

# **TNMS**

## **18.10**

### **SNMP NBI Operation Guide (SNOG)**

**Issue: 3    Issue date: October 2020**



Infinera is continually striving to reduce the adverse environmental effects of its products and services. We would like to encourage you as our customers and users to join us in working towards a cleaner, safer environment. Please recycle product packaging and follow the recommendations for power use and proper disposal of our products and their components.

The information in this document is subject to change without notice and describes only the product defined in the introduction of this documentation. This documentation is intended for the use of Infinera customers only for the purposes of the agreement under which the document is submitted, and no part of it may be used, reproduced, modified or transmitted in any form or means without the prior written permission of Infinera. The documentation has been prepared to be used by professional and properly trained personnel, and the customer assumes full responsibility when using it. Infinera welcomes customer comments as part of the process of continuous development and improvement of the documentation.

The information or statements given in this documentation concerning the suitability, capacity, or performance of the mentioned hardware or software products are given "as is" and all liability arising in connection with such hardware or software products shall be defined conclusively and finally in a separate agreement between Infinera and the customer. However, Infinera has made all reasonable efforts to ensure that the instructions contained in the document are adequate and free of material errors and omissions. Infinera will, if deemed necessary by Infinera, explain issues which may not be covered by the document. Infinera will correct errors in this documentation as soon as possible.

IN NO EVENT WILL INFINERA BE LIABLE FOR ERRORS IN THIS DOCUMENTATION OR FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA, THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT.

This documentation and the product it describes are considered protected by copyrights and other intellectual property rights according to the applicable laws. Other product names mentioned in this document may be trademarks of their respective owners, and they are mentioned for identification purposes only.

Copyright © Infinera, 2020. All rights reserved.

## Table of Contents

<b>1</b>	<b>Preface .....</b>	<b>10</b>
1.1	Intended Audience .....	10
<b>2</b>	<b>Introduction .....</b>	<b>11</b>
2.1	General Description.....	11
2.2	SNMP Protocol Support .....	12
2.3	Terminology .....	12
2.4	Differences to TNMS Core SNMP Proxy .....	13
2.4.1	General Changes .....	13
2.4.2	Tables and fields .....	13
2.4.3	Notification behaviors .....	14
2.4.4	Protocol support changes .....	15
2.5	Installation and Licensing .....	15
2.6	MIB File Location .....	15
2.7	Agent Engine ID .....	15
<b>3</b>	<b>SNMP Agent Configuration .....</b>	<b>17</b>
3.1	System Settings .....	17
3.1.1	Base System Settings .....	17
3.1.2	MIB-II Variables.....	18
3.2	SNMP User Configuration .....	19
3.2.1	SNMP User Identification.....	19
3.2.2	SNMP user access permissions .....	20
3.2.3	User trap destinations.....	21
3.2.4	User inform destinations .....	22
3.2.5	Notification Filtering .....	23
3.3	Heartbeat Notifications .....	25
3.4	Exporting SNMP Agent Configuration.....	25
<b>4</b>	<b>SNMP NBI MIB – General Description .....</b>	<b>26</b>
4.1	Exported Object Model.....	26
4.2	Hierarchical View .....	27
4.3	Notification Model.....	29

4.4	Resynchronization After Network or Manager Errors.....	30
4.5	Recommended Table Retrieval Approaches .....	31
4.5.1	Retrieving all rows of a table.....	31
4.5.2	Retrieving specific rows by index.....	32
4.5.3	Retrieving blocks of rows for a sub-index .....	32
4.5.4	Limited size of responses .....	33
4.5.5	Request timeouts .....	33
4.6	Data Types.....	34
4.7	Strings and Multi-Language Support .....	41
<b>5</b>	<b>Retrieving Network Inventory.....</b>	<b>42</b>
5.1	Network Elements .....	42
5.1.1	List of NEs (enmsNETable) .....	42
5.1.2	Notification of NE object creation (enmsNEObjectCreationTrap) .....	43
5.1.3	Notification of NE object deletion (enmsNEObjectDeletionTrap).....	44
5.1.4	Notification of NE state change (enmsNEStateChangeTrap).....	44
5.1.5	Notification of NE attribute value change (enmsNEAttributeChangeTrap) .....	45
5.2	Modules .....	46
5.2.1	List of Modules (enmsModuleTable).....	46
5.2.2	Notification of Module object creation (enmsModuleObjectCreationTrap).....	47
5.2.3	Notification of Module object deletion (enmsModuleObjectDeletionTrap) .....	47
5.2.4	Notification of Module state change (enmsModuleStateChangeTrap) .....	48
5.2.5	Notification of Module attribute value change (enmsModuleAttributeChangeTrap).....	48
5.3	Ports .....	49
5.3.1	List of Ports (enmsPortTable) .....	49
5.3.2	Notification of Port object creation (enmsPortObjectCreationTrap) ..	50
5.3.3	Notification of Port object deletion (enmsPortObjectDeletionTrap)...	51
5.3.4	Notification of Port state change (enmsPortStateChangeTrap).....	51
5.3.5	Notification of Port attribute value change (enmsPortAttributeChangeTrap).....	52
5.4	Termination Points .....	52

5.4.1	List of TPs (enmsTPTable) .....	52
5.4.2	Notification of TP object creation (enmsTPObjectCreationTrap) .....	54
5.4.3	Notification of TP object deletion (enmsTPObjectDeletionTrap).....	55
5.4.4	Notification of TP state change (enmsTPStateChangeTrap).....	55
5.4.5	Notification of TP attribute value change (enmsTPAttributeChangeTrap).....	56
5.5	Port Connections.....	57
5.5.1	List of Port Connections (enmsPortConnTable).....	57
5.5.2	Notification of Port Connection object creation (enmsPortConnObjectCreationTrap) .....	58
5.5.3	Notification of Port Connection object deletion (enmsPortConnObjectDeletionTrap).....	58
5.5.4	Notification of Port Connection attribute value change (enmsPortConnAttributeChangeTrap) .....	58
5.6	Equipment Holders.....	59
5.6.1	List of Equipment Holders (enmsEquipHolderTable) .....	59
<b>6</b>	<b>Retrieving Paths, Services and Cross-connections .....</b>	<b>60</b>
6.1	Sub-Network Connections (Optical Paths) .....	60
6.1.1	List of SNCs (enmsSNCTable) .....	60
6.1.2	Notification of SNC object creation (enmsSNObjectCreationTrap) 63	
6.1.3	Notification of SNC object deletion (enmsSNObjectDeletionTrap). 63	
6.1.4	Notification of SNC state change (enmsSNStateChangeTrap).....	63
6.1.5	Notification of SNC attribute value change (enmsSNCAAttributeChangeTrap).....	64
6.1.6	Association between SNCs and their client SNCs (enmsSNCCClientSNCTable) .....	65
6.1.7	Receiving protection switch notifications.....	65
6.2	Ethernet Paths .....	66
6.2.1	List of Ethernet Paths (enmsEthernetPathTable).....	66
6.2.2	Notification of Ethernet Path object creation (enmsEthernetPathObjectCreationTrap).....	67
6.2.3	Notification of Ethernet Path object deletion (enmsEthernetPathObjectDeletionTrap) .....	67
6.2.4	Notification of Ethernet Path state change (enmsEthernetPathStateChangeTrap).....	68

6.2.5	Notification of Ethernet Path attribute value change (enmsEthernetPathAttributeChangeTrap).....	68
6.3	Services .....	69
6.3.1	List of Services (enmsServiceTable) .....	69
6.3.2	Notification of Service object creation (enmsServiceObjectCreationTrap) .....	70
6.3.3	Notification of Service object deletion (enmsServiceObjectDeletionTrap).....	70
6.3.4	Notification of Service state change (enmsServiceStateChangeTrap) .....	70
6.3.5	Notification of Service attribute value change (enmsServiceAttributeChangeTrap) .....	71
6.4	Cross Connections.....	72
6.4.1	List of Cross Connections (enmsCCTable).....	72
<b>7</b>	<b>Retrieving and Receiving Alarms .....</b>	<b>74</b>
7.1	Approaches for Monitoring Alarms .....	74
7.1.1	Listening to alarm notifications .....	74
7.1.2	Polling the alarm tables .....	75
7.2	Alarm Tables .....	76
7.2.1	List of all active alarms (enmsAlarmTable) .....	76
7.2.2	List of alarms for NEs (enmsAlarmsForNETable) .....	78
7.2.3	List of alarms for Ports (enmsAlarmsForPortTable) .....	80
7.2.4	List of alarms for TP (enmsAlarmsForTPTable).....	81
7.2.5	List of alarms for Port Connections (enmsAlarmsForPortConnTable) .....	83
7.2.6	List of alarms for Modules (enmsAlarmsForModuleTable).....	85
7.2.7	List of alarms for SNCs (enmsAlarmsForSNCTable) .....	86
7.2.8	List of EMS alarms (enmsAlarmsForEMSTable).....	88
7.3	Alarm Notifications .....	89
7.3.1	Notification of NE alarm (enmsNEAlarmTrap).....	89
7.3.2	Notification of Module alarm (enmsModuleAlarmTrap) .....	90
7.3.3	Notification of Port alarm (enmsPortAlarmTrap) .....	91
7.3.4	Notification of TP alarm (enmsTPAlarmTrap) .....	93
7.3.5	Notification of EMS alarm (enmsEMSAlarmTrap) .....	94
7.3.6	Notifications on alarm acknowledgement.....	95

<b>8</b>	<b>Reading and Setting Agent Parameters via SNMP .....</b>	<b>96</b>
8.1	Agent Parameters (EnmsControl).....	96
8.2	SNMP Agent Notifications .....	97
8.2.1	Notification of agent state change (enmsProxyStateChangeTrap)...	97
8.3	Notification Filtering (enmsTrapFilter).....	97
8.4	Notification History (enmsTrapHistoryTable) .....	99
<b>9</b>	<b>Ethernet Paths – Support of MEF 40.....</b>	<b>100</b>
9.1	MEF-UNI-EVC-MIB .....	100
9.1.1	mefServiceEvcCfgTable .....	100
9.1.2	mefServiceEvcStatusTable.....	101
<b>10</b>	<b>Performance Monitoring.....</b>	<b>102</b>
10.1	PM Requests.....	103
10.1.1	PM request table (enmsPerfMonRequestTable) .....	103
10.2	Creating a PM Request .....	105
10.3	PM Request States .....	107
10.4	PM Request State Change Notifications.....	108
10.5	Actions on PM Requests .....	109
10.5.1	Executing a PM request .....	109
10.5.2	Updating PM request attributes .....	110
10.5.3	Cancelling a PM request .....	110
10.5.4	Discarding PM data associated to a PM request .....	111
10.5.5	Deleting a PM request .....	111
10.5.6	Error exceptions .....	112
10.6	Retrieving PM Data .....	112
10.6.1	PM data retention period .....	113
10.6.2	enmsPerfMonResultPmpTable.....	113
10.6.3	enmsPerfMonResultValueTable .....	115
10.6.4	enmsPerfMonResultThresholdTable .....	115
<b>11</b>	<b>Optical Power Monitoring.....</b>	<b>117</b>
11.1	OPM Requests.....	118
11.1.1	OPM request table (enmsOptPowerMonRequestTable) .....	118
11.2	Creating an OPM Request .....	119

11.3	OPM Request States.....	121
11.4	OPM Request State Change Notifications.....	122
11.5	Actions on OPM Requests .....	123
11.5.1	Executing an OPM request.....	123
11.5.2	Updating OPM request attributes.....	124
11.5.3	Cancelling an OPM request.....	124
11.5.4	Discarding OPM data associated to an OPM request .....	125
11.5.5	Deleting an OPM request .....	125
11.5.6	Error exceptions .....	126
11.6	Retrieving OPM Data .....	126
11.6.1	OPM data retention period.....	127
11.6.2	enmsOptPowerMonResultTable .....	127
<b>12</b>	<b>Net-SNMP Examples for PM Data Retrieval.....</b>	<b>128</b>
12.1	Creating a New PM Request .....	129
12.2	Executing the PM Request .....	130
12.3	Retrieving the PM Request Results.....	131
12.4	Re-executing the PM Request.....	132
12.5	Deleting a PM Request .....	133
<b>13</b>	<b>Troubleshooting.....</b>	<b>134</b>
	<b>Abbreviations.....</b>	<b>138</b>



**History of changes**

Issue	Issue date	Remarks
1	December 2019	First version for 18.10
2	February 2020	Second version for 18.10 Added new table <i>enmsSNCClientSNCTable</i> .
3	October 2020	Third version for 18.10 Added instructions on how to enable/disable alarm acknowledge notifications. Added instructions on how to force maximum message size.

# 1

## Preface

TNMS (Telecommunications Network Management System) is an Infinera standalone application that manages the physical structure of the transport network as well as the logical end-to-end connections and services.

SNMP NBI is a northbound interface that allows clients to access TNMS using the SNMP protocol.

The SNMP NBI Operation Guide describes how to operate SNMP NBI and integrate it with external applications.

### 1.1 Intended Audience

This document is intended for people responsible for:

- Installing, configuring, maintaining and troubleshooting SNMP NBI (Northbound Interface).
- Integrating TNMS with external applications, such as umbrella systems, using the SNMP protocol.

It is assumed that the reader is familiar with basic SNMP protocol concepts.

# 2

## Introduction

### 2.1 General Description

In SNMP terminology, SNMP NBI is an SNMP agent, and the clients, such as umbrella systems or fault monitoring systems, are the SNMP managers.

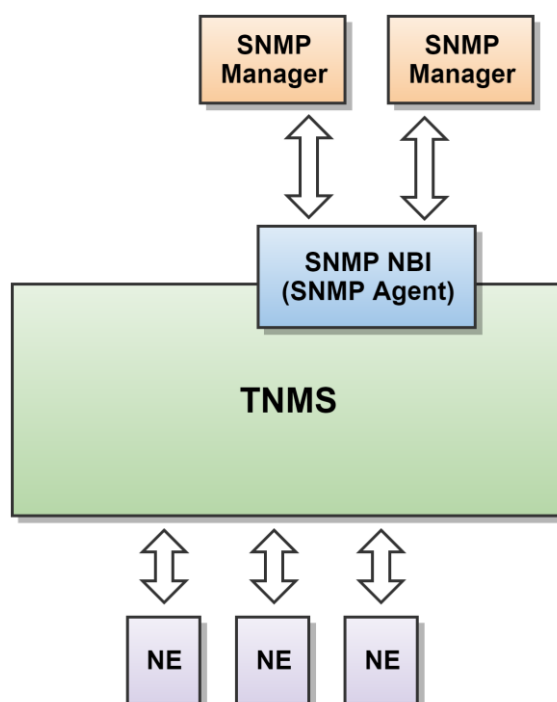


Figure 1 SNMP NBI component in a TNMS system

SNMP NBI provides the following functionalities:

- Network discovery and synchronization:
  - Lists of NEs, Modules, Ports, Termination Points (TPs), Port Connections (PCs) and Equipment Holders.
  - Notifications for network object creation (OC) and deletion (OD), attribute value changes (AVC) and state changes (SC).
- Fault management:
  - Lists of alarms, with filtering by type of affected object.
  - Alarm raise/clear notifications.

- Connection management
  - Lists of Cross-Connections (CCs), Sub-Network Connections (SNCs) and Ethernet Paths.
  - Notifications for object creation (OC) and deletion (OD), attribute value changes (AVC) and state changes (SC).
- Performance Monitoring:
  - Retrieval of history and current PM data.
- Optical Power Monitoring:
  - Retrieval of Optical Power Monitoring data.

SNMP NBI also provides preliminary MEF MIB support for the retrieval of Ethernet Paths (MEF 40).

## 2.2 SNMP Protocol Support

SNMP NBI provides the following protocol capabilities:

- SNMP v2c and v3
- UDP transport protocol
- GET / GETNEXT / GETBULK / SET requests
- TRAP and INFORM notifications
- IPv4 and IPv6
- SNMPv3 authentication protocols: MD5, SHA
- SNMPv3 privacy protocols: DES, 3DES, AES128

## 2.3 Terminology

The term “SNMP manager”, or simply “manager”, is used to refer any external system or application that accesses TNMS via SNMP.

The term “SNMP agent” refers to the SNMP NBI component itself.

The term “EMS” refers to the TNMS system itself.

The term “trap” is commonly used to refer an SNMP notification, regardless of it being sent as SNMP TRAP or SNMP INFORM.

Some objects are named differently in SNMP NBI MIB and TNMS. The table below shows the equivalences.

SNMP NBI MIB naming	TNMS naming
Module	Equipment
Port Connection (PC)	Physical Trail
Port	Physical Termination Point (PTP)
Sub-Network Connection (SNC)	Path
Ethernet Path	Ethernet Service

Table 1 Naming equivalences between SNMP NBI MIB and TNMS

## 2.4 Differences to TNMS Core SNMP Proxy

TNMS Core is the previous generation EMS, which also has an SNMP northbound interface called SNMP Proxy. For seamless transition of existing umbrella systems, TNMS SNMP NBI's MIB has been designed to be structurally similar to that of TNMS Core's SNMP Proxy. However, some differences do exist, as described below.

### 2.4.1 General Changes

- MIB name is "TNMS-NBI-MIB" instead of "TNMS-MIB".
- The Private Enterprise Number (PEN) is 'coriant' (42229) instead of 'sni' (231).
- Enumeration data types were revised and adapted to the TNMS information model.
- Human-readable timestamps have the fixed format yyyy-MM-dd HH:mm:ss, regardless of server regional settings.
- Strings are encoded in UTF-8 instead of ISO-8859-1.
- New MIB objects added associated to new features, such as heartbeat support, Ethernet Path support, Performance Monitoring and Optical Power Monitoring.
- Notification filtering is configured per SNMP user, not globally.

### 2.4.2 Tables and fields

The MIB objects below are present in the SNMP NBI MIB file, but will only be supported in the future.

Type of MIB objects	To be supported in the future
Tables and associated traps	<ul style="list-style-type: none"> <li>• <i>enmsSNCTPTable</i></li> <li>• <i>enmsSNCSNCTable</i></li> <li>• <i>enmsSNCCCTable</i></li> <li>• <i>enmsNeSNCTable</i></li> <li>• <i>enmsSubscriberTable</i></li> <li>• <i>enmsSubscriberTraps</i></li> <li>• <i>enmsAlarmsForServiceTable</i></li> </ul>
Table fields	<ul style="list-style-type: none"> <li>• <i>enmsNETable</i>: <i>enmsNeClass</i></li> <li>• <i>enmsTPTable</i>: <i>enmsTpUsageCountTX</i> <i>enmsTpUsageCountRX</i> <i>enmsTpUsageStateTX</i> <i>enmsTpUsageStateRX</i> <i>enmsTpBandwidthTX</i> <i>enmsTpBandwidthRX</i> <i>enmsTpTerminType</i> (only the AVC)</li> </ul>
Trap fields	<ul style="list-style-type: none"> <li>• <i>enmsSNCTraps</i>: <i>enmsSNCTPRelationshipChangeTrap</i></li> </ul>

Table 2 MIB objects not yet supported

### 2.4.3 Notification behaviors

Notification behaviors for Object Deletion events (OD) have been made consistent among all network object types. Avoiding redundant OD traps helps minimizing network and processing spikes when network changes occur.

TNMS Core SNMP Proxy	TNMS SNMP NBI
<p>When a NE is removed, an OD trap is sent for that NE only – no OD traps are sent for the child objects.</p> <p>When a module or port is removed, OD traps for its children are also sent.</p>	<p>When a NE, module or port is removed, an OD trap is sent for that object only – no OD traps are sent for its children. The manager may implicitly assume that the child objects have been removed also.</p> <p>Note: OD notifications for child objects are still sent if the removed parent object belongs to a UNO network element.</p>

Table 3 OD notification behavior changes

### 2.4.4 Protocol support changes

As for the SNMP protocol support, the differences to TNMS Core's SNMP Proxy are:

- The listening port is now configurable, to allow SNMP NBI to coexist with other SNMP services on the server.
- SNMP v1 is no longer supported, only v2c and v3.
- Timeout and maximum tries of INFORM notifications are configurable.

## 2.5 Installation and Licensing

SNMP NBI is an optional TNMS component. It must be installed during TNMS installation. It is not possible to install SNMP NBI later without reinstalling TNMS.

SNMP NBI requires a license to run. The license can be installed at any time, but only becomes effective after TNMS is restarted. To install it, go to the License Manager in the TNMS Client.

## 2.6 MIB File Location

SNMP NBI implements two MIBs:

- TNMS-NBI-MIB
- MEF-UNI-EVC-MIB (limited, preliminary support)

Files for both MIBs can be found in:

<TNMS Home>\Docs\SNMP-NBI



Other MIB files in the location above are included only because they define data types used by the MEF MIB. SNMP NBI does not implement any tables or objects of those MIBs.



MIB files are only available if the SNMP NBI component is installed.

## 2.7 Agent Engine ID

The SNMP NBI agent Engine ID is generated during TNMS startup based on the MIB Private Enterprise Number ("42229") and the IP address resolved

from the system host name. Consequently, the Engine ID will change if and only if the host IP address changes, that is, if the system host name resolves to a different IP address.



# 3 SNMP Agent Configuration

## 3.1 System Settings

### 3.1.1 Base System Settings

To configure the basic SNMP NBI system settings:

1. Open TNMS Client and go to **Administration > System Preferences > SNMP NBI**.

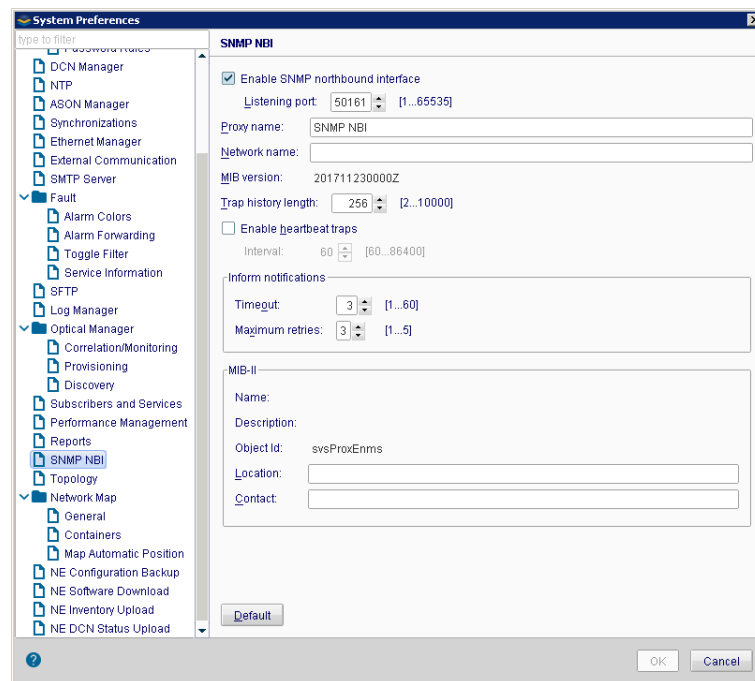


Figure 2 SNMP NBI system preferences

2. Check **Enable SNMP northbound interface**. If this option is disabled, all SNMP communication is blocked: no SNMP requests are accepted and no traps are sent.



If the checkbox is greyed out, an SNMP NBI license has been installed, but the server has not been restarted yet. Restart the server to enable SNMP NBI.

3. In **Listening port** specify an unused UDP port, where SNMP requests are to be received. The default is 50161.

Recommended: either 161, or a number between 49152 and 65535, which is the private port range defined by IANA. Make sure the chosen port is not in use by any other service or application on the server.

4. In **Proxy name** and **Network name**, optionally enter informative names for this SNMP NBI instance and for the network.
5. In **Trap history length**, specify the number of records to be kept in the trap history table. (See also 4.4 - Resynchronization After Network or Manager Errors.)
6. The **MIB version** is read-only and shows the version of the SNMP NBI MIB. It matches the LAST-UPDATE clause in the MIB definition file, and therefore can be used to check if an SNMP manager is using the correct file.
7. Check **Enable heartbeat traps** if heartbeat traps are to be sent to the listening SNMP managers.
  - In **Interval**, specify the number of seconds between heartbeats.
8. In the **Inform** group, configure how SNMP INFORM notifications are managed:
  - In **Timeout**, specify the maximum number of seconds that SNMP NBI will wait for a response before resending an Inform. Default is 3 seconds. Allowed values are between 1 and 60 seconds.
  - In **Maximum retries**, specify the maximum number of times that SNMP NBI will try to send an Inform notification to each destination. Default is 3 tries. Allowed values are between 1 and 5 tries.



The **Proxy name**, **Network name**, **Trap history length** and **MIB version** values above are readable via SNMP under the *enmsControl* MIB branch (see section 8.1).



The **Heartbeat** and **Inform** configuration parameters are also modifiable via SNMP under the *enmsControl* MIB branch (see section 8.1).

### 3.1.2 MIB-II Variables

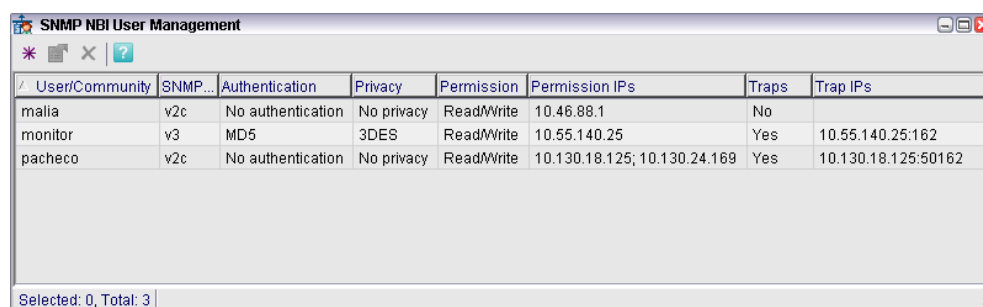
Being an SNMP agent, SNMP NBI also implements the MIB-II System Group variables, which describe the entity on which the agent is running: *sysName*, *sysDescr*, *sysObjectID*, *sysLocation* and *sysContact*. To manage the values of those variables:

1. Open TNMS Client and go to **Administration > System Preferences > SNMP NBI**.
2. The **Name** (*sysName*), **Description** (*sysDescr*) and **Object ID** (*sysObjectID*) variables are set automatically by the SNMP NBI:

3. In **Location** (*sysLocation*), optionally provide an description for the location of this SNMP agent.
4. In **Contact** (*sysContact*), optionally provide contact information, such as the administrator's email or telephone.

## 3.2 SNMP User Configuration

To configure SNMP users and notification destinations, open TNMS Client and go to **Administration > SNMP NBI User Management**.



The screenshot shows a window titled "SNMP NBI User Management" with a table containing three users: malla, monitor, and pacheco. The table has columns for User/Community, SNMP version, Authentication, Privacy, Permission, Permission IPs, Traps, and Trap IPs.

User/Community	SNMP...	Authentication	Privacy	Permission	Permission IPs	Traps	Trap IPs
malla	v2c	No authentication	No privacy	Read/Write	10.46.88.1	No	
monitor	v3	MD5	3DES	Read/Write	10.55.140.25	Yes	10.55.140.25:162
pacheco	v2c	No authentication	No privacy	Read/Write	10.130.18.125; 10.130.24.169	Yes	10.130.18.125:50162

Selected: 0, Total: 3

Figure 3. List of SNMP NBI users

Here you can add, edit or remove SNMP users. Next sections explain how to configure a user.

### 3.2.1 SNMP User Identification

In the **New SNMP NBI User > Identification** tab (Figure 4), specify user details.

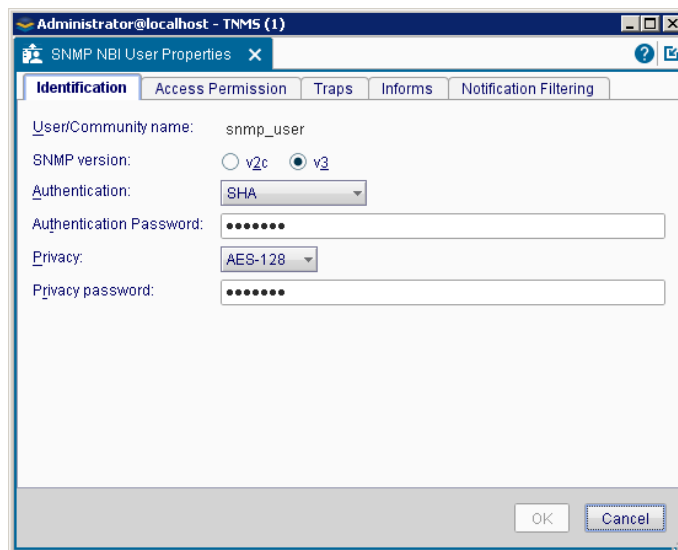


Figure 4 SNMP user identification

The following fields are mandatory:

- **User/Community:** name of the user (SNMPv3) or the community (SNMPv2). Maximum length is 32 characters.
- **SNMP version:** protocol version to be used for this user (v2c or v3).
- **Authentication:** encryption protocol for user authentication (optional, SNMPv3 only). Allowed values: MD5, SHA.
- **Authentication password:** password for user authentication (between 8 and 64 characters).
- **Privacy:** encryption protocol for privacy (optional, SNMPv3 only). Allowed values: DES, 3DES, AES128. You must enable authentication to enable privacy.
- **Privacy password:** password for privacy (between 8 and 64 characters).

### 3.2.2 SNMP user access permissions

In the **Access Permission** tab (Figure 5) configure the user permissions for incoming SNMP requests:

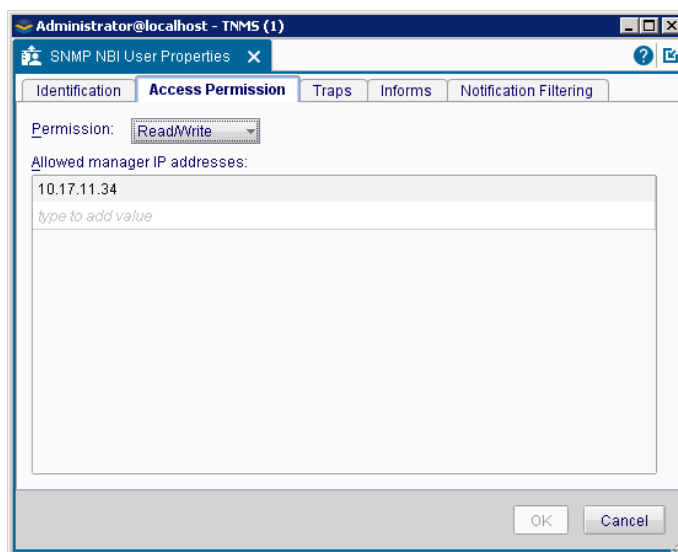


Figure 5 SNMP user access permissions

1. Select the level of permission granted to the user:
  - **No permission**
  - **Read**
  - **Read/Write**
2. Specify the IP addresses from which requests are accepted. Both IPv4 and IPv6 addresses are supported:



Address format restrictions:

- Host names are not supported (only IP addresses).
- IP masks are not supported (only individual IP addresses).

### 3.2.3 User trap destinations

In the **Traps** tab (Figure 6) configure the SNMP TRAP destinations for the SNMP user:

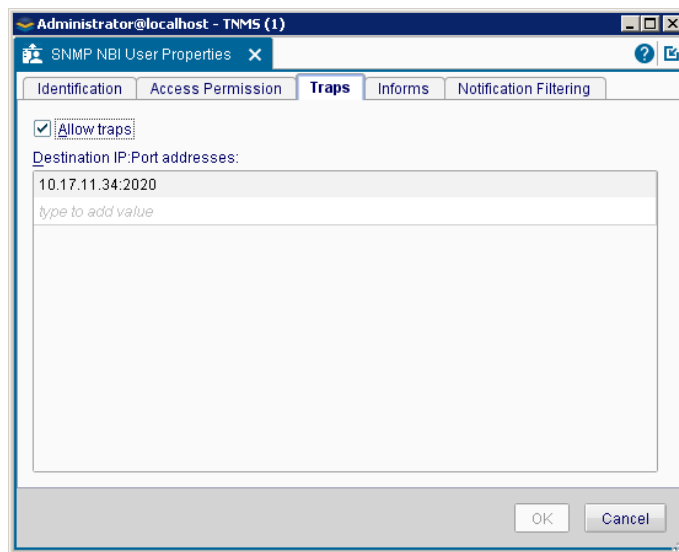


Figure 6 SNMP user trap destinations

To enable the SNMP user to receive traps:

1. Check **Allow traps**.
2. In **Destination IP addresses**, enter the IPv4 or IPv6 addresses of the SNMP managers that will receive the trap notification. You must always specify the port, even if it is the well-known port 162. Examples:
  - 10.55.140.25:162
  - [4f:0:0:0:0:58a3:45fe:38]:50162



Address format restrictions:

- Collapsing of groups of zeros in IPv6 trap destination addresses is not supported.
- Host names are not supported (only IP addresses).

### 3.2.4 User inform destinations

In the **Informs** tab (Figure 7) configure the SNMP INFORM destinations for the SNMP user.

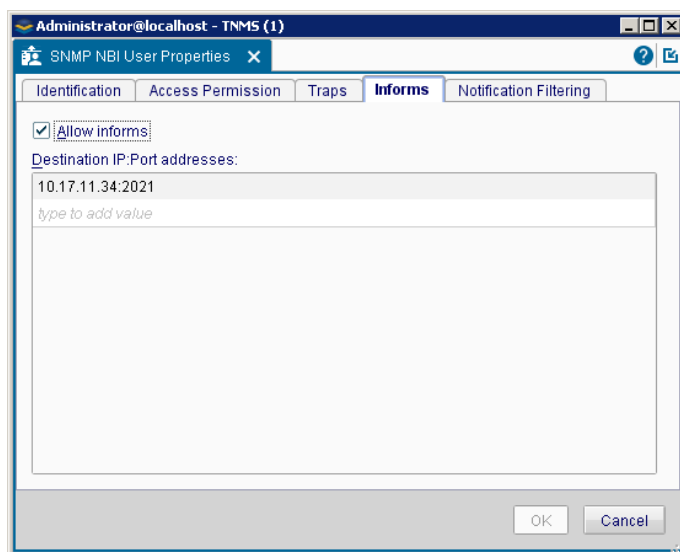


Figure 7 SNMP user inform destinations

To enable the SNMP user to receive informs:

1. Check **Allow informs**.
2. In **Destination IP addresses**, enter the IPv4 or IPv6 addresses of the SNMP managers that will receive the inform notifications. You must always specify the port, even if it is the well-known port 162. Examples:
  - 10.55.140.25:162
  - [4f:0:0:0:0:58a3:45fe:38]:50162



Address format restrictions:

- Collapsing of groups of zeros in IPv6 trap destination addresses is not supported.
- Host names are not supported (only IP addresses).

### 3.2.5 Notification Filtering

In the **Notification Filtering** tab, select what types of notifications are sent to the trap/inform destinations defined for the SNMP user.

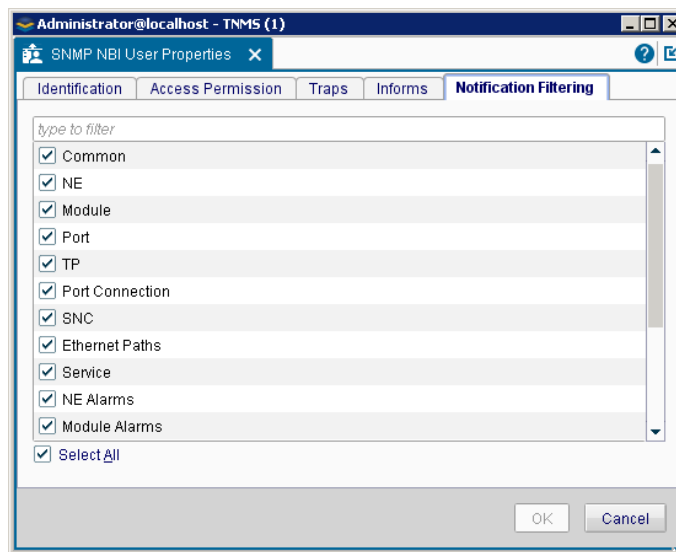


Figure 8 SNMP NBI notification preferences

Check types of notifications to be sent to the SNMP managers. Unchecked types of notifications will be blocked. Available options are:

- **Common:** SNMP NBI agent notification (for example, change of operational state).
- **NE:** All NE notifications, except alarms.
- **Module:** All module notifications, except alarms.
- **Port:** All port notifications, except alarms.
- **TP:** All TP notifications, except alarms.
- **Port Connection:** All port connection notifications.
- **SNC:** All SNC notifications.
- **Service:** All Service notifications.
- **Ethernet Path:** All Ethernet Path notifications.
- **NE Alarms:** Alarms originating in NEs.
- **Module Alarms:** Alarms originating in modules.
- **Port Alarms:** Alarms originating in ports.
- **TP alarms:** Alarms originating in TPs.
- **EMS Alarms:** Alarms originating in TNMS itself.
- **PM Request:** Notifications related to PM requests.
- **OPM Request:** Notifications related to OPM requests.
- **Monitor:** Notifications related to agent monitoring (for example, heartbeats)



Notification filtering is also configurable via SNMP by setting the *enmsTrapFilter* variables (see 8.3).



### 3.3 Heartbeat Notifications

SNMP NBI is able to send heartbeat notifications so that SNMP managers can detect network connectivity and agent availability issues. These notifications are sent to all Trap and Inform destinations configured in the SNMP NBI users.

Heartbeat notifications may be enabled and configured in the SNMP NBI System Preferences (see 3.1) or via SNMP (see 8.1).



Heartbeat notifications are always sent as SNMP TRAP, even for Inform destinations.



Heartbeat notifications do not carry a trap counter nor are stored in the notification history table.

### 3.4 Exporting SNMP Agent Configuration

The TNMS “Export Configuration” functionality may be used to export SNMP NBI configuration data to an XML file. However, exported data cannot be imported – its main use is for external auditing of changes.

For more information on how to export configuration data to XML, refer to TNMS documentation.

# 4 SNMP NBI MIB – General Description

## 4.1 Exported Object Model

Figure 9 diagrams the main objects exported by the SNMP NBI MIB, and how they relate to each other. The modelled objects correspond to table entries, and their relations are represented using primary keys (table indexes) and foreign keys (fields pointing to the indexes of related objects).

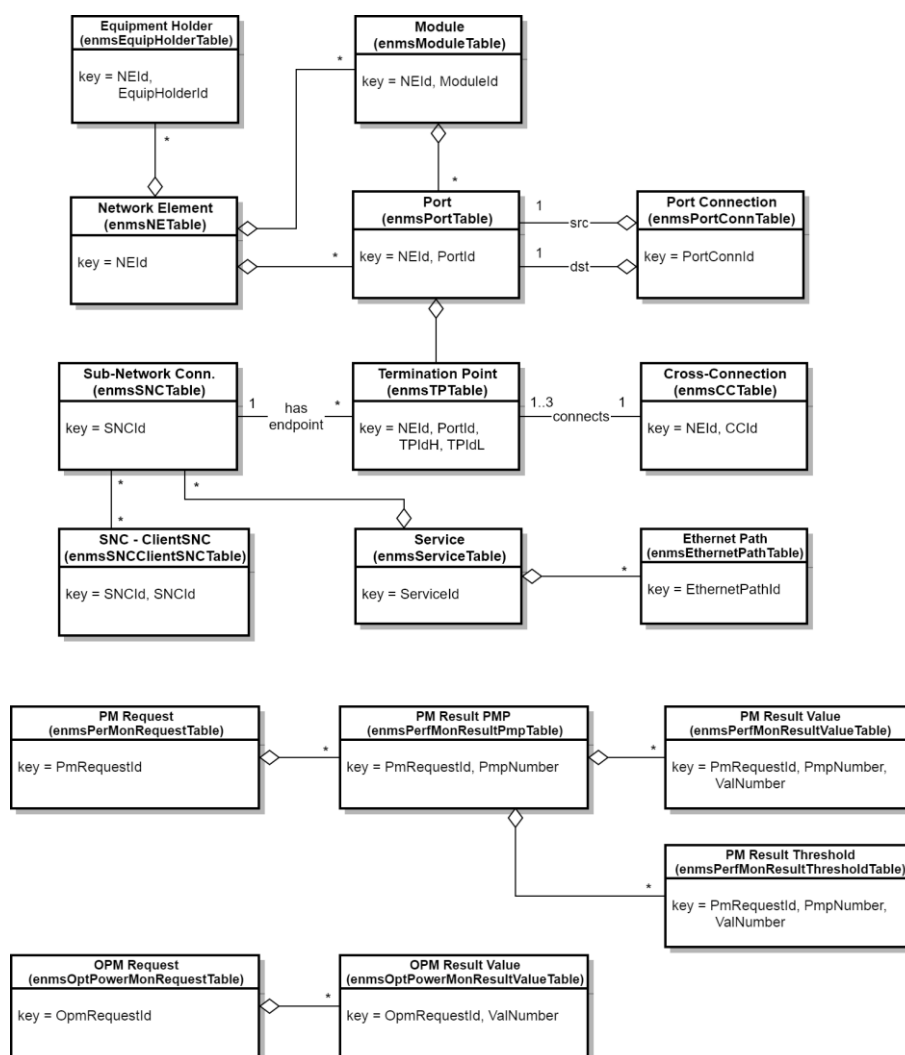


Figure 9 Exported object model

In relation to fault management, all active alarms in TNMS are exported via the table *enmsAlarmTable*. Other tables act as views of that master table, showing only the alarms affecting a specific type of object (Figure 10).



Figure 10 Exported model of alarms tables

## 4.2 Hierarchical View

SNMP NBI MIB is structured as follows:

- *coriant.svsProductMibs.svsProxEmns*
  - *enmsNetworkSetup*
    - *enmsNETable* (5.1.1)
    - *enmsModuleTable* (5.2.1)

- *enmsPortTable* (5.3.1)
- *enmsTPTable* (5.4.1)
- *enmsPortConnTable* (5.5.1)
- *enmsEquipHolderTable* (5.6.1)
- *enmsSNCTable* (6.1.1)
- *enmsCCTable* (6.4.1)
- *enmsEthernetPathTable* (6.2.1)
- *enmsSNCClientSNCTable* (6.1.6)
- *enmsService*
  - *enmsServiceTable* (6.3.1)
- *enmsAlarmTables*
  - *enmsAlarmTable* (7.2.1)
  - *enmsAlarmsForNETable* (7.2.2)
  - *enmsAlarmsForPortTable* (7.2.3)
  - *enmsAlarmsForTPTable* (7.2.4)
  - *enmsAlarmsForPortConnTable* (7.2.5)
  - *enmsAlarmsForSNCTable* (7.2.7)
  - *enmsAlarmsForModuleTable* (7.2.6)
  - *enmsAlarmsForEMSTable* (7.2.8)
- *enmsProxy*
  - *enmsControl* (8.1)
- *enmsTrapGroup*
  - *enmsTrapHistory*
    - *enmsTrapHistoryTable* (8.4)
  - *enmsTrapVariable*
  - *enmsTraps* (5.1, 5.2, 5.3, 5.4, 5.5, 6.1, 6.2, 6.3, 7.3, 8.2, 10.4, 11.4)
  - *enmsTrapFilter* (8.3)
- *enmsPerformanceMonitoring*
  - *enmsPerfMonRequestTable* (10.1.1)
  - *enmsPerfMonResultPmpTable* (10.6.2)
  - *enmsPerfMonResultValueTable* (10.6.3)
  - *enmsPerfMonResultThresholdTable* (10.6.4)
- *enmsOpticalPowerMonitoring*
  - *enmsOptPowerMonRequestTable* (11.1.1)
  - *enmsOptPowerMonResultValueTable* (11.6.2)

## 4.3 Notification Model

SNMP NBI sends notifications on changes and events affecting modelled objects. These notifications are based on the ITU recommendation X.721 and are realized as SNMP Traps.

Supported types of notifications:

Notification Type	Abbrev.	Meaning
ObjectCreation	OC	Creation of an object (for example, new NE in the <i>enmsNETable</i> ).
ObjectDeletion	OD	Deletion of an object (for example, NE removed from <i>enmsNETable</i> ).
StateChange	SC	Change of object state (for example, the operational state).
AttributeChange	AC	Change of a non-state object attribute (for example, the name).
Alarm	Alarm	Raise/clear of alarms originating from a network object or the EMS itself.

Table 4 Notification types

Notifications supported for each type of modelled object:

Modelled object	Supported notifications
NE	OC, OD, SC, AVC, Alarm
Module	OC, OD, SC, AVC, Alarm
Port	OC, OD, SC, AVC, Alarm
TP	OC, OD, SC, AVC, Alarm
CC	None
PortConn	OC, OD, AVC
SNC	OC, OD, SC, AVC
Service	OC, OD, SC, AVC
Ethernet Path	OC, OD, SC
Equipment Holder	None

Table 5 Supported notifications for modelled objects

In addition to the notifications for the modelled objects, there are also notifications associated to the SNMP agent and the EMS themselves:

Entity	Supported notifications
SNMP agent	SC (for example, the operational state)
EMS	Alarm (for example, "Not enough disk space")

Table 6 Supported notifications for SNMP agent and EMS

Finally, there are notifications related to Performance Monitoring and Optical Power Monitoring:

Object	Supported notifications
PM request	SC
OPM request	SC

Table 7 Supported notifications for PM and OPM requests

## 4.4 Resynchronization After Network or Manager Errors

When network errors occur, traps sent by the SNMP NBI may not reach the SNMP manager, causing it to become out-of-sync. An out-of-sync situation may also occur if the SNMP manager does receive the traps, but is unable to process them.

To help the SNMP manager to detect out-of-sync situations, all traps sent by the SNMP NBI include a counter, incremented by one for each trap, allowing the manager to detect if one or more traps are missing.



Using the trap counter to detect missing traps only works if the SNMP manager has all notifications types enabled in the SNMP User Configuration (see 3.2.5).

When an out-of-sync situation is detected, instead of resynchronizing all tables – which can be a heavy and lengthy process – the SNMP manager may try first to identify the objects associated to the missed traps by looking up the trap history table (*enmsTrapHistoryTable*, section 8.4). This table contains information on the last traps sent by the SNMP NBI, and its maximum length is configurable either via the GUI (see 3.1) or via SNMP by setting the *enmsTrapHistoryTableLength* MIB variable (see 8.1).

The trap history table does not contain the full trap details; its purpose is to allow the SNMP manager to resynchronize only the specific objects associated to the missed notifications.

## 4.5 Recommended Table Retrieval Approaches

This section recommends some approaches on how to efficiently access SNMP NBI tables using the GET / GETNEXT / GETBULK operations. The examples use the NE table (*enmsNETable*) and module table (*enmsModuleTable*), but the suggested approaches are similar for the remaining tables.

### 4.5.1 Retrieving all rows of a table

In general it is best to read the tables row by row, as opposed to column by column. The suggested approach to retrieve all rows is:

- Perform successive GETNEXT operations
- In each GETNEXT, request all fields of the row.

Example: NE table (*enmsNETable*) contains the following data:

enmsNeId (index)	enmsNeType	enmsNeName	...	enmsNeSystemContainer
2	ABC	NE2	...	SYS-1
5	XYZ	NE5	...	SYS-1
9	EFG	NE9	...	SYS-2
15	TYU	NE15	...	SYS-2

Table 8 Example data for *enmsNETable*

In this scenario, the SNMP manager first requests the first row of values:

```
GETNEXT(enmsNeId, enmsNeType, ..., enmsNeSystemContainer)
```

```
GETRESPONSE(enmsNeId.2 = 2, enmsNeType.2 = "ABC", ..., enmsNeSystemContainer.2 = "SYS-1")
```

Then it successively performs GETNEXT operations using the OIDs on the last response, until they don't match the columns of the requested OIDs and even fall out to the next table (which in this case is *enmsModuleTable*):

```
GETNEXT(enmsNeId.2, enmsNeType.2, ..., enmsNeSystemContainer.2)
```

```
GETRESPONSE(enmsNeId.5 = 2, enmsNeType.5 = "XYZ", ..., enmsNeSystemContainer.5 = "SYS-1")
```

```

GETNEXT(enmsNeId.5, enmsNeType.5, ..., enmsNeSystemContainer.5)
GETRESPONSE(enmsNeId.9 = 2, enmsNeType.9 = "EFG", ..., enmsNeSystemContainer.9 = "SYS-2")

GETNEXT(enmsNeId.9, enmsNeType.9, ..., enmsNeSystemContainer.9)
GETRESPONSE(enmsNeId.15 = 2, enmsNeType.15 = "TYU", ..., enmsNeSystemContainer.15 = "SYS-2")

GETNEXT(enmsNeId.15, enmsNeType.15, ..., enmsNeSystemContainer.15)
GETRESPONSE(enmsNeType.2 = "ABC", enmsNeName.2 = "NE2", ..., enmsMoNeId.2 = 2)

```



The GETNEXT operations above are requesting the index field (*enmsNeId*) just for clarity. In a real implementation, the SNMP manager may infer the index values from the OIDs of other fields in the same row.

#### 4.5.2 Retrieving specific rows by index

To retrieve a row by index, use the GET operation and request several column values at once:

```

GET(enmsNeType.9, ..., enmsNeIdName.9)
GETRESPONSE(enmsNeType.9 = "ABC", ..., enmsNeIdName.9 = "NE9")

```

#### 4.5.3 Retrieving blocks of rows for a sub-index

Suppose the SNMP manager needs to retrieve all modules of NE 9:

enmsMoNeId (index)	enmsMoModuleId (index)	enmsMoType	...	enmsMoObjectType
2	1	CARDX	...	1234
2	2	CARDY		321
5	1	CARDZ		545
9	4	CARDX	...	1234
9	5	CARDY	...	321
9	6	CARDZ		545
15	3	CARDX	...	1234

Table 9 Example data for enmsModuleTable

The SNMP manager starts by requesting the first row for NE 9. It performs a GETNEXT operation, where the supplied OIDs only include the *enmsMoNeId* part of the index (the *enmsMoModuleId* part of the index is therefore omitted):

```
GETNEXT(enmsMoNeId.9, enmsMoModuleId.9, enmsMoType.9, ..., enmsMoSlot.9)
```



```
GETRESPONSE(enmsMoNeId.9.4 = 9, enmsMoModuleId.9.4 = 4, enmsMoType.9.4 = "CARDX", ...,
enmsMoSlot.9.4 = 6)
```

Then it successively performs GETNEXT operations using the OIDs on the last response, until they belong to a different NE or fall outside of the table:

```
GETNEXT(enmsMoNeId.9.4, enmsMoModuleId.9.4, enmsMoType.9.4, ..., enmsMoSlot.9.4)
```

```
GETRESPONSE(enmsMoNeId.9.5 = 9, enmsMoModuleId.9.5 = 5, enmsMoType.9.5 = "CARDY", ...,
enmsMoSlot.9.5 = 4)
```

```
GETNEXT(enmsMoNeId.9.5, enmsMoModuleId.9.5, enmsMoType.9.5, ..., enmsMoSlot.9.5)
```

```
GETRESPONSE(enmsMoNeId.9.6 = 9, enmsMoModuleId.9.6 = 6, enmsMoType.9.6 = "CARDZ", ...,
enmsMoSlot.9.6 = 12)
```

```
GETNEXT(enmsMoNeId.9.6, enmsMoModuleId.9.6, enmsMoType.9.6, ..., enmsMoSlot.9.6)
```

```
GETRESPONSE(enmsMoNeId.15.3 = 15, enmsMoModuleId.15.3 = 3, enmsMoType.15.3 = "CARDX", ...,
enmsMoSlot.15.3 = 9)
```



The GETNEXT operations above are requesting the index fields (*enmsMoNeId* and *enmsMoModuleId*) just for clarity. In a real implementation, the SNMP manager may alternatively infer the index values from the OIDs of other fields in the same row.

#### 4.5.4 Limited size of responses

Every SNMP response must fit in a single UDP packet, whose maximum size depends on the network and server configuration. If a response will not fit, SNMP NBI normally responds with a *tooBig* error status – most commonly you'll need to reduce the *max-repetitions* parameter of the GETBULK requests.

In some cases SNMP NBI may not be able to automatically determine the maximum message size, causing oversized responses to be silently dropped. To prevent this situation, you may force a maximum message size by editing the SNMP NBI properties file, which can be found in the following location on the server installation:

```
<Product_Installation_Folder>
    /server/bicnet/deployments/bicnet.ear/conf/snmpNbi.properties
```

To set the maximum SNMP message size, add or change the `msgMaxSize` property as exemplified below:

```
msgMaxSize=8000
```

Restart server for changes to take effect.

### 4.5.5 Request timeouts

SNMP request response time depends on several factors, such as server load, network load, table being accessed and volume of data in the table. If timeout errors occur, you may need to increase the timeout value on the SNMP manager.



To confirm that a timeout error is not caused by a generic problem, such as lack of network connectivity or SNMP NBI misconfiguration, first perform a GET operation on a MIB leaf attribute, for example *enmsProxyName* (see 8.1). A response should be returned in a fairly short time.

## 4.6 Data Types

The following table lists the data types used in the SNMP NBI tables and traps. It is included here for convenience; please check MIB file for actual enumeration values.

Data type	Description
<b>AdministrativeState</b>	Administrative state of managed objects: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- locked(1) – Use of resource administratively prohibited.</li> <li>- unlocked(2) – Use of resource administratively permitted.</li> </ul>
<b>AlarmClass</b>	Enumeration of alarm classes: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- communication(1)</li> <li>- quality(2)</li> <li>- processing(3)</li> <li>- equipment(4)</li> <li>- environment(5)</li> <li>- system (6)</li> <li>- threshold (7)</li> <li>- security (8)</li> </ul>
<b>AlarmState</b>	Enumeration of alarm states: <ul style="list-style-type: none"> <li>- noAlarm(1)</li> <li>- acknowledged(2)</li> <li>- unAcknowledged(3)</li> </ul>
<b>Bandwidth</b>	Character string with one or more bandwidths separated by commas.
<b>Boolean</b>	Boolean value: <ul style="list-style-type: none"> <li>- false (0)</li> <li>- true (1)</li> </ul>
<b>CCId</b>	Global identifier of a CC (unsigned 32-bit integer).

Data type	Description
<b>CombinedSwitchState</b>	<p>Enumeration of path switch states:</p> <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- noSelector(1)</li> <li>- clear(2)</li> <li>- forceToWorking(3)</li> <li>- forceToProtecting(4)</li> <li>- manualToWorking(5)</li> <li>- manualToProtecting(6)</li> <li>- forcedExtraToProtection(7)</li> <li>- lockout(8)</li> <li>- forcedClear(9)</li> <li>- sfonwkg(10)</li> <li>- sfonprot(11)</li> <li>- sdonwkg(12)</li> <li>- sdonprot(13)</li> <li>- waitToRestore(14)</li> <li>- reverseRequest(15)</li> <li>- doNotRevert(16)</li> <li>- exercise(17)</li> <li>- activeSwitch(18)</li> <li>- mismatched(19)</li> <li>- disabled(20)</li> </ul>
<b>Directionality</b>	<p>Direction of a resource:</p> <ul style="list-style-type: none"> <li>- unknown(1)</li> <li>- unidirectional(2)</li> <li>- bidirectional(3)</li> </ul>
<b>DisplayString</b>	String of printable characters, meant to be displayed to humans. Non-ASCII characters are encoded in UTF-8.
<b>EnmsTimeStamp</b>	<p>Human-readable timestamp string. Format:</p> <p>yyyy-MM-dd HH:mm:ss</p> <p>Example: 2014-07-09 15:59:30</p> <p>Timestamps are in UTC.</p>
<b>EntityType</b>	<p>Type of entity:</p> <ul style="list-style-type: none"> <li>- other(0)</li> <li>- proxy(1)</li> <li>- module(2)</li> <li>- ne(3)</li> <li>- port(4)</li> <li>- tp(5)</li> <li>- portConn(6)</li> <li>- subNetworkConn(7)</li> <li>- subscriber(8)</li> <li>- service(9)</li> <li>- ems(10)</li> <li>- ethernetPath(11)</li> <li>- pmRequest(12)</li> <li>- opmRequest(13)</li> </ul>

Data type	Description
<b>EquipmentHolderType</b>	Type of equipment holder: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- rack(1)</li> <li>- shelf(2)</li> <li>- subShelf(3)</li> <li>- slot(4)</li> <li>- subSlot(5)</li> </ul>
<b>EquipmentHolderId</b>	Equipment holder identifier within an NE (unsigned 32-bit integer).
<b>EthernetPathId</b>	Global Identifier of an Ethernet Path (unsigned 32-bit integer).
<b>EthernetPathStatus</b>	Status of an Ethernet Path: <ul style="list-style-type: none"> <li>- active</li> <li>- inactive</li> <li>- inconsistent</li> <li>- incomplete</li> <li>- planned</li> <li>- inTest</li> <li>- activating</li> <li>- deactivating</li> <li>- activationFailed</li> <li>- deactivationFailed</li> </ul>
<b>EthernetPathType</b>	Type of Ethernet Path: <ul style="list-style-type: none"> <li>- pointToPoint(1)</li> <li>- multipointToMultipoint(2)</li> <li>- rootedMultipoint(3)</li> </ul>
<b>FilterType</b>	Type of object for which to collect PM/OPM data: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- tpObject(1)</li> <li>- portObject(2)</li> <li>- neObject(3)</li> <li>- sncObject(4)</li> <li>- ethernetPathObject(5)</li> <li>- moduleObject(6),</li> <li>- equipHolderObject(7)</li> </ul>
<b>InfoString</b>	Parsable character string.
<b>LayerSet</b>	Character string with one or more transport layers (for example, VC4 or OTU1) separated by commas.
<b>ModuleId</b>	Module identifier within an NE (unsigned 32-bit integer).
<b>NEClass</b>	(to be supported) Class of an NE. <ul style="list-style-type: none"> <li>- singleNode(1)</li> <li>- repeaterNode(2)</li> <li>- managementNode(3)</li> <li>- masterRingNode(4)</li> <li>- slaveRingNode(5)</li> </ul>
<b>NEId</b>	Global identifier of an NE (unsigned 32-bit integer).

Data type	Description
<b>NotificationType</b>	Type of notification in the history table: <ul style="list-style-type: none"> <li>- objectCreation(1)</li> <li>- objectDeletion(2)</li> <li>- stateChange(3)</li> <li>- attributeValueChange(4)</li> <li>- alarm(5)</li> <li>- relationshipChange(6)</li> </ul>
<b>OperatingMode</b>	Ability of an NE to send notifications: <ul style="list-style-type: none"> <li>- operation(1) – NE sends all notifications.</li> <li>- maintenance(2) – NE suppresses all notifications.</li> </ul>
<b>OperationalState</b>	Operability of a resource: <ul style="list-style-type: none"> <li>- unknown(1) – Operational state cannot be determined (for example, NE disconnected from DCN).</li> <li>- enabled(2) – Resource is operational.</li> <li>- disabled(3) – Resource is not operational.</li> <li>- partiallyEnabled(4) – Resource is partially operational.</li> </ul>
<b>OptPowerMonRequestId</b>	Global identifier of an OPM request.
<b>OptPowerMonRequestState</b>	State of an OPM request: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- created(1)</li> <li>- pending(2)</li> <li>- started(3)</li> <li>- finished(4)</li> <li>- failed(5)</li> <li>- cancelling(6)</li> <li>- cancelled(7)</li> <li>- deleting(8)</li> </ul>
<b>PerceivedSeverity</b>	Enumeration of alarm severities: <ul style="list-style-type: none"> <li>- cleared(0)</li> <li>- warning(1)</li> <li>- minor(2)</li> <li>- major(3)</li> <li>- critical(4)</li> <li>- indeterminate(5)</li> <li>- nonexistent(6)</li> <li>- nonAlarmed(7)</li> <li>- notAlarmed(8)</li> </ul>
<b>PerfMonDirection</b>	PMP direction: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- none(1)</li> <li>- incoming(2)</li> <li>- outgoing(3)</li> <li>- both(4)</li> </ul>
<b>PerfMonGranularity</b>	Granularity of PM data: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- minutes15(1)</li> <li>- hours24(2)</li> </ul>

Data type	Description
<b>PerfMonLocation</b>	Location of the PM measurement: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- nearEnd(1)</li> <li>- farEnd(2)</li> </ul>
<b>PerfMonRequestId</b>	Global identifier of a PM request.
<b>PerfMonRequestState</b>	State of a PM request: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- created(1)</li> <li>- pending(2)</li> <li>- started(3)</li> <li>- finished(4)</li> <li>- failed(5)</li> <li>- cancelling(6)</li> <li>- cancelled(7)</li> <li>- deleting(8)</li> </ul>
<b>PerfMonStatus</b>	Status of a PM parameter value: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- valid(1)</li> <li>- invalid(2)</li> <li>- incomplete(3)</li> </ul>
<b>PerfMonThresholdType</b>	Type of threshold value: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- lowest(1)</li> <li>- low(2)</li> <li>- high(3)</li> <li>- highest(4)</li> </ul>
<b>PerfMonType</b>	Type of PM data to retrieve: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- pmHistory(1)</li> <li>- pmCurrent(2)</li> <li>- pmPoints(3)</li> </ul>
<b>PortConnId</b>	Global identifier of a port connection (unsigned 32-bit integer).
<b>PortId</b>	Port identifier within an NE (unsigned 32-bit integer).
<b>ProbableCause</b>	Enumeration of alarm probable causes. See MIB for enumerated values.
<b>ProtectionState</b>	Protection state of a cross-connection: <ul style="list-style-type: none"> <li>- non(0) – Not applicable (for example, CC is unprotected).</li> <li>- working(1) – All traffic is being selected from the working channels.</li> <li>- protecting(2) – Traffic is being selected from at least one of the protecting channels.</li> </ul>

Data type	Description
<b>PTInterfaceType</b>	Type of interface medium: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- optical(1)</li> <li>- electrical(2)</li> <li>- radio(3)</li> </ul>
<b>PTProtectionType</b>	Protection type of a port, for MS Protections: <ul style="list-style-type: none"> <li>- unprotected(0) – No MS Protection.</li> <li>- msLtpWorking(1) – Working port in an MS-LTP.</li> <li>- msLtpWorkingExtra(2) – Working port in an MS-LTP with extra traffic.</li> <li>- msLtpProtecting(3) – Protecting port in an MS-LTP.</li> <li>- msLtpProtectingExtra(4) – Protecting port in an MS-LTP with extra traffic.</li> <li>- bshr2East(5) – East port in a 2-fibre BSHR.</li> <li>- bshr2EastExtra(6) – East port in a 2-fibre BSHR with extra traffic.</li> <li>- bshr2West(7) – West port in a 2-fibre BSHR.</li> <li>- bshr2WestExtra(8) – West port in a 2-fibre BSHR with extra traffic.</li> <li>- bshr4EastWorking(9) – East working port in a 4-fibre BSHR.</li> <li>- bshr4EastProtecting(10) – East protecting port in a 4-fibre BSHR.</li> <li>- bshr4WestWorking(11) – West working port in a 4-fibre BSHR.</li> <li>- bshr4WestProtecting(12) – West protecting port in a 4-fibre BSHR.</li> <li>- bshr4TransoceanicEastWorking(13) – East working port in a Transoceanic 4-fibre BSHR.</li> <li>- bshr4TransoceanicEastProtecting(14) – East protecting port in a Transoceanic 4-fibre BSHR.</li> <li>- bshr4TransoceanicWestWorking(15) – West working port in a Transoceanic 4-fibre BSHR.</li> <li>- bshr4TransoceanicWestProtecting(16) – West protecting port in a Transoceanic 4-fibre BSHR.</li> </ul>
<b>PTServiceType</b>	Service type of a port. The value is an integer whose bits are mapped as follows: <ul style="list-style-type: none"> <li>- bit 0 (value 1) – Obsolete.</li> <li>- bit 1 (value 2) – Supports bundle SNCs.</li> <li>- bit 2 (value 4) – May not be connected to other ports.</li> <li>- bit 3 (value 8) – Does not support unidirectional SNCs.</li> <li>- bit 4 (value 16) – Does not support flexible SNCs.</li> </ul>

Data type	Description
<b>PTTechnology</b>	<p>Technology of a port. This value is an integer whose bits are mapped as follows:</p> <ul style="list-style-type: none"> <li>- bit 0 (value 1) – PDH</li> <li>- bit 1 (value 2) – SDH</li> <li>- bit 2 (value 4) – ATM</li> <li>- bit 3 (value 8) – WDM</li> <li>- bit 4 (value 16) – Ethernet</li> <li>- bit 5 (value 32) – OTH</li> <li>- bit 6 (value 64) – Data</li> <li>- bit 7 (value 128) – Generic</li> <li>- bit 8 (value 256) – SONET</li> </ul> <p>A value of zero means indeterminate technology, not applicable or unknown.</p>
<b>SNCId</b>	Global identifier of an SNC (unsigned 32-bit integer).
<b>SNCProtectionInfo</b>	<p>Level of protection or broadcast of a connection:</p> <ul style="list-style-type: none"> <li>- unprotected(1) – 1 source and 1 destination endpoint.</li> <li>- simpleProtectionInfo(2) – 1 source and 2 destination endpoints (or vice-versa).</li> <li>- extendedProtectionInfo(3) – More than 2 source or 2 destination endpoints.</li> </ul>
<b>TPEndPointType</b>	<p>TP role in an SNC:</p> <ul style="list-style-type: none"> <li>- sourceTP(1)</li> <li>- destinationTP(2)</li> </ul>
<b>TPId</b>	Higher or lower 32 bits of a TP identifier within a port (unsigned 32-bit integer).
<b>TPIndex</b>	String with hexadecimal representation of a TP index. Example: 2ABF6
<b>TPReliability</b>	<p>Reliability of a TP:</p> <ul style="list-style-type: none"> <li>- preEmptive(1) - Resource is unreliable and will only be used for low priority services (as long it is not needed to protect high priority services).</li> <li>- unprotected(2) - Resource not protected by MSP.</li> <li>- protected(3) – Resource protected by MSP.</li> </ul>
<b>TPTerminationType</b>	<p>Termination type of a TP:</p> <ul style="list-style-type: none"> <li>- connection(1) – CTP</li> <li>- trail(2) – TTP</li> </ul>
<b>TPTYPE</b>	Termination point type. Obsolete.
<b>TrafficDirection</b>	<p>Traffic direction affected by an alarm:</p> <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- none(1)</li> <li>- inbound(2)</li> <li>- outbound(3)</li> <li>- both(4)</li> </ul>



Data type	Description
<b>TrafficDirectionTelcordia</b>	Traffic direction (Telcordia) affected by an alarm: <ul style="list-style-type: none"> <li>- unknown(0),</li> <li>- notApplicable(1)</li> <li>- unidirectionalClientToLine(2)</li> <li>- unidirectionalLineToClient(3)</li> <li>- unidirectionalRcv(4)</li> <li>- unidirectionalTrtm(5)</li> </ul>
<b>TrapFilter</b>	Indicates if traps are sent for a certain type of notifications: <ul style="list-style-type: none"> <li>- sendTrapsOn (1) – Traps are sent</li> <li>- sendTrapsOff (2) – Traps are blocked</li> </ul>
<b>Uniqueld</b>	Unsigned 32-bit unique identifier.
<b>UsageState</b>	The usage state of a resource: <ul style="list-style-type: none"> <li>- unknown(0)</li> <li>- idle(1) – Resource currently not in use.</li> <li>- active(2) – Not used.</li> <li>- busy(3) – Resource in use (spare operating capacity may be available).</li> </ul>

Table 10 Data types used in SNMP NBI MIB

## 4.7 Strings and Multi-Language Support

SNMP NBI supports multi-language by encoding strings in UTF-8. The MIB defines its own *DisplayString* type as an octet string with display hint “255t”, similar to *SnmpAdminString* type in SNMP-FRAMEWORK-MIB.

# 5

## Retrieving Network Inventory

Network infrastructure objects (NEs, Modules, Ports, TPs and Port Connections) can be retrieved by accessing the tables below:

- *enmsNETable*
- *enmsModuleTable*
- *enmsPortTable*
- *enmsTPTable*
- *enmsPortConnTable*
- *enmsEquipHolderTable*

For all tables above except *enmsEquipHolderTable*, SNMP NBI sends object creation (OC) and object deletion (OD) notifications, as well as state change (SC) and attribute value change (AVC) notifications for selected object attributes.

### 5.1 Network Elements

#### 5.1.1 List of NEs (*enmsNETable*)

Table *enmsNETable* contains all NEs in the network. It supports OC, OD, AVC and SC notifications.

Attribute name	Data type	Notif.	Description
<b><u>enmsNeNEId</u></b>	NEId		Global NE identifier (table index).
<b>enmsNeType</b>	DisplayString	AVC	NE type.
<b>enmsNeName</b>	DisplayString	AVC	NE name, as reported by the NE.
<b>enmsNeLocation</b>	DisplayString	AVC	NE location in the network map (topology container).  If the NE is in a system container, the system container name is returned.
<b>enmsNeAlarmSeverity</b>	PerceivedSeverity		Highest severity of all alarms affecting the NE or its modules, ports and TPs.

Attribute name	Data type	Notif.	Description
<b>enmsNeOperatingMode</b>	OperatingMode	AVC	NE's ability to send notifications: - <i>Operation</i> : NE reports all events to TNMS; - <i>Maintenance</i> : NE is suppressing all notifications (typically for maintenance purposes, to prevent heavy DCN load).
<b>enmsNeOpState</b>	OperationalState	SC	NE operational state.
<b>enmsNeCanBroadcast</b>	Boolean		Indicates if at least one port has broadcast capabilities.
<b>enmsNeCanPathProtection</b>	Boolean		Indicates if at least one port allows SNCP creation via TNMS.
<b>enmsNeClass</b>	NEClass		(to be supported)
<b>enmsNeExternalNEId</b>	InfoString		Obsolete.
<b>enmsNelsPseudoNE</b>	Boolean		Obsolete.
<b>enmsNeldName</b>	DisplayString	AVC	NE name, as specified by the operator in TNMS.
<b>enmsNeCommunicationState</b>	OperationalState	AVC	Indicates whether the communication with the NE is fully operational.
<b>enmsNeDCNLocation</b>	DisplayString		Path to the NE in the DCN tree. (to be supported)
<b>enmsNeSystemContainer</b>	DisplayString	AVC	System container name.
<b>enmsNeUserText</b>	DisplayString	AVC	User Text, as specified by the operator in the DCN property page of the NE.

Table 11 enmNETable attributes

### 5.1.2 Notification of NE object creation (enmsNEObjectCreationTrap)

Notification sent when an NE is added to the NE table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsNeNEId</b>	NEId	Global NE identifier.
<b>enmsNeName</b>	DisplayString	NE name, as reported by the NE.
<b>enmsTrapEventDetails</b>	DisplayString	Not used.
<b>enmsTrapNeldName</b>	DisplayString	NE name, as specified by the operator in TNMS.
<b>enmsNeSystemContainer</b>	DisplayString	System container of the NE.
<b>enmsNeUserText</b>	DisplayString	User Text as specified by the operator in the DCN property page of the NE.

Table 12 List of *enmsNEObjectCreationTrap* attributes

### 5.1.3 Notification of NE object deletion (*enmsNEObjectDeletionTrap*)

Notification sent when an NE is removed from the NE table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsNeNEId</b>	NEId	Global NE identifier.
<b>enmsNeName</b>	DisplayString	NE name, as reported by the NE.
<b>enmsTrapEventDetails</b>	DisplayString	Not used.
<b>enmsTrapNeldName</b>	DisplayString	NE name, as specified by the operator in TNMS.
<b>enmsNeSystemContainer</b>	DisplayString	System container of the NE.
<b>enmsNeUserText</b>	DisplayString	User Text as specified by the operator in the DCN property page of the NE.

Table 13 List of *enmsNEObjectDeletionTrap* attributes

### 5.1.4 Notification of NE state change (*enmsNEStateChangeTrap*)

Notification sent when an NE state attribute has a new value.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.

Variable name	Data type	Description
<b>enmsNeNEId</b>	NEId	Global NE identifier.
<b>enmsNeName</b>	DisplayString	NE name, as reported by the NE.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapStateName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapStateOldValue</b>	Integer	Old state value. Check the possible enumerated values of the changed attribute.
<b>enmsTrapStateNewValue</b>	Integer	New state value. Check the possible enumerated values of the changed attribute.
<b>enmsTrapNeldName</b>	DisplayString	NE name, as specified by the operator in TNMS.
<b>enmsNeSystemContainer</b>	DisplayString	System container of the NE.
<b>enmsNeUserText</b>	DisplayString	User Text as specified by the operator in the DCN property page of the NE.

Table 14 List of *enmsNEStateChangeTrap* attributes

### 5.1.5 Notification of NE attribute value change (enmsNEAttributeChangeTrap)

Notification sent when an NE attribute has a new value.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsNeNEId</b>	NEId	Global NE identifier.
<b>enmsNeName</b>	DisplayString	NE name, as reported by the NE.  Note: If the AVC relates to the NE name, this field contains the new value.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.

Variable name	Data type	Description
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapAttributeName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapAttributeOldValue</b>	DisplayString	Old attribute value.
<b>enmsTrapAttributeNewValue</b>	DisplayString	New attribute value.
<b>enmsTrapNeIdName</b>	DisplayString	NE name, as specified by the operator in TNMS.  Note: If the AVC relates to the NE Id name, this field contains the new value.
<b>enmsNeSystemContainer</b>	DisplayString	System container of the NE.
<b>enmsNeUserText</b>	DisplayString	User Text as specified by the operator in the DCN property page of the NE.

Table 15 List of *enmsNEAttributeChangeTrap* attributes

## 5.2 Modules

### 5.2.1 List of Modules (enmsModuleTable)

Table *enmsModuleTable* contains all modules of the NEs in the network. It supports OC, OD and SC notifications.

Attribute name	Data type	Notif.	Description
<b><u>enmsMoNEId</u></b>	NEId		NE identifier (table index).
<b><u>enmsMoModuleId</u></b>	ModuleId		Module Identifier within the NE (table index).
<b>enmsMoType</b>	DisplayString		Type of module. (Obsolete)
<b>enmsMoName</b>	DisplayString		Module name.
<b>enmsMoOpState</b>	OperationalState	SC	Operational state.

Attribute name	Data type	Notif.	Description
<b>enmsMoShelf</b>	DisplayString		Shelf and/or rack of the module, separated by a dash if both are present.
<b>enmsMoSlot</b>	DisplayString		Slot number of the module. If present, the sub-slot is appended, separated by a dash.
<b>enmsMoObjectType</b>	Integer32		Internal object type.
<b>enmsMoNativeLocation</b>	DisplayString	AVC	Optional native object location in the network element.

Table 16 List of *enmsModuleTable* attributes

### 5.2.2 Notification of Module object creation (enmsModuleObjectCreationTrap)

Notification sent when a new module is added to the module table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsMoNEId</b>	NEId	Global NE identifier.
<b>enmsMoModuleId</b>	ModuleId	Module identifier.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.

Table 17 List of *enmsModuleObjectCreationTrap* attributes

### 5.2.3 Notification of Module object deletion (enmsModuleObjectDeletionTrap)

Notification sent when a module is removed from the module table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsMoNEId</b>	NEId	Global NE identifier.
<b>enmsMoModuleId</b>	ModuleId	Module identifier.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.

Table 18 List of *enmsModuleObjectDeletionTrap* attributes

### 5.2.4 Notification of Module state change (enmsModuleStateChangeTrap)

Notification sent when a module state attribute has a new value.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsMoNEId	NEId	Global NE identifier.
enmsMoModuleId	ModuleId	Module identifier.
enmsTrapEventDetails	DisplayString	Empty.
enmsTrapEventSeverity	PerceivedSeverity	Not relevant.
enmsTrapEventProbableCause	ProbableCause	Not relevant.
enmsTrapStateName	Integer	Indicates which table field has changed. Check possible values in MIB definition.
enmsTrapStateOldValue	Integer	Old state value. Check the possible enumerated values of the changed attribute.
enmsTrapStateNewValue	Integer	New state value. Check the possible enumerated values of the changed attribute.

Table 19 List of *enmsModuleStateChangeTrap* attributes

### 5.2.5 Notification of Module attribute value change (enmsModuleAttributeChangeTrap)

Notification sent when a module attribute has a new value.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsMoNEId	NEId	Global NE identifier.
enmsMoModuleId	ModuleId	Module identifier.
enmsTrapEventDetails	DisplayString	Empty.
enmsTrapEventSeverity	PerceivedSeverity	Not relevant.
enmsTrapEventProbableCause	ProbableCause	Not relevant.



Variable name	Data type	Description
<b>enmsTrapAttributeName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapAttributeOldValue</b>	DisplayString	Old attribute value.
<b>enmsTrapAttributeNewValue</b>	DisplayString	New attribute value.

Table 20 List of *enmsModuleAttributeChangeTrap* attributes

## 5.3 Ports

### 5.3.1 List of Ports (enmsPortTable)

Table *enmsPortTable* contains all ports in the network. It supports OC, OD, AVC and SC notifications.

Attribute name	Data type	Notif.	Description
<b><u>enmsPtNEId</u></b>	NEId		NE identifier (table index).
<b><u>enmsPtPortId</u></b>	PortId		Port Identifier within the NE. (table index).
<b>enmsPtName</b>	DisplayString	AVC	Port name.
<b>enmsPtModuleId</b>	ModuleId		Obsolete.
<b>enmsPtTechnology</b>	PTTechnology		Transport technology. A value of zero means indeterminate technology, not applicable or unknown.
<b>enmsPtServiceType</b>	PTServiceType		Supported service types.
<b>enmsPtInterfaceType</b>	PTInterfaceType		Type of interface.
<b>enmsPtBandwidth</b>	Bandwidth		Port bandwidth.
<b>enmsPtOpState</b>	OperationalState		Obsolete.
<b>enmsPtOperatingMode</b>	OperatingMode		Obsolete.
<b>enmsPtDirection</b>	Directionality		Port direction.
<b>enmsPtCanBroadcast</b>	Boolean		Indicates if the port has broadcast capabilities.

Attribute name	Data type	Notif.	Description
<b>enmsPtCanPathProtection</b>	Boolean		Indicates if the port allows SNCP creation via TNMS.
<b>enmsPtAlarmSeverity</b>	PerceivedSeverity		Highest severity of all alarms affecting the port or its TPs.
<b>enmsPtAdminState</b>	AdministrativeState		Obsolete.
<b>enmsPtOpStateTX</b>	OperationalState	SC	Operational state in TX direction.
<b>enmsPtOpStateRX</b>	OperationalState	SC	Operational state in RX direction.
<b>enmsPtModuleIdTX</b>	ModuleId		Module ID for TX direction.
<b>enmsPtModuleIdRX</b>	ModuleId		Module ID for RX direction.
<b>enmsPtLayerSet</b>	LayerSet		Terminated layers.
<b>enmsPtProtectionType</b>	PTProtectionType	AVC	Protection type in the context of MS Protections.
<b>enmsPtObjectType</b>	Integer32		Internal object type.
<b>enmsPtNativeLocation</b>	DisplayString	AVC	Optional native object location in the network element.

Table 21 List of *enmsPortTable* attributes

### 5.3.2 Notification of Port object creation (enmsPortObjectCreationTrap)

Notification sent when a port is added to the port table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsPtNEId</b>	NEId	Global NE identifier.
<b>enmsPtPortId</b>	PortId	Port identifier.
<b>enmsPtName</b>	DisplayString	Port name.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.

Table 22 List of *enmsPortObjectCreationTrap* attributes

### 5.3.3 Notification of Port object deletion (enmsPortObjectDeletionTrap)

Notification sent when a port is removed from the port table.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsPtNEId	NEId	Global NE identifier.
enmsPtPortId	PortId	Port identifier.
enmsPtName	DisplayString	Port name.
enmsTrapEventDetails	DisplayString	Empty.

Table 23 List of *enmsPortObjectDeletionTrap* attributes

### 5.3.4 Notification of Port state change (enmsPortStateChangeTrap)

Notification sent when a port state attribute has a new value.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsPtNEId	NEId	Global NE identifier.
enmsPtPortId	PortId	Port identifier.
enmsPtName	DisplayString	Port name.
enmsTrapEventDetails	DisplayString	Empty.
enmsTrapEventSeverity	PerceivedSeverity	Not relevant.
enmsTrapEventProbableCause	ProbableCause	Not relevant.
enmsTrapStateName	Integer	Indicates which table field has changed. Check possible values in MIB definition.
enmsTrapStateOldValue	Integer	Old state value. Check the possible enumerated values of the changed attribute.
enmsTrapStateNewValue	Integer	New state value. Check the possible enumerated values of the changed attribute.

Table 24 List of *enmsPortStateChangeTrap* attributes

### 5.3.5 Notification of Port attribute value change (enmsPortAttributeChangeTrap)

Notification sent when a port attribute has a new value.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsPtNEId	NEId	Global NE identifier.
enmsPtPortId	PortId	Port identifier.
enmsPtName	DisplayString	Port name. Note: If the AVC relates to the port name, this field contains the new value.
enmsTrapEventDetails	DisplayString	Empty.
enmsTrapEventSeverity	PerceivedSeverity	Not relevant.
enmsTrapEventProbableCause	ProbableCause	Not relevant.
enmsTrapAttributeName	Integer	Indicates which table field has changed. Check possible values in MIB definition.
enmsTrapAttributeOldValue	DisplayString	Old attribute value.
enmsTrapAttributeNewValue	DisplayString	New attribute value.

Table 25 List of *enmsPortAttributeChangeTrap* attributes

## 5.4 Termination Points

### 5.4.1 List of TPs (enmsTPTable)

Table *enmsTPTable* contains all termination points in the network. It supports OC, OD, AVC and SC notifications.

Attribute name	Data type	Notif.	Description
<u>enmsTpNEId</u>	NEId		NE Id (table index).
<u>enmsTpPortId</u>	PortId		Port Id (table index).
<u>enmsTpTPIdH</u>	TPId		Higher 32 bits of TP Id (table index).

Attribute name	Data type	Notif.	Description
<b><u>enmsTpTPIdL</u></b>	TPId		Lower 32 bits of TP Id (table index).
<b>enmsTpTPIndex</b>	TPIndex		TP index relative to the port (might be a timeslot index).
<b>enmsTpNxCount</b>	Integer32	AVC	The number of carried signals in an Inverse Multiplexing group, if applicable. Otherwise, value is 1.
<b>enmsTpName</b>	DisplayString	AVC	TP name.
<b>enmsTpBandwidth</b>	Bandwidth		TP bandwidth.
<b>enmsTpTPTType</b>	TPTType		Obsolete.
<b>enmsTpTerminType</b>	TPTerminationType		Termination type.
<b>enmsTpDirection</b>	Directionality		TP direction.
<b>enmsTpOpStateTX</b>	OperationalState	SC	Operational state in TX direction.
<b>enmsTpOpStateRX</b>	OperationalState	SC	Operational state in RX direction.
<b>enmsTpAlarmSeverity</b>	PerceivedSeverity		Highest severity of all alarms affecting the TPs.
<b>enmsTpAdminState</b>	OperationalState		Obsolete.
<b>enmsTpUsageCountTX</b>	Integer32		(to be supported) Number of cross connections using the TP for the TX direction.
<b>enmsTpUsageCountRX</b>	Integer32		(to be supported) Number of cross connections using the TP for the RX direction.
<b>enmsTpUsageStateTX</b>	UsageState		(to be supported) Usage state in the TX direction.
<b>enmsTpUsageStateRX</b>	UsageState		(to be supported) Usage state in the RX direction.
<b>enmsTpReliability</b>	TPReliability		Obsolete.

Attribute name	Data type	Notif.	Description
<b>enmsTpLayerSet</b>	LayerSet		TP transport layer set.
<b>enmsTpBandwidthTX</b>	Bandwidth		(to be supported) TP bandwidth in the TX direction.
<b>enmsTpBandwidthRX</b>	Bandwidth		(to be supported) TP bandwidth in the RX direction.
<b>enmsTpParentPortId</b>	PortId		Port Id of parent TP (if applicable).
<b>enmsTpParentTPIdH</b>	TPId		Higher 32 bits of TP Id of parent TP (if applicable).
<b>enmsTpParentTPIdL</b>	TPId		Lower 32 bits of TP Id of parent TP (if applicable).
<b>enmsTpFragmentLayer</b>	LayerSet		Fragment layer set, if the TP represents a VC group.
<b>enmsTpObjectType</b>	Integer32		Internal object type.
<b>enmsTpMuxPartnerTPIdH</b>	TPId		Higher 32 bits of TP Id of multiplex partner TP (if applicable).
<b>enmsTpMuxPartnerTPIdL</b>	TPId		Lower 32 bits of TP Id of multiplex partner TP (if applicable).
<b>enmsTpNativeLocation</b>	DisplayString	AVC	Optional native object location in the network element.

Table 26 List of *enmsTPTable* attributes

### 5.4.2 Notification of TP object creation (enmsTPObjectCreationTrap)

Notification sent when a TP is added to the TP table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsTpNEId</b>	NEId	Global NE identifier.
<b>enmsTpPortId</b>	PortId	Port identifier.
<b>enmsTpTPIdH</b>	TPId	Higher 32 bits of TP Id.

Variable name	Data type	Description
enmsTpTPIdL	TPId	Lower 32 bits of TP Id.
enmsTpName	DisplayString	TP name.
enmsTpTPTYPE	TPTYPE	Obsolete. Always <i>unknown</i> .
enmsTrapEventDetails	DisplayString	Empty.

Table 27 List of *enmsTPOBJECTCreationTrap* attributes

### 5.4.3 Notification of TP object deletion (enmsTPOBJECTDeletionTrap)

Notification sent when a TP is removed from the TP table.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsTpNEId	NEId	Global NE identifier.
enmsTpPortId	PortId	Port identifier.
enmsTpTPIdH	TPId	Higher 32 bits of TP Id.
enmsTpTPIdL	TPId	Lower 32 bits of TP Id.
enmsTpName	DisplayString	TP name.
enmsTpTPTYPE	TPTYPE	Obsolete. Always <i>unknown</i> .
enmsTrapEventDetails	DisplayString	Empty.

Table 28 List of *enmsTPOBJECTDeletionTrap* attributes

### 5.4.4 Notification of TP state change (enmsTPStateChangeTrap)

Notification sent when a TP state attribute has a new value.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsTpNEId	NEId	Global NE identifier.
enmsTpPortId	PortId	Port identifier.
enmsTpTPIdH	TPId	Higher 32 bits of TP Id.
enmsTpTPIdL	TPId	Lower 32 bits of TP Id.

Variable name	Data type	Description
<b>enmsTpName</b>	DisplayString	TP name.
<b>enmsTpTPType</b>	TPType	Obsolete. Always <i>unknown</i> .
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapStateName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapStateOldValue</b>	Integer	Old state value. Check the possible enumerated values of the changed attribute.
<b>enmsTrapStateNewValue</b>	Integer	New state value. Check the possible enumerated values of the changed attribute.

Table 29 List of *enmsTPStateChangeTrap* attributes

#### 5.4.5 Notification of TP attribute value change (enmsTPAttributeChangeTrap)

Notification sent when a TP attribute has a new value.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsTpNEId</b>	NEId	Global NE identifier.
<b>enmsTpPortId</b>	PortId	Port identifier.
<b>enmsTpTPIdH</b>	TPId	Higher 32 bits of TP Id.
<b>enmsTpTPIdL</b>	TPId	Lower 32 bits of TP Id.
<b>enmsTpName</b>	DisplayString	TP name.
<b>enmsTpTPType</b>	TPType	Obsolete. Always <i>unknown</i> .
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.



Variable name	Data type	Description
<b>enmsTrapAttributeName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapAttributeOldValue</b>	DisplayString	Old attribute value.
<b>enmsTrapAttributeNewValue</b>	DisplayString	New attribute value.

Table 30 List of *enmsTPAttributeChangeTrap* attributes

## 5.5 Port Connections

### 5.5.1 List of Port Connections (enmsPortConnTable)

Table *enmsPortConnTable* contains all managed ports connections. It supports OC, OD and AVC notifications.

Attribute name	Data type	Notif.	Description
<b><u>enmsPcPortConnId</u></b>	PortConnId		Port Connection identifier (table index).
<b>enmsPcSrcNEId</b>	NEId		NE Id of source port.
<b>enmsPcSrcPortId</b>	PortId		Port Id of source port.
<b>enmsPcDstNEId</b>	NEId		NE Id of destination port.
<b>enmsPcDstPortId</b>	PortId		Port Id of destination port.
<b>enmsPcName</b>	DisplayString	AVC	Port connection name.
<b>enmsPcSrcAlarmSeverity</b>	PerceivedSeverity	AVC	Highest severity of all alarms affecting the source port or module.
<b>enmsPcDstAlarmSeverity</b>	PerceivedSeverity	AVC	Highest severity of all alarms affecting the destination port or module.
<b>enmsPcBandwidth</b>	Bandwidth		Port connection bandwidth.
<b>enmsPcDirection</b>	Directionality		Port connection direction.
<b>enmsPcLayerSet</b>	LayerSet		Port connection layer set.

Table 31 List of *enmsPortConnTable* attributes

### 5.5.2 Notification of Port Connection object creation (enmsPortConnObjectCreationTrap)

Notification sent when a Port Connection is added to the Port Connection table.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsPcPortConnId	PortConnId	Global port connection identifier.
enmsPcName	DisplayString	Port connection name.
enmsTrapEventDetails	DisplayString	Empty.

Table 32 List of *enmsPortConnObjectCreationTrap* attributes

### 5.5.3 Notification of Port Connection object deletion (enmsPortConnObjectDeletionTrap)

Notification sent when a Port Connection is removed from the Port Connection table.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsPcPortConnId	PortConnId	Global port connection identifier.
enmsPcName	DisplayString	Port connection name.
enmsTrapEventDetails	DisplayString	Empty.

Table 33 List of *enmsPortConnObjectDeletionTrap* attributes

### 5.5.4 Notification of Port Connection attribute value change (enmsPortConnAttributeChangeTrap)

Notification sent when a Port Connection attribute has a new value.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsPcPortConnId	PortConnId	Global port connection identifier.
enmsPcName	DisplayString	Port connection name.

Variable name	Data type	Description
		Note: If the AVC relates to the port connection name, this field contains the new value.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapAttributeName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapAttributeOldValue</b>	DisplayString	Old attribute value.
<b>enmsTrapAttributeNewValue</b>	DisplayString	New attribute value.

Table 34 List of *enmsPortConnAttributeChangeTrap* attributes

## 5.6 Equipment Holders

### 5.6.1 List of Equipment Holders (enmsEquipHolderTable)

Table *enmsEquipHolderTable* contains all shelves. No notifications are sent for equipment holders.

Attribute name	Data type	Notif.	Description
<b><u>enmsEhNEId</u></b>	NEId		NE Identifier (table index).
<b>enmsEhEquipHolderId</b>	EquipmentHolderId		Equipment holder identifier within the NE (table index).
<b>enmsEhType</b>	EquipmentHolderType		Type of equipment holder. <b>Note:</b> Currently only shelves are supported.
<b>enmsEhName</b>	DisplayString		Equipment holder name.

Table 35 List of *enmsEquipHolderTable* attributes

# 6

## Retrieving Paths, Services and Cross-connections

Connection-related objects (optical paths, ethernet paths, cross-connections and services) can be retrieved by accessing the following tables:

- *enmsSNCTable*
- *enmsEthernetPathTable*
- *enmsCrossConnectionTable*
- *enmsServiceTable*

For all tables above, except *enmsCrossConnectionTable*, SNMP NBI sends object creation (OC) and object deletion (OD) notifications, as well as state change (SC) and attribute value change (AVC) notifications for selected object attributes.

### 6.1 Sub-Network Connections (Optical Paths)

#### 6.1.1 List of SNCs (*enmsSNCTable*)

Table *enmsSNCTable* contains all paths of the Optical Manager. It supports OC, OD, AVC and SC notifications.

Attribute name	Data type	Notif.	Description
<u><b>enmsScSNCId</b></u>	SNCId		SNC Id (table index).
<b>enmsScSrcNEId</b>	NEId		Identifier of the first A-end in the list of path edges.
<b>enmsScSrcPortId</b>	PortId		
<b>enmsScSrcTPIdH</b>	TPId		
<b>enmsScSrcTPIdL</b>	TPId		
<b>enmsScDestNEId</b>	NEId		Identifier of the first Z-end in the list of path edges.
<b>enmsScDestPortId</b>	PortId		
<b>enmsScDestTPIdH</b>	TPId		
<b>enmsScDestTPIdL</b>	TPId		
<b>enmsScSrc2NEId</b>	NEId		

Attribute name	Data type	Notif.	Description
<b>enmsScSrc2PortId</b>	PortId		Identifier of the second A-end in the list of path edges.
<b>enmsScSrc2TPIdH</b>	TPId		
<b>enmsScSrc2TPIdL</b>	TPId		
<b>enmsScDest2NEId</b>	NEId		Identifier of the second Z-end in the list of path edges.
<b>enmsScDest2PortId</b>	PortId		
<b>enmsScDest2TPIdH</b>	TPId		
<b>enmsScDest2TPIdL</b>	TPId		
<b>enmsScServiceId</b>	ServiceId		Identifier of enclosing service, if applicable.
<b>enmsScName</b>	DisplayString	AVC	SNC name as presented in the TNMS GUI.
<b>enmsScOpState</b>	OperationalState	SC	Operational state of the SNC: <ul style="list-style-type: none"> <li>- <i>enabled</i> if ACS is 'Active' and Operational State is 'Enabled'.</li> <li>- <i>partiallyEnabled</i> if ACS is 'Active' and Operational State is 'Protection Disturbed', 'Server Disturbed', 'Degraded' or 'Disabled Local'.</li> <li>- <i>disabled</i> if ACS is 'Disabled' or Operation State is 'Disabled', 'Server Disabled' or 'Disabled Transport'.</li> <li>- <i>unknown</i> if not applicable or not possible to determine.</li> </ul>
<b>enmsScAdminState</b>	AdministrativeState	SC	Administrative state. Reflects the RCS of the path: 'unlocked' if RCS is 'Active', otherwise 'locked'.

Attribute name	Data type	Notif.	Description
<b>enmsScAlarmSeverity</b>	PerceivedSeverity	AVC	Highest severity of all alarms affecting this SNC.  Depending on TNMS configuration, this field may also reflect the highest severity of alarms affecting the SNC's server SNCs. Note however that <i>enmsAlarmsForSNCTable</i> will not associate the SNC to those server SNCs alarms.
<b>enmsScBandwidth</b>	Bandwidth		SNC bandwidth.
<b>enmsScDirection</b>	Directionality		SNC direction.
<b>enmsScProtectionFlag</b>	Boolean		Obsolete.
<b>enmsScProtectionInfo</b>	SNCProtectionInfo		Level of SNC protection or broadcast.  Obsolete.
<b>enmsScNxCount</b>	Unsigned32		Inverse multiplexing number. Only applicable if the SNC represents a virtual concatenated group.
<b>enmsScSNCOwnerId</b>	SNCId		(to be supported)
<b>enmsSCLayerSet</b>	LayerSet		Layer set of the SNC.
<b>enmsScFragmentLayer</b>	LayerSet		The layer of the fragments. Only applicable if the SNC represents a virtual concatenated group.
<b>enmsScMinBandwidth</b>	Bandwidth	AVC	Minimum actual or required bandwidth of all edges of the SNC connection topology.
<b>enmsScRequiredBandwidth</b>	Boolean	AVC	Indicates whether the value of the attribute Bandwidth is required (explicitly assigned by the operator).
<b>enmsScSwitchState</b>	CombinedSwitchState	SC	SNC protection switch state.

Table 36 List of *enmsSNCTable* attributes

### 6.1.2 Notification of SNC object creation (enmsSNCObjectCreationTrap)

Notification sent when a new SNC is added to the SNC table.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsScSNCId	SNCId	Global SNC identifier.
enmsScName	DisplayString	SNC name.
enmsTrapEventDetails	DisplayString	Empty.

Table 37 List of *enmsSNCObjectCreationTrap* attributes

### 6.1.3 Notification of SNC object deletion (enmsSNCObjectDeletionTrap)

Notification sent when an SNC is removed from the SNC table.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsScSNCId	SNCId	Global SNC identifier.
enmsScName	DisplayString	SNC name.
enmsTrapEventDetails	DisplayString	Empty.

Table 38 List of *enmsSNCObjectDeletionTrap* attributes

### 6.1.4 Notification of SNC state change (enmsSNCStateChangeTrap)

Notification sent when an SNC state attribute has a new value. This notification is also used to report protection switch events on CCs and protection groups affecting SNCs (see 0).

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsScSNCId	SNCId	Global SNC identifier.
enmsScName	DisplayString	SNC name.
enmsTrapEventDetails	DisplayString	Empty.
enmsTrapEventSeverity	PerceivedSeverity	Not relevant.
enmsTrapEventProbableCause	ProbableCause	Not relevant.

Variable name	Data type	Description
<b>enmsTrapStateName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapStateOldValue</b>	Integer	Old state value. Check the possible enumerated values of the changed attribute.
<b>enmsTrapStateNewValue</b>	Integer	New state value. Check the possible enumerated values of the changed attribute.

Table 39 List of *enmsSNCStateChangeTrap* attributes

### 6.1.5 Notification of SNC attribute value change (*enmsSNCAttributeChangeTrap*)

Notification sent when an SNC attribute has a new value.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsScSNCId</b>	SNCId	Global SNC identifier.
<b>enmsScName</b>	DisplayString	SNC name. Note: If the AVC relates to the SNC name, this field contains the new value.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapAttributeName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapAttributeOldValue</b>	DisplayString	Old attribute value.
<b>enmsTrapAttributeNewValue</b>	DisplayString	New attribute value.

Table 40 List of *enmsSNCAttributeChangeTrap* attributes



### 6.1.6 Association between SNCs and their client SNCs (enmsSNCClientSNCTable)

Table *enmsSNCClientSNCTable* associates paths with their direct client paths. No notifications are sent for this table.

Attribute name	Data type	Notif.	Description
<b>enmsScsSNCId</b>	SNCId		SNC Id (table index).
<b>enmsScsClientSNCId</b>	SNCId		SNC Id of a direct client path.

Table 41 List of *enmsSNCClientSNCTable* attributes

### 6.1.7 Receiving protection switch notifications

When a protection switch event occurs on a protected CC or on a protection group (such as OMSP), SNMP NBI notifies that change by sending an *enmsSNCStateChangeTrap* notification on the *most server* affected SNC. These notifications will have *enmsTrapStateName* set to 9 (which means *protectionState*).



Although *enmsSNCStateChangeTrap* is being used to notify protection switch occurrences on the network, it does not reflect any attribute of the SNC itself.

The following table describes *enmsSNCStateChangeTrap* contents for protection switch notifications.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsScsSNCId</b>	SNCId	Global SNC identifier.
<b>enmsScsName</b>	DisplayString	SNC name.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapStateName</b>	Integer	For protection switch notifications, the value is 9 - protectionState.
<b>enmsTrapStateOldValue</b>	Integer	Old state value. For protection state changes on CCs, values may be: - 0 - Not applicable

Variable name	Data type	Description
		<ul style="list-style-type: none"> <li>- 1 - Traffic is selected from a working channel</li> <li>- 2 - Traffic is selected from a protecting channel</li> </ul> <p>For protection state changes on protection groups, this value is unavailable (always 0).</p>
<b>enmsTrapStateNewValue</b>	Integer	<p>New state value. For protection state changes on CCs or protection groups, values may be:</p> <ul style="list-style-type: none"> <li>- 0 - Not applicable</li> <li>- 1 - Traffic is selected from a working channel</li> <li>- 2 - Traffic is selected from a protecting channel</li> </ul>

Table 42 List of *enmsSNCStateChangeTrap* attributes for protection switch notifications



Since SNMP NBI does not model protection groups, the protection switch notifications do not carry the old protection state, only the new one. Also, protection switch events on protection groups may generate redundant notifications or notifications for intermediate conditions.

## 6.2 Ethernet Paths

### 6.2.1 List of Ethernet Paths (enmsEthernetPathTable)

Table *enmsEthernetPathTable* contains all the Ethernet services of the Ethernet Manager. It supports OC, OD, and SC notifications.

Attribute name	Data type	Notif.	Description
<b><u>enmsEvcEthernetPathId</u></b>	EthernetPathId		Ethernet Path Id (table index)
<b>enmsEvcName</b>	DisplayName	AVC	Ethernet Path name.
<b>enmsEvcSVlanId</b>	Integer32		S-VLAN Id
<b>enmsEvcType</b>	EthernetPathType		Ethernet Path type.
<b>enmsEvcServiceId</b>	ServiceIdentifier		Identifier of enclosing service, if applicable.

Attribute name	Data type	Notif.	Description
<b>enmsEvcOpState</b>	OperationalState	SC	Operational State
<b>enmsEvcAdminState</b>	AdministrativeState	SC	Administration State
<b>enmsEvcAlarmSeverity</b>	PerceivedSeverity	AVC	Highest severity of all alarms affecting this Ethernet Path.

Table 43 List of *enmsEthernetPathTable* attributes

### 6.2.2 Notification of Ethernet Path object creation (enmsEthernetPathObjectCreationTrap)

Notification sent when a new Ethernet Path is added to the Ethernet Path table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsEvcEthernetPathId</b>	EthernetPathId	Global Ethernet Path identifier.
<b>enmsEvcName</b>	DisplayString	Ethernet Path name.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.

Table 44 List of *enmsEthernetPathObjectCreationTrap* attributes

### 6.2.3 Notification of Ethernet Path object deletion (enmsEthernetPathObjectDeletionTrap)

Notification sent when an Ethernet Path is removed from the Ethernet Path table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsEvcEthernetPathId</b>	EthernetPathId	Global Ethernet Path identifier.
<b>enmsEvcName</b>	DisplayString	Ethernet Path name.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.

Table 45 List of *enmsEthernetPathObjectDeletionTrap* attributes

### 6.2.4 Notification of Ethernet Path state change (enmsEthernetPathStateChangeTrap)

Notification sent when an Ethernet Path state attribute has a new value.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsEvcEthernetPathId	EthernetPathId	Global Ethernet Path identifier.
enmsEvcName	DisplayString	Ethernet Path name.
enmsTrapEventDetails	DisplayString	Empty.
enmsTrapEventSeverity	PerceivedSeverity	Not relevant.
enmsTrapEventProbableCause	ProbableCause	Not relevant.
enmsTrapStateName	Integer	Indicates which table field has changed. Check possible values in MIB definition.
enmsTrapStateOldValue	Integer	Old state value. Check the possible enumerated values of the changed attribute.
enmsTrapStateNewValue	Integer	New state value. Check the possible enumerated values of the changed attribute.

Table 46 List of *enmsEthernetPathStateChangeTrap* attributes

### 6.2.5 Notification of Ethernet Path attribute value change (enmsEthernetPathAttributeChangeTrap)

Notification sent when an Ethernet Path attribute has a new value.

Variable name	Data type	Description
enmsTrapCounter	Counter32	Trap counter for synchronization.
enmsEvcEthernetPathId	EthernetPathId	Global Ethernet Path identifier.
enmsEvcName	DisplayString	Ethernet Path name.  Note: If the AVC relates to the Ethernet Path name, this field contains the new value.
enmsTrapEventDetails	DisplayString	Empty.

Variable name	Data type	Description
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapAttributeName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapAttributeOldValue</b>	DisplayString	Old attribute value.
<b>enmsTrapAttributeNewValue</b>	DisplayString	New attribute value.

Table 47 List of *enmsEthernetPathAttributeChangeTrap* attributes

## 6.3 Services

### 6.3.1 List of Services (enmsServiceTable)

Table *enmsEthernetPathTable* contains all services defined in TNMS.



A 'service' corresponds to a TNMS container of type 'Service' – not to be confused with 'Ethernet Services', which in SNMP NBI nomenclature are called 'Ethernet Paths'.

Attribute name	Data type	Notif.	Description
<b>enmsSvServiceId (index)</b>	Unsigned32		Service Id (table index).
<b>enmsSvSubscriberId (index)</b>	Unsigned32		(to be supported)
<b>enmsSvLabel</b>	DisplayString	AVC	Name of the service.
<b>enmsSvOpState</b>	OperationalState	SC	Operational state of the service. It corresponds to the most severe operational state of the associated SNCs. From the least severe to the most severe operational state, we have: <ul style="list-style-type: none"> <li>- 'enabled'</li> <li>- 'partiallyEnabled'</li> <li>- 'disabled'</li> <li>- 'unknown'</li> </ul>
<b>enmsSvAdminState</b>	AdministrativeState		Unsupported
<b>enmsSvDirection</b>	Directionality		Obsolete

Attribute name	Data type	Notif.	Description
<b>enmsSvProtectionFlag</b>	Boolean		Obsolete
<b>enmsSvWriteProtected</b>	Boolean		Obsolete
<b>enmsSvServiceOwnerId</b>	Unsigned32		Obsolete

Table 48 List of *enmsServiceTable* attributes

### 6.3.2 Notification of Service object creation (enmsServiceObjectCreationTrap)

Notification sent when a new service is added to the service table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsSvServiceId</b>	ServiceId	Global service identifier.
<b>enmsSvLabel</b>	DisplayString	Service name.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.

Table 49 List of *enmsServiceObjectCreationTrap* attributes

### 6.3.3 Notification of Service object deletion (enmsServiceObjectDeletionTrap)

Notification sent when a service is removed from the service table.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsSvServiceId</b>	ServiceId	Global service identifier.
<b>enmsSvLabel</b>	DisplayString	Service name.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.

Table 50 List of *enmsServiceObjectDeletionTrap* attributes

### 6.3.4 Notification of Service state change (enmsServiceStateChangeTrap)

Notification sent when a service state attribute has a new value.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsSvServiceId</b>	ServiceId	Global service identifier.
<b>enmsSvLabel</b>	DisplayString	Service name.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapStateName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.
<b>enmsTrapStateOldValue</b>	Integer	Old state value. Check the possible enumerated values of the changed attribute.
<b>enmsTrapStateNewValue</b>	Integer	New state value. Check the possible enumerated values of the changed attribute.

Table 51 List of *enmsServiceStateChangeTrap* attributes

### 6.3.5 Notification of Service attribute value change (*enmsServiceAttributeChangeTrap*)

Notification sent when a service attribute has a new value.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsSvServiceId</b>	SNCId	Global service identifier.
<b>enmsSvLabel</b>	DisplayString	Service name.  Note: If the AVC relates to the SNC name, this field contains the new value.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Not relevant.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Not relevant.
<b>enmsTrapAttributeName</b>	Integer	Indicates which table field has changed. Check possible values in MIB definition.

Variable name	Data type	Description
<b>enmsTrapAttributeOldValue</b>	DisplayString	Old attribute value.
<b>enmsTrapAttributeNewValue</b>	DisplayString	New attribute value.

Table 52 List of *enmsServiceAttributeChangeTrap* attributes

## 6.4 Cross Connections

### 6.4.1 List of Cross Connections (enmsCCTable)

Table *enmsCCTable* contains all cross connections in the network. No notifications are sent for this table.

Attribute name	Data type	Notif.	Description
<b>enmsCcNEId (index)</b>	CCId		NE Id (table index).
<b>enmsCcCCId (index)</b>	CCId		CC identifier within NE (table index).
<b>enmsCcSrcNEId</b>	NEId		Identifier of the A-end TP.
<b>enmsCcSrcPortId</b>	PortId		
<b>enmsCcSrcTPIdH</b>	TPId		
<b>enmsCcSrcTPIdL</b>	TPId		
<b>enmsCcDestNEId</b>	NEId		Identifier of the Z-end TP.
<b>enmsCcDestPortId</b>	PortId		
<b>enmsCcDestTPIdH</b>	TPId		
<b>enmsCcDestTPIdL</b>	TPId		
<b>enmsCcSrc2NEId</b>	NEId		Identifier of the second A-end TP.
<b>enmsCcSrc2PortId</b>	PortId		
<b>enmsCcSrc2TPIdH</b>	TPId		
<b>enmsCcSrc2TPIdL</b>	TPId		
<b>enmsCcDest2NEId</b>	NEId		Identifier of the second Z-end TP.
<b>enmsCcDest2PortId</b>	PortId		



Attribute name	Data type	Notif.	Description
<b>enmsCcDest2TPIIdH</b>	TPId		
<b>enmsCcDest2TPIIdL</b>	TPId		
<b>enmsCcOpState</b>	OperationalState		Operational state.
<b>enmsCcDirection</b>	Directionality		CC direction.
<b>enmsCcProtectionFlag</b>	Boolean		Indicates whether CC is protected.
<b>enmsCcProtectionState</b>	ProtectionState		Actual protection state.
<b>enmsCcNxCount</b>	Unsigned32		Used when more than a single termination point is connected as a bundle.

Table 53 List of *enmsCCTable* attributes

# 7

## Retrieving and Receiving Alarms

### 7.1 Approaches for Monitoring Alarms

TNMS alarms can be monitored via SNMP NBI by listening to alarm notifications or by polling the alarm tables.

#### 7.1.1 Listening to alarm notifications

In this approach, the manager listens for alarm notifications sent by SNMP NBI.

SNMP NBI sends alarm raise/clear notifications in one of five different traps, depending on the object that originated the alarm:

- *enmsNEAlarmTrap*
- *enmsModuleAlarmTrap*
- *enmsPortAlarmTrap*
- *enmsTPAlarmTrap*
- *enmsEMSAAlarmTrap*

Each trap above carries a set of variables describing the alarm, such as the identifier of the affected network object, alarm cause, severity, etc. (see section 7.2 for details).



If the severity variable (*enmsTrapEventSeverity*) is set to zero, it means the alarm has been cleared.

Example of module alarm:

```
enmsModuleAlarmTrap {  
    enmsTrapCounter = 12345,  
    enmsMoNEId = 50,  
    enmsMoModuleId = 9876,  
    enmsTrapEventDetails = "",  
    enmsTrapEventSeverity = 4 (critical),  
    enmsTrapEventProbableCause = 30 (CardFailure),  
    enmsAlClass = 4 (equipment),  
    enmsAlState = 3 (unacknowledged),  
    enmsAlTimeStamp = "2015-09-10 22:35:01",  
    enmsAlEntityString = "CARD 001",  
}
```

```

enmsTrapEventProbableCauseString = "Card Failure (CFail)",
enmsNeName = "NETEST1",
enmsTrapNeLocationLct = "",
enmsTrapNeIdName = "NE Test 1",
enmsTrapAffectedLocation = "",
enmsTrapEventTrafficDirection = (0) unknown,
enmsAlServiceAffect = true,
enmsAlAdditionalInformation = "",
enmsAlNeSystemContainer = "",
enmsAlNeNativeLocation = "",
enmsAlAlarmDirectionTelcordia = (0) unknown
}

```

The *enmsTrapEventProbableCause* variable indicates the cause of the alarm. Possible values are defined in the SNMP NBI MIB file, in the 'ProbableCause' enumeration. Excerpt:

```

ProbableCause ::= TEXTUAL-CONVENTION
STATUS      current
DESCRIPTION "Enumeration of alarm probable causes."
SYNTAX      INTEGER
{
    unknown (0),
    acd (1),
    ...
    cardFailure (30),
    ...
    los (132),
    ...
}

```

As an example, if the manager is monitoring Loss of Signal (los) alarms, then it should react whenever it receives an alarm trap with the *enmsTrapEventProbableCause* variable set to 132.

To identify which object the alarm has been raised/cleared for, the manager can use the object ID attributes – in the previous example, those are *enmsMoNEId* and *enmsMoModuleId*. Alternatively, it may use the *enmsAlEntityString* variable, which contains a human-readable string describing the object.

To deal with notification loss, the manager can periodically resynchronize all alarms by retrieving the global alarm table – see next section.

### 7.1.2 Polling the alarm tables

In the polling approach, the manager periodically retrieves all rows in the global alarm table (*enmsAlarmTable*), then processes each entry as for the alarm notification. If a previous alarm is no longer in the table, it means the alarm has been cleared.

Depending on alarm monitoring goals, instead of *enmsAlarmTable* it may be a better option to poll more specialized alarm tables such as *enmsAlarmsForPortConnTable* or *enmsAlarmsForSNCTable*.

## 7.2 Alarm Tables

### 7.2.1 List of all active alarms (*enmsAlarmTable*)

Table *enmsAlarmTable* contains all active alarms in TNMS. Its single index field (*enmsAIAlarmNumber*) corresponds to the current position of the alarm in the list and may change between table retrievals. For that reason, this table is not meant to be accessed randomly via GET operations. Instead, it must be retrieved row by row from top to bottom, using GETNEXT / GETBULK operations.

The following combination of fields may be used to identify an alarm and relate it to other tables and traps:

*enmsAIProbableCause* +  
*enmsAITimeStamp* +  
*enmsAIEntityString* +  
*enmsAINEId* +  
*enmsAIPortId* +  
*enmsAITPIdH* +  
*enmsAITPIdL*



System alarms (that is, raised by the EMS itself) can be distinguished by having the NE Id (*enmsAINEId*) set to zero.

Attribute name	Data type	Description
<u><i>enmsAIAlarmNumber</i></u>	Integer32	Table index. Does not identify the alarm and may change each time the table is retrieved.
<i>enmsAISeverity</i>	PerceivedSeverity	Severity of the alarm.
<i>enmsAIProbableCause</i>	ProbableCause	Probable Cause of the alarm.
<i>enmsAIClass</i>	AlarmClass	Class of the alarm.
<i>enmsAIServiceAffect</i>	Boolean	Indicates whether the alarm is traffic affecting.

Attribute name	Data type	Description
<b>enmsAIState</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsAITimeStampFromNE</b>	Boolean	Indicates whether the timestamp has been generated by the NE or by TNMS.
<b>enmsAITimeStamp</b>	EnmsTimeStamp	Alarm raising timestamp.
<b>enmsAIEntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsAIEntityType</b>	EntityType	Type of alarm originating entity.
<b>enmsAINEId</b>	NEId	NE Id of the originating entity.
<b>enmsAIPortId</b>	PortId	Port Id of the originating entity, if applicable.
<b>enmsAITPIdH</b>	TPId	Higher 32 bits of TP Id of originating entity, if applicable.
<b>enmsAITPIdL</b>	TPId	Lower 32 bits of TP Id of originating entity, if applicable.
<b>enmsAITPName</b>	DisplayString	Name of the originating TP, if applicable.
<b>enmsAIModuleId</b>	ModuleId	Id of the originating module, if applicable.
<b>enmsAIProbableCauseString</b>	DisplayString	Probable Cause in text form.
<b>enmsAINELocation</b>	DisplayString	NE location of originating entity, as reported by the NE itself.
<b>enmsAIAffectedLocation</b>	DisplayString	Extended textual location of the originating entity. For example, for alarms originating in TPs this field contains the chain "Module location / Port location / TP location". Only applicable to alarms originating in modules, ports or TPs.
<b>enmsAITrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsAIAdditionalInformation</b>	DisplayString	Optional additional information.

Attribute name	Data type	Description
<b>enmsAIneSystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsAINativeLocation</b>	DisplayString	Optional native object location in the network element.
<b>enmsAITrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 54 List of *enmsAlarmTable* attributes

### 7.2.2 List of alarms for NEs (enmsAlarmsForNETable)

Table *enmsAlarmsForNETable* contains all alarms originating in an NE and its modules, ports and TPs. Its indexes allow retrieving alarms for a selected NE.

Attribute name	Data type	Description
<b><u>enmsA2NEId</u></b>	NEId	NE Id of the originating entity (table index).
<b><u>enmsA2Severity</u></b>	PerceivedSeverity	Severity of the alarm (table index).
<b><u>enmsA2AlarmNumber</u></b>	Integer32	Additional table index. Does not identify the alarm and may change each time the table is retrieved.
<b>enmsA2ProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsA2Class</b>	AlarmClass	Class of the alarm.
<b>enmsA2ServiceAffect</b>	Boolean	Indicates whether the alarm affects a service.
<b>enmsA2State</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsA2TimeStampFromNE</b>	Boolean	Indicates whether the timestamp has been generated by the NE or by TNMS.
<b>enmsA2TimeStamp</b>	EnmsTimeStamp	Alarm raising timestamp.
<b>enmsA2EntityString</b>	DisplayString	Description of the alarm originating entity.

Attribute name	Data type	Description
<b>enmsA2EntityType</b>	EntityType	Type of alarm originating entity.
<b>enmsA2PortId</b>	PortId	Port Id of the originating entity, if applicable.
<b>enmsA2TPIdH</b>	TPId	Higher 32 bits of TP Id of originating entity, if applicable.
<b>enmsA2TPIdL</b>	TPId	Lower 32 bits of TP Id of originating entity, if applicable.
<b>enmsA2TPName</b>	DisplayString	Name of the originating TP, if applicable.
<b>enmsA2ModuleId</b>	ModuleId	Id of the originating module, if applicable.
<b>enmsA2ProbableCauseString</b>	DisplayString	Probable Cause in text form.
<b>enmsA2NELocation</b>	DisplayString	NE location of originating entity, as reported by the NE itself.
<b>enmsA2AffectedLocation</b>	DisplayString	Extended textual location of the originating entity. For example, for alarms originating in TPs this field contains the chain "Module location / Port location / TP location". Only applicable to alarms originating in modules, ports or TPs.
<b>enmsA2TrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsA2AdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsA2NeSystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsA2NativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsA2TrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 55 List of *enmsAlarmsForNETable* attributes

### 7.2.3 List of alarms for Ports (enmsAlarmsForPortTable)

Table *enmsAlarmsForPortTable* contains all alarms originating in a port or a TP contained in it. Its indexes allow retrieving the alarms for a selected port.

Attribute name	Data type	Description
<u>enmsA3NEId</u>	NEId	NE Id of the originating entity (table index).
<u>enmsA3PortId</u>	PortId	Port Id of the originating entity (table index).
<u>enmsA3Severity</u>	PerceivedSeverity	Severity of the alarm (table index).
<u>enmsA3AlarmNumber</u>	Integer32	Additional table index. Does not identify the alarm and may change each time the table is retrieved.
enmsA3ProbableCause	ProbableCause	Probable Cause of the alarm.
enmsA3Class	AlarmClass	Class of the alarm.
enmsA3ServiceAffect	Boolean	Indicates whether the alarm is traffic affecting.
enmsA3State	AlarmState	Indicates whether the alarm has been acknowledged.
enmsA3TimeStampFromNE	Boolean	Indicates whether the timestamp has been generated by the NE or by TNMS.
enmsA3TimeStamp	EnmsTimeStamp	Alarm raising timestamp.
enmsA3EntityString	DisplayString	Description of the alarm originating entity.
enmsA3EntityType	EntityType	Type of alarm originating entity.
enmsA3TPIdH	TPId	Higher 32 bits of TP Id of originating entity, if applicable.
enmsA3TPIdL	TPId	Lower 32 bits of TP Id of originating entity, if applicable.
enmsA3TPName	DisplayString	Name of the originating TP, if applicable.



Attribute name	Data type	Description
<b>enmsA3ProbableCauseString</b>	DisplayString	Probable Cause in text form.
<b>enmsA3NELocation</b>	DisplayString	NE location of originating entity, as reported by the NE itself.
<b>enmsA3AffectedLocation</b>	DisplayString	Extended textual location of the originating entity. For alarms originating in ports, this field contains the chain "Module location / Port location".
<b>enmsA3TrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsA3AdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsA3NeSystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsA3NativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsA3TrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 56 List of *enmsAlarmsForPortTable* attributes

#### 7.2.4 List of alarms for TP (*enmsAlarmsForTPTable*)

Table *enmsAlarmsForTPTable* contains all alarms originating in a TP. Its indexes allow retrieving the alarms for a selected TP.

Attribute name	Data type	Description
<b><u>enmsA4NEId</u></b>	NEId	NE Id of the originating entity (table index).
<b><u>enmsA4PortId</u></b>	PortId	Port Id of the originating entity (table index).
<b><u>enmsA4TPIdH</u></b>	TPId	Higher 32 bits of TP Id of originating entity (table index).
<b><u>enmsA4TPIdL</u></b>	TPId	Lower 32 bits of TP Id of originating entity (table index).

Attribute name	Data type	Description
<b><u>enmsA4Severity</u></b>	PerceivedSeverity	Severity of the alarm (table index).
<b><u>enmsA4AlarmNumber</u></b>	Integer32	Additional table index. Does not identify the alarm and may change each time the table is retrieved.
<b>enmsA4ProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsA4Class</b>	AlarmClass	Class of the alarm.
<b>enmsA4ServiceAffect</b>	Boolean	Indicates whether the alarm is traffic affecting.
<b>enmsA4State</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsA4TimeStampFromNE</b>	Boolean	Indicates whether the timestamp has been generated by the NE or by TNMS.
<b>enmsA4TimeStamp</b>	EnmsTimeStamp	Alarm raising timestamp.
<b>enmsA4EntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsA4EntityType</b>	EntityType	Type of alarm originating entity.
<b>enmsA4TPName</b>	DisplayString	Name of the originating TP, if applicable.
<b>enmsA4ProbableCauseString</b>	DisplayString	Probable Cause in text form.
<b>enmsA4NELocation</b>	DisplayString	NE location of originating entity, as reported by the NE itself.
<b>enmsA4AffectedLocation</b>	DisplayString	Extended textual location of the originating entity. For alarms originating in TPs, this field contains the chain "Module location / Port location / TP location".
<b>enmsA4TrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsA4AdditionalInformation</b>	DisplayString	Optional additional information.

Attribute name	Data type	Description
<b>enmsA4NeSystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsA4NativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsA4TrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 57 List of *enmsAlarmsForTPTTable* attributes

### 7.2.5 List of alarms for Port Connections (*enmsAlarmsForPortConnTable*)

This table contains all alarms affecting port connections. Its indexes allow retrieving the alarms for a selected port connection.



Alarms for internal port connections are only exposed if internal physical trail alarm correlation is activated in TNMS System Preferences.

Attribute name	Data type	Description
<b><u>enmsA5PortConnId</u></b>	PortConnId	Id of the Port Connection affected by the alarm (table index).
<b><u>enmsA5Severity</u></b>	PerceivedSeverity	Severity of the alarm (table index).
<b><u>enmsA5AlarmNumber</u></b>	Integer32	Additional table index. Does not identify the alarm and may change each time the table is retrieved.
<b>enmsA5ProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsA5Class</b>	AlarmClass	Class of the alarm.
<b>enmsA5ServiceAffect</b>	Boolean	Indicates whether the alarm is traffic affecting.
<b>enmsA5State</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsA5TimeStampFromNE</b>	Boolean	Indicates whether the timestamp has been generated by the NE or by TNMS.
<b>enmsA5TimeStamp</b>	EnmsTimeStamp	Alarm raising timestamp.

Attribute name	Data type	Description
<b>enmsA5EntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsA5EntityType</b>	EntityType	Type of alarm originating entity.
<b>enmsA5NEId</b>	NEId	NE Id of the originating entity.
<b>enmsA5PortId</b>	PortId	Port Id of the originating entity.
<b>enmsA5TPIdH</b>	TPId	Not applicable. Obsolete.
<b>enmsA5TPIdL</b>	TPId	Not applicable. Obsolete.
<b>enmsA5TPName</b>	DisplayString	Not applicable. Obsolete.
<b>enmsA5ModuleId</b>	ModuleId	Id of the originating module, if applicable.
<b>enmsA5ProbableCauseString</b>	DisplayString	Probable Cause in text form.
<b>enmsA5NELocation</b>	DisplayString	NE location of originating entity, as reported by the NE itself.
<b>enmsA5AffectedLocation</b>	DisplayString	Extended textual location of the originating entity. For example, for alarms originating in TPs this field contains the chain "Module location / Port location / TP location". Only applicable to alarms originating in modules, ports or TPs.
<b>enmsA5TrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsA5AdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsA5NativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsA5TrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 58 List of *enmsAlarmsForPortConnTable* attributes

### 7.2.6 List of alarms for Modules (enmsAlarmsForModuleTable)

This table contains all alarms originating in a module. Its indexes allow retrieving the alarms for a selected module.

Attribute name	Data type	Description
<u>enmsA8NEId</u>	NEId	NEId of the originating entity (table index).
<u>enmsA8ModuleId</u>	ModuleId	ModuleId of the originating entity (table index).
<u>enmsA8Severity</u>	PerceivedSeverity	Severity of the alarm (table index).
<u>enmsA8AlarmNumber</u>	Integer32	Additional table index. Does not identify the alarm and may change each time the table is retrieved.
enmsA8ProbableCause	ProbableCause	Probable Cause of the alarm.
enmsA8Class	AlarmClass	Class of the alarm.
enmsA8ServiceAffect	Boolean	Indicates whether the alarm is traffic affecting.
enmsA8State	AlarmState	Indicates whether the alarm has been acknowledged.
enmsA8TimeStampFromNE	Boolean	Indicates whether the timestamp has been generated by the NE or by TNMS.
enmsA8TimeStamp	EnmsTimeStamp	Alarm raising timestamp.
enmsA8EntityString	DisplayString	Description of the alarm originating entity.
enmsA8EntityType	EntityType	Type of alarm originating entity.
enmsA8PortId	PortId	Not applicable. Obsolete.
enmsA8TPIdH	TPId	Not applicable. Obsolete.
enmsA8TPIdL	TPId	Not applicable. Obsolete.
enmsA8TPName	DisplayString	Not applicable. Obsolete.
enmsA8ProbableCauseString	DisplayString	Probable Cause in text form.

Attribute name	Data type	Description
<b>enmsA8NELocation</b>	DisplayString	NE location of originating entity, as reported by the NE itself.
<b>enmsA8AffectedLocation</b>	DisplayString	Extended textual location of the originating entity. For modules, this field only contains the module location.
<b>enmsA8TrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsA8AdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsA8NeSystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsA8NativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsA8TrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 59 List of *enmsAlarmsForModuleTable* attributes

### 7.2.7 List of alarms for SNCs (*enmsAlarmsForSNCTable*)

This table contains all alarms affecting SNCs. Its indexes allow retrieving the alarms for a selected SNC.



This table only associates the SNCs to alarms directly affecting those SNCs – it will not associate alarms affecting the server SNCs.

Attribute name	Data type	Description
<b><u>enmsA6SNCTid</u></b>	SNCTid	Id of the SNC affected by the alarm (table index).
<b><u>enmsA6Severity</u></b>	PerceivedSeverity	Severity of the alarm (table index).
<b><u>enmsA6AlarmNumber</u></b>	Integer32	Additional table index. Does not identify the alarm and may change each time the table is retrieved.
<b>enmsA6ProbableCause</b>	ProbableCause	Probable Cause of the alarm.

Attribute name	Data type	Description
<b>enmsA6Class</b>	AlarmClass	Class of the alarm.
<b>enmsA6ServiceAffect</b>	Boolean	Indicates whether the alarm is traffic affecting.
<b>enmsA6State</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsA6TimeStampFromNE</b>	Boolean	Indicates whether the timestamp has been generated by the NE or by TNMS.
<b>enmsA6TimeStamp</b>	EnmsTimeStamp	Alarm raising timestamp.
<b>enmsA6EntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsA6EntityType</b>	EntityType	Type of alarm originating entity.
<b>enmsA6NEId</b>	NEId	NE Id of the originating entity.
<b>enmsA6PortId</b>	PortId	Port Id of the originating entity.
<b>enmsA6TPIdH</b>	TPId	Not applicable. Obsolete.
<b>enmsA6TPIdL</b>	TPId	Not applicable. Obsolete.
<b>enmsA6TPName</b>	DisplayString	Not applicable. Obsolete.
<b>enmsA6ModuleId</b>	ModuleId	Id of the originating module, if applicable.
<b>enmsA6ProbableCauseString</b>	DisplayString	Probable Cause in text form.
<b>enmsA6NELocation</b>	DisplayString	NE location of originating entity, as reported by the NE itself.
<b>enmsA6AffectedLocation</b>	DisplayString	Extended textual description of the alarm originating entity.
<b>enmsA6TrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsA6AdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsA6NativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.

Attribute name	Data type	Description
<b>enmsA9TrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 60 List of *enmsAlarmsForSNCTable* attributes

### 7.2.8 List of EMS alarms (*enmsAlarmsForEMSTable*)

Table *enmsAlarmsForEMSTable* contains all alarms affecting TNMS itself, that is, not originated in the NEs. Its single index field (*enmsA10AlarmNumber*) corresponds to the current position of the alarm in the list and may change between table retrievals.

Attribute name	Data type	Description
<b><u>enmsA10AlarmNumber</u></b>	Integer32	Table index. Does not identify the alarm and may change each time the table is retrieved.
<b>enmsA10Severity</b>	PerceivedSeverity	Severity of the alarm.
<b>enmsA10ProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsA10Class</b>	AlarmClass	Class of the alarm.
<b>enmsA10State</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsA10TimeStamp</b>	EnmsTimeStamp	Alarm raising timestamp.
<b>enmsA10EntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsA10EntityType</b>	EntityType	Type of alarm originating entity.
<b>enmsA10ProbableCauseString</b>	DisplayString	Probable Cause in text form.
<b>enmsA10AdditionalInformation</b>	DisplayString	Optional additional information.

Table 61 List of *enmsAlarmForEMSTable* attributes



## 7.3 Alarm Notifications

### 7.3.1 Notification of NE alarm (enmsNEAlarmTrap)

Notification sent when an NE alarm is raised or cleared, or when the alarm is acknowledged or unacknowledged.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsNeNEId</b>	NEId	Global NE identifier.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Severity of the alarm, or 'cleared' if alarm is cleared.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsAlClass</b>	AlarmClass	Alarm class.
<b>enmsAlState</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsAlTimeStamp</b>	EnmsTimeStamp	Alarm raise/clear timestamp.
<b>enmsAlEntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsTrapEventProbableCauseString</b>	DisplayString	Probable Cause of the alarm in string format.
<b>enmsNeName</b>	DisplayString	NE name, as reported by the NE.
<b>enmsTrapNeLocationLct</b>	DisplayString	NE location, as reported by the NE.
<b>enmsTrapNeIdName</b>	DisplayString	NE name, as specified by the operator in TNMS.
<b>enmsTrapAffectedLocation</b>	DisplayString	Empty.
<b>enmsTrapEventTrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsAlServiceAffect</b>	Boolean	Indicates whether the alarm is traffic affecting.

Variable name	Data type	Description
<b>enmsAIAdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsAINameSystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsAINativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsAITrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 62 List of *enmsNEAlarmTrap* attributes

### 7.3.2 Notification of Module alarm (enmsModuleAlarmTrap)

Notification sent when a module alarm is raised or cleared, or when the alarm is acknowledged or unacknowledged.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsMoNEId</b>	NEId	Global NE identifier.
<b>enmsMoModuleId</b>	ModuleId	Module identifier.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Severity of the alarm, or 'cleared' if alarm is cleared.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsAIClass</b>	AlarmClass	Class of the alarm.
<b>enmsAIState</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsAITimeStamp</b>	EnmsTimeStamp	Alarm raise/clear timestamp.
<b>enmsAIEntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsTrapEventProbableCauseString</b>	DisplayString	Probable Cause of the alarm in string format.

Variable name	Data type	Description
<b>enmsNeName</b>	DisplayString	NE name, as reported by the NE.
<b>enmsTrapNeLocationLct</b>	DisplayString	NE location, as reported by the NE.
<b>enmsTrapNeIdName</b>	DisplayString	NE name, as specified by the operator in the DCN properties.
<b>enmsTrapAffectedLocation</b>	DisplayString	Extended textual location of the originating entity in the format. For module alarms it only contains the module location.
<b>enmsTrapEventTrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsAIServiceAffect</b>	Boolean	Indicates whether the alarm is traffic affecting.
<b>enmsAIAdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsAINeSystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsAINativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsAITrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 63 List of *enmsModuleAlarmTrap* attributes

### 7.3.3 Notification of Port alarm (enmsPortAlarmTrap)

Notification sent when a port alarm is raised or cleared, or when the alarm is acknowledged or unacknowledged.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsPtNEId</b>	NEId	Global NE identifier.
<b>enmsPtPortId</b>	PortId	Port identifier.

Variable name	Data type	Description
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Severity of the alarm, or 'cleared' if alarm is cleared.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsAIClass</b>	AlarmClass	(to be supported)
<b>enmsAIState</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsAITimeStamp</b>	EnmsTimeStamp	Alarm raise/clear timestamp.
<b>enmsAIEntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsTrapEventProbableCauseString</b>	DisplayString	Probable Cause of the alarm in string format.
<b>enmsNeName</b>	DisplayString	NE name, as reported by the NE.
<b>enmsTrapNeLocationLct</b>	DisplayString	NE location, as reported by the NE.
<b>enmsTrapNeldName</b>	DisplayString	NE name, as specified by the operator in TNMS.
<b>enmsTrapAffectedLocation</b>	DisplayString	Extended textual location of the originating entity. For port alarms it contains the chain "Module location / Port location".
<b>enmsTrapEventTrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsAIServiceAffect</b>	Boolean	Indicates whether the alarm is traffic affecting.
<b>enmsAIAdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsAINESystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsAINativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsAITrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 64 List of *enmsPortAlarmTrap* attributes

### 7.3.4 Notification of TP alarm (enmsTPAlarmTrap)

Notification sent when a TP alarm is raised or cleared, or when the alarm is acknowledged or unacknowledged.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsTpNEId</b>	NEId	Global NE identifier.
<b>enmsTpPortId</b>	PortId	Port identifier.
<b>enmsTpTPIdH</b>	TPId	Higher 32 bits of TP Id.
<b>enmsTpTPIdL</b>	TPId	Lower 32 bits of TP Id.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Severity of the alarm, or 'cleared' if alarm is cleared.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsAlClass</b>	AlarmClass	(to be supported)
<b>enmsAlState</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsAlTimeStamp</b>	EnmsTimeStamp	Alarm raise/clear timestamp.
<b>enmsAlEntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsTrapEventProbableCauseString</b>	DisplayString	Probable Cause of the alarm in string format.
<b>enmsNeName</b>	DisplayString	NE name, as reported by the NE.
<b>enmsTrapNeLocationLct</b>	DisplayString	NE location, as reported by the NE.
<b>enmsTrapNeIdName</b>	DisplayString	NE name, as specified by the operator in TNMS.
<b>enmsTrapAffectedLocation</b>	DisplayString	Extended textual location of the originating entity. For TP alarms this field contains the chain "Module location / Port location / TP location".

Variable name	Data type	Description
<b>enmsTrapEventTrafficDirection</b>	TrafficDirection	Affected traffic direction.
<b>enmsAIServiceAffect</b>	Boolean	Indicates whether the alarm is traffic affecting.
<b>enmsAIAdditionalInformation</b>	DisplayString	Optional additional information.
<b>enmsAIInSystemContainer</b>	DisplayString	System container name of the originating NE.
<b>enmsAINativeLocation</b>	DisplayString	Optional native object location of the originating entity in the network element.
<b>enmsAITrafficDirectionTelcordia</b>	TrafficDirectionTelcordia	Affected traffic direction (Telcordia). Only applicable to TL1 originating entities.

Table 65 List of *enmsTPAlarmTrap* attributes

### 7.3.5 Notification of EMS alarm (enmsEMSAlarmTrap)

Notification sent when an EMS alarm is raised or cleared, or when the alarm is acknowledged or unacknowledged.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsTrapEventSeverity</b>	PerceivedSeverity	Severity of the alarm.
<b>enmsTrapEventProbableCause</b>	ProbableCause	Probable Cause of the alarm.
<b>enmsAIClass</b>	AlarmClass	(to be supported)
<b>enmsAIState</b>	AlarmState	Indicates whether the alarm has been acknowledged.
<b>enmsAITimeStamp</b>	EnmsTimeStamp	Alarm raising timestamp.
<b>enmsAIEntityString</b>	DisplayString	Description of the alarm originating entity.
<b>enmsTrapEventProbableCauseString</b>	DisplayString	Probable Cause of the alarm in string format.

Variable name	Data type	Description
enmsAIAdditionalInformation	DisplayString	Optional additional information.

Table 66 List of *enmsEMSAAlarmTrap* attributes

### 7.3.6 Notifications on alarm acknowledgement

By default, when an alarm is acknowledged or unacknowledged, TRNBI resends a notification for that alarm but with updated *enmsAIState* value.

It is possible to disable notifications on alarm acknowledgment by editing the SNMP NBI properties file, which can be found in the following location on the server installation:

```
<Product_Installation_Folder>  
/server/bicnet/deployments/bicnet.ear/conf/snmpNbi.properties
```

To disable the acknowledge/unacknowledged notifications, add or change the property below and set its value to 0:

```
resendAlarmOnAcknowledge=0
```

Set the value to 1 to re-enable the notifications:

```
resendAlarmOnAcknowledge=1
```

Restart server for changes to take effect.

## 8

## Reading and Setting Agent Parameters via SNMP

### 8.1 Agent Parameters (EnmsControl)

The *enmsControl* branch contains several variables with information about the SNMP NBI component itself.

Attribute name	Data type	R/W	Description
<b>enmsProxyName</b>	DisplayString	R/W	Proxy name, as defined in the SNMP NBI preferences (see 3.1).
<b>enmsProxyOpState</b>	OperationalState	R	Operational state. Always 'enabled', otherwise not possible to retrieve the value.
<b>enmsNetworkName</b>	DisplayString	R/W	Network name, as defined in the SNMP NBI preferences (see 3.1).
<b>enmsTrapHistoryTableLength</b>	Integer32	R/W	Maximum length of the trap history table (see 4.4).
<b>enmsTrapCounter</b>	Counter32	R	Trap counter of the last trap sent. Resets to zero when TNMS is restarted. See also 4.4.
<b>enmsProxyPSTAMP</b>	DisplayString	R	Production stamp of the SNMP NBI component.
<b>enmsEnterpriseld</b>	OID	R	OID of the enterprise node of the SNMP NBI.
<b>enmsMIBVersion</b>	DisplayString	R	Version of the SNMP NBI MIB. Matches the LAST-UPDATE clause in the MIB definition file.
<b>enmsEMSVersion</b>	DisplayString	R	Version of the TNMS installation.
<b>enmsTimeStampFormat</b>	DisplayString	R	Format of the timestamp values. Currently returns always "yyyy-MM-dd HH:mm:ss".



Attribute name	Data type	R/W	Description
<b>enmsInformTimeout</b>	Integer32	R/W	Maximum number of seconds that SNMP NBI will wait for an Inform response before retrying. Allowed values are between 1 and 60 seconds.
<b>enmsInformMaxTries</b>	Integer32	R/W	Maximum number of times that SNMP NBI will try to send an Inform notification to each destination. Allowed values are between 1 and 5 tries.
<b>enmsHeartbeatOpState</b>	OperationalState	R/W	Indicates whether heartbeat notifications are enabled. Valid values: 'enabled' or 'disabled'.
<b>enmsHeartbeatInterval</b>	Integer32	R/W	Heartbeat interval in seconds. Allowed values are between 5 and 86400 seconds.

Table 67 List of *enmsControl* variables

## 8.2 SNMP Agent Notifications

### 8.2.1 Notification of agent state change (*enmsProxyStateChangeTrap*)

Notification sent when the operational state of the SNMP NBI agent changes.

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsProxyName</b>	DisplayString	Agent name, as specified in the properties.
<b>enmsTrapEventDetails</b>	DisplayString	Empty.
<b>enmsProxyOpState</b>	OperationalState	Agent operational state.

Table 68 List of *enmsProxyStateChangeTrap* attributes

## 8.3 Notification Filtering (*enmsTrapFilter*)

The *enmsTrapFilter* branch contains variables that reflect the state of the notification filters of the SNMP user accessing them (see 3.2.5). These

variables are writable – the SNMP manager is able to activate and deactivate notifications for the SNMP user via SET operations.

Variable name	Data type	Description
<b>enmsCommonTrapFilter</b>	TrapFilter	Indicates whether common notifications are enabled.
<b>enmsNETrapFilter</b>	TrapFilter	Indicates whether NE notifications (except alarms) are enabled.
<b>enmsModuleTrapFilter</b>	TrapFilter	Indicates whether Module notifications (except alarms) are enabled.
<b>enmsPortTrapFilter</b>	TrapFilter	Indicates whether Port notifications (except alarms) are enabled.
<b>enmsTPTrapFilter</b>	TrapFilter	Indicates whether TP notifications (except alarms) are enabled.
<b>enmsPortConnTrapFilter</b>	TrapFilter	Indicates whether Port Connection notifications (except alarms) are enabled.
<b>enmsSNCTrapFilter</b>	TrapFilter	Indicates whether SNC notifications are enabled.
<b>enmsServiceTrapFilter</b>	TrapFilter	Indicates whether Service notifications are enabled.
<b>enmsNEAlarmTrapFilter</b>	TrapFilter	Indicates whether NE alarm notifications are enabled.
<b>enmsModuleAlarmTrapFilter</b>	TrapFilter	Indicates whether Module alarm notifications are enabled.
<b>enmsPortAlarmTrapFilter</b>	TrapFilter	Indicates whether Port alarm notifications are enabled.
<b>enmsTPAlarmTrapFilter</b>	TrapFilter	Indicates whether TP alarm notifications are enabled.
<b>enmsEMSAAlarmTrapFilter</b>	TrapFilter	Indicates whether EMS alarm notifications are enabled.
<b>enmsMonitorTrapFilter</b>	TrapFilter	Indicates whether monitor event notifications are enabled.
<b>enmsEthernetPathTrapFilter</b>	TrapFilter	Indicates whether Ethernet Path notifications are enabled.
<b>enmsPerfMonTrapFilter</b>	TrapFilter	Indicates whether Performance Monitoring notifications are enabled.

Variable name	Data type	Description
<b>enmsOptPowerMonTrapFilter</b>	TrapFilter	Indicates whether Optical Power Monitoring notifications are enabled.

Table 69 List of enmsTrapFilter variables

## 8.4 Notification History (enmsTrapHistoryTable)

Table *enmsTrapHistoryTable* contains information about the last notifications sent by SNMP NBI. It is mainly used for data resynchronization in case of missing notifications caused by network errors (see 4.4).

Attribute name	Data type	Description
<b>enmsHiTrapNumber (index)</b>	Integer32	Index for the table. Has no correlation with the trap counter.
<b>enmsHiTrapEntityType</b>	EntityType	Type of entity associated to the trap.
<b>enmsHiTrapFirstId</b>	Uniqueld	First ID of the entity associated to the trap (NEId / PortConnId / SNCLId / EthernetPathId).
<b>enmsHiTrapSecondId</b>	Uniqueld	Second ID of the entity associated to trap (ModuleId, PortId).
<b>enmsHiTrapTPIdH</b>	TPId	Higher 32 bits of TP Id, if applicable.
<b>enmsHiTrapTPIdL</b>	TPId	Lower 32 bits of TP Id, if applicable.
<b>enmsHiTrapNfyType</b>	NotificationType	Type of notification (OC, OD, Alarm, etc.).
<b>enmsHiTrapCounter</b>	Counter32	Trap counter, as has been sent in the trap.

Table 70 List of *enmsTrapHistoryTable* attributes

# 9

## Ethernet Paths – Support of MEF 40

### 9.1 MEF-UNI-EVC-MIB

TNMS SNMP NBI provides preliminary support for the MEF MIB for the management of User Network Interfaces (UNIs) and Ethernet Virtual Connections (EVCs). Only basic Ethernet Path retrieval is supported.



To monitor changes to Ethernet Paths, use SNMP NBI MIB notifications (see section 6.2).

For more information on the MEF MIB, please refer to the MEF 40 Technical Specification available on the MEF web site ([www.mef.net](http://www.mef.net)).

#### 9.1.1 mefServiceEvcCfgTable

The 'mefServiceEvcCfgTable' table lists all the Ethernet Paths. This table implementation is read-only, therefore adding rows is not supported.

Attribute name	Data type	Notes
<b>mefServiceEvcCfgIndex</b>	Unsigned32	Table index.
<b>mefServiceEvcCfgIdentifier</b>	DisplayString	
<b>mefServiceEvcCfgServiceType</b>	INTEGER	
<b>mefServiceEvcCfgMtuSize</b>	Unsigned32	(to be supported)
<b>mefServiceEvcCfgCevlanIdPreservation</b>	MefServicePreservationType	
<b>mefServiceEvcCfgCevlanCosPreservation</b>	MefServicePreservationType	
<b>mefServiceEvcCfgUnicastDelivery</b>	MefServiceDeliveryType	(to be supported)
<b>mefServiceEvcCfgMulticastDelivery</b>	MefServiceDeliveryType	(to be supported)
<b>mefServiceEvcCfgBroadcastDelivery</b>	MefServiceDeliveryType	(to be supported)
<b>mefServiceEvcCfgL2cpGrpIndex</b>	Unsigned32	(to be supported)
<b>mefServiceEvcCfgAdminState</b>	EntityAdminState	

Attribute name	Data type	Notes
mefServiceEvcCfgRowStatus	RowStatus	Not used (table is read-only)

Table 71 mefServiceEvcCfgTable attributes

### 9.1.2 mefServiceEvcStatusTable

The 'mefServiceEvcStatusTable' table shows the status of the Ethernet Paths. It has a 1:1 relation with 'mefServiceEvcCfgTable'.

Attribute name	Data type	Notes
mefServiceEvcCfgIndex	Unsigned32	
mefServiceEvcStatusMaxMtuSize	Unsigned32	(to be supported)
mefServiceEvcStatusMaxNumUni	Unsigned32	(to be supported)
mefServiceEvcStatusOperationalState	INTEGER	

Table 72 mefServiceEvcStatusTable attributes

# 10 Performance Monitoring

SNMP NBI allows managers to retrieve Performance Monitoring (PM) data associated to the network objects managed by TNMS:

- History data
- Current data
- PMP and threshold information



Retrieving history PM data via SNMP NBI requires PM data upload to be enabled in TNMS Server. Refer to TNMS documentation for information on how to enable PM data upload.

Given its nature and potentially large volume, PM data is not readily available in an MIB table. To retrieve PM data, the SNMP manager creates a request by inserting a row in the PM request table. Each request specifies the type of data to retrieve and for which network objects. The request is then executed, after which the resulting PM data may be retrieved.

The following diagrams exemplify the high level interaction between a manager wanting to retrieve PM data and SNMP NBI. In Figure 11, the manager creates a request and waits for a notification indicating that the PM data is ready for retrieval:

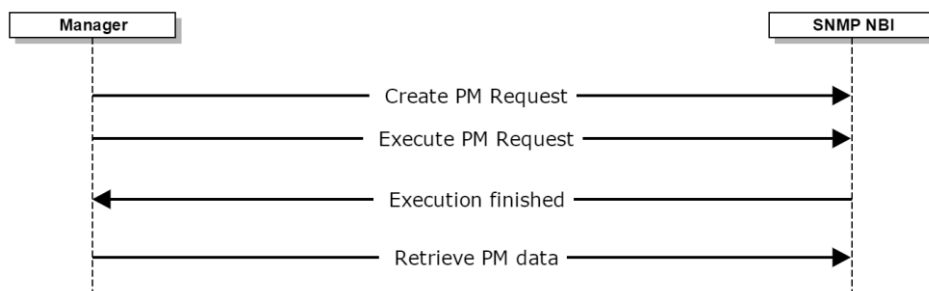


Figure 11 Example of PM data retrieval using notifications

Alternatively, the manager may poll the state of the request (instead of waiting for a notification):

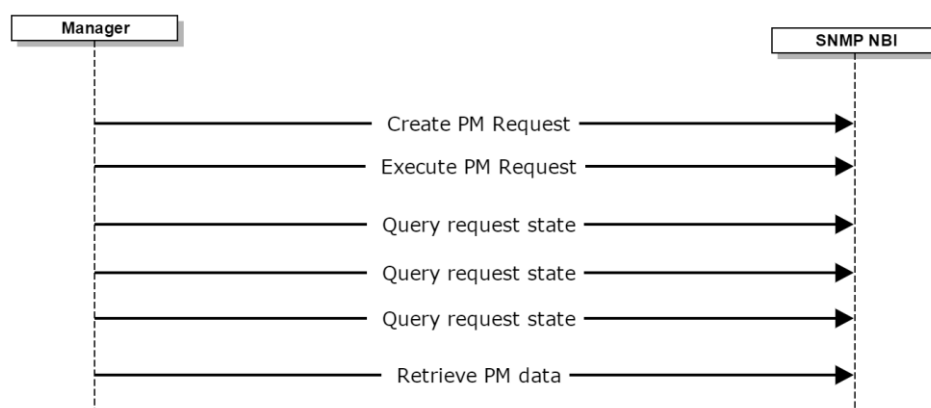


Figure 12 Example of PM data retrieval using polling

## 10.1 PM Requests

PM requests are entries of the MIB table *enmsPerfMonRequestTable*.

Each PM request specifies the type of data to retrieve and for which network objects.

### 10.1.1 PM request table (enmsPerfMonRequestTable)

This table contains all PM requests created by the SNMP manager.

Attribute name	Data type	Settable?	Description
<b><u>enmsPmRequestId</u></b>	PerfMonRequestId	No	Id of the request (table index).
<b>enmsPmRequestName</b>	DisplayString	Yes	Optional request name, for manager reference. Default is empty.
<b>enmsPmRequestRowStatus</b>	RowStatus	Yes (see descr.)	Standard SNMP RowStatus field for controlling row creation.  Set to <i>createAndGo</i> (4) to create a new row, or <i>destroy</i> (6) to remove an existing row. Other values are not supported.
<b>enmsPmRequestState</b>	PerfMonRequestState	Yes (see descr.)	State of the PM request. Set this field to change the state of the request.  Not settable at row creation.
<b>enmsPmRequestLastUpdate</b>	EnmsTimeStamp	No	Time (in UTC) of the last update of the request.
<b>enmsPmRequestInfo</b>	DisplayString	No	Information about request status.

Attribute name	Data type	Settable?	Description
<b>enmsPmRequestType</b>	PerfMonType	Yes	Type of PM data to retrieve: <ul style="list-style-type: none"> <li>- <i>pmHistory</i> (1) – retrieve history data</li> <li>- <i>pmCurrent</i> (2) – retrieve current data</li> <li>- <i>pmPoints</i> (3) – retrieve PM point info and associated thresholds.</li> </ul> Default is <i>pmCurrent</i> .
<b>enmsPmRequestStartTime</b>	EnmsTimeStamp	Yes	For history PM data, start time (in UTC) of the collection period. Default is empty.
<b>enmsPmRequestEndTime</b>	EnmsTimeStamp	Yes	For history PM data, end time (in UTC) of the collection period. Default is empty.
<b>enmsPmRequestGranularity</b>	PerfMonGranularity	Yes	Granularity of PM data: <ul style="list-style-type: none"> <li>- <i>minutes15</i> (1)</li> <li>- <i>hours24</i> (2)</li> </ul> Default is <i>minutes15</i> .
<b>enmsPmRequestFilterType</b>	FilterType	Yes	Type of object for which to retrieve PM data: <ul style="list-style-type: none"> <li>- <i>tpObject</i> (1)</li> <li>- <i>portObject</i> (2)</li> <li>- <i>neObject</i> (3)</li> <li>- <i>sncObject</i> (4)</li> <li>- <i>ethernetPathObject</i> (5)</li> <li>- <i>moduleObject</i> (6)</li> <li>- <i>equipHolderObject</i> (7)</li> </ul> (!) Filtering by NE object is unsupported for request type <i>pmCurrent</i> . Default is <i>sncObject</i> .



Attribute name	Data type	Settable?	Description
<b>enmsPmRequestFilterValue</b>	DisplayString	Yes	<p>Identifier of the object for which to retrieve PM data. The identifier of an object is the index of that object in the corresponding MIB table, with the individual index values separated by " " (pipe character). Examples:</p> <p>TP (enmsTPTable): 173 455 3453 99589454</p> <p>Port (enmsPortTable): 32 6734</p> <p>NE (enmsNETable) - history only 85</p> <p>SNC (enmsSNCTable): 8374</p> <p>Ethernet Path (enmsSNCTable): 2387</p> <p>Module (enmsModuleTable): 32 9334</p> <p>Port (enmsEquipHolderTable): 32 277</p> <p>Specify multiple objects by separating their identifiers with commas. Example for port objects: 32 6734,55 33928</p> <p>Default is 0.</p>

Table 73 List of *enmsPerfMonRequestTable* attributes

## 10.2 Creating a PM Request

Creating a PM request (adding a row to the MIB table *enmsPerfMonRequestTable*) follows the method defined in the SNMP RFC 2579, which uses a *RowStatus* attribute to control the row existence. The manager must perform the following steps:

- 1) Obtain an instance identifier for the new row, by reading the *enmsPmRequestNextId* leaf attribute using SNMP GET (this variable auto-increments with each GET access).
- 2) Using the new instance identifier, send an SNMP SET command to set *enmsPmRequestRowStatus* to *createAndGo* and optionally assign values to other attributes.

If the SNMP SET command succeeds, a new entry is added to *enmsPerfMonRequestTable*. If it fails, no entry will be added. Reasons for failure are listed below.

Error cause	SNMP error
- The provided instance identifier is already in use.	<i>inconsistentValue</i>

Error cause	SNMP error
<ul style="list-style-type: none"> <li>- Attribute <i>enmsPmRequestRowStatus</i> set to <i>notReady</i>, <i>active</i>, <i>notInService</i> or <i>destroy</i>.</li> <li>- An attribute has been assigned a value inconsistent with other attributes. Examples: <ul style="list-style-type: none"> <li>o The request type was set to <i>history</i>, but the start/end times are empty;</li> <li>o The filter type was set to <i>port</i>, but the filter value contains a TP identifier.</li> </ul> </li> </ul>	<i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Attribute <i>enmsPmRequestRowStatus</i> not set.</li> </ul>	<i>inconsistentName</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Attribute <i>enmsPmRequestRowStatus</i> set to <i>createAndWait</i>.</li> <li>- Wrong type or invalid value assigned to an attribute.</li> </ul>	<i>wrongValue</i> <i>badValue</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Values assigned to non-settable attributes.</li> </ul>	<i>notWritable</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Maximum number of requests exceeded.</li> </ul>	<i>resourceUnavailable</i> <i>commitFailed</i>

Table 74 Possible errors during PM request creation.

For new requests, *enmsPmRequestRowStatus* is set to *active*, and *enmsPmRequestState* to *created*. The *enmsPmRequestId* index field is set to the supplied instance identifier. Other fields left unset are assigned default values.

Figure 13 exemplifies the creation of a new PM request:

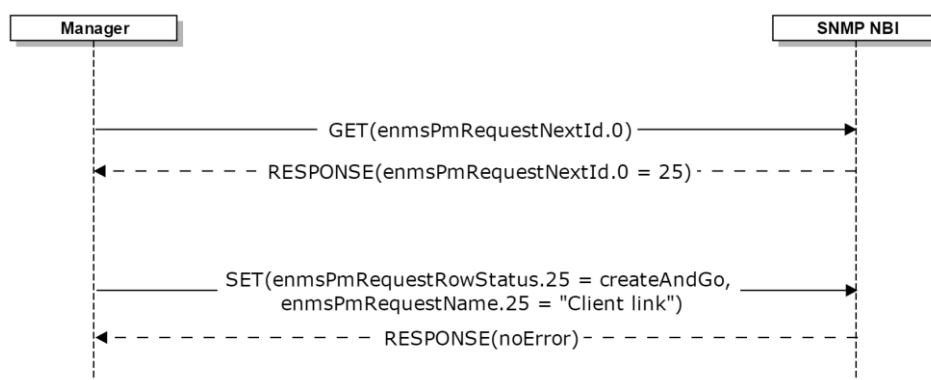


Figure 13 Example of interaction to create PM request

After the interaction above, *enmsPerfMonRequestTable* will contain a new entry for instance 25:

Id	Name	Row Status	State	Last Update	Info	Type	Start Time	End Time	Granul.	Filter Type	Filter Value
11	Access ports	active	finished	2016-03-20 14:01:12	Execution finished.	pmCurrent			minutes15	Port	123 98
19	Middle node	active	started	2016-03-20 14:01:12	Execution started.	pmHistory	2016-03-01 00:00:00	2016-03-15 00:00:00	hours24	SNC	320
25	Client link	active	created	2016-03-22 14:55:35	Request created.	pmCurrent			minutes15	SNC	0

Table 75 Example of *enmsPerfMonRequestTable* after adding a request

New requests are in the state *created*, meaning that the request exists but is not in execution.



Maximum number of PM requests is 5000. After this limit is reached, the manager must delete existing PM requests before adding new ones. Alternatively, the manager may reuse existing PM requests by updating its fields.

## 10.3 PM Request States

Table 76 lists the possible states of a PM request (reflected in the *enmsPmRequestState* attribute) and the actions the manager may perform for each state.

enmsPmRequestState	Description	Possible actions
<b>created</b>	Request is idle. This is the initial state of a newly created request.	<ul style="list-style-type: none"> <li>- Execute the request</li> <li>- Update the request</li> </ul>
<b>pending</b>	Request is awaiting execution.	<ul style="list-style-type: none"> <li>- Cancel the request</li> </ul>
<b>started</b>	Request is being executed.	<ul style="list-style-type: none"> <li>- Cancel the request</li> </ul>
<b>finished</b>	Request finished successfully. PM data is available for retrieving.	<ul style="list-style-type: none"> <li>- Retrieve the PM data</li> <li>- Update the request</li> <li>- Re-execute the request</li> <li>- Discard the PM data</li> </ul>
<b>failed</b>	Request failed and is idle.	<ul style="list-style-type: none"> <li>- Update the request</li> <li>- Re-execute the request</li> </ul>
<b>cancelling</b>	Request is being cancelled.	<ul style="list-style-type: none"> <li>- None</li> </ul>
<b>cancelled</b>	Request has been cancelled and is idle.	<ul style="list-style-type: none"> <li>- Update the request</li> <li>- Re-execute the request</li> </ul>

Table 76 States of a PM request and possible actions

The manager performs actions on a request by setting the *enmsPmRequestState* attribute using the SNMP SET operation (see section 10.5).

Figure 14 summarizes the main transitions between request states.

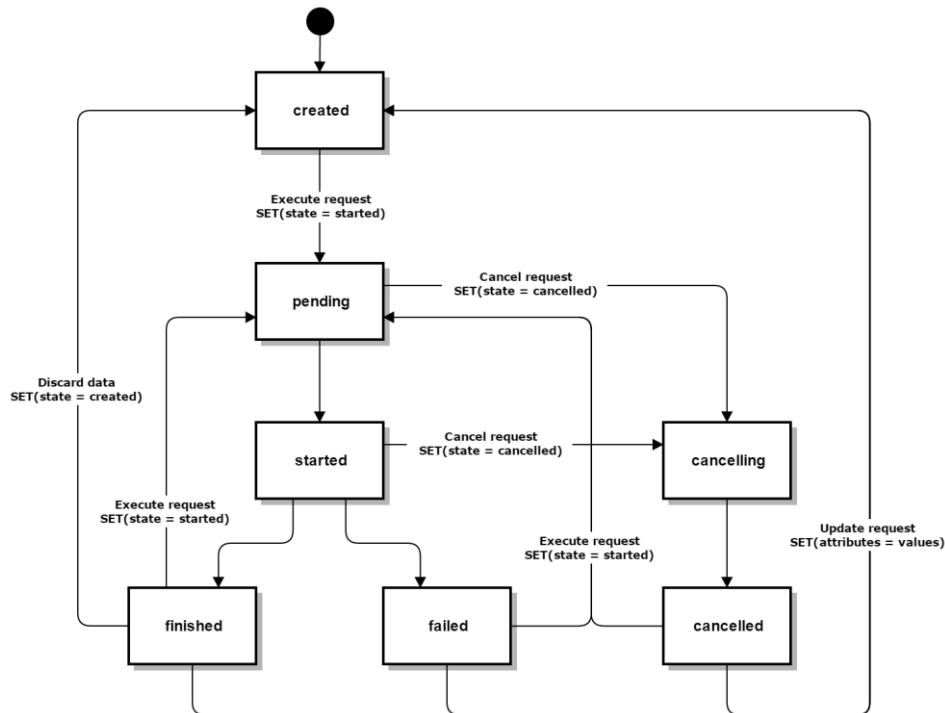


Figure 14 PM request states and main transitions

## 10.4 PM Request State Change Notifications

Whenever the state of a PM request changes, SNMP NBI sends an *enmsPerfMonRequestStateChangeTrap* notification to the managers. This notification carries the following attributes:

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsPmRequestId</b>	PerfMonRequestId	Id of the request.
<b>enmsPmRequestName</b>	DisplayString	Name of the request.
<b>enmsPmRequestState</b>	PerfMonRequestState	New state of the request.
<b>enmsPmRequestInfo</b>	DisplayString	Information about request status.

Table 77 List of *enmsPerfMonRequestStateChangeTrap* attributes

## 10.5 Actions on PM Requests

Actions on PM requests are performed by issuing SNMP SET commands to change the PM request state and other attributes.

### 10.5.1 Executing a PM request

To execute a PM request, the manager sends an SNMP SET command to set *enmsPmRequestState* to *started*. The state will first change to *pending* (the request has been queued for execution). When execution starts, it will change to *started*.



Executing a PM request is only possible if the request is in an idle state (either *created*, *finished*, *failed* or *cancelled*).



Figure 15 Example of interaction for executing a PM request

The manager can also update the PM request attributes (see 10.5.2) and execute it in a single SET command:

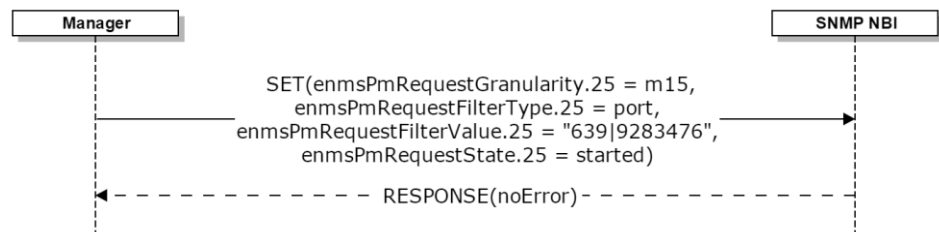


Figure 16 Updating and executing a PM request in a single SET command

If the request is executed successfully, the state will change to *finished*. The manager may retrieve the PM data (see section 10.6). If the request execution fails, the state will change to *failed*. The *enmsPmRequestInfo* field provides a hint on what caused the failure. Typical failure reasons include:

- Network objects for which to obtain PM data do not exist;
- A timeout occurred while collecting data from the NEs;
- An internal server error occurred.

In any case, the manager may update the PM request and re-execute it.



Executing a PM request in state *finished* causes associated PM data to be discarded.

### 10.5.2 Updating PM request attributes

When a PM request is created, unset attributes are assigned default values. Before executing the PM request, the manager must set those attributes with valid values, otherwise the PM request execution will fail. The manager may also want to reuse a previously executed PM request, or correct the attributes of a failed PM request.

To update the attributes of a PM request, the manager sends an SNMP SET. Multiple attributes can be set in a single SET command.



Figure 17 Example of interaction for updating a PM request

When assigning an invalid value to some attribute, the SET command fails with the error code *inconsistentValue*.

Updating a PM request moves it to state *created*, and any associated PM data is discarded.

It is also possible to update the PM request attributes and execute it at the same time, in a single SET command (see 10.5.1).



Updating a PM request is only possible if the request is in an idle state (either *created*, *finished*, *failed* or *cancelled*).



Updating a PM request in state *finished* causes associated PM data to be discarded.

### 10.5.3 Cancelling a PM request

To cancel the execution of a PM request, the manager sends an SNMP SET command to set *enmsPmRequestState* to *cancelled*. The state changes to *cancelling*, and then to *cancelled*.



Figure 18 Example of interaction for cancelling a PM request



Cancelling a PM request is only possible if the request is in state *pending* or *started*.

#### 10.5.4 Discarding PM data associated to a PM request

After a PM request is successfully executed, resulting PM data is preserved until the request is re-executed or deleted.

To keep the request for future re-execution, the manager may discard the data by sending an SNMP SET command to set *enmsPmRequestState* to *created*. This way, resources are freed up from the server, while the PM request is kept for reuse.



Figure 19 Example of interaction for discarding PM data of a PM request

#### 10.5.5 Deleting a PM request

To delete a PM request, the manager sends an SNMP SET command to set *enmsPmRequestRowStatus* to *destroy*.

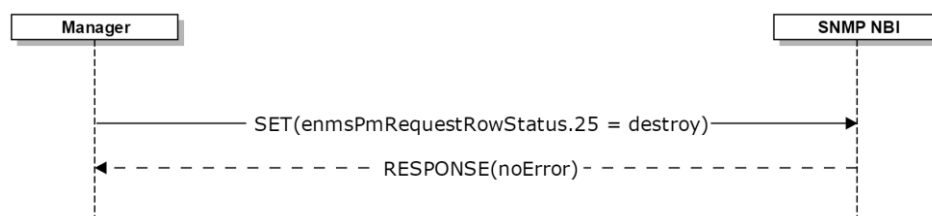




Figure 20 Example of interaction for deleting a PM request

-  Deleting a request is only possible if the request is in an idle state (either *created*, *finished*, *failed* or *cancelled*).
-  Deleting a PM request also deletes associated PM data.

### 10.5.6 Error exceptions

Table 78 describes common errors returned by SNMP NBI while performing SNMP SET operations on existing PM requests:

Error cause	SNMP error
<ul style="list-style-type: none"> <li>- Attribute <i>enmsPmRequestState</i> set an invalid value given the current request state (see 10.3).</li> <li>- An attribute was assigned an inconsistent value. Examples: <ul style="list-style-type: none"> <li>o The request type was set to <i>history</i>, but the start/end times are empty;</li> <li>o The filter type was set to <i>port</i>, but the filter value contains a TP identifier.</li> </ul> </li> </ul>	<i>inconsistentValue</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Attribute <i>enmsPmRequestRowStatus</i> set to other value than <i>destroy</i>.</li> <li>- Wrong type or invalid value assigned to an attribute.</li> </ul>	<i>wrongValue</i> <i>badValue</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Values assigned to non-settable attributes.</li> </ul>	<i>notWritable</i> <i>commitFailed</i>

Table 78 Common errors while performing actions on PM requests

## 10.6 Retrieving PM Data

When the execution of a PM request finishes successfully, the PM request state changes to *finished*. The manager may now retrieve the associated PM data by accessing the following tables:

- *enmsPerfMonResultPmpTable*: Contains the PM Points of the request results (see 10.6.2).
- *enmsPerfMonResultValueTable*: For requests of type *pmHistory* and *pmCurrent*, contains the measured values for each PM Point (see 10.6.3).
- *enmsPerfMonResultThresholdTable*: For requests of type *pmPoints*, contains the threshold values for each PM Point (see 10.6.4).

These tables contain the PM data results for all finished requests. To retrieve data for a specific PM request, follow the method suggested in Section 4.5.3 (all tables above are indexed by PM request identifier).





Because of the potentially large volume of data, request only the strictly needed attributes to speed-up the retrieval operation.

### 10.6.1 PM data retention period

PM data is available for the following approximate times (after the PM request finishes):

- History data: 12 hours
- Current data: 1 hour
- PM points: 1 hour

Results older than the intervals above are periodically discarded and the associated requests are moved to the *created* state.



History data is only available as long as it exists in TNMS Server, which has its own retention period.

It is recommended that the manager forces the data to be discarded as soon as it finishes retrieving it (see section 10.5.4).

### 10.6.2 enmsPerfMonResultPmpTable

Table 79 contains the PM Points of all finished PM request results.

Attribute name	Data type	Description
<u>enmsPmResultPmpReqId</u>	PerfMonRequestId	PerfMon request identifier. (table index)
<u>enmsPmResultPmpPmpNumber</u>	Unsigned32	Number of the PMP in the result set. (table index)
enmsPmResultPmpNeId	NEId	NE Id of the PMP.
enmsPmResultPmpPortId	PortId	Port Id of the PMP.
enmsPmResultPmpTPIdH	TPId	Highest 32-bits of the TP Id of the PMP, if applicable.
enmsPmResultPmpTPIdL	TPId	Lowest 32-bits of the TP Id of the PMP, if applicable.
enmsPmResultPmpNeIdName	DisplayString	NE Id name of the NE of the PMP.
enmsPmResultPmpObjLocation	DisplayString	Object location of the PMP.
enmsPmResultPmpName	DisplayString	PMP name.

Attribute name	Data type	Description
<b>enmsPmResultPmpLocation</b>	PerfMonLocation	PMP location (near end/far end).
<b>enmsPmResultPmpDirection</b>	PerfMonDirection	PMP direction.
<b>enmsPmResultPmpRetrievalTime</b>	EnmsTimeStamp	Retrieval time.
<b>enmsPmResultPmpPeriodEndTime</b>	EnmsTimeStamp	End time of the collection period.
<b>enmsPmResultPmpMonitoredTime</b>	Unsigned32	Total monitored time, or zero if not supported by the NE.
<b>enmsPmResultPmpNumValues</b>	Unsigned32	Number of values collected for this PMP.
<b>enmsPmResultPmpRelatedPaths</b>	DisplayString	Names of the paths related to this PMP (comma-separated list, truncated to field size limits). Only applicable to history data, empty for other types of request. Requires correlation of PM data to paths to be enabled in the server.
<b>enmsPmResultPmpRelatedServices</b>	DisplayString	Names of the services with paths related to this PMP (comma-separated list, , truncated to field size limits). Only applicable to history data, empty for other types of request. Requires correlation of PM data to paths to be enabled in the server.
<b>enmsPmResultPmpRelatedSubscribers</b>	DisplayString	Names of the subscribers related to this PMP (comma-separated list, truncated to field size limits). Only applicable to history data, empty for other types of request. Requires correlation of PM data to paths to be enabled in the server.
<b>enmsPmResultPmpNativeLocation</b>	DisplayString	(to be supported) Optional native object location of the PMP in the network element.

Table 79 List of *enmsPerfMonResultPmpTable* attributes

### 10.6.3 enmsPerfMonResultValueTable

Table 80 contains the measured values for the PM Points of all finished PM request results, for requests of type *pmHistory* or *pmCurrent*.

Attribute name	Data type	Description
<u>enmsPmResultValReqId</u>	PerfMonRequestId	PM request identifier (table index).
<u>enmsPmResultValPmpNumber</u>	Unsigned32	Number of the PMP in the result set. (table index)
<u>enmsPmResultValNumber</u>	Unsigned32	Number of value in the collected values for the PMP. (table index)
enmsPmResultValParam	DisplayString	Parameter name.
enmsPmResultValValue	DisplayString	Parameter value.
enmsPmResultValUnit	DisplayString	Parameter unit.
enmsPmResultValStatus	PerfMonStatus	Status of the collected value.

Table 80 List of *enmsPerfMonResultValueTable* attributes

### 10.6.4 enmsPerfMonResultThresholdTable

Table 81 contains the threshold values for the PM Points of all finished PM request results, for requests of type *pmPoints* only.

Attribute name	Data type	Description
<u>enmsPmResultThresholdReqId</u>	PerfMonRequestId	PM request identifier (table index).
<u>enmsPmResultThresholdPmpNumber</u>	Unsigned32	Number of the PMP in the result set (table index).
<u>enmsPmResultThresholdNumber</u>	Unsigned32	Number of value in the collected values for the PMP. (table index)
enmsPmResultThresholdParam	DisplayString	Parameter name.
enmsPmResultThresholdType	PerfMonThresholdType	Threshold type.
enmsPmResultThresholdTriggerFlag	Boolean	Indicates if the threshold is for the trigger ( <i>true</i> ) or the clear ( <i>false</i> ).

Attribute name	Data type	Description
		Only valid if the NE supports separate values for trigger and clear.
<b>enmsPmResultThresholdValue</b>	DisplayString	Threshold value.
<b>enmsPmResultThresholdUnit</b>	DisplayString	Threshold unit.

Table 81 List of *enmsPerfMonResultThresholdTable* attributes

# 11

## Optical Power Monitoring

SNMP NBI allows managers to retrieve Optical Power Monitoring (OPM) data associated to network objects managed by TNMS.

Given its nature, OPM data is not readily available in an MIB table. To retrieve OPM data, the SNMP manager needs to create a request by inserting a row in the OPM request table. The request is then executed, after which the resulting OPM data may be retrieved.

The following diagrams exemplify the high level interaction between a manager wanting to retrieve OPM data and SNMP NBI. In Figure 21, the manager creates a request and waits for a notification indicating that the OPM data is ready for retrieval:

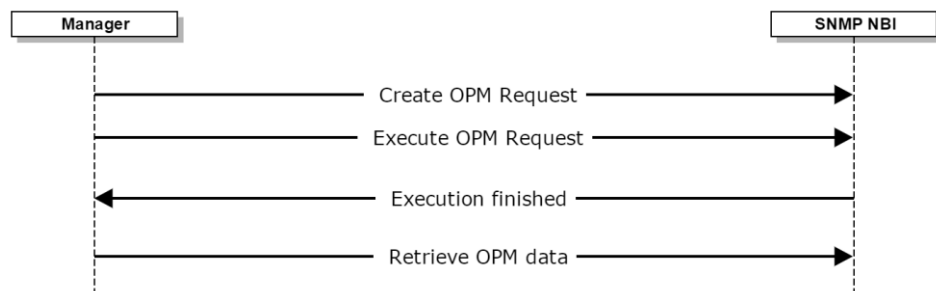


Figure 21 Example of OPM data retrieval using notifications

Alternatively, the manager may poll the state of the request instead of waiting for a notification:

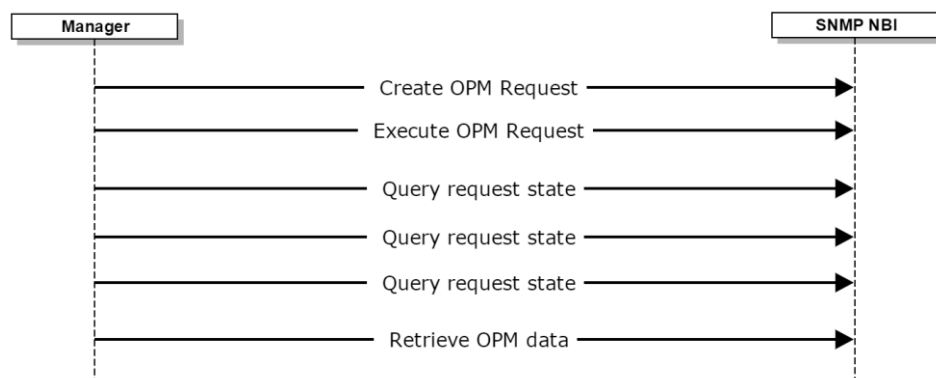


Figure 22 Example of OPM data retrieval using polling

## 11.1 OPM Requests

OPM requests are entries of the MIB table *enmsOptPowerMonRequestTable*. Each OPM request specifies the network objects data to retrieve data for.

### 11.1.1 OPM request table (enmsOptPowerMonRequestTable)

Table 82 contains all OPM requests created by the SNMP manager.

Attribute name	Data type	Settable?	Description
<b>enmsOpmRequestId</b>	OptPowerMonRequestId	No	Id of the request (table index).
<b>enmsOpmRequestName</b>	DisplayString	Yes	Optional request name, for manager reference. Default is empty.
<b>enmsOpmRequestRowStatus</b>	RowStatus	Yes (see description)	Standard SNMP RowStatus field for controlling row creation.  Set to <i>createAndGo</i> (4) to create a new row, or <i>destroy</i> (6) to remove an existing row. Other values are not supported.
<b>enmsOpmRequestState</b>	OptPowerMonRequestState	Yes (see description)	State of the OPM request. Set this field to change the state of the request.  Not settable at row creation.
<b>enmsOpmRequestLastUpdate</b>	EnmsTimeStamp	No	Time (in UTC) of the last update of the request.
<b>enmsOpmRequestInfo</b>	DisplayString	No	Information about request status.
<b>enmsOpmRequestFilterType</b>	FilterType	Yes	Type of object for which to retrieve OPM data (TP, Port or SCN): <ul style="list-style-type: none"> <li>- <i>tpObject</i> (1)</li> <li>- <i>portObject</i> (2)</li> <li>- <i>sncObject</i> (4)</li> </ul> <p>(!) Filtering by other types of object is unsupported. Default is <i>sncObject</i>.</p>

Attribute name	Data type	Settable?	Description
<b>enmsOpmRequestFilterValue</b>	DisplayString	Yes	<p>Identifier of the object for which to retrieve OPM data. The identifier of an object is the index of that object in the corresponding MIB table, with the individual index values separated by the pipe (" ") character. Examples:</p> <p>TP (enmsTPTable): 173   455   3453   99589454</p> <p>Port (enmsPortTable): 32   6734</p> <p>SNC (enmsSNCTable): 8374</p> <p>Specify multiple objects by separating their identifiers with commas. Example for port objects: 32   6734 , 55   33928</p> <p>Default is 0.</p>

Table 82 List of *enmsOptPowerMonRequestTable* attributes

## 11.2 Creating an OPM Request

Creating an OPM request (adding a row to the MIB table *enmsOptPowerMonRequestTable*) follows the method defined in the SNMP RFC 2579, which uses a *RowStatus* attribute to control the row existence. The manager must perform the following steps:

- 1) Obtain an instance identifier for the new row, by reading the *enmsOpmRequestNextId* leaf attribute using SNMP GET (this variable auto-increments with each GET access).
- 2) Using the new instance identifier, send an SNMP SET command to set *enmsOpmRequestRowStatus* to *createAndGo* and optionally assign values to other attributes.

A new entry is added to *enmsOptPowerMonRequestTable*. If it fails, no entry will be added. Reasons for failure are listed in the table below.

Error cause	SNMP error
<ul style="list-style-type: none"> <li>- The provided instance identifier is already in use.</li> <li>- Attribute <i>enmsPmRequestRowStatus</i> set to <i>notReady</i>, <i>active</i>, <i>notInService</i> or <i>destroy</i>.</li> <li>- An attribute was assigned an inconsistent value. For example, the filter type is <i>port</i>, but the filter value contains a TP identifier.</li> </ul>	<i>inconsistentValue</i>  <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Attribute <i>enmsPmRequestRowStatus</i> not set.</li> </ul>	<i>inconsistentName</i>  <i>commitFailed</i>

Error cause	SNMP error
<ul style="list-style-type: none"> <li>- Attribute <i>enmsOpmRequestRowStatus</i> set to <i>createAndWait</i>.</li> <li>- Wrong type or invalid value assigned to an attribute.</li> </ul>	<i>wrongValue</i> <i>badValue</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Values assigned to non-settable attributes.</li> </ul>	<i>notWritable</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Maximum number of requests exceeded.</li> </ul>	<i>resourceUnavailable</i> <i>commitFailed</i>

Table 83 Possible errors during OPM request creation.

New requests have *enmsOpmRequestRowStatus* set to *active*, and *enmsOpmRequestState* set to *created*. The *enmsOpmRequestId* index field is set to the supplied instance identifier. Other fields left unset will be assigned default values.

Figure 23 exemplifies the creation of a new OPM request:

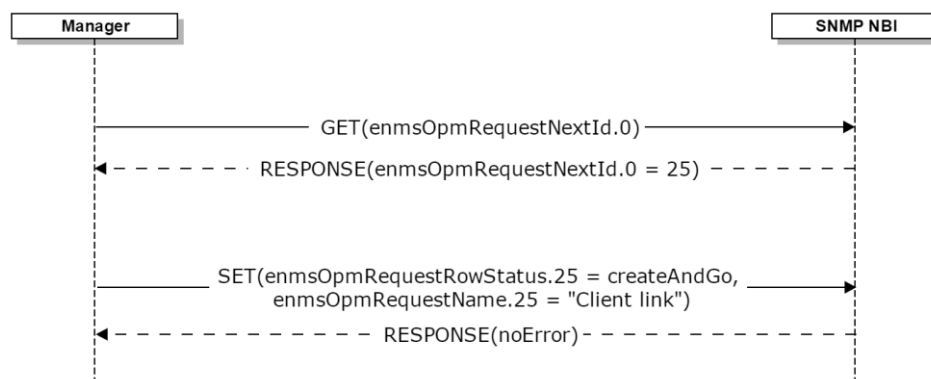


Figure 23 Example of interaction to create OPM request

After this interaction, *enmsOptPowerMonRequestTable* contain a new entry for instance 25:

Id	Name	Row Status	State	Last Update	Info	Filter Type	Filter Value
11	Access Port 1	active	finished	2016-03-20 14:01:12	Execution finished.	Port	123 98
19	Middle node	active	started	2016-03-20 14:01:12	Execution started.	SNC	320
25	Client link	active	created	2016-03-22 14:55:35	Request created.	SNC	0

Table 84 Example of *enmsOptPowerMonRequestTable* after adding a request



Created requests are in the state *created*, meaning the request exists but is not in execution.



Maximum number of OPM requests is 5000. After this limit is reached, the manager must delete existing OPM requests before adding new ones. Alternatively, the manager may reuse existing PM requests by updating its fields.

## 11.3 OPM Request States

Table 85 lists the possible states of an OPM request (reflected in the *enmsOpmRequestState* attribute) and the actions the manager may perform for each state.

<i>enmsOpmRequestState</i>	Meaning	Possible actions
<b>created</b>	Request is idle. This is the initial state of a newly created request.	<ul style="list-style-type: none"> <li>- Execute the request</li> <li>- Update the request</li> </ul>
<b>pending</b>	Request is awaiting execution.	<ul style="list-style-type: none"> <li>- Cancel the request</li> </ul>
<b>started</b>	Request is being executed.	<ul style="list-style-type: none"> <li>- Cancel the request</li> </ul>
<b>finished</b>	Request finished successfully. OPM data is available for retrieving.	<ul style="list-style-type: none"> <li>- Retrieve the OPM data</li> <li>- Update the request</li> <li>- Re-execute the request</li> <li>- Discard the OPM data</li> </ul>
<b>failed</b>	Request failed and is idle.	<ul style="list-style-type: none"> <li>- Update the request</li> <li>- Re-execute the request</li> </ul>
<b>cancelling</b>	Request is being cancelled.	<ul style="list-style-type: none"> <li>- None</li> </ul>
<b>cancelled</b>	Request has been cancelled and is idle.	<ul style="list-style-type: none"> <li>- Update the request</li> <li>- Re-execute the request</li> </ul>

Table 85 States of an OPM request and possible manager actions

The manager performs actions on PM requests by setting the *enmsOpmRequestState* attribute using the SNMP SET command (see section 11.5).

Figure 24 summarizes the main transitions between request states.

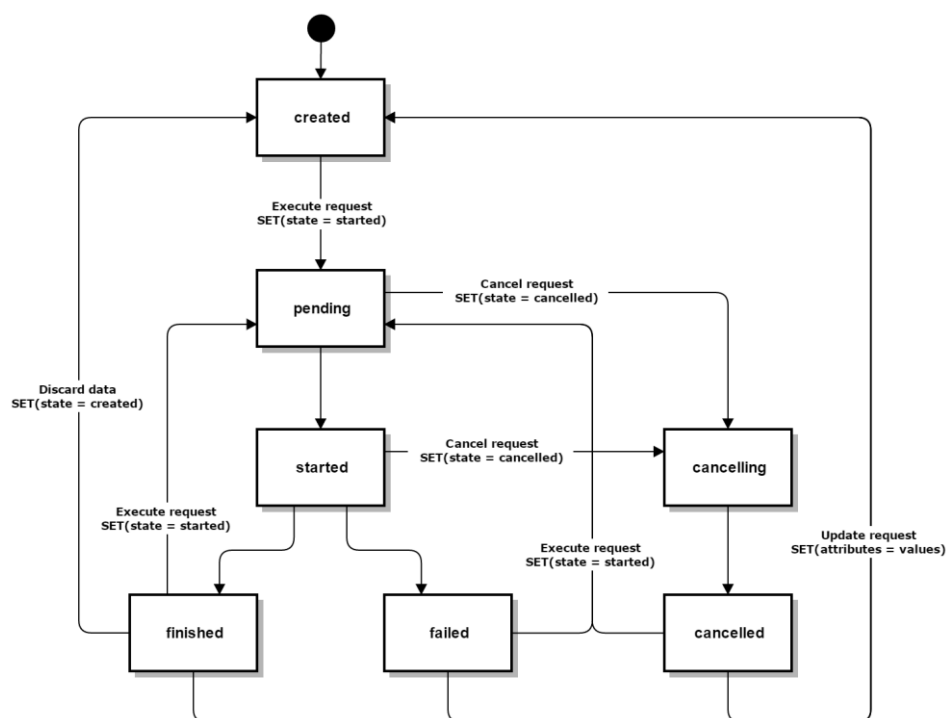


Figure 24 OPM request states and main transitions

## 11.4 OPM Request State Change Notifications

Whenever the state of an OPM request changes, SNMP NBI sends an *enmsOptPowerMonRequestStateChangeTrap* notification to the managers. This notification carries the following attributes:

Variable name	Data type	Description
<b>enmsTrapCounter</b>	Counter32	Trap counter for synchronization.
<b>enmsOpmRequestId</b>	OptPowerMonRequestId	Id of the request.
<b>enmsOpmRequestName</b>	DisplayString	Name of the request.
<b>enmsOpmRequestState</b>	OptPowerMonRequestState	New state of the request.
<b>enmsOpmRequestInfo</b>	DisplayString	Information about request status.

Table 86 List of *enmsOptPowerMonRequestStateChangeTrap* attributes

## 11.5 Actions on OPM Requests

Actions on OPM requests are performed by issuing SNMP SET commands to change the request state and other attributes.

### 11.5.1 Executing an OPM request

To execute an OPM request, the manager sends an SNMP SET command to set *enmsOpmRequestState* to *started*. The state will first change to *pending*, meaning that the request has been queued for execution. When the execution starts, it changes to *started*.



Executing an OPM request is only possible if the request is in an idle state (either *created*, *finished*, *failed* or *cancelled*).



Figure 25 Example of interaction for executing an OPM request

The manager can also update the OPM request attributes (see 11.5.2) and execute it in a single SET command:



Figure 26 Updating and executing an OPM request in a single SET command

If the request is executed successfully, the state will change to *finished*. The manager may retrieve the OPM data (see section 11.6). If the request execution fails, the state changes to *failed*. The *enmsOpmRequestInfo* field normally provides a hint on what caused the failure. Typical failure reasons include:

- Network objects for which to obtain OPM data do not exist.
- A timeout occurred while collecting data from the NEs.

- An internal server error occurred.

In any case, the manager may update the OPM request and re-execute it.



Executing an OPM request in state *finished* causes associated OPM data to be discarded.

### 11.5.2 Updating OPM request attributes

When an OPM request is created, unset attributes are assigned default values. Before executing the OPM request, the manager must set those attributes with valid values, otherwise the OPM request execution will fail. The manager may also want to reuse a previously executed OPM request, or correct the attributes of a failed OPM request.

To update the attributes of an OPM request, the manager may send an SNMP SET. Multiple attributes can be set in a single SET command.



Figure 27 Example of interaction for updating an OPM request

When trying to assign an invalid value to some attribute, the SET command fails with the error code *inconsistentValue*.

Updating an OPM request moves it to state *created*. Any associated OPM data is discarded.

It is also possible to update the OPM request attributes and execute it at the same time in a single SET command (see 11.5.1).



Updating an OPM request is only possible if the request is in an idle state (either *created*, *finished*, *failed* or *cancelled*).



Updating an OPM request in state *finished* causes associated OPM data to be discarded.

### 11.5.3 Cancelling an OPM request

To cancel the execution of an OPM request, the manager sends an SNMP SET command to set *enmsOpmRequestState* to *cancelled*. If the command is successful, the state changes to *cancelling*, and then to *cancelled*.

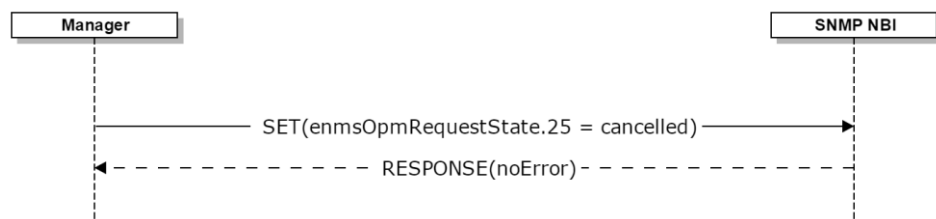


Figure 28 Example of interaction for cancelling an OPM request



Cancelling an OPM request is only possible if the request is in state *pending* or *started*.

#### 11.5.4 Discarding OPM data associated to an OPM request

After an OPM request is successfully executed, the resulting OPM data is preserved until the request is re-executed, or until the request is deleted.

To keep the request for future re-execution, the manager may discard the data by sending an SNMP SET command to set *enmsOpmRequestState* to *created*. This way, resources are freed up from the server, while the OPM request is kept for reuse.





Figure 29 Example of interaction for discarding OPM data of an OPM request

#### 11.5.5 Deleting an OPM request

To delete an OPM request, the manager sends an SNMP SET command to set *enmsOpmRequestRowStatus* to *destroy*.



Figure 30 Example of interaction for deleting an OPM request

-  Deleting a request is only possible if the request is in an idle state (either *created*, *finished*, *failed* or *cancelled*).
-  Deleting an OPM request also deletes associated OPM data.

### 11.5.6 Error exceptions

Table 87 describes the most common errors returned by SNMP NBI while performing SNMP SET commands on existing OPM requests.


Error cause	SNMP error
<ul style="list-style-type: none"> <li>- Attribute <i>enmsOpmRequestState</i> set to a value not allowed given the current request state (see 11.3).</li> <li>- An attribute was assigned an inconsistent value. For example, the filter type was set <i>port</i>, but the filter value contains a TP identifier.</li> </ul>	<i>inconsistentValue</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Attribute <i>enmsOpmRequestRowStatus</i> set to other value than <i>destroy</i>.</li> <li>- Wrong type or invalid value assigned to an attribute.</li> </ul>	<i>wrongValue</i> <i>badValue</i> <i>commitFailed</i>
<ul style="list-style-type: none"> <li>- Values assigned to non-settable attributes.</li> </ul>	<i>notWritable</i> <i>commitFailed</i>

Table 87 Common errors while performing actions on OPM requests

## 11.6 Retrieving OPM Data

When the execution of an OPM request finishes successfully, the OPM request state changes to *finished*. The manager may now retrieve associated OPM data by accessing the *enmsOptPowerMonResultValueTable* (see 11.6.2).

The mentioned tables contain the whole OPM data results for all finished requests. To retrieve the data for a specific OPM request, follow the method suggested in Section 4.5.3 (both tables are indexed by OPM request identifier).

-  Because of the potentially large volume of data, the manager should only request the strictly needed attributes to speed-up the retrieval operation.

### 11.6.1 OPM data retention period

OPM result data is available for 1 hour after the OPM request finishes. Results older than this interval are periodically discarded and the associated requests are moved to the *created* state.

It is recommended that the manager forces the data to be discarded as soon as it finishes retrieving it (see section 11.5.4).

### 11.6.2 enmsOptPowerMonResultTable

Table 88 contains the OPM Points of all finished OPM request results.

Attribute name	Data type	Description
<u>enmsOpmResultValReqId</u>	OptPowerMonRequestId	OptPowerMon request identifier. (table index)
<u>enmsOpmResultValPmpNumber</u>	Unsigned32	Number of the value in the result set of the OPM request. (table index)
<u>enmsOpmResultValNeId</u>	NEId	NE Id of the object.
<u>enmsOpmResultValPortId</u>	PortId	Port Id of the object.
<u>enmsOpmResultValTPIdH</u>	TPId	Highest 32-bits of the TP Id of the object, if applicable.
<u>enmsOpmResultValTPIdL</u>	TPId	Lowest 32-bits of the TP Id of the object, if applicable.
<u>enmsOpmResultValNeIdName</u>	DisplayString	NE Id name of the NE of the object.
<u>enmsOpmResultValObjLocation</u>	DisplayString	Object location.
<u>enmsOpmResultValLane</u>	Unsigned32	Lane of the OPM counter.
<u>enmsOpmResultValLayer</u>	DisplayString	Layer of the OPM counter.
<u>enmsOpmResultValParam</u>	DisplayString	OPM counter name.
<u>enmsOpmResultValValue</u>	DisplayString	OPM counter value.
<u>enmsOpmResultValUnit</u>	DisplayString	OPM counter unit.

Table 88 List of *enmsOptPowerMonResultPmpTable* attributes

# 12

## Net-SNMP Examples for PM Data Retrieval

Net-SNMP (<http://www.net-snmp.org>) is a set of command-line tools widely used to interact with SNMP agents. This chapter provides examples on how to use Net-SNMP to request PM data via TNMS SNMP NBI.

The workflow for retrieving PM data is described in Chapter 10 – read it first.

For simplicity, the NET-SNMP examples in this document omit common arguments for specifying the MIB, credentials and output formatting options. A full complete command may look like the following:

```
snmptable -m TNMS-NBI-MIB -M C:/Home/MIB -v 2c -c public tnmsserver enmsPerfMonRequestTable
```

Common arguments

Please see Net-SNMP documentation for details.



## 12.1 Creating a New PM Request

Get a unique index for the new PM request by reading *enmsPmRequestNextId*, which auto-increments each time it is accessed.

You may use any other index generation method instead of *enmsPmRequestNextId* – just make sure the chosen index is not in use in *enmsPerfMonRequestTable*.

```
snmpget enmsPmRequestNextId.0
```

```
TNMS-NBI-MIB::enmsPmRequestNextId.0 = Gauge32: 4
```

Use the selected index (4 in this example) to create a new PM request. *RowStatus* must be set to *createAndGo(4)*:

```
snmpset enmsPmRequestRowStatus.4 i createAndGo
enmsPmRequestName.4 s "Demo request"
enmsPmRequestType.4 i pmHistory
enmsPmRequestStartTime.4 s "2017-03-07 12:00:00"
enmsPmRequestEndTime.4 s "2017-03-07 17:00:00"
enmsPmRequestGranularity.4 i minutes15
enmsPmRequestFilterType.4 i ethernetPathObject
enmsPmRequestFilterValue.4 s "32"
```

```
TNMS-NBI-MIB::enmsPmRequestRowStatus.4 = INTEGER: createAndGo(4)
TNMS-NBI-MIB::enmsPmRequestName.4 = STRING: Demo request
TNMS-NBI-MIB::enmsPmRequestType.4 = INTEGER: pmHistory(1)
TNMS-NBI-MIB::enmsPmRequestStartTime.4 = STRING: "2017-03-07 12:00:00"
TNMS-NBI-MIB::enmsPmRequestEndTime.4 = STRING: "2017-03-07 17:00:00"
TNMS-NBI-MIB::enmsPmRequestGranularity.4 = INTEGER: minutes15(1)
TNMS-NBI-MIB::enmsPmRequestFilterType.4 = INTEGER: ethernetPathObject(5)
TNMS-NBI-MIB::enmsPmRequestFilterValue.4 = STRING: 32
```

Listing all PM requests to confirm new row:

```
snmptable enmsPerfMonRequestTable
```

```
SNMP table: TNMS-NBI-MIB::enmsPerfMonRequestTable
```

index	Name	RowStatus	LastUpdate	Info	State	Type	StartTime
1	ethernetCarrier	1	"2017-03-03 19:47:00"	Request reinitialized.	1	2	""
2	mpls_7090	1	"2017-03-07 08:36:43"	Request reinitialized.	1	1	"2017-03-06 20:00:00"
3		1	"2017-03-07 16:04:22"	Execution finished. Total values: 84.	4	1	"2017-03-07 13:43:31"
4	Demo request	1	"2017-03-07 16:35:18"	Request created.	1	1	"2017-03-07 12:00:00"

## 12.2 Executing the PM Request

Execute the request by setting *enmsPmRequestState* to *started*:

```
snmpset enmsPmRequestState.4 i started
```

```
TNMS-NBI-MIB::enmsPmRequestState.4 = INTEGER: started(3)
```

To check the execution state, read *enmsPmRequestState* and optionally *enmsPmRequestInfo*. In this example the execution is still pending:

```
snmpget enmsPmRequestState.4 enmsPmRequestInfo.4
```

```
TNMS-NBI-MIB::enmsPmRequestState.4 = INTEGER: pending(2)  
TNMS-NBI-MIB::enmsPmRequestInfo.4 = STRING: "Request queued for execution."
```

When the request execution finishes, the resulting data is ready for retrieval:

```
snmpget enmsPmRequestState.4 enmsPmRequestInfo.4
```

```
TNMS-NBI-MIB::enmsPmRequestState.4 = INTEGER: finished(4)  
TNMS-NBI-MIB::enmsPmRequestInfo.4 = STRING: "Execution finished. Total values: 156."
```

An existing PM request may be updated and re-executed any number of times, by setting *enmsPmRequestState* to *started* (see section 13212.4).

## 12.3 Retrieving the PM Request Results

To get the PM results, first retrieve resulting PMPs by reading the table *enmsPerfMonResultPmpTable*. Relevant rows will have *enmsPmResultPmpReqId* = 4, the PM request index.

**Note:** Net-SNMP does not offer an easy way to retrieve only the rows for a specific sub-index. The example below shows how to retrieve the entire table, which contains the results for all requests.

### snmptable enmsPerfMonResultPmpTable

SNMP table: TNMS-NBI-MIB::enmsPerfMonResultPmpTable

index	ReqId	PmpNumber	NeId	PortId	TPidH	TPidL	NeIdName	ObjLocation	Name	Location	Direction	RetrievalTime
PeriodEndTime												
(...)												
3.80	3	80	105	112010005	0	112010015	7090_100CEM_14	ge.1.5	7090M-RMON-RX		1	2 "2017-03-07 15:16:51" "
3.81	3	81	105	112010005	0	112010015	7090_100CEM_14	ge.1.5	7090M-RMON-RX		1	2 "2017-03-07 15:16:51" "
3.82	3	82	105	112010005	0	112010015	7090_100CEM_14	ge.1.5	7090M-RMON-RX		1	2 "2017-03-07 15:16:51" "
4.1	4	1	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.2	4	2	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.3	4	3	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.4	4	4	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.5	4	5	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.6	4	6	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.7	4	7	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.8	4	8	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.9	4	9	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.10	4	10	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.11	4	11	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.12	4	12	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
4.13	4	13	97	112030004	0	112030014	7090_320M_7	ge.3.4	7090M-RMON-RX		1	2 "2017-03-07 15:16:49" "
(...)												

Next, retrieve counter values for the resulting PMPs, by reading the table *enmsPerfMonResultValueTable*. Again, relevant rows will have *enmsPmResultValReqId* = 4. Fields *enmsPmResultValReqId* + *enmsPmResultValPmpNumber* point to the PMP entries in *enmsPerfMonResultPmpTable*.

### snmptable enmsPerfMonResultValueTable

SNMP table: TNMS-NBI-MIB::enmsPerfMonResultValueTable

index	ReqId	PmpNumber	Number	Param	Value	Unit	Status
(...)							
3.84.6	3	84	6	pOVERSIZE	0.0	1	
3.84.7	3	84	7	pBC	0.0	1	
3.84.8	3	84	8	pMC	876.0	1	
4.1.1	4	1	1	pOK	0.0	1	
4.1.2	4	1	2	pBAD	0.0	1	
4.1.3	4	1	3	oALL	0.0	1	
4.1.4	4	1	4	pUC	0.0	1	
4.1.5	4	1	5	fCSERR	0.0	1	
4.1.6	4	1	6	pOVERSIZE	0.0	1	
4.1.7	4	1	7	pBC	0.0	1	
4.1.8	4	1	8	pMC	0.0	1	
4.2.1	4	2	1	pOK	0.0	1	
4.2.2	4	2	2	pBAD	0.0	1	
4.2.3	4	2	3	oALL	0.0	1	
4.2.4	4	2	4	pUC	0.0	1	
(...)							

## 12.4 Re-executing the PM Request

To reuse a PM request, update its fields (typically the start/end times) and re-execute it by setting *enmsPmRequestState* again to *started*. The example in this chapter re-executes the previously created PM request, whose index is 4.

In the command below, the start/end times are set using relative times instead of absolute, using the ISO 8601 notation for durations. SNMP NBI automatically converts ISO 8601 durations to absolute timestamps by adding the duration to the current UTC time. Start time “-PT30m” will be converted to current UTC time minus 30 minutes, while end time “P0D” (which is a zero duration) will be converted to the current UTC time (now). This means we are requesting PM data for the last 30 minutes.

See [https://en.wikipedia.org/wiki/ISO\\_8601#Durations](https://en.wikipedia.org/wiki/ISO_8601#Durations) for duration syntax information.

**Note:** only days, hours and minutes can be used when specifying durations. Other granularities are not accepted.

```
snmpset enmsPmRequestStartTime.4 s "-PT30m"
        enmsPmRequestEndTime.4 s "P0D"
        enmsPmRequestState.4 i started
```

```
TNMS-NBI-MIB::enmsPmRequestStartTime.4 = STRING: "-PT30m"
TNMS-NBI-MIB::enmsPmRequestEndTime.4 = STRING: "P0D"
TNMS-NBI-MIB::enmsPmRequestState.4 = INTEGER: started(3)
```

Listing the PM requests, the supplied durations were converted to absolute timestamps:

```
snmptable enmsPerfMonRequestTable
```

SNMP table: TNMS-NBI-MIB::enmsPerfMonRequestTable

index	Name	RowStatus	LastUpdate	Info	State	Type	StartTime	EndTime	
1	ethernetCarrier	1	"2017-03-03 19:47:00"	Request reinitialized.	1	2	"	"	1
2	mpls 7090	1	"2017-03-07 08:36:43"	Request reinitialized.	1	1	"2017-03-06 20:00:00"	"2017-03-06 20:30:00"	1
3		1	"2017-03-07 16:04:22"	Execution finished. Total values: 84.	4	1	"2017-03-07 13:43:31"	"2017-03-07 15:03:31"	1
4	Demo request	1	"2017-03-07 18:29:24"	Execution finished. Total values: 23.	4	1	"2017-03-07 17:59:24"	"2017-03-07 18:29:24"	

## 12.5 Deleting a PM Request

Remove an unwanted PM request by setting *enmsPmRequestRowStatus* to *destroy(6)*:

```
snmpset enmsPmRequestRowStatus.4 i destroy
```

```
TNMS-NBI-MIB::enmsPmRequestRowStatus.4 = INTEGER: destroy(6)
```

Confirming that the PM request has been deleted:

```
snmpwalk enmsPerfMonRequestTable
```

```
SNMP table: TNMS-NBI-MIB::enmsPerfMonRequestTable
```

index	Name	RowStatus	LastUpdate	Info	State	Type	StartTime	EndTime	
1	ethernetCarrier	1	"2017-03-03 19:47:00"	Request reinitialized.	1	2	"	"	1
2	mpls_7090	1	"2017-03-07 08:36:43"	Request reinitialized.	1	1	"2017-03-06 20:00:00"	"2017-03-06 20:30:00"	1
3		1	"2017-03-07 16:04:22"	Execution finished. Total values: 84.	4	1	"2017-03-07 13:43:31"	"2017-03-07 15:03:31"	1

# 13

## Troubleshooting

The table below proposes solutions for the most common issues when operating with SNMP NBI. Also check TNMS System Event Log for messages related to SNMP NBI events.

Symptom	Possible cause	Solution
The SNMP NBI menu entries in TNMS Client are missing or greyed out.	SNMP NBI not installed in the server	Reinstall TNMS and select the SNMP northbound interface (see 2.5).
	SNMP NBI license not installed	Install SNMP NBI license (see 2.5).
The “Enable SNMP northbound interface” checkbox (SNMP NBI system settings) is greyed out.	An SNMP NBI license has been installed, but the server has not been restarted yet.	Restart TNMS server (see 2.5).
No response from SNMP NBI or timeout error.	SNMP NBI not installed or not licensed.	Install SNMP NBI or its license (see 2.5).
	Timeout configured on the SNMP manager is too low (see 0).	Increase the timeout value configured on the SNMP manager.
	Incorrect SNMP agent address configured on the SNMP manager.	Make sure that the SNMP agent address configured on the SNMP manager corresponds to the TNMS server machine.
	Incorrect SNMP agent port configured on the SNMP manager.	Make sure that the SNMP agent port configured on the SNMP manager matches the SNMP NBI listening port (see 3.1).
	The source address of the SNMP requests is not in the list of allowed manager addresses for the SNMP user.	Add the SNMP manager's IP address (or addresses, if it has multiple network interfaces) to the list of allowed manager addresses in the SNMP NBI user configuration (see 3.2.2).
	SNMP user doesn't have the appropriate permission.	Change permission of the SNMP user to 'Read' or 'Read/Write' as appropriate (see 3.2.2).
	Incorrect user data or SNMP protocol version configured on the SNMP manager.	Make sure the SNMP manager is using the correct user (SNMPv2 community or SNMPv3 user) and protocol version, as configured in the SNMP NBI user configuration (see 3.2.1).

Symptom	Possible cause	Solution
	SNMP NBI could not bind to the listening port.	Make sure that the configured listening port is not being used by any other service or application on the TNMS server machine (see 3.1). You may use a utility such as 'netstat' to list the ports on which the server computer is listening.
	Network connectivity problem.	Check network connectivity between the TNMS server machine and the SNMP manager machine.  Confirm that the ports chosen for SNMP communication, in particular, the SNMP NBI listening port, are not blocked by any firewall.
SNMP error “No such name” received.	The requested object doesn't exist in the MIB. Usually occurs with GET requests.	Check if the requested OID is valid and belongs to the SNMP NBI MIB.  Verify if the SNMP manager is trying to get a nonexistent table value (that is, the table is valid, but does not contain any value for the index specified in the OID).
	The SNMP manager is still using TNMS Core's SNMP Proxy MIB definition.	If the SNMP manager was previously configured to access TNMS Core's SNMP Proxy, then some adaptations are needed before redirecting it to the SNMP NBI. See 2.4.
	The SNMP manager is accessing the wrong SNMP agent (for example, a TNMS Core's SNMP Proxy installation).	Reconfigure the SNMP manager to access SNMP NBI instead.
SNMP error “Authentication error” received.	Incorrect user data or SNMP protocol version configured on the SNMP manager.	Make sure the SNMP manager is using the correct user authentication details, as configured in the SNMP NBI user configuration (see 3.2.1). This frequently occurs with SNMPv3, so confirm the user name, the authentication and privacy protocols, and the corresponding passwords.
SNMP error “Too big” received.	The response to the request does not fit in a single SNMP packet (see 4.5.4). Typically occurs when the SNMP manager requests too many OIDs in the same operation, or the max-repetitions value for a GETBULK operation is too high.	Split the failing GET / GETNEXT / GETNEXT operations into two or more requests.  Use a lower max-repetitions value for GETBULK requests.

Symptom	Possible cause	Solution
No traps/informs received from SNMP NBI.	SNMP manager address not added to the trap destination list.	Add the SNMP manager address to the trap/inform destination list of the appropriate SNMP NBI user (see 3.2.3).
	Incorrect trap destination port.	Make sure the destination port of the traps/informs matches the port on which the SNMP manager is waiting for traps (see 3.2.3).
	Incorrect SNMP user data.	Make sure the SNMPv2 community or SNMPv3 user for which the traps/informs are sent is configured in the SNMP manager.  For SNMPv3 users, check the user name, the authentication and privacy protocols, and the corresponding passwords.
	The SNMP manager is not listening to the trap receiving port.	Make sure the SNMP manager is really listening for traps/informs on the configured port.
	Network connectivity problem.	Check network connectivity between the TNMS server machine and the SNMP manager machine.  Confirm that the ports chosen for SNMP communication are not blocked by any firewall.
SET operation returns an error. (See below for cases specific to PM/OPM request handling.)	SNMP user doesn't have write permission.	Change permission of the SNMP to 'Read/Write' (see 3.2.2).
	The target MIB object is not writable.	Check the object the SNMP manager is trying to set.
	The type of the value in the SET request is not compatible with the MIB object.	Use the correct data type.
SET operation returns error when creating a PM/OPM request. (See more possible causes below.)	The instance identifier is already in use.	Make sure the instance identifier is not used by another row in the PM/OPM request table. An auto-increment variable may be used to generate instance identifiers (see 10.2 / 11.2).
	RowStatus attribute unset or set to invalid value.	When adding a new request, set the RowStatus attribute to <i>createAndGo</i> .
	The maximum number of PM / OPM requests (5000) has been reached.	Delete unused requests, or reuse an existing request by updating its attributes.



Symptom	Possible cause	Solution
SET operation returns error when creating or updating a PM/OPM request.	Invalid value assigned to an attribute.	Check the values supplied to the attributes – non-existent enumeration value, malformed dates, malformed object identifiers, etc.
	Some attribute value was set to a value inconsistent with other values.	<p>Make sure the request attributes stay consistent with each other. Examples:</p> <ul style="list-style-type: none"> <li>When setting the request filter type to <i>port</i>, the filter value must be a port identifier; the SNMP manager must change the filter type <i>and</i> the filter value at the same time, in a single SET command.</li> <li>When changing a PM request type to <i>history</i>, the start/end times must not be empty, otherwise the manager must set the request type and the start/end times simultaneously, in the same SET command.</li> </ul>
SET operation returns error when updating the state of the PM/OPM request.	The state transition is not valid.	See section 10.3 / 11.3 for possible state transitions.
SET operation returns error when deleting a PM/OPM request.	The request is in state <i>pending</i> , <i>started</i> or <i>cancelling</i> .	It is only possible to delete requests that are in idle state ( <i>created</i> , <i>finished</i> , <i>failed</i> and <i>cancelled</i> ) – the manager must wait for the request to reach one of such states.

Table 89 SNMP NBI troubleshooting table

## Abbreviations

<b>AC</b>	Attribute Change
<b>ACS</b>	Actual Creation State
<b>AES</b>	Advanced Encryption Standard
<b>AVC</b>	Attribute Value Change
<b>DB</b>	Database
<b>DES</b>	Data Encryption Standard
<b>3DES</b>	Triple DES
<b>EMS</b>	Element Management System
<b>EVC</b>	Ethernet Virtual Connection
<b>GUI</b>	Graphical User Interface
<b>HW</b>	Hardware
<b>IANA</b>	Internet Assigned Numbers Authority
<b>ITU</b>	International Telecommunication Union
<b>MD5</b>	Message Digest 5
<b>NBI</b>	Northbound Interface
<b>NE</b>	Network Element
<b>NMS</b>	Network Management System
<b>OC</b>	Object Creation
<b>OD</b>	Object Deletion
<b>OPM</b>	Optical Power Monitoring
<b>PEN</b>	Private Enterprise Number
<b>PDU</b>	Protocol Data Unit
<b>PM</b>	Performance Monitoring
<b>RCS</b>	Required Creation State
<b>RFC</b>	Request for Comments
<b>SC</b>	State Change
<b>SHA</b>	Secure Hash Algorithm
<b>SEL</b>	System Event Log
<b>SNMP</b>	Simple Network Management Protocol
<b>SW</b>	Software
<b>TCP</b>	Transmission Control Protocol

<b>TNMS</b>	Telecommunication Network Management System
<b>UDP</b>	User Datagram Protocol
<b>UNO</b>	Universal Network Object
<b>UI</b>	User Interface
<b>UNI</b>	User Network Interface