEntry 13 }: +

ieee8023MacCRCErrors OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of Cyclical Redundancy Check (CRC) errors detected.”

::= { ieee8023MacEntry 14 }: +

ieee8023MacAlignmentErrors OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames received that were both misaligned and had bad CRC.”

::= { ieee8023MacEntry 15 }: +

ieee8023MacResourceErrors OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames received that were lost due to lack of resources.”

::= { ieee8023MacEntry 16 }: +

ieee8023MacControlFieldErrors OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames received with errors in the control field.”

::= { ieee8023MacEntry 17 }: +

ieee8023MacUnknownProtocolErrors OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of frames dropped because the type field or sap field referenced an invalid protocol.”

::= { ieee8023MacEntry 18 }: +

ieee8023MacMulticastsAccepted OBJECT-TYPE

SYNTAX: Counter

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of accepted multicast addresses.”

::= { ieee8023MacEntry 19 }: +

The Volume Group

volumeMounted OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The number of volumes that are currently mounted on the system.”

::= { volume 1 }

volumeTable OBJECT-TYPE

SYNTAX: SEQUENCE OF volumeEntry

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Volume table.”

::= { volume 2 }

volumeEntry OBJECT-TYPE

SYNTAX: volumeEntry

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Each entry contains objects that define the volume.”

::= { volumeTable 1 }

```
volumeEntry ::= SEQUENCE {
  volumeLDEV,
      INTEGER,
  volumeName,
      DisplayString,
  volumeDriveType,
      DisplayString,
  volumeSectorSize,
      INTEGER
  volumeType,
      INTEGER
  volumeCapacity,
      INTEGER
  volumeMPEOverhead,
      INTEGER
  volumeMPETransOverhead,
      INTEGER
  volumeMPEConfigMaxTrans,
      INTEGER
  volumeDirSpaceOverhead,
      INTEGER
  volumefileLabelOverhead,
      INTEGER
  volumeTransactionMgmtOverhead,
      INTEGER
  volumeSpoolFileDiscUsage,
      INTEGER
  volumePermFiles,
      INTEGER
  volumeTempFiles,
      INTEGER
  volumeTotalFreeSpace,
      INTEGER
  volumeLargestContigFree,
      INTEGER
  volumePercentUtilized,
      INTEGER
}
```

volumeLDEV OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The logical device number for the volume.”

::= { volumeEntry 1 }

volumeName OBJECT-TYPE

SYNTAX: DisplayString

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: This is the volume set name combined with the member name that uniquely distinguishes the actual volume on the system.:

::= { volumeEntry 2 }

volumeDriveType OBJECT-TYPE

SYNTAX: DisplayString

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The type of the actual hardware device, e.g. HP7935.”

::= { volumeEntry 3 }

volumeSectorSize OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The logical sector size of the volume in bytes.”

::= { volumeEntry 4 }

volumeType OBJECT-TYPE

```
SYNTAX: INTEGER {
      system(1),
      nonSystem(2)
      }
```

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The type of volume set.”

::= { volumeEntry 5 }

volumeCapacity OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The capacity of the volume in sectors.”

::= { volumeEntry 6 }

volumeMPEOverhead OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The total overhead which consists of everything on a volume that is not set aside for file space use. This includes volume label, file label table, directory, volume set information table, free space map, transient space, and transaction management overhead.”

::= { volumeEntry 7 }

volumeMPETransOverhead OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The total MPE transient space overhead for the volume.”

::= { volumeEntry 8 }

volumeConfigMaxTrans OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The configured maximum transient space for the volume.”

::= { volumeEntry 9 }

volumeDirSpaceOverhead OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The directory space overhead that is reserved for accounting information.”

::= { volumeEntry 10 }

volumeFileLabelOverhead OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The file label overhead for this volume.”

::= { volumeEntry 11 }

volumeTransactionMgmtOverhead OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The transaction management overhead for this volume.”

::= { volumeEntry 12 }

volumeSpoolFileDiscUsage OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The spool file disk usage which consists of the volume space that is used by hidden spool files that are not part of the permanent file space.”

::= { volumeEntry 13 }

volumePermFiles OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The space used for permanent files on this volume.”

::= { volumeEntry 14 }

volumeTempFiles OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The space used for temporary files on this volume.”

::= { volumeEntry 15 }

volumeTotalFreeSpace OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The total free space for the volume.”

::= { volumeEntry 16 }

volumeLargestContigFree OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The largest contiguous free space area on the volume.”

::= { volumeEntry 17 }

volumePercentUtilized OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The percent of the volume currently being used for file storage and operating system overhead.”

::= { volumeEntry 18 }

The Processor Group

numActive OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Number of processors currently active in the system. A processor is considered active if it is capable of being dispatched.”

::= { processor 1 }

numPresent OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The number of processors physically present in the system.”

::= { processor 2 }

processorMIstate OBJECT-TYPE

```
SYNTAX: INTEGER {
      disabled(0),
      enabled(1)
      }
```

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Setting this object to 1 will result in the measurement interface being turned on for the global processor statistics which will increase the amount of CPU used by the SNMP/XL Agent. Setting this object to 0 will cause the measurement interface to be disabled for the global processor statistics. When the measurement interface is enabled, the **cpuUtilization** object described below may be obtained.”

::= { processor 3 }

cpuUtilization OBJECT-TYPE

SYNTAX: INTEGER

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The overall CPU utilization percentage on the system. If the system has more than one processor, the value returned is averaged out over all of the processors that are present. The measurement interface must be enabled in order to get a valid value returned for this object (see above object). The number returned is the percentage of the CPU that was used since the last time the number was sampled. This value is consistent with various HP performance tools such as Glance/XL. If the measurement interface is not enabled, the value returned will be 0.”

::= { processor 4 }

The Trap Group

trapDestinationNum   OBJECT-TYPE

SYNTAX: Gauge

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “The number of trap destinations.”

::= { trap 1 }: +

trapDestinationTable   OBJECT-TYPE

SYNTAX: SEQUENCE OF TrapDestinationEntry

ACCESS: read-write

STATUS: mandatory

DESCRIPTION: “List of addresses to which the agent sends traps.”

::= { trap 2 }: +

trapDestinationEntry   OBJECT-TYPE

```
SYNTAX: TrapDestinationEntry ::= SEQUENCE {
      trapDestination
      NetworkAddress
      }
```

ACCESS: read-write

STATUS: mandatory

DESCRIPTION: “Each entry contains the address to which the agent sends traps.”

::= { trapDestinationTable 1 }: +

trapDestination   OBJECT-TYPE

SYNTAX: NetworkAddress

ACCESS: read-only

STATUS: mandatory

DESCRIPTION: “Address to which the agent sends traps.”

::= { trapDestinationEntry 1 }: +

Identification of OBJECT Instances for Use with the SNMP

The names for all object types in the HP-MIB are defined explicitly either in the Internet-standard MIB or in other documents which conform to the naming conventions of the Structure of Management Information (SMI). The SMI requires that conformant management protocols define mechanisms for identifying individual instances of those object types for a particular network element.

Each instance of any object type defined in the MIB is identified in SNMP operations by a unique name called its “variable name.” In general, the name of an SNMP variable is an OBJECT IDENTIFIER of the form x.y, where x is the name of a non-aggregate object type defined in the MIB and y is an OBJECT IDENTIFIER fragment that, in a way specific to the named object type, identifies the desired instance.

The type-specific naming of object instances is defined below for a number of classes of object types. Instances of an object type to which none of the following naming conventions are applicable are named by OBJECT IDENTIFIERs of the form x.0, where x is the name of said object type in the MIB definition.

For example, suppose one wanted to identify an instance of the variable `sysDescr` in the Internet-standard MIB. The object class for `sysDescr` would look like:

```
iso org dod internet mgmt mib system sysDescr
 1   3   6     1       2    1    1       1
```

Hence, the object type, x, would be `1.3.6.1.2.1.1.1` to which is appended an instance sub-identifier of `0`. That is, `1.3.6.1.2.1.1.1.0` identifies the one and only instance of `sysDescr`.

ieee8023MacTable Object Type Names

The name of an ethernet-like interface, s, is the OBJECT IDENTIFIER value of the form i, where i has the value of that instance of the `ifIndex` object type associated with s.

For each object type, t, for which the defined name, n, has a prefix of `ieee8023MacTableEntry`, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.s, where s is the name of the ethernet-like interface about which i represents information.

For example, suppose one wanted to identify the instance of the variable `ieee8023MacNotTransmitted`, associated with interface 2. Accordingly, `ieee8023MacNotTransmitted.2` would identify the desired instance.

trapDestinationTable Object Type Names

The name of a trap destination, x, is the OBJECT IDENTIFIER of the form a.b.c.d such that a.b.c.d is the value (in the familiar “dot” notation) of that instance of the `trapDestination` object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of `trapDestinationEntry`, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the `trap destination` about which i represents information.

For example, suppose one wanted to find out if the agent is sending traps to `15.2.112.113`. Accordingly, `trapDestination.15.2.112.113` would identify the desired instance. Note that if the entry does not exist, trapDestination will return the value `0.0.0.0`, rather than noSuchName.

Volume Table Object Type Names

The name of a logical disk drive volume, s, is the OBJECT IDENTIFIER value of the form i, where i has the value of that instance of the `volumeLDEV` object type associated with s.

For each object type, t, for which the defined name, n, has a prefix of `volumeEntry`, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.s, where s is the name of the logical disk drive volume about which i represents information.

For example, suppose one wanted to identify the instance of the variable `volumeName`, associated with volume 2. Accordingly, `volumeName.2` would identify the desired instance.

C SNMP Command Examples

The following are examples of SNMP commands and the use of instance specifications.

```
Below are examples of incorrect (incomplete) MIB specifications taht lead to
an error:

:SNMPGET ‘public’,system.sysdescr
The variable specified does not exist. (SNMPINFO 1802).

:SNMPGET 0, ‘public’,hp.nm.system.general.mpexlsystem.processor
The variable specified does not exist. (SNMPINFO 1802).

The correct syntax is to specify both the STANDARDIZED SPECIFICATION of the
variable AND the INSTANCE SPECIFICATION used to distinguish different
instances of the object in the same MIB:

:SNMPGET 0, ‘public’,system.sysdescr.0
Octet String: HP3000 SERIES 967, MPE XL version ...

If the INSTANCE SPECIFICATION is not known, use the SNMPWALK-command:

:SNMPWALK 0, ‘public’,hp.nm.system.general.mpexlsystem.processor
Name: hp.nm.system.general.mpeXLSystem.processor.numActive.0
Integer: 1

Name: hp.nm.system.general.mpeXLSystem.proessor.numPresent.0
Integer: 1

Name: hp.nm.system.general.mpeXLSystem.processor.processorMIstate.0
Integer: (disabled) 2

Name: hp.nm.system.general.mpeXLSystem.processor.cupUtilization.0
Integer: 0
```

D Time Zones

This appendix contains a list of commonly used time zones and the TZ environment variable strings that correspond to these time zones. You use the TZ strings in the SNMPUDC.NET.SYS file to allow SNMP to adjust to specific time zones.

In Table D-1, column 1 contains the time zone name followed by the Daylight Savings time zone name, if appropriate. Column 2 lists the geographic locations associated with this time zone. Column 3 contains the TZ environment variable string for the time zone.

Table D-1 Time-Zone Codes

Time Zone Name/ Daylight Savings Name	Geographic Locations	TZ Variable
Hawaiian Standard Time/ Hawaiian Daylight Time	United States: Hawaii	HST10
Aleutian Standard Time/ Aleutian Daylight Time	United States: Alaska (parts)	AST10ADT
Yukon Standard Time/ Yukon Daylight Time	United States: Alaska (parts)	YST9YDT
Pacific Standard Time/ Pacific Daylight Time	Canada: British Columbia	PST8PDT — Canada
Pacific Standard Time/ Pacific Daylight Time	United States: California, Idaho (parts), Nevada, Oregon (parts), Washington	PST8PDT
Mountain Standard Time/ Mountain Daylight Time	Canada: Alberta, Saskatchewan (parts)	MST7MDT — Canada
Mountain Standard Time/ Mountain Daylight Time	United States: Colorado, Idaho (parts), Kansas (parts), Montana, Nebraska (parts), New Mexico, North Dakota (parts), Oregon (parts), South Dakota (parts), Texas (parts), Utah, Wyoming	MST7MDT
Mountain Standard Time	United States: Arizona	MST7
Central Standard Time/ Central Daylight Time	Canada: Manitoba, Ontario (parts), Saskatchewan (parts)	CST6CDT - Canada
Central Standard Time/ Central Daylight Time	United States: Alabama, Arkansas, Florida (parts), Illinois, Iowa, Kansas, Kentucky (parts), Louisiana, Michigan (parts), Minnesota, Mississippi, Missouri, Nebraska, North Dakota, Oklahoma, South Dakota, Tennessee (parts), Texas, Wisconsin	CST6CDT

Table D-1 Time-Zone Codes

Time Zone Name/ Daylight Savings Name	Geographic Locations	TZ Variable
Eastern Standard Time/ Eastern Daylight Time	Canada: Ontario (parts), Quebec (parts)	EST5EDT — Canada
Eastern Standard Time/ Central Daylight Time	United States:Indiana (most)	EST6CDT
Eastern Standard Time/ Eastern Daylight Time	United States: Connecticut, Delaware, District of Columbia, Florida, Georgia, Kentucky, Maine, Maryland, Massachusetts, Michigan, New Hampshire, new Jersey, New York, North Carolina, Ohio, Pennsylvania, Rhode Island, South Carolina, Tennessee (parts), Vermont, Virginia, West Virginia	EST5EDT
Atlantic Standard Time/ Atlantic Daylight Time	Canada: Newfoundland (parts), Nova Scotia, Prince Edward Island, Quebec (parts)	AST4ADT
Newfoundland Standard Time/Newfoundland Daylight Time	Canada: Newfoundland (parts)	NST3:30NDT
Western European Time/ Western European Time Daylight Savings Time	Great Britain, Ireland	WET0WETDST
Portuguese Winter Time/ Portuguese Summer Time		PWT0PST
Mitteleuropaeische Zeit/ Mitteleuropaeische Sommerzeit		MEZ-1MESZ
Middle European Time/ Middle European Time Daylight Savings Time	Belgium, Luxembourg, Netherlands, Denmark, Norway, Austria, Poland, Czechoslovakia, Sweden, Switzerland, DDR, DBR, France, Spain, Hungary, Italy, Yugoslavia	MET-2METDST
South Africa Standard Time/ South Africa Daylight Time		SAST-2SADT
Japan Standard Time	Japan	JST-9
Australian Western Standard Time	Australia: Western Australia	WST-8:00
Australian Central Standard Time	Australia: Northern Territory	CST-9:30

Table D-1 Time-Zone Codes

Time Zone Name/ Daylight Savings Name	**Geographic Locations**	**TZ Variable**
Australian Central Standard Time/Australian Central Daylight Time	Australia: South Australia	CST-9:30CDT
Australian Eastern Standard Time	Australia: Queensland	EST-10
Australian Eastern Standard Time/Australian Eastern Daylight Time	Australia: New South Wales, Victoria, Tasmania	EST-10EDT
New Zealand Standard Time/ New Zealand Daylight Time		NZST-12NZDT

To set the TZ variable to a specific time zone, use the MPE SETVAR command. For example, if you want to specify the Pacific Time Zone in the United States, enter the following MPE command:

```
setvar TZ “PST8PDT”
```

In the TZ string above, PST represents the time zone abbreviation for Pacific Standard Time, 8 represents the difference in hours between Pacific Standard Time and Greenwich Mean Time (GMT), and PDT represents the Daylight Savings time zone abbreviation for Pacific Daylight Time.

Index