NMS SNMP Reference Manual

P/N 9000-6744-16

NMS Communications Corporation
100 Crossing Boulevard
Framingham, MA 01702

P/N 9000-6744-16

Revision history

Revision	Release date	Notes
1.0	September, 1999	EPS for CT Access 3.0
1.1	January, 2000	EPS for CT Access 3.0 GA
1.2	July, 2000	EPS / SJC, Platform support for Fusion 4.0
1.3	September, 2000	SJC for CT Access 4.0 GA
1.4	January, 2001	CYF
1.5	August, 2001	CYF, for NACD 2001-1 GA
1.6	May, 2002	MVH, for NACD 2002-1
This manual printed: May 28, 2002		

Refer to the NMS web site (*www.nmscommunications.com*) for product updates and for information about NMS support policies, warranty information, and service offerings.

Table of Contents

Introduction

Simple Network Management Protocol (SNMP) is a protocol defined by the Internet Engineering Task Force (IETF). Network devices supporting this protocol allow a management station to monitor network status, modify network settings, and receive network events.

The *NMS SNMP Reference Manual* explains how to configure and install SNMP for NMS Communications (NMS) products. The manual is intended for customers who want to add SNMP monitoring to NMS Communications boards. It provides an overview of SNMP and describes the *Management Information Bases* (*MIBs*) and *agents* used to support SNMP on NMS hardware.

Overview of NMS SNMP

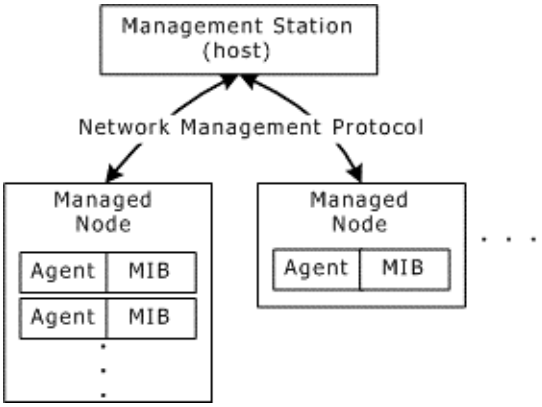
Using network management

Network management allows administrators to maintain network integrity. SNMP is an industry standard protocol that defines a method for performing network management. SNMP was initially made available for IP based enterprise networks, and is now available for telephony networks.

An SNMP network management system consists of:

- One or more *managed nodes*, running one or more SNMP *agents*. An agent keeps information about its managed node in a database called a *Management Information Base* (MIB).
- One or more network *management stations*, which run network management software and display network information. The management station is called the *host*.
- A network *management protocol*, which determines how the managed node and the management station can communicate with each other over the network.

The following illustration shows the relationship of SNMP components:



SNMP network components

In the following illustration, one management station is shown communicating with two managed nodes. The first managed node has more than one agent, and each agent has its own MIB. The dotted lines in the managed node show that there can be more agent/MIB pairs running on a managed node. The dotted lines to the right of the managed nodes show that there can be additional nodes managed by a single management station.

Object identifiers

An object identifier (OID) is a unique sequence of integers that represents how to traverse the MIB tree to get to a managed object. All MIBs have a common root node and all OID integer sequences start from that root. The OIDs are assigned by the IETF.

The entire tree of MIBs is referred to as a *namespace*, which means that each MIB and OID is unique. The namespace for the entire tree is maintained by the IETF and related organizations, who delegate that authority only for MIBs below the Enterprises MIB, whose OID is 1.3.6.1.4.1.

Managed network components

A typical managed network consists of the following components:

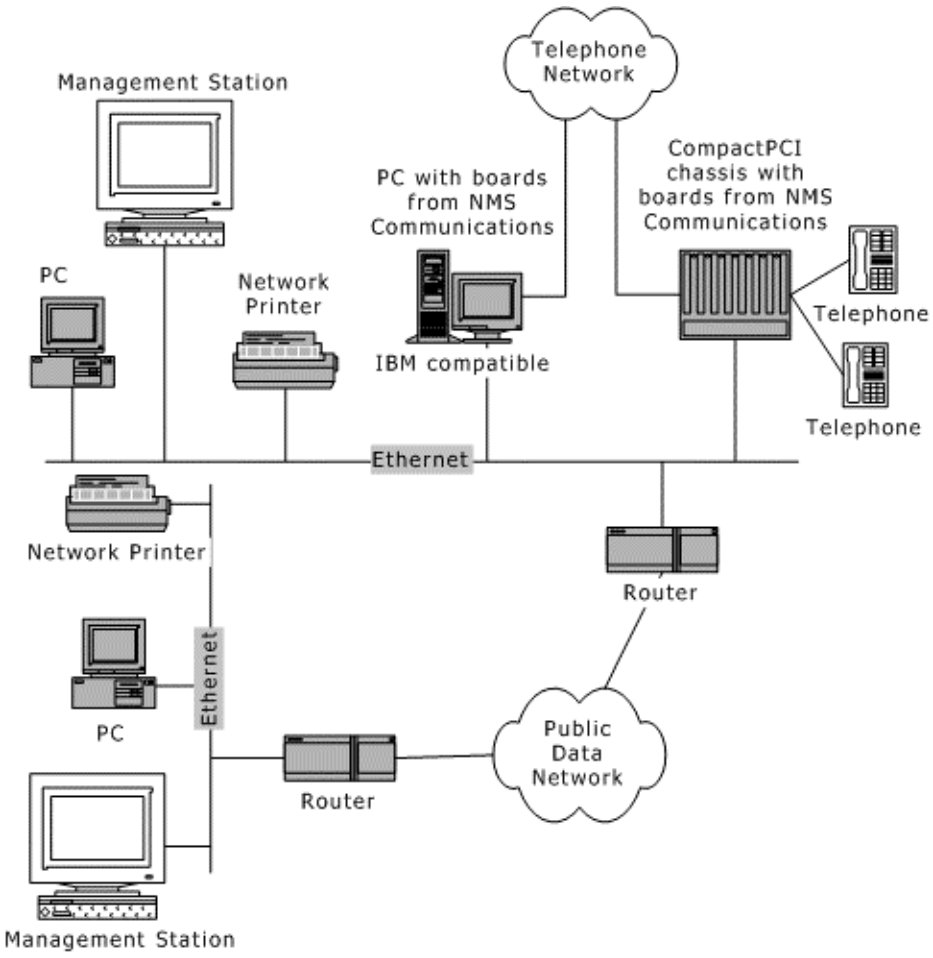
- Managed nodes
- Management information bases
- Management stations
- Management protocol

Managed nodes

Any device which connects to a data network and can execute the SNMP protocol can be an SNMP managed node. A managed node can be:

- A host system, such as a workstation, a printer, a file server, a terminal server, or a mainframe.
- A network router, a bridge, a hub, an analyzer or a repeater.

The following illustration shows managed nodes as grey, and management stations as white.



Managed nodes and management stations

A managed node executes a program called the SNMP service, which communicates with the management station. The SNMP service responds to messages from the host and sends unsolicited messages if a defined event occurs on the managed node.

The SNMP service is a daemon on UNIX systems and a system service under Windows.

The SNMP service runs one or more agents, which are applications that collect information about the managed node and keep it in a MIB. A managed node can have more than one MIB, and has one agent for each MIB. See Installation summary for information about how to activate the SNMP service and load an agent.

The SNMP architecture is designed to be simple and fast. The processing load is placed on the management station, and minimized on the managed node. The set of information contained in the MIB is designed to be simple, so information about the network will not congest the network.

Management Information Bases

A Management Information Base (MIB) defines the information that will be maintained by the associated agent. A MIB is viewed as a database, but is actually a sequential list of managed objects. The managed objects are logically grouped to represent a row in a table, where each object in that group represents a field. The field may be a variable, or a structure of variables. Each managed object is referred to by a unique *object identifier* (OID).

A MIB is often shown as a tree, where the nodes of the tree define the database and its tables, rows, and fields. The collection of all MIBs is organized in a tree structure, where each node on the tree represents a single MIB. The SNMP MIB hierarchy is defined by RFC 1155 and RFC 1213. MIBs fall into two categories:

MIB	Description
Standard	A standard MIB is defined by the IETF. An example of a standard MIB is RFC 2495, the Trunk MIB.
Private	A non-standard, proprietary MIB is defined by an enterprise. The IETF assigns a unique OID number to a company, under which they can define their own OIDs for their specific products. An example of a private MIB is the NMS Communications chassis MIB.

Management stations

A management station is a system running:

- The network management protocol.
- One or more network management applications.

The network management station (host) determines which information is required from the managed node. The host sends queries to a managed node to determine what information is available and to retrieve that information. The host then uses those responses to display the information in human readable form.

Host applications are much larger than agent applications, because they are designed to do most of the work in the SNMP architecture, and because one host application communicates with many agents. One example of a host management station is HP Openview.

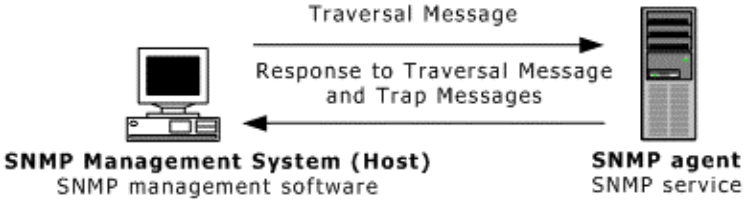
Management protocol

SNMP defines a mechanism to transport network management information. Messages containing queries and replies are sent between the host management system and managed nodes over a connectionless transport service. A commonly used transport service is User Datagram Protocol (UDP), which is part of the IP suite.

Two types of messages are supported:

Message	Description
Traversal	Provides a way for the host to read the values in an agent's MIB.
Trap	Sent by an agent to report events to the host.

The following illustration shows the host and agent message flow:



SNMP message flow

Traversal messages are generated by host commands. These commands are:

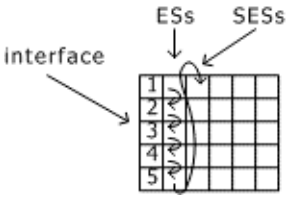
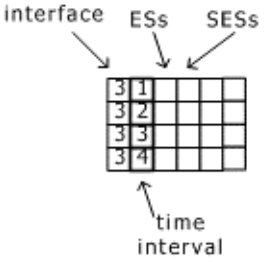
Command	Description
get	Requests a specific value (for example, the amount of hard disk space available).
get-next	Requests the next value in a MIB after using the **get** command. Useful when getting a block of related objects.
set	Changes the value of an object in a MIB. Only objects with read-write access can be set.

Trap messages are sent by an agent to notify the host about an unusual occurrence. The host can then request the value of related variables to determine more about the managed node's condition. The agent can be set to send a trap when certain conditions arise, such as an error on a line. Care must be taken to ensure that trap information does not congest the network or overwhelm the host.

Connectionless transport does not guarantee delivery, which means that traps (and other network messages) are not guaranteed to arrive at the host. You should plan your network management policies to consider lost messages.

Accessing MIB objects

Objects in a MIB can be accessed in the following ways:

Type	Description
Single	Contains a single value. Getting the value for an instance of this object type requires adding a 0 to the end of the OID. For example, if the OID to a single object type is p, then use p.0 to get its value.
Indexed Table	The column is the type of item, and the row (index) is the instance of that item type. The OID of the start of the table is p, and p.column.index describes a field, where index specifies the row.
Doubly Indexed Table	Uses two indices to specify a row. The column is the type of item and the row is defined by two indices that further define the meaning of that row. The OID of the start of the table is p. p.column.index1.index2 specifies a field, where get-next operator finds the next object in the current MIB that has a value. It returns the value of the object and its OID. If the current object is in a table, it returns the next column, which is the last digit in the OID. The OID to a field in an indexed table is get-next retrieves the next get-next moves to the next column. These actions represent reading the table from top to bottom, then left to right. For example, the Trunk MIB has an indexed table called the Current table, where each row is the index of the interface and each column is a statistic. If you **get** ESs for interface 1, then each **get-next** retrieves ESs for the next interface, as shown in the following illustration:  *Indexed table* When **get-next** has retrieved ESs for interface 5, the next **get-next** retrieves SESs for interface 1. The OID to a field in a doubly indexed table is p.column.index1.index2. The field is grouped by index1, and the particular field in that group is specified by get ES for the first time interval of the third interface, **get-next** retrieves ES for the next time interval, as shown in the following illustration:  *Doubly indexed table* When **get-next** has retrieved ES for all intervals of interface 3, the next **get-next** will either get ES for the first interval of the next interface (if there is one), or SES for the first interval of interface 3. For more information, refer to Trunk MIB Current table and Trunk MIB Interval table.

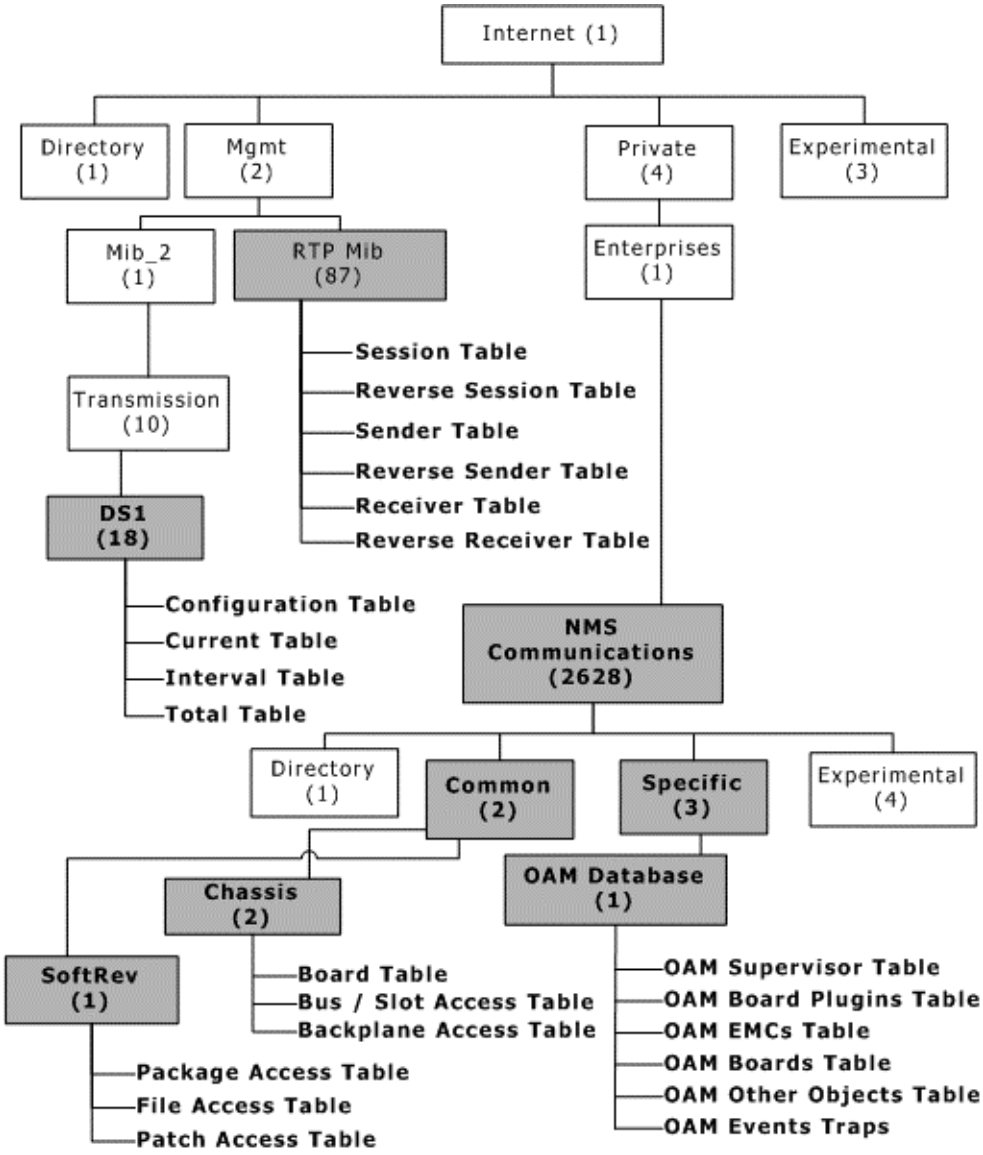
Supported MIBs

NMS produces agents for the following MIBs:

MIB	Description	Ownership	Installed by
Trunk	The Trunk MIB (also called the DS1 MIB), which represents DS1 (and higher speed) lines and is defined by the IETF.	RFC-2945 (obsolete RFC-1406)	OAM package
Chassis	Represents the PCI buses and slots, bus segments, and boards in the chassis. The Chassis MIB detects the presence of each board, and monitors its operational status.	Proprietary	OAM package
Software Revision	Tracks the versions of all NMS software installed in a chassis. The MIB tracks each NMS package, the files in each package, and service packs and patches applied to each package. The Software Revision MIB is modified whenever packages, service packs, or patches are installed or removed.	Proprietary	OAM package
OAM Database	Represents the contents of the NMS OAM database: board, board plug-in, and Extended Management Component (EMC) settings. The contents of the NMS OAM database can be modified via this MIB.	Proprietary	OAM package
RTP	Allows monitoring of the managed objects of the RTP system (configuration is not allowed). Displays only RTP session parameters and statistics using the NMS MSPP service. This subagent does not allow row creation or parameter modification. For more information about the RTP MIB, refer to the *Fusion Developer's Manual*.	RFC-2959	Fusion package

NMS has been assigned a namespace under the Enterprises MIB. The OID for the NMS MIB is 1.3.6.1.4.1.2628, under which the Chassis MIB, Software Revision MIB, and OAM Database MIB reside, and future private MIBs will be created.

In the following illustration, the SNMP subagents are shown with their major tables. The MIBs that are currently implemented are shown in grey:



Enterprise MIB

The NMS subtree consists of the following MIBs:

MIB	Description
Directory	Describes all MIBs defined by NMS.
Common	Contains general-purpose MIBs, applicable across multiple product lines.
Specific	Contains specialized MIBs for individual products.
Experimental	Contains MIBs that are under development and test.

MIB description files (in ASN-1 language) for the NMS SNMP subagents can be found in the *\nms\ctaccess\doc* directory. The following table lists the MIB description files:

MIB Description File	Description
NmsChassis.mib	Chassis MIB
NmsOamDatabase.mib	OAM Database MIB
NmsSmi.mib	NMS hierarchy MIB
NmsSoftRev.mib	Software Revision MIB
NmsTrunk.mib	Trunk MIB (DS1)
NmsRtp.mib	RTP MIB (installed by Fusion package)

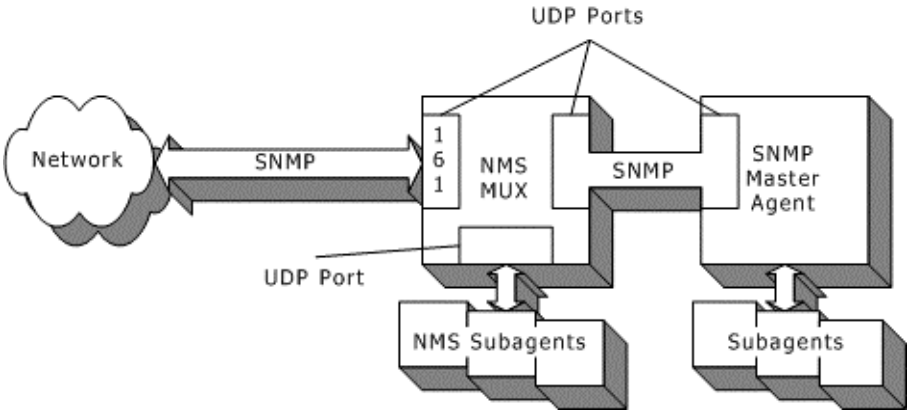
These text files require other MIB description files documented in RFCs (such as SNMPv2-TC, SNMPv2-CONF, SYSAPPL-MIB, etc.). These files can be easily found on the web. The NMS Communications hierarchy shown in the following illustration is defined in the NMS-SMI MIB.

NMS SNMP architecture

NMS SNMP consists of the following components:

- The NMS multiplexer (mux)
- NMS subagents for each MIB

As shown in the following illustration, the NMS multiplexer is located between the native SNMP master agent and the UDP port to the external network. The native master agent is reconfigured to communicate with the NMS multiplexer instead of the external network. The NMS multiplexer communicates with the NMS subagents (one for each MIB).



NMS SNMP architecture

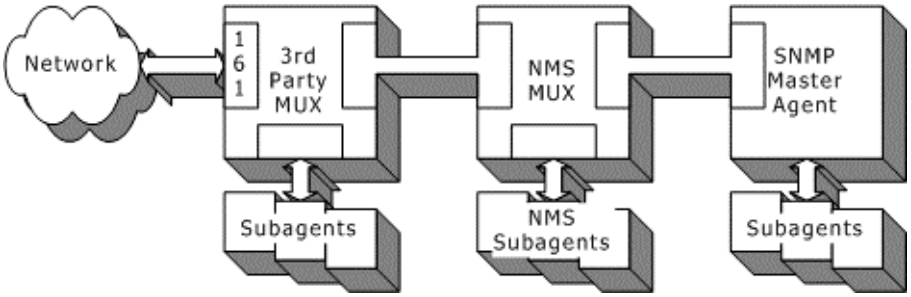
The NMS multiplexer handles all requests coming from the UDP network port, and communicates with the NMS subagents as needed. SNMP requests not addressed to the NMS multiplexer are routed to the native master agent. Each NMS subagent runs in a different process and exchanges information with the multiplexer using a UDP socket connection. The NMS multiplexer is thus connected to three different IP/UDP ports:

- SNMP network port (default value: 161)
- Communication port between the SNMP master agent and the multiplexer (default: 49212)
- Communication port between the SNMP subagents and the multiplexer (default: 49213)

These IP/UDP ports can be changed by editing the *snmp.cfg* file as described in Reconfiguring multiplexer IP/UDP ports.

The multiplexer console program, *muxC*, can read the *snmp.cfg* file and can display the currently used IP/UDP ports. It can also start and stop the agents gracefully without having to kill the process. Also, it can display all currently registered subagents.

As shown in the following illustration, configurable IP/UDP ports allow the NMS multiplexer to be inserted in a "chain" of multiplexers, if necessary. In this configuration, each multiplexer processes incoming SNMP requests. Requests not addressed to a given multiplexer are passed to the next one.

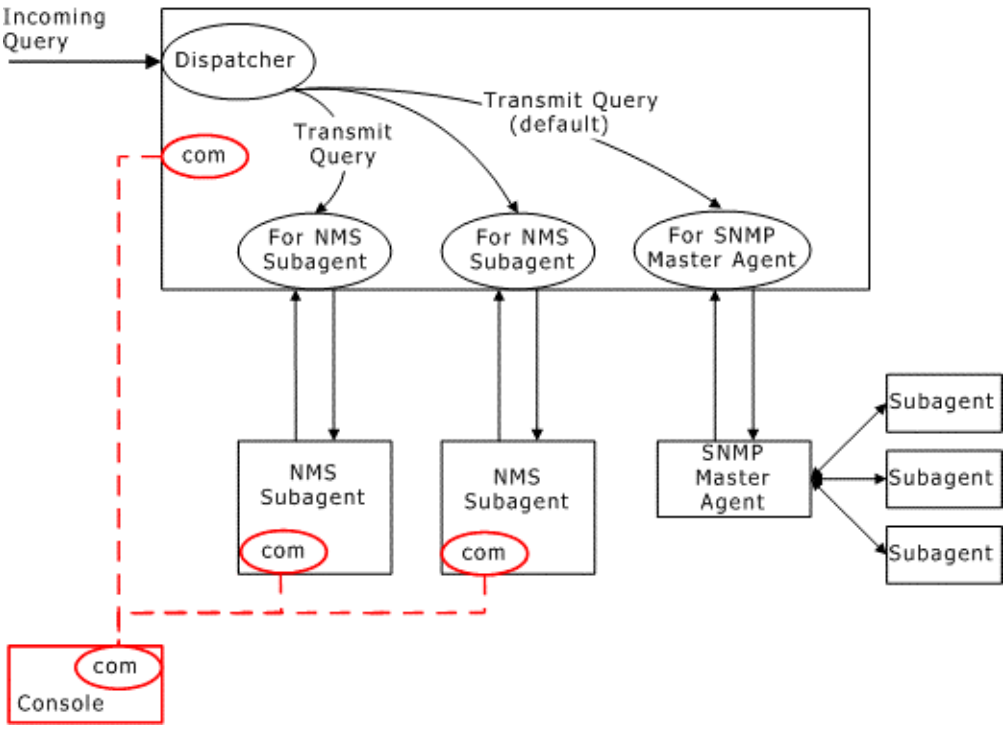


Multiplexer chain

The main reasons for the multiplexer are the following:

- Uniform structure of SNMP agents and subagents
- Dynamic agent and subagent insertion, removal, and update
- Independence from differing master agent implementation and protocols under each operating system
- Uniform trap environment, adopting a SOLARIS-like approach.

The following illustration shows the inner architecture of the NMS multiplexer:



NMS multiplexer internal architecture

Installing and configuring NMS SNMP

Installation summary

This topic summarizes procedures for installing and activating NMS SNMP software.

Note: NMS SNMP software components fully support SNMP version 1, but do not fully support SNMP version 2. For example, the **get-bulk** operator is not correctly supported, and SNMP traps are generated in version 1 format. We thus recommend that you use SNMP request version 1 when accessing NMS subagents.

Supported operating systems

NMS SNMP software is available for the following operating systems:

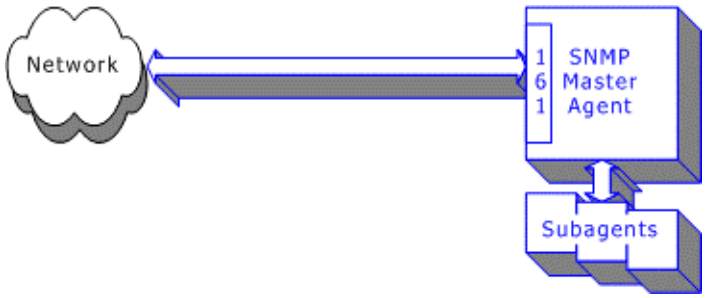
- Windows 2000
- SPARC Solaris
- Intel Solaris
- Red Hat Linux

Installation and configuration overview

This section outlines the steps required to install and configure NMS SNMP. Each step is described in detail in the sections that follow.

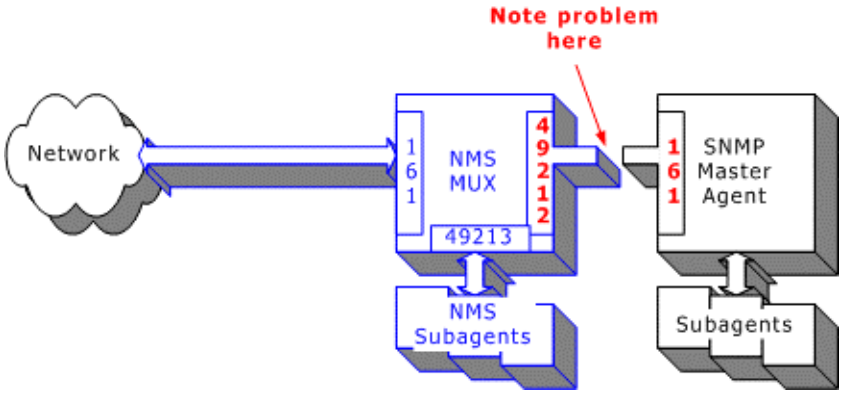
To install and configure NMS SNMP:

1. Install the SNMP master agent. By default, the master agent communicates with the network using UDP port 161:



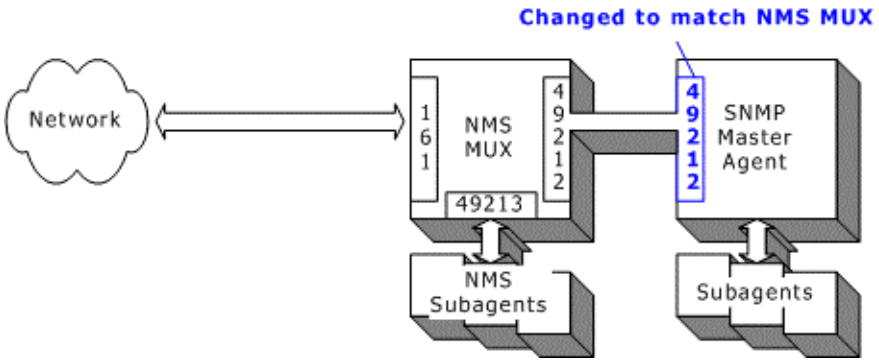
Installing the SNMP master agent

2. Install the NMS multiplexer (mux) and subagents:



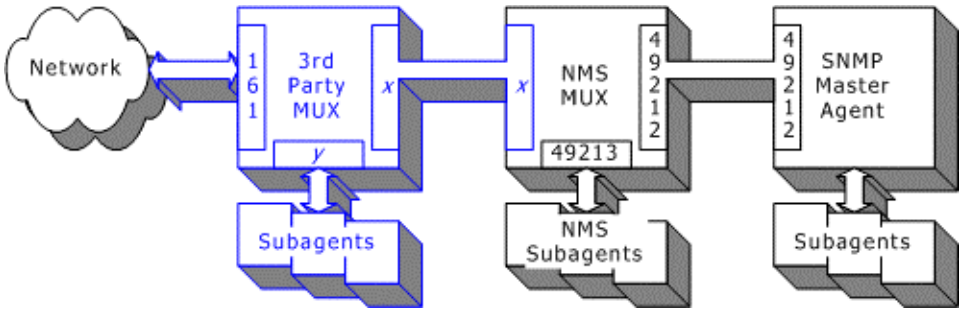
Installing the NMS multiplexer

3. To set up the NMS multiplexer "between" the network and the SNMP master agent, configure the master agent so one of its UDP ports matches the NMS multiplexer. By default, the NMS multiplexer's secondary port is port 49212:



Configuring the SNMP master agent UDP port

4. Start up the SNMP master agent, NMS multiplexer and subagents.
5. If you wish to include a third-party multiplexer, set it up "between" the network and the NMS multiplexer, and configure the ports accordingly:



Configuring a third-party multiplexer

Note: In all cases, all ports must be unique, and the UDP port connecting to the network must be port 161.

Installing the master agent under UNIX

This topic describes how to install, configure, and start the SNMP master agent on SPARC Solaris and Linux systems.

Note: The installation and configuration procedures for NMS subagents are different from that of the master agent. To install the NMS subagents and multiplexer, refer to Installing the NMS subagents and multiplexer. To configure the NMS subagents, modify the file *snmp.cfg* as described in Reconfiguring multiplexer IP/UDP ports.

Installing under Solaris

To install, configure, and start the SNMP master agent on a Solaris system:

Note: For detailed information, see the *Solstice Enterprise Agents User Guide*.

1. Log on as superuser.
2. Install the Solstice Enterprise agent access control.
3. Access a command prompt.
4. To start the master agent, enter the following:

   ```
   /etc/init.d/init.snmpdx start
   ```

5. Verify that the SNMP master agent is properly installed.

   To do so, you can use any SNMP management station. You can also use the *snmpwalk* demonstration program (installed with the SNMP package) to enumerate the contents of the Mib II agents.

6. Install the NMS subagents and multiplexer as described in Installing the NMS subagents and multiplexer.

Installing under Linux

The SNMP package has been tested using the UCD (University of Columbia at Davis) SNMP release *ucd-snmp-4.0.1-4.rpm*. You can find the latest UCD package at *rpmfind.net* or at *net-snmp.sourceforge.net* (previously known as *ucd-snmp.ucdavis.edu*) web site.

To install, configure, and start the SNMP master agent on a Linux system:

1. Install the SNMP package. To do so, access a command prompt and enter:

   ```
   rpm -i ucd-snmp-4.0.1-4.rpm
   ```

2. Start the SNMP master agent by entering:

   ```
   /etc/rc.d/init.d/snmpd start
   ```

3. Verify that the SNMP master agent is properly installed.

   To do so, you can use any SNMP management station. You can also use the *snmpwalk* demonstration program (installed with the SNMP package) to enumerate the contents of the Mib II agents.

4. Install the NMS subagents and multiplexer as described in Installing the NMS subagents and multiplexer.

Installing the master agent under Windows 2000

The installation and configuration procedures for NMS subagents are different from that of the master agent. To install the NMS subagents and multiplexer, refer to Installing the NMS subagents and multiplexer. To configure the NMS subagents, modify the file *snmp.cfg* as described in Reconfiguring multiplexer IP/UDP ports.

To install, configure, and start the SNMP master agent under Windows 2000:

1. Click **Start**-->**Settings**-->**Control Panel**.

   The Control Panel window appears.

2. In the Control Panel window, double-click the **Add/Remove Programs** icon.

   The Add/Remove Programs dialog box appears.

3. Click **Add/Remove Windows Components** in the bar to the left of the window.

   The Windows Components Wizard dialog box appears, displaying Windows packages you can install.

4. Select the **Management and Monitoring Tools** package, and press **Next**.

   The Management and Monitoring Tools package is installed. During installation, you will be prompted to insert the Windows 2000 distribution CD.

5. (Optional) Configure the SNMP master agents. To do so:

   a. In the Control Panel window, double-click on the Administrative Services icon.

      The Administrative Services dialog box appears.

   b. Right-click on **SNMP Service**, and select **Properties** in the menu that appears.

      The SNMP Properties dialog appears.

   c. Select the **Traps** tab.

   d. Add a Community Name. For example: `public`.

   e. Add the addresses of the hosts that you wish to send traps to (if any) to the **Trap Destination** list.

   f. In the **Security** tab, you can modify the access rights.

   g. When you are finished, click **OK**.

6. Open a command prompt window.

7. Enter the following to start the SNMP service:

   ```
   net start snmp
   ```

8. Enter the following to start the SNMP trap service:

   ```
   net start snmptrap
   ```

   The SNMP trap service is not required if you use only NMS subagents. However, you will need it if other standard subagents are attached to the master agent.

9. Enter the following to verify that the master agent is properly installed:

```
netstat -a
```

   If the master agent is installed properly, the following appears:

```
UDP snmplab_3:snmp-trap *:*
UDP snmplab_3:snmp *:*
```

10. Verify that the SNMP master agent is properly installed.

    To do so, you can use any SNMP management station. You can also use the *snmpwalk* demonstration program (installed with the SNMP package) to enumerate the contents of the Mib II agents.

11. Install the NMS subagents and multiplexer as described in Installing the NMS subagents and multiplexer.

Note: Make sure to add the installed components to the registry as described in Modifying the Windows registry.

When you first install the SNMP service under Windows 2000, the public community has only READ_ONLY access.

Note: For more detailed configuration information, see the Windows 2000 documentation for SNMP.

Installing the NMS subagents and multiplexer

Once the SNMP master agent is working properly, you can install the NMS subagents and the NMS SNMP multiplexer included on the Natural Access CD-ROM. Installing the NMS OAM package from a Natural Access CD-ROM installs the NMS SNMP subagents. For information about installing Natural Access, refer to the installation booklet included with the CD-ROM. For further information on NMS OAM, refer to the *NMS OAM System User's Manual*.

When you have installed the NMS subagents and multiplexer, modify the IP/UDP port used by the SNMP master agent as described in Modifying the master agent IP/UDP port.

Modifying the Windows registry (Windows only)

Under Windows 2000, Natural Access automatically registers all installed components in the registry. When Natural Access is uninstalled, the components are automatically removed from the registry.

You can manually add or remove components from the registry. To do so, access a command prompt and enter the following command:

```
component_name directive
```

where:

- ***component_name*** is the name of the component to add. ***component_name*** can be any of the following:

Value	Description
mux	NMS multiplexer
chassisAgent	Chassis MIB agent
ds1Agent	Trunk MIB agent
oamAgent	OAM Database MIB agent
softRevAgent	Software Revision MIB agent
rtpAgent	RTP MIB agent (installed with Fusion package)

directive indicates whether to install or remove the component. ***directive*** can be either of the following:

Option	Description
-I	Install the component
-U	Uninstall the component

For example, to remove the Chassis MIB agent from the registry, enter:

```
chassisAgent -U
```

Modifying the master agent IP/UDP port

Once the NMS subagents and multiplexer are installed, modify the SNMP master agent's IP/UDP port so it connects to the NMS multiplexer port (port 49212) instead of the network port (port 161). The following sections describe how to change the SNMP master agent's port under all supported operating systems.

You can use another port if port 49212 is already in use, or if you are already using an SNMP multiplexer in your system. To configure the secondary port on the multiplexer, edit the snmp.cfg file and modify the value MasterAgentPort in the [common] section (see Reconfiguring multiplexer IP/UDP ports). Then restart the multiplexer and the subagents to make your changes effective.

The following sections describe how to install, configure, and start the SNMP master agent under various operating systems.

Windows 2000

To change the SNMP master agent's UDP port:

1. Open the file *Services* for editing. This file can be found in */WINNT/system32/drivers/etc/Services*.
2. In this file, find the following line:

   ```
   SNMP 161 / udp
   ```

3. Change the line to:

   ```
   SNMP 49212 / udp
   ```

4. Save and close this file.
5. Open a command prompt window.
6. Stop and restart the SNMP service by entering the following commands:

   ```
   net stop snmp

   net start snmp
   ```

Solaris

To change the SNMP master agent's UDP port:

1. Log in as superuser.
2. Open the file `/etc/init.d/init.snmpdx` for editing.
3. In this file, find the line beginning with:

   ```
   /usr/lib/SNMP/snmpdx -p 161 ...
   ```

4. Replace the first section of the line with:

   ```
   /usr/lib/SNMP/snmpdx -p 49212 ...
   ```

5. Save and close this file.
6. Access a console window.
7. Stop and restart the master agent, if it is running. To determine if the master agent is running, enter:

   ```
   ps -A | grep snmpdx
   ```

   If the master agent is running, the command will produce output similar to:

   ```
   136 ? 0:00 snmpdx
   ```

8. If the master agent is running, run the *kill* command to send a kill signal to that process, using the output of the previous command:

   ```
   kill -9 136
   ```

   Another way to stop the master agent process is by entering: `/etc/init.d/init.snmpdx stop`

9. Restart the master agent. To do so, enter:

   ```
   /etc/init.d/init.snmpdx start
   ```

The following is an extract of the file *init.snmpdx*:

```
#
# Copyright (c) 1997 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)init.snmpdx        1.12    97/12/08 SMI"

case "$1" in
'start')
     if [ -f /etc/SNMP/conf/snmpdx.rsrc -a -x /usr/lib/SNMP/snmpdx ]; then
          /usr/lib/SNMP/snmpdx -p 161 -y -c /etc/SNMP/conf -d 0
     fi
     ;;

'stop')
     /usr/bin/pkill -9 -x -u 0 '(snmpdx|snmpv2d|mibiisa)'
     ;;

*)
     echo "Usage: $0 { start | stop }"
```

Linux

To change the SNMP master agent's UDP port:

1. Log in as superuser.
2. Open the file */etc/rc.d/init.d/snmpd* for editing.
3. In this file, find the line beginning with:

   ```
   daemon "/usr/sbin/snmpd" ...
   ```

4. Replace the first section of the line with:

   ```
   daemon "/usr/sbin/snmpd -p 49212" ...
   ```

5. Save and close this file.
6. Open a console window.
7. Now stop and restart the master agent, if it is running. To determine if the master agent is running, enter:

   ```
   ps -A | grep snmpd
   ```

   If the master agent is running, the command will produce output similar to:

   ```
   136 ? 0:00 snmpd
   ```

8. If the master agent is running, run the *kill* command to send a kill signal to that process, using the output of the previous command:

   ```
   kill -9 136
   ```

   Another way to stop the master agent process is by invoking:

   ```
   /etc/rc.d/init.d/snmpd stop
   ```

9. Restart the master agent by entering:

   ```
   /etc/rc.d/init.d/snmpd start
   ```

Configuring NMS SNMP

Use the SNMP configuration file *snmp.cfg* to set the IP/UDP ports used by the multiplexer to communicate with the master agents, receive or send SNMP requests, and communicate with the NMS subagents.

You can also use *snmp.cfg* to

- Set the write access for a given subagent using a community name
- Set the trap destination for one or more subagents
- Set information specific to a given MIB (for example, the Chassis MIB information)

The *snmp.cfg* file is installed in one of the following directories:

OS	Path
Windows 2000	*\nms\ctaccess\cfg*
UNIX	*/opt/nms/ctaccess/cfg*

Configuration file syntax

Statements within the file appear one to a line. Any text appearing after a pound sign (#) is a comment, and is ignored. Statements are case-insensitive, except where operating system conventions prevail (for example, filenames under UNIX).

The *snmp.cfg* file is divided into multiple sections. Each section has a header, appearing in square brackets ([ ]). The statements within each section apply to one or more subagents. The sections are as follows:

Section	Subagent(s)
[common]	All subagents
[chassisAgent]	Chassis subagent only
[ds1Agent]	Trunk subagent only
[oamAgent]	OAM Database subagent only
[softRevAgent]	Software Revision subagent only
[rtpAgent]	RTP subagent only

Statements within a section each consist of a keyword name, followed by an equals sign (=) and then a value:

keyword = ***value***

The [common] section

The [common] section contains statements that apply to all subagents. This section contains the following keywords:

Keyword	**Description**	**Allowed values**	**Manda-tory?**
SnmpPort	Defines the port through which SNMP queries will be sent to the multiplexer	Valid UDP port number	Yes
MasterAgentPort	Defines the port through which the multiplexer will send SNMP requests not addressed to its subagents	Valid UDP port number	Yes
CommunicationPort	Defines the port used by the multiplexer, the subagent, and the console to communicate (for registration, stop or info commands)	Valid UDP port number	Yes
access	Defines the access rights and the defined communities that can be used to send requests to the agents.	***access***,***community***,***host*** ... where: • ***access*** defines the access right: readonly, writeonly, or readwrite • ***community*** is the name of a defined community that can be used to send requests to the agents • ***host*** specifies the name of the host where the SNMP requests are authorized. An asterisk (*) character indicates that any host is allowed.	No
trap	Defines the host where the trap will be sent and the community that will be used.	***community***,***host*** ... where: • ***community*** is the name of a defined community that can be used to send requests to the agents • ***host*** specifies the name of the host where the SNMP requests are authorized. An asterisk (*) character indicates that any host is allowed.	No

The subagent-specific sections

Below the [common] section appear sections containing statements that apply to individual subagents only. Any configuration parameters needed by a given subagent must appear in the section for the subagent.

The access and trap keywords (defined as in the [common] section) can also appear in the subagent-specific sections, to define additional access and trap host settings for individual subagents only. Traps from a given subagent will be sent to all hosts listed in the section for the subagent, as well as the hosts listed in the [common] section.

Sample SNMP configuration file

The following code shows typical entries within an SNMP configuration file. Indentations in the file are optional, for user readability only.

```
#=========================================================================
# snmp.cfg
#
# This is an example of a file that specifies an SNMP configuration.
# This file must be placed in the nms/ctaccess/cfg directory.
#
#=========================================================================

[common]

# Definition of the UDP/IP ports used by the multiplexer to communicate with
# the Master Agent and the NMS agents.
SnmpPort = 161
MasterAgentPort = 49212
CommunicationPort = 49213

# Default access rights to the NMS agents. Format: <r/w>,<community>,<host>
access = readwrite, public, *
#access = readonly, guest, snmplab_3

# Default trap destinations for the NMS agents. Format: <host>,<community>
trap = localhost, public
#trap = snmplab_3, private

# Keep this line to allow the Multiplexer to send requests to NMS subagents:
access = readwrite, *, localhost

[chassisAgent]

# Type of chassis. Allowed values: 1=Unknown chassis
#                                  2=CPCI chassis
#                                  3=Generic PC chassis
#                                  4=Generic Sun chassis
chassType = 3

# Description string for the chassis.
chassDescr = Generic PC development computer

# Descriptions of the boards in the chassis.
# Format: <board no.>,<description string>
#boardDescr = 0, Tested 01/25/1991
#boardDescr = 1, Bad
#boardDescr = 3, Bad

# List of access rights. Format: <r/w>,<community>,<host>
#access = writeonly, private, snmplab_3

# List of trap destinations. Format: <host>,<community>
#trap = localhost, public
#trap = snmplab_3, private
```

```
[ds1Agent]

# List of access rights. Format: <r/w>,<community>,<host>
#access = writeonly, private, snmplab_3

# List of trap destinations. Format: <host>,<community>
#trap = localhost, public

[oamAgent]

# List of access rights. Format: <r/w>,<community>,<host>
#access = writeonly, private, snmplab_3

# List of trap destinations. Format: <host>,<community>
#trap = localhost, public

[softRevAgent]

# List of access rights. Format: <r/w>,<community>,<host>
#access = writeonly, private, snmplab_3

# List of trap destinations. Format: <host>,<community>
#trap = localhost, public

[rtpAgent]
# *** Note: The RTP Agent is installed with the Fusion Package ***

# List of access rights. Format: <r/w>,<community>,<host>
#access = writeonly, private, snmplab_3

# List of trap destinations. Format: <host>,<community>
#trap = localhost, public
```

Activating SNMP

Starting the NMS multiplexer and subagents

To start the NMS multiplexer and the subagents, you can

- Use the *muxC* console program.
- Enter commands at a command prompt.

Note: Under Solaris, you must start the Solstice master agent before starting the multiplexer, because the Solstice agent will not operate if it discovers (on startup) that its IP/UDP port will be shared with the multiplexer. If the Solstice agent is started before the multiplexer, it will operate normally.

Starting SNMP using muxC

To start the NMS multiplexer and subagents using *muxC*:

1. Access a command prompt.
2. Enter the following to start *muxC*:

   ```
   muxC
   ```

   The following appears:

   ```
   ***********************************
   *                                 *
   *       MULTIPLEXER CONSOLE       *
   *                                 *
   ***********************************
   A) Show the ports configuration
   -----------------------------------
   B) Start the SNMP Master Agent
   C) Start the NMS Multiplexer
   D) Start the NMS Sub-agents
   -----------------------------------
   E) Stop the SNMP Master Agent
   F) Stop the NMS Multiplexer
   G) Stop the NMS Sub-agents
   -----------------------------------
   H) Show the running NMS Sub-agents
   -----------------------------------
   I) Refresh the screen
   Q) Quit the console

   COMMAND> _
   ```

3. Enter B to start the SNMP master agent.
4. Enter C to start the NMS Multiplexer.
5. Enter D to start the SNMP subagents.

By default, *muxC* starts and stops the SNMP subagents and the multiplexer as Windows services using the `net start` and `net stop` commands.

The *muxC* command line option `-d` causes a terminal window to be created each time you start the multiplexer and/or the SNMP subagents. The components are started in debug mode with the following command:

```
muxC -d
```

Starting SNMP using the command line

To start the components using the command line:

1. Access a command prompt.
2. Enter the following for each component:

For this operating system type...	Enter...
Windows 2000	`net start` ***component_name***
UNIX	***component_name***

component_name is the name of the component to start. ***component_name*** can be any of the following:

Name	Description
mux	NMS multiplexer
chassisAgent	Chassis MIB agent
ds1Agent	Trunk MIB agent
oamAgent	OAM Database MIB agent
softRevAgent	Software Revision MIB agent
rtpAgent	RTP MIB agent (installed with Fusion package)

Under Windows 2000, the SNMP components are implemented as services. Under UNIX, they are implemented as daemon programs.

To obtain error information, you can start the subagents in console mode directly. To do so, specify the -d option on the command line:

```
softRevAgent -d
```

In console mode, the agent displays information like the following:

```
Inserting : .1.3.6.1.4.1.2628.2.1.1
Inserting : .1.3.6.1.4.1.2628.2.1.2.1.1
Inserting : .1.3.6.1.4.1.2628.2.1.3.1.1
Nms Snmp Software Revision Agent service started
```

Reconfiguring multiplexer IP/UDP ports

This section describes how to change the IP/UDP ports used by the NMS multiplexer, once the master agent, the NMS multiplexer, and the subagents are running.

By default, the following IP/UDP are used by the NMS multiplexer:

IP/UDP Port	Value
Communication port between the NMS multiplexer and the network	161
Communication port between the SNMP master agent and the NMS multiplexer	49212
Communication port between the NMS SNMP subagents and the NMS multiplexer	49213

These values are stored in the *snmp.cfg* file. To change the values, edit this file, as follows:

1. Locate the *snmp.cfg* file in one of the following directories:

Operating system	Directory
NT	*\nms\ctaccess\cfg*
UNIX	*/opt/nms/ctaccess/cfg/*

2. Modify the settings in the file.
3. Save and close the file.
4. To make your changes effective, restart the master agent, the NMS multiplexer, and subagents.

For more information on the *snmp.cfg* file, see Configuring NMS SNMP.

Running NMS SNMP

Once NMS SNMP is installed, in order for the software to operate:

- All Natural Access environment variables must be properly set.

  To learn about Natural Access environment variables, refer to the *Natural Access Developer's Reference Manual*.

- The DTM service must be specified in the *cta.cfg* file.

  By default, this service is specified in the file. Refer to the *Natural Access Developer's Reference Manual* for more information about *cta.cfg*. Refer to the *T1/E1 Digital Trunk Monitor Service Developer's Reference Manual* for information about DTM.

- The Natural Access server (*ctdaemon*) must be running.

  The NMS SNMP agents will not recognize any boards unless *ctdaemon* is running (in order to activate the NMS OAM database). Also, the agents will not report any data to SNMP requests.

  To verify that *ctdaemon* is running:

  1. Access a command prompt.
  2. Enter the *ps* command, as follows:

     ```
     ps -A | grep ctdaemon
     ```

     This command produces output similar to the following:

     ```
     1028 TS 85 pts/3 0:00 ctdaemon
     ```

Note: The SNMP subagents continue to work whether or not *ctdaemon* is running. If you restart *ctdaemon* while a subagent is running, boards will be detected (if configured). Refer to the Natural Access installation booklet and to the *NMS OAM System User's Manual* for more information about starting Natural Access and the NMS OAM service.

Chassis MIB

Chassis MIB representation

The Chassis MIB represents the boards installed in an NMS chassis. Boards and lines (trunks) are numbered sequentially, and are assigned to tables.

The Chassis agent detects each NMS board that has both been registered to NMS OAM and has booted correctly, and monitors its operational status. The board model, type, revision, bus segment and slot, and logical ID are represented. Removing or inserting a board (Hot Swap) is also monitored, and traps are sent if the status of a board changes.

Using the Chassis MIB

This section describes how to use the values in the Chassis MIB and provides other information common to more than one table.

Traps

Traps can be enabled to report a change in board status. The **boardStatusChangeTrapEnable** object in the Board table can be set to enable or disable traps. Traps must also be configured. See Installation summary for more information about configuring traps.

Hot Swap

Extracting a board causes the entry for that board in the Board table to be removed. If all the boards in a bus segment are extracted, that bus segment entry will be removed from the Bus Segment table. If the removed entry creates a non-contiguous numerical sequence, that number will be used the next time a board is inserted (and recognized by the agent). The Hot Swap software sees an inserted board before the agent has access to it.

Note: Hot Swap works only with the CompactPCI bus. The Hot Swap functionality described above is available only if the Hot Swap Manager is running. For more information on running the Hot Swap Manager, refer to the *NMS OAM System User's Manual*. (If using Natural Access 3.x, refer to the *Hot Swap Developer's Manual*.)

Board status

Board status differs depending on which version of Natural Access you are using.

- When using Natural Access 3.x, two board status objects are used: **boardStatus** in the Board table, and **slotStatus** in the Board Access by slot table. **boardStatus** provides a simple (online, offline, or pending) message, and **slotStatus** gives a finer grain value, the Hot Swap state. For more information about Hot Swap states, see the *Hot Swap Developer's Manual*.
- In Natural Access 4.0 (or later), **boardStatus** and **slotStatus** are identical, and their functionality is equivalent to that of **slotStatus** under CT Access 3.x.

The two objects tie together by an index value. **boardIndex** in the Board table matches the **slotBoardIndex** in the Board Access by slot table.

Linking to the trunk MIB

dsx1CircuitIdentifier in the Trunk MIB (RFC 2495) contains the name of the board that the line is on, as well as a board number and trunk number. The board text portion maps to the **boardFamilyId** in the Chassis MIB, and the board number maps to the **boardIndex** in the Chassis MIB (both objects are in the Board table).

For example:

dsx1CircuitIdentifier = AG_Dual_T1_02_01

boardModelText = AG_Dual_T1

boardFamilyNumber = 2

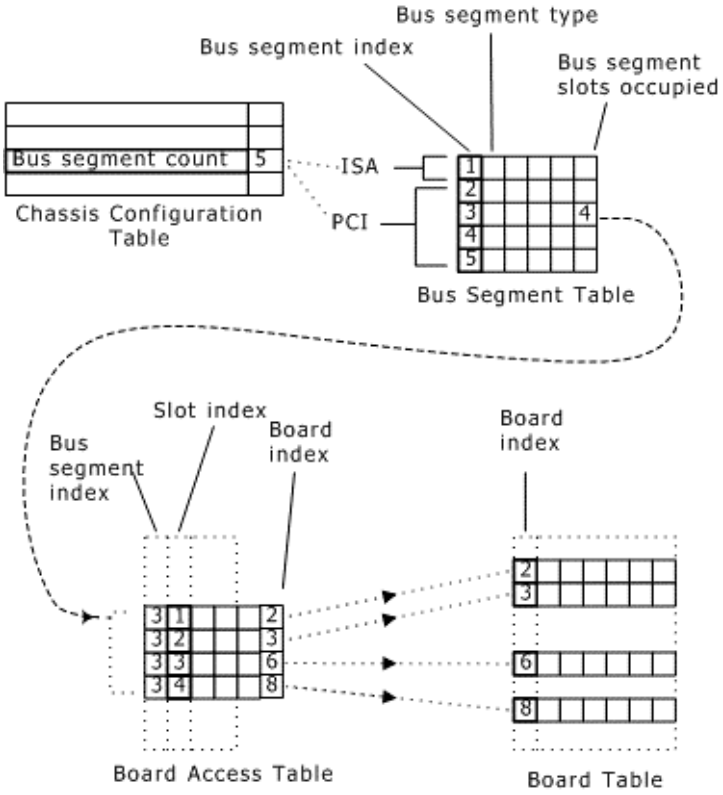
In this example, **dsx1CircuitIdentifier** says that the trunk is on an AG_Dual_T1 board, the family number is 2, and the trunk number is 1 (trunk number has no direct match in the Chassis MIB).

Chassis MIB structure

The Chassis MIB represents a chassis as single managed node that consists of the buses, slots, and devices installed in a chassis. There are five major tables within the Chassis MIB:

Table	Description
Chassis Configuration	Provides information about the chassis.
Bus Segment	Provides information about the bus segments in this chassis.
Board Access by bus slot	Provides an index into the Bus Segment table and the Board table.
Board	Provides information about each board.
Board Access by backplane	Not implemented. (Reserved for future use.)

The following illustration shows how the tables in the Chassis MIB are related to one another:

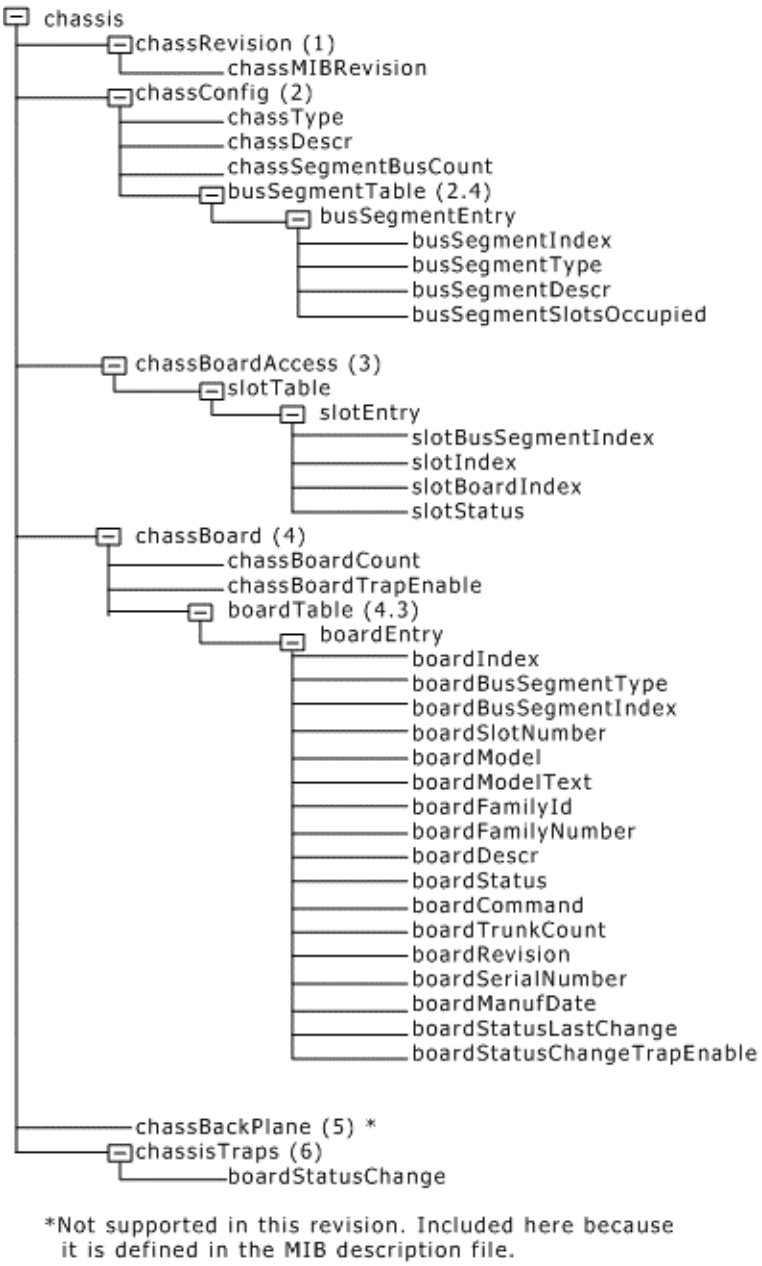


Chassis MIB table relations

This example shows that a busSegmentCount value of 5 in the Chassis Configuration table results in five entries in the Bus Segment table. Bus segment 3 has four occupied slots, so there are four entries in the Board Access table for that bus segment. Each entry in the Board Access table has a boardIndex field, whose value is an index into the Board table for that board.

Two fields in the Board table match parts of **dsx1CircuitIdentifier** in the Trunk MIB. For more information about how the Chassis MIB and Trunk MIB can be used together, see Using the chassis MIB.

The sequence of objects in the Chassis MIB (with relative OIDs for table objects) is shown in the following illustration:



Chassis MIB object

Chassis MIB Configuration table

The Chassis Configuration table contains the following information:

- Type of chassis
- Description
- Number of bus segments within the chassis

Information about each bus segment, such as type of bus segment, description, and number of occupied slots, is contained within an object block that makes up the Bus Segment table. The objects in this table are under the **chassConfig** table of the Chassis MIB. Values are assigned to these objects by the NMS Chassis agent.

The objects in the Chassis Configuration table are:

Object	Description
chassConfig	Top of the table.
chassType	Chassis type.
chassDescr	Description of the chassis.
chassSegmentBusCount	Number of bus segments within the chassis.

Chassis MIB Bus Segment table

The Bus Segment table contains information about each bus segment, such as type of bus segment, description, and number of occupied slots. There can only be one ISA bus segment, but there can be many PCI (or compact PCI) bus segments.

Each **busSegmentEntry** object is identified by a **busSegmentIndex** object, whose value is assigned by the NMS Chassis agent.

busSegmentEntry objects are added to the table when a board is added to a new bus segment. If all boards are extracted, that bus segment will be deleted from the table.

Note: ISA boards are not supported by Natural Access version 4.0 and later.

The objects in the Bus Segment table are:

Object	Description
busSegmentTable	Starts the Bus Segment table.
busSegmentEntry	Starts a row of the Bus Segment table.
busSegmentIndex	Number of this row in the Bus Segment table.
busSegmentType	Bus type.
busSegmentDescr	Describes the bus segment.
busSegmentSlotsOccupied	Number of occupied slots in this bus segment.

Chassis MIB Board Access table

The Board Access table simplifies access to the Board table's variables. The Board table can be sequentially accessed by using a series of **get-next** commands starting from the beginning of the table. But this type of access is not convenient for all types of queries. For example, an application may be interested in the trunk count of all boards on PCI segment 2. Using **get-next** commands, the application must traverse the entire table in order to ensure that all boards are accounted for. With the index table, the application only needs to find the first entry with the **busSegmentNumber** that matches PCI segment 2, and the rest of that segment's boards will be listed next.

The Board Access table provides an index into the Board table that allows an application to directly access specific boards using **get** commands, based on the board's bus type, bus segment number, or logical slot number.

Object	Action
Bus Type	Examine the Bus Segment table to find the bus segment type you are interested in. Look for that entry's **busSegmentIndex** value in the Board Access table, and use each matching entry's **slotBoardIndex** value to find the entry in the Board table.
Bus Segment Number	Find the slotBusSegmentNumber in the Board Access table, and use that entry's **slotBoardIndex** value to find the entry in the Board table.
Slot Number	Find the **slotIndex** value for a chosen bus segment, and use that row's boardIndex value to index into the Board table.

The objects in the Board Access/Slot table include:

Object	Description
chassBoardAccess	Starts the Board Access table.
slotTable	Starts the rows of the Board Access table.
slotEntry	Starts a row in the Board Access table.
slotBusSegmentIndex	Number of the bus segment this board is in.
slotIndex	Logical slot index of a board in the bus segment.
slotBoardIndex	Index into the Board table for this bus segment.
slotStatus	Status of the slot (hot swap status).

Chassis MIB Board table

Each **boardEntry** object in the Board table contains information about a single board in the chassis. This group of objects includes the board model, a textual description of the board model, a family identifier, the board's status, the trunk count, the board revision, the board's serial number, and the board's date of manufacture. Each **boardEntry** is identified by the **boardIndex** object, whose value is assigned by the NMS Chassis agent. New **boardEntry** objects are added to this table and configured for NMS OAM when a board is added to the chassis.

Entries in the Board table are removed if a board physically is extracted. If a board is inserted, a new entry will be added to the Board table using the next free index. Whenever a board is inserted or extracted, a trap is sent (if traps have been enabled).

The objects in the Board table are:

Object	Description
chassBoard	Start of the board descriptions.
chassBoardCount	Number of boards in the chassis.
chassBoardTrapEnable	Sets the default value for the **boardStatusChangeTrapEnable** object for the entries in this table.
boardTable	Starts the Board table.
boardEntry	Starts a row of the Board table.
boardIndex	Number of this row in the Board table.
boardBusSegmentType	Type of bus segment.
boardBusSegmentNumber	Number of the bus segment this board is in.
boardSlotNumber	Number of the slot.
boardModel	Model of this board (numeric).
boardModelText	Model of this board (textual).
boardFamilyId	Family of the board.
boardFamilyNumber	Logical number of the board.
boardDescr	Board description.
boardStatus	Board status (online or offline).
boardCommand	Turns the board on or off.
boardTrunkCount	Number of trunks on this board.
boardRevision	Board revision.
boardSerialNumber	Board serial number.
boardManufDate	Date the board was manufactured.
boardStatusLastChange	When the status of the board last changed.
boardStatusChangeTrapEnable	Determines if boardStatusLastChange traps will be generated.

Chassis Trap group

The Chassis MIB Trap group is used by the agent to specify trap information. It has a valid object identifier, but does not contain usable information for developers.

Using the Chassis MIB object reference

The following sections describe the objects in this MIB. A typical object description includes:

Syntax	The datatype of the object is shown. SNMP data types include:	
	Integer	16-bit signed.
	DisplayString	ASCII text.
	Gauge	Positive integer from 0 to 4294967295 (2^{32} - 1).
	Object	Another object type from this MIB.
	TimeStamp	Positive integer from 0 to 4294967295 (2^{32} - 1).
	TruthValue	Integer value where 1 is True and 2 is False.
Access	The type of access allowed for this object. Options are:	
	Read-only	This object can not be modified by SNMP.
	Read-write	SNMP can configure this object.
OID	The OID defines the path from the root to this object. All OIDs start with ***p***, where ***p*** is is 1.3.6.1.4.1.2628.2.2 (the OID for the Chassis MIB).	
Details	Describes the object.	
Configuration	Describes how to configure the object.	
Example	Shows an example of how the object is used.	

The source from which the NMS MIBs was compiled is supplied with the software, in ASN1 format text files. These files can be found in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX). NMS SNMP MIBs were compiled using the following files:

- Chassis MIB: *chassis-mib.txt*
- Trunk MIB: *chassis-mib.txt*
- Software Revision MIB: *softrev-mib.txt*
- OAM Database MIB: *oamdatabase-mib.txt*

Read the appropriate file using the Windows Console Management function to display the SNMP information for the proprietary agent.

boardBusSegmentNumber

Number of the bus segment in which this board is installed. Corresponds to the busSegmentIndex in the Bus Segment table.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.3.***n***, where ***n*** is the entry number.

Configuration

Not applicable.

boardBusSegmentType

Indicates the Bus type for a particular board.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.2.***n***, where ***n*** is the entry number.

Details

Acceptable values are as follows:

Value	Type
1	ISA bus (not supported in Natural Access versions 4.0 and later)
2	PCI bus

Configuration

None.

boardCommand

Turns the board on or off.

Syntax

Integer

Access

Read-write

OID

p.4.3.1.11.***n***, where ***n*** is the entry number.

Details

Setting the value of this object turns the board on or off. You can then check the boardStatus object to see when the command has completed. Valid values are:

Value	Action
1	On (same as closing the handles in physical Hot Swap).
2	Off (same as opening the handles in physical Hot Swap).

An operation must complete (not be in the pending state) before issuing a second command.

Note: The value of this object only applies to CompactPCI boards.

Configuration

Not applicable.

boardDescr

Provides a textual description of the board (optional).

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.9.***n***, where ***n*** is the entry number.

Details

The default value is the empty string "". A sample description is:

```
Reserved for Fax Apps Only
```

The entry in the *snmp.cfg* file is:

```
BoardDesc = x, Description
```

where ***x*** = **boardFamilyNumber**.

Configuration

This object is configured by editing *snmp.cfg* before starting the NMS Chassis agent.

boardEntry

Starts the series of objects for a row in the Board table.

Syntax

Object

Access

Not accessible

OID

p.4.3.1

Details

A **boardEntry** variable is added to the Board table whenever a board is inserted, and removed from the table when a board is extracted.

Configuration

Not applicable.

boardIndex

Identifies a row in the Board table that is defined by this **boardEntry** block of objects.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.1.***n***, where ***n*** is the entry number.

Details

If the board is turned off using Hot Swap, the board index and values will still exist, but the Trunk MIB will not see any lines. When the board is extracted, the board index will also be removed.

A board that is inserted will use the next available index number.

Configuration

Not applicable.

boardFamilyId

Indicates the board family.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.7.***n***, where ***n*** is the entry number.

Details

Expected values are shown in the following table:

Value	Board family
1	other (default)
2	AG/CG
3	QX
4	TX (not supported)
5	CX

Configuration

Not applicable.

boardFamilyNumber

Indicates the logical number of a board in this family.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.8.***n***, where ***n*** is the entry number.

Details

This value matches the number in the *oamsys.cfg* file, and the board number in the **dsx1CircuitIdentifier** in the Trunk MIB.

For Natural Access 3.x, the logical board number matches the number in the *ag.cfg* file, and the board number in the **dsx1CircuitIdentifier** in the Trunk MIB.

Configuration

Not applicable.

boardManufDate

Indicates the board's manufacturing date.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.15.***n***, where ***n*** is the entry number.

Configuration

Not applicable.

Example

```
week 5 00
```

boardModel

Indicates the board type.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.5.***n***, where ***n*** is the entry number.

Details

Supported board types (for Natural Access 4.0 or later) include:

Value	Board Types
1	other
2	QX 2000/80-1L, QX 2000/80-4L, QX 2000/100-4L, QX 2000/200-4L
3	AG 2000, AG 2000 BRI, AG 2000 E&M, AG 2000 VTG, AG 2000C
4	AG Dual E1, AG Dual T1
5	AG CompactPCI Quad E1, AG CompactPCI Quad T1, AG Quad Connect E1, AG Quad Connect T1, AG Quad E1, AG Quad T1, AG QuadDual E1, AG QuadDual T1
6	AG 4000 Single E, AG 4000 Single T, AG 4000 Dual E1, AG 4000 Dual T1, AG 4000 Quad E1, AG 4000 Quad T1, AG 4000 E1, AG 4000 T1, AG 4000C Dual E1, AG 4000C Dual T1, AG 4000C Quad E1, AG 4000C Quad T1, AG 4000C E1, AG 4000C T1
7	CG 6000, CG 6000 Quad, CG 6000C Quad, CG 6100C, CG 6100C 16, CG 6100C 8
8	CX 2000-16, CX 2000-32, CX 2000C-16, CX 2000C-32, CX 2000C-48, CX 2000

Configuration

Not applicable.

boardModelText

Provides a textual description of the board.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.6.***n***, where ***n*** is the entry number.

Details

Acceptable values are:

Board type	Value
AG	AG_2000 AG_2000_BRI AG_2000_ENM AG_2000_VTG AG_2000C AG_4000_1E1 AG_4000_1T1 AG_4000_2E1 AG_4000_2T1 AG_4000_4E1 AG_4000_4T1 AG_4000_E1 AG_4000_T1 AG_4000C_2E1 AG_4000C_2T1 AG_4000C_4E1 AG_4000C_4T1 AG_4000C_E1 AG_4000C_T1 AG_Dual_E1 AG_Dual_T1 AG_CPCI_Quad_E1 AG_CPCI_Quad_T1 AG_Quad_Connect_E1 AG_Quad_Connect_T1 AG_Quad_E1, AG_Quad_T1 AG_QuadDual_E1 AG_QuadDual_T1
QX	QX 2000/80-1L QX 2000/80-4L QX 2000/100-4L QX 2000/200-4L

Board type	Value
CG	CG_6000 CG_6000_Quad CG_6000C_Quad CG_6100C CG_6100C_16 CG_6100C_8
CX	CX 2000-16 CX 2000-32 CX 2000C-16 CX 2000C-32 CX 2000C-48 CX 2000

The value of this object corresponds to the textual part of the **dsx1CircuitIdentifier** object in the Trunk MIB.

Configuration

Not applicable.

boardRevision

Returns the board revision.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.13.***n***, where ***n*** is the entry number.

Configuration

Not applicable.

boardSerialNumber

Indicates the board's serial number.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.14.***n***, where ***n*** is the entry number.

Configuration

Not applicable.

Example

```
123456754
```

boardSlotNumber

Indicates the slot of the bus segment in which the board is installed.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.4.***n***, where ***n*** is the entry number.

Configuration

Not applicable.

boardStatus

Indicates the board status.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.10.***n***, where ***n*** is the entry number.

Details

Expected values include:

Value	Status
1	Online.
2	Online Pending. The board coming online is in progress.
3	Failed.
4	Offline. The board is turned off, and can be extracted.
5	Offline Pending. Waiting for activity to stop (which can be time consuming).
6	Extracted.

Configuration

Not applicable.

boardStatusChangeTrapEnable

Determines whether traps are generated for the board.

Syntax

Integer

Access

Read-write

OID

p.4.3.1.17.***n***, where ***n*** is the entry number.

Details

Enabling this object will cause traps to be sent to the management station, and update the **boardStatusLastChange** object. Valid values are:

Value	Description
1	Enabled
2	Disabled (default)

Configuration

Not applicable.

boardStatusLastChange

Provides the time stamp of when the status of the board last changed.

Syntax

TimeTicks

Access

Read-only

OID

p.4.3.1.16.***n***, where ***n*** is the entry number.

Configuration

Not applicable.

boardTable

Starts a sequence of **boardEntry** objects, which defines the rows of the Board table.

Syntax

Object

Access

Not accessible

OID

p.4.3

Details

The Board table contains configuration and status information for all boards in the chassis.

This table consists of exactly ***n*** * ***m*** **boardEntry** entries where:

n = **chassSegmentBusCount**

and

m = **busSegmentSlotsOccupied**

Configuration

Not applicable.

boardTrunkCount

Indicate the number of trunks on this board.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.12.***n***, where ***n*** is the entry number.

Details

0 means no trunks.

Configuration

Not applicable.

busSegmentDescr

Proves a textual description of the Bus Segment.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.4.1.3.***n***, where ***n*** ranges from 1 to the number of bus segments

Details

The value can be one of the following:

- PCI bus segment number: 0
- ISA.

Configuration

Not applicable.

busSegmentEntry

Starts a row in the Bus Segment table.

Syntax

Object

Access

Read-only

OID

p.2.4.1

Details

A **busSegmentEntry** object block is added to the Bus Segment table when a board is inserted into a slot on a previously unpopulated bus. This object and the associated block of objects is removed when a board is extracted from the bus.

Configuration

Not applicable.

busSegmentIndex

Identifies this row in the Bus Segment table.

Syntax

Integer

Access

Read-only

OID

p.2.4.1.1.***n***, where ***n*** ranges from 1 to the number of bus segments

Details

Internally assigned by the agent. The value range is
1<= ***n*** <= **chassSegmentBusCount**.

Configuration

Not applicable.

busSegmentSlotsOccupied

Indicates the number of occupied slots in this entry's bus segment.

Syntax

Integer

Access

Read-only

OID

p.2.4.1.4.***n***, where ***n*** ranges from 1 to the number of bus segments

Details

This value determines the number of entries in the Board Access by slot table for this bus segment. Updated by the agent when a board is inserted into or extracted from the associated bus segment.

Configuration

Not applicable.

busSegmentTable

Starts a sequence of **busSegmentEntry** objects that compose a row in the Bus Segment table.

Syntax

Object

Access

Not accessible

OID

p.2.4

Details

The Bus Segment table row is composed of exactly ***n*** **busSegmentEntry** objects, where ***n*** = **chassSegmentBusCount**.

Configuration

Not applicable.

busSegmentType

Indicates the bus type.

Syntax

Integer

Access

Read-only

OID

p.2.4.1.2.***n***, where ***n*** ranges from 1 to the number of bus segments

Details

Valid values are:

Value	Type
1	ISA bus (not supported in Natural Access versions 4.0 and later)
2	PCI or CompactPCI bus

Configuration

Not applicable.

chassBoard

Starts the series of variables that constitutes the Board table.

Syntax

Object

Access

Not accessible

OID

p.4

Configuration

Not applicable.

chassBoardAccess

Starts the sequence of objects that make up the Board Access table in the Chassis MIB.

Syntax

Object

Access

Not accessible

OID

p.3

Configuration

Not applicable.

chassBoardCount

Indicates the number of boards currently installed in the chassis.

Syntax

Integer

Access

Read-only

OID

p.4.1

Details

Corresponds to the number of **boardEntry** objects, which starts a row in the table.

Incremented when a board is inserted, and decremented when a board is extracted.

Configuration

Not applicable.

chassBoardTrapEnable

Sets the default value for the **boardStatusChangeTrapEnable** object for the entries in this table.

Syntax

Integer

Access

Read-write

OID

p.4.2

Details

Valid values are:

Value	Description
1	Enabled
2	Disabled (default)

Configuration

Not applicable.

chassConfig

Starts a group of three objects that describe the chassis.

Syntax

Object

Access

Not accessible

OID

p.2

Configuration

Not applicable.

chassDescr

Provides a textual description of the chassis.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.2

Details

The default value is the empty string. An example string could be:

```
CPCI chassis; location: Floor 2 West Wing
```

The chassis description is specified in the *snmp.cfg* configuration file. The keyword and value are:

```
ChassisDescription = Description
```

Configuration

This object is configured by editing *snmp.cfg* before starting the NMS Chassis agent.

chassMIBRevision

Indicates the revision ID of the Chassis MIB.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.1

Details

The value of the revision ID (set by NMS) is Wednesday, May 24, 2000. This object identifies the MIB, so the management station can tell if it is configured for the correct MIB.

Configuration

Not applicable.

Note: In versions of the software prior to 2001-1, the OID for this node was ***p***.1.

chassRevision

Starts a group for the revision field.

Syntax

Object

Access

Not accessible

OID

p.1

Configuration

Not applicable.

chassSegmentBusCount

Indicates the number of known bus segment types (ISA bus and/or PCI segments) in the chassis.

Syntax

Integer

Access

Read-only

OID

p.2.3

Details

The default value is 0, there are no boards in the chassis. There can only be one ISA bus segment, but there can be more than one PCI bus segment. This value determines how many entries there will be in the Bus Segment table.

This object is updated when a board is inserted into a slot in a previously unpopulated bus segment and is recognized by the agent, or when a board is removed.

Configuration

Not applicable.

chassType

Indicates the chassis type.

Syntax

Integer

Access

Read-only

OID

p.2.1

Details

Valid values include:

Value	Description
1	Unknown chassis (default).
2	CompactPCI chassis.
3	Generic PC chassis.
4	Generic Sun chassis.

The chassis type is specified in the *snmp.cfg* configuration file. The keyword and value are:

```
ChassisType = [1|2|3|4]
```

Configuration

This object is configured by editing *snmp.cfg* before starting the NMS Chassis agent.

slotBoardIndex

Provides an index into the Board table for the board in the associated bus segment and logical slot.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.3.***n***.***m***, where ***n*** is the index of this segment in the Bus Segment table, and ***m*** is the index of the slot in this segment.

Details

This value matches **boardIndex** in the Board table.

Configuration

Not applicable.

slotBusSegmentIndex

Identifies the bus segment to which this slot belongs.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.1.***n***.***m***, where ***n*** is the index of this segment in the Bus Segment table, and ***m*** is the index of the slot in this segment.

Details

This value corresponds to **busSegmentIndex** in the Bus Segment table.

Configuration

Not applicable.

slotEntry

Starts a row in the Board Access table.

Syntax

Object

Access

Not accessible

OID

p.3.1.1

Details

A **slotEntry** block of objects is added to the Board Access table whenever a board is inserted, and removed when a board is extracted.

Objects belonging to this entry belong to a Doubly Indexed table, and are accessed using an OID of:

p.3.1.1.***x***.***n***.***m***

where

x = the object of the group and the column number of this row.

n = the bus segment number (**slotBusSegmentIndex**), the first index.

m = the slot number (**slotIndex**), the second index.

For more information about Doubly Indexed tables, see Accessing MIB objects.

Configuration

Not applicable.

slotIndex

Provides the logical index of a slot within the bus segment.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.2.***n***.***m***, where ***n*** is the index of this segment in the Bus Segment table, and ***m*** is the index of the slot in this segment.

Configuration

Not applicable.

slotStatus

Indicates the Hot Swap status, from the Hot Swap state machine.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.4.***n***.***m***, where ***n*** is the index of this segment in the Bus Segment table, and ***m*** is the index of the slot in this segment.

Details

Acceptable values are:

Value	Status
1	Online.
2	OnLine Pending. The board coming online is in progress.
3	Failed.
4	Offline. The board is turned off, and can be extracted.
5	Offline Pending. Waiting for activity to stop, which can be time consuming.
6	Extracted.

For Natural Access 3.x, acceptable values are:

Value	State	Description
1	P0	Board is not present in the slot.
2	S0	Board is present, but not configured for Hot Swap.
3	S1	Hot Swap device instance is not started.
4	S1F	The Hot Swap device instance failed to start.
5	S1I	The Hot Swap device instance is started, but the board is not prepared for use (insertion is in progress).
6	S1B	Board is being prepared (insertion is in progress).
7	S1BF	Board preparation failed.
8	S2	Board is ready.
9	S2R	Extraction in progress.

Configuration

Not applicable.

slotTable

Starts a sequence of **slotEntry** objects that make up the Board Access table.

Syntax

Object

Access

Not accessible

OID

p.3.1

Details

The Board Access table provides an index into the Board table (**slotBoardIndex**), allowing direct access to a specific board based on its bus characteristics.

This table is composed of exactly ***n*** * ***m*** **slotEntry** objects where:

n = **chassSegmentBusCount** (from the Chassis table)

and

m = **busSegmentSlotsOccupied** (from the Bus Segment table)

Configuration

Not applicable.

Trunk MIB

Trunk MIB structure and limitations

The NMS Communications implementation of the Trunk MIB (RFC 2495). The organization of the tree, detailed descriptions of the nodes, and the available functions are also provided. Compliance to the Trunk MIB is also detailed.

All the boards in a chassis are represented as one managed node. Each trunk is represented by a numerical index, which is generated by sequentially numbering the trunks on all the boards.

RFC 2495 defines the near end and far end of each DS1 interface. The near end is the interface on the board that the agent is monitoring. The far end is the remote end of the trunk connected to that interface. Support is defined for the near end.

The RFC 2495 MIB defines the following groups:

Group	Description
DS1 Near End Group	Contains configuration information about the DS1 interfaces, and statistics collected from the near end interface.
DS1 Far End Group	Optional and not supported.
Fractional Table	Optional and not supported.
Channel Mapping Table	Optional and not supported.
Trap Group	Enables a trap to be sent when the status of the interface changes.

Known limitations

RFC 1573 defines an ifTable for all the interfaces in the system as part of MIB2. The ifTable is not accessible by the NMS SNMP agent. Therefore some portions of the RFC 2495 MIB are not supported.

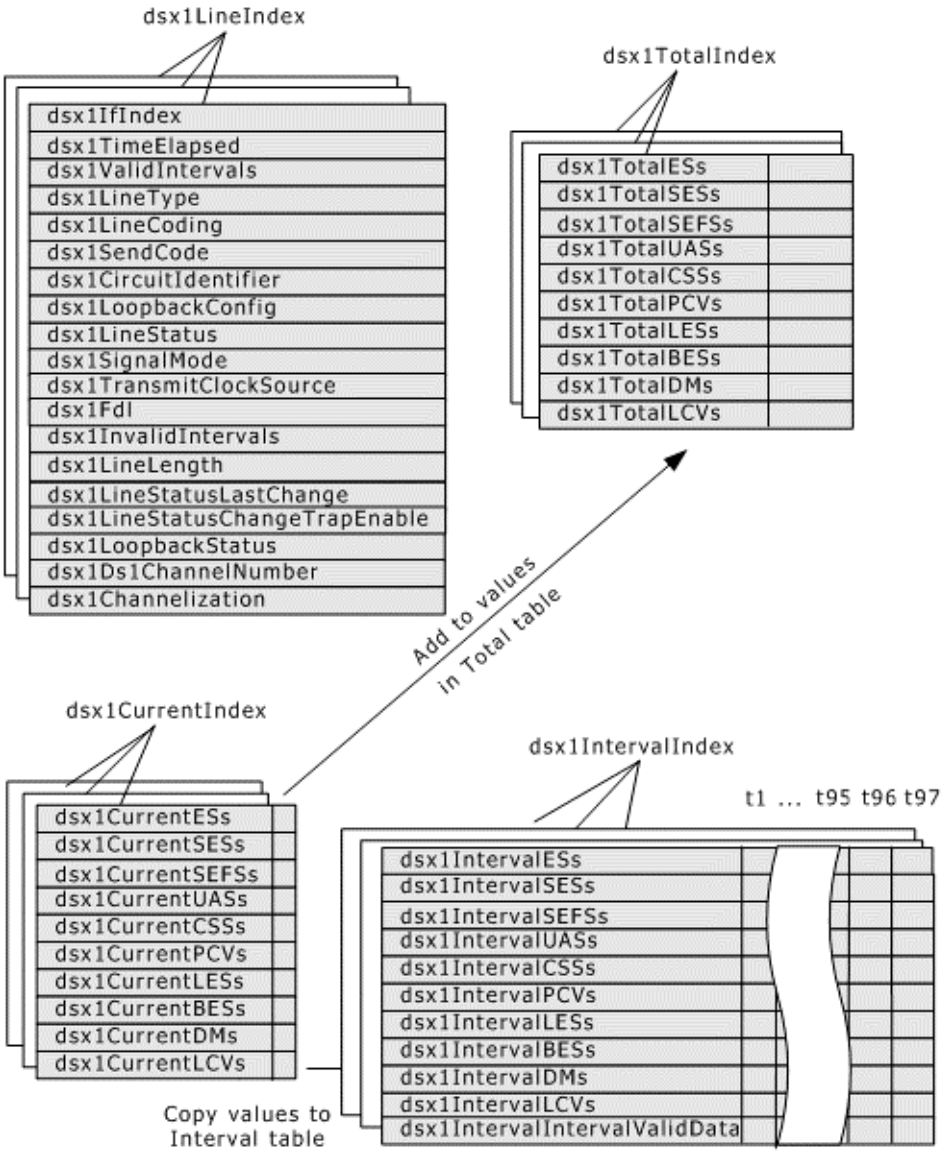
The dsx1ChannelMappingTable is also not available.

The DS1 Near End Group consists of four tables:

Table	Description
Configuration	Contains information about each DS1 interface such as the number of bits per second that the circuit can reasonably carry, variety of Zero Code Suppression, and the vendor's circuit identifier.
Current	Contains statistics for the current 15-minute interval.
Interval	Contains statistics collected by each DS1 interface for the last 24 hours of operation. The past 24 hours are broken into 96 15-minute intervals. After 24 hours, the next interval pushes the oldest one out of the table.
Total	Contains the cumulative sum of the statistics for the period of time since this MIB was first started. Each field in this table contains the sum of the fields in the Current table for a particular interface.

The information in the Current table refreshes continuously. Every 15 minutes, the current table's contents are copied to the Interval table, and the sum of values from the Current table are added to the Total table. The Total table never resets, so the values are sums from the first time you started the DTM agent.

The following illustration shows how the tables for DS1 relate:



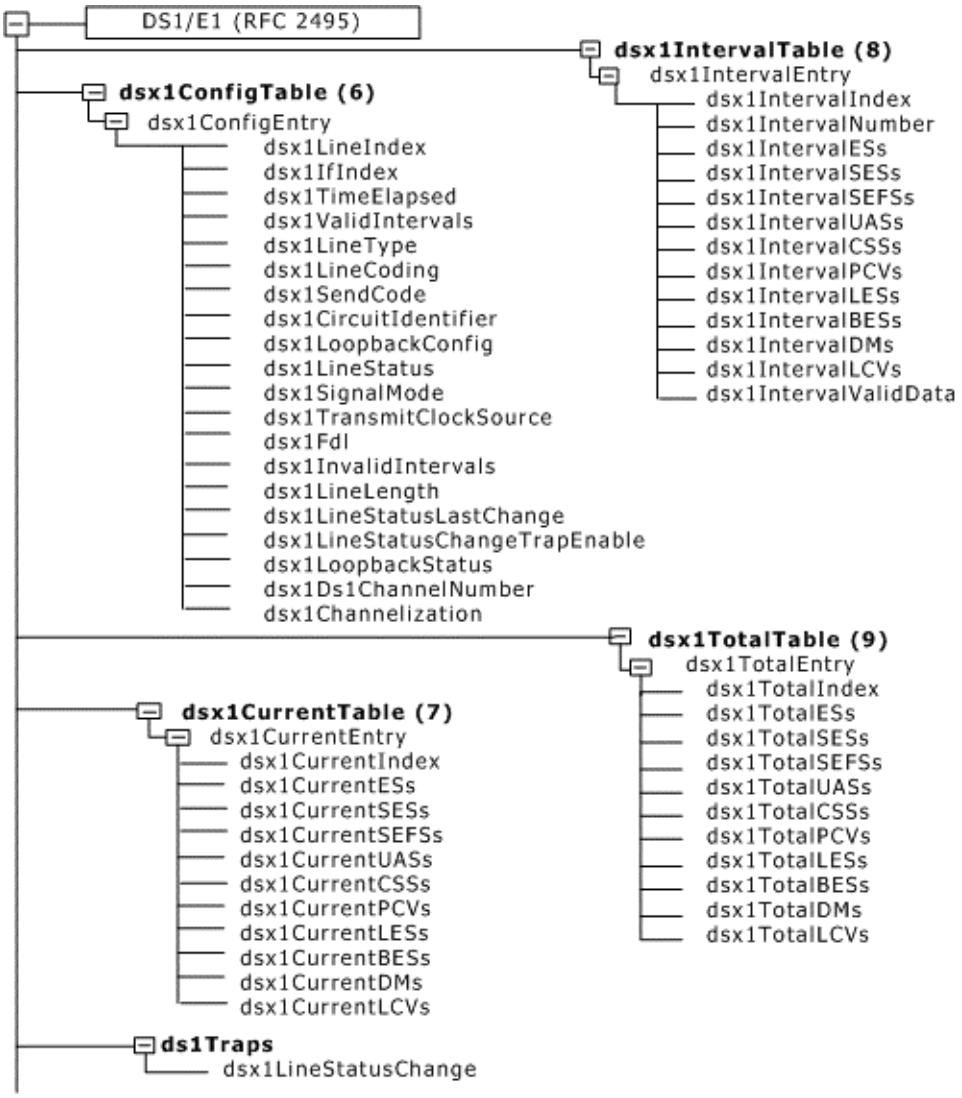
2495 MIB table relationships

The previous illustration shows a logical view of the tables for three DS1 interfaces. The Configuration table has an entry for each DS1 interface, which is identified by dsx1LineIndex. This index corresponds to the index in the other tables, such that all table entries with the same index number are for the same DS1 interface. Three DS1 interfaces are represented, so each table has three pages. Each column of values is started by an entry object.

Every 15 minutes, the values in the Current table are copied to the next available time slot (for example, if t1 was filled 15 minutes ago, t2 will be filled next) in the Interval table. The Current table values are added to the values in the Total table, which continue to add up until the agent is restarted.

If the Interval table is full when a new timeslot is added to the table, the last time slot (t97) is discarded, and the rest of the timeslots slide forward to make room for the new t1 timeslot.

The following illustration shows a tree view of the sequence of objects in the Trunk MIB:



RFC 2495 MIB tree

Trunk MIB node tables

The Trunk MIB includes the following node tables:

- Configuration table
- Interval table
- Current table
- Total table
- Trap group

The source from which the NMS MIBs was compiled is supplied with the software, in ASN1 format text files. These files can be found in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX). The Trunk MIB was compiled from *trunk-mib.txt*. Read this file using the Windows Console Management function to display the SNMP information for this proprietary agent.

Trunk MIB Configuration table

The following table summarizes each object in the Trunk MIB Configuration table, otherwise known as the **dsx1ConfigEntry** block of variables:

Object	Type	Description
dsx1LineIndex	Integer	Identifies a DS1 interface in this managed node.
dsx1IfIndex	Integer	Same as dsx1LineIndex.
dsx1TimeElapsed	Integer	Time of current measurement period.
dsx1ValidIntervals	Integer	Number of 15 minute measured intervals.
dsx1LineType	Integer	Type of DS1 interface.
dsx1LineCoding	Integer	Type of Zero Code Suppression for this interface.
dsx1SendCode	Integer	Type of code in the interface.
dsx1CircuitIdentifier	DisplayString	Transmission vendor's circuit identifier.
dsx1LoopbackConfig	Integer	Loopback configuration.
dsx1LineStatus	Integer	Interface status.
dsx1SignalMode	Integer	Circuit's signal mode.
dsx1TransmitClockSource	Integer	Source of the transmit clock.
dsx1Fdl	Integer	Describes the facilities data link.
dsx1InvalidIntervals	Integer	Indicates the number of intervals with invalid data, which will always be 0 (not supported).
dsx1LineLength	Integer	Indicates the length of the DS1 line in meters, which will always be 0 (not supported).
dsx1StatusLastChange	TimeStamp	Indicates the time when the status of the interface last changed.
dsx1LineStatusChangeTrapEnable	Integer	Determines whether traps should be generated for this interface.
dsx1LoopbackStatus	Integer	Represents the current state of the loopback on the DS1 interface.
dsx1Ds1ChannelNumber	Integer	Represents the channel number of the DS1/E1 on its parent DS2/E2 or DS3/E3.
dsx1Channelization	Integer	Indicates whether this DS1/E1 is channelized or not.

Trunk MIB Interval table

Most of the variables in the Interval table have descriptions that match a variable with a name similar to one in the Current table. For example, **dsx1IntervalESs** in the Interval table matches **dsx1CurrentESs** in the Current table. Both these variables contain the number of errored seconds for a 15 minute interval. The following table shows the matching variables from the two tables:

Current table	Interval table
dsx1CurrentIndex	**dsx1IntervalIndex**
dsx1CurrentESs	**dsx1IntervalESs**
dsx1CurrentSESs	**dsx1IntervalSESs**
dsx1CurrentSEFSs	**dsx1IntervalSEFSs**
dsx1CurrentUASs	**dsx1IntervalUSASs**
dsx1CurrentCSSs	**dsx1IntervalCSSs**
dsx1CurrentPCVs	**dsx1IntervalPCVs**
dsx1CurrentLESs	**dsx1IntervalLESs**
dsx1CurrentBESs	**dsx1IntervalBESs**
dsx1CurrentDMs	**dsx1IntervalDMs**
dsx1CurrentLCVs	**dsx1IntervalLCVs**
	dsx1IntervalNumber
	dsx1IntervalValidData

The Interval table is a doubly indexed table. For information about accessing a doubly indexed table, see Accessing MIB objects.

There are two variables that do not match entries in the Current table: **dsx1IntervalNumber** and **dsx1IntervalValidData**. These are explained in the following sections.

Trunk MIB Current table

The following table summarizes each object in the Trunk MIB Current table, otherwise known the **dsx1CurrentEntry** block of variables:

Object	Syntax	Description
dsx1CurrentIndex	Integer	Number of the DS1 interface.
dsx1CurrentESs	Gauge	Number of errored seconds.
dsx1CurrentSESs	Gauge	Number of severely errored seconds.
dsx1CurrentSEFSs	Gauge	Number of severely errored framing seconds.
dsx1CurrentUASs	Gauge	Number of unavailable seconds.
dsx1CurrentCSSs	Gauge	Number of controlled slip seconds.
dsx1CurrentPCVs	Gauge	Number of path coding violations.
dsx1CurrentLESs	Gauge	Number of interface errored seconds.
dsx1CurrentBESs	Gauge	Number of bursty errored seconds.
dsx1CurrentDMs	Gauge	Number of degraded minutes.
dsx1CurrentLCVs	Gauge	Number of line code violations.

Trunk MIB Total table

The Total table contains the sum of the statistics that the RFC 2495 MIB has kept for the managed node since the agent for this MIB first started. All the descriptions match the variables in the Current table, except that, for the Total table, the values are for the total time the MIB has been written to, and for the Current table the values are for the current 15 minute period.

The names of the variables in the two tables match, except that one starts with **dsx1Current**, and the other starts with **dsx1Total**. For example, **dsx1CurrentESs** matches **dsx1TotalESs**.

Refer to Current table for descriptions of the Total table variables.

Trunk MIB Trap group

The Trunk MIB Trap group has one object, dsx1LineStatusChange, that determines whether a trap is sent when the status of the interface changes.

Using the Trunk MIB object reference

The following sections describe the objects in this MIB. A typical object description includes:

Syntax	The datatype of the object is shown. SNMP data types include:	
	Integer	16-bit signed.
	DisplayString	ASCII text.
	Gauge	Positive integer from 0 to 4294967295 (2^{32} - 1).
	Object	Another object type from this MIB.
	TimeStamp	Positive integer from 0 to 4294967295 (2^{32} - 1).
	TruthValue	Integer value where 1 is True and 2 is False.
Access	The type of access allowed for this object. Options are:	
	Read-only	This object can not be modified by SNMP.
	Read-write	SNMP can configure this object.
OID	The OID defines the path from the root to this object. All OIDs start with ***p***, where ***p*** is is 1.3.6.1.4.1.2628.2.2 (the OID for the Chassis MIB).	
Details	Describes the object.	
Configuration	Describes how to configure the object.	
Example	Shows an example of how the object is used.	

The source from which the NMS MIBs was compiled is supplied with the software, in ASN1 format text files. These files can be found in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX). NMS SNMP MIBs were compiled using the following files:

- Chassis MIB: *chassis-mib.txt*
- Trunk MIB: *chassis-mib.txt*
- Software Revision MIB: *softrev-mib.txt*
- OAM Database MIB: *oamdatabase-mib.txt*

Read the appropriate file using the Windows Console Management function to display the SNMP information for the proprietary agent.

dsx1Channelization

Indicates whether this DS1/E1 is channelized or not.

Syntax

Integer

Access

Read-write

OID

p.6.1.20.***n***, where ***n*** = the index number of the DS1 interface.

Details

Possible values are:

Value	Description
1	disabled
2	enabledDs0
3	enabledDs1

Note: The NMS SNMP agent always returns enabledDs0 because NMS boards are always channelized.

Configuration

Not applicable.

dsx1CircuitIdentifier

Provides the circuit identifier.

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.6.1.8.***n***, where ***n*** = the index number of the DS1 interface.

Details

The circuit identifier is represented by:

name-of-board_board-number_trunk-number

where:

name-of-board is one of the following:

Board type	Value
AG	AG_Dual_E1 AG_Dual_T1 AG_Quad_E1 AG_Quad_T1 AG_Quad_Connect_E1 AG_Quad_Connect_T1 AG_CPCI_Quad_E1 AG_CPCI_Quad_T1 AG_4000_Single_E1 AG_4000_Single_T1 AG_4000_Dual_E1 AG_4000_Dual_T1 AG_4000_Quad_E1 AG_4000_Quad_T1 AG_4000C_Dual_E1 AG_4000C_Dual_T1 AG_4000C_Quad_E1 AG_4000C_Quad_T1
CG	CG_6000C_Quad

For Natural Access 3.x, acceptable values are:

Board type	Value
AG	Ag4000-Dual-T1 Ag4000-Quad-T1 Ag4000-T1 AgT1 Ag-Dual-T1 Ag-Quad-T1-Board Ag-Quad-T1-Connect Ag4000-Dual-E1 Ag4000-E1 Ag4000-Quad-E1 AgE1 Ag-Dual-E1 Ag-Quad-E1-Board Ag-Quad-E1-Connect

board-number is a two digit number, starting at 0.

trunk-number is a two digit number, starting at 0.

For example, board 0, trunk 3, is AG_4000_Single_E1_00_03.

The circuit identifier matches the **boardModelText** object in the Chassis MIB, which allows cross referencing DS1 interfaces between the two MIBs.

Configuration

Not applicable.

dsx1ConfigTable

Starts a sequence of **dsx1ConfigEntry** objects, each representing a DS1 interface.

Syntax

Object

Access

Not accessible

OID

p.6

Configuration

Not applicable.

dsx1ConfigEntry

Starts a sequence of 13 objects that describe the configuration of the DS1 interface identified by **dsx1LineIndex**.

Syntax

Not applicable

Access

Not accessible

OID

p.6.1

Configuration

Not applicable.

dsx1CurrentBESs

Indicates the number of Bursty Errored Seconds (BESs).

Syntax

Gauge

Access

Read-only

OID

p.7.1.9.***n***, where ***n*** = the index number of the DS1 interface.

Details

A Bursty Errored Second (also known as Errored Second type B) is a second with fewer than 320 and more than 1 Path Coding Violation error events, no Severely Errored Frame defects and no detected incoming AIS defects. Controlled slips are not included in this parameter.

This is not incremented during an Unavailable Second (**dsx1CurrentUASs**).

Configuration

Not applicable.

dsx1CurrentCSSs

Indicates the number of Controlled Slip Seconds.

Syntax

Gauge

Access

Read-only

OID

p.7.1.6.***n***, where ***n*** = the index number of the DS1 interface.

Details

A Controlled Slip Second is a one-second interval containing one or more controlled slips.

A Controlled Slip is the replication or deletion of the payload bits of a DS1 frame. A Controlled Slip may occur when there is a difference between the timing of a synchronous receiving terminal and the received signal. A Controlled Slip does not cause an Out of Frame error.

Configuration

Not applicable.

dsx1CurrentDMs

Indicates the number of Degraded Minutes (DMs).

Syntax

Gauge

Access

Read-only

OID

p.7.1.10.***n***, where ***n*** = the index number of the DS1 interface.

Details

A Degraded Minute is one in which the estimated error rate exceeds 1E-6 but does not exceed 1E-3 (see *CCITT Specifications Volume III, Recommendation G.821*).

Degraded Minutes are determined by collecting all of the Available Seconds, removing any Severely Errored Seconds, grouping the result in 60-second long groups, and counting a 60-second long group as degraded if the cumulative errors during the seconds present in the group exceed 1E-6. Available Seconds are merely those seconds which are not Unavailable Seconds (**dsx1CurrentUASs**).

Configuration

Not applicable.

dsx1CurrentEntry

Starts a group of objects that make up a table for the DS1 interface identified by **dsx1CurrentIndex**.

Syntax

Object

Access

Not accessible.

OID

p.7.1

Details

There is one entry object for each DS1 interface.

Configuration

Not applicable.

dsx1CurrentESs

indicates the mumber of Errored Seconds.

Syntax

Gauge

Access

Read-only

OID

p.7.1.2.***n***, where ***n*** = the index number of the DS1 interface.

Details

For ESF and E1-CRC links, an Errored Second is a second with one or more Path Code Violations, OR one or more Out of Frame defects, OR one or more Controlled Slip events, OR a detected AIS defect.

For D4 and E1-noCRC links, the presence of Bipolar Violations also triggers an Errored Second.

This value is not incremented during an Unavailable Second.

Configuration

Not applicable.

dsx1CurrentIndex

Number of the DS1 interface to which block of variables apply.

Syntax

Integer

Access

Read-only

OID

p.7.1.1.***n***, where ***n*** = the index number of the DS1 interface.

Details

The following block of variables apply for Hot Swap:

If a board is...	Then the...
Extracted	Index will be removed.
Inserted	Next available index number will be used.
Replaced	Next available index number will be used.

This is the same as **dsx1LineIndex** in the Configuration table.

Configuration

Not applicable.

dsx1CurrentLCVs

Indicates the number of Line Code Violations (LCVs).

Syntax

Gauge

Access

Read-only

OID

p.7.1.11.***n***, where ***n*** = the index number of the DS1 interface.

Details

A Line Coding Violation (LCV) is the occurrence of either a Bipolar Violation (BPV) or Excessive Zeroes (EXZ) Error Event. Also known as CV-L. See T1.231 Section 6.5.1.1.

An Excessive Zeroes error event for an AMI-coded signal is the occurrence of more than fifteen contiguous zeroes. See ANSI T1.231 Section 6.1.1.1.2. For a B8ZS coded signal, the defect occurs when more than seven contiguous zeroes are detected.

Configuration

Not applicable.

dsx1CurrentLESs

Syntax

Gauge

Access

Read-only

OID

p.7.1.8.***n***, where ***n*** = the index number of the DS1 interface.

Details

Number of Line Errored Seconds.

A Line Errored Second, according to T1M1.3, is a second in which one or more Line Code Violation error events were detected.

Configuration

Not applicable.

dsx1CurrentPCVs

Indicates the number of Path Coding Violations.

Syntax

Gauge

Access

Read-only

OID

p.7.1.7.***n***, where ***n*** = the index number of the DS1 interface.

Details

A Path Coding Violation error event is a frame synchronization bit error in the D4 and E1-noCRC formats, or a CRC or frame synch.bit error in the ESF and E1-CRC formats. Also known as CV-P (see ANSI T1.231, Section 6.5.2.1).

Configuration

Not applicable.

dsx1CurrentSEFSs

Indicates the number of Severely Errored Framing Seconds.

Syntax

Gauge

Access

Read-only

OID

p.7.1.4.***n***, where ***n*** = the index number of the DS1 interface.

Details

A Severely Errored Framing Second is a second with one or more Out of Frame defects, or a detected AIS defect.

Configuration

Not applicable.

dsx1CurrentSESs

Indicates the number of Severely Errored Seconds.

Syntax

Gauge

Access

Read-only

OID

p.7.1.3.***n***, where ***n*** = the index number of the DS1 interface.

Details

This value is defined differently for different signal types:

For this signal type...	**A Severely errored second is...**
ESF signals	A second with 320 or more Path Code Violation Error Events, or one or more Out of Frame defects, or a detected AIS defect.
E1-CRC signals	A second with 832 or more Path Code Violation error events, or one or more Out of Frame defects.
E1-no CRC signals	A second with 2048 Line Code Violations or more.
D4 signals	A count of one-second intervals with Framing Error events, or an OOFdefect, OR 1544 Line Code Violations or more.

Controlled slips are not included in this parameter.

This value is not incremented during an Unavailable Second.

Configuration

Not applicable.

dsx1CurrentTable

Starts the Current table.

Syntax

Object

Access

Not accessible.

OID

p.7

Configuration

Not applicable.

dsx1CurrentUASs

Indicates the number of Unavailable Seconds.

Syntax

Gauge

Access

Read-only

OID

p.7.1.5.***n***, where ***n*** = the index number of the DS1 interface.

Details

Unavailable Seconds (UAS) are calculated by counting the number of seconds that the interface is unavailable. The DS1 interface is said to be unavailable from the onset of 10 contiguous SESs, or the onset of the condition leading to a failure. If the condition leading to the failure was immediately preceded by one or more contiguous SESs (**dsx1CurrentSESs**), then the DS1 interface unavailability starts from the onset of these SESs.

- Once unavailable, and if no failure is present, the DS1 interface becomes available at the onset of 10 contiguous seconds with no SESs.
- Once unavailable, and if a failure is present, the DS1 interface becomes available at the onset of 10 contiguous seconds with no SESs, if the failure clearing time is less than or equal to 10 seconds. If the failure clearing time is more than 10 seconds, the DS1 interface becomes available at the onset of 10 contiguous seconds with no SESs, or the onset period leading to the successful clearing condition, whichever occurs later.

All DS1 error counts are incremented while the DS1 interface is deemed available. While the interface is deemed unavailable, the only count that is incremented is UASs.

A special case exists when the 10 or more second period crosses the 900 second statistics window boundary, because the Severely Errored Second and Unavailable Second counters must be adjusted when the Unavailable Signal State is entered. Successive **get**s of the affected get occurs during the first few seconds of the window. This is an unavoidable side-effect of selecting the managed objects defined by RFC 2495.

Configuration

Not applicable.

dsx1Ds1ChannelNumber

Represents the channel number of the DS1/E1 on its parent DS2/E2 or DS3/E3.

Syntax

Integer (0...28)

Access

Read-only

OID

p.6.1.19.***n***, where ***n*** = the index number of the DS1 interface.

Details

A value of 0 indicates this DS1/E1 does not have a parent DS3/E3.

Note: The NMS SNMP agent always returns 0.

Configuration

Not applicable.

dsx1Fdl

Describes the use of the facilities data link and the sum of its capabilities.

Syntax

Integer

Access

Read-write

OID

p.6.1.13.***n***, where ***n*** = the index number of the DS1 interface.

Details

Valid entries include:

Type	Value	Description
other	1	Unknown protocol used.
dsx1Ansi-T1-403	2	FDL exchange recommended by ANSI.
dsx1Att-54016	4	ESF FDL exchanges.
dsx1Fdl-none	8	Device does not use the FDL.

Note: The NMS SNMP agent always returns dsx1Fdl - none (8). Facilities data link is not supported.

Configuration

Not applicable.

dsx1IfIndex

For NMS boards, identifies a DS1 interface managed by this agent.

Syntax

Integer (0x1..0x7fffffff)

Access

Read-only

OID

p.6.1.2.***n***, where ***n*** = the index number of the DS1 interface.

Details

This value is equal to the value of **dsx1LineIndex** for boards made by NMS Communications.

Configuration

Not applicable.

dsx1IntervalNumber

Number of this dsx1IntervalEntry in the Interval table, where each block of variables covers a fifteen minute interval.

Syntax

Integer (1..96)

Access

Read-only

OID

p.8.1.2.***n***, where ***n*** = the index number of the DS1 interface

Details

There are be 96 rows in the Interval table after the DTM agent has been active for 24 hours.

Configuration

Not applicable.

dsx1IntervalValidData

Indicates if the data for this interval is valid (not supported).

Syntax

TruthValue

Access

Read-only

OID

p.8.1.13.***n***, where ***n*** = the index number of the DS1 interface

Configuration

Not applicable.

dsx1InvalidIntervals

Indicates the number of intervals with invalid data, which will always be `0` (not supported).

Syntax

Integer

Access

Read-only

OID

p.6.1.14.***n***, where ***n*** = the index number of the DS1 interface.

Configuration

Not applicable.

dsx1LineIndex

Identifies a DS1 interface managed by this agent.

Syntax

Integer (0x1..0x7fffffff)

Access

Read-only

OID

p.6.1.1.***n***, where ***n*** = the index number of the DS1 interface.

Details

The number in the index is assigned in the sequence that the agent finds the interfaces on the boards (which does not necessarily represent the physical order of the interfaces).

For Hot Swap, valid values include:

If a board is...	Then the...
Extracted	Index will be removed.
Inserted	Next unused index number will be used.
Replaced	Next unused index number will be used.

Configuration

Not applicable.

dsx1LineCoding

Indicates the type of Zero Code Suppression used on the interface

Syntax

Integer

Access

Read-write

OID

p.6.1.6.***n***, where ***n*** = the index number of the DS1 interface

Details

Valid values include:

Type	Value	Description
dsx1JBZS	1	Jammed Bit Zero Suppression, in which the AT&T specification of at least one pulse every 8 bit periods is implemented by forcing a pulse in bit 8 of each channel. Only seven bits per channel, or 1.344 Mbps, is available for data.
dsx1B8ZS	2	Specified pattern of normal bits and bipolar violations which replace a sequence of eight zero bits.
dsx1HDB3	3	E1 links, with or without CRC, use dsx1HDB3 or dsx1AMI.
dsx1ZBTSI	4	ANSI Clear Channels may use dsx1ZBTSI, or Zero Byte Time Slot Interchange.
dsx1AMI	5	Mode where no zero code suppression is present and the interface encoding does not solve the problem directly. In this application, the higher layer must provide data, which meets or exceeds the pulse density requirements, such as inverting HDLC data.
other	6	Unlisted (default).

Configuration

This object is configured by editing the system configuration file before starting the Chassis MIB agent.

dsx1LineLength

Indicates the length of the DS1 line in meters, which will always be `0` (not supported).

Syntax

Integer

Access

Read-write

OID

p.6.1.15.***n***, where ***n*** = the index number of the DS1 interface.

Configuration

Not applicable.

dsx1LineStatus

Indicates the status of the interface.

Syntax

Integer (1..8191)

Access

Read-only

OID

p.6.1.10.***n***, where ***n*** = the index number of the DS1 interface.

Details

This value provides loopback, failure, received alarm, and transmitted alarm information. Possible status values include:

Status	Value	Description
dsx1NoAlarm	1	No Alarm Present.
dsx1RcvFarEndLOF	2	Yellow Alarm. Not supported.
dsx1XmtFarEndLOF	4	Near end sending LOF Indication. Not supported.
dsx1RcvAIS	8	Far end sending AIS (blue). Not supported.
dsx1XmtAIS	16	Near end sending AIS.
dsx1LossOfFrame	32	Near end LOF (Red Alarm).
dsx1LossOfSignal	64	Near end Loss Of Signal.
dsx1LoopbackState	128	Near end is looped.
dsx1T16AIS	256	E1 TS16 AIS.
dsx1RcvFarEndLOMF	512	Far End Sending TS16 LOMF. Not supported.
dsx1XmtFarEndLOMF	1024	Near End Sending TS16 LOMF. Not supported.
dsx1RcvTestCode	2048	Near End detects a test code.
dsx1OtherFailure	4096	Any interface status not defined.

Note: Far end is not supported.

Configuration

Not applicable.

dsx1LineStatusChange

Sent when the value of an instance **dsx1LineStatus** changes.

Syntax

TruthValue

Access

Read-only

OID

p.15.0.1

Configuration

Not applicable.

dsx1LineType

Indicates the type of DS1 interface implementing the circuit.

Syntax

Integer

Access

Read-write

OID

p.6.1.5.***n***, where ***n*** = the index number of the DS1 interface.

Details

Valid entries are:

Type	Value	Description
Other	1	Unlisted
dsx1ESF	2	Extended SuperFrame DS1
dsx1D4	3	AT&T D4 format DS1
dsx1E1	4	CCITT Recommendation G.704 (Table 4a)
dsx1E1-CRC	5	CCITT Recommendation G.704 (Table 4b)
dsx1E1-MF	6	G.704 (Table 4a) with TS16 multiframing enabled
dsx1E1-CRC-MF	7	G.704 (Table 4b) with TS16 multiframing enabled

Values 3 and 4 are the only options the agent can return.

For example, E1 interfaces return dsx1E1, and T1 interfaces return dsx1D4.

Configuration

Not applicable.

dsx1LineStatusChangeTrapEnable

Determines whether traps should be generated for this interface.

Syntax

Integer

Access

Read-write

OID

p.6.1.17.***n***, where ***n*** = the index number of the DS1 interface.

Details

Possible values are:

Value	Description
1	Enabled
2	Disabled (default)

Configuration

Not applicable.

dsx1LoopbackConfig

Represents the loopback configuration of the DS1 interface.

Syntax

Integer

Access

Read-write

OID

p.6.1.9.***n***, where ***n*** = the index number of the DS1 interface.

Details

The Trunk agent returns badValue in response to a requested loopback state that the interface does not support. Valid types for RFC 2495 are:

Type	Value	Description
dsx1NoLoop	1	Not in the loopback state. A device that is not capable of performing a loopback on the interface will always return this value.
dsx1PayloadLoop	2	The received signal at this interface is looped through the device. Typically, the received signal is looped back for re-transmission after it has passed through the device's framing function.
dsx1LineLoop	3	The received signal at this interface does not go through the device.
dsx1OtherLoop	4	Loopbacks that are not defined.

Note: The agent will only return dsx1NoLoop (1). Loopback is not supported.

Configuration

Not applicable.

dsx1LoopbackStatus

Represents the current state of the loopback on the DS1 interface.

Syntax

Integer (1...127)

Access

Read-only

OID

p.6.1.18.***n***, where ***n*** = the index number of the DS1 interface.

Details

This value contains information about loopbacks established by a manager and remotely from the far end.

dsx1LoopbackStatus is a bit map represented as a sum; therefore, it can represent multiple loopbacks simultaneously.

The bit positions are:

Bit	Value
1	dsx1NoLoopback
2	dsx1NearEndPayloadLoopback
4	dsx1NearEndLineLoopback
8	dsx1NearEndOtherLoopback
16	dsx1NearEndInwardLoopback
32	dsx1FarEndPayloadLoopback
64	dsx1FarEndLineLoopback

Note: The NMS SNMP agent always returns dsx1NoLoopback because loopback is not supported.

Configuration

Not applicable.

dsx1SendCode

Indicates the type of code being sent across the DS1 interface by the device

Syntax

Integer

Access

Read-write

OID

p.6.1.7.***n***, where ***n*** = the index number of the DS1 interface.

Details

Valid values include:

Type	Value	Description
dsx1SendNoCode	1	Sending looped or normal data.
dsx1SendLineCode	2	Sending a request for a line loopback.
dsx1SendPayloadCode	3	Sending a request for a payload loopback.
dsx1SendResetCode	4	Sending a loopback termination request.
dsx1SendQRS	5	Sending a Quasi-Random Signal(QRS) test pattern.
dsx1Send511Pattern	6	Sending a 511 bit fixed test pattern.
dsx1Send3in24Pattern	7	Sending a fixed test pattern of 3 bits set in pattern of 24.
dsx1SendOtherTestPattern	8	Sending a test pattern other than those described by this object.

Note: The SNMP agent returns dsx1SendNoCode (normal data). Loopback is not supported.

Configuration

Not applicable.

dsx1SignalMode

Indicates the signal mode of the circuit.

Syntax

Integer

Access

Read-write

OID

p.6.1.11.***n***, where ***n*** = the index number of the DS1 interface.

Details

Valid entries include:

Type	Value	Description
none	1	No bits are reserved for signaling on this channel.
robbedBit	2	T1 Robbed Bit Signaling is in use.
bitOriented	3	E1 Channel Associated Signaling is in use.
messageOriented	4	Common Channel Signaling is in use either on channel 16 of an E1 link or on channel 24 of a T1 link.

Configuration

Not applicable.

dsx1StatusLastChange

Indicates the time when the status of the interface last changed.

Syntax

TimeStamp

Access

Read-only

OID

p.6.1.16.***n***, where ***n*** = the index number of the DS1 interface.

Configuration

Not applicable.

dsx1TimeElapsed

Number of seconds that have elapsed since the beginning of the current error measurement period.

Syntax

Integer (`0..899`)

Access

Read-only

OID

p.6.1.3.***n***, where ***n*** = the index number of the DS1 interface.

Configuration

Not applicable.

dsx1TransmitClockSource

Indicates the source of the transmit clock that the board uses for synchronization

Syntax

Integer

Access

Read-write

OID

p.6.1.12.***n***, where ***n*** = the index number of the DS1 interface.

Details

Valid values include:

Type	Value	Description
loopTiming	1	The recovered receive clock of this interface is used as the transmit clock. Also know as slave.
localTiming	2	The recovered receive clock from another interface is used as the transmit clock. Also known as master.
throughTiming	3	A local clock source is used.

Configuration

This object is configured by editing the system configuration before starting the Chassis MIB agent.

dsx1ValidIntervals

Indicates the number of 15 minute intervals for which valid data was collected.

Syntax

Integer (0..96)

Access

Read-only

OID

p.6.1.4.***n***, where ***n*** = the index number of the DS1 interface.

Details

The value is always 96, unless the agent has been running for less than 24 hours, in which case it indicates the number of 15 minute intervals that the agent has been running minus 1 (since the time periods start with 1).

Configuration

Not applicable.

Software revision MIB

Software Revision MIB representation

The Software Revision MIB represents all NMS software packages installed in a system. Each file in each installed software revision is tracked in the MIB. The Software Revision agent keeps the MIB up-to-date as packages are installed or removed. However, **t**he agent cannot track revisions of NMS files manually copied to or deleted from a system (that is, without use of NMS installation software).

To keep the MIB up to date, the Software Revision agent relies on information from the module identification signature files (*.sgn* files) installed with each Natural Access product. These files are stored in the *\nms\bin* directory (*/opt/nms/bin* under UNIX). When the Natural Access Server (*ctdaemon*) is restarted, the Software Revision agent modifies the MIB to match the current set of signature files.

Note: Certain NMS patches do not install their *.sgn* files in the *\nms\bin* or */opt/nms/bin* directory. If the *.sgn* file is not installed in one of these directories, locate the file and manually copy it to the correct directory. The MIB cannot track a patch unless its *.sgn* file is in the correct directory.

Once you have installed NMS packages, service packs or patches, the values in the Software Revision MIB are updated automatically when you restart the Natural Access Server (*ctdaemon*).

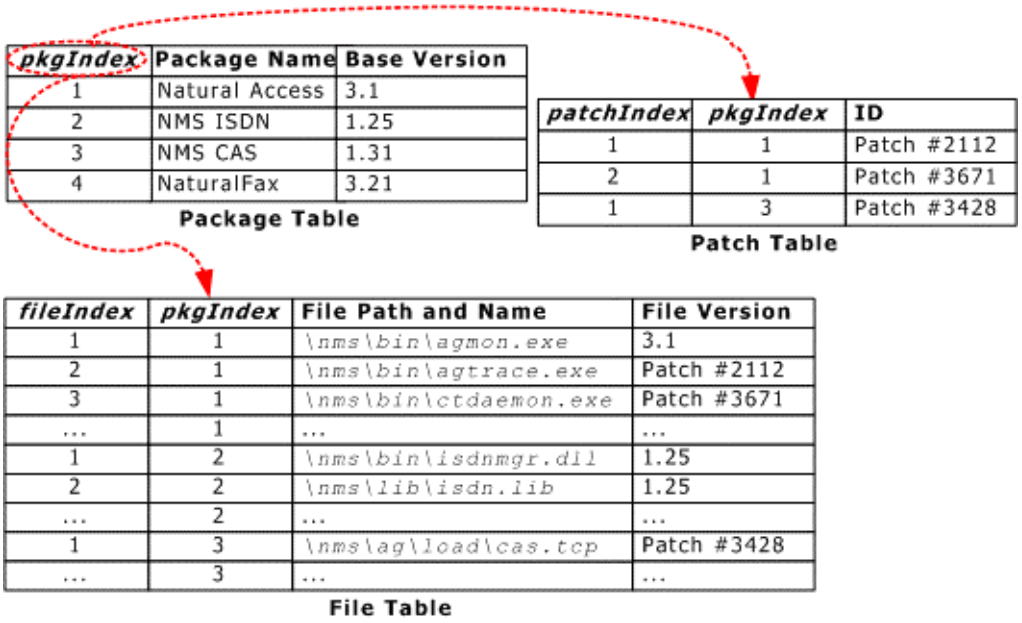
Note: Information in this MIB is not updated if files are added or removed manually (that is, without using NMS installation software).

Software revision MIB structure

The Software Revision MIB represents a system as a single managed node that contains all packages installed within it. There are three major tables within the Software Revision MIB:

Table	Description
Package	Lists each package name and base version.
File	Lists each file in a package, and the file version.
Patch	Lists patches or service packs applied to each package.

The following illustration shows how the tables in the Software Revision MIB are related to one another:



pkgIndex	Package Name	Base Version
1	Natural Access	3.1
2	NMS ISDN	1.25
3	NMS CAS	1.31
4	NaturalFax	3.21

Package Table

patchIndex	*pkgIndex*	ID
1	1	Patch #2112
2	1	Patch #3671
1	3	Patch #3428

Patch Table

fileIndex	*pkgIndex*	File Path and Name	File Version
1	1	\nms\bin\agmon.exe	3.1
2	1	\nms\bin\agtrace.exe	Patch #2112
3	1	\nms\bin\ctdaemon.exe	Patch #3671
...	1	...	...
1	2	\nms\bin\isdnmgr.dll	1.25
2	2	\nms\lib\isdn.lib	1.25
...	2	...	...
1	3	\nms\ag\load\cas.tcp	Patch #3428
...	3	...	...

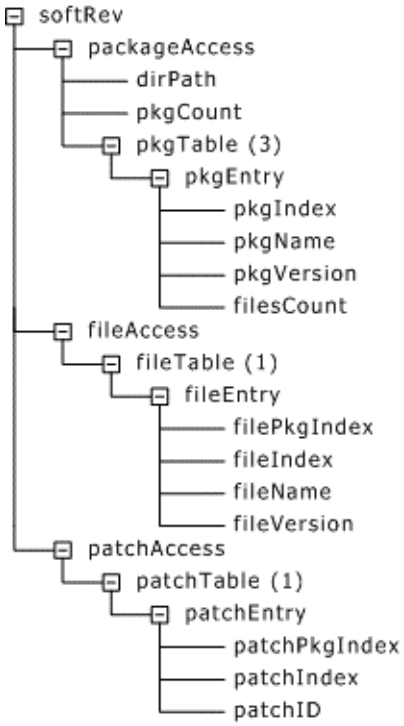
File Table

Software revision MIB table relationships

As shown in the previous illustration, each package is identified by a unique package index ***pkgIndex***, which is assigned to it in the Package Table. In the File Table, files are listed by package index, and each file is assigned a unique file index ***fileIndex***. The file version of each file is also given here. In the following illustration, *agmon.exe*, *agtrace.exe*, and *ctdaemon.exe* are part of the Natural Access 3.1 package (***pkgIndex*** #1). Since the package was installed, *agtrace.exe* has been modified by Patch #2112, and *ctdaemon.exe* has been modified by Patch #3671.

In the Patch Table, each installed service pack and patch is listed by the ***pkgIndex*** of the package it modified. Each patch is assigned a unique patch index ***patchIndex***. In the following illustration, the Natural Access 3.1 package (***pkgIndex*** #1) has been modified twice, by Patch #2112 and Patch #3671.

The sequence of objects in the Software Revision MIB (with relative OIDs for table objects) is shown in the following illustration:



Software revision MIB objects

Software Revision MIB Package table

The Package table contains the following information:

- The name of the directory where NMS packages are installed
- The total number of installed packages
- A Package Entry table containing information about each installed package, including the name of the package, the base version of the package, and the number of files in the package.

The Package table is represented in the MIB by the object **packageAccess**. The objects in the Package table are:

Object	Description
dirPath	Path where the *.sgn* files can be found.
pkgCount	Total number of installed packages.
pkgTable	Package Entry table.

The objects in the Package Entry table are:

Object	Description
pkgEntry	Top of the table.
pkgIndex	Unique identifier for an installed package.
pkgName	Name of the package.
pkgVersion	Base version of the package.
filesCount	Total number of files in the package.

As shown in Software Revision MIB representation, the **pkgIndex** object provides an index into the File and Patch tables described in this topic.

File table

The File table contains a File Entry table. This table contains a list of all files in each package. For each file, the table contains:

- The index of the package to which the file belongs
- The name of the file
- The base version of the file

The File table is represented in the MIB by the object **fileAccess**. The objects in the File table are:

Object	Description
fileTable	File Entry table.

The objects in the File Entry table are:

Object	Description
fileEntry	Top of the table.
filePkgIndex	Index of the package to which the file belongs (matches the **pkgIndex** value for the package in the Package Entry table).
fileIndex	Unique identifier for the file.
fileName	Path and filename of the file.
fileVersion	Base version of the file.

Patch table

The Patch table contains a Patch Entry table. This table contains a list of all service packs or patches applied to each package. For each patch or service pack, the table contains the:

- Index of the package to which the the service pack or patch was applied
- ID of the service pack or patch

The File table is represented in the MIB by the object **patchAccess**. The objects in the Patch table are:

Object	Description
patchTable	Patch Entry table.

The objects in the Patch Entry table are:

Object	Description
patchEntry	Top of the table.
patchPkgIndex	Index of the package to which the patch was applied (matches the **pkgIndex** value for the package in the Package Entry table).
patchIndex	Unique identifier for the patch.
patchID	ID of the patch.

Software Revision MIB object reference

Using the Software revision MIB object reference

The following sections describe the objects in this MIB. A typical object description includes:

Syntax	The datatype of the object is shown. SNMP data types include:	
	Integer	16-bit signed.
	DisplayString	ASCII text.
	Gauge	Positive integer from 0 to 4294967295 (2^{32} - 1).
	Object	Another object type from this MIB.
	TimeStamp	Positive integer from 0 to 4294967295 (2^{32} - 1).
	TruthValue	Integer value where 1 is True and 2 is False.
Access	The type of access allowed for this object. Options are:	
	Read-only	This object can not be modified by SNMP.
	Read-write	SNMP can configure this object.
OID	The OID defines the path from the root to this object. All OIDs start with ***p***, where ***p*** is is 1.3.6.1.4.1.2628.2.2 (the OID for the Chassis MIB).	
Details	Describes the object.	
Configuration	Describes how to configure the object.	
Example	Shows an example of the object.	

The source from which the NMS MIBs was compiled is supplied with the software, in ASN1 format text files. These files can be found in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX). NMS SNMP MIBs were compiled using the following files:

- Chassis MIB: *chassis-mib.txt*
- Trunk MIB: *chassis-mib.txt*
- Software Revision MIB: *softrev-mib.txt*
- OAM Database MIB: *oamdatabase-mib.txt*

Read the appropriate file using the Windows Console Management function to display the SNMP information for the proprietary agent.

Documentation comments?

dirPath

Indicates the name of the directory where the NMS files are installed.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.1

Configuration

This value is set when the first NMS package is installed.

fileAccess

Starts a group containing the File Entry table (**fileEntry**).

Syntax

Object

Access

Not accessible.

OID

p.2

Configuration

Not applicable.

filesCount

Indicates the number of files included in the package.

Syntax

Integer

Access

Read-only

OID

p.1.3.1.4.***n***

Configuration

This value is updated when the Natural Access Server (*ctdaemon*) is restarted.

fileEntry

Starts a row in the File Entry table (**fileTable**).

Syntax

Object

Access

Not accessible.

OID

p.2.1.1

Details

The number of **fileEntry** objects in the table is exactly equal to **filesCount**.

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, rows are added or removed as necessary to reflect the current sets of installed files.

fileIndex

Indicates the index of a file in the package.

Syntax

Integer

Access

Read-only

OID

p.2.1.1.2.***n***

Details

This value is a number between 1 and **filesCount**.

Configuration

This identifier is internally assigned by the agent.

fileTable

Starts a sequence of **fileEntry** objects, each of which composes a row in the File Entry table.

Syntax

Object

Access

Not accessible.

OID

p.2.1

Details

The number of **fileEntry** objects in the table is exactly equal to **filesCount**.

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, rows are added or removed as necessary to reflect the current sets of installed files.

Note: This table is not updated if files are added or removed manually (that is, without using NMS installation programs).

filePkgIndex

Identifies the package to which the file belongs.

Syntax

Integer

Access

Read-only

OID

p.2.1.1.1.***n***

Details

This object matches the **pkgIndex** identifier of an installed package in the Package Entry (**pkgTable**) table.

Configuration

This identifier is internally assigned by the agent.

fileName

Indicates the name of the package.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.1.1.3.***n***

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, all values (including this one) are imported from the *.sgn* files.

fileVersion

Indicates the base version of the package.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.1.1.4.***n***

Details

This value contains a checksum error if the file has been manually modified or corrupted since it was installed by NMS software.

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, all values (including this one) are imported from the *.sgn* files.

patchAccess

Starts a group containing the Patch Entry table (**patchEntry**).

Syntax

Object

Access

Not accessible.

OID

p.3

Configuration

Not applicable.

patchEntry

Starts a row in the Patch Entry table (**patchEntry**).

Syntax

Object

Access

Not accessible.

OID

p.3.1.1

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, rows are added or removed as necessary to reflect the current sets of installed patches and service packs.

patchIndex

Indicates the index of a service pack or patch in the table.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.2.***n***

Configuration

This identifier is internally assigned by the agent.

patchPkgIndex

Identifies the package to which the service pack or patch was applied.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.1.***n***

Details

This object matches the **pkgIndex** identifier of an installed package in the Package Entry (**pkgTable**) table.

Configuration

This identifier is internally assigned by the agent.

patchTable

Starts a sequence of **patchEntry** objects, each of which composes a row in the Patch Entry table.

Syntax

Object

Access

Not accessible.

OID

p.3.1

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, rows are added or removed as necessary to reflect the current sets of installed patches and service packs.

patchID

Indicates the ID or number of the patch.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.3.1.1.3.***n***

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, all values (including this one) are imported from the *.sgn* files.

packageAccess

Starts a group containing the **dirPath**, **pkgCount**, and **pkgTable** objects.

Syntax

Object

Access

Not accessible.

OID

p.1

Details

The group contains the following:

Object	Description
dirPath	Top of the table.
pkgCount	Total number of installed packages.
pkgTable	Package Entry table.

Configuration

Not applicable.

pkgCount

Indicates the total number of installed packages.

Syntax

Integer

Access

Read-only

OID

p.1.2

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, this value is updated to reflect the current number of installed packages.

pkgEntry

Starts a row in the Package Entry table (**pkgTable**).

Syntax

Object

Access

Not accessible.

OID

p.1.3.1

Details

The number of **pkgEntry** objects in the table is exactly equal to **pkgCount**.

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, rows are added or removed as necessary to reflect the current set of installed packages.

pkgIndex

Identifies an installed package in the Package Entry (**pkgTable**) table.

Syntax

Integer

Access

Read-only

OID

p.1.3.1.1.***n***

Details

Each package is assigned a unique **pkgIndex** number in this table, sequentially between 1 and **pkgCount**. **pkgIndex** provides an index into the File Entry (**fileTable**) and Patch Entry (**patchTable**) tables.

Configuration

This identifier is internally assigned by the agent.

pkgName

Indicates the name of the package.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.3.1.2.***n***

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, all values (including this one) are imported from the *.sgn* files.

pkgTable

Starts a sequence of **pkgEntry** objects, each of which composes a row in the Package Entry table.

Syntax

Object

Access

Not accessible.

OID

p.1.3

Details

The number of **pkgEntry** objects in the table is exactly equal to **pkgCount**.

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, rows are added or removed as necessary to reflect the current set of installed packages.

pkgVersion

Indicates the base version of the package.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.3.1.3.***n***

Configuration

When the Natural Access Server (*ctdaemon*) is restarted, all values (including this one) are imported from the *.sgn* files.

OAM Database MIB

NMS OAM database representation

The OAM Database MIB presents an SNMP front end to the contents of the NMS OAM database on a system. Within this database, NMS OAM software maintains tables of configuration data for hardware and software components in the system. Each table of configuration data constitutes a *managed object*: the logical representation of the component to the system. Using the OAM Database MIB, you can query, add, modify, or delete information for managed objects in much the same way as NMS OAM does.

For detailed information on NMS OAM, refer to the *NMS OAM System User's Manual* and to the *NMS OAM Service Developer's Reference Manual*.

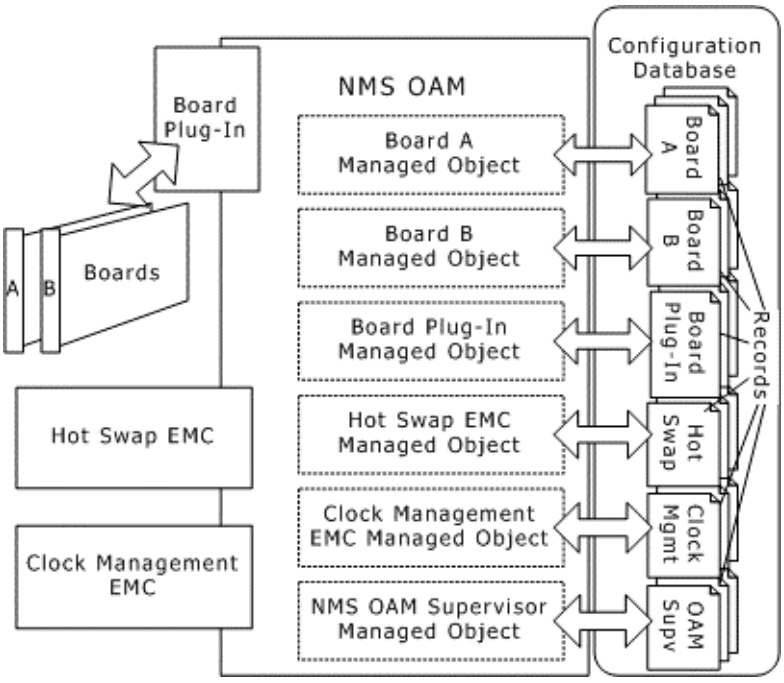
Managed components

NMS OAM manages the following components (see the following illustration):

- **Boards**
  A separate set of configuration information is kept for each AG, CG, CX, and QX board in the system.
- **NMS OAM Supervisor**
  NMS OAM keeps configuration information for its Supervisor process, which oversees all other NMS OAM components.
- **Board plug-ins**
  NMS OAM communicates with boards using software extensions called board plug-ins. There is one plug-in per board family. NMS OAM maintains a separate set of configuration information for each plug-in.
- **Extended management components** (EMCs)
  Extended management components (EMCs) are software modules which add functionality to NMS OAM. A separate set of configuration data is kept for each EMC. Currently, two EMCs are supplied with NMS OAM:
- Hot Swap EMC
- H.100 and H.110 Clock Management EMC

For more information about the Hot Swap EMC and Clock Management EMC, refer to the *NMS OAM System User's Manual*.

The following illustration illustrates the relationship between the components in a system, their representation as managed objects within NMS OAM, and the relationship of managed objects to data within the NMS OAM database:



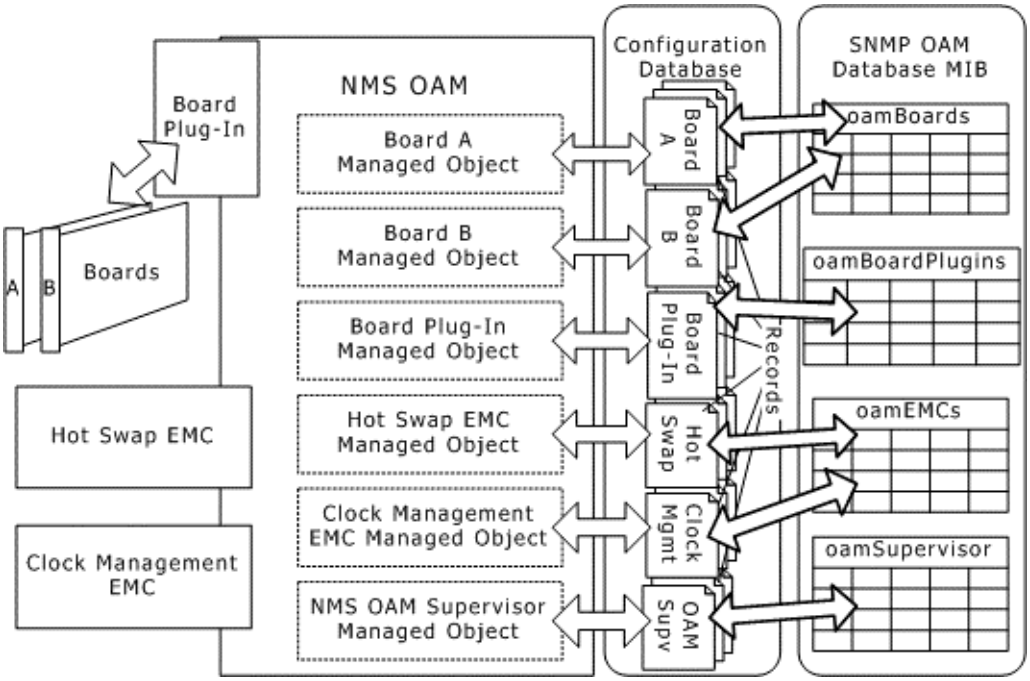
NMS OAM components and managed objects

OAM Database MIB tables and keywords

Within the OAM Database MIB, the data in the NMS OAM database is represented in the following tables:

Table	Contains
oamSupervisor	A table allowing access to the configuration data for the Supervisor managed object. Additional values in this table allow an application to: • Start and stop the Supervisor process • Set up event masking • Configure alert registration • Create new board entries
oamBoardPlugins	A table allowing access to the configuration data for each installed board plug-in.
oamEMCs	A table allowing access to the configuration data for each installed EMC.
oamBoards	Tables allowing access to the configuration data for each board in the system. These tables allow an application to: • Query and change keywords for managed objects • Query and change board names and numbers • Start and stop a board • Test a board • Delete a board configuration from the database
oamOtherObjects	Contains a table allowing access to the configuration data for other managed objects (if any).
oamEventsTraps	Allows you to examine incoming NMS OAM events.

The following illustration illustrates the relationship between the NMS OAM database and SNMP MIBs:



NMS OAM database and SNMP MIB tables

Keywords in the OAM database MIB

Configuration data in both the NMS OAM database and the OAM database MIB is expressed as keyword name/value pairs (for example, AutoStart = YES). Keywords and values can be queried, added, modified, or deleted. Modifying a keyword in the MIB modifies the keyword in the NMS OAM database, and vice versa.

Each keyword has several attributes, called *qualifiers*. For example, the qualifier Type indicates the type of value it accepts (Integer, String, etc.). The qualifier ReadOnly indicates if a keyword is read-only. Within the OAM database MIB, qualifier information for each keyword is stored with the keyword.

The following table lists the information stored in a MIB for each keyword:

Datum	Description	Valid values
Managed Object Index	The index of the managed object to which the keyword belongs.	Any integer from 1 upward.
Index	A unique (within the table) index for the keyword.	Any integer from 1 upward.
Keyword Name	The name of the keyword.	The keyword name, preceded by one or more group keyword names separated by periods (see below).
Keyword Value	The value of the keyword.	Any value permitted by the keyword's type and possible value parameters.
Type	Type of keyword value. Equivalent to the value of the keyword's Type qualifier.	Integer, String, or Object. Keywords of type Object appear only in the Supervisor Keyword table (**supervisorTable**).
Mode	Indicates if keyword value is read-only or not. Reflects the value of the keyword's Readonly qualifier.	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.

Possible Values	Indicates the range of possible values for the keyword. Combines information from the keyword's Base, Min, Max, and Choices qualifiers.	If the keyword type is Integer, and is a yes/no choice, this field contains a string of this format:
		`Nb values=2: Yes,No`
		If the keyword type is Integer, and can take a range of values, this field contains a string of this format:
		BASE ***base***:***min_value*** <> ***max_value***
		...where:
		• ***base*** is a mathematical base of the integer (for example, 16 for a hexadecimal number).
		• ***min_value*** is the minimum allowed value.
		• ***max_value*** is the maximum allowed value.
		For example:
		`BASE 10: 0 <> 65535`
		If the keyword type is String, this field contains all the allowed strings for this keyword, separated by commas (,). For example: YES,NO. If any string is acceptable, this field contains <no range>.
		If the keyword type is Object, no possible values are given.

Description	Text describing the keyword. Equivalent to the value of the keyword's Description qualifier.	A string of text. If no description is given, this keyword contains <none>.

Within NMS OAM, keywords are grouped into a variety of formats which allow an application to enumerate keyword sets to determine their values. These formats include arrays, structs, structs containing arrays, arrays containing structs and so forth. Each group of keywords is represented by a keyword that does not actually contain configuration data, but instead merely represents the group.

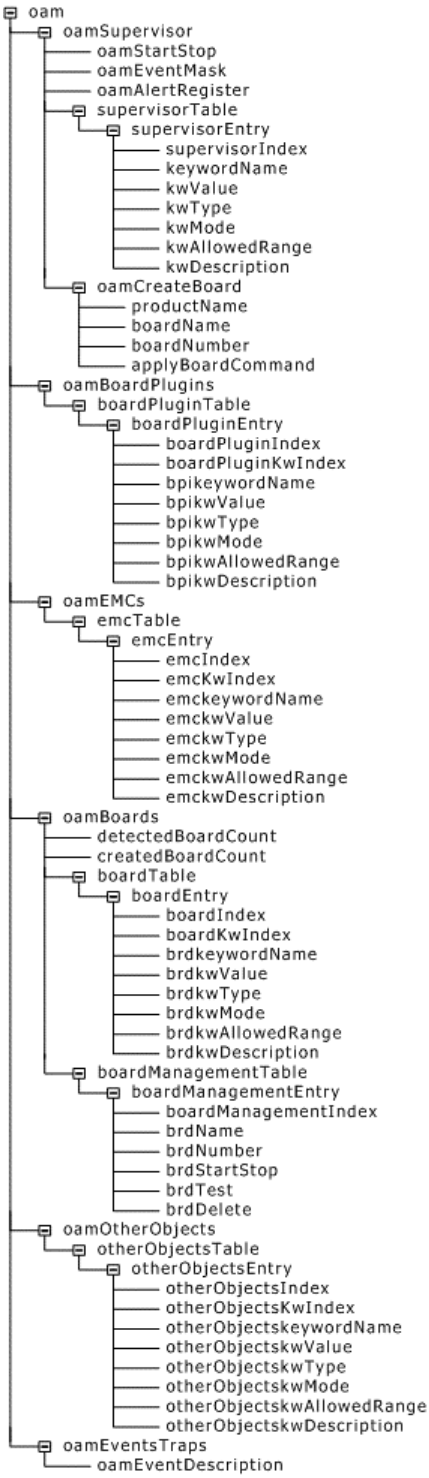
Within the OAM database MIB, keyword enumeration takes place transparently. Thus there is no need to include group name keywords as separate entries in the MIB. Instead, only keywords that actually contain values (that is, keywords of type Integer or String) are given separate entries in the tables. Where a keyword belongs to one or more groups, the group names are appended to the keyword name in the table, separated by periods (.). For example, the keyword FallBackClockSource in the struct HBus which is within Clocking is expressed as Clocking.HBus.FallBackClockSource.

Populating OAM MIB tables

When the OAM Database SNMP agent is launched, it opens the NMS OAM Supervisor managed object. It populates the OAM Database MIB tables based on information it finds in this managed object, and in objects referenced in this object.

For each Integer or String keyword in the NMS OAM Supervisor managed object, the agent creates a row in the Supervisor Keyword table (**supervisorTable**), and stores the keyword and qualifier information as described in OAM database MIB keywords and tables. It also uses some of the Supervisor keywords to access the board plug-ins, EMCs, and board managed objects, so it can populate the other tables in the MIB. This operation is described in greater detail in the illustration that follows.

The sequence of objects in the OAM Database MIB (with relative OIDs for table objects) is shown in the following illustration:



OAM database MIB objects

OAM MIB Supervisor tables

The OAM Supervisor table contains

- A table of Supervisor keywords, values, and qualifiers
- Values that allow you to start or stop the Supervisor process, set up event masks, register for NMS OAM alert events, and create board instances in the database

The objects in the OAM Supervisor table (**oamSupervisor**) are:

Object	Description
oamStartStop	Starts or stops the NMS OAM Supervisor process, or indicates its status.
oamEventMask	Sets the NMS OAM event mask, or indicates its status.
oamAlertRegister	Registers for NMS OAM alert notification, or indicates the status of the registration.
supervisorTable	Supervisor Keyword table, containing NMS OAM Supervisor keywords, values, and qualifiers.
oamCreateBoard	Create Board table, containing values that allow you to create board instances in the database.

The objects in the Supervisor Keyword table (**supervisorTable**) are:

Object	Description
supervisorTable	Top of the table.
supervisorEntry	Starts a row of the Supervisor Keyword table.
supervisorIndex	Unique index (within this table) identifying the keyword.
keywordName	The keyword name, formatted as described in OAM database MIB keywords and tables.
kwValue	The value of the keyword.
kwType	The type of the keyword: `Integer`, `String`, or `Object`
kwMode	`1` indicates keyword value is read-only. `2` indicates keyword value is read/write.
kwAllowedRange	The range of allowable values for the keyword, formatted as described in OAM database MIB keywords and tables.
kwDescription	A short description of the keyword.

The following illustration shows sample HPOpenView output displaying the contents of the Supervisor Keyword table:

	supervisorIndex	keywordName	kwValue	kwType	kwMode	kwAllowedRang	kwDescription
1	1	ExtendedManagementComponents[0]	clkmgr.emc	Object	readOnly	<no range>	<none>
2	2	ExtendedManagementComponents[1]	HotSwap.emc	Object	readOnly	<no range>	<none>
3	3	Products[0]	AG_2000	String	readOnly	<no range>	<none>
4	4	Products[1]	AG_4000_1E1	String	readOnly	<no range>	<none>
5	5	Products[2]	AG_4000_1T1	String	readOnly	<no range>	<none>
6	6	Products[3]	AG_4000_2E1	String	readOnly	<no range>	<none>
7	7	Products[4]	AG_4000_2T1	String	readOnly	<no range>	<none>
8	8	Products[5]	AG_4000_4E1	String	readOnly	<no range>	<none>
9	9	Products[6]	AG_4000_4T1	String	readOnly	<no range>	<none>
10	10	Products[7]	AG_4000C_2E1	String	readOnly	<no range>	<none>
11	11	Products[8]	AG_4000C_2T1	String	readOnly	<no range>	<none>
12	12	Products[9]	AG_4000C_4E1	String	readOnly	<no range>	<none>
13	13	Products[10]	AG_4000C_4T1	String	readOnly	<no range>	<none>
14	14	Products[11]	AG_CPCI_Quad_E1	String	readOnly	<no range>	<none>
15	15	Products[12]	AG_CPCI_Quad_T1	String	readOnly	<no range>	<none>
16	16	Products[13]	AG_Dual_E1	String	readOnly	<no range>	<none>
17	17	Products[14]	AG_Dual_T1	String	readOnly	<no range>	<none>
18	18	Products[15]	AG_Quad_Connect_E1	String	readOnly	<no range>	<none>
19	19	Products[16]	AG_Quad_Connect_T1	String	readOnly	<no range>	<none>
20	20	Products[17]	AG_Quad_E1	String	readOnly	<no range>	<none>
21	21	Products[18]	AG_Quad_T1	String	readOnly	<no range>	<none>
22	22	Products[19]	CG_6000C_Quad	String	readOnly	<no range>	<none>
23	23	Products[20]	CX 2000-16	String	readOnly	<no range>	<none>
24	24	Products[21]	CX 2000-32	String	readOnly	<no range>	<none>
25	25	Products[22]	CX 2000C-16	String	readOnly	<no range>	<none>
26	26	Products[23]	CX 2000C-32	String	readOnly	<no range>	<none>
27	27	Products[24]	CX 2000C-48	String	readOnly	<no range>	<none>
28	28	Products[25]	QX 2000/100-4L	String	readOnly	<no range>	<none>
29	29	Products[26]	QX 2000/200-4L	String	readOnly	<no range>	<none>
30	30	Products[27]	QX 2000/80-1L	String	readOnly	<no range>	<none>
31	31	Products[28]	QX 2000/80-4L	String	readOnly	<no range>	<none>
32	32	BoardPlugins[0]	agplugin.bpi	Object	readOnly	<no range>	<none>
33	33	BoardPlugins[1]	cg6kpi.bpi	Object	readOnly	<no range>	<none>
34	34	BoardPlugins[2]	cx.bpi	Object	readOnly	<no range>	<none>
35	35	BoardPlugins[3]	qx2kpi.bpi	Object	readOnly	<no range>	<none>
36	36	Boards[0]	Name0	Object	readOnly	<no range>	<none>
37	37	Boards[1]	Name1	Object	readOnly	<no range>	<none>
38	38	Name	Supervisor	Object	readOnly	<no range>	<none>

HPOpenView output

The objects in the Create Board table (**oamCreateBoard**) are:

Object	Description
productName	Product type of the board to create. Note: All product names supported by NMS OAM can be found in the Supervisor keyword Products[x]. To learn how to access this keyword in the MIB, see Accessing Keywords for Boards, Plug-ins, or EMCs.
boardName	Name to give the created board.
boardNumber	Board number to give the created board.
applyBoardCommand	Set this to 1 to create the board based upon the **productName**, **boardName**, and/or **boardNumber** values you specified.

OAM MIB Board Plug-in table

The OAM Board Plug-in table contains, for each board plug-in, a table of the plug-in's keywords, values, and qualifiers.

The objects in the OAM Board Plug-in table (**oamBoardPlugins**) are:

Object	Description
boardPluginTable	Top of the table.
boardPluginEntry	Starts a row of the OAM Board Plug-in table.
boardPluginIndex	Plug-in index. This is equivalent to the index number of the BoardPlugins[***x***] keyword listing the board plug-in in the Supervisor managed object (see below).
boardPluginKwIndex	Unique index (within this table) identifying the keyword.
bpikeywordName	The keyword name, formatted as described in OAM database MIB tables and keywords.
bpikwValue	The value of the keyword.
bpikwType	The type of the keyword: Integer or String.
bpikwMode	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.
bpikwAllowedRange	The range of allowable values for the keyword, formatted as described in OAM database MIB tables and keywords.
bpikwDescription	A short description of the keyword.

To populate this table, the OAM Database SNMP agent opens the NMS OAM Supervisor managed object, and retrieves the values in BoardPlugins[x] keyword. This is an array listing the board plug-ins installed and running under the Supervisor. The agent opens the managed object for each listed plug-in, and creates a row in the Board Plug-in table (oamBoardPlugins) for each keyword in the managed object. Each keyword is given two indices:

- The index of the plug-in to which the keyword belongs (boardPluginIndex). This is equivalent to the index of the BoardPlugins[x] keyword listing the managed object.
- A unique numerical index (**boardPluginKwIndex**), from 1 upwards.

OAM MIB EMC table

The Extended Management Component (EMC) table contains, for each EMC, a table of the EMC's keywords, values and qualifiers.

The objects in the Extended Management Component (EMC) table (**oamEMCs**) are:

Object	Description
emcTable	Top of the table.
emcEntry	Starts a row of the Extended Management Component table.
emcIndex	EMC index.
emcKwIndex	Unique index (within this table) identifying the keyword.
emckeywordName	The keyword name, formatted as described in OAM database MIB tables and keywords.
emckwValue	The value of the keyword.
emckwType	The type of the keyword: Integer or String.
emckwMode	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.
emckwAllowedRange	The range of allowable values for the keyword, formatted as described in OAM database MIB tables and keywords.
emckwDescription	A short description of the keyword.

To populate this table, the OAM Database SNMP agent opens the NMS OAM Supervisor managed object, and retrieves the values in the ExtendedManagementComponents[x] keyword. This is an array listing the EMCs installed and running under the Supervisor. The agent opens the managed object for each listed EMC, and creates a row in the Extended Management Object table (**oamEMCs**) for each keyword in the managed object. Each keyword is given two indices:

- The index of the EMC to which the keyword belongs (**emcIndex**). This is equivalent to the index of the ExtendedManagementComponents[x] keyword listing the managed object.
- A unique numerical index (**emcKwIndex**), from 1 upwards.

OAM MIB Boards table

The OAM Boards table contains

- The number of boards automatically detected in the system
- The total number of boards registered to NMS OAM
- A table of boards, each with their keywords, values and qualifiers
- Values that allow you to start or stop a board, test a board, or delete a board instance from the database

The objects in the OAM Boards table (**oamBoards**) are:

Object	Description
detectedBoardCount	The number of boards automatically detected in the system.
createdBoardCount	The total number of boards registered to NMS OAM.
boardTable	Board Keyword table, containing a list of boards, each with their keywords, values, and qualifiers.
boardManagementTable	Board Management table, containing values that allow you to start, stop, test, or delete a board, change the board name or number, or query its status.

The objects in the Board Keyword table (**boardTable**) are:

Object	Description
boardTable	Top of the table.
boardEntry	Starts a row of the Board Keyword table.
boardIndex	Board index.
boardKwIndex	Unique index (within this table) identifying the keyword.
brdkeywordName	The keyword name, formatted as described in OAM database MIB tables and keywords.
brdkwValue	The value of the keyword.
brdkwType	The type of the keyword: `Integer` or `String`.
brdkwMode	`1` indicates keyword value is read-only. `2` indicates keyword value is read/write.
brdkwAllowedRange	The range of allowable values for the keyword, formatted as described in OAM database MIB tables and keywords.
brdkwDescription	A short description of the keyword.

To populate the Board Keyword table (**boardTable**), the OAM Database SNMP agent opens the NMS OAM Supervisor managed object, and retrieves the values in Boards[x] keyword. This is an array listing the boards managed by the board plug-ins running under the Supervisor. The agent opens the managed object for each listed board, and creates a row in the Board Keyword table for each keyword in the managed object. Each keyword is given two indices:

- The index of the board to which the keyword belongs (**boardIndex**). This is equivalent to the index of the Boards[x] keyword listing the managed object.

Note: This index does not necessarily match the board number (the value of the Number keyword for the board).

- A unique numerical index (**boardKwIndex**), from 1 upwards.

The objects in the Board Management table (**boardManagementTable**) are:

Object	Description
BoardManagementEntry	Top of the table.
BoardManagementIndex	Index of the board to manage (matches the **boardIndex** of the board in the Board Keywords table).
BrdName	Use to query or change the board name.
BrdNumber	Use to query or change the board number.
BrdStartStop	Starts or stops the board, or indicates its status.
BrdTest	Tests the board, or indicates the testing status.
BrdDelete	Delete the board instance from the NMS OAM database.

OAM MIB Other Objects table

The Other Objects table is included so that future extensions to NMS OAM do not require changes to the structure of the OAM Database MIB. The Other Objects table will contain, for each object, a table of the objects keywords, values and qualifiers.

The objects in the Other Objects table (**oamOtherObjects**) are:

Object	Description
otherObjectsTable	Top of the table.
otherObjectsEntry	Starts a row of the Other Objects table.
otherObjectsIndex	Object index.
otherObjectsKwIndex	Unique index (within this table) identifying the keyword.
otherObjectskeywordName	The keyword name, formatted as described in OAM database MIB tables and keywords.
otherObjectskwValue	The value of the keyword.
otherObjectskwType	The type of the keyword: `Integer` or `String`.
otherObjectskwMode	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.
otherObjectskwAllowedRange	The range of allowable values for the keyword, formatted as described in OAM database MIB tables and keywords.
otherObjectskwDescription	A short description of the keyword.

OAM MIB Events Traps table

The OAM Events Traps table allows an application to receive OAM events through the MIB. The objects in the OAM Events Traps table (**oamEventsTraps**) are:

Object	Description
oamEventDescription	The last event sent back by OAM.

Using the OAM database MIB object reference

The following sections describe the objects in this MIB. A typical object description includes:

Syntax	The datatype of the object is shown. SNMP data types include:	
	Integer	16-bit signed.
	DisplayString	ASCII text.
	Gauge	Positive integer from 0 to 4294967295 (2^{32} - 1).
	Object	Another object type from this MIB.
	TimeStamp	Positive integer from 0 to 4294967295 (2^{32} - 1).
	TruthValue	Integer value where 1 is True and 2 is False.
Access	The type of access allowed for this object. Options are:	
	Read-only	This object can not be modified by SNMP.
	Read-write	SNMP can configure this object.
OID	The OID defines the path from the root to this object. All OIDs start with ***p***, where ***p*** is is 1.3.6.1.4.1.2628.2.2 (the OID for the Chassis MIB).	
Details	Describes the object.	
Configuration	Describes how to configure the object.	
Example	Shows an example of the object.	

The source from which the NMS MIBs was compiled is supplied with the software, in ASN1 format text files. These files can be found in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX). NMS SNMP MIBs were compiled using the following files:

- Chassis MIB: *chassis-mib.txt*
- Trunk MIB: *chassis-mib.txt*
- Software Revision MIB: *softrev-mib.txt*
- OAM Database MIB: *oamdatabase-mib.txt*

Read the appropriate file using the Windows Console Management function to display the SNMP information for the proprietary agent.

applyBoardCommand

Set this value to 1 to create a new board managed object in the NMS OAM database based on the **productName**, **boardName**, and **boardNumber** values you specified.

Syntax

Integer { create(1), donothing(2) }

Access

Read-write

OID

p.1.5.4

Details

Reading this value always returns 2. For more information, see Creating and deleting board managed objects.

Configuration

Configured by the user as necessary.

boardEntry

Starts a row in the OAM Boards table (boardTable).

Syntax

Object

Access

Not accessible.

OID

p.4.3.1

Details

Each row in the OAM Boards table contains information about a board keyword.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, refer to the OAM Boards table.

boardIndex

Indicates the board managed object to which the keyword belongs.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.1.***n***

Details

This keyword's maps to the index value of the Supervisor keyword Boards[x] listing the board in the NMS OAM database. For example, if Boards[1]=MyBoard, all keywords for this board in the OAM Database MIB will have boardIndex equal to 1.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, refer to the OAM Boards table.

boardKwIndex

Indicates the keyword's index.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.2.***n***

Details

Keywords are numbered sequentially from 1 upward.

Configuration

Determined when the OAM Database SNMP agent populates the OAM Boards table as described in OAM Boards table.

boardManagementEntry

Starts a row in the Board Management table (**boardManagementTable**).

Syntax

Object

Access

Not accessible.

OID

p.4.4.1

Details

Each row contains entries which allow you to start, stop, test, or delete a board, or query its status.

Configuration

Not applicable.

boardManagementIndex

Provides the index of the board to manage.

Syntax

Integer

Access

Read-only

OID

p.4.4.1.1.***n***

Details

The index matches the **boardIndex** of the board in the Board Keywords table.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database.

boardManagementTable

Provides a table that allows management of boards through the OAM Database MIB.

Syntax

Object

Access

Not accessible.

OID

p.4.4

Details

The objects in the Board Management table are:

Object	Description
boardManagementEntry	Top of the table.
boardManagementIndex	Index of the board to manage (matches the **boardIndex** of the board in the Board Keywords table).
brdName	Queries or changes the name of the board.
brdNumber	Queries or changes the board number.
brdStartStop	Starts or stops the board or indicates its status.
brdTest	Tests the board or indicates the testing status.
brdDelete	Deletes the board instance from the NMS OAM database.

Configuration

Not applicable.

boardName

Indicates the name to give the created board.

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.1.5.2

Details

For more information, see Creating and deleting board managed objects.

Configuration

Configured by the user as necessary.

boardNumber

Specifies a number to give the created board.

Syntax

Integer

Access

Read-write

OID

p.1.5.3

Details

For more information, see Creating and deleting board managed objects.

Configuration

Configured by the user as necessary.

boardPluginEntry

Starts a row in the Board Plug-in table (**boardPluginTable**).

Syntax

Object

Access

Not accessible.

OID

p.2.1.1

Details

Each row in the Board table contains information about a board plug-in keyword.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

boardPluginIndex

Indicates the board plug-in to which the keyword belongs.

Syntax

Integer

Access

Read-only

OID

p.2.1.1.1.***n***

Details

This value maps to the index value of the Supervisor keyword BoardPlugins[x] listing the board plug-in in the NMS OAM database. For example, if BoardPlugins[1]=agplugin.bpi (the AG board plug-in), all AG board plug-in keywords in the OAM Database MIB will have **boardPluginIndex** equal to 1.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

boardPluginKwIndex

Indicates the keyword's index.

Syntax

Integer

Access

Read-only

OID

p.2.1.1.2.***n***

Details

Keywords are numbered sequentially starting at 1.

Configuration

This value is determined when the OAM Database SNMP agent populates the Board Plug-in table as described in the OAM Board Plug-in table.

boardPluginTable

Starts a sequence of **boardPluginEntry** objects, each of which composes a row in the Board Plug-in table.

Syntax

Object

Access

Not accessible.

OID

p.2.1

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

boardTable

Starts a sequence of boardEntry objects, each of which composes a row in the OAM Boards table.

Syntax

Object

Access

Not accessible.

OID

p.4.3

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, refer to the OAM Boards table.

bpikeywordName

Specifies a board plug-in keyword name.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.1.1.3.***n***

Details

Where a keyword belongs to one or more arrays or structures, the array and structure names are appended to the keyword name in the table, separated by periods (.). For example, the keyword FallBackClockSource in the struct HBus which is within Clocking is expressed as Clocking.HBus.FallBackClockSource.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

bpikwAllowedRange

Indicates the range of allowed values for the board plug-in keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.1.1.7.***n***

Details

If the keyword type is Integer, and is a yes/no choice, **bpikwAllowedRange** contains a string of this format:

```
Nb values=2: Yes,No
```

If the keyword type is Integer, and can take a range of values, **bpikwAllowedRange** contains a string of this format:

BASE ***base***:***min_value*** <> ***max_value***

where:

- ***base*** is a mathematical base of the integer (for example, 16 for a hexadecimal number).
- ***min_value*** is the minimum allowed value.
- ***max_value*** is the maximum allowed value.

For example:

```
BASE 10: 0 <> 65535
```

If the keyword type is String, **bpikwAllowedRange** contains all the allowed strings for this keyword, separated by commas (,). For example: YES,NO. If any string is acceptable, this field contains <no range>.

bpikwAllowedRange reflects the combined values of the Base, Min, Max, and Choices qualifiers for the keyword in the NMS OAM database.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

bpikwDescription

Provides a short description of the board plug-in keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.1.1.8.***n***

Details

bpikwDescription is equivalent to the value of the Description qualifier for the keyword in the NMS OAM database. If no description is given, **bpikwDescription** contains <none>.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

bpikwMode

Indicates if the Supervisor keyword is read-only or read-write.

Syntax

Integer { readOnly(1), readWrite(2) }

Access

Read-only

OID

p.2.1.1.6.***n***

Details

bpikwMode reflects the value of the keyword's ReadOnly qualifier in the NMS OAM database.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

bpikwType

Indicates the type of the board plug-in keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.1.1.5.***n***

Details

Valid types include:

Type	Description
Integer	An integer.
String	A string of 0 or more characters.

bpikwType is equivalent to the value of the Type qualifier for the keyword in the NMS OAM database.

Keywords of other types (for example: Array, Struct, StructAndArray) are not included as separate entries in MIB tables. For more information, see OAM database representation.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

bpikwValue

Indicates the board plug-in keyword value.

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.2.1.1.4.***n***

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Plug-in table is populated, refer to the OAM Board Plug-in table.

brdDelete

Deletes a board managed object.

Syntax

Integer { enable(1), disable(2) }

Access

Read-write

OID

p.4.4.1.6.***n***

Details

For more information, see Creating and deleting board managed objects.

Configuration

Not applicable.

brdkeywordName

Indicates a board keyword name.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.3.***n***

Details

Where a keyword belongs to one or more arrays or structures, the array and structure names are appended to the keyword name in the table, separated by periods (.). For example, the keyword FallBackClockSource in the struct HBus which is within Clocking is expressed as Clocking.HBus.FallBackClockSource.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, refer to OAM Boards table.

brdkwAllowedRange

Indicates a range of allowed values for the board keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.7.***n***

Details

If the keyword type is Integer, and is a yes/no choice, **brdkwAllowedRange** contains a string of this format:

```
Nb values=2: Yes,No
```

If the keyword type is Integer, and can take a range of values, **brdkwAllowedRange** contains a string of this format:

BASE ***base***:***min_value*** <> ***max_value***

...where:

- ***base*** is a mathematical base of the integer (for example, 16 for a hexadecimal number).
- ***min_value*** is the minimum allowed value.
- ***max_value*** is the maximum allowed value.

For example:

```
BASE 10: 0 <> 65535
```

If the keyword type is String, **brdkwAllowedRange** contains all the allowed strings for this keyword, separated by commas (,). For example: YES,NO. If any string is acceptable, this field contains <no range>.

brdkwAllowedRange reflects the combined values of the Base, Min, Max, and Choices qualifiers for the keyword in the NMS OAM database.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, see OAM Boards table.

brdkwDescription

Provides a short description of the board keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.8.***n***

Details

brdkwDescription is equivalent to the value of the Description qualifier for the keyword in the NMS OAM database. If no description is given, **brdkwDescription** contains <none>.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, see OAM Boards table.

brdkwMode

Indicates if the board keyword is read-only or read-write.

Syntax

Integer { readOnly(1), readWrite(2) }

Access

Read-only

OID

p.4.3.1.6.***n***

Details

brdkwMode reflects the value of the keyword's ReadOnly qualifier in the NMS OAM database.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, see OAM Boards table.

brdkwType

Indicates the type of the board keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.5.***n***

Details

Valid types include:

Type	Description
Integer	An integer.
String	A string of 0 or more characters.

brdkwType is equivalent to the value of the Type qualifier for the keyword in the NMS OAM database.

Keywords of other types (for example: Array, Struct, StructAndArray) are not included as separate entries in MIB tables. For more information, see Keywords in the OAM database MIB.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, see OAM Boards table.

brdkwValue

Indicate the board keyword value.

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.4.3.1.4.***n***

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Board Keyword table is populated, see OAM Boards table.

brdName

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.4.4.1.2.***n***

Details

Sets or determines the name of the board. For more information, refer to Querying and setting the board name and number.

Configuration

Not applicable.

brdNumber

Sets or determines the board number of the board.

Syntax

Integer

Access

Read-write

OID

p.4.4.1.3.***n***

Details

For more information, refer to Querying and setting the board name and number.

Configuration

Not applicable.

brdStartStop

Starts or stops a board, or indicates whether it is started or stopped.

Syntax

Integer { brdStart(1), brdStop(2) }

Access

Read-write

OID

p.4.4.1.4.***n***

Details

For more information, refer to Starting, stopping, and testing boards.

Configuration

Not applicable.

brdTest

Initiates board testing, or indicates if a board is currently testing or not.

Syntax

Integer

Access

Read-write

OID

p.4.4.1.5.***n***

Details

Reading this value always returns -1. For more information, refer to Starting, stopping, and testing boards.

Configuration

Not applicable.

createdBoardCount

The number of boards created within NMS OAM for this board family.

Syntax

Integer

Access

Read-only

OID

p.4.2

Configuration

This value is updated whenever board managed objects are created or deleted.

detectedBoardCount

Indicates the number of boards physically detected for this board family.

Syntax

Integer

Access

Read-only

OID

p.4.1

Configuration

This value is updated whenever the NMS OAM automatic board detection functions are activated. For more information, see the *NMS OAM Service Developer's Reference Manual*.

emcEntry

Starts a row in the Extended Management Component table (**emcTable**).

Syntax

Object

Access

Not accessible.

OID

p.3.1.1

Details

Each row in the Extended Management Component table contains information about an EMC keyword.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, refer to the OAM MIB EMC table.

emcIndex

Indicates the EMC to which the keyword belongs.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.1.***n***

Details

This value maps to the index value of the ExtendedManagementComponents[***x***] Supervisor keyword listing the EMC in the NMS OAM database. For example, if ExtendedManagementComponents[1]=hotswap.emc (the Hot Swap EMC), all Hot Swap EMC keywords in the OAM Database MIB will have **emcIndex** equal to 1.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, refer to the OAM MIB EMC table.

emckeywordName

Provides an EMC keyword name.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.3.1.1.3.***n***

Details

When a keyword belongs to one or more arrays or structures, the array and structure names are appended to the keyword name in the table, separated by periods (.). For example, the keyword FallBackClockSource in the struct HBus which is within Clocking is expressed as Clocking.HBus.FallBackClockSource.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, refer to the OAM MIB EMC table.

emckwAllowedRange

Specifies the range of allowed values for the EMC keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.3.1.1.7.***n***

Details

If the keyword type is Integer, and is a yes/no choice, **emckwAllowedRange** contains a string of this format:

Nb values=2: Yes,No

If the keyword type is Integer, and can take a range of values, **emckwAllowedRange** contains a string of this format:

BASE ***base***:***min_value*** <> ***max_value***

where:

- ***base*** is a mathematical base of the integer (for example, 16 for a hexadecimal number).
- ***min_value*** is the minimum allowed value.
- ***max_value*** is the maximum allowed value.

For example:

```
BASE 10: 0 <> 65535
```

If the keyword type is String, **emckwAllowedRange** contains all the allowed strings for this keyword, separated by commas (,). For example: YES,NO. If any string is acceptable, this field contains <no range>.

emckwAllowedRange reflects the combined values of the Base, Min, Max, and Choices qualifiers for the keyword in the NMS OAM database.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, refer to the OAM MIB EMC table.

emckwDescription

Provides a short description of the EMC keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.3.1.1.8.***n***

Details

emckwDescription is equivalent to the value of the Description qualifier for the keyword in the NMS OAM database. If no description is given, **emckwDescription** contains <none>.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, refer to the OAM MIB EMC table.

emcKwIndex

Indicates the keyword's index.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.2.***n***

Details

Keywords are numbered sequentially starting at 1.

Configuration

This value is determined when the OAM Database SNMP agent populates the Extended Management Component table as described in the OAM MIB EMC table.

emckwMode

Indicates if the EMC keyword is read-only or read-write.

Syntax

Integer { readOnly(1), readWrite(2) }

Access

Read-only

OID

p.3.1.1.6.***n***

Details

emckwMode reflects the value of the keyword's ReadOnly qualifier in the NMS OAM database.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, see OAM MIB EMC table.

emckwType

Indicates the type of the EMC keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.3.1.1.5.***n***

Details

Valid EMC keyword types include:

Type	Description
Integer	An integer.
String	A string of 0 or more characters.

emckwType is equivalent to the value of the Type qualifier for the keyword in the NMS OAM database.

Keywords of other types (for example: Array, Struct, StructAndArray) are not included as separate entries in MIB tables. For more information, see Keywords in the OAM database MIB.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, refer to the OAM MIB EMC table.

emckwValue

Indicates the EMC keyword value

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.3.1.1.4.***n***

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, refer to the OAM EMC table.

emcTable

Starts a sequence of **emcEntry** objects, each of which composes a row in the Extended Management Component table.

Syntax

Object

Access

Not accessible.

OID

p.3.1

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Extended Management Component table is populated, refer to the OAM EMC table.

keywordName

Indicates a Supervisor keyword name.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.4.1.2.***n***

Details

When a keyword belongs to one or more arrays or structures, the array and structure names are appended to the keyword name in the table, separated by periods (.). For example, the keyword FallBackClockSource in the struct HBus which is within Clocking is expressed as Clocking.HBus.FallBackClockSource.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Supervisor Keyword table is populated, refer to the OAM supervisor tables.

kwAllowedRange

Indicates the range of allowed values for the Supervisor keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.4.1.6.***n***

Details

If the keyword type is Integer, and is a yes/no choice, **kwAllowedRange** contains a string of this format:

```
Nb values=2: Yes,No
```

If the keyword type is Integer, and can take a range of values, **kwAllowedRange** contains a string of this format:

BASE ***base***:***min_value*** <> ***max_value***

...where:

- ***base*** is a mathematical base of the integer (for example, 16 for a hexadecimal number).
- ***min_value*** is the minimum allowed value.
- ***max_value*** is the maximum allowed value.

For example:

```
BASE 10: 0 <> 65535
```

If the keyword type is String, **kwAllowedRange** contains all the allowed strings for this keyword, separated by commas (,). For example: YES,NO. If any string is acceptable, this field contains <no range>.

If the keyword type is Object, no possible values are given.

kwAllowedRange reflects the combined values of the Base, Min, Max, and Choices qualifiers for the keyword in the NMS OAM database.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Supervisor Keyword table is populated, refer to the OAM supervisor tables.

kwDescription

Provides a short description of the Supervisor keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.4.1.7.***n***

Details

kwDescription is equivalent to the value of the Description qualifier for the keyword in the NMS OAM database. If no description is given, **kwDescription** contains <none>.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Supervisor Keyword table is populated, refer to the OAM supervisor tables.

kwMode

Indicates if the Supervisor keyword is read-only or read-write.

Syntax

Integer { readOnly(1), readWrite(2) }

Access

Read-only

OID

p.1.4.1.5.***n***

Details

kwMode reflects the value of the keyword's ReadOnly qualifier in the NMS OAM database.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Supervisor Keyword table is populated, refer to the OAM supervisor tables.

kwType

Indicates the type of the Supervisor keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.4.1.4.***n***

Details

Valid types include:

Type	Description
Integer	An integer.
String	A string of 0 or more characters.
Object	An EMC, board plug-in, or board managed object.

kwType is equivalent to the value of the Type qualifier for the keyword in the NMS OAM database.

Keywords of other types (for example, Array, Struct, StructAndArray) are not included as separate entries in MIB tables. For more information, see Keywords in the OAM database MIB.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Supervisor Keyword table is populated, refer to the OAM supervisor tables.

kwValue

Indicates the Supervisor keyword value.

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.1.4.1.3.***n***

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Supervisor Keyword table is populated, refer to the OAM supervisor tables.

oamAlertRegister

Enables or disables the sending of NMS OAM alert messages and events as SNMP traps.

Syntax

Integer { enable(1), disable(2) }

Access

Read-write

OID

p.1.3

Details

Reading this value determines its current setting. For more information, refer to OAM MIB events.

Configuration

Not applicable.

oamBoardPlugins

Provides the start of the OAM Board Plug-in table containing, for each board plug-in, a table of the plug-in's keywords, values, and qualifiers.

Syntax

Object

Access

Not accessible.

OID

p.2

Details

The objects in the OAM Board Plug-in table are:

Object	Description
boardPluginTable	Top of the table.
boardPluginEntry	Starts a row of the OAM Board Plug-in table.
boardPluginIndex	Plug-in index. This is equivalent to the index number of the BoardPlugins[x] keyword listing the board plug-in in the Supervisor managed object (see below).
boardPluginKwIndex	Unique index (within this table) identifying the keyword.
bpikeywordName	The keyword name, formatted as described in OAM database MIB tables and keywords .
bpikwValue	The value of the keyword.
bpikwType	The type of the keyword: `Integer` or `String`.
bpikwMode	`1` indicates keyword value is read/write. `0` indicates keyword value is read-only.
bpikwAllowedRange	The range of allowable values for the keyword, formatted as described in OAM database MIB tables and keywords.
bpikwDescription	A short description of the keyword.

Configuration

Not applicable.

oamBoards

Starts the OAM Boards table,

Syntax

Object

Access

Not accessible.

OID

p.4

Details

This entry contains:

- The number of boards automatically detected in the system
- The total number of boards registered to NMS OAM
- A table of boards, each with their keywords, values and qualifiers
- Values that allow you to start or stop a board, test a board, delete a board instance from the database, change a board's name or number, or query its status.

The objects in the OAM Boards table are:

Object	Description
detectedBoardCount	Number of boards automatically detected in the system.
createdBoardCount	Total number of boards registered to NMS OAM.
boardTable	Board Keyword table, containing a list of boards, each with their keywords, values, and qualifiers.
boardManagementTable	Board Management table, containing values that allow you to start, stop, test, or delete a board, change a board's name or number, or query its status.

Configuration

Not applicable.

oamCreateBoard

Starts the Create Board table containing values that allow you to create board instances in the database.

Syntax

Object

Access

Not accessible.

OID

p.1.5

Details

The objects in the Create Board table are:

Object	Description
productName	Product type of the board to create.
boardName	Name to give the created board.
boardNumber	Board number to give the created board.
applyBoardCommand	Set this to 1 to create the board based upon the **productName**, **boardName**, and/or **boardNumber** values you specified.

Configuration

Not applicable.

oamEMCs

Starts the Extended Management Component (EMC) table containing, for each EMC, a table of the EMC's keywords, values, and qualifiers.

Syntax

Object

Access

Not accessible.

OID

p.3

Details

The objects in the Extended Management Component (EMC) table are:

Object	Description
emcTable	Top of the table.
emcEntry	Starts a row of the Extended Management Component table.
emcIndex	EMC index.
emcKwIndex	Unique index (within this table) identifying the keyword.
emckeywordName	The keyword name, formatted as described in Keywords in the OAM database MIB.
emckwValue	The value of the keyword.
emckwType	The type of the keyword: `Integer` or `String`.
emckwMode	`1` indicates keyword value is read/write. `0` indicates keyword value is read-only.
emckwAllowedRange	Range of allowable values for the keyword, formatted as described in Keywords in the OAM database MIB.
emckwDescription	A short description of the keyword.

Configuration

Not applicable.

oamEventDescription

Returns a string containing the last event sent back by OAM (for more information, refer to OAM MIB events).

Syntax

String (SIZE 0..255)

Access

Read-only

OID

p.6.1

Configuration

Updated whenever a new OAM event is generated.

oamEventMask

Determines the mask to use to filter NMS OAM events.

Syntax

Integer

Access

Read-write

OID

p.1.2

Details

Reading this value returns the current event mask setting. If no mask is set, this value returns -1 (0xFFFFFFFF). For more information, refer to OAM MIB events.

Configuration

Not applicable.

oamEventsTraps

Starts the OAM Traps table allowing an application to receive OAM events through the MIB.

Syntax

Object

Access

Not accessible.

OID

p.6

Details

Objects in the OAM Events Traps table (**oamEventsTraps**) include:

Object	Description
oamEventDescription	The last event sent back by OAM.

Configuration

Not applicable.

oamOtherObjects

Starts the Other Objects table.

Syntax

Object

Access

Not accessible.

OID

p.5

Details

The Other Objects table is included so that future extensions to NMS OAM will not require changes to the structure of the OAM Database MIB. The objects in the Other Objects table are:

Object	Description
otherObjectsTable	Top of the table.
otherObjectsEntry	Starts a row of the Other Objects table.
otherObjectsIndex	Object index.
otherObjectsKwIndex	Unique index (within this table) identifying the keyword.
otherObjectskeywordName	The keyword name, formatted as described in OAM database MIB tables and keywords.
otherObjectskwValue	The value of the keyword.
otherObjectskwType	The type of the keyword: `Integer` or `String`.
otherObjectskwMode	`1` indicates keyword value is read/write. `0` indicates keyword value is read-only.
otherObjectskwAllowedRange	The range of allowable values for the keyword, formatted as described in OAM database MIB tables and keywords.
otherObjectskwDescription	A short description of the keyword.

Configuration

Not applicable.

oamStartStop

Stop or start the NMS OAM Supervisor, or queries its status (for more information, refer to Starting and stopping the supervisor).

Syntax

Integer { oamStart(1), oamStop(2) }

Access

Read-write

OID

p.1.1

Configuration

Not applicable.

oamSupervisor

Starts a Supervisor group.

Syntax

Object

Access

Not accessible.

OID

p.1

Details

The Supervisor group contains the following objects:

Object	Description
oamStartStop	Starts or stops the NMS OAM Supervisor process, or indicates its status.
oamEventMask	Sets the NMS OAM event mask, or indicates its status.
oamAlertRegister	Registers for NMS OAM alert notification, or indicates the status of the registration.
supervisorTable	Supervisor Keyword table, containing NMS OAM Supervisor keywords, values and qualifiers.
oamCreateBoard	Create Board table, containing values that allow you to create board instances in the database.

Configuration

Not applicable.

otherObjectsEntry

Starts a row in the Other Objects table (**otherObjectsTable**).

Syntax

Object

Access

Not accessible.

OID

p.5.1.1

Details

Each row in the Other Objects table contains information about a keyword.

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

otherObjectsIndex

Indicates the managed object to which the keyword belongs.

Syntax

Integer

Access

Read-only

OID

p.5.1.1.1.***n***

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

otherObjectskwAllowedRange

Indicates the range of allowed values for the keyword

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.5.1.1.7.***n***

Details

If the keyword type is Integer, and is a yes/no choice, **otherObjectskwAllowedRange** contains a string of this format:

```
Nb values=2: Yes,No
```

If the keyword type is Integer, and can take a range of values, **otherObjectskwAllowedRange** contains a string of this format:

BASE ***base***:***min_value*** <> ***max_value***

...where:

- ***base*** is a mathematical base of the integer (for example, 16 for a hexadecimal number).
- ***min_value*** is the minimum allowed value.
- ***max_value*** is the maximum allowed value.

For example:

```
BASE 10: 0 <> 65535
```

If the keyword type is String, **otherObjectskwAllowedRange** contains all the allowed strings for this keyword, separated by commas (,). For example: YES,NO. If any string is acceptable, this field contains `<no range>`.

otherObjectskwAllowedRange reflects the combined values of the Base, Min, Max, and Choices qualifiers for the keyword in the NMS OAM database.

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

otherObjectskwDescription

Provides a short description of the keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.5.1.1.8.***n***

Details

otherObjectskwDescription is equivalent to the value of the Description qualifier for the keyword in the NMS OAM database. If no description is given, **otherObjectskwDescription** contains <none>.

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

otherObjectsKwIndex

Indicates the keyword's index.

Syntax

Integer

Access

Read-only

OID

p.5.1.1.2.***n***

Details

Keywords are numbered sequentially from 1 upward.

Configuration

This value is determined when the OAM Database SNMP agent populates the Other Objects table.

otherObjectskwMode

Indicates if the keyword is read-only or read-write.

Syntax

Integer { readOnly(1), readWrite(2) }

Access

Read-only

OID

p.5.1.1.6.***n***

Details

otherObjectskwMode reflects the value of the keyword's ReadOnly qualifier in the NMS OAM database.

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

otherObjectskeywordName

Indicates the name of a keyword in the managed object for the object.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.5.1.1.3.***n***

Details

Where a keyword belongs to one or more arrays or structures, the array and structure names are appended to the keyword name in the table, separated by periods (.). For example, the keyword FallBackClockSource in the struct HBus which is within Clocking is expressed as Clocking.HBus.FallBackClockSource.

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

otherObjectskwType

Indicates the type of the keyword.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.5.1.1.5.***n***

Details

Valid keyword types are:

Type	Description
Integer	An integer.
String	A string of 0 or more characters.

otherObjectskwType is equivalent to the value of the Type qualifier for the keyword in the NMS OAM database.

Keywords of other types (for example: Array, Struct, StructAndArray) are not included as separate entries in MIB tables. For more information, see OAM Database MIB tables and keywords.

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

otherObjectskwValue

Indicates a keyword value.

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.5.1.1.4.***n***

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

otherObjectsTable

Syntax

Object

Access

Not accessible.

OID

p.5.1

Details

Starts a sequence of **otherObjectsEntry** objects, each of which composes a row in the Other Objects table.

Configuration

When the OAM Database SNMP agent starts up, it populates all MIB tables based upon values from the NMS OAM database. For more information, refer to Populating OAM MIB tables.

productName

Product name of the board to create (for more information, refer to Creating and deleting board managed objects).

Syntax

DisplayString (SIZE 0..255)

Access

Read-write

OID

p.1.5.1

Configuration

Configured by the user as necessary.

supervisorIndex

Indicates the keyword's index.

Syntax

Integer

Access

Read-only

OID

p.1.4.1.1.***n***

Details

Keywords are numbered sequentially from 1 upward.

Configuration

When it starts up, the OAM Database SNMP agent populates all MIB tables based upon information from the NMS OAM database. For more information on how the Supervisor Keyword table is populated, refer to OAM MIB supervisor tables.

supervisorEntry

Starts a row in the Supervisor Keyword table (**supervisorTable**).

Syntax

Object

Access

Not accessible.

OID

p.1.4.1

Details

Each row in the Supervisor table contains information about a Supervisor keyword. The number of rows is exactly equal to **supervisorIndex**.

Configuration

The rows in the Supervisor Keyword table are configured by the OAM Database SNMP agent when it starts up. For more information, refer to Populating OAM MIB tables.

supervisorTable

Starts a sequence of **supervisorEntry** objects, each of which composes a row in the Supervisor Keyword table.

Syntax

Object

Access

Not accessible.

OID

p.1.4

Details

The number of rows in the Supervisor table is exactly equal to **supervisorIndex**.

Configuration

The rows in the Supervisor Keyword table are configured by the OAM Database SNMP agent when it starts up. For more information, refer to Populating OAM MIB tables.

Using the NMS OAM database MIB

Accessing keywords for boards, plug-ins, or EMCs

To access a particular keyword for a board, a board plug-in, or an EMC:

1. Determine the index of the managed object containing the keyword. To do so, access the Supervisor Keyword table (**supervisorTable**), and search for the managed object name in one of the following array keywords:

For this managed object...	Search this array...
Board	Boards[***x***]
Board Plug-in	BoardPlugins[***x***]
EMC	ExtendedManagementComponents[***x***]

2. Access the table containing keywords for the managed object type:

For this managed object...	Access this table...
Board	**boardTable**
Board Plug-in	**boardPluginTable**
EMC	**emcTable**

   Each of these tables is doubly linked. The first index, the managed object index, maps to the index you determined in step 1.

3. Within the entries in the table beginning with the desired index, search for the keyword.
4. To set a keyword, first determine that it is read/write. If it is, make sure the type (for example, integer or string) of your setting is correct for the keyword, and is within the range of allowed values.

Board settings do not take effect until the board is stopped and restarted.

Creating and deleting board managed objects

You can use the items in the Create Board table (**oamCreateBoard**) to add a board managed object to the NMS OAM database. To do so:

Note: This operation does not require that the board currently be physically installed in the system.

1. Specify a valid product name for **productName**. A list of valid product names can be retrieved by querying the Supervisor keyword Products[x].
2. (Optional) Specify a board name for **boardName**.
3. (Optional) Specify a board number for **boardNumber**.
4. Set **applyBoardCommand** to 1.

   A board managed object for product **productName** is added to the NMS OAM database. If you did not specify a board name or number, default values are generated.
5. Access and modify the board's keywords (as described in Accessing board, plug-in, and EMC keywords) to perform further configuration. In particular, modify the Location.PCI.Bus and Location.PCI.Slot keywords to specify the location of the board for NMS OAM.
6. If the board is physically installed in the system, start the board, as described in Starting, stopping, and testing boards.

Deleting board managed objects

To delete a board managed object:

1. Stop the board as described in Starting, stopping, and testing boards.
2. Find the **boardManagementIndex** with the index value of the managed object for the board.
3. Set **brdDelete** in this row to 1.

Querying and setting the board name and number

For a board to be available, it must exist as a managed object in the NMS OAM database.

Querying or setting the board number of a board

To set or query the name of a board:

1. Determine the index of the board managed object.

   To do so, access the Supervisor Keyword table (**supervisorTable**), and search for the board name in the Boards[x] array keyword. The index of the board name in the array maps to the index of the board managed object.
2. Find the **boardManagementIndex** with the index value.
3. To set the name of the board, set **brdName** in this row to the new name. To query the board's name, query **brdName**.

Querying or setting the board number of a board

For a board to be available, it must exist as a managed object in the NMS OAM database.

To set or query the board number of a board:

1. Determine the index of the board managed object.

   To do so, access the Supervisor Keyword table (**supervisorTable**), and search for the board name in the Boards[x] array keyword. The index of the board name in the array maps to the index of the board managed object.
2. Find the **boardManagementIndex** with the index value.
3. To set the board number of the board, set **brdNumber** in this row to the new board number. To query the board's number, query **brdNumber**.

Starting, stopping, and testing boards

You can use the OAM database MIB to start, stop, and test boards in the system.

Starting and stopping boards

For a board to be available for starting, it must exist as a managed object in the NMS OAM database.

To start or stop a board, or query its status:

1. Determine the index of the board managed object.

   To do so, access the Supervisor Keyword table (**supervisorTable**), and search for the board name in the Boards[x] array keyword. The index of the board name in the array maps to the index of the board managed object.
2. Find the **boardManagementIndex** with the index value.
3. To start the board, set **brdStartStop** in this row to `1`. To stop the board, set **brdStartStop** in this row to `2`. To query the status of the board, query **brdStartStop**.

Testing boards

For a board to be available for testing, it must exist as a managed object in the NMS OAM database.

To test a board:

1. Determine the index of the board managed object.

   To do so, access the Supervisor Keyword table (**supervisorTable**), and search for the board name in the Boards[x] array keyword. The index of the board name in the array maps to the index of the board managed object.
2. Find the **boardManagementIndex** with the index value.
3. Set **brdTest** in this row to the board test level you wish to run (an integer between 1 and 255). For more information about board testing, refer to the *NMS OAM System User's Manual*.

Starting and stopping the supervisor

You can stop and restart the NMS OAM Supervisor using the OAM Database MIB.

- To stop the Supervisor, set **oamStartStop** to 2.
- To start the Supervisor, set **oamStartStop** to 1.

You can determine the current status (stopped or running) of the Supervisor by querying **oamStartStop**.

Note: If you query this keyword while the Supervisor is in the process of shutting down, the keyword indicates that the Supervisor is running.

OAM MIB events

NMS OAM events (both solicited and unsolicited) are available via SNMP. An SNMP application can receive them either as SNMP traps, or by querying the OAM database MIB.

To receive NMS OAM events as SNMP traps, set **oamAlertRegister** to 1. To stop receiving events as traps, set **oamAlertRegister** to 2.

Regardless of whether SNMP is registered to receive NMS OAM events, an application can always determine the last event received by querying **oamEventDescription**. This value contains a string of the form:

eventname name=***objectname***

... where:

- ***eventname*** is the name of the last event received (for example: OAMEVN_STARTBOARD_DONE).
- ***objectname*** is the name of the object sending the event (for example: MyBoard).

For example: OAMEVN_STARTBOARD_DONE name=MyBoard

The events in the following table are reported slightly differently in **oamEventDescription**:

Event name	String in oamEventDescription
OAMEVN_ALERT	***eventname*** name=***objectname*** message=***message*** ***message*** is the alert message sent
OAMEVN_REPORT	***eventname*** name=***objectname*** message=***message*** ***message*** is the alert message sent
OAMEVN_TRACE	***eventname*** name=***objectname*** message=***message*** ***message*** is the alert message sent
OAMEVN_RENAMED	***eventname*** oldname=***oldname*** newname=***newname*** ***oldname*** is the original name of the board. ***newname*** is the new name of the board

You can mask the alerts received by SNMP (either as traps or by querying the MIB) by setting **oamEventMask**. The following are valid mask values:

Mask	Value	Description
TRAP_MASK_OAMEVN_ALERT	0x1	An OAM alert has been generated
TRAP_MASK_OAMEVN_REPORT	0x2	Special internal code used to log report info
TRAP_MASK_OAMEVN_CREATED	0x4	Object was created
TRAP_MASK_OAMEVN_DELETED	0x8	Object was deleted
TRAP_MASK_OAMEVN_RENAMED	0x10	Object was renamed (text = new name)
TRAP_MASK_OAMEVN_TRACE	0x20	Indicates trace info (potentially high-speed)
TRAP_MASK_OAMEVN_MODIFIED	0x40	Object was modified (closed after write access)
TRAP_MASK_OAMEVN_BOARD_DEAD	0x80	A board has failed
TRAP_MASK_OAMEVN_STARTBOARD_DONE	0x100	A board was successfully started
TRAP_MASK_OAMEVN_STOPBOARD_DONE	0x200	A board was successfully stopped
TRAP_MASK_OAMEVN_TESTBOARD_DONE	0x400	A board test was successfully initiated
TRAP_MASK_HSWEVN_REMOVAL_REQUESTED	0x800	A board extraction has begun, or board extraction was initiated in software
TRAP_MASK_HSWEVN_BOARD_OFFLINE	0x1000	A board has gone off line
TRAP_MASK_HSWEVN_BOARD_REMOVED	0x2000	A board has been removed
TRAP_MASK_HSWEVN_BOARD_INSERTED	0x4000	A board has been inserted
TRAP_MASK_HSWEVN_ONLINE_PENDING	0x8000	A board has been inserted, and is about to go online
TRAP_MASK_HSWEVN_PCI_CONFIG_FAILED	0x10000	A PCI configuration attempt failed
TRAP_MASK_HSWEVN_PREPARATION_FAILED	0x20000	Preparation for board removal failed
TRAP_MASK_HSWEVN_BOARD_READY	0x40000	A board is ready

By default, no masks are set.

For more information about NMS OAM events, refer to your *NMS OAM Service Developer's Reference Manual*.

Demonstration programs

Using SNMP demonstration programs

The demonstration programs show how you can use the information in the NMS MIBs to provide useful information to a network administrator, and how to get and set SNMP variables.

All demonstration programs are run by executing the program from the command line. Each demonstration program resides in its own directory under *\nms\ctaccess\demos\snmp* (or the */opt/nms/ctaccess/demos/snmp* directory under UNIX), along with the source code and makefile.

The following demonstration programs are provided:

Program	Description
snmpget	Retrieves information about the SNMP master agent on the specified host.
snmpnext	Gets the value of the next SNMP variable.
snmpset	Sets the value of the current SNMP variable.
snmpChassScan	Navigates the NmsChassis MIB, and displays information about the chassis and boards.
snmpHsMon	Monitors a CompactPCI chassis for traps.
snmpTrunkLog	Shows the status of digital trunks.

You must start the Natural Access server with *ctdaemon.exe*, and initialize the system hardware with *oamsys* before running the SNMP demonstration programs.

Note: *snmpHsMon* is the only demonstration program that supports board insertion/extraction.

snmpget

Purpose

Demonstrates SNMP **get**. Retrieves and displays information about a specified SNMP agent running at a specified IP address.

Usage

snmpget ***address nmssnmpoid options***

Use the following parameters:

Parameter	Description
address	The address or DNS name of a local or remote host running an SNMP agent about which to return information.
nmssnmpoid	OID of an object in one of the MIBs available on the host for which you wish to see information. The default is sysDescr.

Valid ***options*** include:

Option	Description
-v1	Use SNMPv1 (default).
-v2	Use SNMPv2.
-c***community_name***	Specify a community name. The default is public.
-r***n***	Number of retries. The default is 1 retry.
-t***n***	Timeout in hundredths of a second. The default is 100 (1 second).

Procedure

To run *snmpget*:

1. From the command line, navigate to the *\nms\ctaccess\demos\snmp\snmpget* directory (or the */opt/nms/ctaccess/demos/snmp/snmpget* directory under UNIX).
2. Enter the following:

```
snmpget localhost
```

An example of running this command is:

```
< snmpGet.exe localhost
> SNMP++ Get to localhost SNMPV1 Retries=1 Timeout=100ms Community=public
> oid = 1.3.6.1.2.1.1.1.0
> Value = Hardware: x86 Family 6 Model 3 Stepping 4 AT/AT COMPATIBLE  -
> Software: Windows 2000
```

snmpnext

Purpose

Demonstrates SNMP **get-next**. Retrieves the value of the next object after a specified OID.

Usage

snmpnext ***address cnmssnmpoid options***

Use the following parameters:

Parameter	Description
address	The address or DNS name of a local or remote host running an SNMP agent about which to return information.
nmssnmpoid	OID of an object in one of the MIBs available on the host for which you wish to see information. The default is sysDescr.

Valid ***options*** include:

Option	Description
-v1	Use SNMPv1 (default).
-v2	Use SNMPv2.
-c***community_name***	Specify a community name. The default is public.
-r***n***	Number of retries. The default is 1 retry.
-t***n***	Timeout in hundredths of a second. The default is 100 (1 second).

Procedure

To run *snmpnext*:

1. From the command line, navigate to the *\nms\ctaccess\demos\snmp\snmpnext* directory (or the */opt/nms/ctaccess/demos/snmp/snmpnext* directory under UNIX).
2. Enter the following:

```
snmpnext 10.1.20.46 1.3.6.1.2.1.1.1.0
```

An example of running this command is:

```
>snmpnext 10.1.20.46 1.3.6.1.2.1.1.1.0
SNMP++ GetNext to 10.1.20.46 SNMPV1 Retries=1 Timeout=1000ms Community=public
Oid = 1.3.6.1.2.1.1.2.0
Value = 1.3.6.1.4.1.311.1.1.3.1.1
```

snmpset

Purpose

Demonstrates SNMP **set**. Sets the value of a specified SNMP object.

Usage

snmpset ***address nmssnmpoid options***

Use the following parameters:

Parameter	Description
address	The address or DNS name of a local or remote host running an SNMP agent about which to return information.
nmssnmpoid	OID of an object in one of the MIBs available on the host for which you wish to see information. The default is sysDescr.

Valid ***options*** include:

Option	Description
`-v1`	Use SNMPv1 (default).
`-v2`	Use SNMPv2.
`-c`***community_name***	Specify a community name. The default is public.
`-r`***n***	Number of retries. The default is 1 retry.
`-t`***n***	Timeout in hundredths of a second. The default is 100 (1 second).

Procedure

To run *snmpset*:

1. From the command line, navigate to the *\nms\ctaccess\demos\snmp\snmpset* directory (or the */opt/nms/ctaccess/demos/snmp/snmpset* directory under UNIX).
2. Enter the following:

```
snmpset localhost 1.3.6.1.4.1.2628.2.2.4.2.0
```

Note: An example of running this command to set the chassBoardTrapEnable follows:

```
>snmpSet.exe localhost 1.3.6.1.4.1.2628.2.2.4.2.0
>SNMP++ Set to localhost SNMPV1 Retries=1 Timeout=100ms
>CNmsSnmpOid = 1.3.6.1.4.1.2628.2.2.4.2.0
>Current Value = 2
>Value Type is Integer
>Value ?
```

The program asks for new value. In this example, enter `1` to enable traps.

```
<Value ?1
>Set Status = Success
```

snmpChassScan

Purpose

Demonstrates how to navigate the chassis MIB, how to retrieve chassis type and description, and how to navigate by bus, recognize ISA and PCI boards, and show board description and status information.

Usage

snmpChassScan ***address options***

Use the following parameter:

Parameter	Description
address	The address or DNS name of a local or remote host running an SNMP agent about which to return information.

Valid ***options*** include:

Option	Description
`-c`***community_name***	Specify a community name. The default is public.
`-r`***n***	Number of retries. The default is 1 retry.
`-t`***n***	Timeout in hundredths of a second. The default is 100 (1 second).

Polling is set interactively by the application. See the example in the Procedure section below.

Procedure

To run *nmsChassScan*:

1. From the command line, navigate to the *\nms\ctaccess\demos\snmp\snmpchassscan* directory (or the */opt/nms/ctaccess/demos/snmp/snmpchassscan* directory under UNIX).
2. Enter the following:

```
nmsChassScan
```

The following example shows running *snmpChassScan*:

```
< snmpChassScan.exe
> SNMP Demonstration and Test Program  V.3.0 Nov 15 1999
> NMS Communications Corporation.

>Usage:
>snmpChassScan [Address | DNSName] [options]
>Address:  default is 127.0.0.1
>options: -cCommunity_name, specify community default is 'public'
>         -rN , retries default is N = 1 retry
>         -tN , timeout in hundredths-seconds default is N = 100 = 1 second
>
>H Help  S Sys info  L Board list  P<N> Poll Interval  Q Quit
>
>SEND A REQUEST FOR SYSTEM INFO TO: 10.1.20.45
>System information:
>System:        Hardware: x86 Family 6 Model 3 Stepping 4 AT/AT COMPATIBLE  - >Software:
Windows 2000
>SysUpTime:     1:22:15.66
>SysContact:    Joe Kilroy
>Computer name: KILROY
>Location:      NMS
>
>SEND A REQUEST FOR NMS BOARDS TO: 10.1.20.45
>             PCI bus
>Board 1:       Ag-Quad-E1 Segment:1 Slot:7 Status:OnLine
>Board 2:       Ag-Quad-T1 Segment:1 Slot:6 Status:OnLine
>
```

snmpHsMon

Purpose

Demonstrates how to monitor a CompactPCI chassis: how to receive traps when board status changes, and how to remotely insert or extract a board.

Usage

snmpHsMon ***parameter***

Use the following parameter:

Parameter	Description
address	The address or DNS name of a local or remote host running an SNMP agent about which to return information.

snmpHsMon is similar to the *hsmon* utility. For more information, see the *NMS OAM System User's Manual* for Natural Access 4.0 or later, or the *Hot Swap Manager Developer's Reference Manual* for Natural Access 3.x.

Procedure

To run *snmpHsMon*:

1. From the command line, navigate to the *\nms\ctaccess\demos\snmp\snmphsmon* directory (or the */opt/nms/ctaccess/demos/snmp/snmphsmon* directory under UNIX).
2. Enter the following:

```
snmpHsMon
```

The following example shows running *snmpHsMon*:

```
>snmpHsMon.exe 10.1.20.46
SNMP Demonstration and Test Program  V.3.0 Nov 15 1999
NMS Communications Corporation.

h  Help     r  Refresh    i<N> Insert    e<N> Extract  Q  Quit

SEND A REQUEST FOR SYSTEM INFO TO: 10.1.20.46
System information:
System:        Hardware: x86 Family 5 Model 4 Stepping 3 AT/AT COMPATIBLE  - Software:
Windows 2000
SysUpTime:     1 day 1:34:44.93
SysContact:    Joe kILROY
Computer name: KILROY
Location:      NMS

SEND A REQUEST FOR NMS BOARDS TO: 10.1.20.46
             PCI bus
Board 0:       Ag-CPCI-Quad-E1 Segment:1 Slot:10 Status:OffLine
Board 1:       Ag-CPCI-Quad-T1 Segment:1 Slot:11 Status:OffLine
Board 3:       Ag-CPCI-Quad-T1 Segment:1 Slot:15 Status:OffLine
Board 2:       Ag-Quad-T1 Segment:1 Slot:13 Status:OffLine

>
< 00:20:24   3 Board OnLinePending
< 00:20:24   1 Board OnLinePending
< 00:20:24   3 Board OnLine
< 00:20:24   1 Board OnLine
< 00:20:24   0 Board OnLinePending
< 00:20:24   0 Board OnLine
< 00:20:24   2 Board OnLinePending
< 00:20:24   1 Board OffLinePending
< 00:20:24   1 Board OffLine
< 00:20:24   2 Board OnLine
< 00:20:24   3 Board OffLinePending
< 00:20:24   3 Board OffLine
< 00:20:24   0 Board OffLinePending
< 00:20:24   0 Board OffLine
>q
```

snmpTrunkLog

Purpose

Shows the status of digital trunks of each board in a chassis.

Usage

snmpTrunkLog ***address options***

Use the following parameters:

Parameter	Description
address	The address or DNS name of a local or remote host running an SNMP agent about which to return information.

Valid ***options*** include:

Option	Description
-c***Community_name***	Specify a community name. The default is public.
-r***n***	Number of retries. The default is 1 retry.
-t***n***	Timeout in hundredths of a second. The default is 100 (1 second).

snmpTrunkLog is similar to the *trunkmon* utility. See the *NMS OAM System User's Manual* for more information.

Procedure

To run *snmpTrunkLog*:

1. From the command line, navigate to the *\nms\ctaccess\demos\snmp\snmptrunklog* directory (or the */opt/nms/ctaccess/demos/snmp/snmptruklog* directory under UNIX).
2. Enter the following:

```
snmpTrunkLog 10.1.20.45
```

The following example shows *snmpTrunkLog* being run:

```
> snmpTrunkLog
SNMP Demonstration and Test Program  V.3.0 Nov 15 1999
NMS Communications Corporation.

Usage:
snmpChassScan [Address | DNSName] [options]
Address:  default is 127.0.0.1
options: -cCommunity_name, specify community default is 'public'
         -rN , retries default is N = 1 retry
         -tN , timeout in hundredths-seconds default is N = 100 = 1 second

h  Help     S  Sys info    L  Trunk list    Q  Quit

SEND A REQUEST FOR SYSTEM INFO TO: 10.1.20.45
System information:
System:        Hardware: x86 Family 6 Model 3 Stepping 4 AT/AT COMPATIBLE  - Software:
Windows 2000
SysUpTime:     1:59:37.45
SysContact:    Joe Kilroy
Computer name: KILROY
Location:      NMS

SEND A REQUEST FOR TRUNKS TO: 10.1.20.45

Interface:2 Board:1 (Ag-Quad-E1)  Trunk:0 Status: Loss of frame, NoSgnl
Interface:3 Board:1 (Ag-Quad-E1)  Trunk:1 Status: Loss of frame, NoSgnl
Interface:4 Board:1 (Ag-Quad-E1)  Trunk:2 Status: Loss of frame, NoSgnl
Interface:5 Board:1 (Ag-Quad-E1)  Trunk:3 Status: Loss of frame, NoSgnl
Interface:6 Board:2 (Ag-Quad-T1)  Trunk:0 Status: In service
Interface:7 Board:2 (Ag-Quad-T1)  Trunk:1 Status: Loss of frame, NoSgnl
Interface:8 Board:2 (Ag-Quad-T1)  Trunk:2 Status: In service
Interface:9 Board:2 (Ag-Quad-T1)  Trunk:3 Status: Loss of frame, NoSgnl
```

WBEM support under windows

Overview of WBEM support

The Distributed Management Task Force (DMTF) has launched the Web-Based Enterprise Management (WBEM) initiative that extends the Common Information Model (CIM) to represent management objects. This Common Information Model is an extensible data model for logically organizing management objects in a consistent, unified manner in a managed environment. WBEM is a technology that establishes management infrastructure standards and provides a standardized way to access information from various hardware and software management systems in an enterprise environment. Using WBEM standards, developers can create tools and technologies that reduce the complexity and costs of enterprise management. WBEM provides a point of integration through which data from management sources can be accessed, and it complements and extends existing management protocols and instrumentation such as Simple Network Management Protocol (SNMP), Desktop Management Interface (DMI), and Common Management Information Protocol (CMIP).

The Microsoft Windows Management Instrumentation (WMI) technology is the Microsoft implementation of the WBEM initiative. The Windows Management Instrumentation (WMI) technology is a management infrastructure that supports the syntax of CIM, the Managed Object Format (MOF), and a common programming interface. The MOF syntax defines the structure and contents of the CIM schema in human and machine-readable form. Windows Management Instrumentation offers a powerful set of services, including query-based information retrieval and event notification. These services and the management data are accessed through a Component Object Model (COM) programming interface. The WMI scripting interface also provides scripting support.

When running Windows 2000 installed with SNMP and WMI services, SNMP data can be accessed as WBEM data through WMI mechanisms. The WMI SNMP provider (optionally installed) performs the link between SNMP and WMI. The Microsoft SNMP provider comes with additional MIB and MOF files reflecting the standard RFC.

NMS SNMP includes demonstration programs that show how you can use WBEM to retrieve information contained in the NMS subagents. These programs are for Windows only. They can be found in *\nms\ctaccess\demos\snmp\wbem*.

Installing Microsoft WMI and the WMI SNMP Provider

To install Microsoft WMI and the WMI SNMP Provider:

1. Verify your SNMP installation.
2. Obtain the installation files if they are not included on the Windows 2000 installation CD and install these files.
3. Verify the SNMP Provider installation.

Verifying the SNMP installation

The SMNP provider can interact with an SNMP agent only when the agent is working properly. To make sure the SNMP data will be available through WBEM/WMI, first check the NMS-related information using the demonstration programs described in Using SNMP demonstration programs of this manual.

Obtaining and installing WMI software

The following sections describe how to obtain and install the Microsoft WMI and WMI SNMP Provider software under different Windows 2000 installations.

Windows 2000 server and advanced server

The WMI core is installed by default under Windows 2000. However, the SNMP provider must be manually installed. To do so, run the *wbemsnmp.exe* installation program located in the *System32\WBEM* directory of the current installation, or in the *\i386* directory of the Windows 2000 installation CD.

Windows 2000 Professional

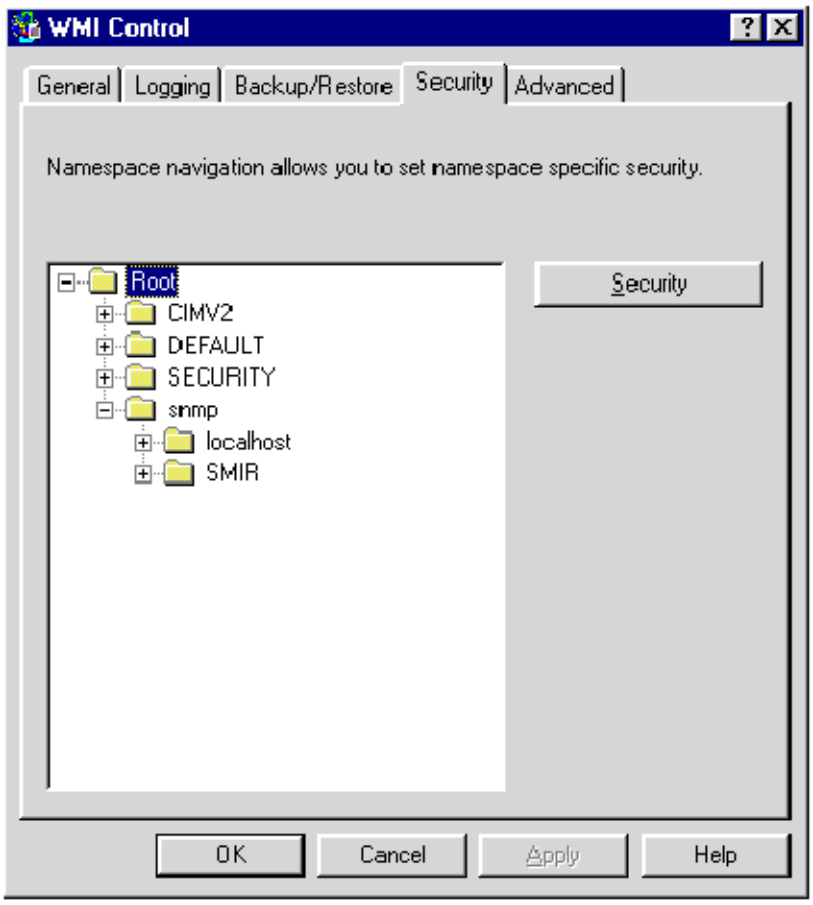
As with the Windows 2000 Server, the WMI core is installed by default, but the SNMP provider must be manually installed. If the software is not located on the installation CD, you can download the installation file *wmisnmp.exe* from Microsoft's Web site. This file can be found at: *http://download.microsoft.com/download/platformsdk/SNMPX86/1.5/NT45/EN-US/wmisnmp.exe.*

Verifying the SNMP provider installation

The SNMP provider installation automatically creates the following namespaces with WMI:

- \root\snmp\localhost
- \root\snmp\SMIR

To check that the namespaces have been properly created, browse for the namespaces in the WMI Control dialog box, shown in the following illustration:



WMI Control dialog box

To access this dialog box in Windows 2000:

1. Double-click on **Administrative Tools**.

   The Administrative Tools window appears.
2. Double-click on **Computer Management**.

   The Computer Management window appears.
3. Under Services and Applications, highlight WMI Control.
4. In the **Action** menu, click **Properties**.

   The WMI Control Properties dialog box appears.
5. Click on the **Security** tab in this dialog box.

To access the WMI Control dialog box in Windows 2000:

1. Click Start --> Programs --> Administrative Tools --> WMI Config Manager.

   The WMI Control Properties dialog box appears.
2. Click on the **Security** tab in this dialog box.

You can also check namespaces using the CIM studio in the WBEM SDK (if installed).

Installing NMS MOF files in the WBEM repository

The following MOF files can be found in the *\nms\ctaccess\demos\snmp\wbem* directory:

Filename	Description
nmsChassis.mof	MOF file of the Chassis MIB.
nmsTrunk.mof	MOF file of the Trunk MIB.
nmsOamDatabase.mof	MOF file of the OAM Database MIB.
nmsSoftRev.mof	MOF file of the Software Revision MIB.
nmsRtp.mof	MOF file of the RTP MIB. Installed with the Fusion package.

If the namespaces are properly created, SNMP-related MOF files can be added into the CIMOM repository. To do so:

1. Open an MS-DOS console.
2. Navigate to the directory *\nms\ctaccess\demos\snmp\wbem*
3. Enter the following for each MOF file:

   ```
   mofcomp mof_filename
   ```

   where ***mof_filename*** is the name of the MOF file associated with the component to start.

   mofcomp responds with information similar to the following:

   ```
   Parsing MOF file: nmsChassis.mof
   MOF file has been successfully parsed
   Storing data in the repository...
   Done!
   ```

To view the contents of the repository, enter:

```
smi2smir /l
```

Information like the following appears:

```
smi2smir : Version 1.50.1085.0000

smi2smir : Modules in the SMIR :
"NMS_CHASSIS"
```

Under certain circumstances, the repository is not updated correctly by the *mofcomp* utility. If you experience this problem, do the following:

1. Delete the entire repository by entering:

   ```
   smi2smir /p
   ```

2. Add the MOF files as described in the above procedure.
3. Stop the WMI service by entering:

   ```
   net stop winmgmt
   ```

4. Restart the WMI service by entering:

   ```
   net start winmgmt
   ```

The WMI repository should be correctly set up.

Testing MOF files

Once the MOF files have been successfully compiled and inserted, test your setup using one of the SNMP enumeration example programs provided with the NMS WBEM software:

Program	Description
enumsnmp.js	A JScript program which enumerates SNMP objects in the system.
enumsnmp.htm	An HTML file containing an embedded JScript program, which enumerates SNMP objects in the system.

Both programs can be found in *\nms\ctaccess\demo\snmp\wbem*.

Note: The console mode WSH interpreter is faster than using the Windows WSH interpreter or the embedded Jscript.

Using enumsnmp.js

To launch *enumsnmp.js*, you can do either of the following:

- Double click on the file *enumsnmp.js* in a Windows Explorer window.

  This launches the script with *wscript.exe*, the default WSH (Windows Scripting Host) interpreter. If *enumsnmp.js* is launched this way, a dialog box appears for each SNMP object found through WBEM and for each property/value pair.

- Open an MS DOS console window, and enter: `cscript enumsnmp.js`

  If *enumsnmp.js* is launched this way, the console mode WSH interpreter (*cscript.exe*) is used instead of *wscript.exe*, and the entire list of SNMP objects, properties and values in the system appears in the console window.

The following example is partial output of *enumsnmp.js* when launched with *cscript*:

```
C:\NMS\CTAccess\Demos\snmp\wbem>cscript enumsnmp.js
Microsoft (R) Windows Script Host Version 5.1 for Windows
Copyright (C) Microsoft Corporation 1996-1999. All rights reserved.

Object of class : SNMP_OAMDATABASE_MIB_oamCreateBoard : 4 propertie(s)
   Property : applyBoardCommand         Value : donothing
   Property : boardName         Value :
   Property : boardNumber               Value : -1
   Property : productName               Value :
Object of class : SNMP_OAMDATABASE_MIB_emcTable : 8 propertie(s)
   Property : emcIndex          Value : 1
   Property : emckeywordName            Value : Name
   Property : emckwAllowedRange         Value : <no range>
   Property : emckwDescription          Value : <none>
   Property : emcKwIndex                Value : 1
   Property : emckwMode         Value : readOnly
   Property : emckwType         Value : Object
   Property : emckwValue                Value : clkmgr.emc
Object of class : SNMP_OAMDATABASE_MIB_oamEventsTraps : 1 propertie(s)
   Property : oamEventDescription               Value :
Object of class : SNMP_OAMDATABASE_MIB_boardPluginTable : 8 propertie(s)
   Property : boardPluginIndex          Value : 1
   Property : boardPluginKwIndex                Value : 1
   Property : bpikeywordName            Value : BootDiagnosticLevel
   Property : bpikwAllowedRange         Value : Base 10: 0 <> 3
   Property : bpikwDescription          Value : <none>
   Property : bpikwMode         Value : readWrite
   Property : bpikwType         Value : Integer
   Property : bpikwValue                Value : 0
Object of class : SNMP_OAMDATABASE_MIB_oamBoards : 2 propertie(s)
   Property : createdBoardCount         Value : 0
   Property : detectedBoardCount                Value : 0
Object of class : SNMP_OAMDATABASE_MIB_oamSupervisor : 3 propertie(s)
   Property : oamAlertRegister          Value : disable
   Property : oamEventMask              Value : -1
   Property : oamStartStop              Value : oamStop
Object of class : SNMP_OAMDATABASE_MIB_supervisorTable : 7 propertie(s)
   Property : keywordName             Value : ExtendedManagementComponents[0]
            .
            .
            .
```

Using enumsnmp.htm

To launch *enumsnmp.htm*:

Launch Internet Explorer and open the file.

Note: If you already have an Internet Explorer window opened, you can simply drag and drop *enumsnmp.htm* into the Internet Explorer window.

The following illustration shows the results output by *enumsnmp.htm* when opened in the Internet Explorer:

	supervisorIndex	keywordName	kwValue	kwType	kwMode	kwAllowedRang	kwDescription
1	1	ExtendedManagementComponents[0]	clkmgr.emc	Object	readOnly	<no range>	<none>
2	2	ExtendedManagementComponents[1]	HotSwap.emc	Object	readOnly	<no range>	<none>
3	3	Products[0]	AG_2000	String	readOnly	<no range>	<none>
4	4	Products[1]	AG_4000_1E1	String	readOnly	<no range>	<none>
5	5	Products[2]	AG_4000_1T1	String	readOnly	<no range>	<none>
6	6	Products[3]	AG_4000_2E1	String	readOnly	<no range>	<none>
7	7	Products[4]	AG_4000_2T1	String	readOnly	<no range>	<none>
8	8	Products[5]	AG_4000_4E1	String	readOnly	<no range>	<none>
9	9	Products[6]	AG_4000_4T1	String	readOnly	<no range>	<none>
10	10	Products[7]	AG_4000C_2E1	String	readOnly	<no range>	<none>
11	11	Products[8]	AG_4000C_2T1	String	readOnly	<no range>	<none>
12	12	Products[9]	AG_4000C_4E1	String	readOnly	<no range>	<none>
13	13	Products[10]	AG_4000C_4T1	String	readOnly	<no range>	<none>
14	14	Products[11]	AG_CPCI_Quad_E1	String	readOnly	<no range>	<none>
15	15	Products[12]	AG_CPCI_Quad_T1	String	readOnly	<no range>	<none>
16	16	Products[13]	AG_Dual_E1	String	readOnly	<no range>	<none>
17	17	Products[14]	AG_Dual_T1	String	readOnly	<no range>	<none>
18	18	Products[15]	AG_Quad_Connect_E1	String	readOnly	<no range>	<none>
19	19	Products[16]	AG_Quad_Connect_T1	String	readOnly	<no range>	<none>
20	20	Products[17]	AG_Quad_E1	String	readOnly	<no range>	<none>
21	21	Products[18]	AG_Quad_T1	String	readOnly	<no range>	<none>
22	22	Products[19]	CG_6000C_Quad	String	readOnly	<no range>	<none>
23	23	Products[20]	CX 2000-16	String	readOnly	<no range>	<none>
24	24	Products[21]	CX 2000-32	String	readOnly	<no range>	<none>
25	25	Products[22]	CX 2000C-16	String	readOnly	<no range>	<none>
26	26	Products[23]	CX 2000C-32	String	readOnly	<no range>	<none>
27	27	Products[24]	CX 2000C-48	String	readOnly	<no range>	<none>
28	28	Products[25]	QX 2000/100-4L	String	readOnly	<no range>	<none>
29	29	Products[26]	QX 2000/200-4L	String	readOnly	<no range>	<none>
30	30	Products[27]	QX 2000/80-1L	String	readOnly	<no range>	<none>
31	31	Products[28]	QX 2000/80-4L	String	readOnly	<no range>	<none>
32	32	BoardPlugins[0]	agplugin.bpi	Object	readOnly	<no range>	<none>
33	33	BoardPlugins[1]	cg6kpi.bpi	Object	readOnly	<no range>	<none>
34	34	BoardPlugins[2]	cx.bpi	Object	readOnly	<no range>	<none>
35	35	BoardPlugins[3]	qx2kpi.bpi	Object	readOnly	<no range>	<none>
36	36	Boards[0]	Name0	Object	readOnly	<no range>	<none>
37	37	Boards[1]	Name1	Object	readOnly	<no range>	<none>
38	38	Name	Supervisor	Object	readOnly	<no range>	<none>

enumsnmp.htm output

Index

P

R

S

T

U

V

W