Dialogic
Making Innovation Thrive™

Dialogic® NaturalAccess™ SNMP API Developer's Manual

October 2010 64-0494-03 Rev A

www.dialogic.com

Copyright and Legal Notice

Copyright © 2005-2010 Dialogic Corporation. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Corporation at the address provided below.

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation or its subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.

Dialogic, Dialogic Pro, Brooktrout, Diva, Diva ISDN, Making Innovation Thrive, Video is the New Voice, Diastar, Cantata, TruFax, SwitchKit, SnowShore, Eicon, Eicon Networks, NMS Communications, NMS (stylized), Eiconcard, SIPcontrol, TrustedVideo, Exnet, EXS, Connecting to Growth, Fusion, Vision, PacketMedia, PowerMedia, NaturalAccess, NaturalCallControl, NaturalConference, NaturalFax and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation or its subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic.

Such permission may only be granted by Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Revision history

Revision	Release date	Notes
9000-6744-10	September, 1999	EPS for CT Access 3.0 Beta
9000-6744-11	January, 2000	EPS for CT Access 3.0
9000-6744-12	July, 2000	EPS / SJC, Platform support for Fusion 4.0
9000-6744-13	September, 2000	SJC for CT Access 4.0
9000-6744-14	January, 2001	CYF
9000-6744-15	August, 2001	CYF, Natural Access 2001-1
9000-6744-16	May, 2002	MVH, Natural Access 2002-1
9000-6744-17	April, 2003	MVH, Natural Access 2003-1
9000-6744-18	April, 2004	MCM, Natural Access 2004-1
9000-6744-19	March, 2005	LBG, Natural Access 2005-1
9000-6744-20	March, 2009	LBG, Natural Access 8.1
64-0494-01	October, 2009	LBG, NaturalAccess R9.0
64-0494-02	December, 2009	LBG, NaturalAccess R9.0.1
64-0494-03 Rev A	October, 2010	LBG, NaturalAccess R9.0.4
This manual printed: 2010-10-25		

Refer to www.dialogic.com for product updates and for information about Dialogic support policies, warranty information, and service offerings.

Table Of Contents

1. Introduction

The *Dialogic® NaturalAccess™ SNMP API Developer's Manual* explains how to install and configure simple network management protocol (SNMP) for Dialogic® NaturalAccess™ products. This manual is intended for customers who want to add SNMP monitoring to NaturalAccess boards. It provides an overview of SNMP and describes the management information bases (MIBs) and agents used to support SNMP on AG and CG boards.

Terminology

Note: The product to which this document pertains is part of the NMS Communications Platforms business that was sold by NMS Communications Corporation ("NMS") to Dialogic Corporation ("Dialogic") on December 8, 2008. Accordingly, certain terminology relating to the product has been changed. Below is a table indicating both terminology that was formerly associated with the product, as well as the new terminology by which the product is now known. This document is being published during a transition period; therefore, it may be that some of the former terminology will appear within the document, in which case the former terminology should be equated to the new terminology, and vice versa.

Former terminology	Dialogic terminology
CG 6060 Board	Dialogic® CG 6060 PCI Media Board
CG 6060C Board	Dialogic® CG 6060C CompactPCI Media Board
CG 6565 Board	Dialogic® CG 6565 PCI Media Board
CG 6565C Board	Dialogic® CG 6565C CompactPCI Media Board
CG 6565e Board	Dialogic® CG 6565E PCI Express Media Board
CX 2000 Board	Dialogic® CX 2000 PCI Station Interface Board
CX 2000C Board	Dialogic® CX 2000C CompactPCI Station Interface Board
AG 2000 Board	Dialogic® AG 2000 PCI Media Board
AG 2000C Board	Dialogic® AG 2000C CompactPCI Media Board
AG 2000-BRI Board	Dialogic® AG 2000-BRI Media Board
NMS OAM Service	Dialogic® NaturalAccess™ OAM API
NMS OAM System	Dialogic® NaturalAccess™ OAM System
NMS SNMP	Dialogic® NaturalAccess™ SNMP API
Natural Access	Dialogic® NaturalAccess™ Software
Natural Access Service	Dialogic® NaturalAccess™ Service
Fusion	Dialogic® NaturalAccess™ Fusion™ VoIP API
ADI Service	Dialogic® NaturalAccess™ Alliance Device Interface API

Former terminology	Dialogic terminology
CDI Service	Dialogic® NaturalAccess™ CX Device Interface API
Digital Trunk Monitor Service	Dialogic® NaturalAccess™ Digital Trunk Monitoring API
MSPP Service	Dialogic® NaturalAccess™ Media Stream Protocol Processing API
Natural Call Control Service	Dialogic® NaturalAccess™ NaturalCallControl™ API
NMS GR303 and V5 Libraries	Dialogic® NaturalAccess™ GR303 and V5 Libraries
Point-to-Point Switching Service	Dialogic® NaturalAccess™ Point-to-Point Switching API
Switching Service	Dialogic® NaturalAccess™ Switching Interface API
Voice Message Service	Dialogic® NaturalAccess™ Voice Control Element API
NMS CAS for Natural Call Control	Dialogic® NaturalAccess™ CAS API
NMS ISDN	Dialogic® NaturalAccess™ ISDN API
NMS ISDN for Natural Call Control	Dialogic® NaturalAccess™ ISDN API
NMS ISDN Messaging API	Dialogic® NaturalAccess™ ISDN Messaging API
NMS ISDN Supplementary Services	Dialogic® NaturalAccess™ ISDN API Supplementary Services
NMS ISDN Management API	Dialogic® NaturalAccess™ ISDN Management API
NaturalConference Service	Dialogic® NaturalAccess™ NaturalConference™ API
NaturalFax	Dialogic® NaturalAccess™ NaturalFax™ API
SAI Service	Dialogic® NaturalAccess™ Universal Speech Access API
NMS SIP for Natural Call Control	Dialogic® NaturalAccess™ SIP API
NMS RJ-45 interface	Dialogic® MD1 RJ-45 interface
NMS RJ-21 interface	Dialogic® MD1 RJ-21 interface

Former terminology	Dialogic terminology
NMS Mini RJ-21 interface	Dialogic® MD1 Mini RJ-21 interface
NMS Mini RJ-21 to NMS RJ-21 cable	Dialogic® MD1 Mini RJ-21 to MD1 RJ-21 cable
NMS RJ-45 to two 75 ohm BNC splitter cable	Dialogic® MD1 RJ-45 to two 75 ohm BNC splitter cable
NMS signal entry panel	Dialogic® Signal Entry Panel
Video Access Utilities	Dialogic® NaturalAccess™ Video Access Toolkit Utilities
Video Mail Application Demonstration Program	Dialogic® NaturalAccess™ Video Access Toolkit Video Mail Application Demonstration Program
Video Messaging Server Interface	Dialogic® NaturalAccess™ Video Access Toolkit Video Messaging Server Interface
3G-324M Interface	Dialogic® NaturalAccess™ Video Access Toolkit 3G-324M Interface

2. Overview of NaturalAccess SNMP

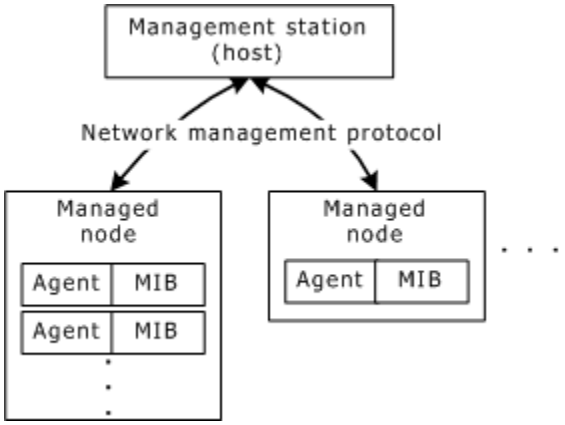
Using network management

Network management allows administrators to maintain network integrity. Simple Network Management Protocol (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, running network management software and displaying network information. The management station is called the host.
- A network management protocol that determines how the managed node and the management station communicate over the network.

The following illustration shows the relationship between the SNMP components:



In this 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 represent how to traverse the MIB tree to access a managed object. All MIBs have a common root node. All OID integer sequences start from that root. The OIDs are assigned by the IETF.

The tree of MIBs is referred to as a namespace. Each MIB and OID is unique. The namespace for the tree is maintained by the IETF and related organizations, who delegate authority only for MIBs below the Enterprise's MIB, whose OID is 1.3.6.1.4.1.

Managed network components

A typical managed network consists of the following components:

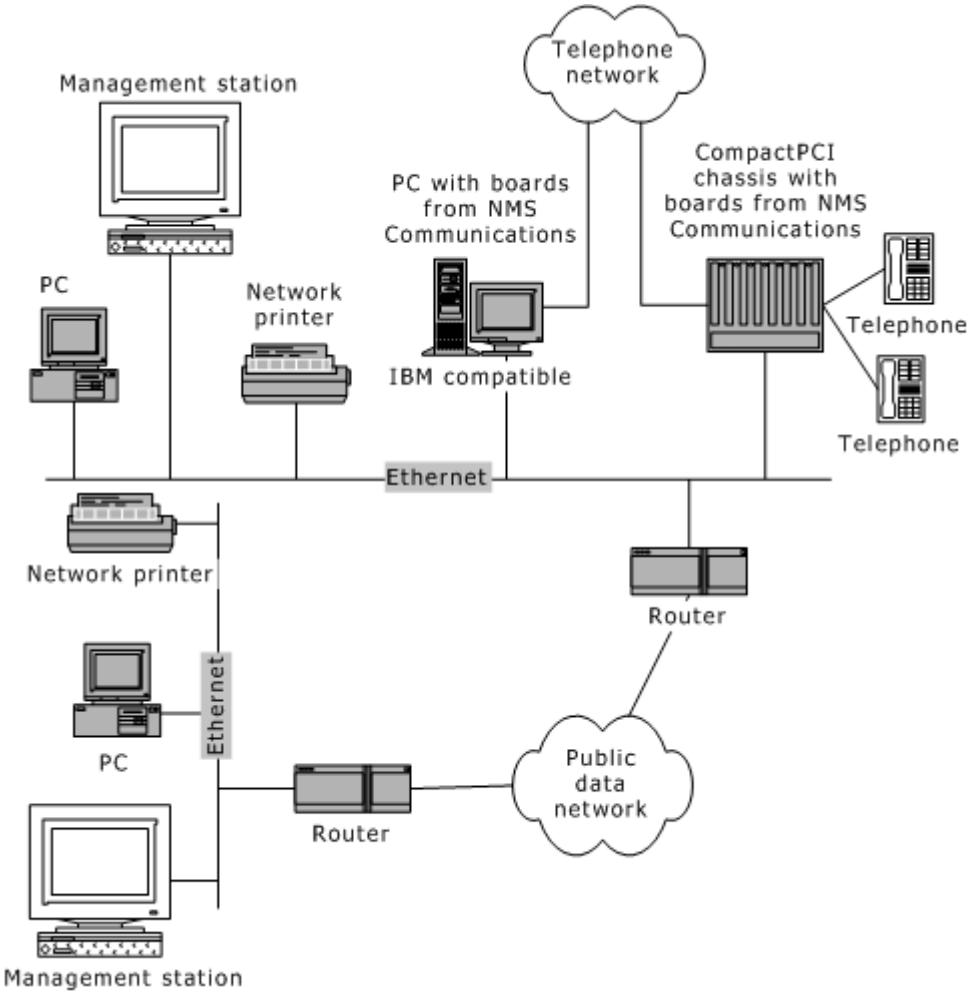
- Managed nodes
- Management information bases
- Management stations
- Management protocol

Managed nodes

An SNMP managed node can be any device that connects to a data network and can execute the SNMP protocol. A managed node can be a:

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

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



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 that 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.

For information about how to activate the SNMP service and load an agent, refer to the Installation summary.

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 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 can be a variable or a structure of variables. Each managed object is assigned 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.

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 the information 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 a readable format.

Host applications are 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. An example of a host management station is HP Openview.

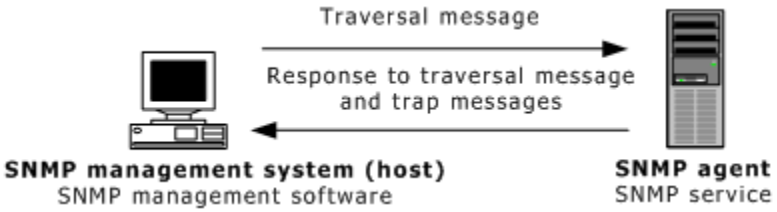
Management protocol

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.

SNMP supports the following message types:

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

The following illustration shows the SNMP host and SNMP agent message flow:



The following host commands generate traversal messages:

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.

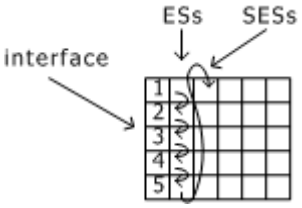
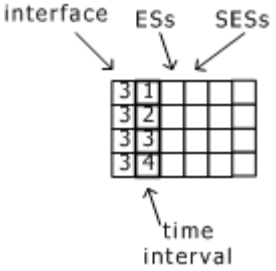
An agent sends trap messages to notify the host about an unusual occurrence. The host can then request the value of related variables to determine more about the managed node's condition. The agent can be set to send a trap when certain conditions arise, such as an error on a line. Care must be taken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

Connectionless transport does not guarantee delivery. Traps and other network messages are not guaranteed to arrive at the host. Plan your network management policies to consider lost messages.

Accessing MIB objects

The following table describes the ways you can access objects in a MIB:

Access method	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.

Access method	Description
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 the get-next command finds the next object in the current MIB that has a value. The get-next command 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. 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, in which each row is the index of the interface and each column represents a statistic. If you use the get command to retrieve errored seconds (ESs) for interface 1, then each time you use the get-next command this retrieves ESs for the next interface. When get-next has retrieved ESs for interface 5, the next get-next command retrieves severely errored seconds (SESs) for interface 1, as shown in the following illustration:  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 using get for the ES for the first time interval of the third interface. Using 0b get-next retrieves ES for the next time interval, as shown in the following illustration:  When get-next has retrieved ES for all intervals of interface 3, the next get-next command will either retrieve the ES for the first interval of the next interface (if there is one), or the SES for the first interval of interface 3. For more information, refer to Current table and Interval table.

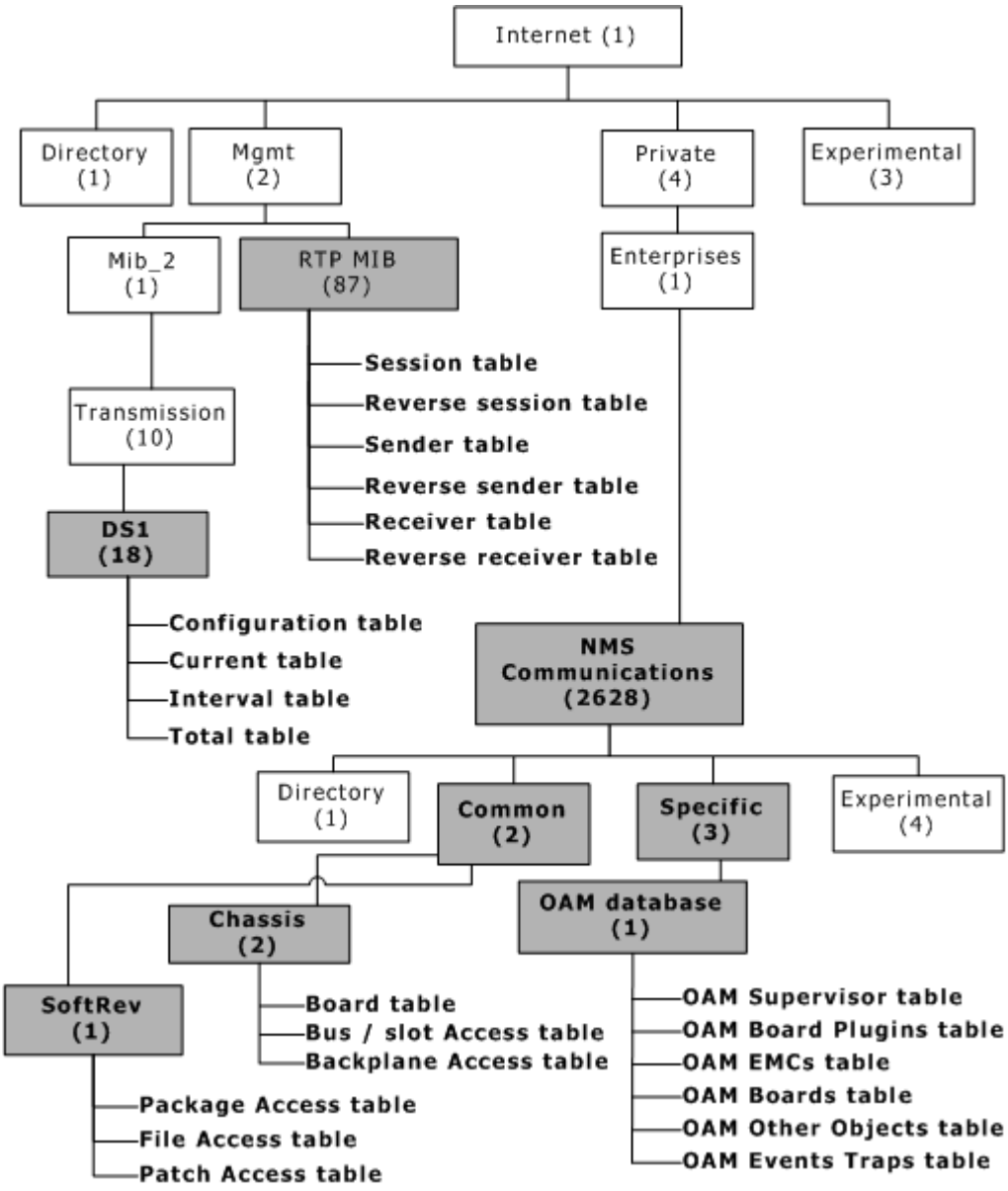
Supported MIBs

The following table describes the supported MIBs:4

MIB	Description	Ownership	Installed by
Trunk	Represents DS1 (and higher speed) lines and is defined by the IETF. Also called the DS1 MIB.	RFC-2945 (obsolete RFC-1406)	OAM API
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 API
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 API
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 using this MIB.	Proprietary	OAM API
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 API

NMS is assigned a namespace under the Enterprises MIB. The OID for the NMS MIB is 1.3.6.1.4.1.2628. The Chassis MIB, the Software Revision MIB, the OAM Database MIB, and future private MIBs reside under this OID.

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



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 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)

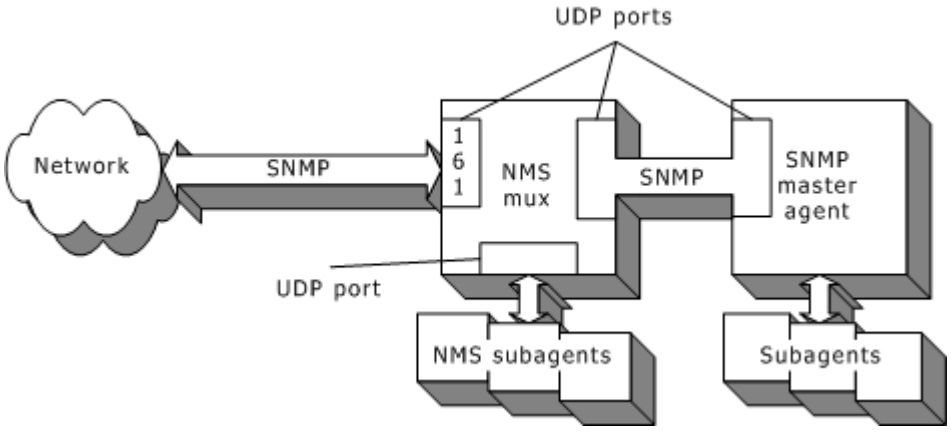
These text files require other MIB description files documented in RFCs (such as SNMPv2-TC, SNMPv2-CONF, SYSAPPL-MIB). These files can be easily found on the web. The NMS Communications hierarchy shown in the previous illustration is defined in the NMS-SMI MIB.

NaturalAccess SNMP API architecture

NaturalAccess SNMP API architecture 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).



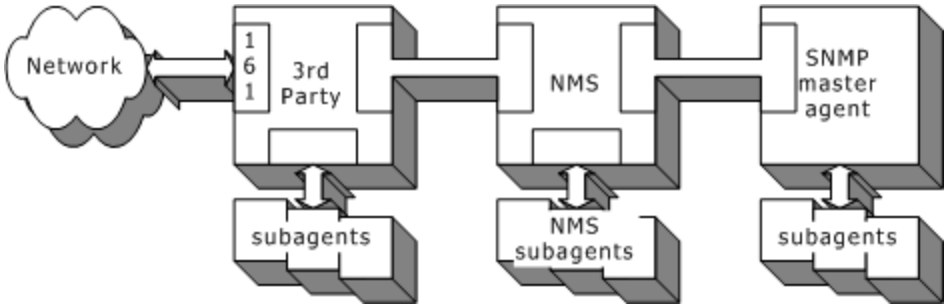
The NMS multiplexer handles all requests coming from the UDP network port and communicates with the NMS subagents. SNMP requests not addressed to the NMS multiplexer are routed to the native master agent. Each NMS subagent runs in a different process and exchanges information with the multiplexer using a UDP socket connection. The NMS multiplexer is 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 display the currently used IP/UDP ports. It can also start and stop the agents without having to kill the process. It can display all currently registered subagents.

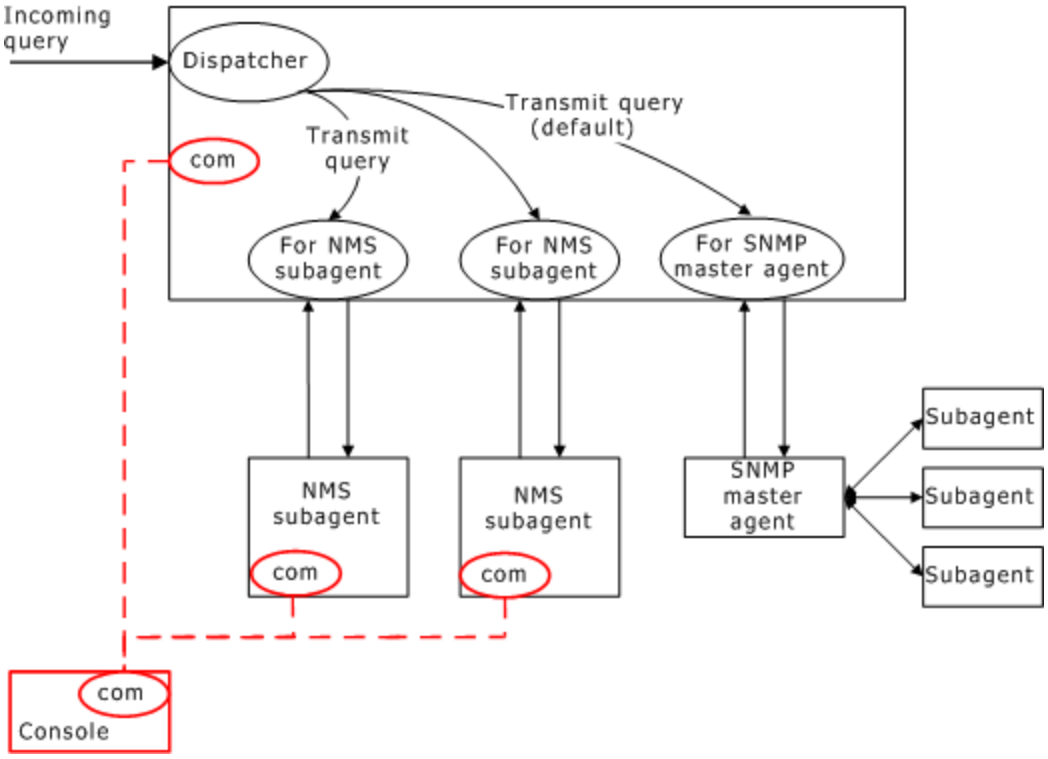
As shown in the following illustration, configurable IP/UDP ports allow the NMS multiplexer to be inserted in a chain of multiplexers. In this configuration, each multiplexer processes incoming SNMP requests. Requests not addressed to a given multiplexer are passed to the next one.



The main benefits of the multiplexer are:

- 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:



3. Installing and configuring SNMP

Installation summary

NMS SNMP software components fully support SNMP version 1 but do not fully support SNMP version 2. For example, the get-bulk command is not fully supported and SNMP traps are generated in version 1 format. NMS recommends that you use SNMP request version 1 when accessing NMS subagents.

Supported operating systems

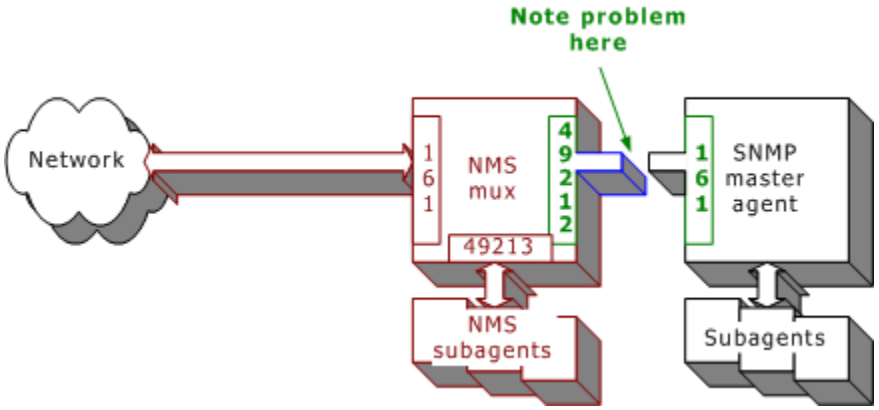
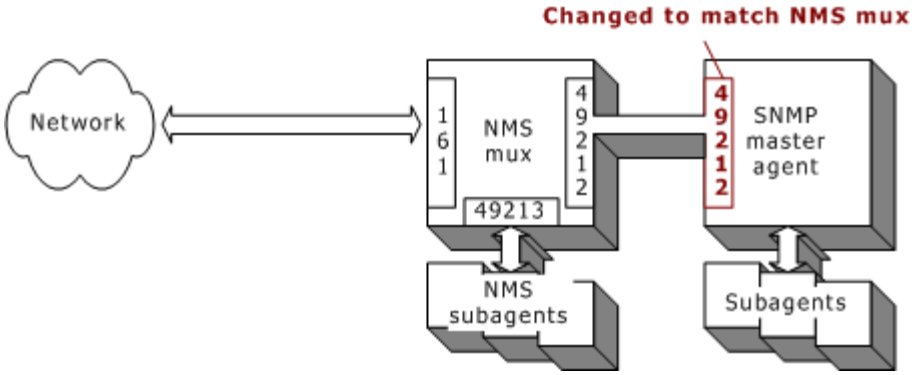
NMS SNMP software is available for the following operating systems:

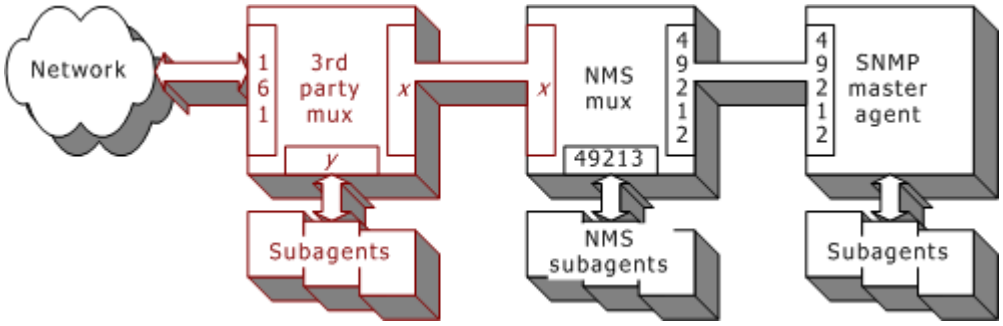
- Windows
- UNIX
- Red Hat Linux

Installation and configuration overview

The following table provides an overview of the steps required to install and configure NMS SNMP:

Step	Action
1	Install the SNMP master agent under UNIX or under Windows. By default, the master agent communicates with the network using UDP port 161. The following illustration shows installing the SNMP master agent: 

Step	Action
2	Install the NMS subagents and multiplexer (mux) as shown in the following illustration: 
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. The following illustration shows configuring the SNMP master agent UDP port: 
4	Start up the SNMP master agent, NMS multiplexer, and subagents.

Step	Action
5	To include a third-party multiplexer, set it up between the network and the NMS multiplexer, and configure the ports accordingly. The following illustration shows configuring a third-part 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 provides procedures for installing under Solaris and for installing under Linux.

Installing under Solaris

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

Complete the following steps to install the SNMP master agent on a Solaris system:

Step	Action
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 command: `/etc/init.d/init.snmpdx start`
5	Verify that the SNMP master agent is properly installed. To do so, 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.

Installing under Linux

To ensure that SNMP management services work properly, install the following net-snmp packages:

- net-snmp-5.3.1-19.el5.***<arch>***.rpm
- net-snmp-libs-5.3.1-19.el5.***<arch>***.rpm (installed by default)

where <***arch***> is a system architecture. For example, i386 for a 32-bit x86 system generic build, X86_64 for 64-bit processors, and so on.

These packages are provided with Red Hat Enterprise Linux, version 5.1 and are located in the /Server directory. To access the most recent net-snmp packages, refer to www.net-snmp.org.

Complete the following steps to install the SNMP master agent on a Linux system:

Step	Action
1	Install the SNMP package. To do so, access a command prompt and enter: `rpm –i net-snmp-5.3.1-19.el5.<arch>.rpm`
2	Start the SNMP master agent by entering the following command: `/etc/rc.d/init.d/snmpd start`
3	Verify that the SNMP master agent is properly installed. To do so, 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.

Installing the master agent under Windows

Complete the following steps to install the SNMP master agent under Windows:

Step	Action
1	Click **Start > Control Panel > Programs**.
2	Click the Programs icon in the Control Panel menu. The Programs dialog box displays.
3	Click **Turn Windows features on or off** in the **Programs and Features** section. The Server Manager dialog box displays the server information.
4	Click **Features** on the left pane. The Features dialog box displays on the right pane, listing the features installed on the server.
5	Click **Add Features** on the right. The Add Features Wizard dialog box displays. In the **Features** scroll down list, select **SNMP Services** > **Next**.

Step	Action
6	View the features selected in the **Confirm Installation Selections** dialog box and click **Install**. After the installation succeeded message displays, click **Close** and reboot the server.

7	(Optional) Configure the SNMP master agents by completing the following steps:
	a. In the **Control Panel** (Classic View) **> Administrative Tools** menu, click **Services**. The Services dialog box displays.
	b. Right-click on **SNMP Service** and select **Properties**. The SNMP Properties dialog displays.
	c. Select the **Traps** tab.
	d. Add a Community Name. For example: public
	e. Add the addresses of the hosts that you want 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 **Apply**, then click **OK**.
	h. Open a command prompt window.
	i. Enter the following command to stop or start the SNMP service:
	`net stop snmp`
	`net start snmp`

8	Enter the following command to stop or start the SNMP trap service:
	`net stop snmptrap`
	`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 command to verify that SNMP service is started:
	`netstat -a`
	The following text displays:
	`UDP snmplab_3:snmp-trap *:*`
	`UDP snmplab_3:snmp *:*`

10	Verify that the SNMP master agent is properly installed. To do so, 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. **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, the public community has only READ_ONLY access.

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

Installing the NMS subagents and multiplexer

Once the SNMP master agent is working properly, install Natural Access, which installs the NMS subagents and the NMS SNMP multiplexer. For information about installing Natural Access, refer to the Natural Access installation booklet.

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

Modifying the Windows registry

Under Windows, 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).

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.

This topic describes how to change the SNMP master agent's UDP port under:

- Windows
- UNIX

Windows

Complete the following steps to change the SNMP master agent's UDP port:

Step	Action
1	Open the file *Services* for editing. This file can be found in *\Windows\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`

UNIX

Solaris

Complete the following steps to change the SNMP master agent's UDP port:

Step	Action
1	Log on 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 produces 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 example 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

Complete the following steps to change the SNMP master agent's UDP port:

Step	Action
1	Log in as superuser.
2	Open the file */etc/snmp/snmpd.conf* for editing.
3	In this file, insert or modify the line beginning with agentaddress so it matches the following example: `agentaddress udp:49212`
4	Save and close this file.
5	Open a console window.

6	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 produces output similar to:
	`136 ? 0:00 snmpd`
	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`

7	Another way to stop the master agent process is by invoking: `/etc/rc.d/init.d/snmpd stop`
8	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, to receive or send SNMP requests, and to 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)

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

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

Configuration file syntax

Statements within the file display one to a line. Text appearing after a number sign (#) is a comment and is ignored. Statements are not case sensitive, except where operating system conventions prevail (for example, file names under UNIX).

snmp.cfg is divided into sections. Each section has a header in square brackets ([ ]). The statements in each section apply to one or more subagents. The following table provides a description of the *snmp.cfg* sections:

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 in a section 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:

Keyword	Description	Mandatory?
SnmpPort	Defines the port through which SNMP queries are sent to the multiplexer. Allowed values: Valid UDP port number	Yes
MasterAgentPort	Defines the port through which the multiplexer sends SNMP requests not addressed to its subagents. Allowed values: Valid UDP port number	Yes
CommunicationPort	Defines the port used by the multiplexer, the subagent, and the console to communicate (for registration, stop, or information commands). Allowed values: Valid UDP port number	Yes
access	Defines the access rights and the defined communities that can be used to send requests to the agents. Allowed values are: ***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

Keyword	Description	Mandatory?
trap	Defines the host where the trap is sent and the community that is used. Allowed values are: ***host,community*** where: • ***host*** specifies the name of the host where the SNMP requests are authorized. You must specify either an IP address or IP address mask for the host in the trap. • ***community*** is the name of a defined community that can be used to send requests to the agents.	No

The subagent-specific sections

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

The access and trap keywords (defined 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 are 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
```

4. Activating SNMP

Starting the NMS multiplexer and subagents

Before starting the NMS multiplexer, make sure that the network IP address is assigned to the host. Use one of the following methods to start the NMS multiplexer and the subagents:

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

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

Starting SNMP using muxC

Complete the following steps to start the NMS multiplexer and subagents using *muxC*:

Step	Action
1	Access a command prompt.

2	Enter the following command to start *muxC*:
	`muxC`
	The following text displays:
	`**********`
	`*  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.

Use the *muxC* command line option -d to create a terminal window each time you start the multiplexer, the SNMP subagents, or both. The components are started in debug mode with the following command:

```
muxC -d
```

Starting SNMP using the command line

Complete the following steps to start the components using the command line:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Access a command prompt.</td></tr>
<tr><td>2</td><td>Enter the following command for each component:
<table>
<tr><th>For this operating system...</th><th>Enter...</th></tr>
<tr><td>Windows</td><td>net start component_name</td></tr>
<tr><td>UNIX</td><td>component_name</td></tr>
</table>
component_name is the name of the component to start. component_name can be any of the following:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>mux</td><td>NMS multiplexer</td></tr>
<tr><td>chassisAgent</td><td>Chassis MIB agent</td></tr>
<tr><td>ds1Agent</td><td>Trunk MIB agent</td></tr>
<tr><td>oamAgent</td><td>OAM Database MIB agent</td></tr>
<tr><td>softRevAgent</td><td>Software Revision MIB agent</td></tr>
<tr><td>rtpAgent</td><td>RTP MIB agent (installed with NMS Fusion)</td></tr>
</table>
</td></tr>
</table>

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

To obtain error information, 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 topic describes how to change the IP/UDP ports used by the NMS multiplexer after the master agent, the NMS multiplexer, and the subagents are running.

By default, the NMS multiplexer uses the following IP/UDP ports:

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 port values are stored in the *snmp.cfg*. To change the values, edit *snmp.cfg* by completing the following steps:

Step	Action
1	Locate *snmp.cfg* in one of the following directories: **Operating system** / **Directory** Windows / *\nms\ctaccess\cfg* UNIX / */opt/nms/ctaccess/cfg/*
2	Modify the settings in the file.
3	Save and close the file.
4	To make the changes effective, restart the master agent, the NMS multiplexer, and the subagents.

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

Running NMS SNMP

Ensure that the following conditions apply to run SNMP after it is installed:

Condition	Description
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, the DTM service is specified in *cta.cfg*. Refer to the *Natural Access Developer's Reference Manual* for more information about *cta.cfg*. Refer to the *Digital Trunk Monitor Service Developer's Reference Manual* for information about DTM.

The Natural Access Server (*ctdaemon*) must be running.	*ctdaemon* activates the NMS OAM database. *ctdaemon* must be running for the NMS SNMP agents to recognize boards and report data to SNMP requests.
	To verify that *ctdaemon* is running access a command prompt and 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 are 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 NMS OAM.

5. Chassis MIB overview

Chassis MIB representation

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

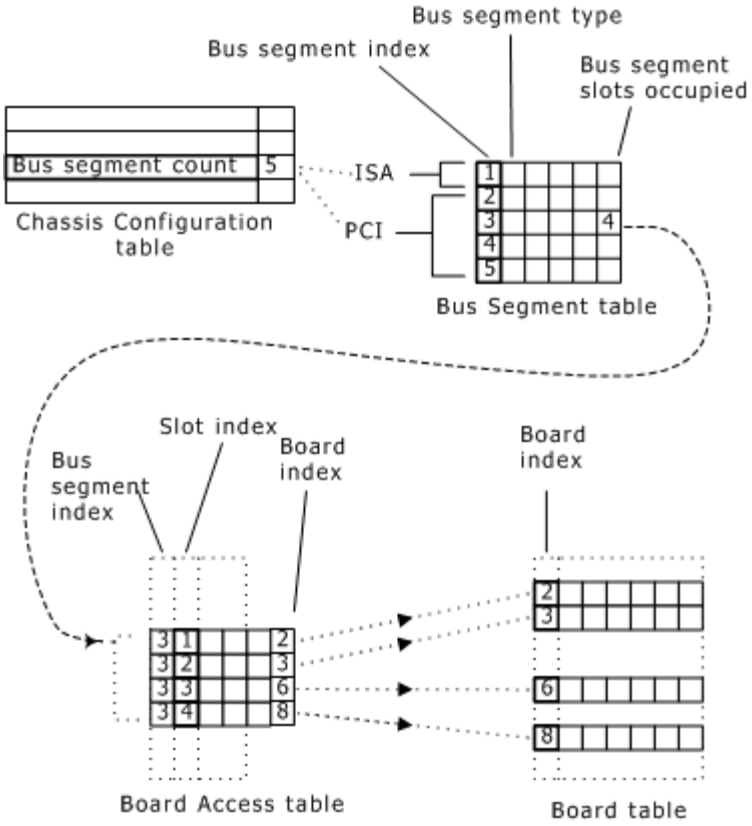
The Chassis agent detects each NMS board that is registered to NMS OAM and booted correctly, and monitors its operational status. The board model, type, revision, bus segment and slot, and logical ID are represented. Removing or inserting a board (Hot Swap) is also monitored, and traps are sent if the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

Chassis MIB structure

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	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.)

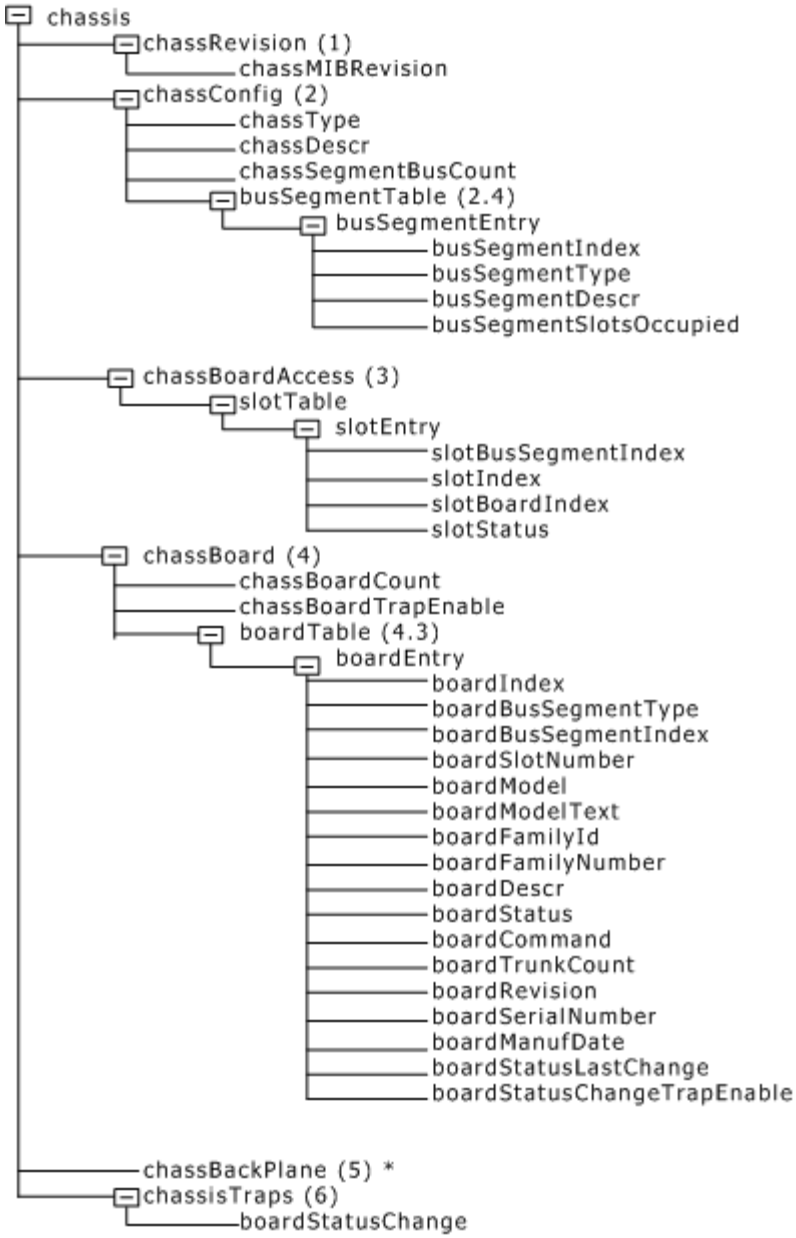
The following illustration shows the relationship between the tables in the Chassis MIB:



This illustration shows that a bus segment count 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 following illustration shows the sequence of objects in the Chassis MIB (with relative OIDs for table objects):



*Not supported in this revision.

Using the Chassis MIB

To enable or disable traps, set the **boardStatusChangeTrapEnable** object in the Board table. You must also configure traps. See Configuring NMS SNMP for more information about configuring traps.

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

Hot Swap

Extracting a board removes the entry for that board from the Board table. If all the boards in a bus segment are extracted, that bus segment entry is removed from the Bus Segment table. If the removed entry creates a non-contiguous numerical sequence, that number is 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 and 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*.

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_4040_1TE_02_01
boardModelText = AG_4040_1TE
boardFamilyNumber = 2
```

In this example, **dsx1CircuitIdentifier** says that the trunk is on an AG_4040_1TE board, the family number is 2, and the trunk number is 1. The trunk number has no direct match in the Chassis MIB.

Chassis 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. The NMS Chassis agent assigns values to these objects.

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.

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 be many PCI (or CompactPCI) bus segments.

A **busSegmentIndex** object, whose value is assigned by the NMS Chassis agent, identifies each **busSegmentEntry** object.

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 is deleted from the table.

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	Specifies the number of occupied slots in this bus segment.

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. 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 needs only to find the first entry with the bus segment number that matches PCI segment 2, and the rest of that segment's boards are 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 slot bus segment number 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 board index value to index into the Board table.

The objects in the Board Access 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	Specifies the number of the bus segment this board is in.
slotIndex	Specifies the logical slot index of a board in the bus segment.
slotBoardIndex	Specifies the index into the Board table for this bus segment.
slotStatus	Specifies the slot's Hot Swap status.

Chassis 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.

The **boardIndex** object, whose value is assigned by the NMS Chassis agent, identifies each boardEntry. New **boardEntry** objects are added to this table and configured for the OAM API when a board is added to the chassis.

If a board is physically extracted, entries in the Board table are removed. If a board is inserted, a new entry is added to the Board table using the next free index. Whenever a board is inserted or extracted, a trap is sent if traps are enabled.

The objects in the Board table are:

Object	Description
chassBoard	Beginning of the board descriptions.
chassBoardCount	Number of boards in the chassis.
chassBoardTrapEnable	Default value for the **boardStatusChangeTrapEnable** object for the entries in this table.
boardTable	Beginning of the Board table.
boardEntry	Beginning of 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).

Object	Description
boardCommand	Command for turning 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	Date and time the status of the board last changed.
boardStatusChangeTrapEnable	Switch for determining if **boardStatusLastChange** traps are generated.

6. Chassis MIB object reference

Using the Chassis MIB object reference

This section describes the objects in the Chassis MIB. A typical object description includes:

Syntax	Datatype of the object. 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	Type of access allowed for this object. Options are: • Read-only: The object cannot be modified by SNMP. • Read/write: SNMP can configure this object.
OID	Path from the root to this object. All OIDs start with ***p***, where ***p*** is 1.3.6.1.4.1.2628.2.2 (the OID for the CHASSIS-MIB).
Details	Object description.
Configuration	How to configure the object, if configurable.
Example	Example of how the object is used.

NMS SNMP MIBs are compiled using the following text files, located in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX):

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

To display the SNMP information for the proprietary agent, read the appropriate file using the Windows Console Management function.

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.

boardBusSegmentType

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:

Value	Type
2	PCI bus

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. Check the **boardStatus** object to see when the command completes. Valid values:

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 applies only to CompactPCI boards.

boardDescr

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

To configure this object, edit *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.

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 still exist, but the Trunk MIB does not see any lines. When the board is extracted, the board index is also removed.

A board that is inserted uses the next available index number.

boardFamilyId

Board family.

Syntax

Integer

Access

Read-only

OID

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

Details

The following table shows expected values:

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

boardFamilyNumber

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.

boardManufDate

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.

Example

```
week 5 00
```

boardModel

Board type.

Syntax

Integer

Access

Read-only

OID

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

Details

Supported board types include:

- Other
- AG_2000, AG_2000_BRI, AG_2000C
- CG_6060, CG_6060_4,.CG_6060_8
- CG_6060C, CG_6060C_4, CG_6060C_8, CG_6060C_16
- CG_6565, CG_6565_4, CG_6565_8
- CG_6565C, CG_6565C_8, CG_6565C_16
- CG_6565E, CG 6565E_4, CG_6565E_8
- CX 2000, CX 2000-16, CX 2000-32
- CX 2000C-16, CX 2000C-32, CX 2000C-48

boardModelText

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_2000C
CG	CG_6060, CG_6060_4,.CG_6060_8 CG_6060C, CG_6060C_4, CG_6060C_8, CG_6060C_16 CG_6565, CG_6565_4, CG_6565_8 CG_6565C, CG_6565C_8, CG_6565C_16 CG 6565E, CG _6565E_4, CG _6565E_8
CX	CX 2000, CX 2000-16, CX 2000-32 CX 2000C-16, CX 2000C-32, CX 2000C-48

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

boardRevision

Board revision.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

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

boardSerialNumber

Board serial number.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

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

Example

```
123456754
```

boardSlotNumber

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.

boardStatus

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. This can be time consuming.
6	Extracted.

boardStatusChangeTrapEnable

Whether or not to generate traps 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 causes traps to be sent to the management station and updates the **boardStatusLastChange** object. Valid values are:

Value	Description
1	Enabled
2	Disabled (default)

boardStatusLastChange

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.

boardTable

Starts a sequence of **boardEntry** objects that 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**

m = **busSegmentSlotsOccupied**

boardTrunkCount

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.

busSegmentDescr

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.

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 are removed when a board is extracted from the bus.

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**.

busSegmentSlotsOccupied

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 table for this bus segment.

The agent updates the object when a board is inserted into or extracted from the associated bus segment.

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**.

busSegmentType

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:

Value	Type
2	PCI or CompactPCI bus

chassBoard

Starts the series of variables that constitutes the Board table.

Syntax

Object

Access

Not accessible

OID

p.4

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

chassBoardCount

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 that starts a row in the table.

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

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:

Value	Description
1	Enabled
2	Disabled (default)

chassConfig

Starts a group of three objects that describe the chassis.

Syntax

Object

Access

Not accessible

OID

p.2

chassDescr

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. For example:

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

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

```
ChassisDescription = Description
```

Configuration

To configure this object, edit the *snmp.cfg* before starting the NMS Chassis agent.

chassMIBRevision

Revision ID of the Chassis MIB.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.1.1

Details

This object identifies the MIB so the management station can tell if it is configured for the correct MIB.

chassRevision

Starts a group for the revision field.

Syntax

Object

Access

Not accessible

OID

p.1

chassSegmentBusCount

Number of known bus segment types in the chassis.

Syntax

Integer

Access

Read-only

OID

p.2.3

Details

The default value is 0 (no boards in the chassis). This value determines how many entries there are 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.

chassType

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* file. The keyword and value are:

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

Configuration

To configure this object, edit *snmp.cfg* before starting the NMS Chassis agent.

slotBoardIndex

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.

slotBusSegmentIndex

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.

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 = Object of the group and the column number of this row.

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

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

For more information about doubly indexed tables, see Accessing MIB objects.

slotIndex

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.

slotStatus

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

Valid values:

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 (can be time consuming).
6	Extracted.

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)

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

7. Trunk MIB overview

Trunk MIB representation

This topic describes 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. A numerical index represents each trunk. This number 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.

Trunk MIB structure

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

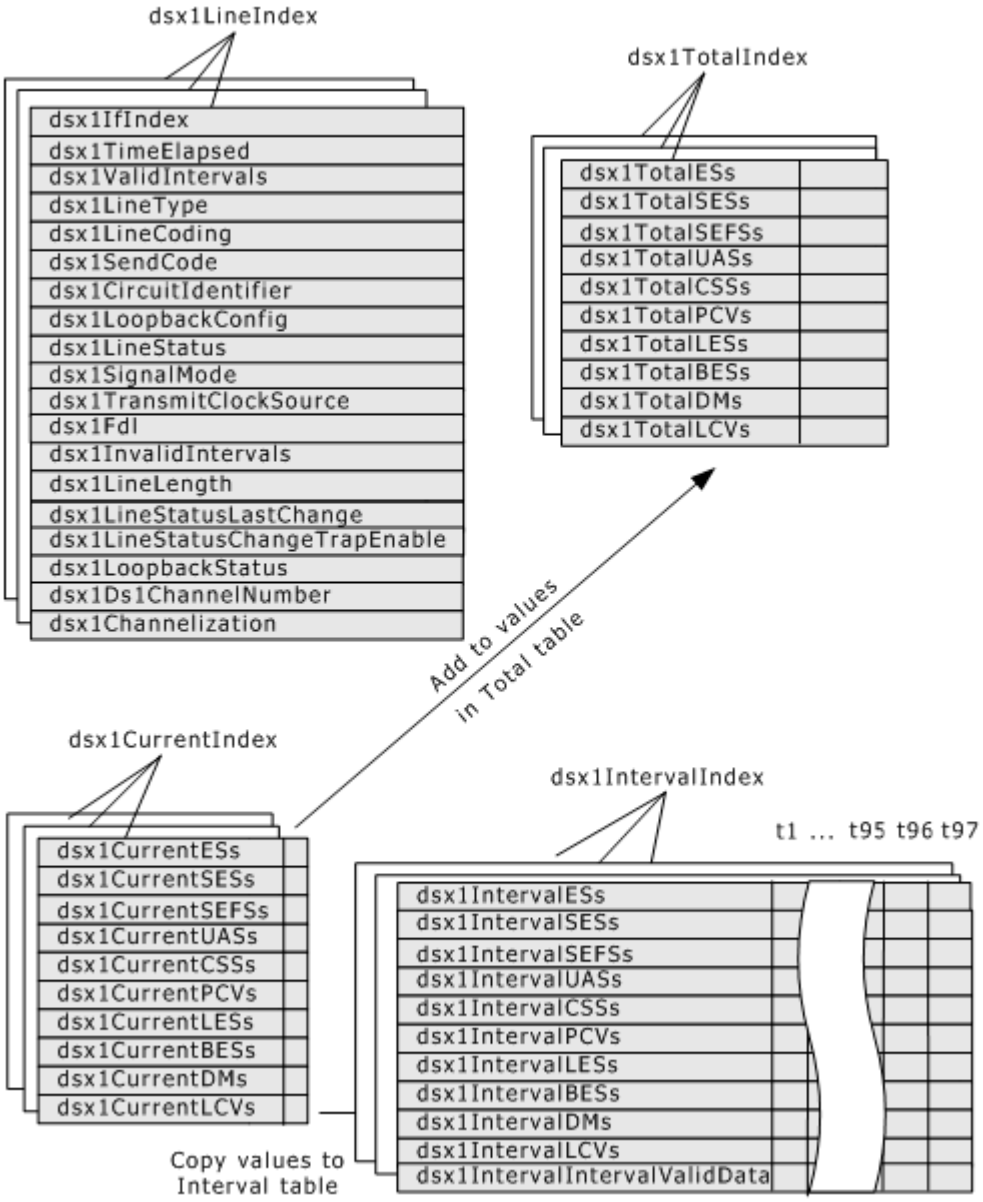
The **dsx1ChannelMappingTable** is also not available.

The DS1 near end group consists of the following tables:

Table	Contains...
Trunk configuration	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	Statistics for the current 15-minute interval.
Interval	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	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 is 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 the relationship between the DS1 tables:

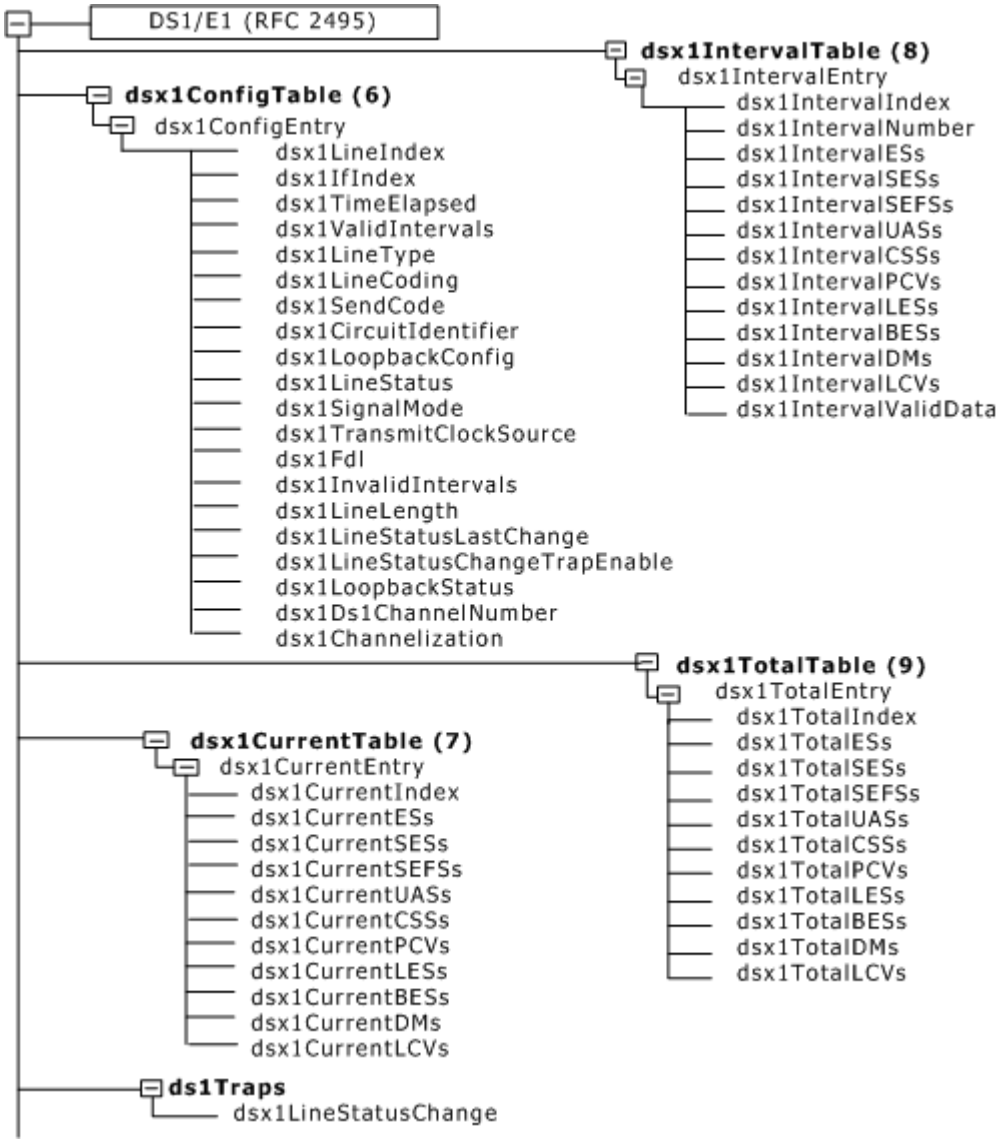


The illustration shows a logical view of the tables for three DS1 interfaces. The Configuration table has an entry for each DS1 interface, which is identified by **dsx1LineIndex**. This index corresponds to the index in the other tables, such that all table entries with the same index number are for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

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 is filled next) in the Interval table. The Current table values are added to the values in the Total table and 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:



Trunk Configuration table

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

Object	Type	Description
dsx1LineIndex	Integer	DS1 interface in this managed node.
dsx1IfIndex	Integer	Same as **dsx1LineIndex**.
dsx1TimeElapsed	Integer	Time of current measurement period.

Object	Type	Description
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	Facilities data link.
dsx1InvalidIntervals	Integer	Number of intervals with invalid data (always 0, not supported).
dsx1LineLength	Integer	Length of the DS1 line in meters (always 0, not supported).
dsx1StatusLastChange	TimeStamp	Time the status of the interface last changed.
dsx1LineStatusChangeTrapEnable	Integer	Whether traps are generated for this interface.
dsx1LoopbackStatus	Integer	Current state of the loopback on the DS1 interface.
dsx1Ds1ChannelNumber	Integer	Channel number of the DS1/E1 on its parent DS2/E2 or DS3/E3.
dsx1Channelization	Integer	Whether this DS1/E1 is channelized or not.

Interval table

Most of the variables in the Interval table have descriptions that match a variable 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.

The **dsx1IntervalNumber** and **dsx1IntervalValidData variables** do not match entries in the Current table. These variables are explained in the following topics.

Current table

The following table summarizes the objects 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.

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**.

For descriptions of the Total table variables, refer to Current table.

8. Trunk MIB object reference

Using the Trunk MIB object reference

This section describes the objects in the Trunk MIB. A typical object description includes:

Syntax	Datatype of the object. 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	Type of access allowed for this object. Options are: • Read-only: The object cannot be modified by SNMP. • Read/write: SNMP can configure this object.
OID	Path from the root to this object. All OIDs start with ***p***, where ***p*** is 1.3.6.1.1.2.1.10.18 (the OID for the DS1-MIB, RFC 2495, and RFC 1902).
Details	Object description.
Configuration	How to configure the object, if configurable.
Example	Example of how the object is used.

NMS SNMP MIBs are compiled using the following text files, located in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX):

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

To display the SNMP information for the proprietary agent, read the appropriate file using the Windows Console Management function.

dsx1Channelization

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

Valid values:

Value	Description
1	disabled
2	enabledDs0
3	enabledDs1

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

dsx1CircuitIdentifier

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_2000, AG_2000_BRI, AG_2000C
CG	CG_6060, CG_6060_4,.CG_6060_8 CG_6060C, CG_6060C_4, CG_6060C_8, CG_6060C_16 CG_6565, CG_6565_4, CG_6565_8 CG_6565C, CG_6565C_8, CG_6565C_16 CG 6565E, CG _6565E_4, CG _6565E_8
CX	CX 2000, CX 2000-16, CX 2000-32 CX 2000C-16, CX 2000C-32, CX 2000C-48

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

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

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

dsx1ConfigTable

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

Syntax

Object

Access

Not accessible

OID

p.6

dsx1ConfigEntry

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

Syntax

Not applicable

Access

Not accessible

OID

p.6.1

dsx1CurrentBESs

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 value is not incremented during an unavailable second (**dsx1CurrentUASs**).

dsx1CurrentCSSs

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 can 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.

dsx1CurrentDMs

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 the following process:

1. Collecting all of the available seconds.
2. Removing any severely errored seconds.
3. Grouping the result in 60-second long groups.
4. 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 those seconds that are not unavailable seconds (**dsx1CurrentUASs**).

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

Each DS1 interface has one entry object.

dsx1CurrentESs

Number 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, one or more out-of-frame defects, 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.

dsx1CurrentIndex

Number of the DS1 interface to which a 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 to Hot Swap:

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

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

dsx1CurrentLCVs

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 code violation (LCV) is the occurrence of either a bipolar violation (BPV) or an 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.

dsx1CurrentLESs

Number of line errored seconds.

Syntax

Gauge

Access

Read-only

OID

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

Details

A line errored second is a second in which one or more line code violation error events were detected. See T1M1.3.

dsx1CurrentPCVs

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).

dsx1CurrentSEFSs

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.

dsx1CurrentSESs

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, 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, 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.

dsx1CurrentTable

Starts the Current table.

Syntax

Object

Access

Not accessible.

OID

p.7

dsx1CurrentUASs

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 the interface is unavailable. The DS1 interface is 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**), the DS1 interface unavailability starts from the onset of these SESs.

Once unavailable, and if...	The DS1 interface becomes available at the...
No failure is present	Onset of 10 contiguous seconds with no SESs.
A failure is present	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 gets of the affected get occur during the first few seconds of the window. This is an unavoidable side effect of selecting the managed objects defined by RFC 2495.

dsx1Ds1ChannelNumber

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.

dsx1Fdl

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.

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.

dsx1IntervalNumber

Number of this **dsx1IntervalEntry** in the Interval table, where each block of variables covers a 15-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 96 rows in the Interval table after the DTM agent is active for 24 hours.

dsx1IntervalValidData

If the data for this interval is valid. This variable is not supported.

Syntax

TruthValue

Access

Read-only

OID

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

dsx1InvalidIntervals

Number of intervals with invalid data. This variable is not supported.

Syntax

Integer

Access

Read-only

OID

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

dsx1LineIndex

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 in which the agent finds the interfaces on the boards. This order does not necessarily represent the physical order of the interfaces.

For Hot Swap, valid values include:

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

dsx1LineCoding

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 can 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 that meets or exceeds the pulse density requirements, such as inverting HDLC data.
other	6	Unlisted (default).

Configuration

To configure this object, edit the system configuration file before starting the Chassis MIB agent.

dsx1LineLength

Length of the DS1 line in meters. This variable is not supported.

Syntax

Integer

Access

Read/write

OID

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

dsx1LineStatus

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.

Status	Value	Description
dsx1OtherFailure	4096	Interface status not defined.

Note: Far end is not supported.

dsx1LineStatusChange

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

Syntax

TruthValue

Access

Read-only

OID

p.15.0.1

dsx1LineType

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.

dsx1LineStatusChangeTrapEnable

Whether traps are 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

Valid values:

Value	Description
1	Enabled
2	Disabled (default)

dsx1LoopbackConfig

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 always returns 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 passes 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 always returns dsx1NoLoop (1) because loopback is not supported.

dsx1LoopbackStatus

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.

dsx1SendCode

Type of code 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.

dsx1SignalMode

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.

dsx1StatusLastChange

Time 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.

dsx1TimeElapsed

Number of seconds 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.

dsx1TransmitClockSource

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 known as a slave.
localTiming	2	The recovered receive clock from another interface is used as the transmit clock. Also known as the master.
throughTiming	3	A local clock source is used.

Configuration

To configure this object, edit the system configuration before starting the Chassis MIB agent.

dsx1ValidIntervals

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 this case, the value indicates the number of 15-minute intervals that the agent has been running minus 1 (because the time periods start with 1).

9. Software Revision MIB overview

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, the 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 Natural Access. 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.

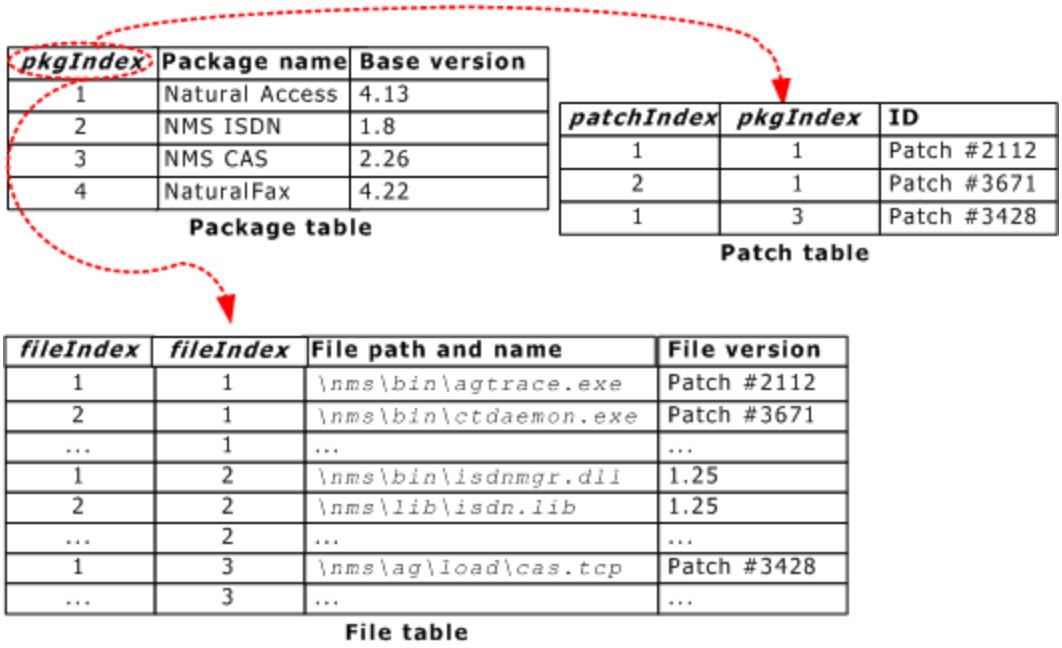
Once you have installed NMS packages, service packs, or patches, the values in the Software Revision MIB are updated automatically when you restart the Natural Access Server (*ctdaemon*).

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 the relationship between the tables in the Software Revision MIB:



pkgIndex	Package name	Base version
1	Natural Access	4.13
2	NMS ISDN	1.8
3	NMS CAS	2.26
4	NaturalFax	4.22

Package table

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

Patch table

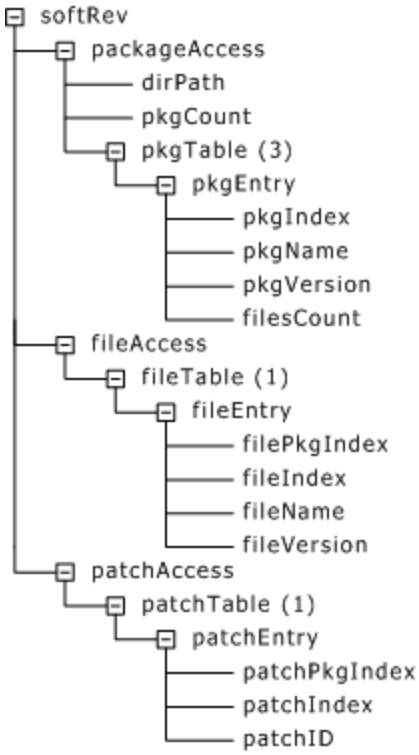
fileIndex	*fileIndex*	File path and name	File version
1	1	\nms\bin\agtrace.exe	Patch #2112
2	1	\nms\bin\ctdaemon.exe	Patch #3671
...	1	...	...
1	2	\nms\bin\isdnmgr.dll	1.25
2	2	\nms\lib\isdn.lib	1.25
...	2	...	...
1	3	\nms\ag\load\cas.tcp	Patch #3428
...	3	...	...

File table

As the illustration shows, 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 illustration, *agtrace.exe* and *ctdaemon.exe* are part of the Natural Access 4.13 package (***pkgIndex*** #1). Since the package was installed, Patch #2112 has modified *agtrace.exe* and Patch #3671 has modified *ctdaemon.exe*.

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***.

The following illustration shows the sequence of objects in the 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 are located.
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.

The **pkgIndex** object provides an index into the File and Patch tables described in this topic. For more information, refer to the Software Revision MIB representation.

File table

The File table contains a File Entry table. The File Entry 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 file name of the file.
fileVersion	Base version of the file.

Patch table

The Patch table contains a Patch Entry table. The Patch Entry 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 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.

10. Software Revision MIB object reference

Using the Software Revision MIB object reference

This section describes the objects in the Software Revision MIB. A typical object description includes:

Syntax	Datatype of the object. 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	Type of access allowed for this object. Options are: • Read-only: The object cannot be modified by SNMP. • Read/write: SNMP can configure this object.
OID	Path from the root to this object. All OIDs start with ***p***, where ***p*** is 1.3.6.1.4.1.2628.2.1 (the OID for the SOFTREV-MIB).
Details	Object description.
Configuration	How to configure the object, if configurable.
Example	Example of how the object is used.

NMS SNMP MIBs are compiled using the following text files, located in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX):

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

To display the SNMP information for the proprietary agent, read the appropriate file using the Windows Console Management function.

dirPath

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

filesCount

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 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

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

The agent internally assigns this identifier.

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 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

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

The agent internally assigns this identifier.

fileName

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 are imported from the *.sgn* files.

fileVersion

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 was manually modified or corrupted since it was installed by NMS software.

Configuration

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

patchAccess

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

Syntax

Object

Access

Not accessible.

OID

p.3

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

Index of a service pack or patch in the table.

Syntax

Integer

Access

Read-only

OID

p.3.1.1.2.***n***

Configuration

The agent internally assigns this identifier.

patchPkgIndex

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

The agent internally assigns this identifier.

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

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 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.

pkgCount

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 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

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

The agent internally assigns this identifier.

pkgName

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 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 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

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 are imported from the *.sgn* files.

11. OAM Database MIB overview

OAM Database MIB representation

The OAM Database MIB presents an SNMP front end to the contents of the OAM API 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 about NMS OAM, refer to the *NMS OAM System User's Manual* and to the *NMS OAM Service Developer's Reference Manual*.

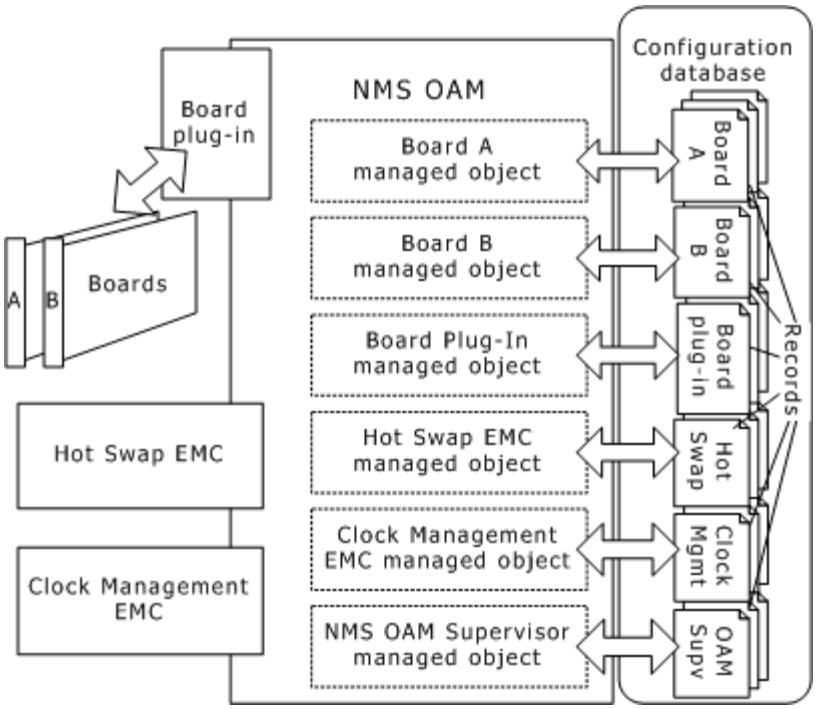
Managed components

NMS OAM manages the following components:

Component	Description
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 that 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 that add functionality to the OAM API. 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 information about the Hot Swap EMC and the Clock Management EMC, refer to the *NMS OAM System User's Manual*.

The following illustration shows 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 OAM API database:



OAM Database MIB tables and keywords

This topic presents:

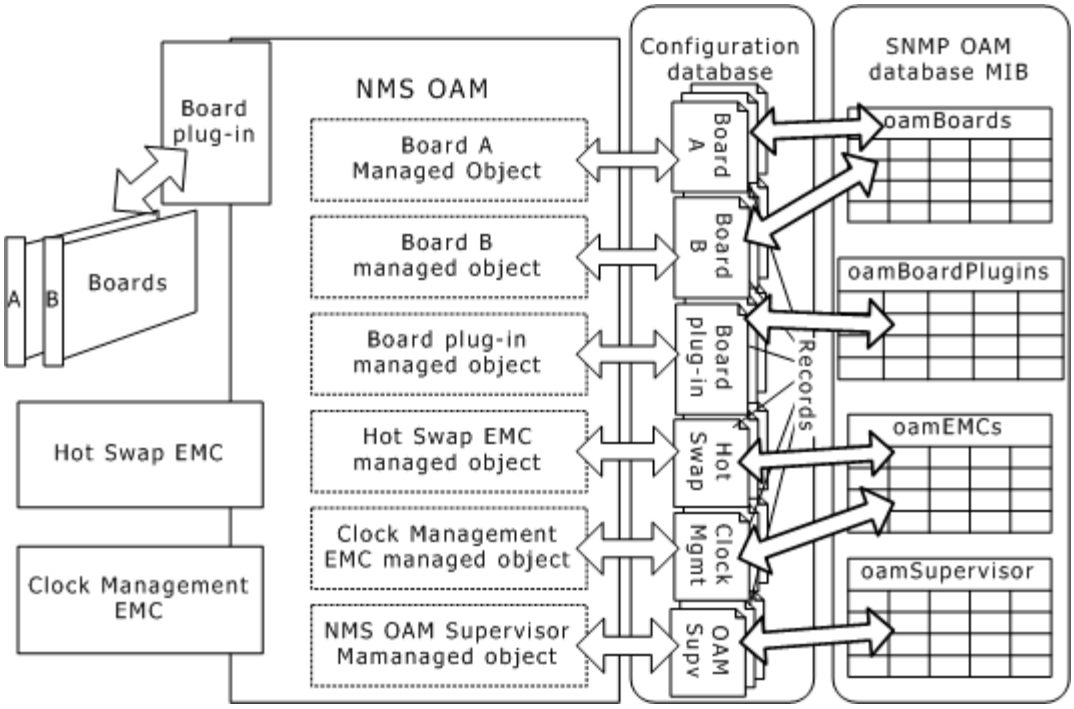
- OAM Database MIB tables
- Keywords in the OAM Database MIB
- Populating OAM Database MIB tables

OAM Database MIB tables

The OAM Database MIB consists of the following tables:

Table	Contains
oamSupervisor	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	Configuration data for each installed board plug-in.
oamEMCs	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	Configuration data for other managed objects (if any).
oamEventsTraps	NMS OAM events.

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



Keywords in the OAM Database MIB

Configuration data in both the OAM API 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 OAM API database, and vice versa.

Each keyword has attributes, called qualifiers. For example, the qualifier Type indicates the type of value it accepts (integer, string). 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:

Information	Description	Valid values
Managed object index	The index of the managed object to which the keyword belongs.	Any integer from 1 and higher.
Index	A unique (within the table) index for the keyword.	Any integer from 1 and higher.
Keyword name	The name of the keyword.	The keyword name, preceded by one or more group keyword names separated by periods.

Information	Description	Valid values
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	Keyword value is read-only or read/write. 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 or no choice, this field contains a string in 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 in 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 that enable 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 that is within clocking is expressed as Clocking.HBus.FallBackClockSource.

Populating OAM Database 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.

The agent creates a row in the Supervisor Keyword table (**supervisorTable**) for each integer or string keyword in the NMS OAM Supervisor managed object, and stores the keyword and qualifier information. 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.

OAM Supervisor table

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 the OAM API 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 OAM API alert notification or indicates the status of the registration.
supervisorTable	Supervisor Keyword table. Contains NMS OAM Supervisor keywords, values, and qualifiers.
oamCreateBoard	Create Board table. Contains values that enable 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	Beginning of a row of the Supervisor Keyword table.
supervisorIndex	Unique index (within this table) identifying the keyword.
keywordName	Keyword name, formatted as described in OAM Database MIB tables and keywords.
kwValue	Value of the keyword.
kwType	Type of the keyword: integer, string, or object.
kwMode	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.
kwAllowedRange	Range of allowable values for the keyword, formatted as described in OAM Database MIB tables and keywords.
kwDescription	Short description of the keyword.

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

	supervisorInd	keywordName	kwValue	kwType	kwMode	kwAllowedRang	kwDescriptipt
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>

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

Object	Description
productName	Product type of the board to create. 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 board, plug-in, and EMC keywords.
boardName	Name to give the created board.
boardNumber	Board number to give the created board.
applyBoardCommand	Set this value to 1 to create the board based upon the **productName**, **boardName**, and **boardNumber** values you specified.

OAM Board Plug-in table

The OAM Board Plug-in table contains keywords, values, and qualifiers for each board plug-in.

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

Object	Description
boardPluginTable	Top of the table.
boardPluginEntry	Beginning of a row of the OAM Board Plug-in table.
boardPluginIndex	Plug-in index. This value is equivalent to the index number of the BoardPlugins[***x***] keyword listing the board plug-in in the Supervisor managed object.
boardPluginKwIndex	Unique index (within this table) identifying the keyword.
bpikeywordName	Keyword name, formatted as described in OAM Database MIB tables and keywords.
bpikwValue	Value of the keyword.
bpikwType	Type of the keyword: integer or string.
bpikwMode	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.
bpikwAllowedRange	Range of allowable values for the keyword, formatted as described in OAM Database MIB tables and keywords.
bpikwDescription	Short description of the keyword.

To populate this table, the OAM Database SNMP agent opens the OAM API Supervisor managed object, and retrieves the values in BoardPlugins[x] keyword. This value 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 value is equivalent to the index of the BoardPlugins[x] keyword listing the managed object.
- A unique numerical index (**boardPluginKwIndex**), 1 or higher.

OAM EMC table

The OAM EMC (extended management component) table contains keywords, values, and qualifiers for each EMC.

The objects in the OAM EMC table (**oamEMCs**) are:

Object	Description
emcTable	Top of the table.
emcEntry	Beginning a row of the EMC table.
emcIndex	EMC index.
emcKwIndex	Unique index (within this table) identifying the keyword.
emckeywordName	Keyword name, formatted as described in OAM Database MIB tables and keywords.
emckwValue	Value of the keyword.
emckwType	Type of the keyword: integer or string.
emckwMode	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.
emckwAllowedRange	Range of allowable values for the keyword, formatted as described in OAM Database MIB tables and keywords.
emckwDescription	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 value 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 **oamEMC** table 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 value is equivalent to the index of the ExtendedManagementComponents[x] keyword listing the managed object.
- A unique numerical index (**emcKwIndex**), 1 or higher.

OAM Boards table

The OAM Boards table contains:

- The number of boards automatically detected in the system.
- The total number of boards registered to the OAM API.
- 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	Number of boards automatically detected in the system.
createdBoardCount	Total number of boards registered to the OAM API.
boardTable	Board Keyword table that contains 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	Beginning of a row of the Board Keyword table.
boardIndex	Board index.
boardKwIndex	Unique index (within this table) identifying the keyword.
brdkeywordName	Keyword name, formatted as described in OAM Database MIB tables and keywords.
brdkwValue	Value of the keyword.
brdkwType	Type of the keyword: integer or string.
brdkwMode	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.
brdkwAllowedRange	Range of allowable values for the keyword, formatted as described in OAM Database MIB tables and keywords.

Object	Description
brdkwDescription	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 value 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 value is equivalent to the index of the Boards[x] keyword listing the managed object. This index does not necessarily match the board number (the value of the Number keyword for the board).
- A unique numerical index (**boardKwIndex**), 1 or higher.

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	Queries or changes the board name.
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 OAM API database.

Other objects table

The OAM 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 contains a table of the keywords, values, and qualifiers for each object.

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

Object	Description
otherObjectsTable	Top of the table.
otherObjectsEntry	Beginning of a row of the Other Objects table.
otherObjectsIndex	Object index.
otherObjectsKwIndex	Unique index (within this table) identifying the keyword.
otherObjectskeywordName	Keyword name, formatted as described in OAM Database MIB tables and keywords.
otherObjectskwValue	Value of the keyword.
otherObjectskwType	Type of the keyword: integer or string.
otherObjectskwMode	1 indicates keyword value is read-only. 2 indicates keyword value is read/write.
otherObjectskwAllowedRange	Range of allowable values for the keyword, formatted as described in OAM Database MIB tables and keywords.
otherObjectskwDescription	Short description of the keyword.

12. OAM Database MIB object reference

Using the OAM Database MIB object reference

This section describes the objects in the OAM Database MIB. A typical object description includes:

Syntax	Datatype of the object. 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	Type of access allowed for this object. Options are: • Read-only: The object cannot be modified by SNMP. • Read/write: SNMP can configure this object.
OID	Path from the root to this object. All OIDs start with ***p***, where ***p*** is 1.3.6.1.4.1.2628.3.1 (the OID for the OAMDATABASE-MIB).
Details	Object description.
Configuration	How to configure the object, if configurable.
Example	Example of how the object is used.

SNMP API MIBs are compiled using the following text files, located in *\nms\ctaccess\doc* (*/opt/nms/ctaccess/doc* under UNIX):

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

To display the SNMP information for the proprietary agent, read the appropriate file using the Windows Console Management function.

applyBoardCommand

Creates a new board managed object in the NMS OAM database.

Syntax

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

Access

Read/write

OID

p.1.5.4

Details

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. 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 OAM API database. For more information, refer to the OAM Boards table.

boardIndex

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 OAM API database. For example, if Boards[1]=MyBoard, all keywords for this board in the OAM Database MIB have boardIndex equal to 1.

Configuration

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

boardKwIndex

Keyword's index.

Syntax

Integer

Access

Read-only

OID

p.4.3.1.2.***n***

Details

Keywords are numbered sequentially starting at 1.

Configuration

Keyword indexes are 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 that enable you to start, stop, test, delete, or query the status of a board.

boardManagementIndex

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

Enables 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.

boardName

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

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, refer to the OAM Board Plug-in table.

boardPluginIndex

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), **boardPluginIndex** is equal to 1 for all AG board plug-in 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, refer to the OAM Board Plug-in table.

boardPluginKwIndex

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, 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 OAM API database. For more information, refer to the OAM Boards table.

bpikeywordName

Board plug-in keyword name.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.2.1.1.3.***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 that 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, refer to the OAM Board Plug-in table.

bpikwAllowedRange

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 or no choice, **bpikwAllowedRange** contains a string in 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, refer to the OAM Board Plug-in table.

bpikwDescription

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, refer to the OAM Board Plug-in table.

bpikwMode

Whether 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, refer to the OAM Board Plug-in table.

bpikwType

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 MIB 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, refer to the OAM Board Plug-in table.

bpikwValue

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, 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.

brdkeywordName

Board keyword name.

Syntax

DisplayString (SIZE 0..255)

Access

Read-only

OID

p.4.3.1.3.***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 that 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, refer to OAM Boards table.

brdkwAllowedRange

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 or no choice, **brdkwAllowedRange** contains a string in this format:

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

If the keyword type is integer, and can take a range of values, **brdkwAllowedRange** contains a string in 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, see OAM Boards table.

brdkwDescription

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, see OAM Boards table.

brdkwMode

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, see OAM Boards table.

brdkwType

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, see OAM Boards table.

brdkwValue

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, see OAM Boards table.

brdName

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

Syntax

DisplayString (SIZE 0..255)

Access

Read/write

OID

p.4.4.1.2.***n***

brdNumber

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

Syntax

Integer

Access

Read/write

OID

p.4.4.1.3.***n***

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.

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.

createdBoardCount

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 when board managed objects are created or deleted.

detectedBoardCount

Number of boards physically detected for this board family.

Syntax

Integer

Access

Read-only

OID

p.4.1

Configuration

This value is updated when 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 EMC table (**emcTable**).

Syntax

Object

Access

Not accessible.

OID

p.3.1.1

Details

Each row in the EMC 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, refer to OAM EMC table.

emcIndex

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, refer to OAM EMC table.

emckeywordName

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 that 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, refer to OAM EMC table.

emckwAllowedRange

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 or no choice, **emckwAllowedRange** contains a string in this format:

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

If the keyword type is integer, and can take a range of values, **emckwAllowedRange** contains a string in 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, refer to OAM EMC table.

emckwDescription

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, refer to OAM EMC table.

emcKwIndex

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 EMC table as described in OAM EMC table.

emckwMode

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, see OAM EMC table.

emckwType

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, refer to the OAM EMC table.

emckwValue

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, refer to 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, refer to OAM EMC table.

keywordName

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 that 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, refer to OAM supervisor table.

kwAllowedRange

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 or no choice, **kwAllowedRange** contains a string in this format:

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

If the keyword type is integer, and can take a range of values, **kwAllowedRange** contains a string in 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, refer to OAM Supervisor table.

kwDescription

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, refer to OAM Supervisor table.

kwMode

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, refer to OAM Supervisor table.

kwType

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, refer to OAM Supervisor table.

kwValue

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, refer to OAM Supervisor table.

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.

oamBoardPlugins

Starts the OAM Board Plug-in table containing keywords, values, and qualifiers for each board plug-in.

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	Begins a row of the OAM Board Plug-in table.
boardPluginIndex	Plug-in index. This value is equivalent to the index number of the BoardPlugins[x] keyword listing the board plug-in in the Supervisor managed object.
boardPluginKwIndex	Unique index (within this table) identifying the keyword.
bpikeywordName	Keyword name, formatted as described in OAM Database MIB tables and keywords.
bpikwValue	Value of the keyword.
bpikwType	Type of the keyword: integer or string.
bpikwMode	1 indicates keyword value is read/write. 0 indicates keyword value is read-only.
bpikwAllowedRange	Range of allowable values for the keyword, formatted as described in OAM Database MIB tables and keywords.
bpikwDescription	Short description of the keyword.

oamBoards

Starts the OAM Boards table.

Syntax

Object

Access

Not accessible.

OID

p.4

Details

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 enable you to start, stop, test, or delete a board, change a board's name or number, or query its status.

oamCreateBoard

Starts the Create Board table containing values that enable 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 value to 1 to create the board based upon the **productName**, **boardName**, and **boardNumber** values you specified.

oamEMCs

Starts the EMC table containing keywords, values, and qualifiers for each EMC.

Syntax

Object

Access

Not accessible.

OID

p.3

Details

The objects in the EMC table are:

Object	Description
emcTable	Top of the table.
emcEntry	Begins a row of the EMC table.
emcIndex	EMC index.
emcKwIndex	Unique index (within this table) identifying the keyword.
emckeywordName	Keyword name, formatted as described in Keywords in the OAM Database MIB.
emckwValue	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	Short description of the keyword.

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 when a new OAM event is generated.

oamEventMask

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.

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	Last event sent back by NMS OAM.

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 do 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	Beginning of a row of the Other Objects table.
otherObjectsIndex	Object index.
otherObjectsKwIndex	Unique index (within this table) identifying the keyword.
otherObjectskeywordName	Keyword name, formatted as described in OAM Database MIB tables and keywords.
otherObjectskwValue	Value of the keyword.
otherObjectskwType	Type of the keyword: Integer or String.
otherObjectskwMode	1 indicates keyword value is read/write. 0 indicates keyword value is read-only.
otherObjectskwAllowedRange	Range of allowable values for the keyword, formatted as described in OAM Database MIB tables and keywords.
otherObjectskwDescription	Short description of the keyword.

oamStartStop

Stops or starts the NMS OAM Supervisor, or queries its status. For more information, refer to Starting and stopping the OAM Supervisor.

Syntax

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

Access

Read/write

OID

p.1.1

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 that contains NMS OAM Supervisor keywords, values, and qualifiers.
oamCreateBoard	Create Board table that contains values that enable you to create board instances in the database.

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 Database MIB tables.

otherObjectsIndex

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 Database MIB tables.

otherObjectskwAllowedRange

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 or no choice, **otherObjectskwAllowedRange** contains a string in this format:

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

If the keyword type is integer, and can take a range of values, **otherObjectskwAllowedRange** contains a string in 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 Database MIB tables.

otherObjectskwDescription

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 Database MIB tables.

otherObjectsKwIndex

Keyword's index.

Syntax

Integer

Access

Read-only

OID

p.5.1.1.2.***n***

Details

Keywords are numbered sequentially starting at 1.

Configuration

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

otherObjectskwMode

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 Database MIB tables.

otherObjectskeywordName

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 that 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 Database MIB tables.

otherObjectskwType

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 Database MIB tables.

otherObjectskwValue

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 Database MIB tables.

otherObjectsTable

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

Syntax

Object

Access

Not accessible.

OID

p.5.1

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 Database 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.

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 OAM Database SNMP agent configures the rows in the Supervisor Keyword table when it starts up. For more information, refer to Populating OAM Database MIB tables.

supervisorIndex

Keyword index.

Syntax

Integer

Access

Read-only

OID

p.1.4.1.1.***n***

Details

Keywords are numbered sequentially starting at 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, refer to OAM Supervisor table.

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 OAM Database SNMP agent configures the rows in the Supervisor Keyword table when it starts up. For more information, refer to Populating OAM Database MIB tables.

13. Using the NMS OAM Database MIB

Accessing board, plug-in, and EMC keywords

Complete the following steps to access a particular keyword for a board, a board plug-in, or an EMC:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Determine the index of the managed object containing the keyword. Access the Supervisor Keyword table (supervisorTable) and search for the managed object name in one of the following array keywords:
<table>
<tr><th>For this managed object...</th><th>Search this array...</th></tr>
<tr><td>Board</td><td>Boards[x]</td></tr>
<tr><td>Board Plug-in</td><td>BoardPlugins[x]</td></tr>
<tr><td>EMC</td><td>ExtendedManagementComponents[x]</td></tr>
</table></td></tr>
<tr><td>2</td><td>Access the table containing keywords for the managed object type:
<table>
<tr><th>For this managed object...</th><th>Access this table...</th></tr>
<tr><td>Board</td><td>boardTable</td></tr>
<tr><td>Board Plug-in</td><td>boardPluginTable</td></tr>
<tr><td>EMC</td><td>emcTable</td></tr>
</table>
Each of these tables is doubly linked. The first index, the managed object index, maps to the index you determined by accessing the Supervisor Keyword table.</td></tr>
<tr><td>3</td><td>Within the entries in the table beginning with the desired index, search for the keyword.</td></tr>
<tr><td>4</td><td>To set a keyword, first determine that it is read and 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.</td></tr>
</table>

Board settings do not take effect until the board is stopped and restarted.

Creating and deleting board managed objects

Use the items in the Create Board table (**oamCreateBoard**) to add a board managed object to the OAM API database.

Note: This operation does not require that the board currently be physically installed in the system.

Complete the following steps to add a board managed object:

Step	Action
1	Specify a valid product name for **productName**. To retrieve a list of valid product names, query 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 OAM API 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

Complete the following steps to delete a board managed object:

Step	Action
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 board names and numbers

For a board to be available, it must exist as a managed object in the OAM API database.

Querying a board name

Complete the following steps to set or query the name of a board:

Step	Action
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 a board number

Complete the following steps to set or query the board number of a board:

Step	Action
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 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. For a board to be available for starting, it must exist as a managed object in the OAM API database.

Complete the following steps to start or stop a board, or query its status:

Step	Action
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.**

Complete the following steps to test a board:

Step	Action
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 want to run (an integer between 1 and 255). For more information, refer to the *NMS OAM System User's Manual*.

Starting and stopping the OAM Supervisor

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.

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 with 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, such as 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 similarly 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 table lists valid mask values:

Mask	Value	Indication
TRAP_MASK_OAMEVN_ALERT	0x1	OAM alert generated.
TRAP_MASK_OAMEVN_REPORT	0x2	Special internal code used to log report info.
TRAP_MASK_OAMEVN_CREATED	0x4	Object created.
TRAP_MASK_OAMEVN_DELETED	0x8	Object deleted.

Mask	Value	Indication
TRAP_MASK_OAMEVN_RENAMED	0x10	Object renamed (text = new name).
TRAP_MASK_OAMEVN_TRACE	0x20	Trace info (potentially high-speed).
TRAP_MASK_OAMEVN_MODIFIED	0x40	Object modified (closed after write access).
TRAP_MASK_OAMEVN_BOARD_DEAD	0x80	Board failed.
TRAP_MASK_OAMEVN_STARTBOARD_DONE	0x100	Board successfully started.
TRAP_MASK_OAMEVN_STOPBOARD_DONE	0x200	Board successfully stopped.
TRAP_MASK_OAMEVN_TESTBOARD_DONE	0x400	Board test successfully initiated.
TRAP_MASK_HSWEVN_REMOVAL_REQUESTED	0x800	Board extraction began or board extraction initiated in software.
TRAP_MASK_HSWEVN_BOARD_OFFLINE	0x1000	Board off line.
TRAP_MASK_HSWEVN_BOARD_REMOVED	0x2000	Board removed.
TRAP_MASK_HSWEVN_BOARD_INSERTED	0x4000	Board inserted.
TRAP_MASK_HSWEVN_ONLINE_PENDING	0x8000	Board inserted and is about to go online.
TRAP_MASK_HSWEVN_PCI_CONFIG_FAILED	0x10000	PCI configuration attempt failed.
TRAP_MASK_HSWEVN_PREPARATION_FAILED	0x20000	Preparation for board removal failed.
TRAP_MASK_HSWEVN_BOARD_READY	0x40000	Board is ready.

By default, no masks are set. For more information about NMS OAM events, refer to the *NMS OAM Service Developer's Reference Manual*.

14. Demonstration programs

Using SNMP demonstration programs

The SNMP demonstration programs show how to use the NMS MIBs to provide information to a network administrator and how to get and set SNMP variables.

To run the demonstration programs, execute 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	Retrieves 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.

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 and extraction.

snmpGet

Retrieves and displays information about a specified SNMP agent running at a specified IP address.

Usage

```
snmpGet address nmssnmpoid options
```

where:

Parameter	Description
address	Address or DNS name of a local or remote host running the SNMP agent about which to return information.
nmssnmpoid	OID of an object in one of the MIBs available on the host for which you want 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

Complete the following steps to run *snmpGet*:

Step	Action
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 command: `snmpGet localhost`

The following example shows how to run *snmpGet*:

```
< snmpGet 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

Retrieves the value of the next object after a specified OID.

Usage

```
snmpNext address cnmssnmpoid options
```

where:

Parameter	Description
address	Address or DNS name of a local or remote host running the SNMP agent about which to return information.
nmssnmpoid	OID of an object in one of the MIBs available on the host for which you want 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

Complete the following steps to run *snmpnext*:

Step	Action
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 command: `snmpNext 10.1.20.46 1.3.6.1.2.1.1.1.0`

The following example shows how to run *snmpnext*:

```
>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

Sets the value of a specified SNMP object.

Usage

```
snmpSet address nmssnmpoid options
```

where:

Parameter	Description
address	Address or DNS name of a local or remote host running the SNMP agent about which to return information.
nmssnmpoid	OID of an object in one of the MIBs available on the host for which you want 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

Complete the following steps to run *snmpSet*:

Step	Action
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 command: `snmpSet localhost 1.3.6.1.4.1.2628.2.2.4.2.0`

An example of running this command to set **chassBoardTrapEnable** follows:

```
>snmpSet 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
```

snmpChassScan

Demonstrates how to:

- Navigate the chassis MIB.
- Retrieve chassis type and description.
- Navigate by bus.
- Show board description and status information.

Usage

```
snmpChassScan address options
```

where:

Parameter	Description
address	Address or DNS name of a local or remote host running the 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.

Procedure

Complete the following steps to run *snmpChassScan*:

Step	Action
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 command: `snmpChassScan`

The following example shows how to run *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_4040_1TE Segment:1 Slot:7 Status:OnLine
>Board 2:       AG_4040_2TE Segment:1 Slot:6 Status:OnLine
>
```

snmpHsMon

Demonstrates how to monitor a CompactPCI chassis, including how to receive traps when board status changes and how to remotely insert or extract a board.

Usage

```
snmpHsMon address
```

where:

Parameter	Description
address	Address or DNS name of a local or remote host running the 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*.

Procedure

Complete the following steps to run *snmpHsMon*:

Step	Action
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 command: `snmpHsMon`

The following example shows how to run *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_4040C_1TE Segment:1 Slot:10 Status:OffLine
Board 1:       AG_4040C_2TE Segment:1 Slot:11 Status:OffLine
Board 3:       AG_4040C_2TE Segment:1 Slot:15 Status:OffLine
Board 2:       CG_6000C 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

Shows the status of digital trunks of each board in a chassis.

Usage

```
snmpTrunkLog address options
```

where:

Parameter	Description
address	Address or DNS name of a local or remote host running the 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 information.

Procedure

Complete the following steps to run *snmpTrunkLog*:

Step	Action
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 command: `snmpTrunkLog <address>` where address is the IP address for the machine.

The following example shows how to run *snmpTrunkLog*:

```
> 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_4040_1TE)  Trunk:0 Status: Loss of frame, NoSgnl
Interface:3 Board:1 (AG_4040_1TE)  Trunk:1 Status: Loss of frame, NoSgnl
Interface:4 Board:1 (AG_4040_1TE)  Trunk:2 Status: Loss of frame, NoSgnl
Interface:5 Board:1 (AG_4040_1TE)  Trunk:3 Status: Loss of frame, NoSgnl
Interface:6 Board:2 (AG_4040_2TE)  Trunk:0 Status: In service
Interface:7 Board:2 (AG_4040_2TE)  Trunk:1 Status: Loss of frame, NoSgnl
Interface:8 Board:2 (AG_4040_2TE)  Trunk:2 Status: In service
Interface:9 Board:2 (AG_4040_2TE)  Trunk:3 Status: Loss of frame, NoSgnl
```

15. 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. The common information model is an extensible data model for logically organizing management objects in a consistent, unified manner in a managed environment. WBEM technology 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. 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. 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. WMI 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 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

This topic describes how to install and verify the Microsoft WMI and the WMI SNMP provider.

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 is available through WBEM and WMI, check the NMS-related information using the SNMP demonstration programs. For more information, refer to Using SNMP demonstration programs.

Installing WMI software

Complete the following steps to install the SNMP Provider:

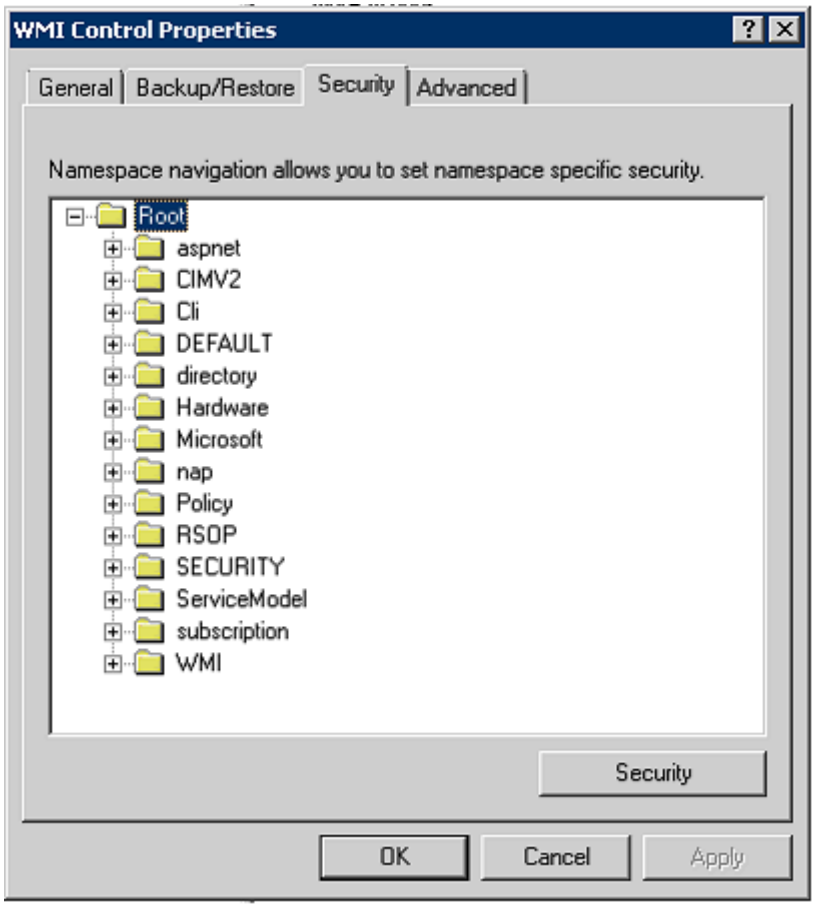
Step	Action
1	From the **Control Panel**, select **Programs**.
2	Under **Programs and Features**, select **Turn Windows features on or off**.
3	In the Windows features list, scroll down to **SNMP feature** and expand the list so that you can see **WMI SNMP Provider**.
4	Select the check box for WMI SNMP Provider. The check box for SNMP feature is selected automatically because the provider requires SNMP.
5	Click **OK**.
6	From a command prompt or the **Start** menu, run *Services.msc* and ensure that the SNMP service is started.

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 are properly created, browse for the namespaces in the WMI Control dialog box:



Complete the following steps to access the WMI Control dialog box in Windows 2008 Server:

Step	Action
1	Double-click on **Administrative Tools**. The Administrative Tools window displays.
2	Double-click on **Computer Management**. The Computer Management window displays.
3	Under **Services and Applications**, highlight **WMI Control**.
4	In the **Action** menu, click **Properties**. The WMI Control Properties dialog box displays.
5	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 are found in the *\nms\ctaccess\demos\snmp\wbem* directory:

This file...	Is the MOF file for the...
nmsChassis.mof	Chassis MIB.
nmsTrunk.mof	Trunk MIB.
nmsOamDatabase.mof	OAM Database MIB.
nmsSoftRev.mof	Software Revision MIB.
nmsRtp.mof	RTP MIB that is installed with NMS Fusion.

If the namespaces are properly created, SNMP-related MOF files can be added into the CIMOM repository by completing the following steps:

Step	Action
1	Open an MS-DOS console.
2	Navigate to the directory *\nms\ctaccess\demos\snmp\wbem*.

3	Enter the following command 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!`

4	To view the contents of the repository, enter:
	`smi2smir /l`
	Information similar to the following displays:
	`smi2smir : Version 1.50.1085.0000`
	
	`smi2smir : Modules in the SMIR :`
	`"NMS_CHASSIS"`

If the repository is not correctly updated by the *mofcomp* utility, complete the following steps:

Step	Action
1	Delete the entire repository by entering: `smi2smir /p`
2	Add the MOF files as described in the previous procedure.
3	Stop the WMI service by entering: `net stop winmgmt`
4	Restart the WMI service by entering: `net start winmgmt`

Testing MOF files

Once the MOF files are 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	JScript program that enumerates SNMP objects in the system.
enumsnmp.htm	HTML file containing an embedded JScript program that 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

Use one of the following methods to launch *enumsnmp.js*:

- Double-click on the file *enumsnmp.js* in a Windows Explorer window.

  This action launches the script with *wscript.exe*, the default WSH (Windows Scripting Host) interpreter. If *enumsnmp.js* is launched this way, a dialog box displays 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 displays in the console window.

  The following example shows partial output of *enumsnmp.js* when launched with *cscript*:

```
1.  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 sample results returned by *enumsnmp.htm* when opened in Internet Explorer:

	supervisorInd	keywordName	kwValue	kwType	kwMode	kwAllowedRang	kwDescripti
1	1	ExtendedMana [0]	clkmgr.emc	Object	readOnly	<no range>	<none>
2	2	ExtendedMana [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>

Index