



**Simple Network Management Protocol in  
the MPS Environment  
(Software Release 5.5.0)**

**Publication#:** P0607269

**Document Release:** 1.1

**Release Date:** September 24, 2004

## Important Notice

Nortel Networks reserves the right to make changes to the contents of this publication including functions and specifications identified herein without notice.



The material contained in this document is intended for Nortel Networks personnel and licensed customers with a non-disclosure agreement or standard contract.

In the absence of a written agreement to the contrary, Nortel Networks assumes no liability for applications assistance, customer's product/application/concepts, or infringements of patents or copyrights of third parties arising from the use of systems and architectures described herein. Nor does Nortel Networks warrant or represent that any license, either expressed or implied, is granted under any patent right, copyright, or other combination of technology, architecture, or software as might be or is already in use.

This document should not be reproduced, disseminated, or otherwise disclosed without prior written consent from an officer of Nortel Networks.

This document has been copyrighted by Nortel Networks and may not be duplicated.

© 2003 Nortel Networks, All Rights Reserved

---

## Table of Contents

<b>Preface</b> .....	<b>i</b>
Scope .....	ii
Intended Audience .....	ii
How to Use This Manual .....	ii
Organization of This Manual .....	iii
Conventions Used in This Manual .....	iv
Solaris and Windows 2000 Conventions .....	v
Trademark Conventions .....	vi
 <b>Introduction</b> .....	 <b>1</b>
Simple Network Management Protocol .....	2
Network Management Concepts .....	2
Management Information Base (MIB) .....	3
Structure of Management Information .....	5
Protocol .....	6
Sample Manager and Agent Interaction .....	7
get-request .....	7
get-next-request .....	7
set-request .....	8
get-bulk-request .....	8
get-response .....	8
inform-request .....	9
trap .....	9
SNMP and MPS Integration .....	10
Nortel Networks, Periphonics Enterprise MIB .....	10
PeriView and SNMP Integrations .....	11
Understanding Components .....	12
 <b>Installation and Configuration</b> .....	 <b>13</b>
SNMP Installation .....	14
Package Licensing .....	14
Defining Trap Destinations Automatically .....	15
In Solaris .....	15
In Windows 2000 .....	16
Verify Agents' Response .....	17
Installation Upgrades .....	17
PERIsnmp Upgrade .....	17
Solaris Upgrade .....	17
SNMP System Startup .....	18
Solaris Systems .....	18
Windows 2000 Systems .....	18
MIB Files .....	20

Configuration Files .....	21
Environment Variables .....	22
Editing .....	22
In Solaris .....	22
In Windows 2000 .....	22
Master/Subagent Related .....	23
Manager Related .....	23
Command Line Options .....	23
In Solaris .....	23
In Windows 2000 .....	24
Configuring snmpd.cnf .....	25
Default MIB-II Values .....	25
Security Access Rights (View Subtrees) .....	26
Groups and Access Rights .....	28
Community Strings .....	30
Assigning Principles to Groups .....	32
Defining Notifications .....	34
Defining Notification Target Entries .....	34
Defining Target Parameters .....	36
Error Logging .....	37
Executables .....	38
Agents .....	38
SNMP Command Line Utilities .....	39
 <b>MPS Functions and Administration .....</b>	 <b>41</b>
SNMP Utilization .....	42
Communication .....	42
System Architecture .....	43
MIB Variables .....	44
Access .....	44
Status .....	45
Types .....	46
Instance Identification .....	46
Direct Referencing .....	46
Table Referencing .....	46
SNMP Applied .....	48
Getting a Value .....	48
Setting a Value .....	49
Viewing Traps .....	49
The MIB Applied .....	51
Subtrees .....	51
Events Group Notifications Subtree .....	51
Component Events Alarms Subtree .....	52
VRU Subtree .....	54
Line Subtree .....	55

Span Table Subtree .....	56
Applications Subtree .....	57
Applications Stats Subtree .....	57
Host Subtree .....	58
LU Subtree .....	59
Components Subtree .....	60
<b>PeriView/PERIsnmp Mappings .....</b>	<b>61</b>
Mappings .....	62
MIB-II .....	62
Enterprise MIB .....	63
Events Group Notifications Subtree .....	63
Component Events Alarms Subtree .....	63
VRU Subtree .....	66
Line Subtree .....	66
Span Table Subtree .....	69
Applications Subtree .....	70
Applications Stats Subtree .....	71
Host Subtree .....	71
LU Subtree .....	71
Components Subtree .....	73
<b>Periphonics Enterprise MIB Reference .....</b>	<b>75</b>
Basic Tree Structure .....	76
Detailed Component Structure .....	80
<b>Troubleshooting/FAQs .....</b>	<b>113</b>
Troubleshooting .....	114
Frequently Asked Questions (FAQs) .....	119
<b>Request For Comment (RFC) .....</b>	<b>123</b>
Using the Service .....	124
Tables of RFCs .....	124

Index ..... 127

# **Preface**

## Scope

This document contains an overview and description of the Simple Network Management Protocol, commonly known by the acronym SNMP, its developmental structure and its function in managing network components. The Nortel Networks enterprise Management Information Base (MIB), its relationship to SNMP, and its practical use in managing the Nortel Networks Media Processing Server (MPS) Series system is discussed in detailed terms. Comparisons to analogous tools and functions in Nortel's PeriView MPS management Graphical User Interface (GUI) are also included.

## Intended Audience

This guide assumes that the MPS system operators have completed an on-site system familiarization training program conducted as part of the initial system installation. In addition, they should be familiar with other site-specific operating procedures relating to the MPS that are due to specific application functions performed by the MPS and other equipment to which the MPS may be connected. Basic knowledge of the Solaris and/or Windows 2000 operating system(s) is also assumed.

## How to Use This Manual

This manual uses many standard terms relating to computer system and software application functions. However, it contains some terminology that can only be explained in the context of the MPS system. Refer to the *Glossary of MPS Terminology* for definitions of MPS specific terms.

Initially, you should read this manual at least once, from start to finish. Later, you can use the Table of Contents to locate topics of interest for reference and review.

If you are reading this document online, use the cross-reference links (shown in [blue](#)) to quickly locate related topics. <LEFT> click once with your mouse while positioned with your cursor over the cross-reference link. Click on any point in a Table of Contents entry to move to that topic. Click on the page number of any Index entry to access that topic page.

To familiarize yourself with various specialized textual references within the manual, see [Conventions Used in This Manual on page iv](#).



Periphonics is part of Nortel Networks. The name Periphonics, and variations thereof, appear in this manual only where referred to in a product. (For examples, a PeriProducer application, the PERImps package, the **perirev** command, etc.)



## Organization of This Manual

This manual contains separate chapters that describe SNMP, its application and functions within the MPS environment, its use as a “generic” manager and in relation to GUI-oriented packages such as PeriView. To this end, a chapter has been developed to familiarize the user with correlating functions between such GUI tools and objects in the MIB. Chapter/appendix overviews are as follows:

### **Chapter 1 — Introduction**

Provides broad explanations, concepts, definitions, and examples of the application and integration of SNMP and the MIB in the MPS environment.

### **Chapter 2 — Installation and Configuration**

Discusses the installation, system startup, environment variables, and configuration of SNMP. Provides configuration default values and the process to modify these values for those users who wish to alter community strings, default MIB-II values, MIB views, access control, traps, and transports parameters. Also discusses command line utilities and options.

### **Chapter 3 — MPS Functions and Administration**

Describes in generic terms the functions and administration of the MPS which are controlled or monitored by the SNMP manager in a non-GUI environment.

### **Chapter 4 — PeriView Mappings**

Details cross-references between the Nortel Networks, Periphonics Enterprise MIB components and analogous tools and functions in PeriView.

### **Appendix A — Periphonics Enterprise MIB Reference**

Illustrates the Nortel Networks, Periphonics Enterprise MIB in both tree form and detailed textual form.

### **Appendix B — Troubleshooting/FAQs**

Contains a troubleshooting guide, a list of commonly asked questions and the answers thereto.



### **Appendix C — Utilizing Request For Comment (RFC)**

Contains information on using and accessing RFCs and their relevance to the SNMP environment.



### Conventions Used in This Manual

This manual uses different fonts and symbols to differentiate between document elements and types of information. These conventions are summarized in the following table.

Conventions Used in This Manual Sheet 1 of 2

Notation	Description
Normal text	Normal text font is used for most of the document.
<i>important term</i>	The Italics font is used to introduce new terms, highlight meaningful words or phrases, or distinguish specific terms from nearby text.
<b>system command</b>	This font indicates a system command and/or its arguments. Such keywords are to be entered exactly as shown (i.e., users are not to fill in their own values).
<b>command, condition and alarm</b>	Command, Condition and Alarm references appear on the screen in bold text and reference the <i>Command Reference Manual</i> , the <i>Condition Reference Manual</i> , or the <i>Alarm Reference Manual</i> . Refer to these documents for detailed information about <b>Commands</b> , <b>Conditions</b> , and <b>Alarms</b> .
file name / directory	This font is used for highlighting the names of disk directories, files, and extensions for file names. It is also used to show displays on text-based screens (e.g., to show the contents of a file.)
on-screen field	This font is used for field labels, on-screen menu buttons, and action buttons.
<KEY NAME>	A term that appears within angled brackets denotes a terminal keyboard key, a telephone keypad button, or a system mouse button.
<i>Book Reference</i>	This font indicates the names of other publications referenced within the document.
cross reference	A cross reference appears on the screen in blue text. Click on the cross reference to access the referenced location. A cross reference that refers to a section name accesses the first page of that section.
	The Note icon identifies notes, important facts, and other keys to understanding.
	The Caution icon identifies procedures or events that require special attention. The icon indicates a warning that serious problems may arise if the stated instructions are improperly followed.

## Conventions Used in This Manual Sheet 2 of 2

Notation	Description
	The flying Window icon identifies procedures or events that apply to the Windows 2000 operating system only. <sup>a</sup>
	The Solaris icon identifies procedures or events that apply to the Solaris operating system only. <sup>b</sup>
<p>a. Windows 2000 and the flying Window logo are either trademarks or registered trademarks of the Microsoft Corporation.</p> <p>b. Solaris is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.</p>	

## Solaris and Windows 2000 Conventions

This manual depicts examples (command line syntax, configuration files, and screen shots) in Solaris format. In certain instances Windows 2000 specific commands, procedures, or screen shots are shown where required. The following table lists examples of general operating system conventions to keep in mind when using this manual with either the Solaris or Windows 2000 operating system.

	Solaris	Windows 2000
Environment	<code>\$MPSHOME</code>	<code>%MPSHOME%</code>
Paths	<code>\$MPSHOME/common/etc</code>	<code>%MPSHOME%\common\etc</code>
Command	<code>&lt;command&gt; &amp;</code>	<code>start /b &lt;command&gt;</code>

### Trademark Conventions

The following trademark information is presented here and applies throughout for third party products discussed within this manual. Trademarking information is not repeated hereafter.

UNIX<sup>®</sup> is a registered trademark of The Open Group in the U.S. and other countries.

Solaris is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.

Microsoft, Windows, Windows 2000, Internet Explorer, and the Flying Windows logo are either trademarks or registered trademarks of Microsoft Corporation.

Netscape<sup>®</sup> and Netscape Navigator<sup>®</sup> are registered trademarks of Netscape Communications Corporation in the United States and other countries. Netscape's logos and the Netscape product and service names are also trademarks of Netscape Communications Corporation, which may be registered in other countries.



# 1

## Introduction

This chapter covers:

1. Simple Network Management Protocol
2. SNMP and MPS Integration

The Simple Network Management Protocol (SNMP) provides a uniform method for managing a network of diverse products from multiple vendors. A Nortel Networks MPS system with the Periphonics Enterprise Management Information Base (MIB) with SNMP can be integrated into existing TCP/IP networks controlled by third-party management tools. In addition, the Nortel Networks Periphonics Enterprise MIB can be used to develop custom MPS management applications.

## Simple Network Management Protocol

### Network Management Concepts

Network management involves management personnel utilizing a workstation to communicate with one or more network elements to be monitored and controlled. The architecture of network management may be broken into 4 core components:

- **Management Station:** Network management personnel monitor and control the network from the network management station. The management station communicates with network elements to do data analysis, fault recovery et al., and is responsible for generating requests and receiving responses and event reports. In SNMP, the Management Station is referred to as the manager (the manager would be considered a client in a client/server architecture). The Management Station is analogous to a PerView workstation.
- **Management Agent:** A Management Agent is responsible for receiving and processing the Management Station requests for information about a network element, responding to requests, and generating event reports. In SNMP, a Management Agent is referred to as an agent (agents would be considered servers in a client/server architecture).
- **Management Information Base (MIB):** A MIB is a collection of objects that describe one or more network elements. A MIB can be considered to be a form of database schema, which contains the set of system components that can be queried or controlled. For additional information, see [Management Information Base \(MIB\)](#) on page 3.
- **Network Management Protocol:** A Network Management Protocol is the means by which the Management Station and Agents communicate.

In summary, SNMP provides a network management protocol for allowing managers (clients) to perform status and control functions by querying agents (servers) about the network elements described in various MIBs (databases).

## Management Information Base (MIB)

The Management Information Base (MIB) specifies the variables that describe a network element. These variables are considered to be common knowledge to the Management Station and Agent. This “commonality” is achieved by requiring the MIB to be specified in a standard syntax called Abstract Syntax Notation One (ASN.1). Typically a Management Station will query MIB variables in order to determine information about a network element and its state. The Management Station may also change an MIB variable in order to modify a property of the network element or its state. Through the act of modifying MIB variables, the Management Station exerts control over network elements.

The MIB is a database of objects that work in attribute/value pairs and whose hierarchy is a tree-like structure. The definition of SNMP requires a single standard MIB which can be extended by vendors to include product specific subtrees to the base MIB. In [RFC 1213](#) a standard subtree called MIB-II is defined and must be implemented by all Management Agents (for additional information, see [Appendix Using the Service on page 124](#)). MIB-II is the core branch of SNMP, where information about attributes can be set and/or queried. MIB-II is divided into the following groups:

- **system:** Overall system information
- **interfaces:** Information about the interfaces from system to an attached subnetwork
- **at:** Address-translation table for internet-to-subnet address mapping
- **ip:** Information about the Internet Protocol on this system
- **icmp:** Information about the Information Control Protocol on this system
- **tcp:** Information about the Transmission Control Protocol on this system
- **udp:** Information about the User Datagram Protocol on this system
- **egp:** Information about External Gateway Protocol on this system
- **transmission:** Information about transmission schemes and access protocols at each system interface
- **snmp:** Information about the Simple Network Management Protocol on this system.

In addition to MIB-II, there is a subtree called enterprises. This subtree allows vendors to extend the MIB to support their products.

```

+---internet(1)
|
+--- mgmt(2)
|
|   +---mib-2(1)
|   |
|   |   +--- system(1)
|   |   |
|   |   |   :
|   |   |
|   |   +--- interfaces(2)
|   |   |
|   |   |   :
|   |   |
|   |   +--- at(3)
|   |   |
|   |   |   :
|   |   |
|   |   +--- ip(4)
|   |   |
|   |   |   :
|   |   |
|   |   +--- icmp(5)
|   |   |
|   |   |   :
|   |   |
|   |   +--- tcp(6)
|   |   |
|   |   |   :
|   |   |
|   |   +--- udp(7)
|   |   |
|   |   |   :
|   |   |
|   |   +--- egp(8)
|   |   |
|   |   |   :
|   |   |
|   |   +--- transmission(10)
|   |   |
|   |   |   :
|   |   |
|   |   +--- snmp(11)
|   |   |
|   |   |   :
|   |   |
|   |
|   +--- private(4)
|   |
|   |   +--- enterprises(1)
|   |   |
|   |   |   :
|   |   |

```

Vendors extend the MIB via the enterprise branch to define their own products so management tools can recognize them. For additional information, see [Nortel Networks, Periphonics Enterprise MIB](#) on page 10.



## Structure of Management Information

The Structure of Management Information (SMI) defines a set of common structures and an identification scheme used to reference variables within the MIB. SNMP allows for two simple data types - scalar and two dimensional arrays of scalar (tables). SMI data types can be divided into two main components - Universal Types and Application Types. In turn, each encompasses the following:

### Universal Types:

- **INTEGER:** May or may not have restrictions. These restrictions include ranges (ie TCP/IP port numbers are between 0 and 65535) or enumerated types (ie State of a system unknown(1), up(2), down(3), etc...).
- **OCTET STRING:** String of 0 or more 8-bit bytes. These strings are not NULL terminated.
- **OBJECT IDENTIFIER:** Sequence of decimal separated integers used to identify a variable within the MIB.
- **NULL:** Used to indicate that the variable has no value.
- **SEQUENCE:** Similar to a structure in C.
- **SEQUENCE OF:** Defines a vector whose elements are a simple data type. This is used to define tables within SNMP.

### Application Types:

- **DisplayString:** String of 0 or more 8-bit bytes where each byte must be a character from NVT ASCII set. The string length must be [0...255].
- **IpAddress:** Octet string of length 4 specifying the IP address. Each byte represents a byte of the IP address.
- **Counter:** Non-negative integer whose value increases from 0 to  $2^{32}-1$ . When the maximum value is reached, it will reset to zero.
- **Gauge:** Non-negative integer which may increase or decrease. The range of values is 0 to  $2^{32}-1$ .
- **TimeTicks:** Counter representing the time in hundreds of seconds since some event.

SMI also specifies how MIB variables are referenced. Since MIB variables are arranged in a tree structure, they are referenced as a set of node names (separated by periods) from the root of the tree traversing the path to the object being referenced. For example, the reference (or object identifier or even OID) for the enterprises portion of the MIB is iso.org.dod.internet.private.enterprises. Each object identifier also has a numeric representation. The enterprises example above in numeric representation is 1.3.6.1.4.1.

It is these sequences of integers that both the manager and agent use to uniquely identify MIB variables.

### Protocol

The SNMP protocol is the method via which managers and agents communicate. Currently only version 1 of SNMP (SNMPv1) is a standard protocol. All other versions are a draft or a proposed standard. Typically the manager makes requests of the agent to get or modify the value of a variable defined in the MIB and waits for the agent's response. The manager receives responses from the agent as well as process event reports generated by the agent. In the case of the MPS implementation, the SNMP agent is bilingual (v1 and v2c supported); that is, the agent detects the version of SNMP stipulated by the manager, and automatically replies in kind.

SNMPv1 defines five types of messages that are exchanged between the manager and agent.

- **get-request:** Message sent from the manager to the agent asking for the value of one or more variables.
- **get-next-request:** Message sent from the manager to the agent asking for the value of the next variable after one or more variables.
- **set-request:** Message sent from the manager to the agent asking to change the value of one or more variables.
- **get-response:** Message sent from the agent to the manager in response to a get-request, get-next-request, or set-request.
- **trap:** Unsolicited message sent from the agent to the manager to notify the occurrence of some asynchronous event. SNMPv1 defines 6 traps: coldStart, warmStart, linkDown, linkUp, authenticationFailure, egpNeighborLoss, and enterpriseSpecific. The enterpriseSpecific allows organizations to define their own traps. Traps are analogous to alarms in the MPS environment.

In addition, SNMPv2c extends the v1 message types to include:

- **get-bulk-request:** Allows retrieval of large numbers of variables at one time (similar to numerous individual iterations of get-next-requests).
- **inform-request:** Used for manager-to-manager communications.

## Sample Manager and Agent Interaction

Consider the MIB-II system subtree which is defined as:

```
system(mib-2) (1)
|
+--- sysDescr(1)
|
+--- sysObjectID(2)
|
+--- sysUpTime(3)
|
+--- sysContact(4)
|
+--- sysName(5)
|
+--- sysLocation(6)
|
+--- sysServices(7)
```

### get-request

Suppose the network manager wanted a description of the network entity. The network manager would then send a get-request message on the variable:

```
iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0
```

The agent would then send a get-response message indicating that this variable contained the value "Nortel Networks MPS Network."

### get-next-request

Suppose the network manager wanted to know what variable came after the variable sysContact. The network manager would send a get-next-request message on the variable:

```
iso.org.dod.internet.mgmt.mib-2.system.sysContact.0
```

The agent would send a get-response message indicating the next variable after sysContact is iso.org.dod.internet.mgmt.mib-2.system.sysName.0 and its current value is "peri.com".

The above example is not really useful in itself, since the sysName variable is directly accessible. In practical use, an SNMP manager would use get-next-request to iterate through MIB tables. MIB tables are much like arrays; they allow the specification of sets of similarly defined elements.

An SNMP manager could also use get-next-request to iterate through the list of variables supported by an SNMP agent to see if a particular facility were available.

### set-request

Suppose that the location of the MPS network has moved to the second floor. The network manager now needs to update the value of the sysLocation variable. He would indicate the new value is “4000 Vets Hwy 2nd Floor” by sending a set-request message on the variable:

iso.org.dod.internet.mgmt.mib-2.system.sysLocation.0

The agent would send a get-response message indicating that the new value of sysLocation has been changed to “4000 Vets Hwy 2nd Floor”.

More importantly, an agent can invoke actions when a variable is modified. For example, if an enterprise MIB contained a variable called sysState, which was defined as having the enumerated values sysUp(1) and sysDown(2), the agent might use a set-request to bring the system up or down depending on what value was set by the manager.

It is this mechanism that the Nortel Networks SNMP implementation uses to control MPS systems via the Nortel Networks, Periphonics Enterprise MIB.

### get-bulk-request

Available when using SNMPv2c only, this request allows the manager to poll for large blocks of data efficiently. If the manager wanted to iterate through the MIB-II system subtree to search for a particular set of data, the manager would issue a get-bulk-request on the variable:

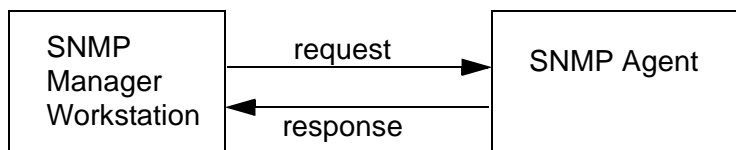
iso.org.dod.internet.mgmt.mib-2.system

The agent would return all the variables on this subtree in one response. The manager could then determine if the necessary information were available and at what variable.

### get-response

This message is the generic reply sent by the agent to the manager in response to any get-request, get-next-request, or set-request.

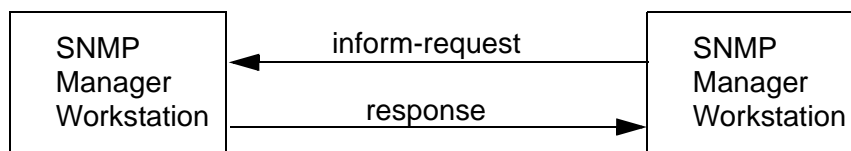
#### Sample request-response interaction



### inform-request

Available when using SNMPv2c only, this message is used for manager-to-manager communications.

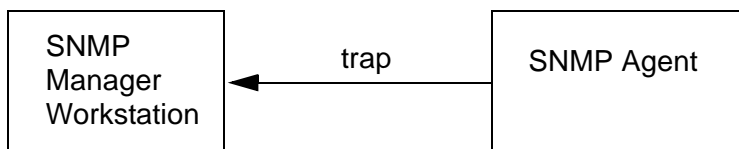
#### Sample inform-request interaction



### trap

The messages described so far allow the manager to actively poll the agent for status and control information, but it is also desirable for the agent to send unsolicited messages to the manager in case of unexpected internal conditions that need attention. SNMP defines the trap message for this purpose. No formal mechanism exists in SNMPv1 for managers to request to be sent traps. Trap destinations can be set automatically by using the `trapcfg.pl` portion of the installation script (see [SNMP Installation on page 14](#) and [Defining Trap Destinations Automatically on page 15](#)); otherwise the configuration file (`snmpd.conf`) will have to be edited to add, delete, or change the manager parameters to which traps are sent (see [Configuring snmpd.conf on page 25](#)).

#### Sample generation of trap



SNMP functionality dictates that manager requests are sent via a different port (port 161) from that which receives the traps generated by the agent (port 162).

### SNMP and MPS Integration

#### Nortel Networks, Periphonics Enterprise MIB

The Periphonics Enterprise MIB has been created by Nortel Networks for managing the MPS network. This MIB has been registered with the Internet Assigned Numbers Authority (IANA) and has been assigned the number 1357. The Nortel Networks, Periphonics Enterprise MIB object identifier is iso.org.dod.internet.private.enterprises.periphonics (.1.3.6.1.4.1.1357).

The Nortel Networks, Periphonics Enterprise MIB allows network managers to determine the status and have control of the following components:

- VRU systems
- telephony ports
- IVR applications
- external host interfaces
- digital telephony interfaces (T1 spans)

In addition, the Nortel Networks SNMP agent sends enterprise-specific traps when one of the following events occurs:

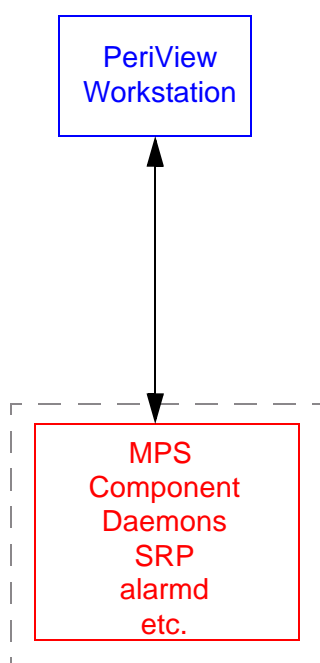
- system and component alarms
- user defined application alarms
- real time updates regarding component status.

For a detailed look at the Nortel Networks, Periphonics Enterprise MIB, see Appendix *The Nortel Networks MPS uses the Periphonics Enterprise MIB. The Periphonics Enterprise MIB is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is:* on page 76.

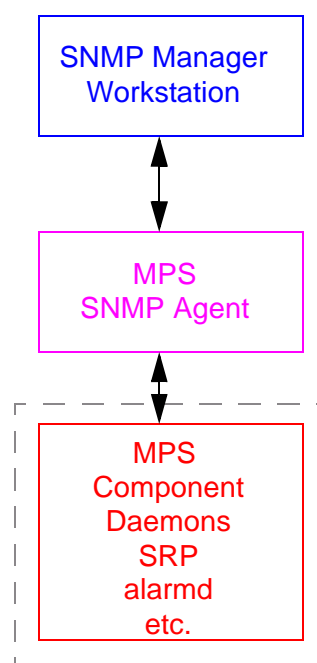
## PeriView and SNMP Integrations

In a standard MPS system, PeriView is configured on network nodes as the management application. It communicates directly with the MPS network to monitor and control through proprietary protocol. This means that workstations which are not configured to run PeriView cannot normally communicate with the MPS network and components. SNMP acts as a “translator” between management stations that are not configured for PeriView and the MPS network, both from a command line aspect as well as through third-party tools that are SNMP compliant. The Nortel Networks, Periphonics Enterprise MIB offers the basis for the items that can be monitored and controlled. For additional information, see [SNMP Utilization](#) on page 42. The illustration that follows offers a basic view of this comparison.

*Standard MPS Configuration*



*SNMP MPS Configuration*



### Understanding Components

Components are software driven aspects of the MPS network which reside on nodes and provide functionality to the network. Nodes, in turn, provide the central point of control for its resident components.

Presently, MPS components include (but may not be limited to) - MPS (enables phone lines to interact with applications to process calls), VAS (enables node to run ASE processes on behalf of an MPS component that resides on another node), and OSCAR (enables both large vocabulary and text-to-speech processing). Each component is capable of generating alarms specific to its own processes. SNMP can also monitor the status of each component through use of the MIB.





# 2

## Installation and Configuration

This chapter covers:

1. SNMP Installation
2. SNMP System Startup
3. MIB Files
4. Configuration Files
5. Environment Variables
6. Configuring snmpd.cnf
7. Error Logging
8. Executables

### SNMP Installation

Installation steps are outlined in the applicable guides for both the Solaris and Windows 2000 operating systems. The information below covers related topics that should be used in conjunction with the guides.

To verify installation on either system, use the **perirev** command. To verify that agents are responding, use the **getmany** command. Refer to [Verify Agents' Response on page 17](#).



Optionally, to free up space on disk upon completion of the package installation, the <tempdir> originally created to hold the contents of the tape should be deleted. To delete the entire contents of the temporary directory, use the following syntax:

```
rm -rf <tempdir>
```

### Package Licensing

This Nortel Networks package requires a license to use it. See both the Solaris and Windows 2000 *Install Guides* for details on installing and configuring the license file and server.

## Defining Trap Destinations Automatically

Trap destinations can be set automatically using the `trapcfg.pl` portion of the installation script (see [Defining Notification Target Entries](#) on page 34 and the script examples below). The following information presents the procedures necessary to do so in each operating system.

### In Solaris

During installation of your PERIsnmp package, you will be presented with the prompt shown below for automated processing of trap destinations. Enter **y** to proceed with this feature, **?** for more information, or **q** to quit the install. If **n** is selected, traps may still be configured at some later point either automatically by entering `trapcfg.pl` from a command line to run the script, or manually by editing the configuration file (see [Configuring snmpd.cnf](#) on page 25).

```
Configure traps during install [y,n,?,q] y
Traps will be configured during postinstall phase....

Enter IP address(s) of Management Station(s)
in a space separated list: 192.84.160.227
You have entered the following for IP addresses
192.84.160.227

Is this correct [y,n,?,q] y
```

You must, at a minimum, include the IP address of the station on which the SNMP agents are running.

The actual configuring of the traps is performed later in the installation process, given that the “y” option for trap configuration were [invoked earlier](#). The `trapcfg.pl` script configures trap destinations based on the information provided at that point.

```
Configuring snmp traps....

Stopping all SNMP processes

Backing up /opt/vps/common/etc/snmp/snmpd.cnf to
/opt/vps/common/etc/snmp/snmpd.cnf.bak
snmpNotifyEntry already exists for Console.
snmpTargetParamsEntry already exists for v1ExampleParams.
Configure trap destination for 192.84.160.227
Added snmpTargetAddrEntry for Console on 192.84.160.227
Starting all SNMP agents
```

Once the above information is processed for each of the IP addresses you had entered earlier, the stations are capable of receiving traps generated from the node on which the SNMP agents are located.

### In Windows 2000

After the installation reboot process has transpired and the logon has been completed, the `trapcfg.pl` portion of the installation procedure will appear in a DOS prompt window. Enter one or more management station IP addresses in a space separated list (these will be the stations to which traps are sent), and hit <RETURN>. The script automatically configures the required files (for additional information, see [Defining Notification Target Entries](#) on page 34).



IP addresses *must* be entered as a numerical (octet) string. Do not use management station names in lieu of this information; even though the script will appear to acknowledge configuration of the necessary files, it will not actually do so.

If the only station to receive traps is that on which PERIsnmp is running, you do not need to enter its IP address (simply hit <RETURN> at the prompt); doing this will cause duplicate traps to be issued.

```
C:\Program Files\Periphonics\bin\Perl.exe
Please enter IP addresses of all Managment Stations: 192.84.160.13

C:\>echo off
Stopping all SNMP processes.
Stopping service Periphonics_vrsnmpd
Stopping service msnsa
Stopping service SNMPPDM
Backing up C:\Program Files\Periphonics\common\etc\snmp\snmpd.cnf to C:\Program
Files\Periphonics\common\etc\snmp\snmpd.cnf.bak
1 file(s) copied.
snmpNotifyEntry already exists for Console.
snmpTargetParamsEntry already exists for v1ExampleParams.
Configure trap destination for 192.84.160.13
Added snmpTargetAddrEntry for Console on 192.84.160.13

C:\>echo off
Starting all SNMP processes.
Starting service SNMPPDM
Starting service msnsa
Starting service Periphonics_vrsnmpd
C:\>
```

After the trap configuration is completed, the window will disappear and your computer will come back up. You are now ready to use your PERIsnmp package.



Management stations can be added at a later time as trap destinations. To do this automatically, simply enter `trapcfg.pl` at a command line to run the script, then [proceed as before](#). This process can also be accomplished manually by editing the configuration file (see [Configuring snmpd.cnf](#) on page 25).

```
Command Prompt - trapcfg.pl
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>trapcfg.pl
Please enter IP addresses of all Managment Stations: 123.45.678.910_
```

## Verify Agents' Response

To verify that agents are responding, use the **getmany** command.

### getmany Command Example:

```
$ getmany is9516 periphonics
  alarmLogMax.0 = 1024
  alarmLogNumber.0 = 0
  vruIpAddress.116 = 192.84.161.52
  vruDescr.116 =
  vruLineCnt.116 = 96
  vruSpanCnt.116 = 4
  vruHostCnt.116 = 2
  vruAdminState.116 = other(1)
  vruState.116 = up(4)
  :
  :
```

## Installation Upgrades

The following conditions should be followed when upgrading from pre-5.4 SNMP releases or using Solaris 2.8 or higher.

### PERIsnmp Upgrade

Customers who are upgrading from pre-5.4 SNMP releases to this one will need to reconfigure the SNMP master agent `snmpdm`. This may include reconfiguring trap destinations previously established and reincorporating any other modifications that were made to the `snmpd.cnf` file (this file is overwritten during the installation of release 5.4). For additional information, see [Defining Notification Target Entries on page 34](#) and [Configuring snmpd.cnf on page 25](#).

Customers upgrading from PERIsnmp release 5.2.1 to 5.5.0 must manually delete the `snmpd.cnf` file, if they want the newest version of the file installed. The `snmpd.cnf` file is not overwritten when upgrading from 5.2.1 to 5.5.0, nor is it eliminated when performing a package removal using the Solaris **pkgrm** command.

### Solaris Upgrade

The PERIsnmp package conflicts with the Solaris SNMP agent found on systems running Solaris 2.8 or higher. Executing the following commands, in order, will disable the Solaris SNMP agent.

```
mv /etc/rc3.d/S76snmpdx /etc/rc3.d/old.S76snmpdx
mv /etc/rc3.d/S77dmi /etc/rc3.d/old.S77dmi
```



**You must reboot your Solaris workstation subsequent to issuing these commands for the changes to take effect.**

### SNMP System Startup



If a CPS is added to an already running system, PERIsnmp should be stopped and then restarted to recognize this new CPS.

### Solaris Systems

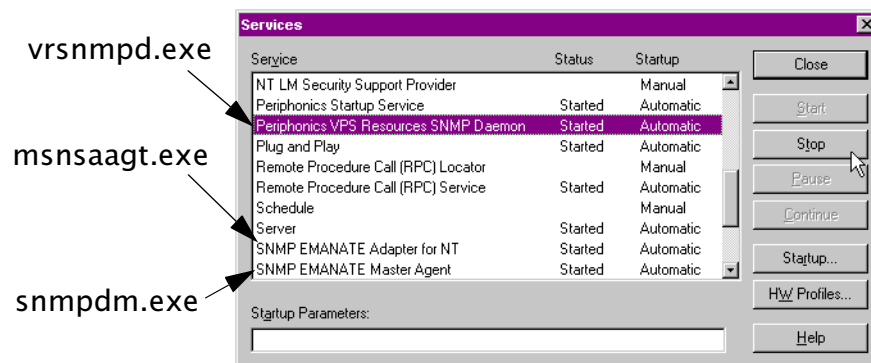
The script **s19snmp.startup** is located in the `/etc/rc3.d` subdirectory, and is responsible for starting/stopping the master agent (`snmpdm`), mib-2 subagent (`mib2agt`), and the MPS components subagent (`vrsmnpd`).

- To start the agents, use the command  
`/etc/rc3.d/s19snmp.startup start`
- To stop the agents, use the command  
`/etc/rc3.d/s19snmp.startup stop`

The `snmp.sh` file found in `$MPSHOME/common/etc/snmp` should be modified when changes to any of the command line options for `snmpdm`, `mib2agt`, or `vrsmnpd` are needed (see [Command Line Options](#) on page 23).

### Windows 2000 Systems

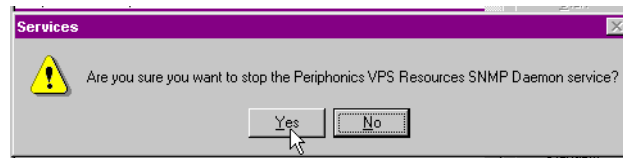
There are two methods of starting and stopping the SNMP agents on your Windows 2000 system. The Services window in the Control Panel can be used to stop and start the master agent (`snmpdm.exe`), subagent adapter (`msnsaagt.exe`), and the MPS components subagent (`vrsmnpd.exe`) individually. To do so, <LEFT> click the **Start** button on the taskbar, then follow the **Settings—Control Panel** path. Double <LEFT> click the **Services** icon in the Control Panel window to display the Services window.



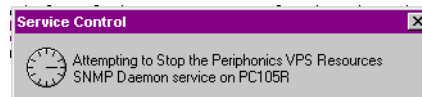
By default, the SNMP services are started automatically when you install the package. If you wish to stop the services at some point, you may do so by selecting the service in the window (as shown above), then <LEFT> clicking the **Stop** button (or pressing the keyboard letter “T”). You will be prompted on whether you wish to proceed with this action.



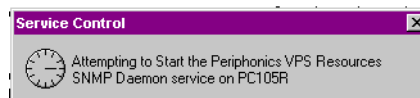
The following window prompts are used as examples. The actual windows will contain the name of the particular service you are working with.



<LEFT> click **Yes** (or press the keyboard letter “Y”) to proceed, or select **NO** to cancel the operation. If you continue, the Service Control window will advise you that an attempt is being made to stop the service. When this attempt is successful, the window will disappear and the column labeled Status in the Services window will be blank.



To start/restart a service, simply select the service as before, then <LEFT> click on the Start button (or press the keyboard letter “S”). The Service Control window will advise you that an attempt is being made to start the service. When this attempt is successful, the window will disappear and the column labeled Status in the Services window will display the word **Started**.



To stop and start all PERIsnmp processes at one time, simply use the **snmpstop** or **snmpstart** command from a DOS command prompt. When the command prompt returns, execution of the directive is successful.

```

MS-DOS Command Prompt
C:\>snmpstop
C:\>echo off
Stopping all SNMP processes.
Stopping service Periphonics_vrsnmpd
Stopping service mnsa
Stopping service SNMPDM
C:\>snmpstart
C:\>echo off
Starting all SNMP processes.
Starting service SNMPDM
Starting service mnsa
Starting service Periphonics_vrsnmpd
C:\>

```

### MIB Files

The Nortel Networks, Periphonics Enterprise MIB files are contained in `$MPSHOME/common/etc/snmp`. Currently, Nortel Networks provides its MIB in both SMIV1 and SMIV2 formats. The following information defines these files, their contents, and use:

- `Periphonicsv1.mib` and `Periphonicsv2.mib`  
Defines Nortel Networks enterprise in SMIV1 or SMIV2 format.
- `VRU_Networkv1.mib` and `VRU_Networkv2.mib`  
Defines Nortel Networks VRU Network in SMIV1 or SMIV2 format.

Prior to loading the MIBs into network management software, it is necessary to determine whether the software supports SMIV1 or SMIV2. If the software supports SMIV1, load the `Periphonicsv1.mib` file followed by the `VRU_Networkv1.mib` file. Conversely, if the software supports SMIV2, load the `Periphonicsv2.mib` file followed by the `VRU_Networkv2.mib` file.



**The order in which the MIB files are installed is mandatory. In addition, only one SMI version should be loaded and used.**



## Configuration Files

The PERIsnmp configuration files are located in `$MPSHOME/common/etc/snmp`. The following information defines these files and their contents:

- `snmpd.cnf`  
Configuration file containing information for community strings, default MIB-II values, access rights, traps, and notifications. This file is used by the master agent. Refer to [Configuring snmpd.cnf](#) on page 25.



This file must be owned by `root` and have read permission for that owner *only*. This is accomplished by setting the `root` permission mode to 400.



This file is installed as a read-only file. In order for changes to take effect, you must change its properties to allow for writing by deselecting the read-only attribute. To do this, launch the Windows 2000 Explorer tool. Locate this file, place your cursor over it, and <RIGHT> click to display the file menu. Scroll down to Properties and <LEFT> click. Go to the bottom of the `snmpd.cnf` Properties window, <LEFT> click on the Read-only box to deselect it (no check mark), then click on APPLY and/or OK.

- `snmpinfo.dat`  
Contains name to OID translation information for the MIB-II and Nortel Networks, Periphonics Enterprise MIB. This is used by all SNMP command line utilities.
- `snmp.sh`  
Contains environment variables and command line options. You must source this file after editing it or subsequent to installing the PERIsnmp package for the first time, to have the information contained therein to take effect.



### Environment Variables

#### Editing



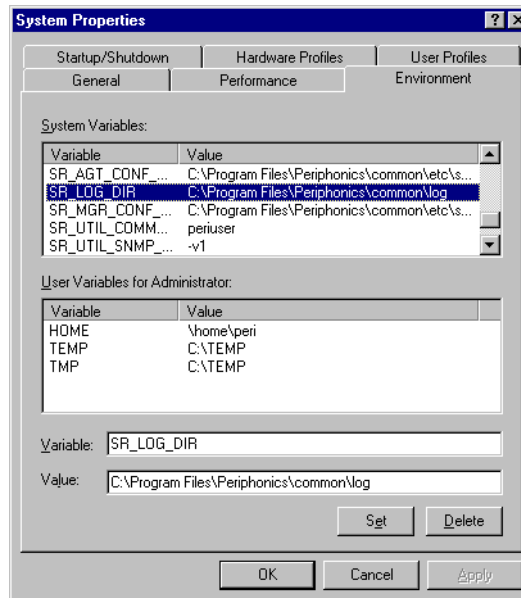
Neither `SR_MGR_CONF_DIR` nor `SR_AGT_CONF_DIR` should ever be edited by the user.

#### In Solaris

These variables can be changed by editing the file `$MPSHOME/common/etc/snmp/snmp.sh`. However, you must have write permission to do so.

#### In Windows 2000

The environment variables can be changed by editing the lines in the System Properties window (you must have administrative privileges to perform these edits). To access this window, click on the **Start** button on the task bar and follow the **Settings—Control Panel—Systems** path. Double <LEFT> click on the Systems icon to display the Systems Properties window. <LEFT> click on the Environment tab to display a listing of all system and user variables.



To change the parameters of the variable, select it from the System Variables list by <LEFT> clicking it. Now highlighted, the name of the variable and its value are displayed at the bottom of the window. Edit the Value field, then <LEFT> click the **Set** button, at which point the **Apply** button will become active. <LEFT> click it to apply the changes. When you are done using the window, <LEFT> click the **OK** button to close it.

## Master/Subagent Related

- `SR_AGT_CONF_DIR`  
Specifies the location of master agent configuration file (`snmpd.cnf`) and should be set to `$MPSHOME/common/etc/snmp`.
- `SR_LOG_DIR`  
Specifies the location of the master agent log file. This should be set to `$MPSHOME/common/log`.

## Manager Related

The first two environment variables are optional - the use of these variables eliminates the need to specify certain options when using SNMP command line utilities.

- `SR_UTIL_SNMP_VERSION`  
Specifies the default SNMP version to use for all command line utilities and should be set to `-v1` or `-v2c`.
- `SR_UTIL_COMMUNITY`  
Specifies the default SNMP community string used for all command line utilities and should be set to `perouser` or `perisAdmin`.
- `SR_MGR_CONF_DIR`  
Specifies the location of the manager configuration files (`mgr.cnf` and `snmpinfo.dat`) which are used by command line utilities and should be set to `$MPSHOME/common/etc/snmp`.

## Command Line Options

### In Solaris

The following command line options can be set by editing the file `$MPSHOME/common/etc/snmp/snmp.sh`.

- `SNMPDM_CMDLINEARGS`  
Command line arguments for `snmpdm`
- `VRSNMPD_CMDLINEARGS`  
Command line arguments for `vrtnmpd`
- `MIB2AGT_CMDLINEARGS`  
Command line arguments for `mib2agt`



Note that if environment variables are edited, the arguments need to be enclosed within double quotes (“arguments”).

### In Windows 2000

To set options for any of the SNMP agents, use the Start Up Parameters field at the bottom of the Control Panel—Services window. You must have administrative privileges and stop the service prior to entering the options as start up parameters. Restart the service to have these choices take effect. The following services affect the corresponding SNMP agents:

- SNMP EMANATE Master Agent  
Master agent snmpdm.exe
- SNMP EMANATE Adapter for Windows 2000  
Subagent adapter msnsaagt.exe
- Nortel Networks MPS Resources SNMP Daemon  
MPS components subagent vrsnmpd.exe

For further information on accessing and using the Services window, see [Windows 2000 Systems](#) on page 18.

## Configuring snmpd.cnf

Users who need to configure trap destinations will need to edit this file if the `trapcfg.pl` portion of the installation script is not used, or alternatively execute the script (see [SNMP Installation on page 14](#)). The editing tends to be the most comprehensive activity necessary, and involves both the notification and notification target fields (see [Defining Notifications on page 34](#) and [Defining Notification Target Entries on page 34](#)). This file should also be edited by those users who want to modify current security access rights, default MIB-II values, or master agent performance parameters. Otherwise, most customers can use the `snmpd.cnf` file which is shipped.



**This file must be edited when the master agent is *not* running. Stop the master agent, use any text editor to modify the fields, then restart the agent after the editing is done. For additional information on starting and stopping the agents, see [SNMP System Startup on page 18](#).**

### Default MIB-II Values

The only MIB-II object values that may need to be changed to suit individual parameters are those for `sysDescr`, `sysContact`, `sysName`, and `sysLocation`. This is done by changing the strings next to the applicable keywords, defined as follows:

- `sysDescr`  
Contains a textual description of the system.
- `sysContact`  
Contains the name of a contact person or entity and how they may be contacted.
- `sysName`  
Contains the fully qualified domain name.
- `sysLocation`  
Contains the physical location of the node that SNMP is installed on.

All strings must be enclosed in quotes. By default, Nortel Networks SNMP package is shipped with the following values:

```
sysDescr      "Nortel Networks MPS Network"
sysContact     "Nortel Networks - 1-800-645-1120"
sysName       -
sysLocation    "TBD"
```

### Security Access Rights (View Subtrees)

A view subtree defines object instances included in or excluded from a particular party of users. View subtrees may be added or deleted by adding or deleting the respective `vacmViewTreeFamilyEntry` tag. By default, Nortel Networks SNMP agents are configured with the following view subtrees:

- |                          |   |  |
|--------------------------|---|--|
| <code>Public</code>      | — | includes 0.0 and iso subtrees.   |
| <code>Network</code>     | — | includes <code>mib_2</code> , <code>snmpTrap</code> , and <code>snmpTraps</code> subtrees. |
| <code>Periphonics</code> | — | includes 0.0 and iso subtrees.   |

The view subtrees format consists of the tag `viewTreeEntry` followed by the fields:

- `vacmViewTreeFamilyViewName`  
Textual name for the family of view trees.
- `vacmViewTreeFamilySubtree`  
OID of the subtree that should be included or excluded from this view.
- `vacmViewTreeFamilyMask`  
Octet string represented by a sequence of hexadecimal numbers separated by colons. Each octet is within the range 0x00 through 0xff. Use the hyphen symbol (-) for a zero length octet string.
- `vacmViewTreeFamilyType`  
Specifies whether the `vacmViewTreeFamilySubtree` is accessible (set to included) or not accessible (set to excluded) from this family of view subtrees.
- `vacmViewTreeFamilyStorageType`  
Must be one of `nonVolatile`, `permanent`, or `readOnly`.

For example, the default views "Public" "Network" and "Periphonics" have the following entries in the `snmpd.cnf` file:

```
vacmViewTreeFamilyEntry Public iso - included  
nonVolatile
```

```
vacmViewTreeFamilyEntry Public 0.0 - included  
nonVolatile
```

```
vacmViewTreeFamilyEntry Network mib_2 - included  
nonVolatile
```

```
vacmViewTreeFamilyEntry Network snmpTrap - included  
nonVolatile
```

```
vacmViewTreeFamilyEntry Network snmpTraps - included  
nonVolatile
```

```
vacmViewTreeFamilyEntry Periphonics iso - included  
nonVolatile
```

```
vacmViewTreeFamilyEntry Periphonics 0.0 - included  
nonVolatile
```

### Groups and Access Rights

This section describes and defines groups and their associated access rights. Group and access rights may be added or deleted by adding or deleting the respective `vacmAccessEntry` tag. By default, Nortel Networks SNMP agents are configured with the following access rights:

Anyone	—	SNMPv1/SNMPv2c read-only access to Public view.
Periuser	—	SNMPv1/SNMPv2c read-only access to Periphonics view.
Periadmin	—	SNMPv1/SNMPv2c read-write access to Periphonics view.

The access rights format consists of the tag `vacmAccessEntry` followed by the fields:

- `vacmGroupName`  
Name of the group associated with these access rights.
- `vacmAccessContextPrefix`  
Entire or partial context identifier. Contains the value `-` (hyphen) if no context identifier is defined.
- `vacmAccessSecurityModel`  
Must have the value `snmpv1` or `snmpv2c`.
- `vacmAccessSecurityLevel`  
Must be set to `noAuthNoPriv` for no authentication and no privacy.
- `vacmAccessContextMatch`  
Specifies the way in which the context of a request must match `vacmAccessContextPrefix`. Must have the value `exact` or `prefix`.
- `vacAccessReadViewName`  
The `vacmViewTreeFamilyViewName` which identifies the view subtrees accessible for `get`, `get-next`, and `get-bulk` requests.
- `vacmAccessWriteViewName`  
The `vacmViewTreeFamilyViewName` which identifies the view subtrees accessible for `set` requests. Contains the value `-` (hyphen) if no view subtree is defined.
- `vacmAccessNotifyViewName`  
The `vacmViewTreeFamilyViewName` which identifies the view subtrees accessible for trap messages and `inform` requests.
- `vacmAccessStorageType`  
Must be one of `nonVolatile`, `permanent`, or `readOnly`.



For example, the default access rights groups "Anyone" "Periuser" and "Periadmin" have the following entries in the `snmpd.cnf` file:

```
vacmAccessEntry Anyone - snmpv1 noAuthNoPriv exact
Public - Public nonVolatile

vacmAccessEntry Anyone - snmpv2c noAuthNoPriv exact
Public - Public nonVolatile

vacmAccessEntry Periuser - snmpv1 noAuthNoPriv exact
Periphonics - Periphonics nonVolatile

vacmAccessEntry Periuser - snmpv2c noAuthNoPriv
exact Periphonics - Periphonics nonVolatile

vacmAccessEntry Periadmin - snmpv1 noAuthNoPriv
exact Periphonics Periphonics Periphonics
nonVolatile

vacmAccessEntry Periadmin - snmpv2c noAuthNoPriv
exact Periphonics Periphonics Periphonics
nonVolatile
```

### Community Strings

This section specifies the configuration of community strings which can be used in SNMP requests to the Nortel Networks SNMP agent. Community strings can be added or deleted by adding or deleting the respective `srCommunityEntry` tag. By default, the Nortel Networks SNMP agents are configured with the following community strings:

- `public` — Access control of the community group named Anyone.
- `periadmin` — Access control of the community group named Periadmin.
- `periuser` — Access control of the community group named Periuser.

The community string format consists of the tag `communityEntry` followed by the fields:

- `srCommunityAuthSnmpID`  
Must have the value `localSnmpID`.
- `srCommunityName`  
Name associated with this community string.
- `srCommunityGroupName`  
Name of this community's group. This name must be defined by at least one `vacmAccessEntry` and appear with the community string name in a `vacmSecurityToGroup` entry. (For related information, see [Groups and Access Rights](#) on page 28 and [Assigning Principles to Groups](#) on page 32, respectively.)
- `srCommunityContextSnmpID`  
Must have the value `localSnmpID`.
- `srCommunityContextName`  
Must have the value `default`.
- `srCommunityTransportLabel`  
Used to select a set of entries in the `snmpTargetAddrTable` for source address checking (see [Defining Notification Target Entries](#) on page 34). Use the value `-` (hyphen) if source address checking is not desired.
- `srCommunityMemoryType`  
Must be one of `nonVolatile`, `permanent`, or `readOnly`.

For example, the default community strings "periadmin" "periuser" and "public" have the following entries in the `snmpd.cnf` file:

```
communityEntry localSnmpID periadmin Periadmin  
localSnmpID default - nonVolatile
```

```
communityEntry localSnmpID periuser Periuser  
localSnmpID default - nonVolatile
```

```
communityEntry localSnmpID public Anyone localSnmpID  
default - nonVolatile
```

### Assigning Principles to Groups

This section describes how a principle (i.e. [community string](#)) is assigned to a group. Principle assignments may be added or deleted by adding or deleting their respective `vacmSecurityToGroupEntry` tag. By default, Nortel Networks SNMP agents are configured with the following principle/group assignments:

- |                        |   |
|------------------------|---|
| <code>public</code>    | — Assigned to Anyone group for both SNMPv1 and SNMPv2c security model.    |
| <code>periadmin</code> | — Assigned to Periadmin group for both SNMPv1 and SNMPv2c security model. |
| <code>periuser</code>  | — Assigned to Periuser group for both SNMPv1 and SNMPv2c security model.  |

The principle/group format consists of the tag `vacmSecurityToGroupEntry` followed by the fields:

- `vacmSecurityModel`  
Specifies whether the SNMPv1 (value of `snmpv1`) or SNMPv2c (value of `snmpv2c`) security model should be used.
- `vacmSecurityName`  
Name of the principle assigned to the group.
- `vacmGroupName`  
Name of the group principle is assigned to.
- `vacmSecurityToGroupStorageType`  
Must be one of `nonVolatile`, `permanent`, or `readOnly`.

For example, the default principles "public" "periadmin" and "periuser" have the following entries in the `snmpd.cnf` file:

```
vacmSecurityToGroupEntry snmpv1 public Anyone
nonVolatile
vacmSecurityToGroupEntry snmpv1 periadmin Periadmin
nonVolatile

vacmSecurityToGroupEntry snmpv1 periuser Periuser
nonVolatile

vacmSecurityToGroupEntry snmpv2c public Anyone
nonVolatile

vacmSecurityToGroupEntry snmpv2c periuser Periuser
nonVolatile

vacmSecurityToGroupEntry snmpv2c periadmin Periadmin
nonVolatile
```

For related information regarding groups, see [Groups and Access Rights](#) on page 28.

### Defining Notifications

This section defines the format and configuration of a notification. Notifications can be of the trap or inform-request variety, and can be added or deleted by adding or deleting the respective `snmpNotifyEntry` tag. By default, Nortel Networks SNMP agents are configured to send notifications to the Console, which is the workstation the SNMP agent is running on.

The notification format consists of the tag `snmpNotifyEntry` followed by the fields:

- `snmpNotifyName`  
“Name” of this notification.
- `snmpNotifyTag`  
Name used to select a set of entries in the `snmpTargetAddrTable` (see [Defining Notification Target Entries on page 34](#)) and that which corresponds to the value of `snmpTargetAddrTagList`.
- `snmpNotifyType`  
Stipulates whether this notification is a trap (value of `trap`) or inform-request (value of `inform`).
- `snmpNotifyStorageType`  
Must be one of `nonVolatile`, `permanent`, or `readOnly`.

For example, the default notification entry in the `snmpd.cnf` file is as follows:

```
snmpNotifyEntry 31 Console trap nonVolatile
```

### Defining Notification Target Entries

Specifies the endpoints (targets) that may receive notifications. These endpoints may be added or deleted by adding or deleting a `snmpTargetAddrEntry` tag. The default endpoint is the master agent workstation, defined with the name `Console`. An `snmpTargetAddrEntry` will have to be added for each management station other than the default value to which notifications are to be sent. This can be done manually or, for traps in particular, handled automatically by running the `trapcfg.pl` portion of the installation script (for additional information, see [Defining Trap Destinations Automatically on page 15](#)). The `trapcfg.pl` script must be run on the same node that the SNMP agents are running on. By specifying the IP address(es) when prompted, the script automatically stops all SNMP agents, backs up the `snmpd.cnf` file (to `snmpd.cnf.bak`), adds the `snmpTargetAddrEntry` for the specified IP address, then restarts the SNMP agents. If more than one management station is designated to receive traps, the IP addresses can be entered separated by a single space, at the appropriate prompt.

In addition, the `snmpTargetAddrTagList` entry should match that of `snmpNotifyTag` (see [Defining Notifications on page 34](#)).

The target entry format consists of the tag `snmpTargetAddrEntry` followed by the fields:

- `snmpTargetAddrName`  
Name of the notification target.
- `snmpTargetAddrTDomain`  
An OID which indicates the network type. This should be set to `snmpUDPDomain`.
- `snmpTargetAddrTAddress`  
A valid IP address in the transport domain specified by `snmpTargetAddrTDomain`.
- `snmpTargetAddrTimeout`  
Expected round trip time, expressed in hundredths of a second, for communicating with the `snmpTargetAddrTAddress`.
- `snmpTargetAddrRetryCount`  
The number of times the SNMP entity will attempt to retransmit an inform-request when a response is not received.
- `snmpTargetAddrTagList`  
A quoted string containing one or more space separated tags, which correspond to the value of the `snmpNotifyTag`.
- `snmpTargetAddrParams`  
Used to select a set of entries in the `snmpTargetParamsTable` (see [Defining Target Parameters on page 36](#)).
- `snmpTargetAddrStorageType`  
Must be one of `nonVolatile`, `permanent`, or `readOnly`.
- `tgtAddressMask`  
Should always be set to `255.255.255.255:0`.

For example, the default notification target entries appear as follows in the `snmpd.cnf` file:

```
snmpTargetAddrEntry 31 snmpUDPDomain 127.0.0.1:0 3
Console v1ExampleParams nonVolatile
255.255.255.255:0

snmpTargetAddrEntry 32 snmpUDPDomain 127.0.0.1:0 3
Console v2cExampleParams nonVolatile
255.255.255.255:0
```

where `127.0.0.1:0` is the IP address of the endpoint that receives the notification.

### Defining Target Parameters

This section describes the parameters to be used when sending notifications. Parameters may be added or deleted by adding or deleting the respective `snmpTargetParamsEntry` tag. By default, Nortel Networks SNMP agents are configured with the following parameters:

- |                               |   |   |
|-------------------------------|---|---|
| <code>v1ExampleParams</code>  | — | SNMPv1 trap with community string set to public.  |
| <code>v2cExampleParams</code> | — | SNMPv2c trap with community string set to public. |

The target parameter format consists of the tag `snmpTargetParamsEntry` followed by the fields:

- `snmpTargetParamsName`  
Name given to this parameter.
- `snmpTargetParamsMPPModel`  
Assign a value of 0 for SNMPv1 or 1 for SNMPv2c.
- `snmpTargetParamsSecurityModel`  
Must have a value of `snmpv1` or `snmpv2c`, in conjunction with `snmpTargetParamsMPPModel` above.
- `snmpTargetParamsSecurityName`  
Specifies the principle to use in the notification.
- `snmpTargetParamsSecurityLevel`  
Must always be set to `noAuthNoPriv`.
- `snmpTargetParamsStorageType`  
Must be one of `nonVolatile`, `permanent`, or `readOnly`.

For example, the default target parameter entries appear as follows in the `snmpd.cnf` file:

```
snmpTargetParamsEntry v1ExampleParams 0 snmpv1
public noAuthNoPriv nonVolatile

snmpTargetParamsEntry v2cExampleParams 1 snmpv2c
public noAuthNoPriv nonVolatile
```

For related information, see [Community Strings on page 30](#) and [Assigning Principles to Groups on page 32](#).



## Error Logging

Error logging is now available with this version of PERIsnmp. The error log files are located in `$MPSHOME/common/log`, and can be read with any text-based application or command. These log files are identified as follows:

File Name....	...Logs Errors for the....	...Whose Executable is...	
		On Windows 2000	On Solaris
<code>vrsnmpd.log</code>	MPS components subagent	<code>vrsnmpd.exe</code>	<code>vrsnmpd</code>
<code>snmpd.log</code>	Master agent	<code>snmpd.exe</code>	<code>snmpd</code>
<code>msnsa.log</code>	Subagent adapter	<code>msnsaagt.exe</code>	(not implemented)



You should also check `/var/admin` for logged error messages (specifically concerned with licensing problems).

### Executables

The following executables are integral components of your PERIsnmp package and responsible for its functionality. On Solaris systems, these files are located in `$MPSHOME/PERIsnmp/bin`; on Windows 2000 systems, they are located in `%MPSHOME%\bin`.

### Agents



In the following listing, Windows 2000 agents are identified as those appended with the `.exe` extension.

- `snmpdm/snmpdm.exe`  
SNMP master agent. Responsible for authenticating formatting, security, access, and ownership parameters of requests, replies, and notifications. Acts as an intermediary between the management station and the subagents.
- `mib2agt`  
SNMP subagent for MIB-II subtree. Responsible for providing information concerning MIB-II items to the master agent.
- `msnsaagt.exe`  
SNMP subagent adapter for MIB-II subtree. Responsible for providing information concerning MIB-II items to the master agent.
- `vrtnmpd/vrtnmpd.exe`  
MPS components subagent for Nortel Networks `vrtnetwork` subtree. Acts as a proxy for the MPS network, using data in the Nortel Networks, Periphonics Enterprise MIB to construct replies to queries. Also sends requests to and receives replies from resource daemons outside the SNMP agents when requested data is not available internally.



## SNMP Command Line Utilities

For examples of command line utilities in use, see [SNMP Applied](#) on page 48.

- **getbulk**  
Gets as many MIB variable(s) as possible in one request.
- **getmany**  
Gets the MIB variables in the specified MIB subtree.
- **getmet**  
Gets MIB-II variables containing metric information.
- **getnext**  
Gets the MIB variable(s) which is (are) lexicographically next after the specified MIB variable(s).
- **getone**  
Gets the specified MIB variable(s).
- **getroute**  
Gets MIB-II variables containing routing information.
- **getsub**  
Gets MIB-II variables containing routing information for specified subnet.
- **gettab**  
Gets MIB variable(s) from specified table and display in tabular form.
- **setany**  
Sets one or more MIB variable(s) to specified values.
- **traprcv**  
Displays contents of received trap.

**This page has been intentionally left blank.**

# **MPS Functions and Administration**

**This chapter covers:**

- 1. SNMP Utilization**
- 2. MIB Variables**
- 3. Instance Identification**
- 4. SNMP Applied**
- 5. The MIB Applied**

This chapter discusses, in generic terms, the functions that can be controlled and monitored using SNMP in the MPS environment. These functions generally fall into the administrative and maintenance categories. The user is directed to the Nortel Networks, Periphonics Enterprise MIB (hereafter simply called MIB) variables that are monitored only, those that can be both monitored and controlled, and those that act as indexes into tables within the MIB. In addition, notifications generated by the agent and sent to the manager for disposition are also discussed. For a detailed description of each variable and its associated data, see Appendix *[The Nortel Networks MPS uses the Periphonics Enterprise MIB. The Periphonics Enterprise MIB is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is: on page 76.](#)*

## SNMP Utilization

### Communication

At a basic level, SNMP uses five types of messages for processing information. These messages are exchanged between the manager and the agent. Three messages are sent directly from the manager to the agent - get-request, get-next-request, and set-request. In addition, SNMPv2c includes a message termed the get-bulk request which is also sent from the manager to the agent. The remaining two - get-response and the trap - are sent from the agent to the manager. The “get-” and “set-” messages are simply request-reply protocols; the trap is initiated directly by the agent and sent to the manager’s station. For a further discussion of these messages, see *[Protocol on page 6.](#)* For details on changing destinations for traps as they apply to the Nortel Networks product in particular, see *[Configuring snmpd.cnf on page 25.](#)*

The Nortel Networks SNMP product also contains a group of command line utilities that bundle many functions into small, self-contained programs. This provides a quick and simple means of manipulation and retrieval of data by the user. The use of these utilities results in the generation of a Protocol Data Unit (PDU). These utilities can be broken down into three main areas - the “get-”, “set-” and trap designations. It is these utilities that the SNMP manager uses when querying or posting variables within the MIB. For a further definition of each command line utility, see *[SNMP Command Line Utilities on page 39.](#)*

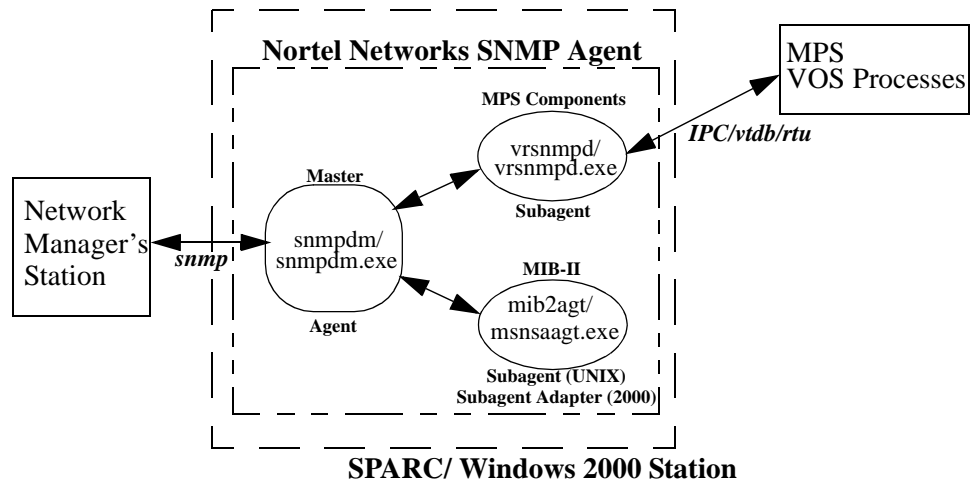
## System Architecture



In the following discussion, Windows 2000 agents are identified as those appended with the .exe extension.

When the network management station issues an SNMP request, it is sent to the master agent (`snmpdm/snmpdm.exe`) which confirms that the request is formatted properly and contains the proper security and community strings. If the data is not accurate, the agent returns the request to the manager with an appropriate message indicating the problem; if the request is properly configured, it forwards the request to the applicable subagents.

For MIB-II items, the master agent forwards the request to the subagent/adaptor `mib2agt/msnsaagt.exe`. For information concerning the MPS system, the master agent forwards the request on to the MPS components subagent `vrtnmpd/vrtnmpd.exe`, which acts as a proxy for the MPS network. It uses the data in the Nortel Networks, Periphonics Enterprise MIB to formulate a reply. If the data is not presently available, the subagent sends the request to the appropriate resource daemons outside the SNMP agent complex. The information from these daemons travels the opposite route through the subagent, on to the master agent, and ultimately back to the management station which originally issued the request. Traps travel in much the same manner along this return path - changes in status in the MPS are monitored by the subagent `vrtnmpd/vrtnmpd.exe`, which formulates the appropriate data and issues it to the master agent, which in turn builds the trap and then passes it on to the designated management station. The graphic below illustrates the context of the MPS management architecture.



### MIB Variables

#### Access

Many variables in the Nortel Networks, Periphonics Enterprise MIB database contain access information which stipulates the characteristics of querying and control. These levels of access are defined as follows:

- **read-write** - indicates variable may be both read and written to by the manager
- **read-only** - indicates variable may only be read by the manager
- **accessible-for-notify** - indicates variable may only appear in notification
- **not-accessible** - indicates variable may not be directly read or written to.

If a request is made on a variable for which no access is allowed, an error message will be generated and returned to the manager, making the request indicate this fact. If the manager attempts to set a value on a variable which has read-only access, an error message will also be returned. Several variables with not-accessible status are used as indexes into tables in this MIB.



## Status

The status of objects in the MIB is generally that of current, meaning that the object is presently supported by the SNMP product. However, several objects have been changed to a status of deprecated, meaning that although they still exist in the MIB, they will no longer be supported (essentially acting as placeholders in the MIB hierarchy). The following table represents a historical view of these objects as well as the objects that supercede them. For a full listing of all MIB objects and their states, see Appendix *The Nortel Networks MPS uses the Periphonics Enterprise MIB. The Periphonics Enterprise MIB is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is:* on page 76.

Deprecated Object and OID	Replaced by this Object and OID
vruNetworkMIBCompliance .1.3.6.1.4.1.1357.1.1.1.	vruNetworkMIBComplianceBy Component .1.3.6.1.4.1.1357.1.1.1.2
vruNetworkMIBComplianceBy Component .1.3.6.1.4.1.1357.1.1.1.2	vruNetworkMIBComplianceBy Component2 .1.3.6.1.4.1.1357.1.1.1.3
vruNotificationsGroup .1.3.6.1.4.1.1357.1.1.2.1	componentNotificationsGroup .1.3.6.1.4.1.1357.1.1.2.8
componentNotificationsGroup .1.3.6.1.4.1.1357.1.1.2.8	componentNotificationsGroup2 .1.3.6.1.4.1.1357.1.1.2.10
vruEventsGroup .1.3.6.1.4.1.1357.1.1.2.2	componentEventsGroup .1.3.6.1.4.1.1357.1.1.2.9
componentEventsGroup .1.3.6.1.4.1.1357.1.1.2.9	componentEventsGroup2 .1.3.6.1.4.1.1357.1.1.2.11
notifyAlarm .1.3.6.1.4.1.1357.1.2.2.1	notifyAlarmByComponent .1.3.6.1.4.1.1357.1.2.2.9
notifyAlarmByComponent .1.3.6.1.4.1.1357.1.2.2.9	notifyAlarmByComponentEx .1.3.6.1.4.1.1357.1.2.2.10
notifyVruStateChg .1.3.6.1.4.1.1357.1.2.2.2	notifyComponentStateChg .1.3.6.1.4.1.1357.1.2.2.8
alarmLogVruId .1.3.6.1.4.1.1357.1.2.3.6.1.4	alarmLogComponentId .1.3.6.1.4.1.1357.1.2.3.6.1.11

### Types

There are two entries in the MIB which the SNMP manager is primarily concerned with. The notification-type generates traps in response to an event occurrence (and these traps are forwarded to the manager). While this entry cannot be directly controlled by the manager, traps can alert the user to an instance that needs further attention through other means. The object-type is accessible to the manager within the constraints of their assigned values as discussed in [Access on page 44](#). It is these variables that the manager can use to control or monitor the status of the MPS systems.

### Instance Identification

#### Direct Referencing

There are variables in the Nortel Networks, Periphonics Enterprise MIB which can be queried directly (that is, an instance does not have to be specified). A command can be issued on the variable and its value will be returned by simply appending a .0 to the object identifier (OID). These are termed scalar requests because no further instance identification is required. See [Getting a Value on page 48](#) for an example of this type of transaction.

#### Table Referencing

Object values are sometimes maintained in tabular form in the enterprise MIB. This most often occurs when specific locations or object identifications need to be individually delineated. In cases such as this, tables can be built in the MIB through sequencing of variables which act as indexes into the rows and columns of data (see Appendix [The Nortel Networks MPS uses the Periphonics Enterprise MIB. The Periphonics Enterprise MIB is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is: on page 76](#)). The variables themselves that specify these indexes are not directly accessible. Instead, they act as a means of indexing into the table to access the desired information for the specified location. The combination of the variable and its index forms the object identifier (OID) for that line.

The tables themselves are “built” in much the same fashion as a spreadsheet. When the Nortel Networks MPS components subagent starts up, it determines, through the use of proprietary APIs, the components that are present on the system and the makeup of these components. It uses this topological configuration to assign the data to and structure the tables. As stated previously, they may contain one or more variables which act as indexes into the table. Values specific to this index will be maintained in that row of the table. Since tables are built lexicographically (entries ordered by their object identifiers), the **getnext** command comes in handy by allowing the manager to search through tables to find the desired data. Simply start with the object name that defines the table (no instance is applicable in these cases) and use the **getnext** command: by repeating this command and substituting the previous reply’s object and instance, you can iterate through the table until the desired

information is obtained. When the object name changes, you will know that you have reached the end of the table you had been accessing.

The **getmany** command extracts all the data in the specified subtree at one time. The **gettab** command displays blocks of data extracted from tables. In SNMPv2c, the **getbulk** command can obtain large amounts of information by replicating several **getmany** requests at one time. If more than one variable is used to index the table, these instances will be appended one after the other, separated by periods.



If a table is “empty” (contains no data), the OID returned on a request will be that of the next lexicographical table variable that *does* contain data (even though that returned value may not seem to apply to your original request).

For example, the `vrTable` is built from the following sequence of variables: `vrId`, `vrIpAddress`, `vrDescr`, `vrLineCnt`, `vrSpanCnt`, `vrHostCnt`, `vrAdminState`, `vrState`, and `vrStateLastChange`. The first variable is a unique value that is used as an index into the table, and the remaining variables are values associated with this index. Note that the `vrDescr` variable has not been implemented at the present time. Conceptually then, this table might look like this:

**vrTable**

vrId	vrIp Address	vr Desc	vr Line Cnt	vr Span Cnt	vr Host Cnt	vr Admin State	vr State	vr State Last Change
1	192.84.160.227	VRU1	48	2	1	other(1)	up(4)	07 cd 05 08 0c 14 07 00
2	192.85.160.228	VRU2	48	4	1	other(1)	up(4)	07 cd 05 08 0c 14 08 00

A **getnext** request on this table would return an OID including the instance in the first column of the table. Depending on the variable queried, its value for that line would also be returned. For example then, if **getnext nodename vrTable** were sent by the manager, the agent would respond with `vrIpAddress.1 = 192.84.160.227`, where `.1` is the instance which identifies the VRU the information is related to; the manager could then send **getnext nodename vrIpAddress.1** and would receive `vrIpAddress.2 = 192.85.160.228`; issue **getnext nodename vrIpAddress.2** and receive a reply of `vrDescr.1 = VRU1`, and so on. The syntax shown is the default, where **nodename** would be the actual name of your management station, and the `vrDescr` variable is presently not used, but this example serves to show how indexing and tables work within SNMP. The manager could continue polling through the table until the desired result(s) were obtained. If the exact index into the table were known, the manager could use it to directly query the variable in question (i.e., **getone nodename vrLineCnt.2** ==> `vrLineCnt.2 = 48`). For a further applied example of indexing, see [Setting a Value on page 49](#).

### SNMP Applied

#### Getting a Value

With these criteria in mind, the user can now manage the MPS using SNMP. For instance, the overall basic structure of a command line request, in general, might be as follows. To specify that a single MIB variable be queried, we would enter the command **getone**. Next, we would stipulate the version of SNMP that we are using (**-v1** or **-v2c**); the entry **hostname** would in actuality be the exact name of the host on which the agent is located; the community is set to **periuser** (the manager would set this to **periadmin** to allow for reading/writing permissions); the object or objects being queried are listed last. Performing these actions results in the **getone** command generating a get-request PDU containing all the OIDs stipulated for that line. The get-request PDU is sent to the agent on the **hostname** entry. The management station waits for a response from the agent. If no response is received within a specified time, the command times out. At this point, the request would have to be reissued if an answer were still desired. Alternatively, the agent sends a get-response PDU containing the values of the variable queried, along with the OID data originally issued on the command line. This result is then output at the command line on the manager's station. To illustrate, let's say you want to know the maximum number of alarms that may be logged into the alarm log table at one time. Using the command line utilities, you would enter at the prompt on your management station the following, which generates the get-request PDU:

```
getone -v1 hostname periuser alrmLogMax.0
```

and the manager would receive the following command line output as a result of the agent generating a get-response PDU:

```
alrmLogMax.0 = 1024
```

The numerical value shown above is the default value. Also, note that the instance **.0** must be included with all scalar requests: if this request were being made on a table, and the index into the table were known, you would then enter that data in place of the zero.

By default, the PERIsnmp package has been set up so that your command line requests will consist simply of the command, the hostname, and the variable:

```
getone hostname alrmLogMax.0
```

Remember that other arguments can be added to or deleted from the environment variables, which will then affect that syntax of your command line input. Unless these variables have been edited by the user, the above example should be sufficient. For additional, related information, see [Environment Variables on page 22](#).

## Setting a Value

Setting a value in a variable is similarly executed. For instance, if you wished to start an application on a particular line, you would enter the following command line at the prompt:

```
setany -v1 hostname periadmin lineAppAdminState.2.1  
-i startapp
```

and the manager would receive the following result:

```
lineAppAdminState.2.1 = startapp(2)
```

As in the previous example, we have specified that a single MIB variable be addressed, using SNMPv1 (this could instead be -v2c); the entry **hostname** would in actuality be the exact name of the host on which the agent is located; the community is set to **periadmin** to allow for reading/writing permissions; and the object to be set is listed next. In this case, the numerical representation appended to the variable (**.2.1**) dictates that line number one on MPS number two be addressed (since the variables `lineVruId` and `lineId` act as indexes into the object `lineTable`). The entry **-i** indicates that the item that follows has an integer value, though for ease of use to the manager it is represented in a textual format: it shows that the manager wishes to start an application on the indicated line and MPS. The agent sends a get-response PDU indicating the location and value of the variable that was set. This tells the manager that the execution to start an application on MPS 2, line 1 was successful.

By default, the PERIsnmp package has been set up so that your command line requests will consist simply of the command, the hostname, the variable, the variable type, and the value:

```
setany hostname lineAppAdminState.2.1 -i startapp
```

Remember that other arguments can be added to or deleted from the environment variables, which will then affect that syntax of your command line input. Unless these variables have been edited by the user, the above example should be sufficient. For additional, related information, see [Environment Variables on page 22](#).

## Viewing Traps

As stated previously (see [trap on page 9](#)), traps are unsolicited messages sent from the agent to the manager in the event of unexpected internal conditions within the MPS system. These traps can consist of SNMPv1-defined events as well as enterprise-specific events (see [Protocol on page 6](#)). These enterprise-specific traps are predefined, and include notifications of alarms generated by the MPS system (for definitions of these traps, see [Events Group Notifications Subtree on page 51](#)).

In order for the `notifyAlarmByComponentEx` trap to be generated and forwarded to the management station, the **alarmd** process must be running on the MPS node (each node contains one **alarmd** daemon). To verify that the node is connected to the **alarmd** daemon, enter **alarm** at any command line. You should receive a response similar to the one that follows to indicate you are connected:

```
<<< alarm: Connected to [alarmd#common.0,gen/  
pc105r@pc105r:1034] >>>
```

To view traps, enter the **traprcv** command at the command line of any management station designated to receive traps (for related information, see [Configuring snmpd.cnf on page 25](#)). If more than one station has been so configured, the traps generated at all nodes which included a particular management station during the configuration will be viewable at that station.



No more than one **traprcv** process should be started per management station.

For example, if node A, node B, and node C are all configured to receive traps, but node A also has the IP addresses for nodes B and C in its `snmpd.cnf` file, then those two nodes will also receive traps generated by node A. Initially, the **traprcv** window will display the prompt `Waiting for traps.` until traps are generated.

To verify that both commands are working properly:

- Leave both the **alarm** and **traprcv** command windows open.
- Stop and start SRP on the node that you wish to check.



Issue the commands `/etc/rc3.d/S20vps.startup stop` and `/etc/rc3.d/S20vps.startup start` in order.



Stop and start the Nortel Networks Startup Service in the Control Panel—Services window (for generalized information on stopping and starting services, see [Windows 2000 Systems on page 18](#)).

- You should see alarms generated in the **alarm** window, followed immediately thereafter by traps being generated in the **traprcv** window. You can then close, minimize, or leave the windows open. Alarms and traps continue to be generated no matter what the condition of the respective window.



Starting and stopping SRP may have other consequences regarding your system. Be aware that running this test may affect overall system performance.

All other predefined enterprise-specific traps will be generated as they occur. For additional information regarding starting and stopping SRP and using system Services, see the *System Operator's Guide* for the appropriate operating system.

## The MIB Applied

### Subtrees

There are several subtrees in the MIB that can be used to monitor and/or control the MPS network, and each contains variables particular to certain aspects of the network. In this section each subtree is defined generically, and any variables that have been implemented and which fall into the monitor/control disposition are discussed further. In addition, those variables used as indexes into tables are also identified. The OID for each variable is provided for further consideration. For a complete definition of the MIB, see Appendix *The Nortel Networks MPS uses the Periphonics Enterprise MIB. The Periphonics Enterprise MIB is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is: on page 76.*

### Events Group Notifications Subtree

Defines traps that are sent to network management stations. These traps are generated by the MPS agent directly and sent to the manager. As a result the manager could, for example, write a script that runs when a trap is received (i.e.

notifyAlarmByComponent causes script to run that pages the network administrator).

Variable	OID	Condition
notifyLineAppStateChg	.1.3.6.1.4.1.1357.1.2.2.3	This trap is forwarded when an application associated with a VRU line changes state.
notifySpanStateChg	.1.3.6.1.4.1.1357.1.2.2.4	This trap is forwarded when a VRU span changes state.
notifyHostStateChg	.1.3.6.1.4.1.1357.1.2.2.5	This trap is forwarded when a VRU host changes state.
notifyLuStateChg	.1.3.6.1.4.1.1357.1.2.2.6	This trap is forwarded when a VRU host LU changes state.
notifyTopologyChg	.1.3.6.1.4.1.1357.1.2.2.7	This trap is forwarded when a VRU network object topology status changes.
notifyComponentStateChg	.1.3.6.1.4.1.1357.1.2.2.8	This trap is forwarded when a network component changes state.
notifyAlarmByComponentEx	.1.3.6.1.4.1.1357.1.2.2.10	This trap forwards alarms generated by an exceptional condition occurring in a component in the MPS network, and includes the IP address of the component generating the alarm.



Although the objects `notifyAlarm`, `notifyAlarmByComponent`, and `notifyVruStateChg` are still present in the MIB, each has a status of deprecated and is no longer supported.

### Component Events Alarms Subtree

Defines the component alarms that are recorded (through the `notifyAlarmByComponent` trap) and the number of alarms that are kept. The variables described (other than indexing variables) are all of monitor (read-only) status. Requests on the table in this subtree must include the instance identification provided by `almLogIdx`.



**If the Nortel Networks MPS components subagent is stopped and restarted for any reason, the contents of the `almLogTable` is cleared.**



The connection, or lack thereof, between the MPS components subagent and **alarmed** will affect the alarms actually recorded in the `almLogTable`.

Variable	OID	Condition
<code>almLogMax</code>	<code>.1.3.6.1.4.1.1357.1.2.3.4</code>	Specifies the maximum number of alarms that may be logged into the alarm log table at one time. When this number is reached, the initial alarms will be deleted as new ones are added.
<code>almLogNumber</code>	<code>.1.3.6.1.4.1.1357.1.2.3.5</code>	The current number of log entries.
<code>almLogIdx</code>	<code>.1.3.6.1.4.1.1357.1.2.3.6.1.1</code>	Serves as an index into <code>almLogTable</code> , and is a number between 1 and <code>almLogNumber</code> (inclusive).
<code>almLogSeverity</code>	<code>.1.3.6.1.4.1.1357.1.2.3.6.1.2</code>	Integer describing how severe the alarm is. The higher the value, the higher the severity; a value of 0 indicates that no severity has been assigned.
<code>almLogCode</code>	<code>.1.3.6.1.4.1.1357.1.2.3.6.1.3</code>	Code assigned to identify the alarm.
<code>almLogLineId</code>	<code>.1.3.6.1.4.1.1357.1.2.3.6.1.5</code>	The phone line associated with the alarm; a value of 0 indicates no phone line.
<code>almLogHostId</code>	<code>.1.3.6.1.4.1.1357.1.2.3.6.1.6</code>	Identifies the host associated with the alarm; a value of 0 indicates no host association.
<code>almLogProcName</code>	<code>.1.3.6.1.4.1.1357.1.2.3.6.1.7</code>	The name of the process that generated the alarm.
<code>almLogMessage</code>	<code>.1.3.6.1.4.1.1357.1.2.3.6.1.8</code>	A textual message describing the alarm that occurred.



Variable	OID	Condition
alarmLogTime	.1.3.6.1.4.1.1357.1.2.3.6.1.9	The time that the alarm was generated.
alarmLogComponent Type	.1.3.6.1.4.1.1357.1.2.3.6.1.10	Component type that generated the alarm.
alarmLogComponent Id	.1.3.6.1.4.1.1357.1.2.3.6.1.11	Component identifier of the component that generated the alarm.
alarmLogComponent IpAddress	.1.3.6.1.4.1.1357.1.2.3.6.1.12	IP address of the component which generated the alarm.



Although the object `alarmLogVruId` is still present in the MIB, it has a status of deprecated and is no longer supported.



The component type name and its corresponding component type ID can both be determined by accessing the objects in the [Component Type Table](#) (see page 60).

### VRU Subtree

Defines the parameters of the VRU system. The variables can both monitor and control the VRU. Requests on the table in this subtree must include the instance identification provided by `vruid`.

	Variable	OID	Condition
Index	<code>vruid</code>	.1.3.6.1.4.1.1357.1.3.1.1	The VRU number assigned to a particular VRU. This number is unique within the VRU network, and is used as an index into <code>vrutable</code> .
Monitor	<code>vruiPAddress</code>	.1.3.6.1.4.1.1357.1.3.1.2	VRU network IP address.
	<code>vruiLineCnt</code>	.1.3.6.1.4.1.1357.1.3.1.4	Number of physical phone lines on the system.
	<code>vruiSpanCnt</code>	.1.3.6.1.4.1.1357.1.3.1.5	Number of spans on the system. Also indicates number of rows in the <code>spanTable</code> for the VRU.
	<code>vruiHostCnt</code>	.1.3.6.1.4.1.1357.1.3.1.6	Number of hosts defined for the system. Also indicates number of rows in the <code>hostTable</code> for the VRU.
	<code>vruiState</code>	.1.3.6.1.4.1.1357.1.3.1.8	The current status of the VRU entity.
	<code>vruiStateLastChange</code>	.1.3.6.1.4.1.1357.1.3.1.9	Date and time the VRU entity last changed state.
Control	<code>vruiAdminState</code>	.1.3.6.1.4.1.1357.1.3.1.7	Allows the network manager to bring the VRU system up or down.



The objects `vruid`, `vruiPAddress`, `vruiState`, and `vruiStateLastChange`, though still supported, are similarly represented by the objects `componentId`, `componentIPAddress`, `componentState`, and `componentStateLastChange`, respectively, when the component is an MPS.

## Line Subtree

Defines the parameters of the lines on the system. The variables can both monitor and control the lines. Requests on the table in this subtree must include the instance identification provided by both `lineVruId` and `lineId`.

	Variable	OID	Condition
Indexes	<code>lineVruId</code>	.1.3.6.1.4.1.1357.1.4.1.1	Identifies the VRU that the phone line is associated with. Serves as the initial index into <code>lineTable</code> .
	<code>lineId</code>	.1.3.6.1.4.1.1357.1.4.1.2	Identifies the line number of this phone line, and serves as the second index into <code>lineTable</code> .
Monitor	<code>lineType</code>	.1.3.6.1.4.1.1357.1.4.1.3	Specifies whether the line is a logical or physical phone line or whether this is unknown. A value of unknown indicates that the agent is unable to connect to the VRU to determine the line type.
	<code>lineProtocol</code>	.1.3.6.1.4.1.1357.1.4.1.4	A value which indicates the set of protocols the phone line supports, presented as a sum.
	<code>lineState</code>	.1.3.6.1.4.1.1357.1.4.1.5	Current state of the line. A value of unknown indicates that the agent is unable to connect to the VRU to determine the line state. A logical line will always have a value of unknown.
	<code>lineAppOverlay</code>	.1.3.6.1.4.1.1357.1.4.1.7	The name of the application linked to the line.
	<code>lineAppState</code>	.1.3.6.1.4.1.1357.1.4.1.9	Current state of the application assigned to the logical line.
	<code>lineAppStateLast Change</code>	.1.3.6.1.4.1.1357.1.4.1.10	Date and time the current application assigned to the line last changed state.
	<code>lineAppCfgLast Change</code>	.1.3.6.1.4.1.1357.1.4.1.11	Date and time of last application configure or unconfigure on the line.

Variable		OID	Condition
Control	lineAppName	.1.3.6.1.4.1.1357.1.4.1.6	The name of the application currently assigned to the line, which will be returned by any response PDU on this object. This variable can only be set by setting lineAppAdminState to assign and this object to the application name.
	lineAppAdminState	.1.3.6.1.4.1.1357.1.4.1.8	Allows the network manager to set the desired state of the application.

### Span Table Subtree

Defines the parameters of the spans on the system. The variables described (other than indexing variables) are all of monitor (read-only) status. Requests on the table in this subtree must include the instance identification provided by both `spanVruId` and `spanId`.

Variable	OID	Condition
spanVruId	.1.3.6.1.4.1.1357.1.5.1.1	Identifies the VRU that the span is associated with. Serves as the initial index into <code>spanTable</code> .
spanId	.1.3.6.1.4.1.1357.1.5.1.2	Identifier of this span, and serves as the second index into <code>spanTable</code> .
spanLineIdStart	.1.3.6.1.4.1.1357.1.5.1.3	Line identification of the first line in the span.
spanLineNumber	.1.3.6.1.4.1.1357.1.5.1.4	Number of lines in the span.
spanEnabled	.1.3.6.1.4.1.1357.1.5.1.5	Indicates whether span is enabled or disabled.
spanState	.1.3.6.1.4.1.1357.1.5.1.6	Current state of the span.
spanStateLastChange	.1.3.6.1.4.1.1357.1.5.1.7	Date and time the span last changed state.

### Applications Subtree

Describes the applications associated with the VRU entity. Two variables are used for monitoring these conditions. Requests on the table in this subtree must include the instance identification provided by both `appVruId` and `appName`.

Variable	OID	Condition
<code>appVruId</code>	.1.3.6.1.4.1.1357.1.6.1.1	VRU that the application has been assigned to. Serves as the initial index into <code>appTable</code> .
<code>appName</code>	.1.3.6.1.4.1.1357.1.6.1.2	The name of an the application that can be assigned to the VRU. Serves as the second index into <code>appTable</code> .
<code>appOptions</code>	.1.3.6.1.4.1.1357.1.6.1.4	The options that are used during application startup.
<code>appLineCnt</code>	.1.3.6.1.4.1.1357.1.6.1.5	The number of lines the application is currently assigned to.

### Applications Stats Subtree

Describes the application statistic associated with an application. Two variables are used for monitoring only. Requests on the table in this subtree must include the instance identification provided by both `appStatsVruId` and `appStatsName`.

Variable	OID	Condition
<code>appStatsVruId</code>	.1.3.6.1.4.1.1357.1.7.1.1	VRU that the application statistic is associated with. Serves as the initial index into <code>appStatsTable</code> .
<code>appStatsName</code>	.1.3.6.1.4.1.1357.1.7.1.2	Terminology which identifies this application statistic. Serves as the second index into the <code>appStatsTable</code> .
<code>appStatsValue</code>	.1.3.6.1.4.1.1357.1.7.1.3	An application stats' value, which is gathered in 15 minute intervals.
<code>appStatsLastChange</code>	.1.3.6.1.4.1.1357.1.7.1.4	Date and time when the application stat was recorded.

### Host Subtree

Describes and defines the parameters of the host associated with the VRU entity. The variables can both monitor and control these parameters. Requests on the table in this subtree must include the instance identification provided by both `hostVruId` and `hostId`.

	Variable	OID	Condition
Indexes	<code>hostVruId</code>	.1.3.6.1.4.1.1357.1.8.1.1	Identifies the VRU that the host is associated with. Serves as the initial index into <code>hostTable</code> .
	<code>hostId</code>	.1.3.6.1.4.1.1357.1.8.1.2	Identifier of this host, and serves as the second index into <code>hostTable</code> .
Monitor	<code>hostLuCnt</code>	.1.3.6.1.4.1.1357.1.8.1.4	Number of LU's defined for the host. Also indicates the number of rows in the <code>luTable</code> for the host.
	<code>hostProtocol</code>	.1.3.6.1.4.1.1357.1.8.1.5	Protocol which is used for host communication.
	<code>hostMedia</code>	.1.3.6.1.4.1.1357.1.8.1.6	Media used for communicating with the host.
	<code>hostState</code>	.1.3.6.1.4.1.1357.1.8.1.9	Current state of the host.
	<code>hostStateLast Change</code>	.1.3.6.1.4.1.1357.1.8.1.10	Date and time host last changed state.
Control	<code>hostAdminState</code>	.1.3.6.1.4.1.1357.1.8.1.8	Allows the network manager to set the desired state of the host for hardware based hosts only (when <code>hostMedia=hardwarebased</code> ).

## LU Subtree

Describes and defines the parameters of the LUs associated with the system. The variables described are all of monitor (read-only) status. Requests on the table in this subtree must include the instance identification provided by the variables `luVruId`, `luHostId`, and `luId`.

Variable	OID	Condition
<code>luVruId</code>	.1.3.6.1.4.1.1357.1.9.1.1	Identifies the VRU that the LU is associated with. Serves as the initial index into <code>luTable</code> .
<code>luHostId</code>	.1.3.6.1.4.1.1357.1.9.1.2	Identifies the host that the LU is associated with. Serves as the second index into <code>luTable</code> .
<code>luId</code>	.1.3.6.1.4.1.1357.1.9.1.3	Identifier of this LU, and serves as the subsequent index into <code>luTable</code> .
<code>luPoolName</code>	.1.3.6.1.4.1.1357.1.9.1.5	Pool name that the LU is assigned to.
<code>luState</code>	.1.3.6.1.4.1.1357.1.9.1.6	Current state of the LU.
<code>luStateLastChange</code>	.1.3.6.1.4.1.1357.1.9.1.7	Date and time LU last changed state.

### Components Subtree

This subtree contains two tables that describe and define the components and their presence on the MPS network.

#### Component Type Table

Defines all the known component types. Requests on the read-only object in this table must be made using the instance identification provided by the variable `componentTypeId`.

Variable	OID	Condition
<code>componentTypeId</code>	.1.3.6.1.4.1.1357.1.10.1.1.1	Identifier for the component type. Acts as the index into <code>componentTypeTable</code> .
<code>componentTypeName</code>	.1.3.6.1.4.1.1357.1.10.1.1.2	Common name of the component type.

#### Component Table

Defines and describes all the components on the system that the agent has identified. Requests on the read-only variables in this table must include the instance identification provided by the variables `componentType` and `componentId`.

Variable	OID	Condition
<code>componentType</code>	.1.3.6.1.4.1.1357.1.10.3.1.1	Identifies the component type. Acts as the initial index into <code>componentTable</code> .
<code>componentId</code>	.1.3.6.1.4.1.1357.1.10.3.1.2	Identifier of the component, and the second index into <code>componentTable</code> . <sup>a</sup>
<code>componentIpAddress</code>	.1.3.6.1.4.1.1357.1.10.3.1.3	IP address of the component.
<code>componentState</code>	.1.3.6.1.4.1.1357.1.10.3.1.4	Current state of the component.
<code>componentStateLastChange</code>	.1.3.6.1.4.1.1357.1.10.3.1.5	Date and time the component last changed state.

a. To avoid possible problems, each component of a specific type on a particular network *must* be assigned a unique component ID.



The objects `componentId`, `componentIpAddress`, `componentState`, and `componentStateLastChange` will contain the same data as the objects `vrId`, `vrIpAddress`, `vrState`, and `vrStateLastChange`, respectively, when the component is an MPS.





# 4

## PeriView/PERIsnmp Mappings

This chapter covers:

1. Mappings

This chapter provides the SNMP manager a cross-reference to tools and functions available in PeriView that will accomplish the same or similar results discussed earlier in generic terms for utilizing the Periphonics enterprise MIB (see Chapter *This chapter discusses, in generic terms, the functions that can be controlled and monitored using SNMP in the MPS environment. These functions generally fall into the administrative and maintenance categories. The user is directed to the Nortel Networks, Periphonics Enterprise MIB (hereafter simply called MIB) variables that are monitored only, those that can be both monitored and controlled, and those that act as indexes into tables within the MIB. In addition, notifications generated by the agent and sent to the manager for disposition are also discussed. For a detailed description of each variable and its associated data, see Appendix The Nortel Networks MPS uses the Periphonics Enterprise MIB. The Periphonics Enterprise MIB is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is: on page 76. on page 42*). It is not meant as a learning guide or instructional tool in the use of PeriView. The chapter is formatted in enterprise MIB group order for ease of use by the manager, and includes only those variables for which an analogous PeriView function exists. For definitions of individual MIB objects, refer to the applicable table in *The MIB Applied on page 51*. For a detailed view of this MIB, see Appendix *The Nortel Networks MPS uses the Periphonics Enterprise MIB. The Periphonics Enterprise MIB is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is: on page 76*. For specific controls and functions beyond those discussed here, and a more detailed explanation and graphical representation of PeriView usage (including instructions on accessing tools and windows), refer to the *PeriView Reference Manual*.

## Mappings

### MIB-II

While the intention of this chapter is to provide information particular to Periphonics MPS product, there are a few items that fall within the parameters of the system portion of MIB-II that may be of interest to the user. These variables are used to describe in strictly textual form certain aspects of the MPS system. They are `sysDescr`, which describes the workstation in use; `sysLocation`, which indicates the physical location of the workstation; `sysName`, which is a textual description of the MPS network; and `sysContact`, which gives information pertaining to personal contact available with regard to the station.

## Enterprise MIB

### Events Group Notifications Subtree

This group defines the traps sent to the network management station. Traps are analogous to alarms in PeriView. As such, particular objects themselves are not *directly* supported by PeriView, but there are certain indications in PeriView that can be tied into the issuance of a trap. They are as follows:

This MIB Object is analogous to...	This representation...	In this PeriView tool.
notifyAlarmByComponentEx <sup>a</sup>	Alarm Icon	Alarm Viewer
notifyComponentStateChg <sup>b</sup>	1) MPS, OSCAR icons (change in color).  2) MPS icons (change in color).  3) Failure of graph to update, or beeping/flashing graph icon.	1) Tree  2) APPMAN  3) Line Status (also displayed in Tree).
notifyLineAppStateChg <sup>c</sup>	Application icon (change in color).	APPMAN (also displayed in Tree)
notifySpanStateChg <sup>d</sup>	Span state symbology (change of symbol).	Span Status (also displayed in Tree)
notifyHostStateChg <sup>e</sup>	Host state symbology (change of symbol).	Host Status
notifyLuStateChg <sup>f</sup>	VT state symbology (change in symbol). VTs in PeriView are equivalent to LUs in MIB.	Host Status (when indexed by VT).

- The alarms themselves are detailed and defined in the Component Events Alarms Subtree (see [Component Events Alarms Subtree](#) on page 63).
- Generated when a change in either `vruseState` (see [vruseState](#) on page 66) or `componentState` (see [componentState](#) on page 73) is detected. Both `vruseState` and `componentState` will contain the same data for MPSs.
- Generated when a change in `lineAppState` is detected (see [lineAppState](#) on page 67).
- Generated when a change in `spanState` is detected (see [spanState](#) on page 69).
- Generated when a change in `hostState` is detected (see [hostState](#) on page 71).
- Generated when a change in `luState` is detected (see [luState](#) on page 72).

### Component Events Alarms Subtree

This group describes the aspects of component generated alarms and the parameters for maintaining them in the database, and the Periphonics enterprise MIB variables shown below are used to define specifically the contents of the `notifyAlarmByComponentEx` trap (see [notifyAlarmByComponentEx](#) on page 63). Most of the Periphonics enterprise MIB variables in this group have comparable indications in the PeriView Alarm Viewer. The following illustration shows a suitably configured file displaying typical entries containing information analogous to each of the MIB variables.

Alarm Viewer [ No Filter ]

Alarm Viewer

M	Arrival Time	Number	Counts	Severity	Source	Node	Line	Host
X	Thu Sep 18, 2003 11:03:47	02040	6	info	console	mps	0	0
X	Thu Sep 18, 2003 11:11:41	00001	2	info	srp	mps	0	0
X	Thu Sep 18, 2003 11:11:41	12012	2	info	srp	mps	0	0
X	Thu Sep 18, 2003 11:11:24	12001	1	warning	srp	mps	0	0

Alarm Messages

Time: Thu Sep 18 11:11:24, From <srp>, Alarm Number [12001], Source: common:0  
( Line# 0, Host# 0, Severity Class: warning, Component: common, Node: mps )  
Alarm Message:  
nriod#common.0/MPS: DOWN (pid: 15968, exit code: 15) - restarting  
  
\*\*\*\*\* Total Number of Displayed Alarms: [ 1 ] \*\*\*\*\*

Using the first entry of the preceding illustration as an example, we can compare the variables to their equivalent PeriView representations. PeriView alarms are predefined, and information for each is stored in the alarm database.

- **alarmLogSeverity:** Indication of how severe an alarm is considered. There are three severity types - Info, Warning and Alarms.
- **alarmLogCode:** Individual code assigned to particular alarms. The example entry shows this at 21410.
- **alarmLogLineId:** The phone line associated with the alarm. In the MIB, if no line is associated, the value is set to 0. In the PeriView illustration, the line identification is defined under the Line 10 entry. In this case, no entry would exist if no phone line were associated with the alarm.



Note that the line numbers in PeriView are indicative of the logical line with which the physical line is associated. For additional information pertaining to this logical-to-physical line mapping, refer to the *PeriView Reference Manual*.

- **alarmLogHostId:** Identifies the host associated with the alarm. The MIB has an assigned value of 0 for this variable if no host is related to the alarm. For PeriView, the entry `Host 1`, for example, would indicate the host association. This entry would appear in the same location as (but instead of) the line identification cited above. All other associated data in the alarm would likewise change. Should no host be associated with the alarm, no entry would be made.
- **alarmLogProcName:** The name of the process that generated the alarm. This is indicated in brackets in the example as `<ccm>`.
- **alarmLogMessage:** A textual description of the alarm. In PeriView, this is carried on the second and subsequent line of each entry made.
- **alarmLogTime:** The date and time that the alarm was generated. In the example illustrated, this is shown by the initial entry `Tue Sep 17 11:36:43`.
- **alarmLogComponentType:** Identifies the component that generated the alarm. The example above indicates that a MPS generated the alarm. Had another component generated the alarm, it would be identified at the same location.
- **alarmLogComponentId:** Component identifier of the component that generated the alarm. In the example this is the number 6 following MPS (the component type) in the entry `MPS 6`.

All of these variables are also viewable in the Alarm Viewer. However, the Alarm Viewer window shows live alarms, which are added to the Alarm Log File as they are generated.

## VRU Subtree

Defines the parameters of the VRU system. In the PeriView environment, this is represented by the MPS system. The MIB variables and their PeriView counterparts are as follows:

This <b>MIB Object</b> is analogous to...	This representation...	In this <b>PeriView tool</b> .
vruId	1) Listed under current MPS number. 2) Number after colon in MPS identification	1) vsh <b>vhman</b> 2) APPMAN, Line Status, Host Status, Alarm Filter, MPS Control Center, Span Status, Tree
vruLineCnt	1) Total sum of represented states. 2) Sum of physical phone line icons; cursor placement over last physical phone line icon (line number representation); line number representation in right margin. <sup>a</sup> 3) Line count.	1) Line Status Graph (with vertical axis lines set to absolute). 2) Line Status window. 3) vsh Phone Line Status report (system-wide). <sup>b</sup>
vruSpanCnt	Quantity of indicated spans.	Span Status (also displayed in Tree)
vruHostCnt	Quantity of indicated hosts.	Host Status
vruState <sup>c</sup>	1) MPS icons (color of). 2) MPS state symbology (type of symbol).	1) APPMAN, Tree 2) Span Status and Host Status (also displayed in Tree)
vruStateLastChange	Time and date stamp in report of MPS command line. <sup>d</sup>	vsh SRP Status and SRP Group Status reports.

- Second and third items assume all applicable rows of table are completely covered with physical phone line icons; otherwise, appropriate adjustments should be made. Note that although this icon represents a physical line, the line numbers in the table are indicative of the logical line with which the physical line is associated. For additional information pertaining to this logical-to-physical line mapping, refer to the *PeriView Reference Manual*.
- The quantity of lines shown in the report is indicative of the total number of *physical* phone lines on the system. However, the actual phone line number indicated at each entry within the body of the report is the *logical* phone line location on the MPS. Though the physical-to-logical phone line mapping is most often on a one-to-one basis, it should not be assumed that the report line numbers indicate the actual location of the physical phone line on the MPS. For additional information, refer to the *PeriView Reference Manual*.
- Any change to this particular data is analogous to one of the reasons for the `notifyComponentStateChg` trap being generated (see [notifyComponentStateChg](#) on page 63).
- The time displayed in the report differs by a few seconds from the actual time the change took place. This is due to the delay between the time the report is initiated (data queried) and the time the data is received by the management station.

## Line Subtree

Defines the parameters of the lines on the system. This is handled in PeriView through the tools and windows that follow.



The representation of lines shown in PeriView is indicative of the *physical* phone lines on the system. However, the actual phone line *number* indicated at each entry is the *logical* phone line location on the MPS. Though the physical-to-logical phone line

mapping is most often on a one-to-one basis, it should not be assumed that the line numbers indicate the actual location of the physical phone line on the MPS. For additional, specific information, please refer to the *PeriView Reference Manual*.

This <b>MIB Object</b> is analogous to...	This representation...	In this <b>PeriView tool</b> .
lineVruId	1) MPS number at command line request. 2) Listed as MPS at command line prompt. 3) Number after colon in MPS identification	1) vsh SRP Status report. 2) vsh CCM Status report. 3) APPMAN, Span Status, Host Status, Line Status, Tree
lineId	1) Listed under <code>LINE</code> column. 2) Listed in body of report by line number. 3) Listed in windows by line number.	1) vsh SRP Status report. 2) vsh CCM Status report. 3) APPMAN, Line Status
lineType	Phone line icon.	APPMAN, Tree
lineState	Listed in body of report next to line number.	vsh CCM Status report; Activity Monitor CCM Status graphs.
lineAppName	1) Textual representation of application name and line number. 2) Textual representation of application name with corresponding unique color resident on phone lines to which it applies.	1) APPMAN 2) APPMAN, Span Status, Host Status (when indexed by line), Tree
lineAppOverlay	Not directly correlated on a per line basis: linked applications are displayed in this tool on a per-component basis.	Linked APP graph.
lineAppAdminState	Window names imply the functions managed by these tools.	APPMAN
lineAppState <sup>a</sup>	1) Phone line icon (by color). 2) Presence or absence of applications listed on phone lines shown.	1) APPMAN, Tree 2) Span Status

This <b>MIB Object</b> is analogous to...	This representation...	In this <b>PeriView</b> tool.
lineAppStateLastChange	Application name listed in right-most column, use time and date stamp in report. <sup>b</sup>	vsh SRP Status report.
lineAppCfgLastChange	Entry under STATE of RUNNING indicates configure, EXITED indicates unconfigure; use time and date stamp in report. <sup>c</sup>	vsh SRP Status report.

- a. Any change to this particular data is analogous to the reason for the notifyLineAppStateChg trap being generated (see [notifyLineAppStateChg](#) on page 63).
- b. The time displayed in the report differs by a few seconds from the actual time the change took place. This is due to the delay between the time the report is initiated (data queried) and the time the data is received by the management station.
- c. The results of the SRP Status Report are generated at the time the report is called by the user. Each time the user repeats the procedure, a new set of data is generated. By using the scroll bar along the right side of the screen, comparisons can be made amongst the reports generated. By leaving the screen active and generating reports by choice, a running tabulation can be kept, and the above information extrapolated, for the applications (for example the addition or deletion of applications can be determined).



## Span Table Subtree

Defines the parameters of the spans on the system. These parameters are all displayed by PeriView in the Span Status tool.



The line numbers in PeriView are indicative of the logical line with which the physical line is associated. For additional information pertaining to this logical-to-physical line mapping, refer to the *PeriView Reference Manual*.

This MIB Object is analogous to...	This representation...	In this PeriView tool.
spanVruId	Number after colon in MPS identification	Span Status.
spanId	Listed by span name.	Span Status.
spanLineIdStart	First number under Line Range next to each span listing.	Span Status.
spanLineNumber	Number of lines listed under Line Range. This information becomes cumulative when more than one span exists: simple mathematics for those spans above the first will yield the desired result. Alternatively, add up number of lines per span shown in Lines: field.	Span Status.
spanEnabled	Span state symbology (type of symbol).	Span Status.
spanState <sup>a</sup>	Span state symbology (type of symbol).	Span Status.

- a. Any change to this particular data is analogous to the reason for the `notifySpanStateChg` trap being generated (see [notifySpanStateChg](#) on page 63).

### Applications Subtree

Describes the applications that are associated with a VRU. PeriView APPMAN functions handle the analogous relationships to this.



Note that the line numbers in PeriView are indicative of the logical line with which the physical line is associated. For additional information pertaining to this logical-to-physical line mapping, refer to the *PeriView Reference Manual*.

This <b>MIB Object</b> is analogous to...	This representation...	In this <b>PeriView tool</b> .
appVruId	Number after colon in MPS identification.	APPMAN
appName	1) Textual representation of application name.  2) Textual representation of application name with corresponding unique color.	1) APPMAN  2) APPMAN, Span Status, Host Status
appOptions	Options listed when launched from an application object.	APPMAN, Application Configuration
appLineCnt	1) Textual representation of application name and line number. Quantities of each application assignment can be counted up.  2) Textual representation of application name with corresponding unique color resident on phone lines to which it applies. Quantities of each application assignment can be counted up.  3) Sum of all states of application shown.	1) APPMAN  2) APPMAN (configurations only) Host Status(when indexed by line).  3) Line Status bar graph (set to Absolute and launched on an application object).

## Applications Stats Subtree

Describes the application statistic associated with an application. In PeriView, this data is presented in the form of an application report through the PeriReporter Tools. These tools generate user-defined application and system reports. For detailed information on using these tools, refer to the *PeriReporter User's Guide*.

This MIB Object is analogous to...	This representation...	In this PeriView tool.
appStatsVruId	Can be identified by checking the column mapping menu.	PeriReporter.
appStatsName	Defined at top of each column.	PeriReporter.
appStatsValue	Listed in column(s) if application statistics is selected in the Data Mapping window.	PeriReporter.
appStatsLastChange	Shown in Date column of report based on date and time constraints set up in Schedule Statistics Report tool.	PeriReporter.

## Host Subtree

Describes and defines the parameters of the host associated with the VRU. In PeriView, this is accomplished in the Host Status tool. The MIB variables and their PeriView counterparts are discussed below.

This MIB Object is analogous to...	This representation...	In this PeriView tool.
hostVruId	Number after colon in MPS identification	Host Status
hostId	Listed by host name.	Host Status
hostLuCnt	Symbology when tool is indexed by VT (PeriView identifies LUs as VTs). Count up the number of symbols in the VTs : field other than that of unassigned.	Host Status
hostProtocol	The tool has a specific entry for this.	Host Status
hostState <sup>a</sup>	Host state symbology (type of symbol).	Host Status

a. Any change to this particular data is analogous to the reason for the notifyHostStateChg trap being generated (see [notifyHostStateChg](#) on page 63).

## LU Subtree

Describes and defines the parameters of the LUs associated with the system. PeriView equates LUs with VTs. The status of LUs can be determined in PeriView through the

use of the Host Status tool. The tool must be indexed by VT to access this information.

This <b>MIB Object</b> is analogous to...	This representation...	In this <b>PeriView</b> tool.
luVruId	Number after colon in MPS identification	Host Status
luHostId	Host name selected.	Host Status
luId	Numbers along right side of tool adjacent to symbolic representation of VTs.	Host Status
luState <sup>a</sup>	VT state symbology (type of symbol) in VTs: field.	Host Status

- a. Any change to this particular data is analogous to the reason for the `notifyLuStateChg` trap being generated (see [notifyLuStateChg](#) on page 63).

## Components Subtree

This subtree contains two tables that describe and define the components and their presence on the MPS network.

### Component Type Table

The only MIB object in this table that has a comparable representation in PeriView is `componentTypeName`. This object contains the common noun name (acronym) associated with each component type (for example, MPS and OSCAR).

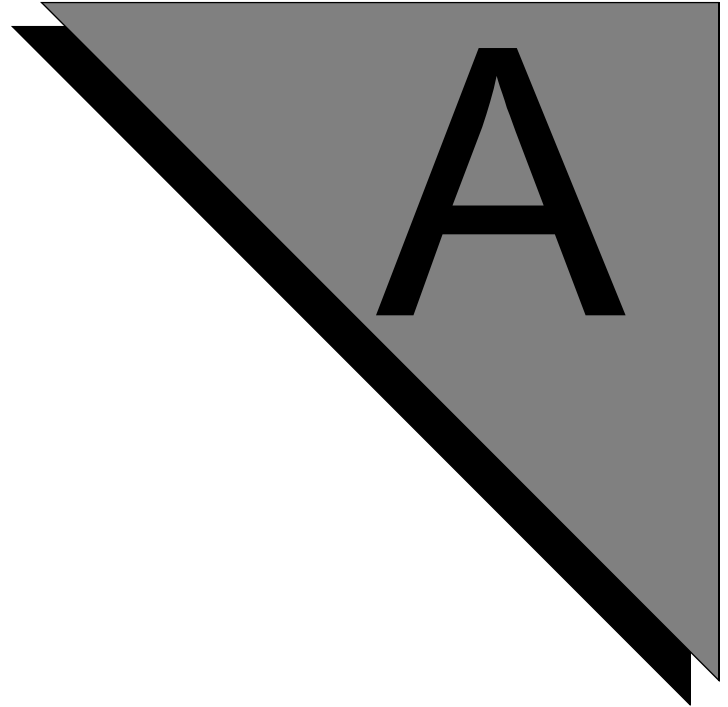
### Component Table

This table defines and describes all the components on the network that the SNMP agent knows about.

This <b>MIB Object</b> is analogous to...	This representation...	In this <b>PeriView tool</b> .
<code>componentId</code>	Number after colon in component identification	APPMAN, Line Status, Span Status, Host Status, MPS Control Center, Alarm Filter, Tree
<code>componentState<sup>a</sup></code>	1) MPS, OSCAR icons (color of icon). 2) MPS icon (color of icon). 3) MPS icons (color of icons). 4) MPS state symbology (type of symbol).	1) APPMAN, Tree 2) APPMAN 3) APPMAN NA
<code>componentStateLastChange</code>	Time and date stamp in report of component command line. <sup>b</sup>	vsh SRP Status and SRP Group Status reports.

- a. Any change to this particular data is analogous to the reason for the `notifyComponentStateChg` trap being generated (see [notifyComponentStateChg](#) on page 63).
- b. The time displayed in the report differs by a few seconds from the actual time the change took place. This is due to the delay between the time the report is initiated (data queried) and the time the data is received by the management station.

**This page has been intentionally left blank.**



# **Peripherals Enterprise MIB Reference**

**This chapter covers:**

- 1. Basic Tree Structure**
- 2. Detailed Component Structure**

The Nortel Networks MPS uses the *Periphonics Enterprise MIB*. The *Periphonics Enterprise MIB* is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is:

```
iso.org.dod.internet.private.enterprises.periphonics
1.3.6.1.4.1.1357
```

This appendix shows the objects that make up the *Periphonics Enterprise MIB*. It includes both a graphical hierarchy in tree form for easy reference, and a more comprehensive view detailing the structure of each entry in the enterprise MIB. It is provided for those people who desire a better understanding of the function of these items as they apply to the MPS environments.

### Basic Tree Structure

This figure provides a simple way to navigate through the MIB and determine the relationship of objects within it.



The following variables, though present, have not been implemented for use at this time: `vruDescr`, `appDescr`, `hostDescr`, and `luDescr`. In addition, the objects `vruNetworkMIBCompliance`, `vruNetworkMIBComplianceByComponent`, `vruNotificationsGroup`, `vruEventsGroup`, `componentNotificationsGroup`, `componentEventsGroup`, `notifyAlarm`, `notifyAlarmByComponent`, `notifyVruStateChg`, and `almLogVruId`, though still present in the MIB, have a status of deprecated and are no longer supported.

```
periphonics (1357)
|
+-- vruNetwork(1)
|
+-- vruNetworkMIBConformance(1)
| |
| | +-- vruNetworkMIBCompliances(1)
| | |
| | | +-- vruNetworkMIBCompliance(1)
| | | +-- vruNetworkMIBComplianceByComponent (2)
| | | +-- vruNetworkMIBComplianceByComponent2 (3)
| | |
| | +-- vruNetworkMIBGroups(2)
| | |
| | | +-- vruNotificationsGroup(1)
| | | +-- vruEventsGroup(2)
| | | +-- vruGroup(3)
| | | +-- lineGroup(4)
| | | +-- applicationGroup(5)
| | | +-- hostGroup(6)
| | | +-- componentGroup(7)
| | | +-- componentNotificationsGroup(8)
| | | +-- componentEventsGroup(9)
```



```

|   +-- componentNotificationsGroup2(10)
|   +-- componentsEventsGroup2(11)
|
+-- vruEvents(2)
| |
| | +-- vruEventVars(1)
| | |
| | | +-- -N- Object  evObjectId(1)
| | | +-- -N- EnumVal evChange(2)
| | |
| | +-- vruNotifications(2)
| | |
| | | +-- notifyAlarm(1)
| | | +-- notifyVruStateChg(2)
| | | +-- notifyLineAppStateChg(3)
| | | +-- notifySpanStateChg(4)
| | | +-- notifyHostStateChg(5)
| | | +-- notifyLuStateChg(6)
| | | +-- notifyTopologyChg(7)
| | | +-- notifyComponentStateChg(8)
| | | +-- notifyAlarmByComponent(9)
| | | +-- notifyAlarmByComponentEx(10)
| | |
| | +-- alarms(3)
| | |
| | | +-- -R- Integer alrmLogMax(4)
| | | +-- -R- Integer alrmLogNumber(5)
| | | +-- alrmLogTable(6)
| | | |
| | | | +-- alrmLogTableEntry(1)
| | | | |
| | | | | +-- -N- Integer alrmLogIdx(1)
| | | | | +-- -R- Integer alrmLogSeverity(2)
| | | | | +-- -R- Integer alrmLogCode(3)
| | | | | +-- -R- Integer alrmLogVruId(4)
| | | | | +-- -R- Integer alrmLogLineId(5)
| | | | | +-- -R- Integer alrmLogHostId(6)
| | | | | +-- -R- TextVal alrmLogProcName(7)
| | | | | +-- -R- TextVal alrmLogMessage(8)
| | | | | +-- -R- Object alrmLogTime(9)
| | | | | +-- -R- Integer alrmLogComponentType(10)
| | | | | +-- -R- Integer alrmLogComponentId(11)
| | | | | +-- -R- IPAddr alrmLogComponentIpAddress(12)
|

```

```
+-- vruTable(3)
| |
| +-- vruTableEntry(1)
| |
| +-- -N- Integer vruId(1)
| +-- -R- IPAddr vruIpAddress(2)
| +-- -RW TextVal vruDescr(3)
| +-- -R- Integer vruLineCnt(4)
| +-- -R- Integer vruSpanCnt(5)
| +-- -R- Integer vruHostCnt(6)
| +-- -RW EnumVal vruAdminState(7)
| +-- -R- EnumVal vruState(8)
| +-- -R- Object vruStateLastChange(9)
|
+-- lineTable(4)
| |
| +-- lineTableEntry(1)
| |
| +-- -N- Integer lineVruId(1)
| +-- -N- Integer lineId(2)
| +-- -R- EnumVal lineType(3)
| +-- -R- Integer lineProtocol(4)
| +-- -R- EnumVal lineState(5)
| +-- -RW TextVal lineAppName(6)
| +-- -R- TextVal lineAppOverlay(7)
| +-- -RW EnumVal lineAppAdminState(8)
| +-- -R- EnumVal lineAppState(9)
| +-- -R- Object lineAppStateLastChange(10)
| +-- -R- Object lineAppCfgLastChange(11)
|
+-- spanTable(5)
| |
| +-- spanTableEntry(1)
| |
| +-- -N- Integer spanVruId(1)
| +-- -N- Integer spanId(2)
| +-- -R- Integer spanLineIdStart(3)
| +-- -R- EnumVal spanLineNumber(4)
| +-- -R- EnumVal spanEnabled(5)
| +-- -R- EnumVal spanState(6)
| +-- -R- Object spanStateLastChange(7)
|
```

```

+-- appTable(6)
| |
| +-- appTableEntry(1)
| |
| | +-- -N- Integer  appVruId(1)
| | +-- -N- TextVal  appName(2)
| | +-- -RW TextVal  appDescr(3)
| | +-- -R- TextVal  appOptions(4)
| | +-- -R- Integer  appLineCnt(5)
|
+-- appStatsTable(7)
| |
| +-- appStatsTableEntry(1)
| |
| | +-- -N- Integer  appStatsVruId(1)
| | +-- -N- TextVal  appStatsName(2)
| | +-- -R- Integer  appStatsValue(3)
| | +-- -R- Object   appStatsLastChange(4)
|
+-- hostTable(8)
| |
| +-- hostTableEntry(1)
| |
| | +-- -N- Integer  hostVruId(1)
| | +-- -N- Integer  hostId(2)
| | +-- -RW TextVal  hostDescr(3)
| | +-- -R- Integer  hostLuCnt(4)
| | +-- -R- EnumVal  hostProtocol(5)
| | +-- -R- EnumVal  hostMedia(6)
| | +-- -RW EnumVal  hostAdminState(8)
| | +-- -R- EnumVal  hostState(9)
| | +-- -R- Object   hostStateLastChange(10)
|
+-- luTable(9)
| |
| +-- luTableEntry(1)
| |
| | +-- -N- Integer  luVruId(1)
| | +-- -N- Integer  luHostId(2)
| | +-- -N- Integer  luId(3)
| | +-- -R- TextVal  luDescr(4)
| | +-- -R- TextVal  luPoolName(5)
| | +-- -R- EnumVal  luState(6)
| | +-- -R- Object   luStateLastChange(7)

```

```
+-- components(10)
|
| +-- componentTypeTable(1)
| |
| | +-- componentTypeTableEntry(1)
| | |
| | | +-- -N- Integer componentTypeId(1)
| | | +-- -R- TextVal componentTypeName(2)
| |
|
+-- componentTable(3)
|
| +-- componentTableEntry(1)
| |
| | +-- -N- Integer componentType(1)
| | +-- -N- Integer componentId(2)
| | +-- -R- IPAddr componentIpAddress(3)
| | +-- -R- EnumVal componentState(4)
| | +-- -R- Object componentStateLastChange(5)
```

### Detailed Component Structure

This section provides more defined references to each component of the *Periphonics Enterprise MIB*.



The following variables, though present, have not been implemented for use at this time: vruDescr, appDescr, hostDescr, and luDescr. In addition, the objects vruNetworkMIBCompliance, vruNetworkMIBComplianceByComponent, vruNotificationsGroup, vruEventsGroup, componentNotificationsGroup, componentEventsGroup, notifyAlarm, notifyAlarmByComponent, notifyVruStateChg, and alrmLogVruId, though still present in the MIB, have a status of deprecated and are no longer supported.

```
vruNetwork MODULE-IDENTITY
    LAST-UPDATED "9808050000Z"
    ORGANIZATION "Nortel Networks Corporation"
    CONTACT-INFO
        ""
    DESCRIPTION
        "SNMP MIB module for a VRU Network."
    ::= { periphonics 1 }
```

```
--
-- Conformance Information
--
```

```
vrNetworkMIBConformance OBJECT IDENTIFIER ::= { vrNetwork 1 }

vrNetworkMIBCompliances OBJECT IDENTIFIER
::={ vrNetworkMIBConformance 1 }
vrNetworkMIBGroups OBJECT IDENTIFIER
::= { vrNetworkMIBConformance 2 }

--
-- Compliance Statements
--

vrNetworkMIBCompliance MODULE-COMPLIANCE
    STATUS deprecated
    DESCRIPTION
        "The compliance statement for SNMPv2 entities
        which implement the VRU-NETWORK MIB."

    MODULE -- this module
        MANDATORY-GROUPS { vrNotificationsGroup, vrEventsGroup,
                           vrGroup, lineGroup, applicationGroup, hostGroup }
    ::= { vrNetworkMIBCompliances 1 }

vrNetworkMIBComplianceByComponent MODULE-COMPLIANCE
    STATUS deprecated
    DESCRIPTION
        "The compliance statement for SNMPv2 entities
        which implement the VRU-NETWORK MIB."

    MODULE -- this module
        MANDATORY-GROUPS { vrGroup, lineGroup, applicationGroup,
                           hostGroup, componentGroup,
                           componentNotificationsGroup,
                           componentEventsGroup }
    ::= { vrNetworkMIBCompliances 2 }

vrNetworkMIBComplianceByComponent2 MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
        "The compliance statement for SNMPv2 entities
        which implement the VRU-NETWORK MIB."

    MODULE -- this module
        MANDATORY-GROUPS { vrGroup, lineGroup, applicationGroup,
                           hostGroup, componentGroup,
                           componentNotificationsGroup2,
                           componentEventsGroup2 }
    ::= { vrNetworkMIBCompliances 3 }

--
-- units of conformance
--
```

### vrnNotificationsGroup NOTIFICATION-GROUP

NOTIFICATIONS { notifyAlarm, notifyVruStateChg,  
notifyLineAppStateChg, notifySpanStateChg,  
notifyHostStateChg, notifyLuStateChg,  
notifyTopologyChg }

STATUS deprecated

#### DESCRIPTION

"A collection of notifications related to a Voice Response Unit (VRU)."

::= { vruNetworkMIBGroups 1 }

### vrnEventsGroup OBJECT-GROUP

OBJECTS { evObjectId, evChange, alrmLogMax, alrmLogNumber,  
alrmLogSeverity, alrmLogCode, alrmLogVruId,  
alrmLogLineId, alrmLogHostId, alrmLogProcName,  
alrmLogMessage, alrmLogTime }

STATUS deprecated

#### DESCRIPTION

"A collection of objects related to a VRU."

::= { vruNetworkMIBGroups 2 }

### vrnGroup OBJECT-GROUP

OBJECTS { vruIpAddress, vruDescr, vruLineCnt,  
vruSpanCnt, vruHostCnt, vruAdminState, vruState,  
vruStateLastChange }

STATUS current

#### DESCRIPTION

"A collection of objects related to a VRU."

::= { vruNetworkMIBGroups 3 }

### lineGroup OBJECT-GROUP

OBJECTS { lineType, lineProtocol, lineState, lineAppName,  
lineAppOverlay, lineAppAdminState, lineAppState,  
lineAppStateLastChange, lineAppCfgLastChange,  
spanLineIdStart, spanLineNumber, spanEnabled,  
spanState, spanStateLastChange }

STATUS current

#### DESCRIPTION

"A collection of objects related to a line on a VRU."

::= { vruNetworkMIBGroups 4 }

### applicationGroup OBJECT-GROUP

OBJECTS { appDescr, appOptions, appLineCnt, appStatsValue,

```

                                appStatsLastChange }
STATUS                          current
DESCRIPTION
    "A collection of objects related to a VRU application."
::= { vruNetworkMIBGroups 5 }

```

hostGroup OBJECT-GROUP

```

OBJECTS      { hostDescr, hostLuCnt, hostProtocol, hostMedia,
                hostAdminState, hostState, hostStateLastChange,
                luDescr, luPoolName, luState, luStateLastChange }
STATUS       current
DESCRIPTION
    "A collection of objects related to a VRU host."
::= { vruNetworkMIBGroups 6 }

```

componentGroup OBJECT-GROUP

```

OBJECTS      { componentTypeName, componentIpAddress,
                componentState, componentStateLastChange }
STATUS       current
DESCRIPTION
    "A collection of objects related to components."
::= { vruNetworkMIBGroups 7 }

```

componentNotificationsGroup NOTIFICATION-GROUP

```

NOTIFICATIONS { notifyLineAppStateChg, notifySpanStateChg,
                notifyHostStateChg, notifyLuStateChg,
                notifyTopologyChg, notifyComponentStateChg,
                notifyAlarmByComponent }
STATUS        deprecated
DESCRIPTION
    "A collection of notifications related to the MPS network."
::= { vruNetworkMIBGroups 8 }

```

componentEventsGroup OBJECT-GROUP

```

OBJECTS      { evObjectId, evChange, alrmLogMax, alrmLogNumber
                alrmLogSeverity, alrmLogCode, alrmLogLineId,
                alrmLogHostId, alrmLogProcName, alrmLogMessage,
                alrmLogTime, alrmLogComponentType,
                alrmLogComponentId }
STATUS       deprecated
DESCRIPTION
    "A collection of objects related to a VRU."
::= { vruNetworkMIBGroups 9 }

```

componentNotificationsGroup2 NOTIFICATION-GROUP

```

NOTIFICATIONS { notifyLineAppStateChg, notifySpanStateChg,
                notifyHostStateChg, notifyLuStateChg,

```

```

                                notifyTopologyChg, notifyComponentStateChg,
                                notifyAlarmByComponentEx }
STATUS                          current
DESCRIPTION
    "A collection of notifications related to the MPS network."
 ::= { vruNetworkMIBGroups 10 }

componentEventsGroup2 OBJECT-GROUP
    OBJECTS                    { evObjectId, evChange, alrmLogMax, alrmLogNumber
                                alrmLogSeverity, alrmLogCode, alrmLogLineId,
                                alrmLogHostId, alrmLogProcName, alrmLogMessage,
                                alrmLogTime, alrmLogComponentType,
                                alrmLogComponentId, alrmLogComponentIpAddress }
    STATUS                      current
    DESCRIPTION
        "A collection of objects related to a VRU."
 ::= { vruNetworkMIBGroups 11 }

--
-- VRU Events Subtree
--

vruEvents      OBJECT IDENTIFIER ::= { vruNetwork 2 }

--
-- VRU Events Variables Subtree
--
-- Defines variables that may be included within a VRU Network trap. These
-- object IDs cannot be retrieved from the SNMP Agent. Defining them here
-- serves as documentation for the contents of NMS traps.
---
vruEventVars OBJECT IDENTIFIER ::= { vruEvents 1 }

evObjectId OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  accessible-for-notify
    STATUS      current
    DESCRIPTION
        "Object identifier associated with the source of the trap. This
        object cannot be retrieved from the SNMP agent."
 ::= { vruEventVars 1 }

evChange OBJECT-TYPE
    SYNTAX      INTEGER {
        other(1),
        add(2),
        del(3),
```



```

        mod(4)
    }
    MAX-ACCESS    accessible-for-notify
    STATUS        current
    DESCRIPTION
        "Describes the type of change which just occurred. This object
        cannot be retrieved from the SNMP agent."
    ::= { vruEventVars 2 }

--
-- Events Group Notifications Subtree
--
-- Defines traps that are sent to Network Management Stations.
--

vruNotifications OBJECT IDENTIFIER ::= { vruEvents 2 }

notifyAlarm NOTIFICATION-TYPE
    OBJECTS        { alrmLogSeverity, alrmLogCode, alrmLogVruId,
                    alrmLogLineId, alrmLogHostId, alrmLogProcName,
                    alrmLogMessage, alrmLogTime }
    STATUS        deprecated
    DESCRIPTION
        "When a exceptional condition occurs the VRU system will
        generate an alarm. This trap forwards these alarms to the
        Network Management Station.

        **NOTE: this trap is deprecated and replaced by
        notifyAlarmByComponent trap."
    ::= { vruNotifications 1 }

notifyVruStateChg NOTIFICATION-TYPE
    OBJECTS        { vruState }
    STATUS        deprecated
    DESCRIPTION
        "This trap is forwarded to the Network Management Station when
        the VRU changes its state.

        **NOTE: this trap is deprecated and replaced by the
        notifyComponentStateChg trap."
    ::= { vruNotifications 2 }

notifyLineAppStateChg NOTIFICATION-TYPE
    OBJECTS        { lineAppState, lineAppName, lineAppOverlay }
    STATUS        current
    DESCRIPTION
        "This trap is forwarded to the Network Management Station when
        an application attached to a VRU line changes its state."
    ::= { vruNotifications 3 }

```

### notifySpanStateChg NOTIFICATION-TYPE

OBJECTS { spanState }

STATUS current

#### DESCRIPTION

"This trap is forwarded to the Network Management Station when a VRU span changes its state."

::= { vruNotifications 4 }

### notifyHostStateChg NOTIFICATION-TYPE

OBJECTS { hostState }

STATUS current

#### DESCRIPTION

"This trap is forwarded to the Network Management Station when a VRU host changes its state."

::= { vruNotifications 5 }

### notifyLuStateChg NOTIFICATION-TYPE

OBJECTS { luState }

STATUS current

#### DESCRIPTION

"This trap is forwarded to the Network Management Station when a VRU host lu changes its state."

::= { vruNotifications 6 }

### notifyTopologyChg NOTIFICATION-TYPE

OBJECTS { evObjectId, evChange }

STATUS current

#### DESCRIPTION

"This trap is forwarded to the Network Management Station when a VRU Network object topology status has changed."

::= { vruNotifications 7 }

### notifyComponentStateChg NOTIFICATION-TYPE

OBJECTS { componentState }

STATUS current

#### DESCRIPTION

"This trap is forwarded to the Network Management Station when a VRU Network component changes its state."

::= { vruNotifications 8 }

### notifyAlarmByComponent NOTIFICATION-TYPE

OBJECTS { alrmLogComponentType, alrmLogComponentId,  
alrmLogSeverity, alrmLogCode, alrmLogLineId,  
alrmLogHostId, alrmLogProcName, alrmLogMessage,  
alrmLogTime }

STATUS deprecated

#### DESCRIPTION

"When an exceptional condition occurs in the MPS network a component will generate an alarm. This trap forwards these alarms to the Network Management Station."

\*\*NOTE: this trap is deprecated and replaced by the notifyAlarmByComponentEx trap."  
::= { vruNotifications 9 }

notifyAlarmByComponentEx NOTIFICATION-TYPE

OBJECTS { almLogComponentType, almLogComponentId, almLogSeverity, almLogCode, almLogLineId, almLogHostId, almLogProcName, almLogMessage, almLogTime, almLogComponentIpAddress }

STATUS current

DESCRIPTION

"When an exceptional condition occurs in the MPS network a component will generate an alarm. This trap forwards these alarms to the Network Management Station. This is an extended version of notifyAlarmByComponent and includes the IP address of the component generating the alarm."

::= { vruNotifications 10 }

--  
-- Component Events Alarms Subtree  
--  
-- Defines what component alarms are recorded and for how long.  
--

alarms OBJECT IDENTIFIER ::= { vruEvents 3 }

almLogMax OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Specifies the maximum number of alarms that may be logged in the alarm log table at once. When almLogNumber is equivalent to almLogMax, the first alarm in the table will be removed when the next alarm is added"

::= { alarms 4 }

almLogNumber OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of log entries currently defined."

::= { alarms 5 }

almLogTable OBJECT-TYPE

SYNTAX SEQUENCE OF AlrmLogTableEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"A list of alarm entries. The number of entries is given by alrmLogNumber. This list could be thought of as a queue. When the component generates an alarm, it is added to the end of the list. If the number of entries reaches alrmLogMax then the first entry will be deleted before a new one is added."

::= { alarms 6 }

almLogTableEntry OBJECT-TYPE

SYNTAX AlrmLogTableEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An alarm entry containing objects that describe that alarm."

INDEX { alrmLogIdx }

::= { alrmLogTable 1 }

AlrmLogTableEntry ::= SEQUENCE {

almLogIdx	INTEGER,
almLogSeverity	INTEGER,
almLogCode	INTEGER,
almLogVruId	INTEGER,
almLogLineId	INTEGER,
almLogHostId	INTEGER,
almLogProcName	DisplayString,
almLogMessage	DisplayString,
almLogTime	DateAndTime,
almLogComponentType	INTEGER,
almLogComponentId	INTEGER,
almLogComponentIpAddress	IPAddress

}

almLogIdx OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Index into the alarm log table between one and alrmLogNumber."

::= { alrmLogTableEntry 1 }

**alarmLogSeverity OBJECT-TYPE****SYNTAX** INTEGER (0..9)**MAX-ACCESS** read-only**STATUS** current**DESCRIPTION**

"Integer describing how severe this alarm is. The lower the value, the lower the severity. A severity value of 0 indicates that no severity has been assigned to this alarm."

**::=** { alarmLogTableEntry 2 }**alarmLogCode OBJECT-TYPE****SYNTAX** INTEGER (0..2147483647)**MAX-ACCESS** read-only**STATUS** current**DESCRIPTION**

"Code assigned to identify this alarm."

**::=** { alarmLogTableEntry 3 }**alarmLogVruId OBJECT-TYPE****SYNTAX** INTEGER (0..2147483647)**MAX-ACCESS** read-only**STATUS** deprecated**DESCRIPTION**

"VRU identifier which generated this alarm. A value of 0 is used to indicate no VRU is associated with this alarm."

**\*\*NOTE:** this object is deprecated and has been replaced by alarmLogComponentId object."

**::=** { alarmLogTableEntry 4 }**alarmLogLineId OBJECT-TYPE****SYNTAX** INTEGER (0..2147483647)**MAX-ACCESS** read-only**STATUS** current**DESCRIPTION**

"The phone line number associated with this alarm. A value of 0 indicates no phone line."

**::=** { alarmLogTableEntry 5 }**alarmLogHostId OBJECT-TYPE****SYNTAX** INTEGER (0..2147483647)**MAX-ACCESS** read-only**STATUS** current**DESCRIPTION**

"The host identifier associated with this alarm. A value of 0 indicates that no host is associated with this alarm."

::= { alrmLogTableEntry 6 }

alrmLogProcName OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The name of the process that generated this alarm."

::= { alrmLogTableEntry 7 }

alrmLogMessage OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"A message describing the alarm that occurred."

::= { alrmLogTableEntry 8 }

alrmLogTime OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The time that this alarm was generated."

::= { alrmLogTableEntry 9 }

alrmLogComponentType OBJECT-TYPE

SYNTAX INTEGER (0..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Component type of the MPS component which generated this alarm."

::= { alrmLogTableEntry 10 }

alrmLogComponentId OBJECT-TYPE

SYNTAX INTEGER (0..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Component identifier of the MPS component which generated this alarm."

::= { alrmLogTableEntry 11 }

## alarmLogComponentIpAddress OBJECT-TYPE

SYNTAX           IpAddress

MAX-ACCESS      read-only

STATUS           current

## DESCRIPTION

"Component IP address of the MPS component which generated  
this alarm"

::= { alarmLogTableEntry 12 }

--

-- VRU Subtree

--

## vruTable OBJECT-TYPE

SYNTAX SEQUENCE OF VruTableEntry

MAX-ACCESS      not-accessible

STATUS           current

## DESCRIPTION

"All Voice Response Units (VRU) an agent knows about"

::= { vruNetwork 3 }

## vruTableEntry OBJECT-TYPE

SYNTAX           VruTableEntry

MAX-ACCESS      not-accessible

STATUS           current

## DESCRIPTION

"A (conceptual) row entry for one VRU."

INDEX { vruId }

::= { vruTable 1 }

## VruTableEntry ::= SEQUENCE {

vruId                 INTEGER,

vruIpAddress         IpAddress

vruDescr             DisplayString

vruLineCnt           INTEGER,

vruSpanCnt           INTEGER,

vruHostCnt           INTEGER,

vruAdminState        INTEGER,

vruState             INTEGER,

```
vruParamLastChange    DateAndTime
}
```

### vruParam OBJECT-TYPE

```
SYNTAX                INTEGER (1..2147483647)
```

```
MAX-ACCESS            not-accessible
```

```
STATUS                current
```

#### DESCRIPTION

"The VRU number associated with this VRU entity. Note that is a unique number within VRU network."

```
::= { vruTableEntry 1 }
```

### vruParamIp OBJECT-TYPE

```
SYNTAX                IpAddress
```

```
MAX-ACCESS            read-only
```

```
STATUS                current
```

#### DESCRIPTION

"VRU network ip address."

```
::= { vruTableEntry 2 }
```

### vruParamDescr OBJECT-TYPE

```
SYNTAX                DisplayString
```

```
MAX-ACCESS            read-write
```

```
STATUS                current
```

#### DESCRIPTION

"A textual description of this VRU entity. It is mandatory that this contain only printable ASCII characters."

```
::= { vruTableEntry 3 }
```

### vruParamLineCnt OBJECT-TYPE

```
SYNTAX                INTEGER (0..2147483647)
```

```
MAX-ACCESS            read-only
```

```
STATUS                current
```

#### DESCRIPTION

"Number of physical phone lines on this system."

```
::= { vruTableEntry 4 }
```

### vruParamSpanCnt OBJECT-TYPE

```
SYNTAX                INTEGER (0..2147483647)
```

```
MAX-ACCESS            read-only
```

```
STATUS                current
```

#### DESCRIPTION

"Number of spans on this system. This also indicates the number of rows in the spanTable(4) for this VRU."

```
::= { vruTableEntry 5 }
```

### vruParamHostCnt OBJECT-TYPE



SYNTAX            INTEGER (0..2147483647)  
MAX-ACCESS       read-only  
STATUS            current  
DESCRIPTION  
    "Number of hosts defined for this system. This also indicates  
    the number of rows in the hostTable(6) for this VRU."  
::= { vruTableEntry 6 }

vruAdminState OBJECT-TYPE

SYNTAX            INTEGER {  
    other(1),  
    startup(2),  
    shutdown(3),  
    recycle(4),  
    selftest(5),  
    checkpoint(6)-- save any dynamic info to disk  
    }  
MAX-ACCESS       read-write  
STATUS            current  
DESCRIPTION  
    "The network managers desired state of this VRU entity."  
::= { vruTableEntry 7 }

vruState OBJECT-TYPE

SYNTAX            INTEGER {  
    other(1),  
    unknown(2),  
    init(3),  
    up(4),  
    down(5),  
    selftest(6)  
    }  
MAX-ACCESS       read-only  
STATUS            current  
DESCRIPTION  
    "The current status of the VRU entity."  
::= { vruTableEntry 8 }

vruStateLastChange OBJECT-TYPE

SYNTAX            DateAndTime  
MAX-ACCESS       read-only  
STATUS            current  
DESCRIPTION  
    "Date and time that this VRU entity changed its state."  
::= { vruTableEntry 9 }

--  
-- Line Subtree

--

### lineTable OBJECT-TYPE

SYNTAX SEQUENCE OF LineTableEntry

MAX-ACCESS not-accessible

STATUS current

#### DESCRIPTION

"All physical and logical lines that are attached to a Voice Response Unit (VRU)."

::= { vruNetwork 4 }

### lineTableEntry OBJECT-TYPE

SYNTAX LineTableEntry

MAX-ACCESS not-accessible

STATUS current

#### DESCRIPTION

"A (conceptual) entry for one physical/logical line."

INDEX { lineVruId, lineId }

::= { lineTable 1 }

### LineTableEntry ::= SEQUENCE {

lineVruId	INTEGER,
lineId	INTEGER,
lineType	INTEGER,
lineProtocol	INTEGER,
lineState	INTEGER,
lineAppName	DisplayString,
lineAppOverlay	DisplayString,
lineAppAdminState	INTEGER,
lineAppState	INTEGER,
lineAppStateLastChange	DateAndTime,
lineAppCfgLastChange	DateAndTime

}

### lineVruId OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

#### DESCRIPTION

"The VRU identifier that this phone line is associated with."

::= { lineTableEntry 1 }

### lineId OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The line number of this phone line. This number is unique for each VRU."

::= { lineTableEntry 2 }

lineType OBJECT-TYPE

SYNTAX INTEGER {

other(1),  
unknown(2),  
virtual(3),  
analog(4),  
digital(5)  
}

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Specifies whether this line is a logical(3) or a physical(4) phone line. A line type of unknown(2) indicates that we can't determine the line type at this moment."

::= { lineTableEntry 3 }

### lineProtocol OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS read-only

STATUS current

#### DESCRIPTION

"A value which indicates the set of protocols this phone line offers. The value is a sum. This value initially takes the value of zero which indicates that the protocol is unknown. Then for each protocol that this line supports, its protocol value is added to the sum.

Value	Functionality
0x00001	Uses some other type of protocol
0x00002	Some analog protocol
0x00004	AT&T Bell release
0x00008	British Paging (BTUK/MCP)
0x00010	Channel bank with 2 wire option
0x00020	Channel bank with 2 wire FXOR else FXO
0x00040	Channel bank with 4 wire E&M
0x00080	Channel bank FSX[r] else FXO[R]
0x00100	Channel bank loop start else ground
0x00200	Channel bank standard access
0x00400	Common channel signalling (SS7)
0x00800	Galaxy 5/2 ACD
0x01000	ISDN
0x02000	Loop hangup supervision
0x04000	Mercury CAS signalling
0x08000	Network side protocol
0x10000	R1 signaling system
0x20000	R2 signaling with DTMF outdial
0x40000	R2 signaling system
"	

::= { lineTableEntry 4 }

**lineState OBJECT-TYPE**

**SYNTAX**            **INTEGER** {  
                  other(1),  
                  unknown(2),  
                  connected(3),  
                  idle(4),  
                  busy(5),  
                  referral(6),  
                  down(7)  
                  }

**MAX-ACCESS**    read-only

**STATUS**            current

**DESCRIPTION**

"Current state of this line. The line may be currently servicing a call (connected), in service but not in a call service or referral state (idle), not in service nor down (busy), referral (referral), or down (down). If we are unable to connect to the VRU than the line state will be unknown (unknown). A logical line (line) state will always be unknown."

::= { lineTableEntry 5 }

**lineAppName OBJECT-TYPE**

**SYNTAX**            **DisplayString**

**MAX-ACCESS**    read-write

**STATUS**            current

**DESCRIPTION**

"The name of the application that is currently assigned to this line. Any response PDU on this object will return the name of the application assigned to this line. If there is no application assigned to this line, a 0 length string is returned.

Assigning a new application to this line is achieved by setting lineAppAdminState to assign and this object to the applications name."

::= { lineTableEntry 6 }

**lineAppOverlay OBJECT-TYPE**

**SYNTAX**            **DisplayString**

**MAX-ACCESS**    read-only

**STATUS**            current

**DESCRIPTION**

"The name of the application that has been linked to this line."

::= { lineTableEntry 7 }

```
lineAppAdminState OBJECT-TYPE
    SYNTAX      INTEGER {
        other(1),
        startapp(2),
        softterm(3),
        hardterm(4),
        assign(5),
        unassign(6),
        hardrestart(7),
        softrestart(8)
    }
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "Desired state that Network Manager wants the application in."
    ::= { lineTableEntry 8 }
```

```
lineAppState OBJECT-TYPE
    SYNTAX      INTEGER {
        other(1),
        unknown(2),
        noapp(3),
        init(4),
        up(5),
        down(6),
        config(7),
        exit(8)
    }
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Current state of the application that is assigned to this
        logical line. If no applications are assigned this entry will
        be noapp(3)"
    ::= { lineTableEntry 9 }
```

```
lineAppStateLastChange OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Date and time that the current application assigned to this
        line last changed its state."
    ::= { lineTableEntry 10 }
```

```
lineAppCfgLastChange OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Date and time of last application configure or unconfigure
        on this line."
    ::= { lineTableEntry 11 }

---
--- Span Table
---

spanTable OBJECT-TYPE
    SYNTAX SEQUENCE OF SpanTableEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "All spans which are attached to a Voice Response Unit (VRU)."
    ::= { vruNetwork 5 }

spanTableEntry OBJECT-TYPE
    SYNTAX      SpanTableEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "A (conceptual) entry for one span."
    INDEX { spanVruId, spanId }
    ::= { spanTable 1 }

SpanTableEntry ::= SEQUENCE {
    spanVruId      INTEGER,
    spanId         INTEGER,
    spanLineIdStart INTEGER,
    spanLineNumber INTEGER,
    spanEnabled    TruthValue,
    spanState      INTEGER,
    spanStateLastChange DateAndTime
}

spanVruId OBJECT-TYPE
    SYNTAX      INTEGER (1..2147483647)
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "VRU identifier that this span is associated with."
    ::= { spanTableEntry 1 }
```

### spanId OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

#### DESCRIPTION

"Identifier of this span."

::= { spanTableEntry 2 }

### spanLineIdStart OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS read-only

STATUS current

#### DESCRIPTION

"Line id of the first line in this span."

::= { spanTableEntry 3 }

### spanLineNumber OBJECT-TYPE

SYNTAX INTEGER {  
    europaLineNumber(30),  
    usaLineNumber(24)  
}

MAX-ACCESS read-only

STATUS current

#### DESCRIPTION

"Number of lines in this span."

::= { spanTableEntry 4 }

### spanEnabled OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

#### DESCRIPTION

"Indicates whether span is enabled or disabled. A value of true(1) indicates span is enabled and a value of false(2) indicates span is disabled."

::= { spanTableEntry 5 }



## spanState OBJECT-TYPE

```
SYNTAX      INTEGER {
    other(1),
    unknown(2),-- span state can not be determined
    init(3),-- span is in an initial state
    green(4),-- span is online and ready
    fatal(5),-- span entered failed signal state
    yellow(6),-- span is in yellow state
    red(7)-- span is in red state
}
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Current state of this span."
::= { spanTableEntry 6 }
```

## spanStateLastChange OBJECT-TYPE

```
SYNTAX      DateAndTime
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Date and time that the span last changed its state."
::= { spanTableEntry 7 }
```

```
--
-- Applications Subtree
--
```

## appTable OBJECT-TYPE

```
SYNTAX SEQUENCE OF AppTableEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "All applications that VRU currently knows about."
::= { vruNetwork 6 }
```

## appTableEntry OBJECT-TYPE

```
SYNTAX      AppTableEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A (conceptual) entry for one VRU application"
INDEX { appVruId, IMPLIED appName }
::= { appTable 1 }
```

```
AppTableEntry ::= SEQUENCE {  
    appVruId    INTEGER,  
    appName     DisplayString,  
    appDescr    DisplayString,  
    appOptions  DisplayString,  
    appLineCnt  INTEGER  
}
```

```
appVruId OBJECT-TYPE  
    SYNTAX      INTEGER(1..2147483647)  
    MAX-ACCESS  not-accessible  
    STATUS      current  
    DESCRIPTION  
        "VRU id that this application has been assigned to."  
    ::= { appTableEntry 1 }
```

```
appName OBJECT-TYPE  
    SYNTAX      DisplayString ( SIZE(1..255) )  
    MAX-ACCESS  not-accessible  
    STATUS      current  
    DESCRIPTION  
        "The name of the application that can be assigned to a  
        line. This name must be unique for each VRU"  
    ::= { appTableEntry 2 }
```

```
appDescr OBJECT-TYPE  
    SYNTAX      DisplayString  
    MAX-ACCESS  read-write  
    STATUS      current  
    DESCRIPTION  
        "A textual description of this application. It is mandatory  
        that this contain only printable ASCII characters."  
    ::= { appTableEntry 3 }
```

```
appOptions OBJECT-TYPE  
    SYNTAX      DisplayString  
    MAX-ACCESS  read-only  
    STATUS      current  
    DESCRIPTION  
        "The options that are needed during this applications startup."  
    ::= { appTableEntry 4 }
```

## appLineCnt OBJECT-TYPE

SYNTAX INTEGER(0..2147483647)

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"The number of lines that this application is assigned to."

::= { appTableEntry 5 }

--

-- Applications Stats Subtree

--

## appStatsTable OBJECT-TYPE

SYNTAX SEQUENCE OF AppStatsTableEntry

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"All application specific stats that have been collected."

::= { vruNetwork 7 }

## appStatsTableEntry OBJECT-TYPE

SYNTAX AppStatsTableEntry

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"A (conceptual) entry for one application stat"

INDEX { appStatsVruId, IMPLIED appStatsName

::= { appStatsTable 1 }

## AppStatsTableEntry ::= SEQUENCE {

appStatsVruId INTEGER,

appStatsName DisplayString,

appStatsValue INTEGER,

appStatsLastChange DateAndTime

}

## appStatsVruId OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"The VRU identifier that this application statistic is associated with."

::= { appStatsTableEntry 1 }

```
appStatsName OBJECT-TYPE
    SYNTAX      DisplayString ( SIZE(1..255) )
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "Name which identifies this application stat."
    ::= { appStatsTableEntry 2 }

appStatsValue OBJECT-TYPE
    SYNTAX      INTEGER (1..2147483647)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "An application stats value. This value is gathered on 15 minute
        intervals."
    ::= { appStatsTableEntry 3 }

appStatsLastChange OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Date and time when this application stat was recorded."
    ::= { appStatsTableEntry 4 }

--
-- Host Subtree
--

hostTable OBJECT-TYPE
    SYNTAX SEQUENCE OF HostTableEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "All hosts defined for a VRU."
    ::= { vruNetwork 8 }

hostTableEntry OBJECT-TYPE
    SYNTAX      HostTableEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "A (conceptual) entry for one VRU host"
    INDEX { hostVruId, hostId }
    ::= { hostTable 1 }
```

```
HostTableEntry ::= SEQUENCE {  
    hostVruId          INTEGER,  
    hostId             INTEGER,  
    hostDescr          DisplayString,  
    hostLuCnt          INTEGER,  
    hostProtocol       INTEGER,  
    hostMedia          INTEGER,  
    hostAdminState     INTEGER,  
    hostState          INTEGER,  
    hostStateLastChange DateAndTime  
}
```

hostVruId OBJECT-TYPE

```
SYNTAX      INTEGER (1..2147483647)  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
    "The VRU identifier that this host is associated with."  
::= { hostTableEntry 1 }
```

hostId OBJECT-TYPE

```
SYNTAX      INTEGER (1..2147483647)  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
    "Unique identifier associated with this host."  
::= { hostTableEntry 2 }
```

hostDescr OBJECT-TYPE

```
SYNTAX      DisplayString  
MAX-ACCESS  read-write  
STATUS      current  
DESCRIPTION  
    "A textual description of this host. It is mandatory that  
    this contain only printable ASCII characters."  
::= { hostTableEntry 3 }
```

hostLuCnt OBJECT-TYPE

```
SYNTAX      INTEGER (0..2147483647)  
MAX-ACCESS  read-only  
STATUS      current  
DESCRIPTION  
    "Number of lu's defined for this host. This also indicates  
    the number of rows in the luTable(7) for this host."  
::= { hostTableEntry 4 }
```

### hostProtocol OBJECT-TYPE

```
SYNTAX      INTEGER {
    other(1),
    unknown(2),
    atte(3),
    async(4),
    lu62(5),
    sna3270(6),
    sdlcexp(7),
    vpsn3270(8)
}
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Protocol which is used for host communications."
::= { hostTableEntry 5 }
```

### hostMedia OBJECT-TYPE

```
SYNTAX      INTEGER {
    other(1),
    unknown(2),
    hardwarebased(3),
    softwarebased(4)
}
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Media used for communicating with the host."
::= { hostTableEntry 6 }
```

### hostAdminState OBJECT-TYPE

```
SYNTAX      INTEGER {
    other(1),
    init(2),
    up(3),
    down(4),
    recycle(5),
    selftest(6)
}
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "Desired state that Network Manager wants the host in."
::= { hostTableEntry 8 }
```

## hostState OBJECT-TYPE

```
SYNTAX      INTEGER {
    other(1),
    unknown(2),
    init(3),
    up(4),
    down(5),
    selftest(6)
}
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Current state of the host."
::= { hostTableEntry 9 }
```

## hostStateLastChange OBJECT-TYPE

```
SYNTAX      DateAndTime
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Date and time this host last changed it state."
::= { hostTableEntry 10 }
```

```
--
-- LU Subtree
--
```

## luTable OBJECT-TYPE

```
SYNTAX SEQUENCE OF LuTableEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "All logical units (LU) defined for hosts on a VRU."
::= { vruNetwork 9 }
```

## luTableEntry OBJECT-TYPE

```
SYNTAX      LuTableEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A (conceptual) entry for one LU"
INDEX { luVruId, luHostId, luId }
::= { luTable 1 }
```

```
LuTableEntry ::= SEQUENCE {
    luVruId      INTEGER,
    luHostId     INTEGER,
    luId         INTEGER,
    luDescr      DisplayString,
    luPoolName   DisplayString,
    luState      INTEGER,
    luStateLastChange  DateAndTime
}

luVruId OBJECT-TYPE
    SYNTAX      INTEGER (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "VRU entity that this LU is associated with."
    ::= { luTableEntry 1 }

luHostId OBJECT-TYPE
    SYNTAX      INTEGER (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Host entity that this LU is associated with."
    ::= { luTableEntry 2 }

luId OBJECT-TYPE
    SYNTAX      INTEGER (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "LU identifier that is associated with this LU."
    ::= { luTableEntry 3 }

luDescr OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A textual description of this LU. It is mandatory that
        this contain only printable ASCII characters."
    ::= { luTableEntry 4 }
```



## luPoolName OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"Pool name that this LU is assigned to."

::= { luTableEntry 5 }

## luState OBJECT-TYPE

SYNTAX INTEGER {

other(1),

unknown(2),

init(3),

up(4),

down(5)

}

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"Current state of this LU."

::= { luTableEntry 6 }

## luStateLastChange OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"Date and time this lu last changed it state."

::= { luTableEntry 7 }

--

-- Component Subtree

--

components OBJECT IDENTIFIER ::= { vruNetwork 10 }

--

--Table of all defined component types

--

## componentTypeTable OBJECT-TYPE

SYNTAX SEQUENCE OF ComponentTypeTableEntry

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"All defined component types."

::= { components 1 }

```
componentTypeTableEntry OBJECT-TYPE
    SYNTAX          ComponentTypeTableEntry
    MAX-ACCESS      not-accessible
    STATUS           current
    DESCRIPTION
        "A (conceptual) entry for one component type."
    INDEX { componentTypeId }
    ::= { componentTypeTable 1 }
```

```
ComponentTypeTableEntry ::= SEQUENCE {
    componentTypeId      INTEGER,
    componentTypeName    DisplayString
}
```

```
componentTypeId OBJECT-TYPE
    SYNTAX          INTEGER (1..2147483647)
    MAX-ACCESS      not-accessible
    STATUS           current
    DESCRIPTION
        "Identifier for this component type."
    ::= { componentTypeTableEntry 1 }
```

```
componentTypeName OBJECT-TYPE
    SYNTAX          DisplayString
    MAX-ACCESS      read-only
    STATUS           current
    DESCRIPTION
        "Name of this component type."
    ::= { componentTypeTableEntry 2 }
```

```
--
--Table of all components an agent knows about.
--
```

```
componentTable OBJECT-TYPE
    SYNTAX SEQUENCE OF ComponentTableEntry
    MAX-ACCESS      not-accessible
    STATUS           current
    DESCRIPTION
        "All components found in the system."
    ::= { components 3 }
```

## componentTableEntry OBJECT-TYPE

SYNTAX ComponentTableEntry

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"A (conceptual) entry for one component."

INDEX { componentType, componentId }

::= { componentTable 1 }

## ComponentTableEntry ::= SEQUENCE {

componentType INTEGER,

componentId INTEGER,

componentIpAddress IpAddress,

componentState INTEGER,

componentStateLastChange DateAndTime

}

## componentType OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"Type of this component."

::= { componentTableEntry 1 }

## componentId OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"Identifier of this component."

::= { componentTableEntry 2 }

## componentIpAddress OBJECT-TYPE

SYNTAX IpAddress

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

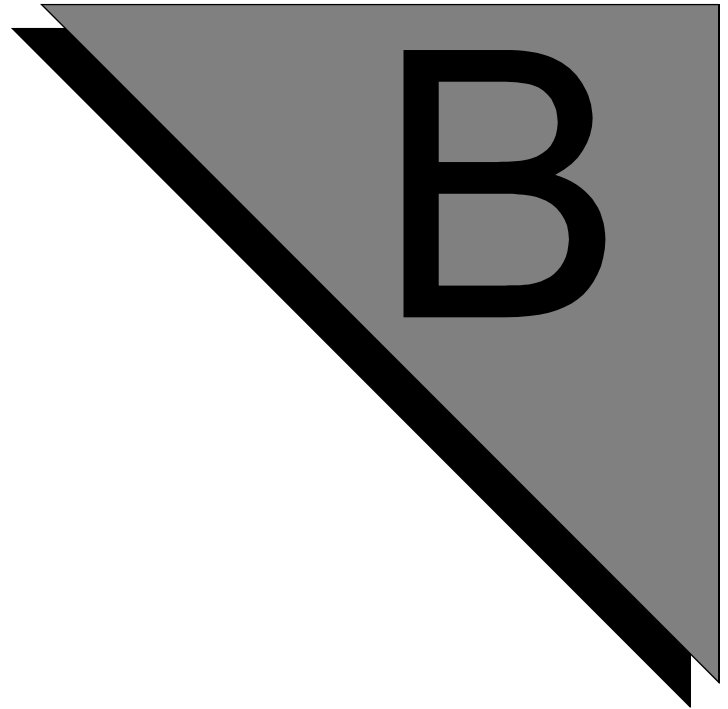
"Ip address of this component."

::= { componentTableEntry 3 }

```
componentState OBJECT-TYPE
    SYNTAX      INTEGER {
                                other (1)
                                unknown (2)
                                init (3)
                                up (4)
                                down (5)
                        }
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Current state of this component."
    ::= { componentTableEntry 4 }

componentStateLastChange OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Date and time this component last changed its state."
    ::= { componentTableEntry 5 }

END
```



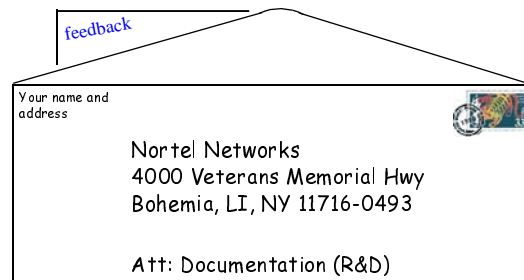
# Troubleshooting/ FAQs

This chapter covers:

1. Troubleshooting
2. Frequently Asked Questions (FAQs)

This appendix is divided into two sections designed to help you use your PERIsnmp package. The first section covers possible undesirable situations you might encounter during your use of PERIsnmp, their causes, and suggestions on how to alleviate them (see [Troubleshooting on page 114](#)). The second section lists some Frequently Asked Questions (and not-so-frequently asked questions!) and provides answers and/or links within the manual to them (see [Frequently Asked Questions \(FAQs\) on page 119](#)).

If you are reading this document online and wish to comment on or contribute to this information, please use the “mail to” link at the bottom of this page. If you are paperbound, you can mail us your feedback at:



## Troubleshooting

This section presents some common fault conditions, their causes, and suggested actions. Try using these entries prior to contacting Periphonics technical support.

- . **Condition:**  
The following message is found in the `vrnsnmpd.log` file (see [Error Logging on page 37](#)):  
Tue Aug 18 08:51:00 1998 Problem with license. Cannot talk to the LicenseServ server on host >>scrubjay<< Server not running?. FILE: vpsresources/vrnsnmp\_license.c LINE: 198
- . **Cause:**
  - 1) No license exists for Periphonics SNMP agent.
  - 2) License server is not running or is configured incorrectly.
  - 3) LSHOST is not set to the correct node running the license server.
- . **Action:**
  - 1) Make sure you've obtained and properly installed the license file (see [Package Licensing on page 14](#)).
  - 2) Check if you have a properly configured and running license server (consult your systems administrator for assistance if necessary).
  - 3) Make sure the environment variable LSHOST contains the name of the node running the license server.

- 
- . **Condition:**  
The following message is found in the `vrnsnmpd.log` file (see [Error Logging](#)

on page 37):

Tue Aug 25 09:35:00 1998 Error on master agent file descriptor, attempt to reconnect. FILE: vpsresources/vrsnmp\_ipc.c LINE: 1438

- **Cause:**  
Periphonics MPS components subagent is unable to connect to the master agent (see [Agents on page 38](#)).
- **Action:**  
Verify that the master agent is running (see [Verify Agents' Response on page 17](#)).

- 
- **Condition:**  
When trying to run **snmpstart**, you receive the following message:  
ERROR: Unable to start service Periphonics\_vrsnmpd
  - **Cause:**  
Periphonics MPS components subagent (see [Agents on page 38](#)) could not be started.
  - **Action:**  
Make sure that the service is not already running. Use **snmpstop** or the services panel to stop the service, then repeat the **snmpstart** command (see [Windows 2000 Systems on page 18](#)).



This scenario can apply to any of the Periphonics Windows 2000 SNMP agents.

- 
- **Condition:**  
When trying to run **snmpstop**, you receive the following message:  
ERROR: Unable to stop service Periphonics\_vrsnmpd
  - **Cause:**  
Periphonics MPS components subagent (see [Agents on page 38](#)) could not be stopped.
  - **Action:**  
Check to make sure that the service was not already stopped. Use **snmpstart** or the services panel to start the service, then repeat the **snmpstop** command (see [Windows 2000 Systems on page 18](#)).



This scenario can apply to any of the Periphonics Windows 2000 SNMP agents.

- . **Condition:**  
When trying to run **snmpstop** or **snmpstart**, you receive the following message:  
ERROR: Unable to connect to service control manager
- . **Cause:**  
The process responsible for managing services is unavailable.
- . **Action:**  
Try stopping/starting the services through the control panel (see [Windows 2000 Systems on page 18](#)). Otherwise, reboot the system to re-establish contact with the service control manager.

- 
- . **Condition:**  
When trying to run get- or set- command line utilities, you receive the following message:  
Error code set in packet - No such variable name.  
Index: 1.
  - . **Cause:**
    - 1) The SNMP agent(s) are not running.
    - 2) There is no information available for the requested object.
    - 3) You have failed to append **.0** to a scalar request.
  - . **Action:**
    - 1) Make sure the agents (particularly the MPS components subagent) are running (see [Windows 2000 Systems on page 18](#)).
    - 2) Perform a **getnext**, **getmany**, or **gettab** command on the object table to see if the information you are requesting exists at that time (see [Table Referencing on page 46](#)).
    - 3) Re-enter the command with a **.0** following the object name (i.e. **alarmLogMax.0**) (see [Direct Referencing on page 46](#)).

- 
- . **Condition:**  
When trying to run a get- command line utility, you receive the following message:  
Cannot translate variable class: <variable\_name>  
(where variable name is the name of the object you were performing the “get” on).
  - . **Cause:**  
You entered an incorrect object name.
  - . **Action:**  
Be sure that you’ve correctly entered the object name, including case-dependency, no spaces, and “abbreviated” portions within the object name (i.e. in **alarmLogMax**, **alarm** is the “abbreviated” version of the word “alarm”, and the “L” in **Log** and “M” in **Max** need to be upper-case) (see Appendix [The Nortel Networks MPS uses the Periphonics Enterprise MIB. The Periphonics Enterprise MIB is registered with the IANA and has been assigned the number 1357. The Periphonics MIB OID is:](#) on page 76).



- **Condition:**  
Traps are not being generated/received.
- **Cause:**
  - 1) MPS components subagent is not running.
  - 2) The management station you are on was not configured to receive traps.
- **Action:**
  - 1) Make sure the MPS components subagent is running (see [Windows 2000 Systems on page 18](#)).
  - 2) Run the `trapcfg.pl` script or edit the `snmpd.cnf` file to enable the management station to receive traps (see [Defining Target Parameters on page 36](#)).

- 
- **Condition:**  
`trapcfg.pl` does not run. You receive the following message when trying to run `trapcfg.pl`:  
ERROR: Environment variable SR\_AGT\_CONF\_DIR not set
  - **Cause:**  
The environment variable specifying the location of `snmpd.cnf` was not set, or does not point to the proper location.
  - **Action:**  
Make sure `SR_AGT_CONF_DIR` is properly configured (see [Master/Subagent Related on page 23](#)).

- 
- **Condition:**  
Cannot view traps.
  - **Cause:**
    - 1) You have not run the `traprcv` executable.
    - 2) You are not configured as a station allowed to receive traps.
    - 3) Your eyes are closed.
  - **Action:**
    - 1) Enter `traprcv` at a command line (see [Viewing Traps on page 49](#)).
    - 2) Run the `trapcfg.pl` script or edit the `snmpd.cnf` file to enable the management station to receive traps (see [Defining Target Parameters on page 36](#)).
    - 3) Open your eyes.

- 
- **Condition:**  
`trapcfg.pl` does not configure the station you entered.
  - **Cause:**  
You did not enter a valid IP address
  - **Action:**  
Enter a proper numerical IP address (see [Defining Target Parameters on page 36](#)).

- 
- . **Condition:**  
An OID is received that does not seem to coincide with the request sent.
  - . **Cause:**  
The object table that you performed the request on is empty.
  - . **Action:**  
Check the Periphonics MIB to see if the returned value follows that of the one you requested. If a request is made on an “empty” table, the next available object containing information will normally be returned (see [Table Referencing](#) on page 46).

- 
- . **Condition:**  
A request on the `alarmLogTable` does not return any results even though alarms have been generated.
  - . **Cause:**  
The table that you performed the request on is empty.
  - . **Action:**  
This occurs if the MPS components subagent is stopped: each time that occurs, the table is emptied. The `alarmLogTable` will begin collecting alarms generated subsequent to a restart of the subagent (see [Component Events Alarms Subtree](#) on page 52).

- 
- . **Condition:**  
An alarm message longer than 255 characters (6 lines) is truncated.
  - . **Cause:**  
This is a physical limitation within the program.
  - . **Action:**  
There should be adequate information available to you based on the number of characters allowable in the alarm. If additional information is needed, first try performing a get-request on the object `alarmLogMessage` (see [Getting a Value](#) on page 48). If this still does not meet your needs, please contact your Periphonics support personnel.
-

## Frequently Asked Questions (FAQs)

This section presents the answers to some frequently (and some not-so-frequently) asked questions regarding SNMP and its relationship to the MPS environment.

**Question:**

What is SNMP?

**Answer:**

SNMP is a network management protocol by which management stations and agents communicate. In summary, SNMP provides a network management protocol for allowing managers (clients) to perform status and control functions by querying agents (servers) about the network elements described in various MIBs (databases) (see [Simple Network Management Protocol](#) on page 2).

**Question:**

Okay, so what is the MIB and what changes would customers make?

**Answer:**

The Management Information Base (MIB) is a collection of objects which describe one or more network elements. An MIB can be considered a form of database schema that contains the set of system components that can be queried or controlled (see [Management Information Base \(MIB\)](#) on page 3). *Therefore, customers should never make any changes to the MIBs!*

**Question:**

How do I use SNMP on an MPS system?

**Answer:**

This question has numerous parts to its answer, all of which are covered throughout this manual. The basic requirements, besides licensing and installation, are:

- Configure traps (specify which management stations will receive them) (see [Defining Target Parameters](#) on page 36)
- Make sure agents are running (see [Verify Agents' Response](#) on page 17)
- Understand the Periphonics enterprise MIB (see [The MIB Applied](#) on page 51)
- Use command line utilities to monitor and control the MPS environment (see [Instance Identification](#) on page 46 and [SNMP Applied](#) on page 48).

**Question:**

What is the difference between SNMP and TCP as protocols?

**Answer:**

SNMP “lives” on UDP and integrates MPS control and monitoring into existing management software (see [Management Information Base \(MIB\)](#) on page 3).

**Question:**

Does SNMP run over the same network structure as TCP, and does it use the same configuration files that the system normally uses?

**Answer:**

SNMP uses UDP ports 161 and 162, and has its own configuration files (see [Configuration Files](#) on page 21). Port 162 is used to receive traps on the management station, and port 161 is used by the SNMP master agent to receive get- and set-requests.

**Question:**

How can I tell that the PERIsnmp package is installed and running on my machine?

**Answer:**

You can check your MPSHOME directory to see if you find the PERIsnmp package listed, or use the **perirev** command to see if its listed there (see the *Nortel Networks Media Processing Server Series Solaris System Operator's Guide* and *Nortel Networks Media Processing Server Series Windows System Operator's Guide*, for information related to these procedures). To determine if the package is running, use the **getmany** command (see [Verify Agents' Response](#) on page 17).

**Question:**

How would SNMP packets be identified when using Snoop?

**Answer:**

Uses Snoop ports 161 and/or 162.

**Question:**

Does PERIsnmp need to be licensed?

**Answer:**

Yes! You must properly install and modify the license package and/or file as applicable (see [Package Licensing](#) on page 14). If the license doesn't exist, a message error will be logged to the `vrsmnpd.log` (see [Error Logging](#) on page 37).

**Question:**

What are some of the possible problems and conflicts I may encounter when using the product?

**Answer:**

Apparently you didn't read the first part of this appendix! Go back and check out [Troubleshooting on page 114](#). You might also want to see [Installation Upgrades on page 17](#).

**Question:**

If I encounter problems, is there somewhere I can look to see what might have happened?

**Answer:**

Yes there is! Error logging is now part of your PERIsnmp package. To find out more, see [Error Logging on page 37](#).

**Question:**

What impact might the package have on processor overhead and network traffic?

**Answer:**

Using the command line utilities will have a fairly small impact on your processor's overhead. The greatest impact on both the processor and network occurs when multiple management stations are designated as trap recipients (see [Defining Target Parameters on page 36](#)). The more stations designated as such, the greater the traffic and demand on processor power.

**Question:**

What, exactly, are traps anyway? How do I know if they are being generated, and how can I see them?

**Answer:**

Traps are unsolicited messages sent from the agent to the manager in the event of unexpected internal conditions within the MPS system. As a test, you can stop and start SRP to purposely generate a trap. To view traps, enter the **traprcv** command at the command line of any management station designated to receive traps. For important information related to these procedures, see [Viewing Traps on page 49](#).

**Question:**

What happens if I add a component to a system that is already running PERIsnmp?

**Answer:**

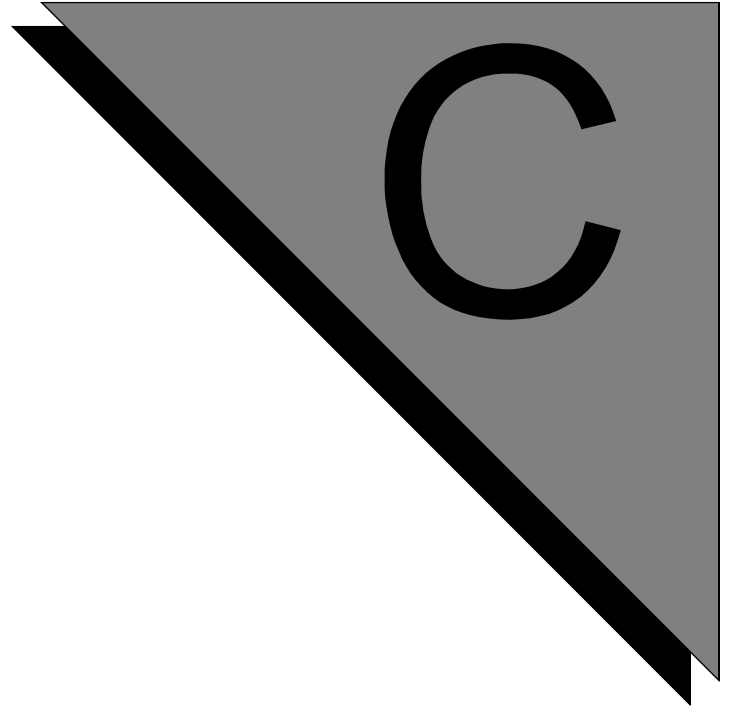
You will have to stop and then restart PERIsnmp so that it can recognize the new CPS (see [SNMP System Startup on page 18](#)).

**Question:**

Okay, you've answered all of those questions well. Now tell me, why do I need PERIsnmp if I already have PeriView?

**Answer:**

PeriView communicates directly with the MPS network for monitoring and control through proprietary protocol. This means that workstations which are not configured to run PeriView cannot normally communicate with the MPS network and components. SNMP acts as a “translator” between management stations that are not configured for PeriView, and the MPS network, both from a command line aspect as well as through third-party tools that are SNMP compliant (see [\*SNMP and MPS Integration on page 10\*](#)). Okay?!



# Request For Comment (RFC)

This chapter covers:

1. Using the Service
2. Tables of RFCs

### Using the Service

Additional useful information about SNMP, the MIB, and related topics may be had by obtaining RFCs. An RFC is a report or paper on specific topics put together by authoritative sources within the subject community. Though not specific to the Nortel Networks, Periphonics Enterprise MIB itself, the information provided in the RFCs can be used to better understand the integration and relevance of SNMP in the Nortel Networks product.

RFC-INFO is a mail-based server for listing and retrieving RFCs. An RFC may be obtained by sending e-mail to: RFC-INFO@ISI.EDU. The message contents should be formulated as follows:

Retrieve: RFC  
Doc-Id: RFCxxxx

where xxxx is the RFC number being requested. Note that in your mail, the subject line is ignored, and the message text is case independent.

For help with this service, send e-mail to the above address with the body of your message containing the following:

HELP: HELP

This will provide you with information on using the service and other topics that may aid you in your endeavor. Two specific help topics that may be of use are:

HELP: RETRIEVE (Explains how to use the retrieve request)  
HELP: TOPICS (Lists help topics)

To receive a list of current RFCs, your message should contain the following entry:

LIST: RFC  
Keywords: somestring

where "somestring" is a title or key component word you wish to search for. For a list of keywords currently in use, send a message containing the entry:

LIST: KEYWORDS

### Tables of RFCs

The following table lists topic RFCs that will be of relevance with regard to SNMPv1.



<b>RFC Number</b>	<b>RFC Title</b>
RFC 1155	Structure of Management Information
RFC 1156	Management Information Base for Network Management of TCP/IP-Based Internets
RFC 1157	SNMP V1 Protocol
RFC 1212	Concise MIB Definitions
RFC 1213	Management Information Base II
RFC 1215	A Convention for Defining Traps for use with the SNMP

The following table lists topic RFCs that will be of relevance with regard to SNMPv2c.

<b>RFC Number</b>	<b>RFC Title</b>
RFC 1901	Community Based SNMPv2c
RFC 1902	MIB for SNMPv2c
RFC 1903	Textual Conventions for SNMPv2c
RFC 1904	Conformance Statements for SNMPv2c
RFC 1905	Protocol Operations for SNMPv2c
RFC 1906	Transport Mappings for SNMPv2c
RFC 1907	MIB for SNMPv2c
RFC 1908	Coexistence Between SNMPv1 and SNMPv2c

**This page has been intentionally left blank.**

# Index

### A

- Abstract Syntax Notation One [3](#)
- access control [21](#)
- access privileges [28](#)
- access rights [28–29](#)
  - and groups [28](#)
  - default values [28, 29](#)
  - fields [28](#)
  - format [28](#)
  - tag [28](#)
- agent [2, 38](#)
  - configuring notifications [34](#)
  - interaction with manager [7](#)
  - master [25](#)
    - configuration files, location of [23](#)
    - environment variables [23](#)
    - log file, location of [23](#)
    - snmpdm [18](#)
    - starting/stopping [18, 25](#)
  - master workstation [34](#)
  - starting [14, 18](#)
  - stopping [18](#)
  - validating SNMP requests to [30](#)
  - verifying response [17](#)
  - See also* management agent
- alarm** command [50](#)
- alarmd**
  - and the MPS (VPS/is) components subagent [52](#)
  - process and daemon [50, 52](#)
- alarms
  - defining [52](#)
  - in Nortel Networks, Periphonics MIB [52](#)
  - See also* Nortel Networks, Periphonics Enterprise MIB: variables
- alarmLogCode [52](#)
- alarmLogComponentId [45, 53](#)
- alarmLogComponentIpAddress [53](#)
- alarmLogComponentType [52](#)
- alarmLogHostId [52](#)
- alarmLogIdx [52](#)
- alarmLogLineId [52](#)
- alarmLogMax [52](#)
- alarmLogMessage [52](#)
- alarmLogNumber [52](#)
- alarmLogProcName [52](#)
- alarmLogSeverity [52](#)
- alarmLogTable [52](#)
- alarmLogTime [52](#)
- alarmLogVruId [45](#)

- applications
  - defining parameters of [57](#)
- applications statistics
  - defining parameters of [57](#)
- appLineCnt [57](#)
- appName [57](#)
- appOptions [57](#)
- appStatsLastChange [57](#)
- appStatsName [57](#)
- appStatsTable [57](#)
- appStatsValue [57](#)
- appStatsVruId [57](#)
- appTable [57](#)
- appVruId [57](#)
- architecture
  - client/server [2](#)
  - MPS (VPS/is) environment [43](#)
  - MPS (VPS/is) environment illustration [43](#)
- ASN.1
  - See also* Abstract Syntax Notation One
- assigning principles to groups [32–33](#)
- audience, intended [ii](#)
- automatically configuring trap destinations [15](#)
- automatically defining trap destinations [15–16, 25, 34](#)

### C

- client [2](#)
- client/server architecture [2](#)
- command line
  - options [21, 23](#)
  - utilities [21, 23, 39](#)
    - default community string [23](#)
    - default SNMP version [23](#)
    - manager configuration files [23](#)
    - used with MIB [46](#)
    - used with SNMP [39, 42](#)
- community strings [21, 30–31](#)
  - and groups [32](#)
  - as principles [32](#)
  - default values [30, 31](#)
  - fields [30](#)
  - format [30](#)
  - tag [30](#)
- component types, defining [60](#)
- componentEventsGroup [45](#)
- componentEventsGroup2 [45](#)
- componentId [60](#)
- componentIpAddress [60](#)

- componentNotificationsGroup 45
- componentNotificationsGroup2 45
- components, defining 60
- components, understanding 12
- componentState 60
- componentStateLastChange 60
- componentTable 60
- componentType 60
- componentTypeId 60
- componentTypeName 60
- componentTypeTable 60
- configuration files 21
  - snmpd.cnf 9, 21
- conventions
  - manual iv

## D

- data types
  - arrays of scalar, two dimensional 5
  - scalar 5
- database 2
- defining component types 60
- defining components 60
- defining notification target entries 34–35
- defining notifications 34
- defining target parameters 36
- deprecated objects (and their replacements) 45

## E

- enterprise, Nortel Networks 20
- enterprises, private
  - See also MIB
- environment variables 22–24
  - data file in 21
  - editing 22
  - manager related 23
  - master/subagent related 23
- executables 38

## G

- getbulk** 39
- get-bulk-request 6, 8, 28
  - See also SNMP messages
- getmany** 39
- getmany command example 17
- getmet** 39
- getnext** 39

- get-next-request 6, 7, 28
  - See also SNMP messages
- getone** 39
- get-request 6, 7, 28
  - See also SNMP messages
- get-response 6, 8
  - See also SNMP messages
- getroute** 39
- getsub** 39
- gettab** 39
- groups
  - and access rights 28
  - and community strings 32
  - and principles 32

## H

- host
  - defining parameters of 58
- hostAdminState 58
- hostId 58
- hostLuCnt 58
- hostMedia 58
- hostProtocol 58
- hostState 58
- hostStateLastChange 58
- hostTable 58
- hostVruId 58
- how to
  - use SNMP with the MPS (VPS/is) system 43
- hypertext links
  - types of ii
  - use of ii
  - See also manual: hypertext links in, using on line

## I

- inform-request 6, 9, 28, 34
  - See also SNMP messages
- installing the PERIsnmp package 14
- interaction, manager and agent 7–9
- IP address 35, 51

## L

- lineAppAdminState 56
- lineAppName 56
- lineAppOverlay 55
- lineAppState 55
- lineAppStateCfgChange 55

- lineAppStateLastChange [55](#)
- lineId [55](#)
- lineProtocol [55](#)
- lines
  - defining parameters of [55](#)
- lineState [55](#)
- lineTable [55](#)
- lineType [55](#)
- lineVruId [55](#)
- linked applications [55](#)
- logical unit
  - See also* LU
- LU
  - defining parameters of [59](#)
- luHostId [59](#)
- luId [59](#)
- luPoolName [59](#)
- luState [59](#)
- luStateLastChange [59](#)
- luTable [59](#)
- luVruId [59](#)

## M

- management agent [2, 3](#)
- Management Information Base
  - See also* MIB
- management station [2, 3, 34](#)
- manager [2](#)
  - configuration files, location of [23](#)
  - interaction with agent [7](#)
  - IP address of [34](#)
  - See also* management station
- manual
  - how to use [ii, iii](#)
  - hypertext links in [ii](#)
  - intended audience for [ii](#)
  - organization of [iii](#)
  - scope of [ii](#)
  - using on line [ii](#)
- master agent [18](#)
  - See also* agent, master
- messages, SNMP [6](#)
- mgr.cnf [23](#)
- MIB
  - as database [2, 3](#)
  - as defined by SNMP [3](#)
  - definition of [2, 3](#)
  - general use of [3](#)
  - private enterprises

- use of [4](#)
  - tree illustration [4](#)
  - views [21](#)
    - See also* view subtrees
- mib2agt [18, 38](#)
  - command line arguments for [23](#)
- MIB2AGT\_CMDLINEARGS [23](#)
- MIB-II
  - and SNMP [3](#)
  - default values [21, 25](#)
  - definition of [3](#)
  - groups [3](#)
  - OID translations [21](#)
  - protocols [3](#)
  - subtree [3](#)
  - system tree illustration [7](#)
  - tree illustration [4](#)
- MPS (VPS/is) environment
  - defining parameters of [54](#)
  - SNMP system architecture [43](#)
- msnsaagt.exe [38](#)
  - entering start up parameters for [24](#)

## N

- network elements [3](#)
- network management components
  - management agent [2](#)
  - management station [2](#)
  - MIB [2](#)
  - network management protocol [2](#)
- node names [5](#)
- Nortel Networks enterprise
  - SMIv1 format [20](#)
  - SMIv2 format [20](#)
- Nortel Networks Enterprise, Periphonics MIB
  - variables
    - spanVruId [56](#)
- Nortel Networks MPS/VPS Resources Daemon [24](#)
- Nortel Networks, Periphonics Enterprise MIB [10](#)
  - components monitored by [10](#)
  - control and administration [42–49](#)
  - data files [21](#)
  - events monitored by [10](#)
  - monitoring and control [51–60](#)
  - objects
    - almLogTable [52](#)
    - appStatsTable [57](#)
    - appTable [57](#)
    - componentEventsGroup [45](#)

- componentEventsGroup2 45
- componentNotificationsGroup 45
- componentNotificationsGroup2 45
- componentTable 60
- deprecated and replacements 45
- hostTable 58
- lineTable 55
- luTable 59
- notifyAlarm 45
- notifyAlarmByComponent 45
- notifyAlarmByComponentEx 45, 50, 51
- notifyComponentStateChg 45, 51
- notifyHostStateChg 51
- notifyLineAppStateChg 51
- notifyLuStateChg 51
- notifySpanStateChg 51
- notifyTopologyChg 51
- notifyVruStateChg 45
- spanTable 56
- status information 45
- vruEventsGroup 45
- vruNetworkMIBCompliance 45
- vruNetworkMIBComplianceByComponent 45
- vruNetworkMIBComplianceByComponent2 45
- vruNotificationsGroup 45
- vruTable 54
- OID translations 21
- setting a value 49
- SMIv1 format 20
- SMIv2 format 20
- subtrees 51–60
  - Applications 57
  - Applications Stats 57
  - Components 60
  - Events Group Notifications 51
  - Host 58
  - Line 55–56
  - LU 59
  - Span Table 56
  - VRU 54
  - VRU Events Alarms 52
- tables in 46–47
- variables
  - access information 44
  - alarmLogCode 52
  - alarmLogComponentId 45, 53
  - alarmLogComponentIpAddress 53
  - alarmLogComponentType 52
  - alarmLogHostId 52
  - alarmLogIdx 52
  - alarmLogLineId 52
  - alarmLogMax 52
  - alarmLogMessage 52
  - alarmLogNumber 52
  - alarmLogProcName 52
  - alarmLogSeverity 52
  - alarmLogTime 52
  - alarmLogVruId 45
  - appLineCnt 57
  - appName 57
  - appOptions 57
  - appStatsLastChange 57
  - appStatsName 57
  - appStatsValue 57
  - appStatsVruId 57
  - appVruId 57
  - componentId 60
  - componentIpAddress 60
  - componentState 60
  - componentStateLastChange 60
  - componentType 60
  - componentTypeId 60
  - componentTypeName 60
  - componentTypeTable 60
  - hostAdminState 58
  - hostId 58
  - hostLuCnt 58
  - hostMedia 58
  - hostProtocol 58
  - hostState 58
  - hostStateLastChange 58
  - hostVruId 58
  - lineAppAdminState 56
  - lineAppCfgLastChange 55
  - lineAppName 56
  - lineAppOverlay 55
  - lineAppState 55
  - lineAppStateLastChange 55
  - lineId 55
  - lineProtocol 55
  - lineState 55
  - lineType 55
  - lineVruId 55
  - luHostId 59
  - luId 59
  - luPoolName 59
  - luState 59

- luStateLastChange [59](#)
- luVruId [59](#)
- OIDs [51–60](#)
- querying directly [46](#)
- querying in tables [46](#)
- spanEnabled [56](#)
- spanId [56](#)
- spanLineIdStart [56](#)
- spanLineNumber [56](#)
- spanState [56](#)
- spanStateLastChange [56](#)
- vruAdminState [54](#)
- vruHostCnt [54](#)
- vruId [54](#)
- vruIpAddress [54](#)
- vruLineCnt [54](#)
- vruSpanCnt [54](#)
- vruState [54](#)
- vruStateLastChange [54](#)

*See also* MIB

notification targets

- default values [34, 35](#)

- defining [34](#)

- IP address [35](#)

notifications

- and target parameters [36](#)

- default values [34](#)

- defining [34](#)

- fields [34](#)

- format [34](#)

- tag [34](#)

- targets

  - fields [35](#)

  - format [35](#)

  - tag [35](#)

*See also* target parameters

notifyAlarm [45](#)

notifyAlarmByComponent [45](#)

notifyAlarmByComponentEx [45, 50–52](#)

notifyComponentStateChg [45, 51](#)

notifyHostStateChg [51](#)

notifyLineAppStateChg [51](#)

notifyLuStateChg [51](#)

notifySpanStateChg [51](#)

notifyTopologyChg [51](#)

notifyVruStateChg [45](#)

## O

Object Identifier

*See also* OID

object instances [26](#)

objects

- deprecated and replacements [45](#)

- status information [45](#)

*See also* variables, traps, alarms

OID [5](#)

- for Nortel Networks, Periphonics MIB variables [51–60](#)

- translation information [21](#)

- view subtrees [26](#)

## P

Periphonics Enterprise MIB

- detailed component structure [66–98](#)

- numerical OID [62](#)

- textual OID [62](#)

- tree structure [62–66](#)

Periphonicsv1.mib [20](#)

Periphonicsv2.mib [20](#)

PeriView

- versus SNMP [11](#)

- workstation [2](#)

principle/group assignments [32](#)

principle/group field format [32](#)

principles

- default values [33](#)

- fields [32](#)

- format [32](#)

- tag [32](#)

principles of components [12](#)

private enterprises

- OID

  - numerical [5](#)

  - textual [5](#)

*See also* MIB

protocol

- host [58](#)

- line [55](#)

- network management [2](#)

- SNMPv1 [6](#)

- SNMPv2c [6](#)

- used in MIB-II [3](#)

*See also* SNMP

## R

rebooting workstation [14](#)

Request For Comment



- SNMPv1 related [111](#)
  - SNMPv2c related [111](#)
  - using the service [110](#)
  - RFC, *See also* Request For Comment
  - rights, access [28](#)
  - rights, security access [26](#)
- ## S
- S19snmp.startup** [18](#)
  - scope [ii](#)
  - security access rights [26–27](#)
  - server [2](#)
  - setany** [39](#)
  - set-request [6, 8, 28](#)
    - See also* SNMP messages
  - setting a value [49](#)
  - Simple Network Management Protocol
    - See also* SNMP
  - SMI
    - application types [5](#)
    - definition of [5](#)
    - universal types [5](#)
  - SMIv1 [20](#)
  - SMIv2 [20](#)
  - SNMP [2–9](#)
    - and MPS (VPS/is) integration [10](#)
    - and the MPS (VPS/is) environment [2](#)
    - andMPS (VPS/is) integration [43](#)
    - as MPS (VPS/is) network translator [11](#)
    - command line options [23](#)
    - configuration files [21](#)
    - configuring system [21–36](#)
    - default community string [23](#)
    - default version [23](#)
    - installation of [14](#)
    - messages
      - get-bulk-request [6, 8](#)
      - get-next-request [6, 7](#)
      - get-request [6, 7](#)
      - get-response [6, 8](#)
      - inform-request [6, 9](#)
      - set-request [6, 8](#)
      - trap [6, 9](#)
      - using [7–9, 42](#)
    - network management concepts [2](#)
      - See also* network management components
    - overview [2](#)
    - purpose of [2](#)
    - requests [30](#)
      - system startup [18](#)
      - versus PeriView [11](#)
  - SNMP EMANATE Adapter for Windows 2000 [24](#)
  - SNMP EMANATE Master Agent [24](#)
  - snmp.sh [18, 21](#)
  - snmpd.cnf [9, 21, 23](#)
    - configuring [25–36](#)
    - See also* configuration files
  - snmpdm [18, 23, 25, 34, 38](#)
    - command line arguments for [23](#)
  - snmpdm.exe [25, 38](#)
    - entering start up parameters for [24](#)
  - SNMPDM\_CMDLINEARGS [23](#)
  - snmpinfo.dat [21, 23](#)
  - spanEnabled [56](#)
  - spanId [56](#)
  - spanLineIdStart [56](#)
  - spanLineNumber [56](#)
  - spans
    - defining parameters of [56](#)
  - spanState [56](#)
  - spanStateLastChange [56](#)
  - spanTable [56](#)
  - spanVruId [56](#)
  - SR\_AGT\_CONF\_DIR [23](#)
  - SR\_LOG\_DIR [23](#)
  - SR\_MGR\_CONF\_DIR [23](#)
  - SR\_UTIL\_COMMUNITY [23](#)
  - SR\_UTIL\_SNMP\_VERSION [23](#)
  - SRP
    - starting and stopping [50](#)
  - starting agent [14](#)
  - starting and stopping SRP [50](#)
  - Structure of Management Information
    - See also* SMI
  - subagent [38](#)
    - command line option changes [18](#)
    - environment variables [23](#)
    - MIB-II (mib2agt) [18](#)
    - MPS (VPS/is) resources (vrsmnpd) [18](#)
    - starting/stopping [18, 52](#)
  - subtree
    - Applications [57](#)
    - Applications Stats [57](#)
    - Components [60](#)
      - component table [60](#)
      - component type table [60](#)
    - enterprises
      - definition of [3](#)

- Host [58](#)
- Line [55–56](#)
- LU [59](#)
- MIB-II [3](#)
- MIB-II illustration [4](#)
- MIB-II system illustration [7](#)
- naming [26](#)
- Nortel Networks, Periphonics Enterprise MIB  
[51–60](#)
- Span Table [56](#)
- views [28](#)
- VRU [54](#)
- VRU Events Alarms [52](#)
- sysContact [25](#)
- sysDescr [25](#)
- sysLocation [25](#)
- sysName [25](#)

## T

### tables

- alarm log [52](#)
- application statistics [57](#)
- applications [57](#)
- component [60](#)
- component type [60](#)
- host [58](#)
- indexing in [46](#)
- line [55](#)
- LU [59](#)
- referencing in [46](#)
- span [56](#)
- vrु [54](#)

### target parameters

- and notifications [36](#)
- default values [36](#)
- defining [36](#)
- fields [36](#)
- format [36](#)
- tag [36](#)

### transports [21](#)

### trap destinations

- configuring [25](#)
- configuring automatically [15](#)
- defining automatically [15, 16, 25, 34](#)

### trap messages [28](#)

### **trapcfg.pl** [16](#)

trapcfg.pl [9, 15, 16, 25, 34](#)

### **traprcv** [39, 50, 107](#)

### traps [6, 9, 21](#)

- and view subtrees [28](#)
- as notifications [34](#)
- defining [51](#)
- in Periphonics Enterprise MIB [51](#)
- multiple configuration example [50, 107](#)
- receiving [49](#)
- using **traprcv** to view [50, 107](#)
- viewing [49](#)
- See also* SNMP messages
- See also* Periphonics Enterprise MIB:objects

## U

- understanding components [12](#)
- using MIB variables [51–60](#)
- using SNMP messages [7–9](#)
- using SNMP with the MIB [6](#)

## V

### variables

- access information [44](#)
- as used in MIB [3, 51–60](#)
  - See also* Nortel Networks, Periphonics Enterprise MIB
- environment [21, 22](#)
  - See also* environment variables
- identifying [5](#)
- OIDs in Nortel Networks, Periphonics MIB [51–60](#)
- querying directly [46](#)
- referencing to [5](#)
- setting a value [49](#)
- structure of [5](#)
- table querying [46](#)

### view subtrees

- adding or deleting [26](#)
- and traps [28](#)
- default values [26, 27](#)
- fields [26](#)
- format [26](#)
- tag [26](#)

### views

- subtree [28](#)

vrnsnmpd [18, 38](#)

- command line arguments for [23](#)

vrnsnmpd.exe [38](#)

- entering start up parameters for [24](#)

VRNSMPD\_CMDLINEARGS [23](#)

### VRU

*Same as MPS*

VRU Network [20](#), [54](#)

VRU\_Networkv1.mib [20](#)

VRU\_Networkv2.mib [20](#)

vruAdminState [54](#)

vruEventsGroup [45](#)

vruHostCnt [54](#)

vruId [54](#)

vruIpAddress [54](#)

vruLineCnt [54](#)

vruNetworkMIBCompliance [45](#)

vruNetworkMIBComplianceByComponent  
[45](#)

vruNetworkMIBComplianceByComponent2  
[45](#)

vruNotificationsGroup [45](#)

vruSpanCnt [54](#)

vruState [54](#)

vruStateLastChange [54](#)

vruTable [47](#), [54](#)

VT

*See also* LU

## W

workstation [2](#)

    master agent [34](#)

    rebooting [14](#)

**This page has been intentionally left blank.**