MG-SOFT Corporation

Visual MIB Builder 2013

USER MANUAL

(Document Version: 3.7)

Document published on Thursday, 25-October-2012

In order to improve the design or performance characteristics, MG-SOFT reserves the right to make changes in this document or in the software without notice.

Depending on your license, certain functions described in this document may not be available in the version of the software that you are currently using.

Screenshots used in this document may slightly differ from those on your display.

MG-SOFT may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

Thank you for choosing the MG-SOFT Visual MIB Builder.

MG-SOFT Corporation, established in March 1990, is the world's leading supplier of SNMP, SMI, NETCONF, YANG and general network management applications, toolkits and solutions for Windows, Linux, Mac OS X and Solaris platforms. MG-SOFT provides major IT companies worldwide with network management applications as well as with toolkits implementing core network management technologies. Furthermore, MG-SOFT provides customers with consulting services, custom made turn-key software products, solutions and/or services and network management integration solutions based on our extensive know-how and vast experience in network management technologies.

MG-SOFT has developed the world's first 32-bit SNMP protocol stack implementation for MS Windows operating systems and one of the first SNMPv3 implementations for Win32 platforms. As of today, MG-SOFT's WinSNMP API implementation provides a solid base for all MG-SOFT's SNMP applications (as well as for thousands of third-party applications, built by our clients who licensed our WinSNMP API) running on a number of operating system platforms: MS Windows (32-bit, 64-bit, embedded CE), Linux (32-bit and 64-bit), Mac OS X (PPC and Intel platforms, 32-bit and 64-bit), Mac iOS (iPad) and Solaris (Sparc and Intel platforms).

MG-SOFT is also active in the network configuration management area and offers a full line of NETCONF and YANG software products, ranging from a graphical YANG and YIN file explorer, over Visual YANG definition file designer up to full blown NETCONF configuration manager.

For additional information about MG-SOFT Corporation, please contact the following address:

MG-SOFT Corporation
Strma ulica 8
2000 Maribor
Slovenia

Phone: +386 2 2506565
Fax: +386 2 2506566
E-mail: info@mg-soft.com
URL: http://www.mg-soft.com/

1.1 Product Description

MG-SOFT Visual MIB Builder is an application for designing and editing SNMP MIB module definition files. Visual MIB Builder has an easy to use drag-and-drop user interface that lets you design a MIB definition file within minutes, no matter if you are creating a SMIv1 or SMIv2 MIB, whether the MIB module contains tables or not, etc. The software incorporates the New MIB Module Wizard, a user-friendly tool that guides you through the process of designing the basic structure of new MIB modules. To create additional MIB objects, simply drag appropriate nodes to the MIB tree and specify their attributes in the Properties panel. Thus, you do not need to have extensive knowledge of ASN.1 syntax or MIB module definition language to create syntactically and semantically correct MIBs that will compile in any SMI compliant MIB Compiler. This means that the software is suitable for beginners in SMI. On the other hand, due to full support for SMIv1 and SMIv2 standards and advanced features like SMI revision control, Visual MIB Builder is also suitable for experienced SMI gurus.

Note: MG-SOFT Visual MIB Builder is a stand-alone application that rounds up MG-SOFT's SMI product line, along with MIB Explorer, MG-SOFT MIB Compiler and WinMIB API.

Visual MIB Builder incorporates a strict SMI consistency checker that verifies whether the MIB definitions fully comply with the SMIv1 or SMIv2 specification rules and generates descriptive messages that help you eliminate all MIB definition inconsistencies, before exporting the built module into the MIB module definition file format. Visual MIB Builder also lets you convert SMIv1 MIBs into SMIv2 MIBs and vice-versa.

Additionally, SMI Consistency Wizard significantly enhances Visual MIB Builder's ability to open and repair inconsistent MIB definition files. Using this wizard, you can fix 'broken' MIB definition files in minutes.

1.2 Introduction to SNMP – Based Management

Simple Network Management Protocol (SNMP) is a communication specification that defines how management information is exchanged between network management application (SNMP manager) and SNMP agents. SNMP manager can monitor or modify the attributes of any SNMP-manageable device by retrieving and modifying management information through SNMP agent implemented in that device. In addition, SNMP agents issue SNMP notification messages to SNMP managers to report events that occur on the managed device.

All systems directly manageable by SNMP must implement a processing entity called an SNMP agent that has access to the system’s management information. An SNMP agent receives SNMP requests from the management stations to retrieve or modify system’s management information and send responses to those messages. Management information is viewed as a collection of managed objects, residing in a virtual information store, termed the Management Information Base (MIB). Collections of related objects are defined in MIB modules. These modules are written in MIB module definition language. The MIB module definition language and rules for writing MIBs are defined in the Structure of Management Information (SMI) specification.

MIB files are typically produced by the vendors of an SNMP-manageable devices and contain a definition of objects, object hierarchy, and object attributes that describe the characteristics of the managed device and serve as a roadmap for monitoring and managing that device.

Visual MIB Builder is a specialized application that lets you create and edit MIB modules in a highly productive way. Due to its intuitive graphical drag-and-drop environment, where most of edits are visual, you do not need much knowledge about MIB module definition language or ASN.1 syntax to create MIBs. In Visual MIB Builder, a MIB module is graphically displayed as a hierarchically arranged tree structure (MIB tree), where different types of management information in the MIB tree are represented with different icons, i.e., each type (or class) of MIB object is represented by a unique icon. To define a MIB object, the user only has to drag-and-drop appropriate icon from the Components window onto the MIB tree, and specify the attributes of MIB object represented by such icon in the accompanying dialog.

The MIB module definition language and rules for writing MIBs are defined in a collection of documents called the SMI. SMI specifies exactly how individual type of management information should be defined. For each type of management information different construct is used, e.g., OBJECT IDENTIFIER construct, OBJECT-TYPE construct, NOTIFICATION-TYPE construct, etc. MIB module definition language, which is an adapted subset of the ASN.1 (Abstract Syntax Notation One) language, makes use of constructs taken directly from ASN.1, and constructs created for defining SNMP MIB modules through use of ASN.1 macro facility. Both types of constructs are used for defining MIB modules, i.e., to define SNMP management information, SNMP events, administrative assignments, describe MIB modules, group management information and events, and specify the requirements for conformance and implementation characteristics.

Currently there are two versions of SMI specification, SMIv1 and SMIv2. SMIv2 offers a richer and more precise syntax for defining MIB modules than SMIv1. However, some MIB compilers and other MIB processing tools may still support SMIv1 MIBs only. Whether you decide to build SMIv1 or SMIv2 MIBs is completely up to you - Visual MIB Builder lets you design both, and what’s even more, it also lets you convert MIB modules from SMIv1 into SMIv2 syntax and vice-versa.

Note: As you may have noticed, different terminology is used when referring to types of management information. The term ***MIB object*** is used when talking about the definition of managed objects, i.e., when we are defining the characteristics of a managed device.

The term ***node*** is used when referring to a graphically represented MIB object in the MIB tree. Each type of node is assigned a unique icon so that you can tell at a glance what type of information is the node associated with (e.g., scalar nodes, columnar nodes, trap nodes, object group nodes…).

Tip: For more information on how to build a MIB module that conforms to SMI rules, please refer to Understanding SNMP MIBs book (Perkins & McGinnis, Prentice Hall, ISBN 0-13-437708-7).

1.3 About This Manual

This manual contains detailed information about the MG-SOFT Visual MIB Builder program. The manual will guide you through the installation process and the use of the program. It also contains a *Usage example of creating an SMIv2 MIB* module that should give you an idea of what is like to build a MIB and bring you into the action. Please go through all the topics for complete knowledge on building a MIB module.

It is supposed that you are familiar with basic actions in a Windows environment such as choosing a main menu command or a pop-up command, dragging and dropping icons, etc.

Almost all MG-SOFT Visual MIB Builder operations can be accessed or started in different ways. You can either use:

- Main menu commands (e.g., ***Edit*** / ***Copy*** The construction ***Edit*** / ***Copy*** means: click the ***Edit*** command in the menu bar and select ***Copy*** from the sub-menu.)
- Toolbar buttons (e.g., ⎘)
- Pop-up commands (e.g., Copy - to use the ***Copy*** pop-up command, right-click inside a window, panel, or frame…).
- Keyboard shortcuts (e.g., ***Ctrl+C*** - hold down the ***Ctrl*** key and at the same time press the C key.)

The most of the procedures in this manual are described by using the main menu commands. However, you can use any of the above-mentioned shortcuts if available.

2 GETTING STARTED

This section presents the system requirements your computer has to meet to install MG-SOFT MIB Browser Professional Edition package.

2.1 Configuration Requirements

The MG-SOFT Visual MIB Builder application runs on Microsoft Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2012 and Windows 8.

Note: MG-SOFT Visual MIB Builder application is bundled in MG-SOFT MIB Browser Professional Edition package. Therefore, you have to install the full MIB Browser Professional Edition package in order to be able to use Visual MIB Builder. The MIB Browser Professional Edition package contains MIB Browser, Visual MIB Builder, MIB Compiler, and MIB Explorer applications.

For detailed system requirements, as well as installation and uninstallation procedures, please refer to the corresponding sections in MIB Browser User Manual.

3 STARTING VISUAL MIB BUILDER

Note: MG-SOFT Visual MIB Builder is a stand-alone application available in the MG-SOFT MIB Browser Professional Edition package. Therefore, you have to install all components of the MIB Browser Professional Edition package in order to be able to use Visual MIB Builder.

3.1 Starting Visual MIB Builder

1. From the Windows taskbar, select the ***Start*** / ***Programs*** / ***MG-SOFT MIB Browser*** / ***MIB Builder*** command.
2. As the program starts, the MG-SOFT Visual MIB Builder splash screen appears. To disable displaying the splash screen, uncheck the ***Splash on startup*** checkbox in the Preferences dialog box | Options tab (***View*** / ***Preferences***).
3. In a few seconds, the Visual MIB Builder desktop appears displaying the Tip of The Day message box. After reading the tips in this message box, click the ***Close*** button to start using the software.

Tip: To read the tips at any later time, select the ***Help*** / ***Tip of the Day*** command.

3.2 Visual MIB Builder Desktop

The Visual MIB Builder desktop follows the conventions of the appearance and functionality of the general Windows interface in that it has a title bar, menu bar, toolbar, status bar, and minimize, maximize and close buttons, but it differs in specific areas.

Menu bar

The bar near the top of the application window that contains names of the program menus. By clicking a menu name, a list of commands used to access various functions is displayed.

Toolbar

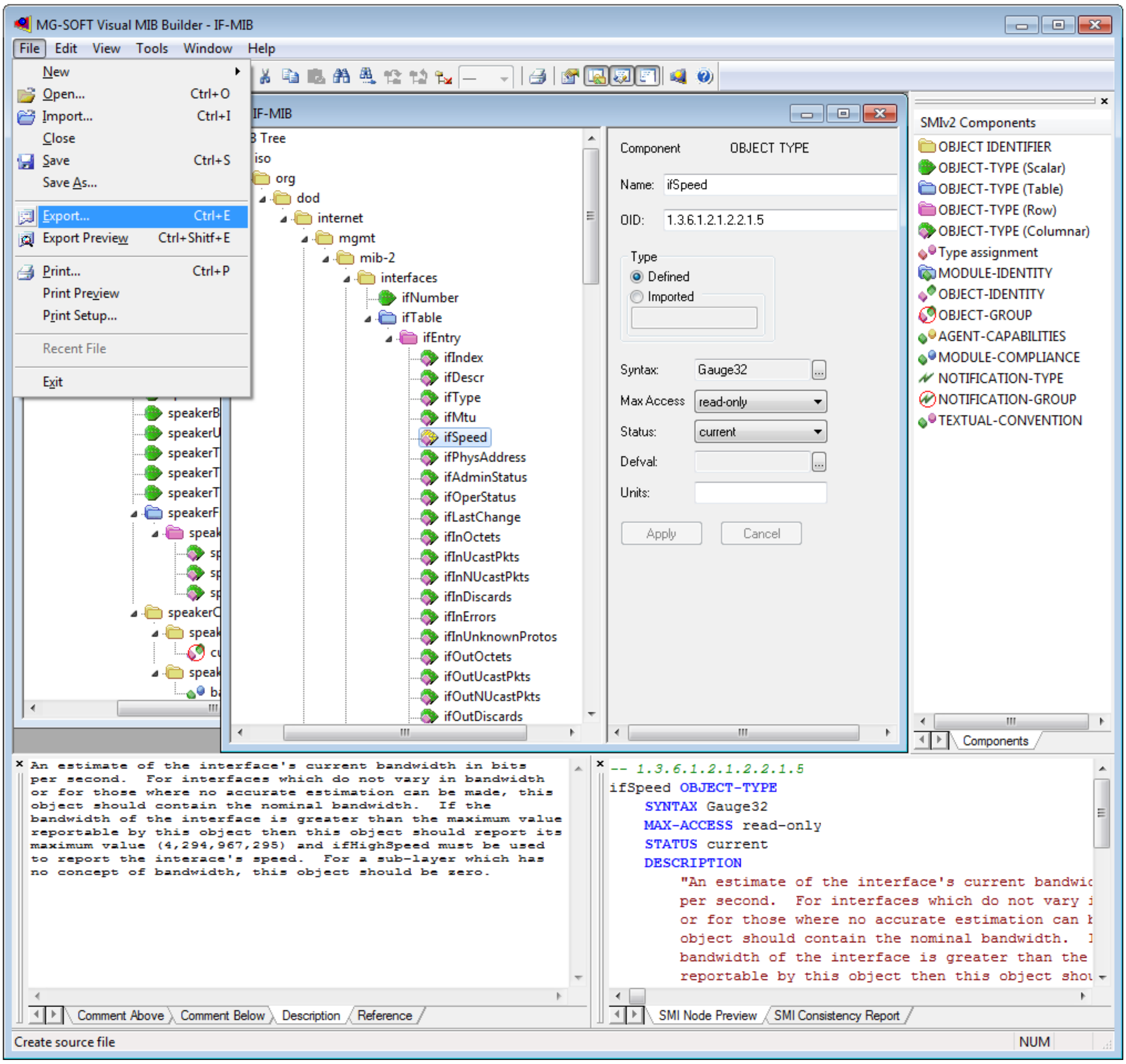
The toolbar is a group of buttons that provide quick access to a series of the most common commands. You can get a brief description of a task behind each toolbar button either in a tooltip, or in the Status bar, by placing the mouse cursor on the toolbar button (without clicking).

Working area

The working area is the area between the toolbar and the status bar. Each window or dialog box you open is displayed in this working area. After launching MG-SOFT Visual MIB Builder, it displays the **Components** window, the **Additional Properties** window, and the **Node Preview** window. Additionally, you can open any number of **MIB Builder** and **MIB Database** windows. For a detailed description of Visual MIB Builder windows, please see Visual MIB Builder help documentation (***Help / Help Topics / Visual MIB Builder Windows***).

Status bar

The status bar is the on-screen display area at the bottom of the main window, that shows information about tasks behind the toolbar button when placing the mouse cursor on the toolbar button (without clicking).



Component OBJECT TYPE

Name: ifSpeed

OID: 1.3.6.1.2.1.2.2.1.5

Type
● Defined
○ Imported

Syntax: Gauge32

Max Access: read-only

Status: current

Defval:

Units:

```
An estimate of the interface's current bandwidth in bits
per second.  For interfaces which do not vary in bandwidth
or for those where no accurate estimation can be made, this
object should contain the nominal bandwidth.  If the
bandwidth of the interface is greater than the maximum value
reportable by this object then this object should report its
maximum value (4,294,967,295) and ifHighSpeed must be used
to report the interace's speed.  For a sub-layer which has
no concept of bandwidth, this object should be zero.
```

```
-- 1.3.6.1.2.1.2.2.1.5
ifSpeed OBJECT-TYPE
    SYNTAX Gauge32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "An estimate of the interface's current bandwic
        per second.  For interfaces which do not vary i
        or for those where no accurate estimation can k
        object should contain the nominal bandwidth.  I
        bandwidth of the interface is greater than the
        reportable by this object then this object shou
```

Create source file

Figure 1: Visual MIB Builder desktop

4 OPENING EXISTING MIB MODULES IN VISUAL MIB BUILDER

Visual MIB Builder lets you open MIB modules saved in the following file formats:

- ❑ MIB Builder XML file format (**.buix**). This is a MG-SOFT's XML file format for storing MIB modules created by MG-SOFT Visual MIB Builder from version 2010 (8.0) onwards.
- ❑ MIB Builder file format (**.bui**). This is a MG-SOFT's proprietary binary file format for storing MIB modules created by MG-SOFT Visual MIB Builder prior to version 2010 (8.0). This file format is obsolete. It has been replaced by the .buix format.
- ❑ WinMIB database file format (**.smidb**). This is a MG-SOFT's proprietary binary file format for storing MIB modules compiled by MG-SOFT MIB Compiler. SMIDB files can be loaded and used by all MG-SOFT's applications, as well as by any other application implementing the industry-standard WinMIB interface.
- ❑ MIB module definition file format (e.g., ***.my, *.mib, *.smi, *.mi2, *.sm2, .txt,** ...). This is a plain ASCII file written in MIB module definition language. MIB modules published by the international standards bodies and the majority of MIB modules supplied by the vendors of SNMP-manageable devices are in this format.

To open an existing MIB module in Visual MIB Builder, do the following:

1. Start the application and select the ***File* / *Open*** command. The standard Open dialog box appears.
2. From the ***Files of type*** drop-down list select one of the following file masks:
   - ❑ Builder files (.buix). This option is selected by default.
   - ❑ WinMIB database files (.smidb)
   - ❑ MIB Source Files (*.mib, *.smi, *.mi2, *.sm2, *.my)
   - ❑ SMI files (*.buix, *.bui, *.smidb, *.mib, *.smi, *.mi2, *.sm2, *.my)+
   - ❑ All files (*.*)

   + Use this option to open **.bui** files created with previous versions of MIB Builder.
3. MIB files with filename extensions matching the selected file mask will be displayed in the upper list. Navigate to the folder containing a MIB file you want to open and click the ***Open*** button.

> **Note:** If you choose to open a MIB source file, such file is compiled in the background before it is displayed in the MIB Builder window. Visual MIB Builder can open MIB definition files that contain some commonly found errors. In such case, it may prompt you with a message box informing you of invalid or missing nodes. See the Using SMI Consistency Wizard section for instructions on how to correct these errors.

4. The selected MIB module appears in a new MIB Builder window. You can edit the MIB module by using the vast range of functions that the Visual MIB Builder offers and which are described in the following sections of this manual.
5. Make sure to save the changes you made to a MIB module by using the ***File* / *Save*** or ***File* / *Save As*** commands. For more information on saving a MIB module, read the Saving MIB Modules section of this manual.

5 DESIGNING NEW MIB MODULES

Creating MIB modules with Visual MIB Builder is a simple and quick procedure. In MIB Builder, each MIB module is visually represented in hierarchically arranged tree structure (MIB tree) and individual types of information (MIB objects) in the MIB tree are represented with icons based on the type of the information. The term *node* is used when referring to a graphically presented MIB object in the MIB tree (e.g., a scalar node, a columnar node, a notification node…). We use the term *MIB object* when talking about the SMI definition of managed objects. To define a MIB object in Visual MIB Builder, the user has to select appropriate node in the Components window, drag-and-drop it onto the MIB tree, and specify its parameters in the Properties window panel. Node icons in the Components window represent constructs used for defining MIB objects, e.g., OBJECT IDENTIFIER construct, OBJECT-TYPE construct, OBJECT-IDENTITY construct, etc.

Some constructs are used not to define the characteristics of the managed device but to describe and define the MIB module itself (e.g., MODULE-IDENTITY, MODULE-COMPLIANCE). Sometimes a MIB module contains only a small amount of information and sometimes MIB modules are very complex, according to the complexity of the device they describe. However, it is not necessary for a single MIB module to contain all information that is needed for managing a device. Instead of putting all definitions into a single MIB module, you can divide definitions into areas and put them in separate MIB modules. Of course, the division has to be logical and carefully planned.

The MIB module definition language and rules for writing MIBs are defined in a collection of documents called the Structure of Management Information (SMI). SMI specifies exactly how individual type of management information should be defined. To define different types of information, different constructs are used, e.g., OBJECT IDENTIFIER construct, OBJECT-TYPE construct, AGENT-CAPABILITIES construct, etc. Some of these constructs are taken directly from ASN.1 (e.g., OBJECT IDENTIFIER, Type Assignment, etc.) and some of them have been created for defining SNMP MIB modules through use of the ASN.1 macro facility. By means of this facility, SMIv1 defines two constructs: OBJECT-TYPE and TRAP-TYPE. SMIv2 redefines the OBJECT-TYPE construct and introduces eight additional constructs. Although SMIv2 offers a richer and more precise syntax for defining a MIB module than SMIv1, some MIB compilers and other tools may still support only SMIv1 MIBs.

Whether you decide to build SMIv1 or SMIv2 MIB modules is completely up to you - Visual MIB Builder lets you design both, SMIv1 and SMIv2 MIBs, and what’s even more, it lets you convert MIB modules from SMIv1 into SMIv2 syntax and vice-versa.

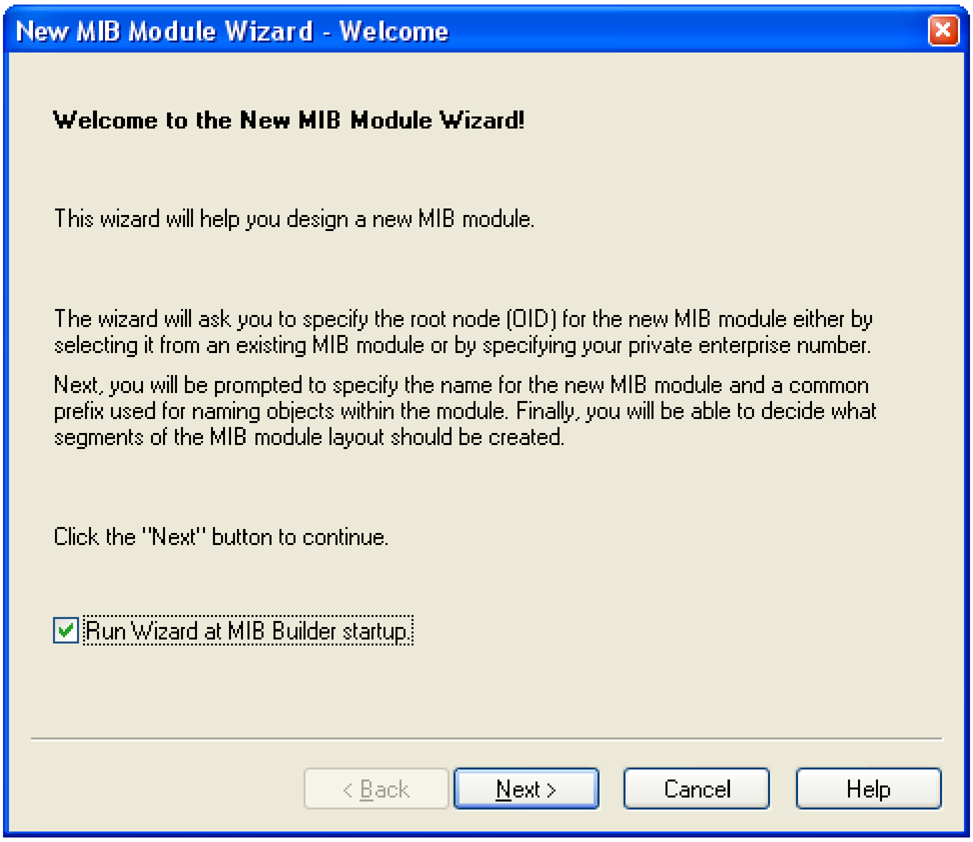
This section provides detailed instructions on designing and editing MIB modules. The section starts with description of using the New MIB Module Wizard to create a basic structure of a new SMIv2 MIB module. Then, instructions for manually designing and editing SMIv1 and SMIv2 MIB modules are provided. Please go through all the topics for complete knowledge on designing a MIB module.

5.1 Designing MIB Modules by Using New MIB Module Wizard

MG-SOFT Visual MIB Builder implements the New MIB Module wizard, a utility that helps you create the basic structure (skeleton) of a new SMIv2 MIB module in a few steps. To use the New MIB Module wizard, proceed as follows:

1. Select the ***File*** / ***New*** / ***MIB Module Wizard*** command. The `New MIB Module Wizard - Welcome` screen appears (Figure 2).

> **Note:** By default, the New MIB Module wizard is started automatically anytime you open Visual MIB Builder. To disable this option, uncheck the ***Run wizard at MIB Builder startup*** checkbox.



New MIB Module Wizard - Welcome

Welcome to the New MIB Module Wizard!

This wizard will help you design a new MIB module.

The wizard will ask you to specify the root node (OID) for the new MIB module either by selecting it from an existing MIB module or by specifying your private enterprise number.

Next, you will be prompted to specify the name for the new MIB module and a common prefix used for naming objects within the module. Finally, you will be able to decide what segments of the MIB module layout should be created.

Click the "Next" button to continue.

☑ Run Wizard at MIB Builder startup

Figure 2: New MIB Module Wizard - Welcome screen

2. Click the ***Next*** button to proceed to the next step.
3. The `Select MIB Module to Import` screen appears (Figure 3), prompting you to specify the MIB module from which the root OID for the new MIB module will be imported. You can either:

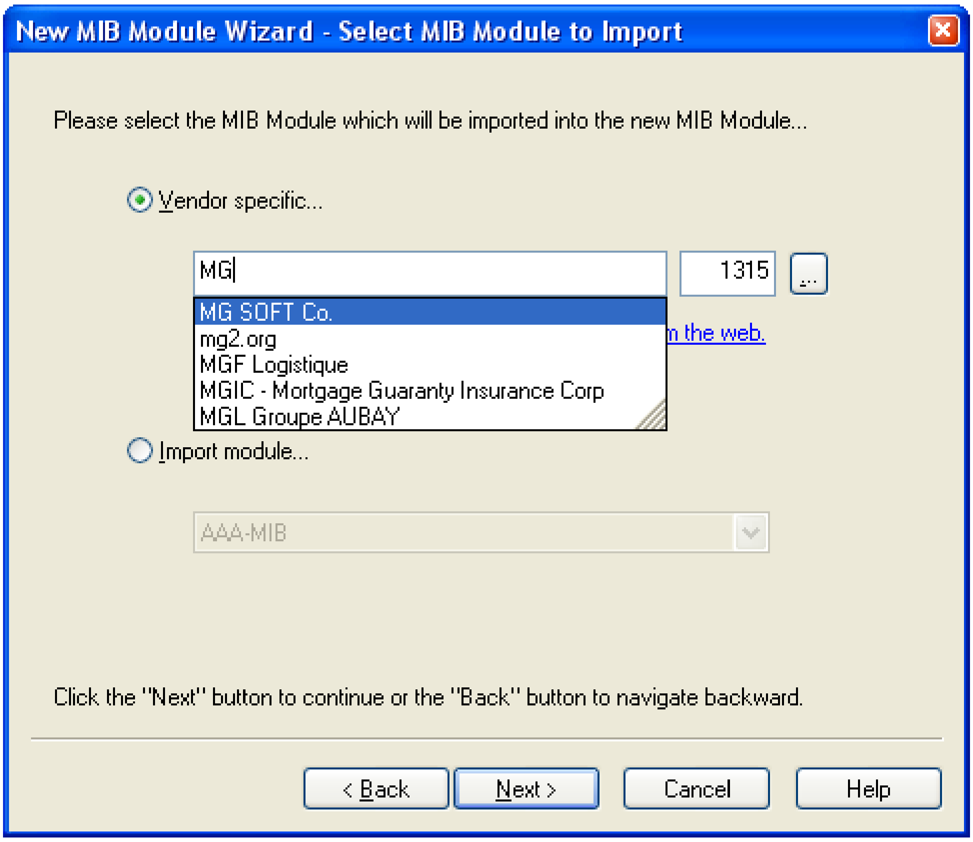
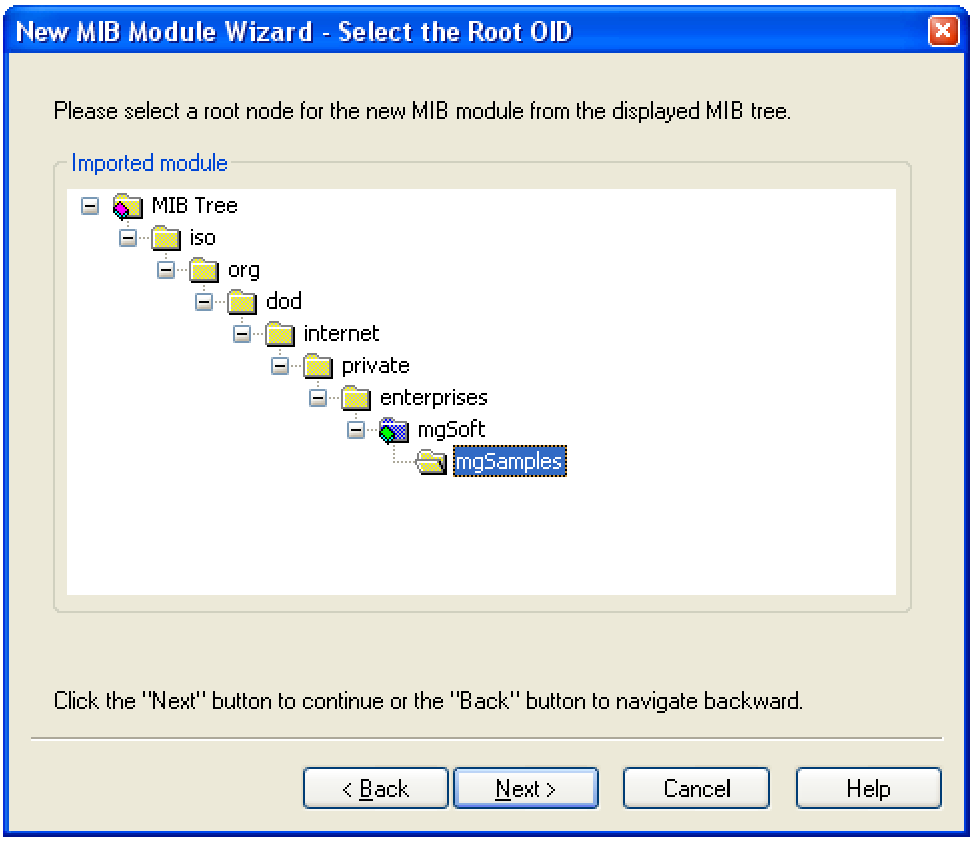

New MIB Module Wizard - Select MIB Module to Import
Please select the MIB Module which will be imported into the new MIB Module...
● Vendor specific...
MG
1315
...
MG SOFT Co.
mg2.org
MGF Logistique
MGIC - Mortgage Guaranty Insurance Corp
MGL Groupe AUBAY
n the web.
○ Import module...
AAA-MIB
Click the "Next" button to continue or the "Back" button to navigate backward.
< Back
Next >
Cancel
Help

Figure 3: New MIB Module Wizard - Select MIB Module to Import screen

- Choose the ***Import module*** option and select a MIB module from the list of MIB modules registered on your system. Selected MIB module must define the OID that will be used as the root OID for the MIB module you are designing, or
- Choose the ***Vendor specific*** option and specify your private enterprise name and/or number in the accompanying input lines. Clicking the ***Browse*** (**…**) button lets you select your private enterprise number from the list of private numbers registered with IANA. If your computer has access to the Internet, you can update the list of IANA registered enterprise numbers by clicking the “Download the latest IANA vendors definition file” hyperlink below the private enterprise number input lines.

> **Tip:** Choose the ***Vendor specific*** option if you do not know the name of the MIB module that defines the root OID for the MIB module you are building or if such MIB module does not exist yet (e.g., when designing your company’s top level MIB module).

4. Click the ***Next*** button to proceed to the next step.
5. If you have selected an existing MIB module in the previous step, the `Select the Root OID` screen appears (Figure 4), prompting you to choose the root OID from the MIB tree displayed in the central panel of the dialog. Select the desired node (OID) and click the ***Next*** button.



New MIB Module Wizard - Select the Root OID

Please select a root node for the new MIB module from the displayed MIB tree.

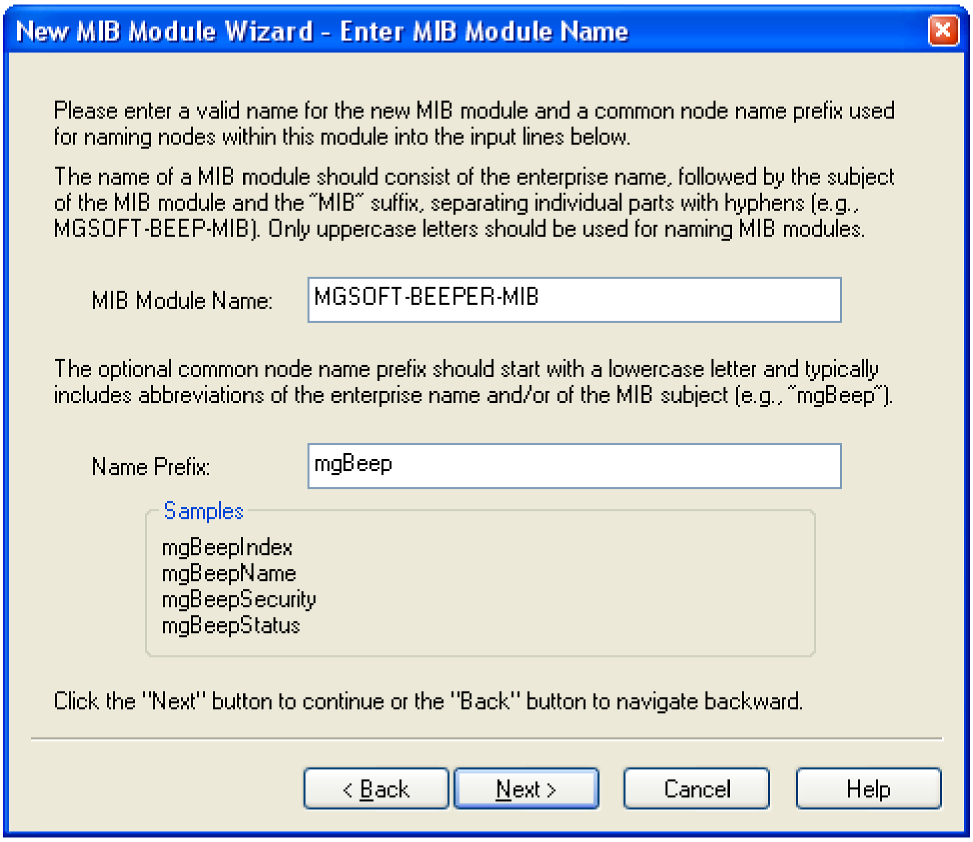
Imported module

Click the "Next" button to continue or the "Back" button to navigate backward.

Figure 4: New MIB Module Wizard - Select the Root OID screen

1. If you have specified your private enterprise number in the previous step, the New MIB Module wizard searches all MIB modules registered on your computer to determine if any of them already contains a definition of your enterprise OID. If such MIB module is found, the wizard displays its MIB tree and prompts you to select the desired OID from it to be used as a root OID for the new MIB module (as described in the previous step). Otherwise, the `Enter MIB Module Name` screen appears (Figure 5), as described below.
2. The `Enter MIB Module Name` screen asks you to enter a valid name for the new MIB module and (optionally), a common node name prefix used for naming nodes within the new MIB module.

   - Into the ***MIB module name*** input line, enter a name for the new MIB module. A name for a private MIB module should consist of three parts: the enterprise name, the subject of the MIB module, and a suffix “MIB”, “REG”, “TC”, etc. Individual parts a MIB name should be separated with hyphens (e.g., MGSOFT-BEEPER-MIB). Only uppercase letters should be used for naming MIB modules.
   - Into the ***Common node name prefix*** optionally enter a prefix that will be automatically inserted by MIB Builder as the name (label) for every node in this MIB module. Such prefix should start with a lowercase letter and typically includes the abbreviation of the enterprise name and/or of the MIB subject (e.g., “mgBeep”).



Please enter a valid name for the new MIB module and a common node name prefix used for naming nodes within this module into the input lines below.

The name of a MIB module should consist of the enterprise name, followed by the subject of the MIB module and the "MIB" suffix, separating individual parts with hyphens (e.g., MGSOFT-BEEP-MIB). Only uppercase letters should be used for naming MIB modules.

MIB Module Name: MGSOFT-BEEPER-MIB

The optional common node name prefix should start with a lowercase letter and typically includes abbreviations of the enterprise name and/or of the MIB subject (e.g., "mgBeep").

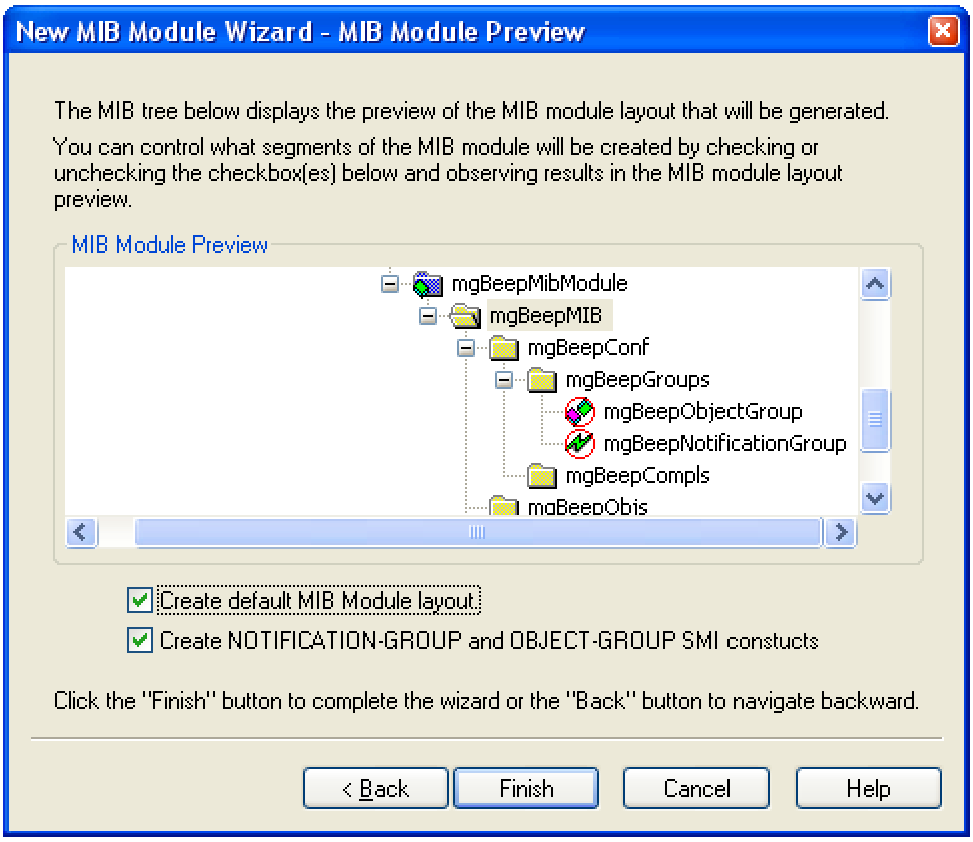
Name Prefix: mgBeep

Samples
mgBeepIndex
mgBeepName
mgBeepSecurity
mgBeepStatus

Click the "Next" button to continue or the "Back" button to navigate backward.

Figure 5: New MIB Module Wizard - Enter MIB Module Name screen

3. Click the ***Next*** button to proceed to the final screen (Figure 6).
4. The `MIB Module Preview` screen displays the preview of the MIB module tree structure that will be generated. You can control what segments of the MIB module will be created by checking or unchecking the following checkbox(es) and observing results in the MIB module layout preview:
   - ***Create default MIB module layout***
     This option builds the default MIB module layout, which can be global or product-specific (depending on the type of the MIB module you are building). For more details on this, see the About MIB Module Layouts Generated by Visual MIB Builder section.
   - ***Create NOTIFICATION-GROUP and OBJECT-GROUP definitions***
     This option includes the NOTIFICATION-GROUP and OBJECT-GROUP nodes into the MIB module as required by the SMIv2 specification.



The MIB tree below displays the preview of the MIB module layout that will be generated.

You can control what segments of the MIB module will be created by checking or unchecking the checkbox(es) below and observing results in the MIB module layout preview.

☑ Create default MIB Module layout.

☑ Create NOTIFICATION-GROUP and OBJECT-GROUP SMI constucts

Click the "Finish" button to complete the wizard or the "Back" button to navigate backward.

Figure 6: New MIB Module Wizard - final screen

5. Click the ***Finish*** button to complete the wizard.
6. The wizard will generate the new MIB module layout and display it in the new MIB Builder window (Figure 8).

Tip 1: For instructions on adding additional nodes to the generated MIB module or changing the properties of existing nodes, consult the Manually Designing New MIB Modules section.

Tip 2: To delete a node from the MIB module, select in it in the MIB tree and choose the ***Remove*** pop-up command or use the `CTRL+DELETE` keyboard keys.

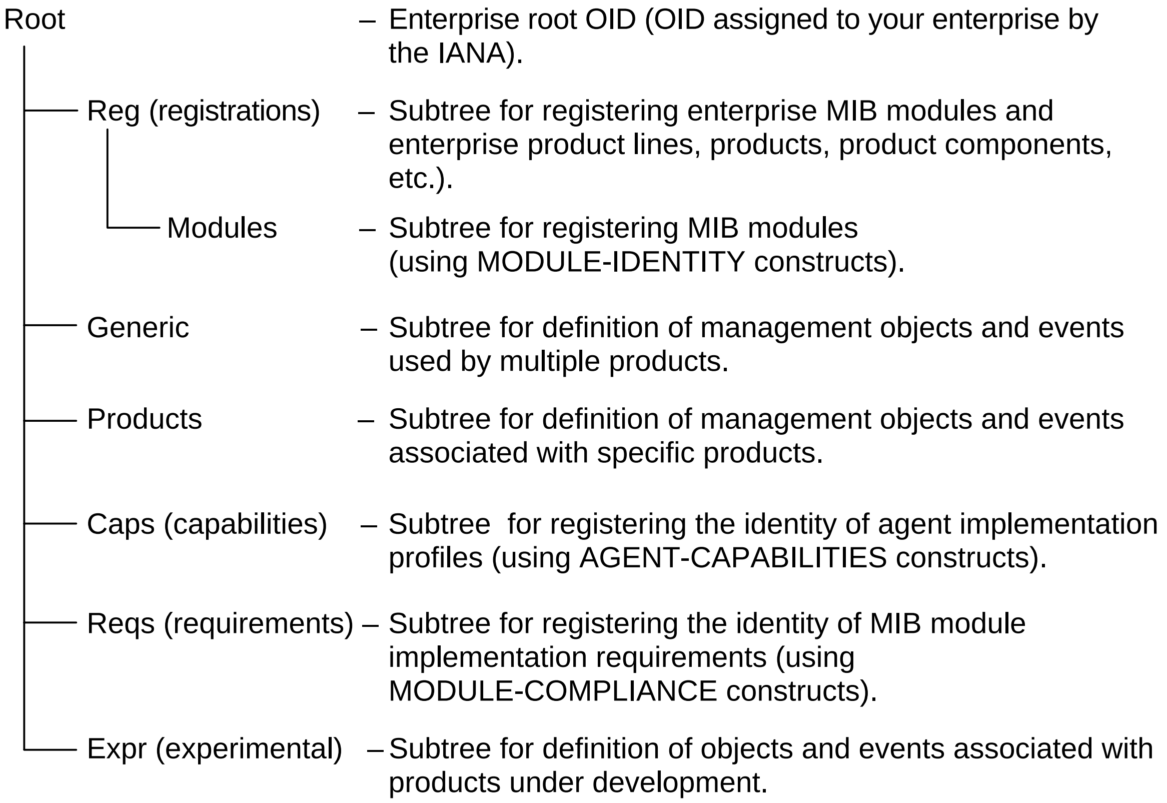
Tip 3: For instruction on moving nodes to different locations in the MIB tree, consult the Editing Existing MIB Modules section.

5.1.1 About MIB Module Layouts Generated by Visual MIB Builder

MG-SOFT Visual MIB Builder follows the suggestions for private OID namespace organization by D. Perkins and E. McGinnis (“Understanding SNMP MIBs” book, pages 214-225, Prentice Hall, ISBN 0-13-437708-7). Following these suggestions, the New MIB Module wizard can generate two types of private enterprise MIB module layouts:

- Enterprise global MIB module layout (registrations of enterprise top-level OIDs).
  This type of MIB module layout will be generated when none of the already registered MIB modules contains a definition of the root OID for the selected enterprise.
- Enterprise product-specific MIB module layout.
  This type of MIB module layout will be generated when the definition of the enterprise root OID already exists in one or more registered MIB modules.

Below is a description of the enterprise top-level (global) MIB module structure generated by Visual MIB Builder:



Root – Enterprise root OID (OID assigned to your enterprise by the IANA).

Reg (registrations) – Subtree for registering enterprise MIB modules and enterprise product lines, products, product components, etc.).

Modules – Subtree for registering MIB modules (using MODULE-IDENTITY constructs).

Generic – Subtree for definition of management objects and events used by multiple products.

Products – Subtree for definition of management objects and events associated with specific products.

Caps (capabilities) – Subtree for registering the identity of agent implementation profiles (using AGENT-CAPABILITIES constructs).

Reqs (requirements) – Subtree for registering the identity of MIB module implementation requirements (using MODULE-COMPLIANCE constructs).

Expr (experimental) – Subtree for definition of objects and events associated with products under development.

Figure 7: Enterprise top-level MIB module layout generated by Visual MIB Builder

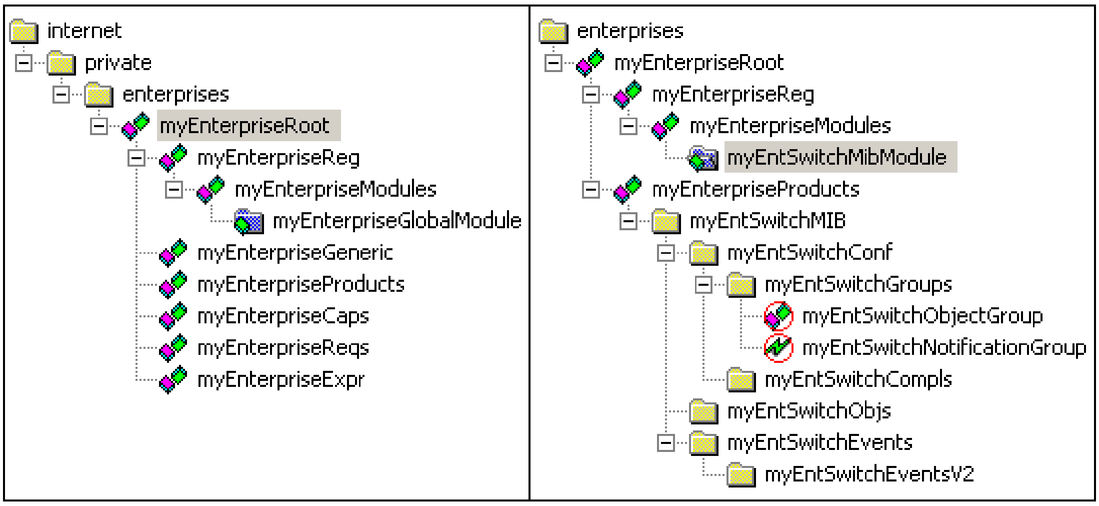
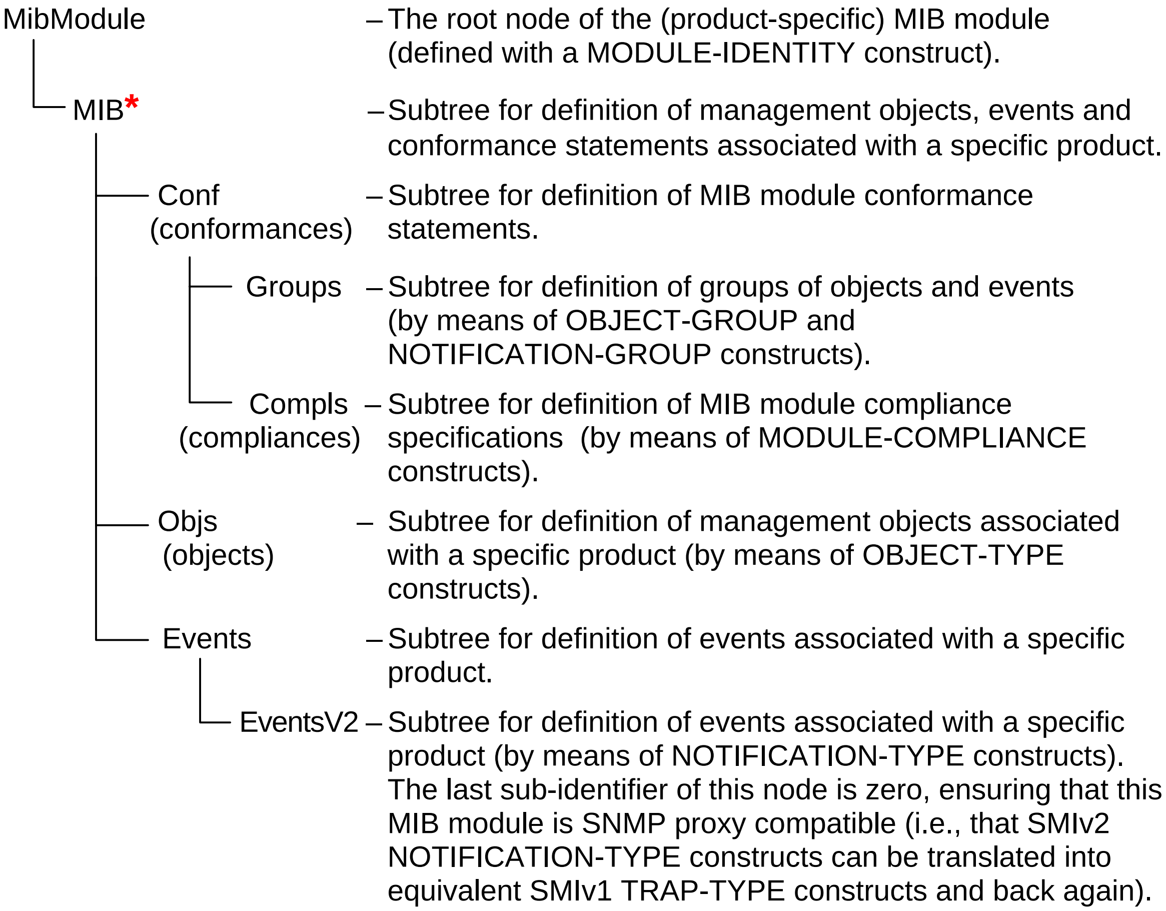


Figure 8: Example of a global and a product-specific MIB module layout

Below is a description of the product-specific MIB module structure generated by Visual MIB Builder:



MibModule – The root node of the (product-specific) MIB module (defined with a MODULE-IDENTITY construct).

MIB* – Subtree for definition of management objects, events and conformance statements associated with a specific product.

Conf (conformances) – Subtree for definition of MIB module conformance statements.

Groups – Subtree for definition of groups of objects and events (by means of OBJECT-GROUP and NOTIFICATION-GROUP constructs).

Compls (compliances) – Subtree for definition of MIB module compliance specifications (by means of MODULE-COMPLIANCE constructs).

Objs (objects) – Subtree for definition of management objects associated with a specific product (by means of OBJECT-TYPE constructs).

Events – Subtree for definition of events associated with a specific product.

EventsV2 – Subtree for definition of events associated with a specific product (by means of NOTIFICATION-TYPE constructs). The last sub-identifier of this node is zero, ensuring that this MIB module is SNMP proxy compatible (i.e., that SMIv2 NOTIFICATION-TYPE constructs can be translated into equivalent SMIv1 TRAP-TYPE constructs and back again).

Figure 9: Enterprise product-level MIB module layout generated by Visual MIB Builder

* This subtree is placed under the “Products” node in the enterprise top-level (global) MIB module structure, except in cases where the enterprise’s top-level OID organization does not follow the pattern used by Visual MIB Builder (see Figure 7). (All subtrees can be further subdivided, up to the desired level.)

5.2 Manually Designing New MIB Modules

In Visual MIB Builder, you can design a new MIB module either manually or by running the New MIB Module wizard, as described in the previous section. This section describes how to design a new MIB module manually.

5.2.1 Choosing the SMI Version

When creating a new MIB module, you first have to decide whether you are going to build a SMIv1 or SMIv2 MIB module. Depending on this selection, Visual MIB Builder lets you use different sets of components for visually designing a MIB module.

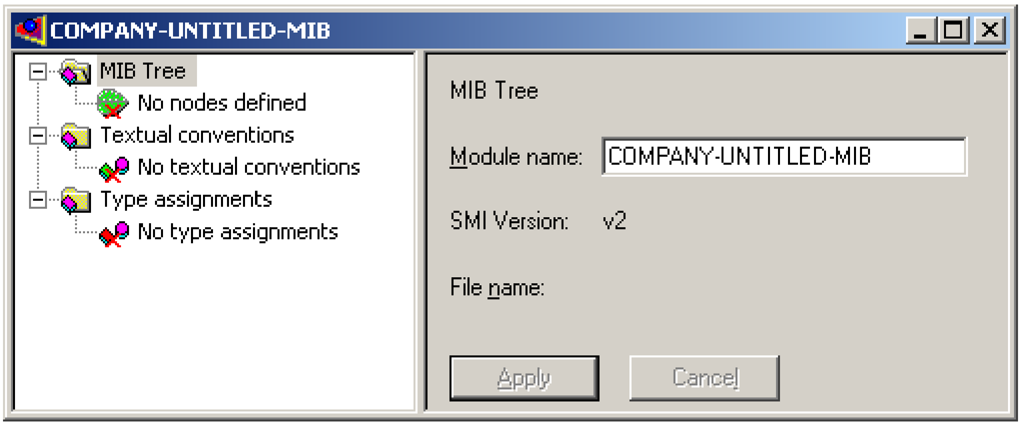
Note: The components (that represent SMI constructs for building MIB modules) available in the Components window differ for SMIv1 and SMIv2 MIB modules. If you open a SMIv1 MIB module or create a new MIB module and choose the SMIv1 version, the Components window will list only component available in the SMIv1 (i.e., OBJECT IDENTIFIER, OBJECT-TYPE, TRAP-TYPE and Type Assignment component). Similarly, if you open a SMIv2 MIB module or create a new MIB module and choose the SMIv2 version, the Components window will list the components available in the SMIv2.

1. Select the ***File / New / SMIv1 MIB Module*** or the ***File / New / SMIv2 MIB Module*** command, depending on the SMI syntax version you want to use to design a new MIB module.

   Tip: If you wish to convert a SMIv2 MIB into SMIv1 MIB or vice-versa, use the **Tools** / ***SMI Conversion*** command. For more information on how to convert a MIB module, read the Converting MIB Modules section of this manual.

2. A new MIB Builder window appears. The window contains an empty (new) MIB Module called `COMPANY-UNTITLED-MIB` (Figure 10) and in the Properties window panel you can see its default properties.

3. Into the ***Module name*** input line in the Properties window panel, type the name of the new module (e.g., `MGSOFT-BEEPER-MIB`) and click the ***Apply*** button.



Module name: COMPANY-UNTITLED-MIB

SMI Version: v2

File name:

Figure 10: MIB Builder window

5.2.2 Importing Objects to MIB Module

Visual MIB Builder lets you import nodes from other MIB modules into the MIB module you are currently building. As Visual MIB Builder conforms to SMIv1 and SMIv2 specifications and ASN.1 rules for MIB definitions, MIB objects from any MIB module following this standard can be imported. However, you should always import nodes from MIB modules that are of the same SMI version as the module you are building. While MIB Builder lets you import all nodes from SMIv1 MIBs into SMIv2 MIBs (except the TRAP-TYPE nodes), you cannot import nodes from SMIv2 MIBs into SMIv1 MIB modules (other than OBJECT-IDENTIFIER and OBJECT-TYPE nodes).

When building more complex MIB modules or when defining textual conventions that are supposed to be widely used, it is very common to create a separate MIB module that contains only textual conventions and then import them into the target module(s). However, note also that standard SMIv1 and SMIv2 MIB modules already contain some standard object definitions, which can be imported into any MIB module you are building or editing.

Tip:

Standard SMIv1 MIBs are:

RFC1155-SMI
RFC1212

RFC1213

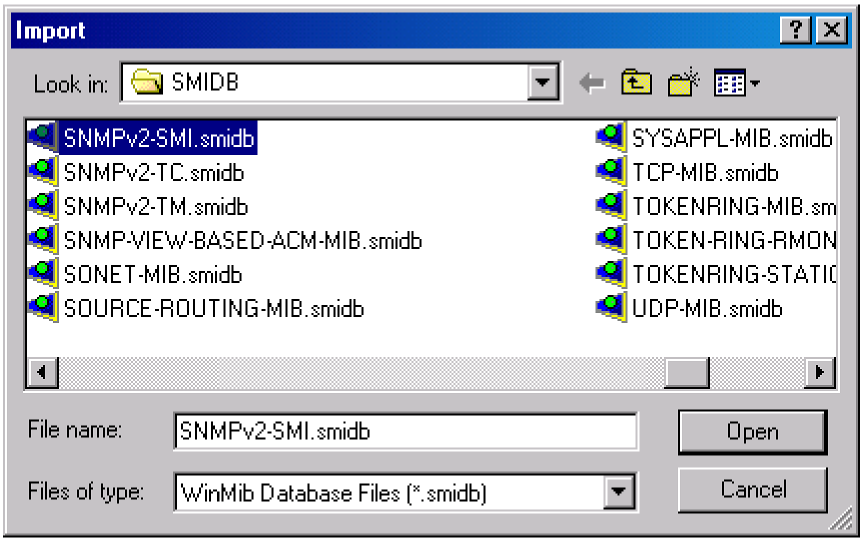
Standard SMIv2 MIBs are:

SNMPv2-SMI
SNMPv2-CONF
SNMPv2-TC

SNMPv2-MIB

Note: MIB modules, opened with the ***File*** / ***Import*** command cannot be edited or saved. They are used only for importing nodes into other MIB modules.

1. Select the ***File*** / ***Import*** command or click the ***Import*** toolbar button to open the Import dialog box (Figure 11).



Look in: SMIDB

File name: SNMPv2-SMI.smidb

Files of type: WinMib Database Files (*.smidb)

Figure 11: Import dialog box

Note: WinMib database files (SMIDB) are produced by MG-SOFT MIB Compiler.

2. From the ***Files of type*** drop-down list select the `WinMIB Database Files` or the `MIB Source Files`. All MIB files of the selected type will be displayed in the upper list. Choose one by double-clicking it. This will open a new MIB Database window displaying a MIB tree of the selected MIB module.

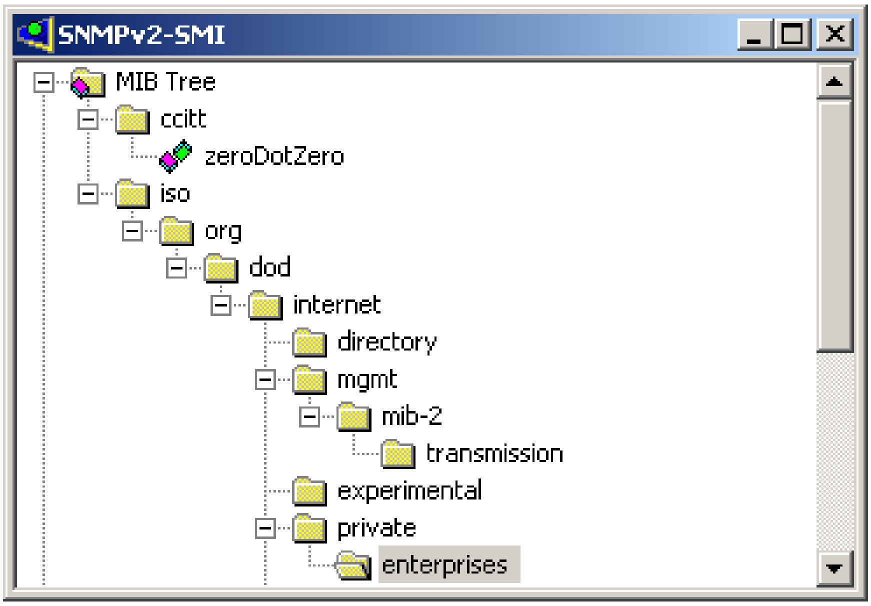


Figure 12: MIB Database window

3. To import a node into the MIB module, select a node in the MIB database window, and drag-and-drop it onto the MIB tree in the MIB Builder window.
4. Instead of using the drag-and-drop technique, you can also select the desired node in the MIB Database window and use the ***Edit*** / ***Copy*** command. This command will copy the selected node to the clipboard. Then select a node under which you wish to insert the copied node and use the ***Edit*** / ***Paste*** command.

Tip: To expand or collapse the MIB tree, select a node in the MIB tree and use the ***Edit*** / ***Expand*** or ***Edit*** / ***Collapse*** command. Instead, you can also use:

Expand toolbar button

Collapse toolbar button

5. If there is no parent node in the MIB Builder window that the imported node can link to, MIB Builder will ask you if you want to import all missing parent nodes. If you choose the ***Yes*** option, the current node and all of its parent nodes will be imported. If the current node has any child nodes, Visual MIB Builder will ask you if you also want to import all the child nodes as well.

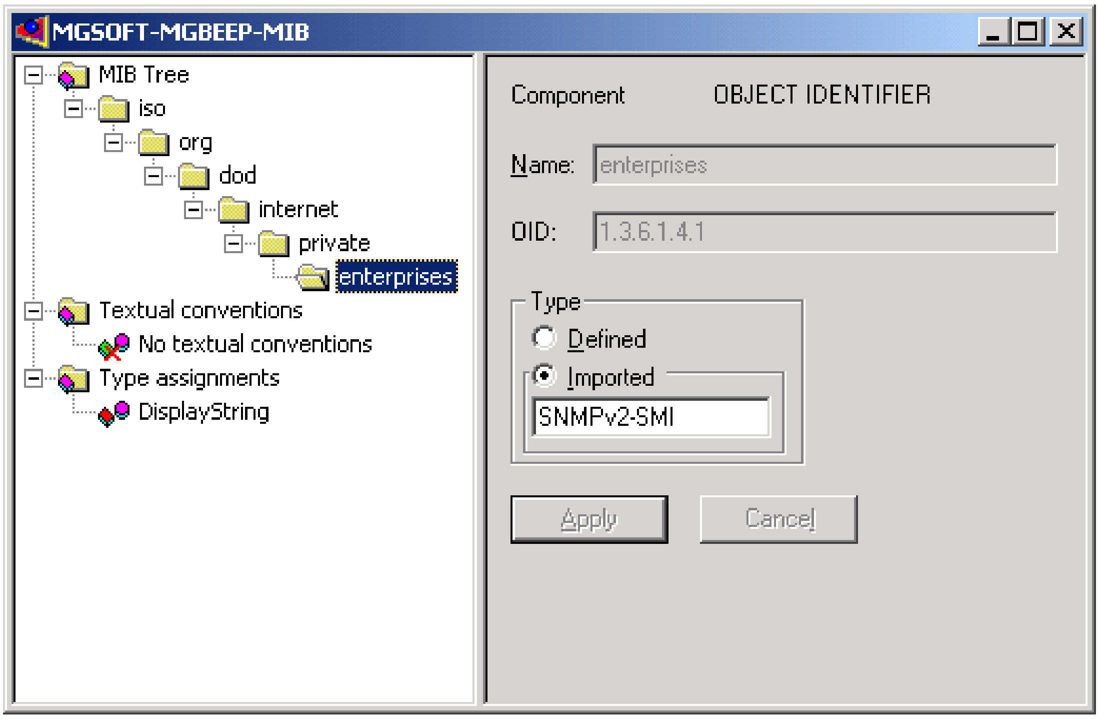
Note: In case the MIB Builder window already contains a node in the MIB Tree that has the same OID value as any of the nodes you wish to import, Visual MIB Builder will warn you that it cannot import the node.

6. Imported node and all its parent and child nodes (if you have selected this option) will appear in the MIB tree (Figure 13). The Properties window panel displays the properties of imported nodes in the read-only mode, since the properties of imported MIB objects cannot be edited.

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View** / **Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit** / **Remove*** command. When removing a parent node all its child nodes will be removed as well.

When you are done with importing nodes, you can start building the desired MIB tree by adding nodes from the Components window and defining their properties.



Component OBJECT IDENTIFIER

Name: enterprises

OID: 1.3.6.1.4.1

Type

○ Defined

● Imported

SNMPv2-SMI

Figure 13: MIB Builder window containing imported nodes

5.2.3 Adding OBJECT IDENTIFIER Constructs to MIB Module

OBJECT IDENTIFIER construct is used for assigning a name to an object identifier (OID). For example, the name `system` is assigned to the OID value `1.3.6.1.2.1.1`. Assigned name is used as a shorthand for referencing an OID value within a MIB module. The OBJECT IDENTIFIER construct is used for example when creating a new branch in the MIB tree (e.g., the `system` node identifies that the child nodes below this node include attributes of a system).

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View / Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit / Remove*** command. When removing a parent node all its child nodes will be removed as well.

1. From the Components window (***View / Components***) select the OBJECT IDENTIFIER component and drag-and-drop it to the desired position in the MIB tree displayed in the MIB Builder window. As you drag the node over the MIB tree, Visual MIB Builder will indicate where in the MIB tree the node will be inserted when you release the mouse button. There are two possible cases:
   - ❑ The target node is selected. That means that the node you are about to drop will be added as the last child node of the selected node.
   - ❑ A line is drawn between two nodes. That means that the node you are about to drop will be inserted between these two nodes.

2. Instead of using the drag-and-drop technique, you can also select a component in the Components window and use the ***Edit / Copy*** command. This command will place the selected node to the clipboard. In the MIB tree, select a node under which you wish to insert the copied node and use the ***Edit / Paste*** command. Although this action behaves identically as the drag-and-drop technique, there are some limitations. First, there is no visual feedback on whether the node can be inserted at the desired location, and second, the node can only be inserted as the last child node of the target node.

3. The new node has a default name and an automatically assigned object identifier value (OID) that corresponds to the location of the node in the MIB tree. If you want to change the default OID value, and thus reposition the node within the MIB tree, you can modify the last sub-identifier number in the ***OID*** input line.

> **Note:** Object identifier (OID) value is an ordered sequence of non-negative numbers usually written as a sequence of numbers (sub-identifiers) separated by a dot (.). OID values uniquely identify objects in MIB modules. For example, OID value assigned to the `sysUpTime` node is 1.3.6.1.2.1.1.3.

4. In the Properties window panel enter the name of the OBJECT-IDENTIFIER node into the ***Name*** input line. The name must start with lower case letter (e.g., `mgSamples`) and must be unique for the current MIB module.

> **Note:** SMIv1 allows the use of hyphens in MIB object names and does not impose any restrictions on the name length. SMIv2 disallows hyphens in MIB object names and restricts the length of names to at most 64 characters.

5. To add a comment to the node, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** views.
6. Click the ***Apply*** button.

5.2.4 Adding OBJECT-TYPE Constructs to MIB Module

The OBJECT-TYPE constructs are used to define *scalar*, *columnar*, *row* and *table objects*. These objects are called *object types* and are either a type of management information (*scalar* and *columnar objects*) or mechanisms for organizing related information (*table* and *row objects*).

The following guidelines hold for adding any *object type* (*scalar*, *table*, *row* or *columnar*) to a MIB.

Note: When adding a new *table object*, a *row object* and one *columnar object* are by default automatically added to the table node. To disable this function, uncheck the ***Automatically add row and columnar object*** checkbox in the Preferences dialog box | Builder tab (***View / Preferences***).

1. From the Components window (***View / Components***) select an OBJECT-TYPE component and drag-and-drop it onto the MIB tree in the MIB Builder window. As you drag the node over the MIB tree, Visual MIB Builder will indicate where in the MIB tree the node will be inserted when you release the mouse button. There are two possible cases:
   - ❑ The target node is selected. That means that the node you are about to drop will be added as the last child node of the selected node.
   - ❑ A line is drawn between two nodes. That means that the node you are about to drop will be inserted between these two nodes.
2. Instead of using the drag-and-drop technique, you can also select a component in the Components window and use the ***Edit / Copy*** command. This command will copy the selected node to the clipboard. In the MIB tree, select a node under which you wish to insert the copied node and use the ***Edit / Paste*** command. Although this action behaves identically as the drag-and-drop technique, there are some limitations. First, there is no visual feedback on whether the node can be inserted at the desired location, and second, the node can only be inserted as the last child node of the target node.

Tip: To remove a node from the MIB tree, select the node and use the ***Edit / Remove*** command. When removing a parent node all its child nodes will be removed as well.

3. When the node is added to the MIB tree, Visual MIB Builder automatically assigns the default properties to it, like the name, OID, syntax, etc. These properties are displayed and can be modified in the Properties window panel of the MIB Builder window when the node is selected. Further details on editing OBJECT-TYPE nodes can be found in the following sections.

Note 1: SMIv1 allows the use of hyphens in MIB object names and does not impose any restrictions on the name length. SMIv2 disallows hyphens in MIB object names and restricts the length of names to maximal 64 characters.

Note 2: **Object identifier (OID)** value is an ordered sequence of non-negative numbers usually written as a sequence of numbers (sub-identifiers) separated by a dot (.). OIDs uniquely identify objects in MIB modules. For example, OID value assigned to the `sysUpTime` MIB object is 1.3.6.1.2.1.1.3.

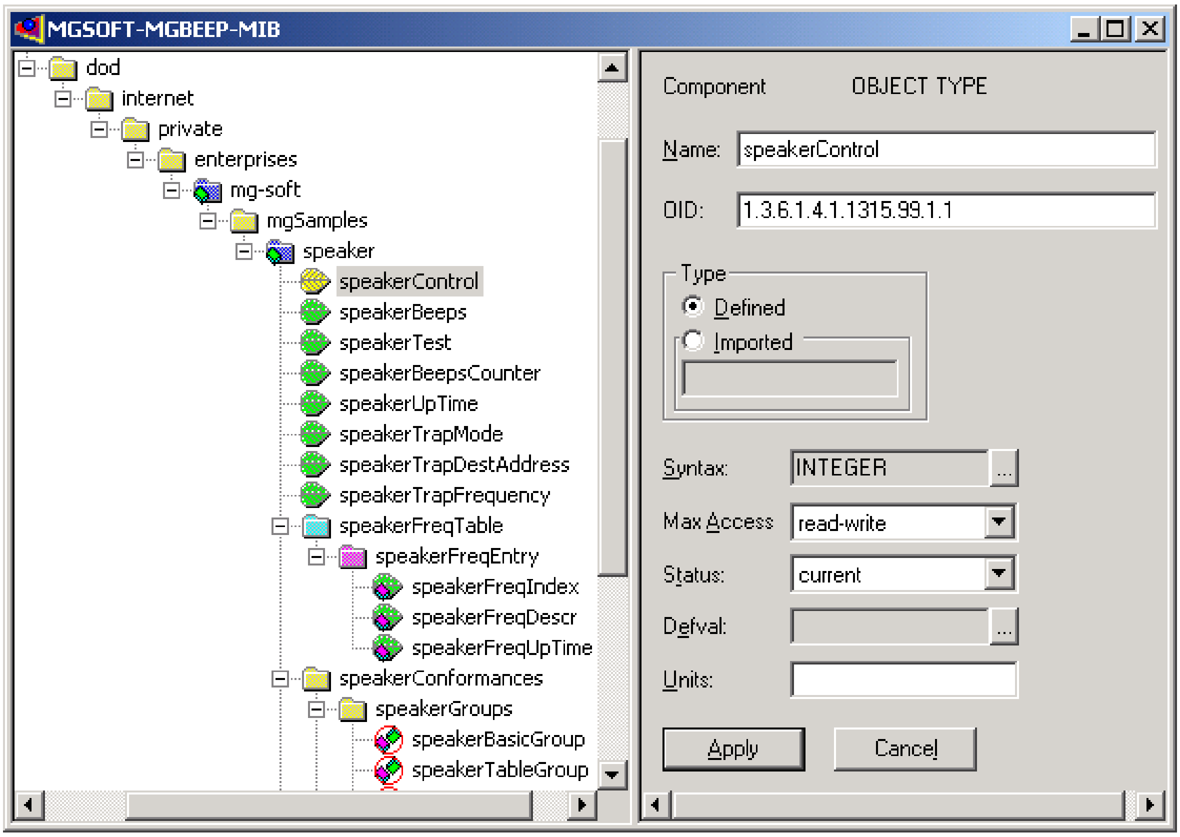
Scalar Objects

A *scalar object* has exactly one object instance, which can have one value. Therefore, scalar objects are used for defining management information that can take only one value (e.g., a managed system location). The name of the *scalar object instance* consists of the object name (or its OID) and the suffix `0` (e.g., the only instance of the `sysLocation` scalar object is `sysLocation.0`).

1. Once the *scalar* node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.
2. Into the ***Name*** input line, enter the name of the *scalar* node. The name must start with a lower case letter (e.g., `speakerControl`) and must be unique for the current MIB module.

Tip: For more information on how to add a *scalar object,* read the Adding OBJECT-TYPE Constructs to MIB Module section.

3. To set the ***Syntax*** definition, click the ***Browse*** button next to the input line. The Syntax Details dialog box appears (Figure 15); where you can choose from the base syntax types, and Type Assignments and textual-conventions that were imported or defined in this MIB module. Besides, you can specify sizes, ranges and enumerated values for the syntax types that allow sub-typing (refinement of the syntax).



Component OBJECT TYPE

Name: speakerControl

OID: 1.3.6.1.4.1.1315.99.1.1

Type
● Defined
○ Imported

Syntax: INTEGER

Max Access: read-write

Status: current

Defval:

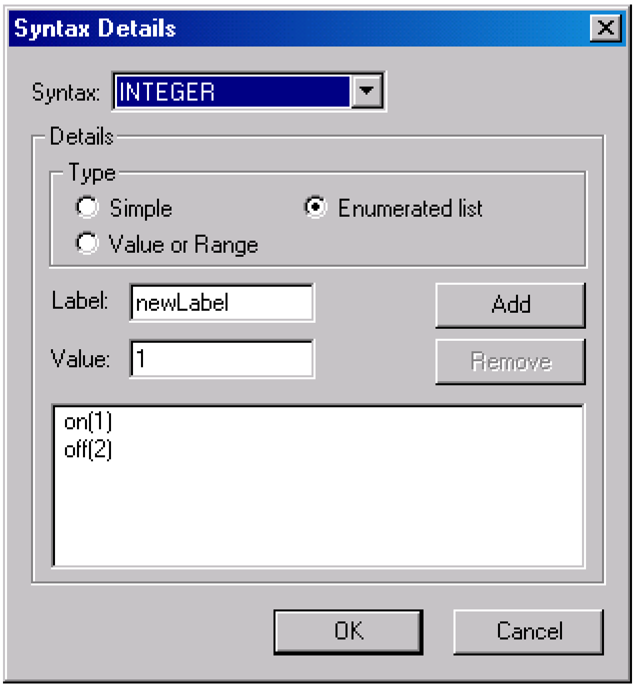
Units:

Figure 14: Properties window panel for scalar objects

4. From the ***Syntax*** drop-down list in the Syntax Details dialog box select the syntax type to be used for the target node.

5. Appearance of the ***Details*** frame changes according to the selected syntax type. Depending on the selected syntax type, the following options may be available:

   - ***Simple*** - If you select this option, no syntax refinement occurs.
   - ***Enumerated List*** – Lets you define a list of labels and corresponding values (e.g., `Label – on; Value - 1`) for the selected syntax type.



Syntax Details

Syntax: INTEGER

Details

Type

○ Simple ● Enumerated list

○ Value or Range

Label: newLabel

Value: 1

on(1)
off(2)

Figure 15: Syntax Details dialog box

   - ***Value or Range*** - Lets you specify one or more integer values (e.g., 1, 5, 10) and/or one or more integer ranges (30-50, 55-60) that the selected syntax type can take.
   - **Size or Size Range -** Lets you specify one or more string length sizes (e.g., 1, 5, 10) and/or one or more string length ranges (30-50, 55-60) that limit the string size of the selected syntax type.

6. From the ***Access*** drop-down list (***Max. Access*** in SMIv2), select the value for the access clause, which determines the allowed access to the object.

   - `not-accessible` – The object is not accessible for any kind of SNMP operation. This value is not allowed for scalar objects.
   - `accessible-for-notify` (only in SMIv2) – The object is accessible only for event report operations.
   - `read-only` – The object is accessible for SNMP retrieval and event report operations.
   - `write-only` (only in SMIv1) – The object may be an operand in SNMP retrieval, modifying and event report operations. This value should not be used, since it is obsolete according to the valid SMI specification.

- ❑ `read-write` – The object may be an operand in SNMP retrieval, modifying and event report operations.
- ❑ `read-create` (only in SMIv2) – The object may be an operand in SNMP retrieval, modifying and event report operations. Besides, the object may be an operand in SNMP Set operations that create a new object instance. This access value is not allowed for scalar objects.

Note: Every *object type* that has the ***Max Access*** clause value set to any other value than `not-accessible` must be a member of at least one object group. Read the OBJECT-GROUP Constructs section for more information.

7. From the ***Status*** drop-down list, select the value for the status clause for this object.
   - ❑ `Mandatory` (only in SMIv1) - indicates that the object definition is valid and that the object must be implemented.
   - ❑ `Current` (only in SMIv2) - indicates that the object definition is valid.
   - ❑ `Deprecated` - indicates that the object definition is valid in certain circumstances, but has been replaced by another. In SMIv1 MIBs, this value also denotes that the implementation of this object is required for conformance.
   - ❑ `Obsolete` - indicates that the object definition is not valid (and in SMIv1 that the object should not be implemented).
   - ❑ `Optional` (only in SMIv1) - indicates that the object definition is valid and may or may not be implemented.
8. The ***Defval*** (default value) should not be specified for *scalar objects*. The DEFVAL clause is intended for use only for columnar objects in tables that allow row creation via SNMP Set operation.

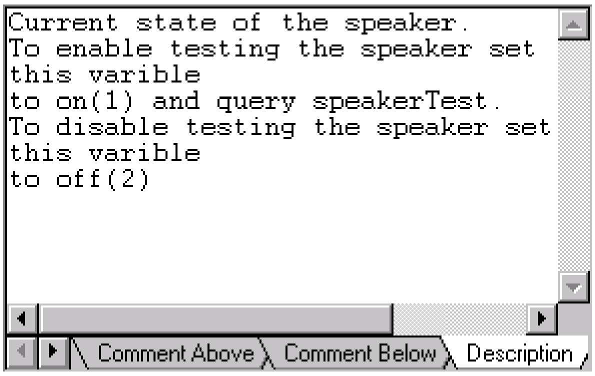
Note: Although the DEFVAL clause should not be used for “regular” scalar objects, SMIv2 specification permits its use in a special case, i.e., for scalar objects that are dynamically created by agents. For more information about specifying the ***Defval*** values, see the Columnar Objects section of this manual.

9. Into the ***Units*** input line (only in SMIv2 MIBs), you can optionally enter a textual description of the units associated with the data type (e.g., for an object whose values represent time measured in seconds the unit `seconds` can be specified).
10. To add a comment to the *scalar object*, open the Additional Properties dialog box (***View*** / ***Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** views.
11. To add a description of the *scalar object* or any information that is needed to understand the object or module, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Description*** tab (Figure 16). The text can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands.

Tip: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands. For more information on spell checking, read the Spell section of this manual.

12. To add a reference, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is optional for all constructs.

13. After setting all parameters, click the ***Apply*** button for the changes to take effect.

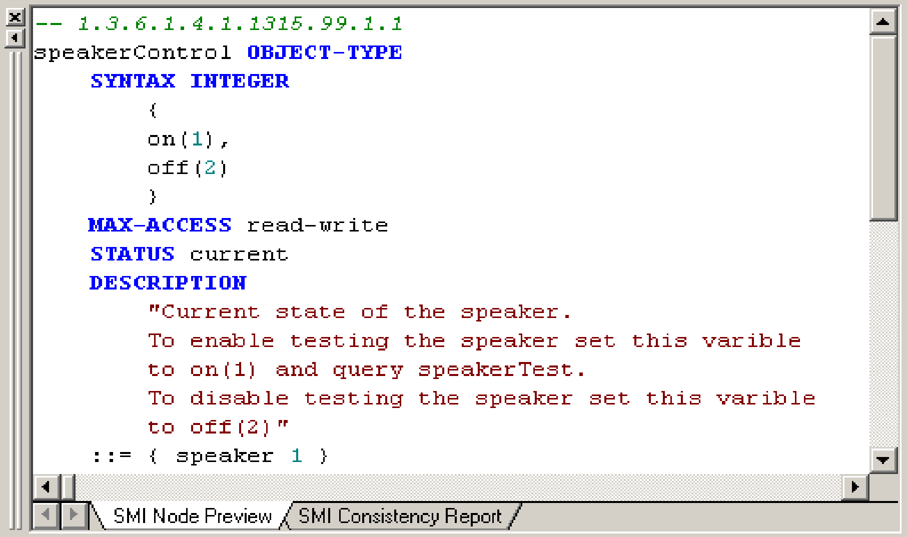


```
Current state of the speaker.
To enable testing the speaker set
this varible
to on(1) and query speakerTest.
To disable testing the speaker set
this varible
to off(2)
```

Figure 16: Description tab

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View*** / ***Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit*** / ***Remove*** command. When removing a parent node all its child nodes will be removed as well.



```
-- 1.3.6.1.4.1.1315.99.1.1
speakerControl OBJECT-TYPE
    SYNTAX INTEGER
        {
        on(1),
        off(2)
        }
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "Current state of the speaker.
        To enable testing the speaker set this varible
        to on(1) and query speakerTest.
        To disable testing the speaker set this varible
        to off(2)"
    ::= { speaker 1 }
```

Figure 17: Viewing SMI definition of a MIB object

Table Objects

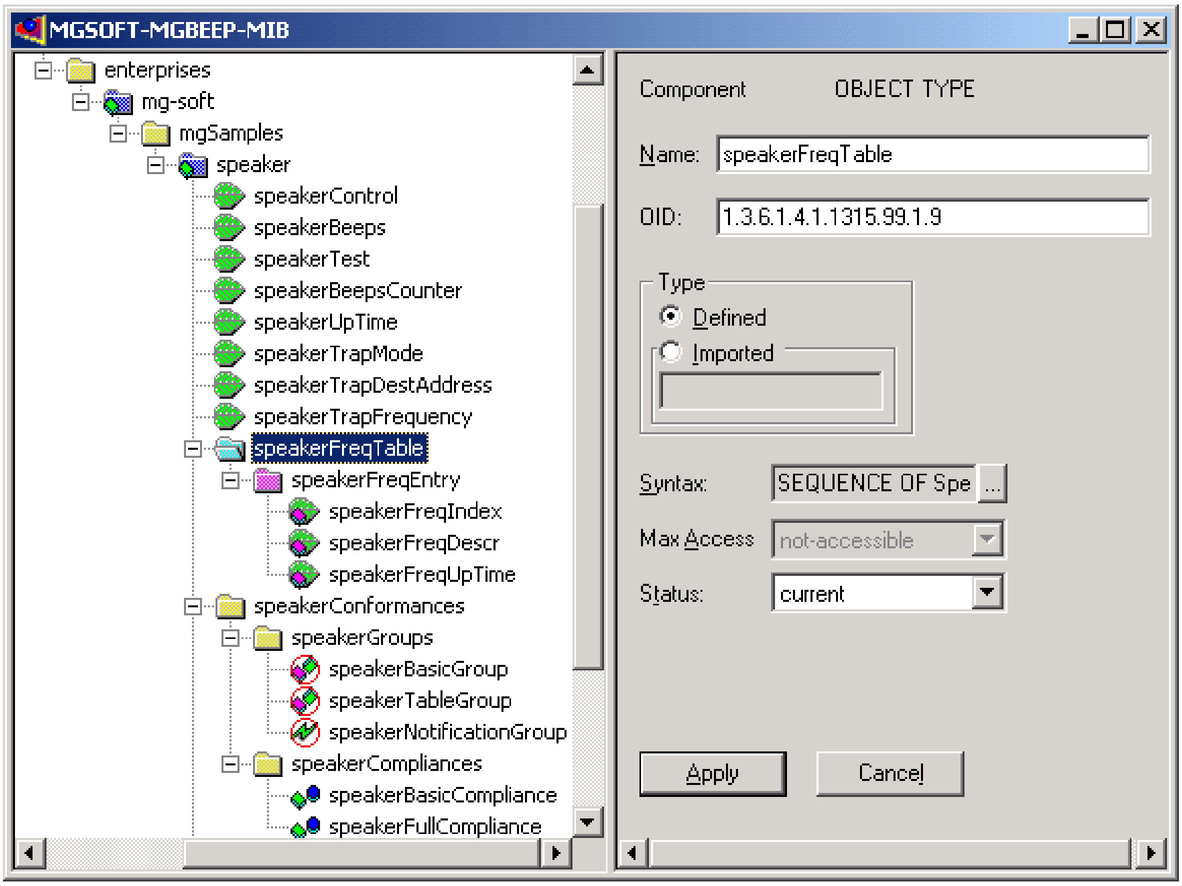
A group of related management information can be organized in an imaginary, tabular structure. Such tabular structures are called (conceptual) tables. A table consists of a *table object, a row object*, and one or more *columnar objects*. In contrast to *scalar objects,* which have only one instance, c*olumnar objects* can have multiple instances. Instances of columnar objects are identified by the table index. Each table has an indexing scheme that is defined either with one or more index columns from that table or by column(s) from another table (in SMIv2).

Note: When adding a new *table object*, a *row object* and one *columnar object* are by default automatically added to the table node. To disable this function, uncheck the ***Automatically add row and columnar object*** checkbox in the Preferences dialog box | Builder tab (***View / Preferences***).

Creating a *table* takes three steps: defining a *table object*, defining a *row object* and defining one or more *columnar objects*. For more information on defining individual *object types*, read this and the following sections of this manual.

1. Once the *table* node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.

Tip: For more information on how to add a *table object*, read the Adding OBJECT-TYPE Constructs to MIB Module section of this manual.



Component OBJECT TYPE

Name: speakerFreqTable

OID: 1.3.6.1.4.1.1315.99.1.9

Type
● Defined
○ Imported

Syntax: SEQUENCE OF Spe ...

Max Access: not-accessible

Status: current

Figure 18: Properties window panel for table objects

2. Into the ***Name*** input line, enter the name of the *table* node. By convention, the name must start with a lower case letter and must end with a suffix `Table` (e.g., `speakerFreqTable`). The name must be unique for the current MIB module.

3. The ***Syntax*** value is set automatically as `SEQUENCE OF`…This is a special value of the SYNTAX, which indicates that a *table object* is being defined. After you specify the *row object*, the syntax will change and conform to the *row object*.

4. The value for the ***Access*** clause is automatically set to `not-accessible` and cannot be modified since a *table* cannot be accessed with SNMP operations.

5. From the ***Status*** drop-down list, select the value for the status clause.

   - `Mandatory` (only in SMIv1) - indicates that the object definition is valid and that the object must be implemented.
   - `Current` (only in SMIv2) - indicates that the object definition is valid.
   - `Deprecated` - indicates that the object definition is valid in certain circumstances, but has been replaced by another. In SMIv1 MIBs, this value also denotes that the implementation of this object is required for conformance.
   - `Obsolete` - indicates that the object definition is not valid (and in SMIv1 that the object should not be implemented).
   - `Optional` (only in SMIv1) - indicates that the object definition is valid and may or may not be implemented.

6. To add a comment to the *table object*, open the Additional Properties window (***View*** / ***Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.

7. To add a description of the *table object* or any information that is needed to understand the object or module, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Description*** tab (Figure 16). The text can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands.

8. To add a reference, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is not obligatory for OBJECT-TYPE constructs.

9. After setting all parameters, click the ***Apply*** button or the changes will not take effect.

Tip 1: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands. For more information on spell checking, read the Check Spelling section of this manual.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit*** / ***Remove*** command. When removing a parent node all its child nodes will be removed as well.

Tip: To view SMI definition of the selected node, open the Node Preview window (***View*** / ***Node Preview***) and select the ***SMI Node Preview*** tab.

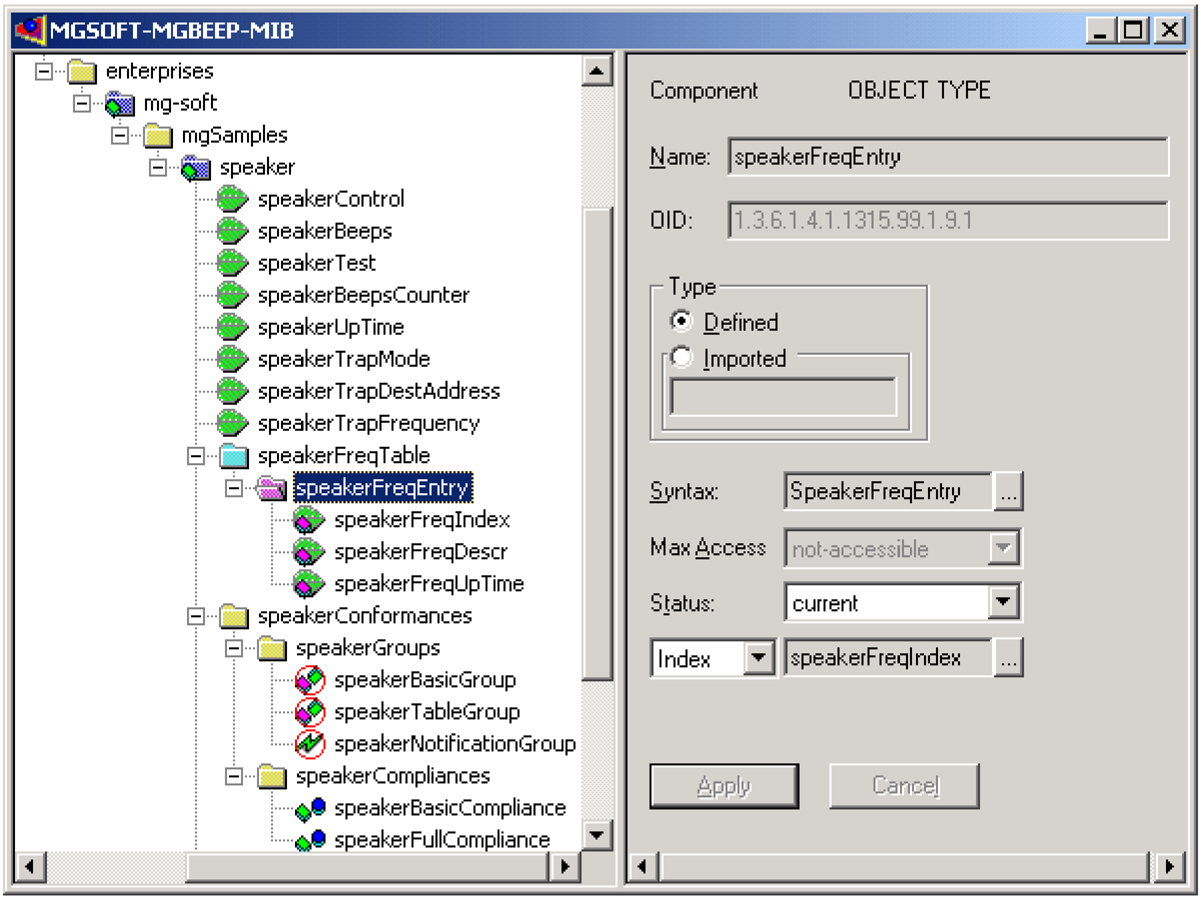
Row Objects

Row object types are mechanisms for organizing related objects into tables. A *row object* contains one or more *columnar objects* and defines the *indexing scheme* for the table either by using one or more columns from the same table or by using one or more columns from another table. The first approach involves use of the INDEX clause, while the second approach employs either the INDEX or the AUGMENTS clause (latter is available in SMIv2 only). The INDEX or AUGMENTS clause determines how rows (or more precisely, instances of columnar objects) in the table are identified.

Note: SNMP operations can retrieve or modify values of *scalar* and *columnar object instances* only. *Table* and *row objects* cannot be directly accessed by SNMP operations.

1. Once the *row* node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.

Tip: For more information on how to add a *row object*, read the Adding OBJECT-TYPE Constructs to MIB Module section of this manual.



Component OBJECT TYPE

Name: speakerFreqEntry

OID: 1.3.6.1.4.1.1315.99.1.9.1

Type
● Defined
○ Imported

Syntax: SpeakerFreqEntry ...

Max Access: not-accessible

Status: current

Index: speakerFreqIndex ...

Figure 19: Properties window panel for row objects

2. The name of a *row object* is the same as the name of the table, where the suffix `Table` is replaced by the suffix `Entry` (e.g., if a table is named `speakerFreqTable`, then the *row* is named `speakerFreqEntry`).

> **Note:** You cannot change the name of the *row object.* However, you can change the name of the *table object* containing the *row object*, which will automatically update the name of the *row object* and the SYNTAX clause for both objects.

3. The ***Syntax*** value is set automatically and cannot be changed.
4. The value for the ***Access*** clause is automatically set to `not-accessible` and cannot be modified since table rows cannot be accessed with SNMP operations.
5. From the ***Status*** drop-down list, select the value for the STATUS clause. Note that the value of the STATUS clause must be the same as the value in the STATUS clause for the *table object*.
6. Next, you have to specify value for either INDEX or AUGMENTS clause (the AUGMENTS clause is available only in SMIv2). First select either `Index` or `Augment` from the drop-down list, depending on which clause you wish to use. By clicking the ***Browse*** button ('...') next to the ***Index (Augment)*** drop-down list, the Index (Augment) Selection dialog box appears (Figure 20).

> **Note:** The ***INDEX / AUGMENTS*** clause specifies how (conceptual) rows are indexed in the table.
>
> The ***INDEX*** clause lists the ordered index items for a *table*. Usually, the index items are names of *columnar objects* in the *table* that have the Access/Max. access clause value set to `not-accessible`.
>
> The ***AUGMENTS*** clause documents a special relationship between two tables, where one table “expands” the other. The item specified in the AUGMENTS clause is a *row object* of another *table*. By specifying the AUGMENTS clause, indexing scheme from another (base) table is used for identifying rows in the current (augmenting) table. The AUGMENTS clause may only be used when for every (conceptual) row in the base table there is a corresponding row in the augmenting table, which is identified by the same indices, and when the number of rows in both tables are identical (rows in the base table are mapped one-to-one to the rows in the augmenting table).

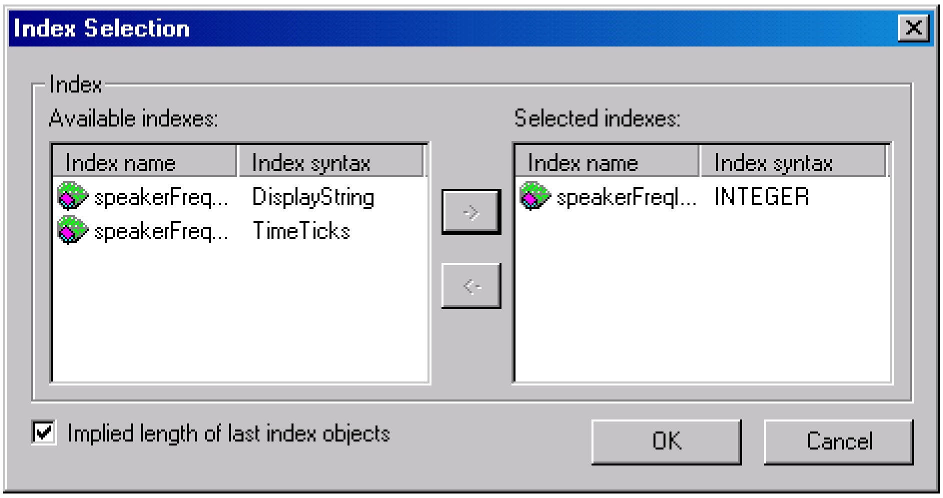
7. The ***Available indexes*** (***Available nodes for augmenting***) list displays all possible index items (items available for augmenting) for this entry. Select item(s) from the ***Available*** list and click the right arrow button to add an index (augment). Selected index (augment) will appear in the ***Selected*** list.
8. To remove an index (augment) item, select it from the ***Selected*** list and click the left arrow button, or drag the selected node from the ***Selected*** onto the ***Available*** list.

> **Tip**: To add an index (augment) select the item from the ***Available*** list and drag-and-drop it onto the ***Selected*** list.

9. To rearrange the order in which the indexes will appear in the generated MIB Module definition file, select the node you wish to move in the ***Selected*** list and drag-and-drop it onto the desired place. As you drag the node, you will get visual feedback on where the dragged node will be placed when you release the mouse button.
10. If the ***Implied length of last index objects*** checkbox (in SMIv2 only) at the bottom of the Index Selection dialog box is checked, the last object in the Selected Indexes list

box is specified as IMPLIED. The IMPLIED keyword can only be used if the last index object has a variable-length syntax that cannot have a value of zero-length.

11. Click the ***OK*** button to close the window and save the changes.



Available indexes:

Index name	Index syntax
speakerFreq...	DisplayString
speakerFreq...	TimeTicks

Selected indexes:

Index name	Index syntax
speakerFreqI...	INTEGER

☑ Implied length of last index objects

Figure 20: Index (Augment) Selection dialog box

12. To add a comment to the *row object*, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.

13. To add a description of the *row object* or any information that is needed to understand the object or module, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Description*** tab (Figure 16). The text can be checked for spelling errors by using the ***Tools / Check spelling*** commands.

Tip: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools / Check spelling*** commands. For more information on spell checking, read the Spell section of this manual.

14. To add a reference, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is optional for OBJECT-TYPE constructs.

15. After setting all parameters, click the ***Apply*** button in the Properties window panel or the changes will not take effect.

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View / Node Preview***) and select the ***SMI Node Preview*** tab.

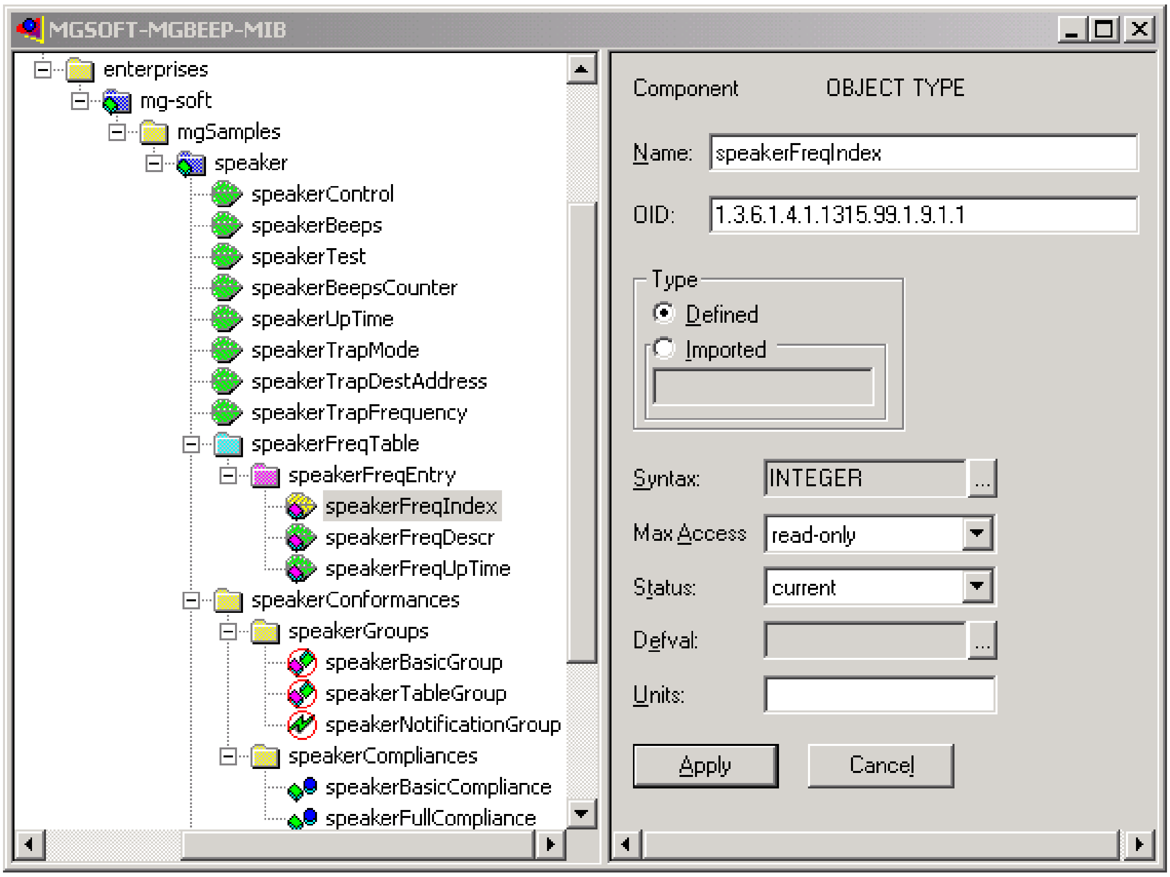
Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit / Remove*** command. When removing a parent node all its child nodes will be removed as well.

Columnar Objects

Columnar objects can have zero, one, or multiple instances. As the SMI requires, all *columnar objects* must be organized into tables.

1. Once the *columnar node* is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.

> **Tip:** For more information on how to add a *columnar object*, read the Adding OBJECT-TYPE Constructs to MIB Module section of this manual.



Component OBJECT TYPE

Name: speakerFreqIndex

OID: 1.3.6.1.4.1.1315.99.1.9.1.1

Type
● Defined
○ Imported

Syntax: INTEGER

Max Access: read-only

Status: current

Defval:

Units:

Figure 21: Properties window panel for columnar objects

2. Into the ***Name*** input line, enter the name of the *columnar node*. The name must start with a lower case letter (e.g., `speakerFreqIndex`) and must be unique for the current MIB module. By default, the first part of the name is automatically given considering the name of the table that contains this column.

> **Note:** At least one column of the *table* should be defined as the Index column.

> **Tip:** To disable this option, uncheck the ***Add table prefix to columnar objects*** checkbox in the MIB Builder Preferences dialog box | Builder tab.

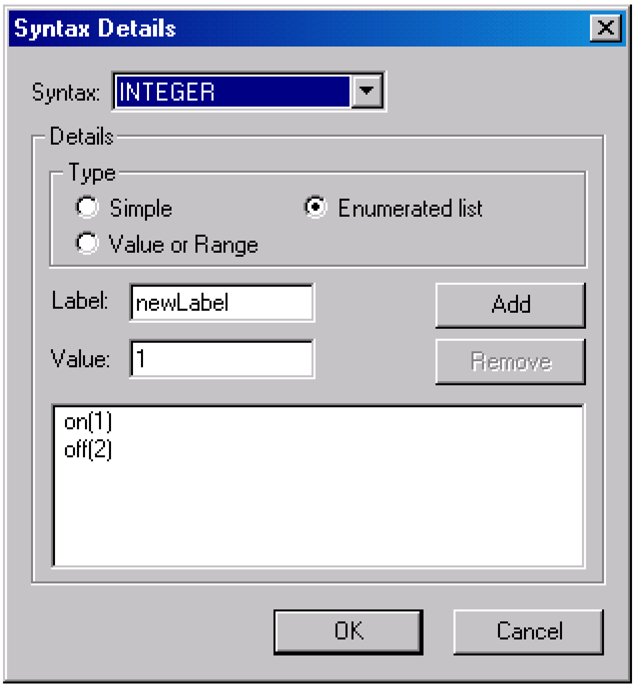
3. To configure the ***Syntax*** definition, click the ***Browse*** button next to the input line. The Syntax Details dialog box appears (Figure 15); where you can choose from the base syntax types, as well as from the Type Assignments and textual-conventions that were imported or defined in this MIB module. Besides, you can specify sizes, ranges and enumerated values for the syntax types that allow this type of sub-typing (refinement of the syntax).

4. From the ***Syntax*** drop-down list in the Syntax Details dialog box select the syntax type to be used for the target node.

5. Appearance of the ***Details*** frame changes according to the selected syntax type. Depending on the selected syntax type, the following options may be available:

   - ***Simple*** - If you select this option, no syntax refinement occurs. If the ***Defval*** parameter is set, the syntax will be formatted as specified in the ***Defval*** clause.

   - ***Enumerated List*** – Lets you define a list of labels and corresponding values (e.g., `Label – on; Value - 1`) for the selected syntax type.



Syntax Details

Syntax: INTEGER

Details

Type

○ Simple ● Enumerated list

○ Value or Range

Label: newLabel

Value: 1

on(1)
off(2)

Figure 22: Syntax Details dialog box

   - ***Value or Range*** - Lets you specify one or more integer values (e.g., 1, 5, 10) and/or one or more integer ranges (30-50, 55-60) that the selected syntax type can take.

   - **Size or Size Range -** Lets you specify one or more string length sizes (e.g., 1, 5, 10) and/or one or more string length ranges (30-50, 55-60) that limit the string size of the selected syntax type.

6. From the ***Access*** drop-down list (***Max. Access*** in SMIv2), select the value for the access clause, which determines the allowed access to the object.

   - `not-accessible` – The object is not accessible for any kind of SNMP operation. SMIv2 requires this value be used for index columnar objects.

- ❑ `accessible-for-notify` (only in SMIv2) – The object is accessible only for event report operations.
- ❑ `read-only` – The object is accessible for SNMP retrieval and event report operations.
- ❑ `write-only` (only in SMIv1) – The object may be an operand in SNMP retrieval, modifying and event report operations. This value should not be used, since it is obsolete according to the valid SMI specification.
- ❑ `read-write` – The object may be an operand in SNMP retrieval, modifying and event report operations.
- ❑ `read-create` (only in SMIv2) – The object may be an operand in SNMP retrieval, modifying and event report operations. Besides, the object may be an operand in SNMP Set operations that create a new object instance.

Note: Every *object type* that has the `Max. Access` clause value set to any other value than `not-accessible` must be a member of at least one object group (in SMIv2 MIB modules). Read the OBJECT-GROUP Constructs section for more information.

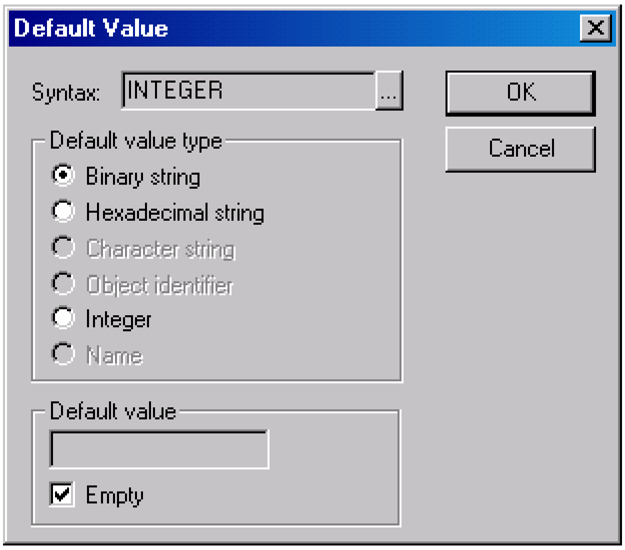
7. From the ***Status*** drop-down list, select the value for the status clause for this object.

Note: The value of the STATUS clause for the columnar object must be consistent with the value of the STATUS clause in the containing table.

- ❑ `Mandatory` (only in SMIv1) - indicates that the object definition is valid and that the object must be implemented.
- ❑ `Current` (only in SMIv2) - indicates that the object definition is valid.
- ❑ `Deprecated` - indicates that the object definition is valid in certain circumstances, but has been replaced by another. In SMIv1 MIBs, this value also denotes that the implementation of this object is required for conformance.
- ❑ `Obsolete` - indicates that the object definition is not valid (and in SMIv1 that the object should not be implemented).
- ❑ `Optional` (only in SMIv1) - indicates that the object definition is valid and may or may not be implemented.

8. To set the default value type for the object, click the ***Browse*** button next to the ***Defval*** input line. The Default value dialog box appears.

Note: The DEFVAL clause should be used only for columnar objects in tables that allow row creation via SNMP Set operation.



Default Value

Syntax: INTEGER

Default value type

● Binary string

○ Hexadecimal string

○ Character string

○ Object identifier

○ Integer

○ Name

Default value

☑ Empty

Figure 23: Default Value dialog box

9. By clicking the ***Browse*** button next to the ***Syntax*** input line, you can view the syntax details. In the Default Value dialog box, you can also set the formatting of the value and the default value that the instances of this columnar object will have when a new row in the table is created and the value for this instance is not specified in the SNMP Set operation that creates a new row.

10. If the ***Empty*** checkbox is checked, an empty DEFVAL clause will be generated.

11. After setting all parameters in the Default Value dialog box, click the ***OK*** button to save the settings and close the dialog box.

12. Into the ***Units*** input line (only in SMIv2 MIBs), enter an optional textual description of the units associated with the data type (e.g., `inches`).

13. To add a comment to the *scalar object*, open the Additional Properties window (***View*** / ***Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.

14. To add a description of the *columnar object* or any information that is needed to understand the object or module, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Description*** tab (Figure 16). The text can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands.

15. To add a reference, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is not obligatory for OBJECT-TYPE constructs.

16. After setting all parameters, click the ***Apply*** button in the Properties window panel or the changes will not take effect.

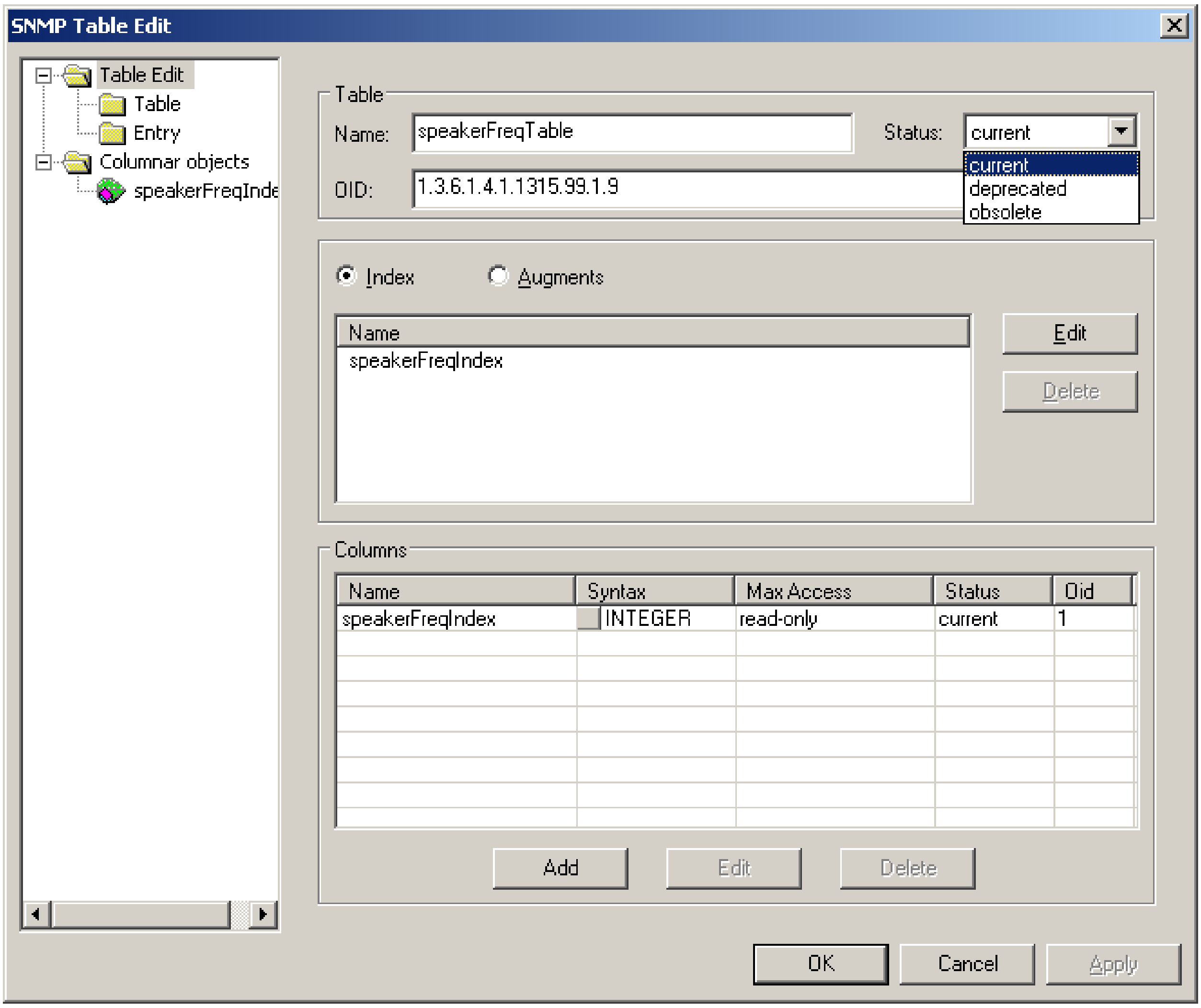
> **Tip 1:** To view SMI definition of the selected node, open the Node Preview window (***View*** / ***Node Preview***) and select the ***SMI Node Preview*** tab.

Adding or Editing Entire Table in SNMP Table Edit Dialog Box

The SNMP Table Edit dialog box provides a convenient interface for adding and/or editing the entire table definition (i.e., a table object, a row object and all columnar objects in the given SNMP table).

About the SNMP Table Edit dialog box

To open the SNMP Table Edit dialog box and view the properties of an existing table, right-click a table, a row or a columnar node in the MIB tree and choose the ***Edit Table*** pop-up command. Alternatively, to create a new table and display its properties in the SNMP Table Edit dialog box, select an OBJECT IDENTIFIER or OBJECT-IDENTITY node under which you wish to create a new SNMP table and choose the ***Add Table*** pop-up command.



Name: speakerFreqTable

Status: current

current
deprecated
obsolete

OID: 1.3.6.1.4.1.1315.99.1.9

● Index ○ Augments

Name
speakerFreqIndex

Name	Syntax	Max Access	Status	Oid
speakerFreqIndex	INTEGER	read-only	current	1

Figure 24: SNMP Table Edit dialog box – Table Edit panel

The left panel of the SNMP Table Edit dialog box contains a hierarchical navigation tree. Clicking an entry in the navigation tree displays its properties in the right section of the SNMP Table Edit dialog box, where the properties can be edited.

The following navigation entries and corresponding panels (views) exist in the SNMP Table Edit dialog box:

Table Edit	– For editing the main properties of the entire table (including adding, editing and removing columnar objects)
├─ Table	– For editing the *table object* properties
└─Entry	– For editing the *row object* properties
Columnar Objects	– Entry under which the columnar objects are listed
├─ Columnar Object 1	– For editing the properties of the *columnar object 1*
├─ …	
└─ Columnar Object N	– For editing the properties of the *columnar object N*

The `Table Edit`, `Table`, `Entry` and `Columnar objects` entries in the navigation tree are static elements, i.e., they are always present in the navigation tree, while the sub-entries of the `Columnar objects` entry are dynamic, i.e., they appear and disappear when adding and removing columnar objects to/from the table.

Table Edit panel

1. Click the ***Table Edit*** entry in the navigation tree to view or edit the basic parameters for the entire table (Figure 24).
2. Into the ***Name*** input line enter a name for the table. By convention, the name should start with a lower case letter and should end with a suffix `Table` (e.g., `speakerFreqTable`). The name must be unique for the current MIB module. By changing the table name and applying the change, the table entry node name is also changed accordingly (i.e., carrying the same name prefix as the table node name).
3. From the ***Status*** drop-down list, select the value for the status clause for this object.
   - `Mandatory` (only in SMIv1) - indicates that the object definition is valid and that the object must be implemented.
   - `Current` (only in SMIv2) - indicates that the object definition is valid.
   - `Deprecated` - indicates that the object definition is valid in certain circumstances, but has been replaced by another. In SMIv1 MIBs, this value also denotes that the implementation of this object is required for conformance.
   - `Obsolete` - indicates that the object definition is not valid (and in SMIv1 that the object should not be implemented).
   - `Optional` (only in SMIv1) - indicates that the object definition is valid and may or may not be implemented.

> **Note:** After changing the status value and clicking the ***OK*** or the ***Apply*** button, Visual MIB Builder prompts you with a dialog box asking if you would also like to change the status of the row object and columnar objects. Clicking the ***Yes*** button will set the status of the table object itself as well as the status of all other objects within the table to a new value.

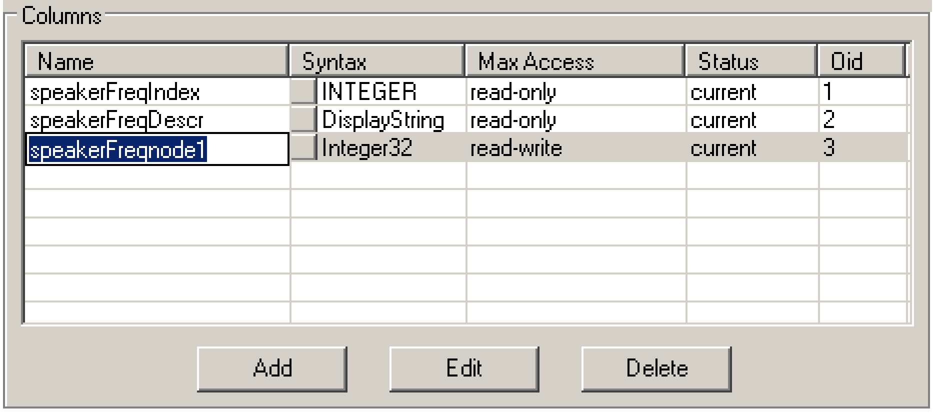
4. The ***OID*** input line displays the OID value assigned to the table object in form of an ordered sequence of non-negative numbers (sub-identifiers) separated by dots (.).

Only the last sub-identifier in this input line can be modified. Modifying the OID value re-positions table in the MIB tree and automatically adjusts OID values of the subordinated row and columnar objects.

5. Next, you need to define the table indexing scheme by specifying value for either INDEX or AUGMENTS clause (the AUGMENTS clause is available only in SMIv2). First, select either ***Index*** or ***Augments*** radio button, depending on which clause you wish to use. Then, click the ***Edit*** button next to the Index/Augments checkboxes to create a new or edit the existing index/augments list.

> **Note:** The ***INDEX / AUGMENTS*** clause specifies how (conceptual) rows are indexed in the table.
>
> The ***INDEX*** clause lists the ordered index items for a *table*. Usually, the index items are names of *columnar objects* in the table that have the Access/Max. access clause value set to `not-accessible`.
>
> The ***AUGMENTS*** clause documents a special relationship between two tables, where one table “expands” the other. The item specified in the AUGMENTS clause is a *row object* of another *table*. By specifying the AUGMENTS clause, indexing scheme from another (base) table is used for identifying rows in the current (augmenting) table. The AUGMENTS clause may only be used when for every (conceptual) row in the base table there is a corresponding row in the augmenting table, which is identified by the same indices, and when the number of rows in both tables are identical (rows in the base table are mapped one-to-one to the rows in the augmenting table).

6. The Index (Augments) Selection dialog box appears (Figure 20). The ***Available indexes*** (***Available nodes for augmenting***) list displays all possible index items (items available for augmenting) for this entry. Select item(s) from the ***Available*** list and click the right arrow button to add an index (augment). Selected index (augment) item will appear in the ***Selected*** list.
7. To remove an index (augment) item, select it from the ***Selected*** list and click the left arrow button, or drag the selected node from the ***Selected*** onto the ***Available*** list.
8. To rearrange the order in which the indexes will appear in the generated MIB Module definition file, select the node you wish to move in the ***Selected*** list and drag-and-drop it onto the desired position. As you drag the node, you will get visual feedback on where the dragged node will be placed when you release the mouse button.
9. If the ***Implied length of last index objects*** checkbox (in SMIv2 only) at the bottom of the Index Selection dialog box is checked, the last object in the Selected Indexes list box is specified as IMPLIED. The IMPLIED keyword can only be used if the last index object has a variable-length syntax that cannot have a zero-length value.
10. Click the ***OK*** button to close the dialog box and apply the changes.
11. The ***Columns*** list displays all columnar objects in a table by listing their name, syntax, (max.) access, status and the last OID sub-identifier values (Figure 25).



Name	Syntax	Max Access	Status	Oid
speakerFreqIndex	INTEGER	read-only	current	1
speakerFreqDescr	DisplayString	read-only	current	2
speakerFreqnode1	Integer32	read-write	current	3

Figure 25: The Columns list in the SNMP Table Edit dialog box

- To add a new columnar object to the table, click the ***Add*** button. A new columnar object with the default properties will be added to the ***Columns*** list.
- To edit a property of the columnar object within the ***Columns*** list, double-click the relevant cell (i.e., name, syntax, access, status, OID) and modify the current value (Figure 25). To change the syntax property, click the square button in front of the displayed syntax value in the Syntax cell, which will open the standard Syntax Details dialog box from which you can select the desired syntax value.
- To edit columnar object properties other than those displayed in the ***Columns*** list, select the row that represents a columnar object in the ***Columns*** list and click the ***Edit*** button. The panel dedicated to the selected columnar object with all its properties will be displayed in the SNMP Table Edit dialog box.
- To delete a columnar object from the table, select the relevant row in the ***Columns*** list and click the ***Delete*** button.

12. Switch to another entry in the in the navigation tree or apply the changes and close the SNMP Table Edit dialog box by clicking the ***OK*** button.

Table panel

1. Click the ***Table Edit*** entry in the SNMP Table Edit navigation tree to display the Table panel and view or edit all *table* object parameters (Figure 26).
2. Into the ***Name*** input line enter a name for the table. By convention, the name should start with a lower case letter and should end with a suffix `Table` (e.g., `speakerFreqTable`). The name must be unique for the current MIB module. By changing the table name and applying the change, the table entry node name is also changed accordingly (i.e., carrying the same name prefix as the table node name).
3. The value for the ***Access/Max Access*** clause is automatically set to `not-accessible` and cannot be modified because the table object cannot be directly accessed via SNMP operations.
4. From the ***Status*** drop-down list, select the value for the status clause for this object:
   - `Mandatory` (only in SMIv1) - indicates that the object definition is valid and that the object must be implemented.
   - `Current` (only in SMIv2) - indicates that the object definition is valid.

- `Deprecated` - indicates that the object definition is valid in certain circumstances, but has been replaced by another. In SMIv1 MIBs, this value also denotes that the implementation of this object is required for conformance.
- `Obsolete` - indicates that the object definition is not valid (and in SMIv1 that the object should not be implemented).
- `Optional` (only in SMIv1) - indicates that the object definition is valid and may or may not be implemented.

> **Note:** After changing the status value and clicking the ***OK*** or the ***Apply*** button, Visual MIB Builder displays a dialog box asking if you would also like to change the status of the row object and all columnar objects. Clicking the ***Yes*** button will set the status of the table object itself as well as the status of all other objects within the table to a new value.

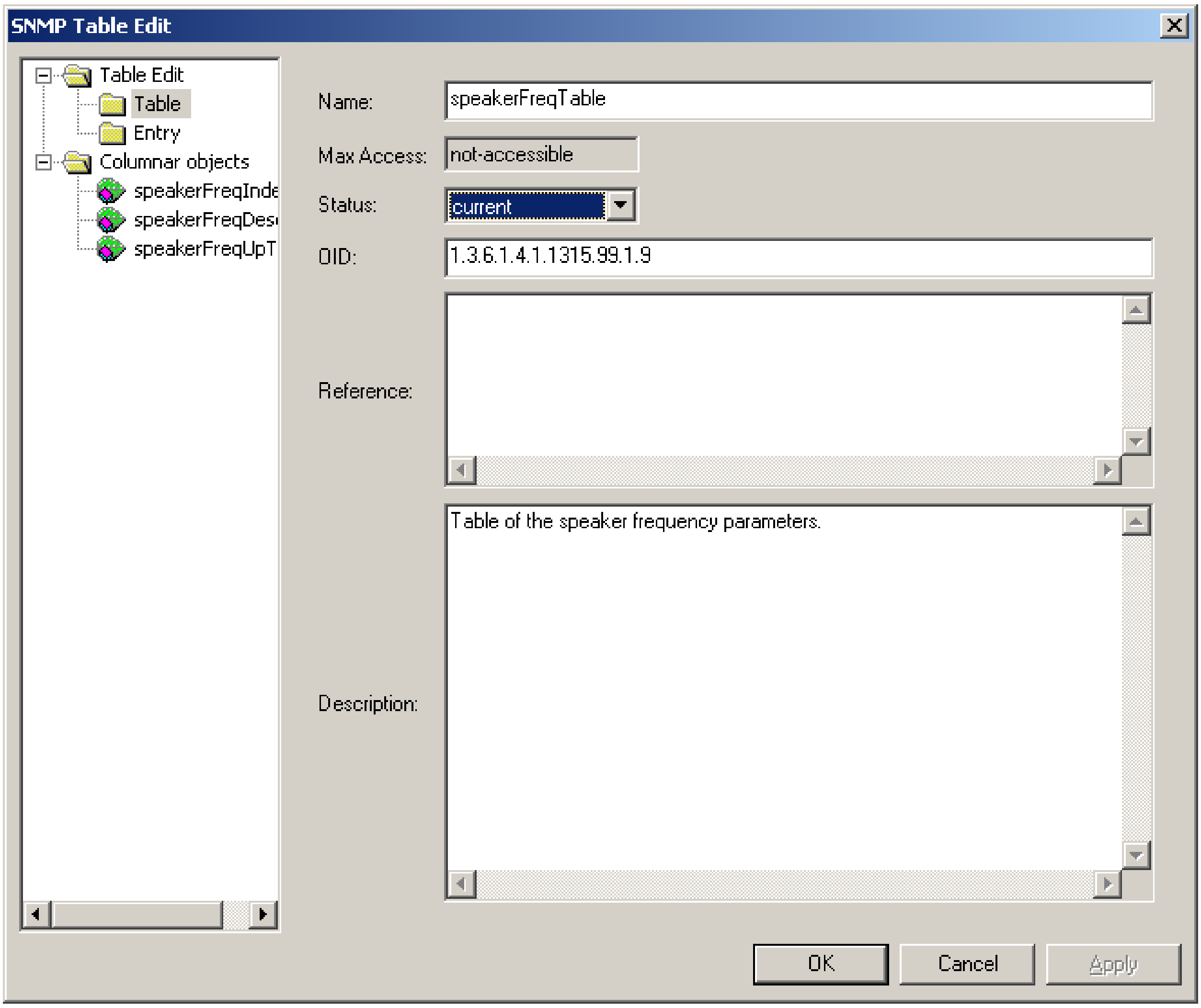

SNMP Table Edit

Table Edit
Table
Entry
Columnar objects
speakerFreqInde
speakerFreqDes
speakerFreqUpT

Name: speakerFreqTable

Max Access: not-accessible

Status: current

OID: 1.3.6.1.4.1.1315.99.1.9

Reference:

Description: Table of the speaker frequency parameters.

Figure 26: SNMP Table Edit dialog box – Table panel

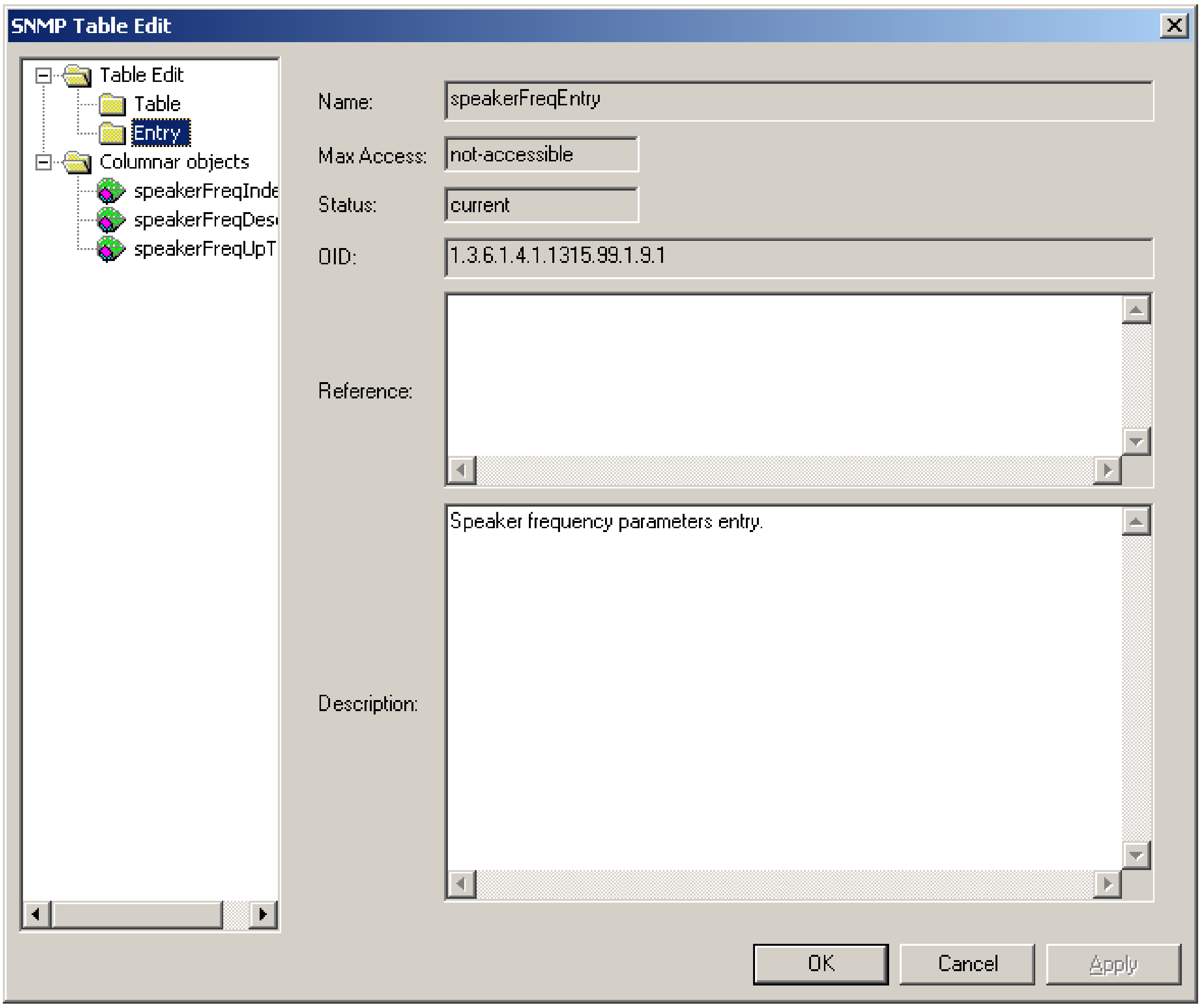
5. The ***OID*** input line displays the OID value assigned to the table object. Only the last sub-identifier in this input line can be modified. Modifying the OID value re-positions

table in the MIB tree and automatically adjusts OID values of the subordinated row and columnar objects.

6. To add a reference, enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in understanding the object definition. The reference clause is not obligatory for table objects.
7. To add a table description or any other information that is needed to understand the object or module, enter the description text into the ***Description*** input field. The text can be checked for spelling errors by using the ***Check Spelling*** pop-up command.

Entry panel

1. Click the ***Entry*** item in the SNMP Table Edit navigation tree in order to view or edit the *entry (row)* object parameters (Figure 27).



Name: speakerFreqEntry

Max Access: not-accessible

Status: current

OID: 1.3.6.1.4.1.1315.99.1.9.1

Reference:

Description: Speaker frequency parameters entry.

Figure 27: SNMP Table Edit dialog box – Entry panel

2. The ***Name*** input line displays the name of the *row object,* which is the same as the name of the table, except that the suffix `Table` is replaced by the suffix `Entry` (e.g., if a table is named `speakerFreqTable`, then the *row* is named `speakerFreqEntry`).

Note: You cannot directly change the name, syntax, status and OID value of the *row object.* However, you can change the corresponding properties of the *table object* (containing the *row object)* to automatically update the values of those parameters for both, table and row objects.

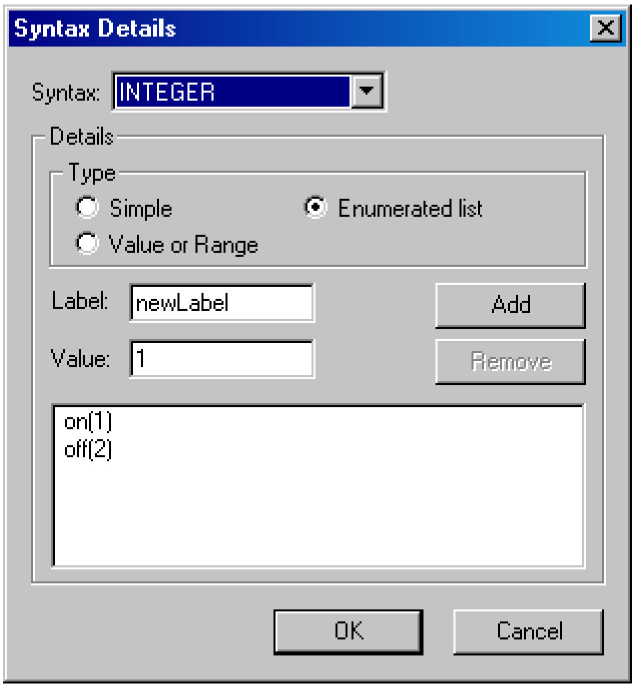
3. The value for the ***Access/Max Access*** clause is automatically set to `not-accessible` and cannot be modified since the *row* object cannot be directly accessed via SNMP operations.
4. The ***Status*** drop-down list is automatically set to match the status value of the containing *table* object and cannot be modified, because the status of the *row* object must be consistent with the status for the *table* object.
5. To add a reference, enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation. The reference clause is not obligatory for row objects.
6. To add a table description or any other information that is needed to understand the object or module, enter the description text into the ***Description*** input field. The text can be checked for spelling errors by using the ***Check Spelling*** pop-up command.

Columnar object panel

1. Click an entry below the ***Columnar Objects*** entry in the SNMP Table Edit navigation tree to view or edit the properties of the selected *columnar* object (Figure 30).
2. Into the ***Name*** input line, enter the name of the *columnar* node. The name must start with a lower case letter (e.g., `speakerFreqIndex`) and must be unique for the current MIB module. By default, the first part of the name is automatically given, based on the name of the table that contains this columnar object.

Tip: To disable this option, uncheck the ***Add table prefix to columnar objects*** checkbox in the MIB Builder Preferences dialog box | Builder tab.

3. To configure the ***Syntax*** definition, click the ***Browse*** button next to the input line. The Syntax Details dialog box appears (Figure 28), where you can choose from the base syntax types, as well as from the Type Assignments and textual-conventions that were imported form other MIB modules or defined in this MIB module. Besides, you can specify sizes, ranges and enumerated values for the syntax types that allow this type of sub-typing (refinement of the syntax).
4. From the ***Syntax*** drop-down list in the Syntax Details dialog box select the syntax type to be used for the target node.



Syntax: INTEGER

○ Simple ● Enumerated list
○ Value or Range

Label: newLabel

Value: 1

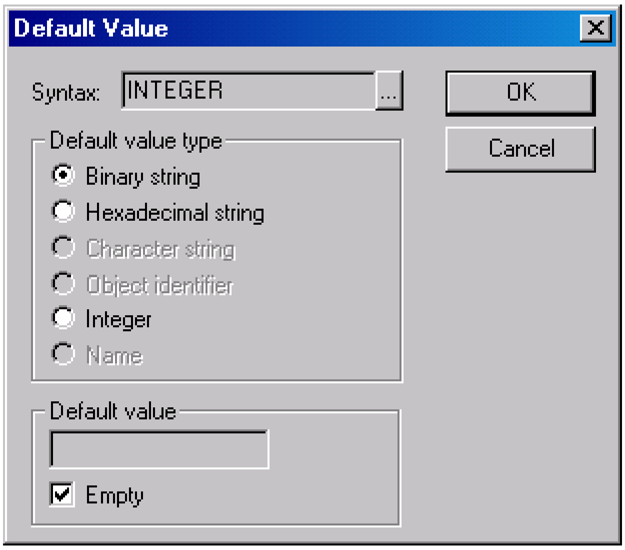
on(1)
off(2)

Figure 28: Syntax Details dialog box

5. Appearance of the ***Details*** frame changes according to the selected syntax type. Depending on the selected syntax type, the following options may be available:
   - ***Simple*** - If you select this option, no syntax refinement occurs. If the ***Defval*** parameter is set, the syntax will be formatted as specified in the ***Defval*** clause.
   - ***Enumerated List*** – Lets you define a list of labels and corresponding values (e.g., `Label – on; Value - 1`) for the selected syntax type.
   - ***Value or Range*** - Lets you specify one or more integer values (e.g., 1, 5, 10) and/or one or more integer ranges (30-50, 55-60) that the selected syntax type can take.
   - **Size or Size Range -** Lets you specify one or more string length sizes (e.g., 1, 5, 10) and/or one or more string length ranges (30-50, 55-60) that limit the string size of the selected syntax type.
6. To set the default value type for the object, click the ***Browse*** button next to the ***Defval*** input line. The Default Value dialog box appears (Figure 29).

> **Note:** The DEFVAL clause should be used only for columnar objects in tables that allow row creation via SNMP Set operation.

7. By clicking the ***Browse*** (**…**) button next to the ***Syntax*** input line in the Default Value dialog box, you can view the syntax details. In the Default Value dialog box, you can also set the formatting of the value and the default value that the instance of this columnar object will have when a new row in the table is created and the value for this instance is not specified in the SNMP Set operation that creates a new row.
8. If the ***Empty*** checkbox is checked, an empty DEFVAL clause will be generated.



Default Value

Syntax: INTEGER

Default value type

● Binary string
○ Hexadecimal string
○ Character string
○ Object identifier
○ Integer
○ Name

Default value

☑ Empty

Figure 29: Default Value dialog box

9. After setting all parameters in the Default Value dialog box, click the ***OK*** button to save the settings and close the dialog box.

10. From the ***Access*** drop-down list (***Max. Access*** in SMIv2), select the value for the access clause, which determines the allowed access to the object.

    - `not-accessible` – The object is not accessible for any kind of SNMP operation. SMIv2 requires this value be used for index columnar objects.
    - `accessible-for-notify` (only in SMIv2) – The object is accessible only for event report operations.
    - `read-only` – The object is accessible for SNMP retrieval and event report operations.
    - `write-only` (only in SMIv1) – The object may be an operand in SNMP retrieval, modifying and event report operations. This value should not be used, since it is obsolete according to the valid SMI specification.
    - `read-write` – The object may be an operand in SNMP retrieval, modifying and event report operations.
    - `read-create` (only in SMIv2) – The object may be an operand in SNMP retrieval, modifying and event report operations. Besides, the object may be an operand in SNMP Set operations that create a new object instance.

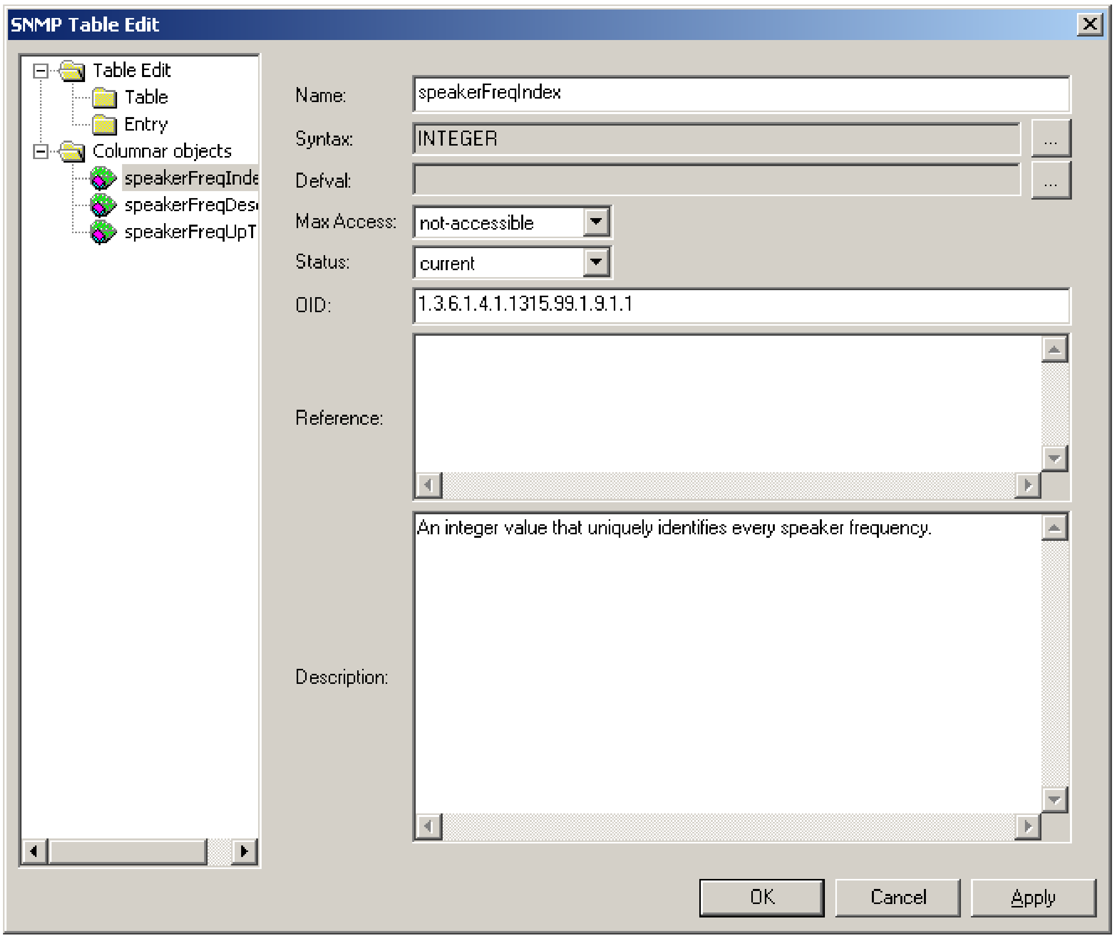
Note: Every *object type* that has the `Max. Access` clause value set to any other value than `not-accessible` must be a member of at least one object group (in SMIv2 MIB modules). Read the OBJECT-GROUP Constructs section for more information.

11. From the ***Status*** drop-down list, select the value for the status clause for this object.

    - `Mandatory` (only in SMIv1) - indicates that the object definition is valid and that the object must be implemented.
    - `Current` (only in SMIv2) - indicates that the object definition is valid.

- `Deprecated` - indicates that the object definition is valid in certain circumstances, but has been replaced by another. In SMIv1 MIBs, this value also denotes that the implementation of this object is required for conformance.
- `Obsolete` - indicates that the object definition is not valid (and in SMIv1 that the object should not be implemented).
- `Optional` (only in SMIv1) - indicates that the object definition is valid and may or may not be implemented.

12. The ***OID*** input line displays the OID value assigned to the columnar object. Only the last sub-identifier in this input line can be modified.
13. To add a reference, enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in understanding the object definition. The reference clause is not obligatory for columnar objects.
14. To add a description or any other information that is needed to understand the object or module, enter the description text into the ***Description*** input field. The text can be checked for spelling errors by using the ***Check Spelling*** pop-up command.



SNMP Table Edit

Name: speakerFreqIndex

Syntax: INTEGER

Defval:

Max Access: not-accessible

Status: current

OID: 1.3.6.1.4.1.1315.99.1.9.1.1

Reference:

Description: An integer value that uniquely identifies every speaker frequency.

Figure 30: SNMP Table Edit dialog box – Columnar object panel

15. Switch to another entry in the in the navigation tree or apply the changes and close the SNMP Table Edit dialog box by clicking the ***OK*** button.

5.2.5 Adding TRAP-TYPE / NOTIFICATION-TYPE Constructs to MIB Module

The TRAP-TYPE (SMIv1) and NOTIFICATION-TYPE (SMIv2) constructs are used for specifying the events that an agent can report to SNMP managers.

Note: The TRAP-TYPE construct can be used only in SMIv1 MIB modules and the NOTIFICATION-TYPE construct can be used only in SMIv2 MIB modules.

1. From the Components window (***View / Components***) select the TRAP-TYPE or the NOTIFICATION-TYPE component and drag-and-drop it onto the MIB tree in the MIB Builder window. As you drag the node to the MIB tree, Visual MIB Builder will indicate where in the MIB tree the node will be inserted when you release the mouse button. There are two possible cases:
   - ❑ The target node is selected. That means that the node you are about to drop will be added as the last child node of the selected node.
   - ❑ A line is drawn between two nodes. That means that the node you are about to drop will be inserted between these two nodes.

2. Instead of using the drag-and-drop technique, you can also select a component in the Components window and use the ***Edit / Copy*** command. This command will place the selected node to the clipboard. In the MIB Builder window, select a node under which you wish to insert the copied node and use the ***Edit / Paste*** command. Although this action behaves identically as the drag-and-drop technique, there are some limitations. First, there is no visual feedback on whether the node can be inserted at the desired location, and second, the node can only be inserted as the last child node of the target node.

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View / Node Preview***) and select the ***SMI Node Preview*** tab.

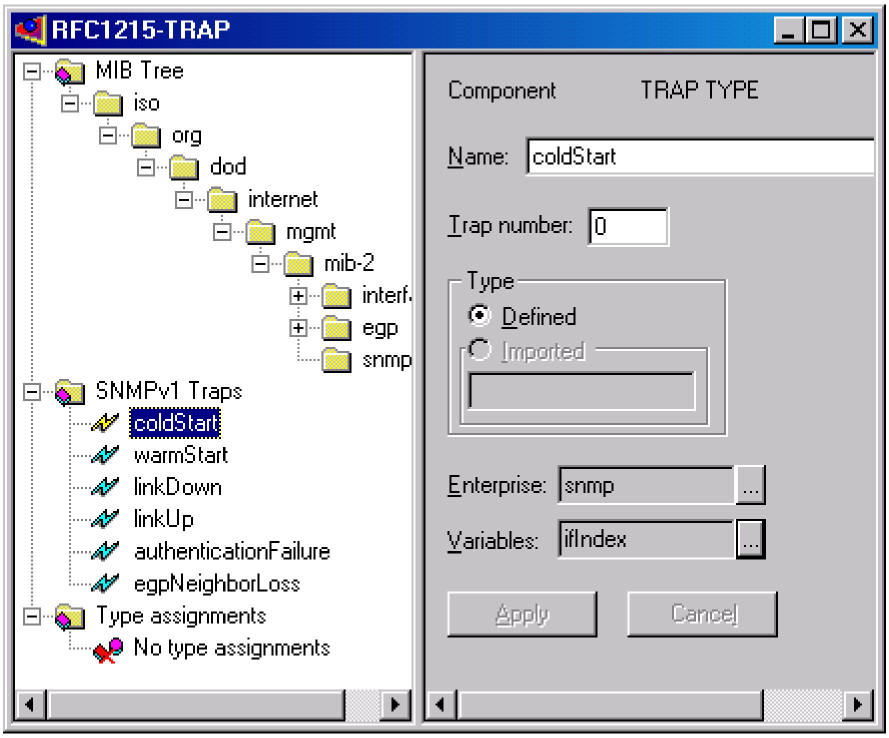
Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit / Remove*** command. When removing a parent node all its child nodes will be removed as well.

TRAP-TYPE Constructs

The TRAP-TYPE construct is used in SMIv1 MIB modules to specify the events, which agents can report to SNMP managers via SNMPv1 Trap messages.

1. Once the TRAP-TYPE node is added to the MIB tree, click on the node to view the Properties window panel.

> **Tip:** For more information on how to add a TRAP-TYPE component, read the Adding TRAP-TYPE / NOTIFICATION-TYPE Constructs to MIB Module section.



Component TRAP TYPE

Name: coldStart

Trap number: 0

Type
● Defined
○ Imported

Enterprise: snmp

Variables: ifIndex

Figure 31: TRAP-TYPE Properties window panel

2. Into the ***Name*** input line enter the name of the TRAP-TYPE node. The name must start with a lower case letter (e.g., `myDeviceFailureTrap`) and must be unique for the current MIB module.

3. Into the ***Trap number*** input line, specify a non-negative integer number identifying the enterprise specific trap. Note that all “generic traps” are already defined by IETF and that only new enterprise specific traps can be defined by private parties.

4. To specify the trap type, click the ***Browse*** button next to the ***Enterprise*** input line. The Enterprise Selection dialog box appears (Figure 32). The event report message is identified by the values in this field. When the value specified in the Enterprise input line is `snmp` (i.e., `iso.org.dod.internet.mgmt.mib-2.snmp`), then the value is set for generic trap. If any other value is specified, then the value is set for the specific trap. No new generic traps can be defined by private parties.

5. From the ***Available enterprises*** list select the appropriate node of your enterprise and click the right arrow button. The selected node will appear in the ***Selected enterprises*** list. Click the ***OK*** button and the selected value will appear also in the ***Enterprises*** input line.

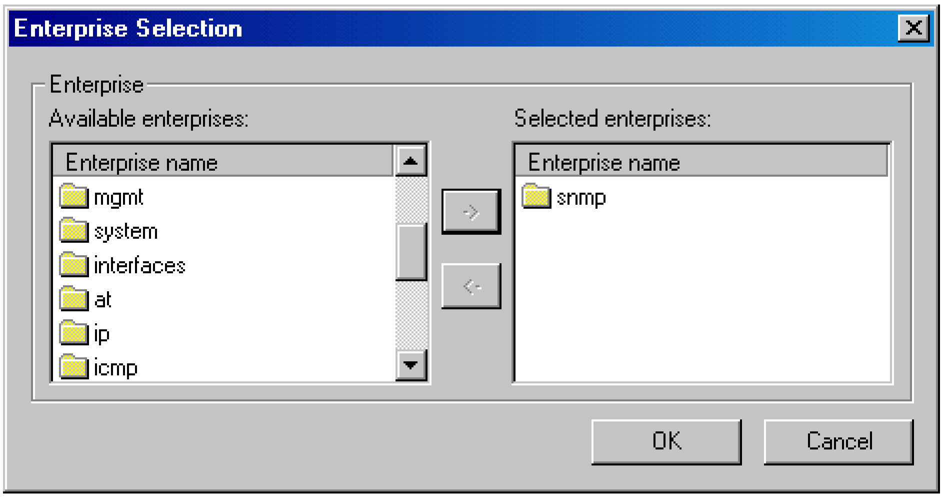


Figure 32: Enterprise Selection dialog box

6. To specify the object(s) whose value describes the event and should be included in the SNMP Trap messages, click the ***Browse*** button next to the ***Variables*** input line to open the Variable Selection dialog box. From the ***Available variables*** list select the object(s) and click the right arrow button. Selected object will appear in the ***Selected variables*** list. Click the ***OK*** button to save the settings and close the window. The selected value will appear in the ***Variables*** input line. There is no sub clause to specify exactly which instance of a columnar object should be returned in the event report message, so the Description clause must be used for this purpose.

Note: An instance of each object specified in the ***Variables*** clause together with the value of this object instance is sent in the variable bindings list of a Trap message. To specify which instance of a columnar object to include in the Trap message, the Description clause must be used ***(View / Additional Properties / Description tab)***.

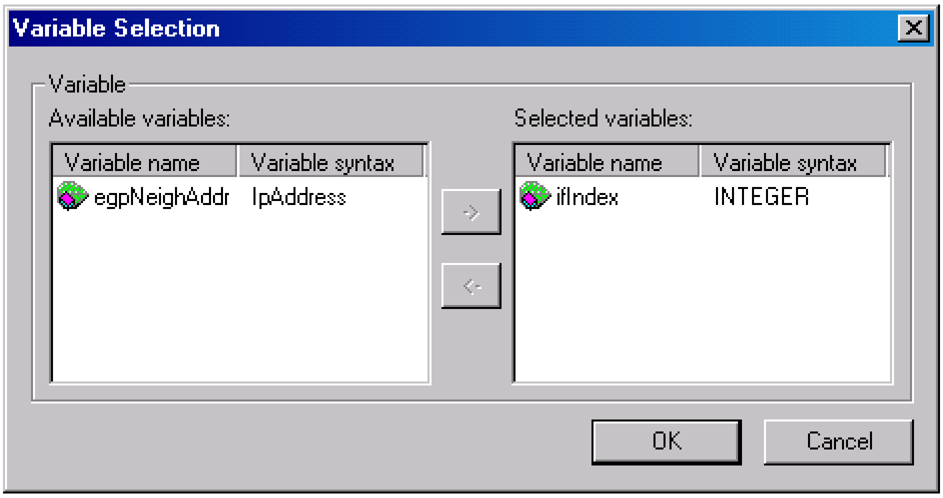


Figure 33: Variable Selection dialog box

7. To add a comment to the object, open the Additional Properties window (***View*** / ***Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.

8. To add a description of the object or any information that is needed to understand a MIB object or module, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Description*** tab. The text can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands.

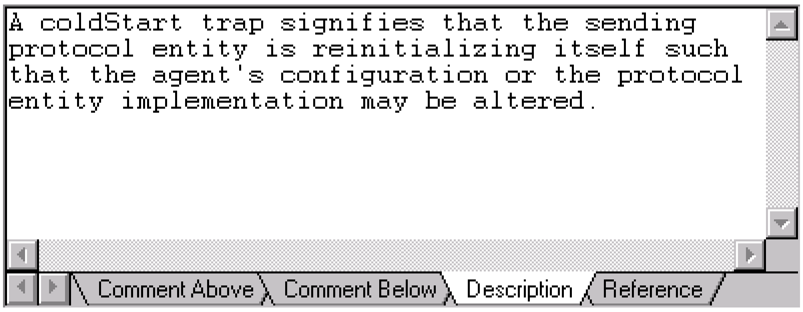

A coldStart trap signifies that the sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation may be altered.

Figure 34: Description tab

Tip: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands. For more information on spell checking, read the Check Spelling section of this manual.

9. To add a reference, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is optional.

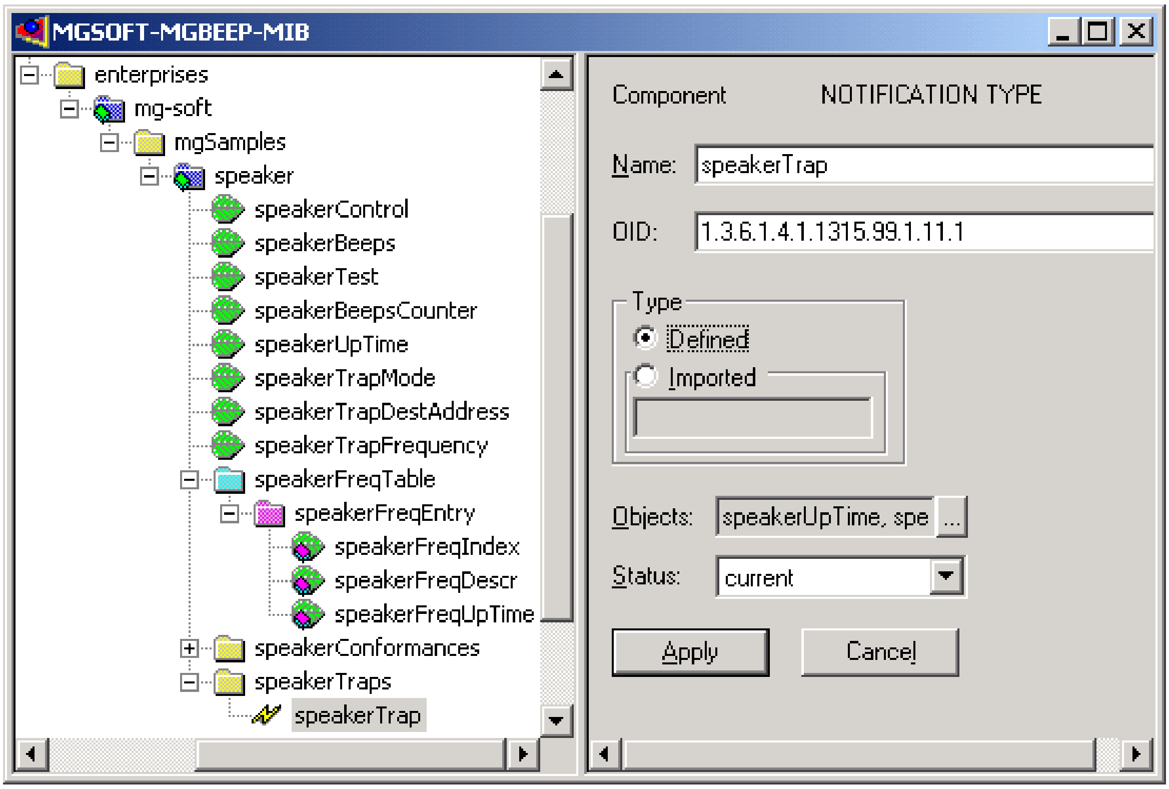
10. After setting all parameters, press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

NOTIFICATION-TYPE Constructs

The NOTIFICATION-TYPE construct is used in SMIv2 MIB modules to specify the events, which agents can report to SNMP managers via SNMP Trap or Inform messages.

> **Tip 1:** The SNMP Trap messages represent unacknowledged notifications, meaning that they do not initiate any response from the receiver. The SNMP Inform messages, on the other hand, require that the receiver replies with a response message, confirming that the notification has been received.
>
> **Tip 2:** For more information on how to add a NOTIFICATION-TYPE construct, read the Adding TRAP-TYPE / NOTIFICATION-TYPE Constructs to MIB Module section.

1. Once the NOTIFICATION-TYPE node is added to the MIB tree, click the node to modify its default properties in the Properties window panel.
2. Into the ***Name*** input line enter the name of the NOTIFICATION-TYPE node. The name must start with a lower case letter (e.g., `speakerTrap`) and must be unique for the current MIB module.



Component NOTIFICATION TYPE

Name: speakerTrap

OID: 1.3.6.1.4.1.1315.99.1.11.1

Type
● Defined
○ Imported

Objects: speakerUpTime, spe ...

Status: current

Figure 35: NOTIFICATION -TYPE Properties window panel

3. To specify the object(s) whose value describes the event and will be send in the event report message as a variable bindings, click the ***Browse*** button next to the ***Objects*** input line to open the Object Selection dialog box. From the ***Available objects*** list select the object(s) and click the right arrow button. Selected object will appear in the ***Selected objects*** list. Click the ***OK*** button to save the settings and close the window. The selected value will appear in the ***Objects*** input line (Figure 33).

4. From the ***Status*** drop-down list, select the value for the status clause.
   - ❑ `Current` - indicates that the object definition is valid.
   - ❑ `Deprecated` - indicates that the object definition is valid in certain circumstances, but has been replaced by another.
   - ❑ `Obsolete` - indicates that the object definition is not valid

> **Note 1:** A single instance of each object and its value specified in the Object clause is included in the variable bindings list of an event report message. To specify which instance of a columnar object to include, the Description clause must be used ***(View / Additional Properties / Description tab)***.
>
> **Note 2:** The status of the object specified in the Object clause must comply with the status of the event definition, meaning, if the status of the event definition is `current`, then all objects in the Object clause must also have status of `current`.

5. To add a comment to the object, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.

6. To add a description of the object or any information that is needed to understand a MIB object or module, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Description*** tab (Figure 34). The text can be checked for spelling errors by using the ***Tools / Check spelling*** commands.

> **Tip:** The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools / Check spelling*** commands. For more information on spell checking, read the Spell section of this manual.

7. To add a reference, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is optional for all constructs.

> **Note:** Each NOTIFICATION-TYPE node must be a member of at least one notification group. To define a collection of related NOTIFICATION-TYPEs, the NOTIFICATION-GROUP construct is used.

8. After setting all parameters, press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

5.2.6 Adding Type Assignments and TEXTUAL-CONVENTION Constructs to MIB Module

Type Assignment (in SMIv1 and SMIv2) and *TEXTUAL-CONVENTION* (in SMIv2 only) constructs are used to create a “new” data type (syntax). Such “new” data types can be defined by adding restrictions to an existing base type or previously created Type Assignment or TEXTUAL-CONVENTION constructs (no new SMI base types may be defined). Type Assignments and TEXTUAL-CONVENTIONS do not actually create a new ASN.1 type although they look and function as new data types.

Note: Type assignment and TEXTUAL-CONVENTION constructs are used for the same purpose, but with one difference. Type Assignments are used for creating new data types in SMIv1 MIB modules, and TEXTUAL-CONVENTION constructs are used for creating new data types in SMIv2 MIB modules.

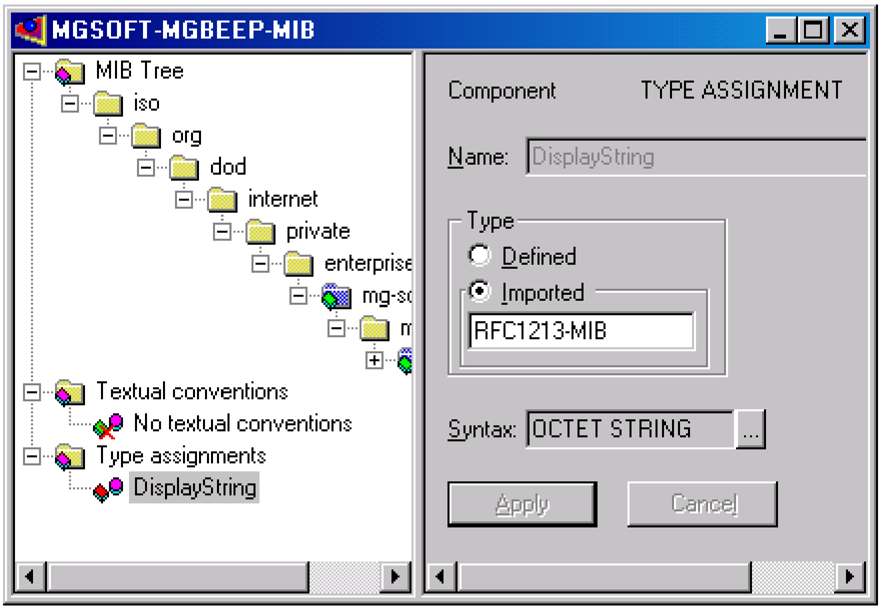
1. To add a Type Assignment or TEXTUAL-CONVENTION node to the MIB module, open the Components window (***View / Components***), select a *Type Assignments* or the TEXTUAL-CONVENTION component and drag-and-drop it onto the MIB tree in the MIB Builder window. As you drag the node over the MIB tree, Visual MIB Builder will indicate where in the MIB tree the node will be inserted when you release the mouse button. There are two possible cases:
   - ❑ The target node is selected. That means that the node you are about to drop will be added as the last child node of the selected node.
   - ❑ A line is drawn between two nodes. That means that the node you are about to drop will be inserted between these two nodes.
2. Instead of using the drag-and-drop technique, you can also select a component in the Components window and use the ***Edit / Copy*** command. This command will place the selected node to the clipboard. In the MIB tree, select a node under which you wish to insert the copied node and use the ***Edit / Paste*** command. Although this action behaves identically as the drag-and-drop technique, there are some limitations. First, there is no visual feedback on whether the node can be inserted at the desired location, and second, the node can only be inserted as the last child node of the target node.

Tip 1: To view the selected node in the MIB module definition language, open the Node Preview window (***View / Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit / Remove*** command. When removing a parent node all its child nodes will be removed as well.

Type Assignment Constructs

1. Once the Type Assignments node is added to the MIB tree, click on the node to view the Properties window panel.



Component TYPE ASSIGNMENT

Name: DisplayString

Type

○ Defined

● Imported

RFC1213-MIB

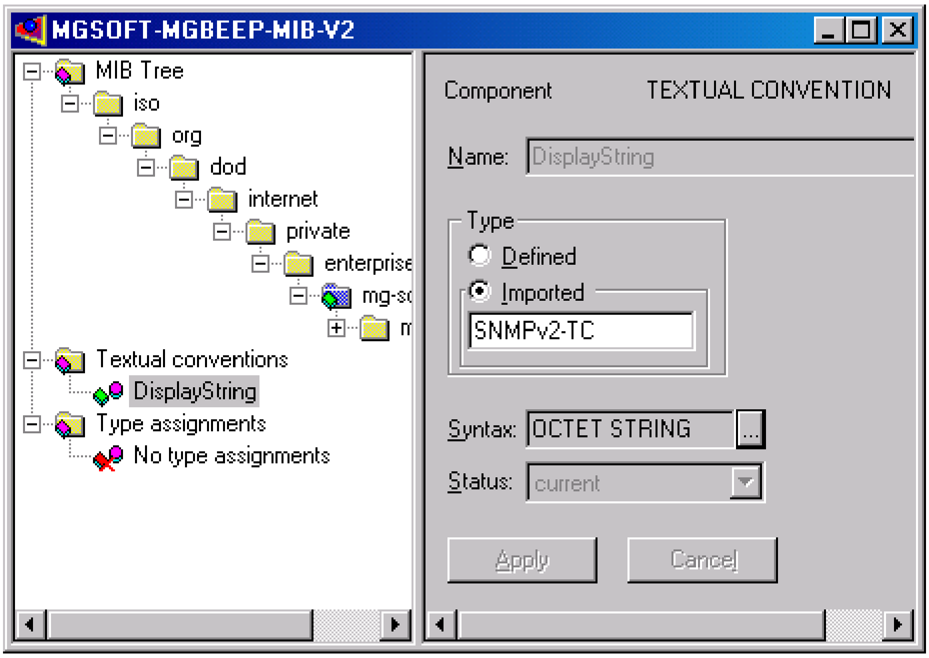
Syntax: OCTET STRING ...

Figure 36: Type assignment Properties window panel

2. Into the ***Name*** input line enter the name of the *Type Assignments* node. The name must start with an upper case letter (e.g., `DisplayString`) and must be unique for the current MIB module.
3. Click the ***Browse*** button next to the ***Syntax*** input line to open the Syntax Details dialog box. From the ***Syntax*** drop-down list select the syntax type to be used for the target node.
4. Appearance of the ***Details*** frame changes according to the selected syntax type. Depending on the selected syntax type, the following options may be available:
   - ***Simple*** - If you select this option, no syntax refinement occurs. If the ***Defval*** parameter is set, the syntax will be formatted as specified in the ***Defval*** clause.
   - ***Enumerated List*** – Lets you define a list of labels and corresponding values (e.g., `Label – on; Value - 1`) for the selected syntax type.
   - ***Value or Range*** - Lets you specify one or more integer values (e.g., 1, 5, 10) and/or one or more integer ranges (30-50, 55-60) that the selected syntax type can take.
   - **Size or Size Range -** Lets you specify one or more string length sizes (e.g., 1, 5, 10) and/or one or more string length ranges (30-50, 55-60) that limit the string size of the selected syntax type.
5. To add a comment to the object, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the Comment Above and Comment Below tabs.
6. After setting all parameters, click the ***Apply*** button in the Properties window panel or the changes will not take effect.

TEXTUAL-CONVENTION Constructs

1. Once the TEXTUAL-CONVENTION node is added to the MIB tree, click on the node to view the Properties window panel.



Component TEXTUAL CONVENTION

Name: DisplayString

Type: ○ Defined ● Imported SNMPv2-TC

Syntax: OCTET STRING ...

Status: current

Figure 37: TEXTUAL-CONVENTION Properties window panel

Note: When building large and more complex SMIv2 MIB modules or when defining textual conventions that are supposed to be widely used, it is common to create a separate MIB module that contains only textual conventions and then import them into the target module(s). However, note also that the standard SNMPv2-TC MIB module already contains all standard textual convention definitions, which can be imported in any SMIv2 MIB module.

2. Into the ***Name*** input line enter the name of the TEXTUAL-CONVENTION node. The name must start with an upper case letter (e.g., `DisplayString`) and must be unique for the current MIB module.

3. Click the ***Browse*** button next to the ***Syntax*** input line to open the Syntax Details dialog box (Figure 15). From the ***Syntax*** drop-down list select the syntax type to be used for the target node.

4. Appearance of the ***Details*** frame changes according to the selected syntax type. Depending on the selected syntax type, the following options may be available:

   - ***Simple*** - If you select this option, no syntax refinement occurs. If the ***Defval*** parameter is set, the syntax will be formatted as specified in the ***Defval*** clause.

   - ***Enumerated List*** – Lets you define a list of labels and corresponding values (e.g., `Label – on; Value - 1`) for the selected syntax type.

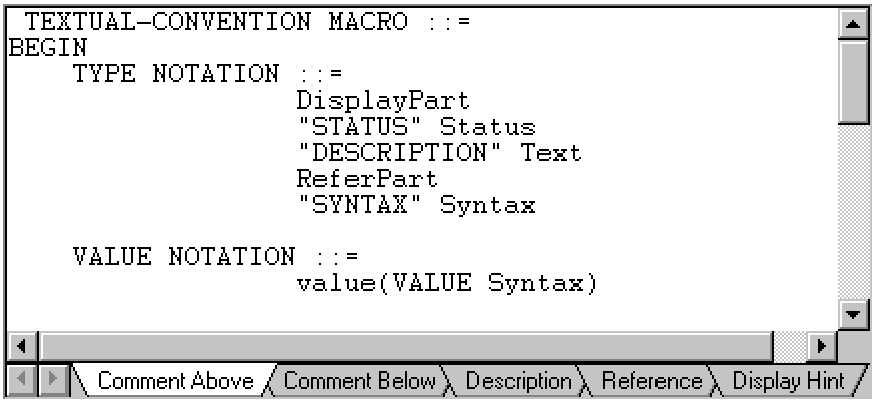
   - ***Value or Range*** - Lets you specify one or more integer values (e.g., 1, 5, 10) and/or one or more integer ranges (30-50, 55-60) that the selected syntax type can take.

- **Size or Size Range -** Lets you specify one or more string length sizes (e.g., 1, 5, 10) and/or one or more string length ranges (30-50, 55-60) that limit the string size of the selected syntax type.

5. From the ***Status*** drop-down list, select the value for the status clause.

   - `Current` - indicates that the object definition is valid.
   - `Deprecated` - indicates that the object definition is still valid in certain circumstances, but has been replaced by another.
   - `Obsolete` - indicates that the object definition is not valid

6. To add a comment to the object, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the Comment Above and Comment Below tabs.

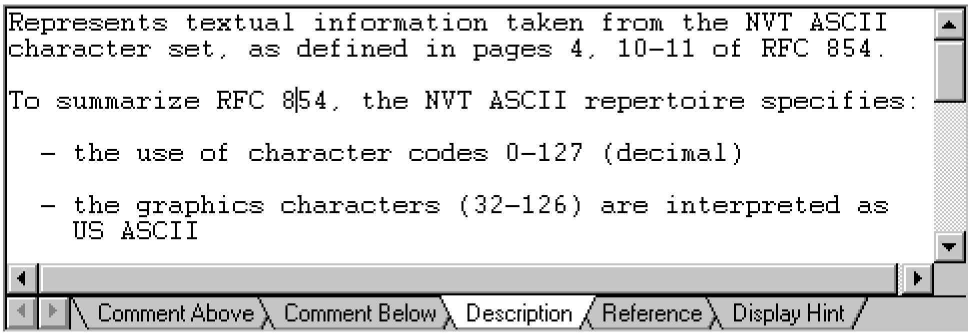


```
 TEXTUAL-CONVENTION MACRO ::=
BEGIN
    TYPE NOTATION ::=
                  DisplayPart
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  "SYNTAX" Syntax

    VALUE NOTATION ::=
                  value(VALUE Syntax)
```

Figure 38: Comment for DisplayString TEXTUAL-CONVENTION node

7. To add a description of the object or any information that is needed to understand a MIB object or module, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Description*** tab (Figure 16). The text can be checked for spelling errors by using the ***Tools / Check spelling*** commands.



```
Represents textual information taken from the NVT ASCII
character set, as defined in pages 4, 10-11 of RFC 854.

To summarize RFC 854, the NVT ASCII repertoire specifies:

  - the use of character codes 0-127 (decimal)

  - the graphics characters (32-126) are interpreted as
    US ASCII
```

Figure 39: Description for DisplayString TEXTUAL-CONVENTION node

8. To add a reference, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is not obligatory for the TEXTUAL-CONVENTION construct.

9. Use the ***Display Hint*** tab in the Additional Properties window (***View*** / ***Additional Properties***) to specify a hint for displaying a value and/or describing sub structuring of the value. The display hint clause is not obligatory for the TEXTUAL-CONVENTION construct.

10. After setting all parameters, click the ***Apply*** button in the Properties window panel or the changes will not take effect.

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View*** / ***Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit*** / ***Remove*** command. When removing a parent node all its child nodes will be removed as well.

Tip 3: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands. For more information on spell checking, read the Spell section of this manual.

5.2.7 Adding Constructs Specific to SMIv2 to MIB Modules

The rules for writing MIBs are defined in a collection of documents called the Structure of Management Information (SMI). SMI defines exactly how individual type of management information should be defined. Currently there are two versions of SMI specification, SMIv1 and SMIv2. In SMIv2, some new constructs were added and the SMIv1 OBJECT-TYPE construct was changed to accommodate the changes in Status and Access limits. The following constructs were introduced by the SMIv2:

- ❑ OBJECT-IDENTITY construct
- ❑ OBJECT-GROUP construct
- ❑ NOTIFICATION-TYPE construct
- ❑ NOTIFICATION-GROUP construct
- ❑ MODULE-IDENTITY construct
- ❑ MODULE-COMPLIANCE construct
- ❑ AGENT-CAPABILITIES construct
- ❑ TEXTUAL-CONVENTION construct

SMIv2 offers a richer and more precise syntax for defining MIB modules. Although it is recommended for all new MIBs to be defined by using the SMIv2 syntax, some MIB compilers and other MIB processing applications may still support SMIv1 MIBs only. Therefore, it is your decision whether you design your MIBs in SMIv1 or in SMIv2 syntax.

Note: The contents of the Components window depends on the SMI version used in the currently selected MIB module (MIB Builder window). If a MIB Builder window containing a SMIv1 MIB is currently active (selected), the Components window will list only SMIv1 constructs, if a SMIv2 MIB is currently selected, the Components window will list only SMIv2 constructs.

Due to similarities with their SMIv1 counterpart constructs, the procedures of defining MIB objects with SMIv2 OBJECT-TYPE, NOTIFICATION-TYPE and TEXTUAL-CONVENTION constructs are already described in previous sections. This section describes the procedure of adding and specifying the properties of MODULE-IDENTITY, OBJECT-IDENTITY, OBJECT-GROUP, NOTIFICATION-GROUP, MODULE-COMPLIANCE, and AGENT-CAPABILITIES SMIv2 constructs to MIB modules.

To add an SMIv2 component to the MIB Tree, do the following:

1. From the Components window (***View* / *Components***) select an SMIv2 component and drag-and-drop it to the desired position in the MIB tree in the MIB Builder window. As you drag the node over the MIB tree, Visual MIB Builder will indicate where in the MIB tree the node will be inserted when you release the mouse button. There are two possible cases:

   - ❑ The target node is selected. That means that the node you are about to drop will be added as the last child node of the selected node.

- ❑ A line is drawn between two nodes. That means that the node you are about to drop will be inserted between these two nodes.

2. Instead of using the drag-and-drop technique, you can also select a component in the Components, MIB Builder or in Database window and use the ***Edit*** / ***Copy*** command. This command will place the selected node to the clipboard. In the MIB Builder window, select a node under which you wish to insert the copied node and use the ***Edit*** / ***Paste*** command. Although this action behaves identically as the drag-and-drop technique, there are some limitations. First, there is no visual feedback on whether the node can be inserted at the desired location, and second, the node can only be inserted as the last child node of the target node.

> **Tip 1:** To view the selected node in the MIB module definition language, open the Node Preview window (***View*** / ***Node Preview***) and select the ***SMI Node Preview*** tab.
>
> **Tip 2:** To remove a node from the MIB tree, select the node and use the ***Edit*** / ***Remove*** command. When removing a parent node all its child nodes will be removed as well.

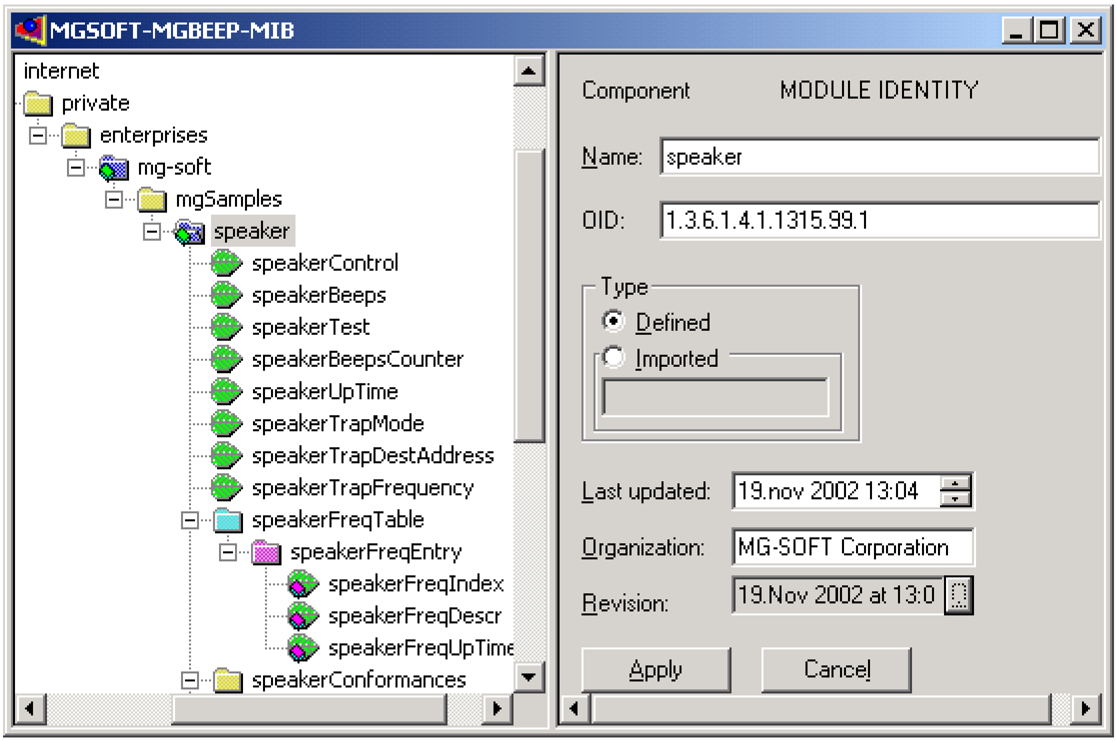
3. The new object has a default name and an automatically assigned OID. Further details on editing SMIv2 MIB nodes can be found in the following sections.

MODULE-IDENTITY Constructs

The MODULE IDENTITY construct is used to specify general information about SMIv2 MIB modules such as the name of the organization that has the authority over the module, the date of the last update, etc. As the SMIv2 requires, an SNMPv2 MIB module must contain exactly one module identity definition that must precede all other definitions in the module.

TIP: For more information on how to add a MODULE-IDENTITY object read the Adding Constructs Specific to SMIv2 to MIB Modules section.

1. Once the MODULE-IDENTITY node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.



Component MODULE IDENTITY

Name: speaker

OID: 1.3.6.1.4.1.1315.99.1

Type
● Defined
○ Imported

Last updated: 19.nov 2002 13:04

Organization: MG-SOFT Corporation

Revision: 19.Nov 2002 at 13:0

Figure 40: MODULE-IDENTITY Properties window panel

2. Into the ***Name*** input line enter the name of the MODULE-IDENTITY node. The name must start with a lower case letter (e.g., `speaker`) and must be unique for the current MIB module.

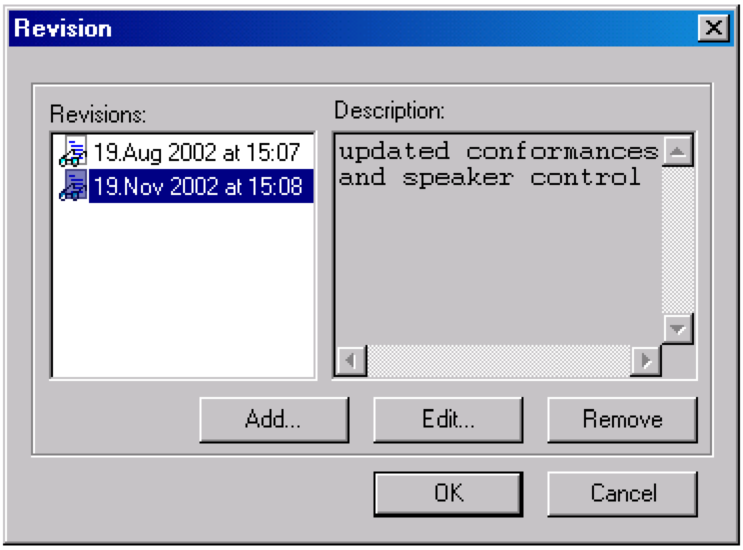
Note: If you do not want the Last updated value to be automatically set, uncheck the ***Update MODULE-IDENTITY component*** checkbox in the Revision Control dialog box.

3. The value in the ***Last updated*** input line should be identical to the date and time from the ***Revision*** input line, if set. By default, this value will be updated automatically if there is at least one revision entered. Otherwise, the Last updated value can be updated to the current date and time. To change the date and time, click on the unit you wish to change (e.g., day, month…) and use the spin buttons to set the desired value. The date and time are represented in UTC Time format.

4. Into the ***Organization*** input line enter the name of the organization that has the authority over this MIB module (e.g., `mg-soft`).

5. The ***Revision*** input line is used for specifying information about the revision of the module. To add a revision, click the ***Browse*** button next to the input line. The Revision dialog box appears (Figure 41). The Revision dialog box allows you to freely edit the date and time of the revision and its description. Adding a revision will lock all objects defined in the current MIB module. As you add a new revision, the ***Last updated*** value will be updated too. Removing a revision will also remove all subsequent revisions and unlock all associated objects.

Tip: For more information on revisions, read the Revising MIB Modules section.



Revisions:
19.Aug 2002 at 15:07
19.Nov 2002 at 15:08

Description:
updated conformances and speaker control

Figure 41: Revision dialog box

6. To add a comment to the object, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.

Tip: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***File / Check spelling*** commands. For more information on spell checking, read the Spell section of this manual.

7. To add a description of the object or any information that is needed to understand a MIB object or module, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Description*** tab.

8. To specify the contact information, open the Additional Properties window (***View / Additional Properties***) and enter the name, postal address, telephone number, and e-mail address of the person to whom technical queries concerning this information module should be sent into the ***Contact info*** tab.

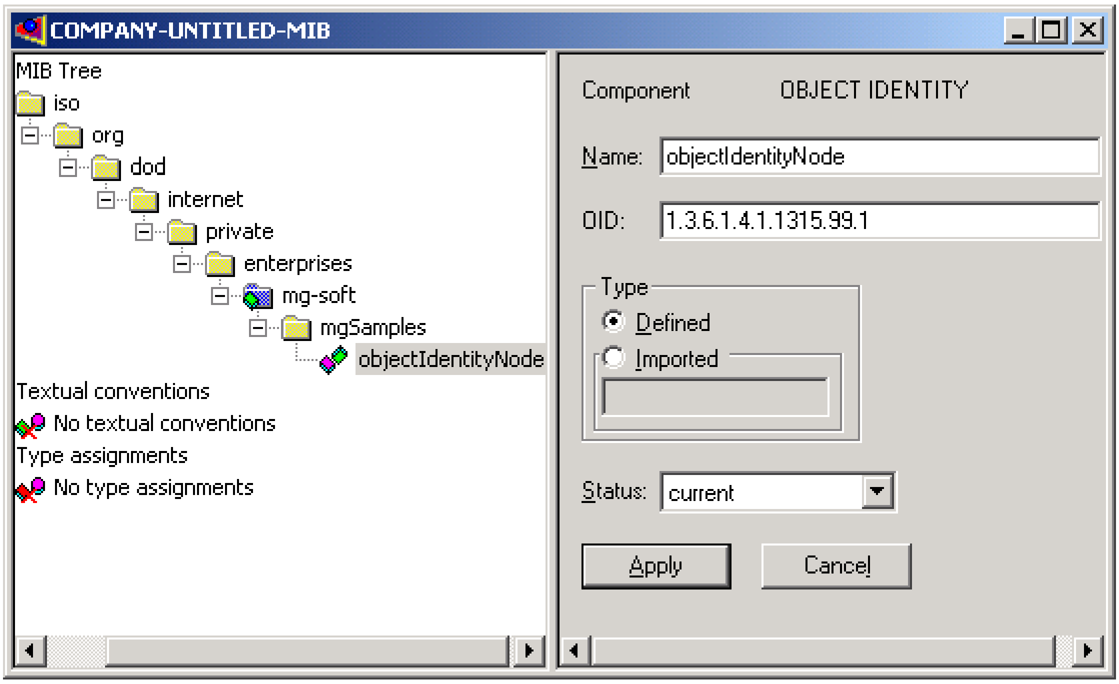
9. After setting all parameters, click the ***Apply*** button in the Properties window panel or the changes will not take effect.

OBJECT-IDENTITY Constructs

The OBJECT-IDENTITY is an SMIv2 construct used to assign an OID value to an identifier in the MIB module and to register the assigned OID value. The registration is a permanent assignment of an OID, which means that no other item may be registered with the same OID value.

1. Once the OBJECT-IDENTITY node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.

Tip: For more information on how to add an OBJECT-IDENTITY MIB object, read the Adding Constructs Specific to SMIv2 to MIB Modules section.



Component OBJECT IDENTITY

Name: objectIdentityNode

OID: 1.3.6.1.4.1.1315.99.1

Type

● Defined

○ Imported

Status: current

Figure 42: OBJECT-IDENTITY Properties window panel

2. Into the ***Name*** input line enter the name of the OBJECT-IDENTITY node. The name must start with a lower case letter (e.g., `objectIdentityNode`) and must be unique for the current MIB module.
3. From the ***Status*** drop-down list, select the value for the status clause.
   - `Current` - indicates that the object definition is valid.
   - `Deprecated` - indicates that the object definition is still valid in certain circumstances, but has been replaced by another.
   - `Obsolete` - indicates that the object definition is not valid
4. To add a comment to the object, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.
5. To add a description of the object or any information that is needed to understand a MIB object or module, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Description*** tab. The text can be checked for

spelling errors by using the ***Tools*** / ***Check spelling*** commands. Note that for the OBJECT-IDENTITY construct the description clause must be present.

6. To add a reference, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. This is not an obligatory clause for the OBJECT-IDENTITY construct.

Tip: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools*** / ***Check spelling*** commands. For more information on spell checking, read the Check Spelling section of this manual.

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View*** / ***Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit*** / ***Remove*** command. When removing a parent node all its child nodes will be removed as well.

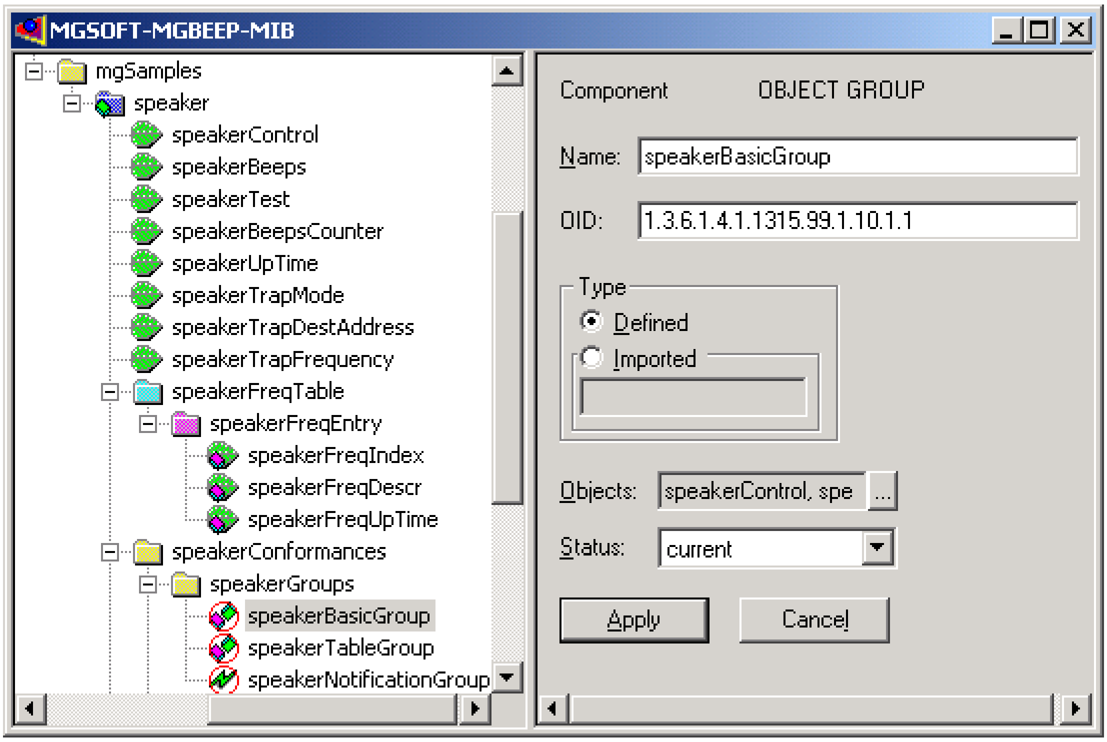
OBJECT-GROUP Constructs

The OBJECT-GROUP is an SMIv2 construct used to define a collection of related object type definitions.

> **Note:** Every OBJECT-TYPE construct that has the ***Max. Access*** clause value set to any other value than `not-accessible` must be a member of at least one object group.

1. Once the OBJECT-GROUP node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.

> **Tip:** For more information on how to add an OBJECT-GROUP MIB object, read the **Adding Constructs Specific to SMIv2 to MIB Modules** section.



Component OBJECT GROUP

Name: speakerBasicGroup

OID: 1.3.6.1.4.1.1315.99.1.10.1.1

Type
● Defined
○ Imported

Objects: speakerControl, spe ...

Status: current

Figure 43: OBJECT-GROUP Properties window panel

2. Into the ***Name*** input line enter the name of the OBJECT-GROUP node. The name must start with a lower case letter (e.g., `speakerBasicGroup`) and must be unique for the current MIB module.

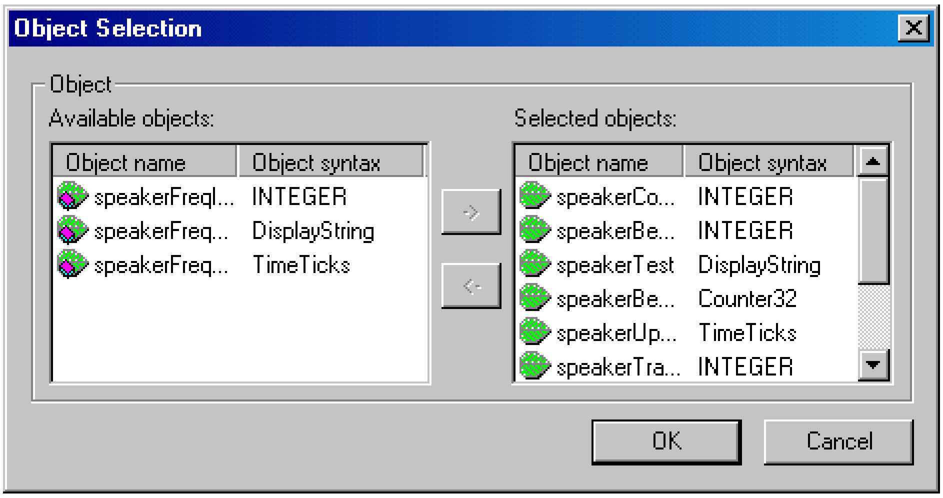
3. To specify the MIB objects to be added to the group, click the ***Browse*** button next to the ***Objects*** input line. The Object Selection dialog box appears (Figure 44).

4. From the ***Available objects*** list select an object and click the right arrow button. The selected object will appear in the ***Selected objects*** list. Repeat the same procedure to add additional objects to the group. Click the ***OK*** button and the

> **Note:** The status of the OBJECT-TYPE nodes specified in the OBJECT-GROUP must conform to the status of the object group definition.

selected objects will appear also in the ***Objects*** input line.

5. From the ***Status*** drop-down list, select the value for the status clause.
   - `Current` - indicates that the object definition is valid.
   - `Deprecated` - indicates that the object definition is still valid in certain circumstances, but has been replaced by another.
   - `Obsolete` - indicates that the object definition is not valid



Available objects:

Object name	Object syntax
speakerFreqI...	INTEGER
speakerFreq...	DisplayString
speakerFreq...	TimeTicks

Selected objects:

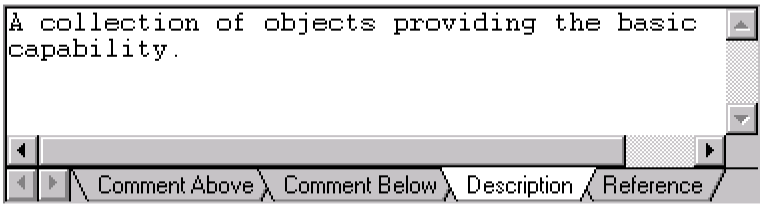
Object name	Object syntax
speakerCo...	INTEGER
speakerBe...	INTEGER
speakerTest	DisplayString
speakerBe...	Counter32
speakerUp...	TimeTicks
speakerTra...	INTEGER

Figure 44: Object Selection dialog box

6. To add a comment to the object, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.

7. To add a description of the object or any information that is needed to understand a MIB object or module, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Description*** tab. The text can be checked for spelling errors by using the ***Tools / Check spelling*** commands.

Tip: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools / Check spelling*** commands. For more information on spell checking, read the Check Spelling section of this manual.



```
A collection of objects providing the basic
capability.
```

Figure 45: Description of the “speakerBasicGroup” node

8. To add a reference, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is not obligatory for the OBJECT-GROUP construct.

9. After setting all parameters, press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View / Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit / Remove*** command. When removing a parent node all its child nodes will be removed as well.

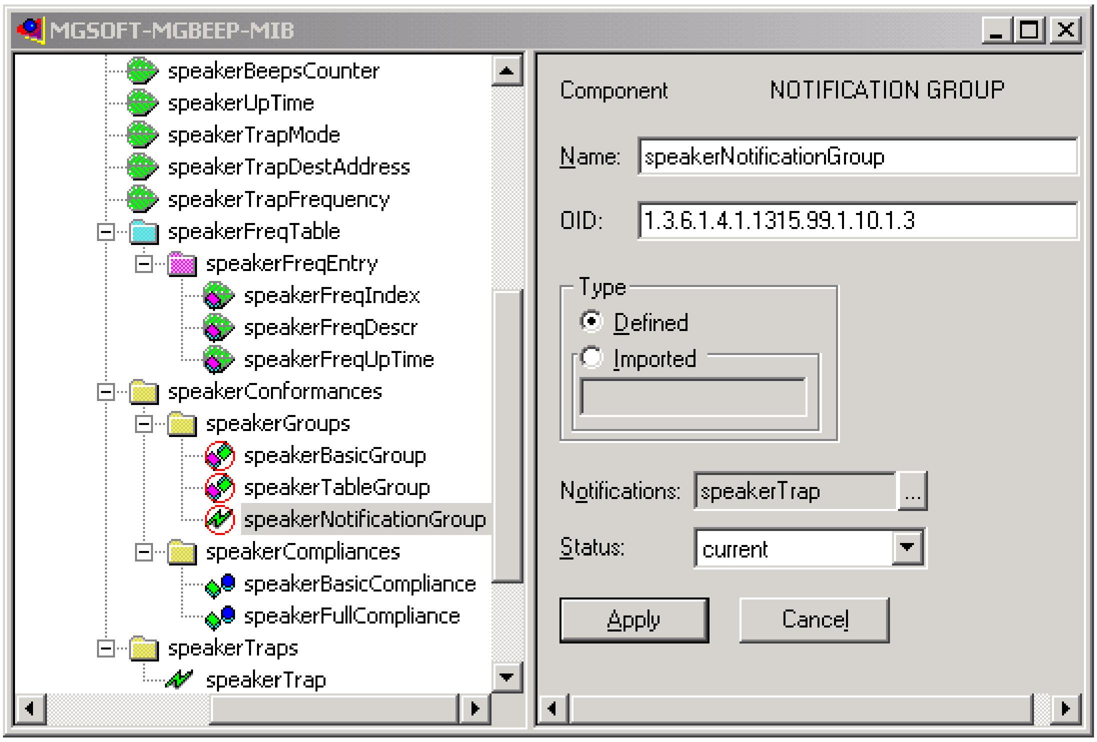
NOTIFICATION-GROUP Constructs

A NOTIFICATION-GROUP is an SMIv2 construct used to define a collection of related NOTIFICATION-TYPE definitions.

Note: Each NOTIFICATION-TYPE object must be a member of at least one notification group. To define a collection of related NOTIFICATION-TYPEs, the NOTIFICATION-GROUP construct is used.

The procedure of defining MIB objects with the NOTIFICATION-TYPE construct is described in the NOTIFICATION-TYPE Constructs section of this manual.

1. Once the NOTIFICATION-GROUP node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.



Component NOTIFICATION GROUP

Name: speakerNotificationGroup

OID: 1.3.6.1.4.1.1315.99.1.10.1.3

Type

● Defined

○ Imported

Notifications: speakerTrap ...

Status: current

Figure 46: NOTIFICATION-GROUP Properties window panel

2. Into the ***Name*** input line enter the name of the NOTIFICATION-GROUP node. The name must start with a lower case letter (e.g., `notificationGroup`) and must be unique for the current MIB module.

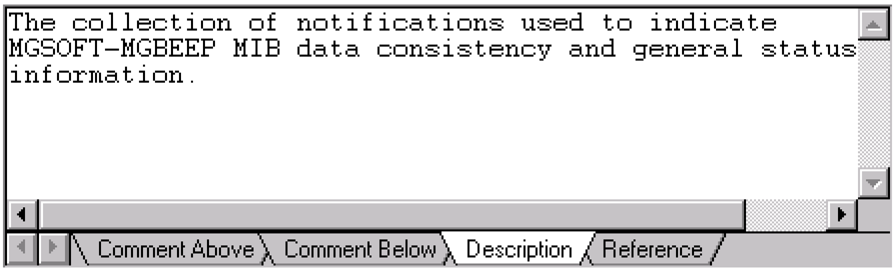
3. Click the ***Browse*** button next to the ***Notifications*** input line to open the Notifications Selection dialog box. From the ***Available notifications*** list, select the notification(s) you wish to add to the group and click the right arrow button. The selected notifications will appear in the ***Selected notifications*** list.

Tip: For more information on how to add a NOTIFICATION-GROUP construct, read the Adding Constructs Specific to SMIv2 to MIB Modules section.

4. From the ***Status*** drop-down list, select the value for the status clause.
   - `Current` - indicates that the object definition is valid.
   - `Deprecated` - indicates that the object definition is still valid in certain circumstances, but has been replaced by another.
   - `Obsolete` - indicates that the object definition is not valid

Note: The status of the NOTIFICATION-TYPE nodes specified in the NOTIFICATION-GROUP must conform to the status of the notification group definition.

5. To add a comment to the object, open the Additional Properties window (***View / Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.
6. To add a description of the object or any information that is needed to understand a MIB object or module, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Description*** tab. The text can be checked for spelling errors by using the ***Tools / Check spelling*** commands.



```
The collection of notifications used to indicate
MGSOFT-MGBEEP MIB data consistency and general status
information.
```

Figure 47: Description tab

7. To add a reference, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Reference*** tab. The Reference clause should contain a textual cross-reference to additional supporting documentation that may be of assistance in interpreting the rules or basis for the type. The reference clause is not obligatory for the NOTIFICATION-GROUP construct.
8. After setting all parameters, click the ***Apply*** button in the Properties window panel or the changes will not take effect.

Tip: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools / Check spelling*** commands. For more information on spell checking, read the Check Spelling section of this manual.

Tip 1: To view SMI definition of the selected node, open the Node Preview window (***View / Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 2: To remove a node from the MIB tree, select the node and use the ***Edit / Remove*** command. When removing a parent node all its child nodes will be removed as well.

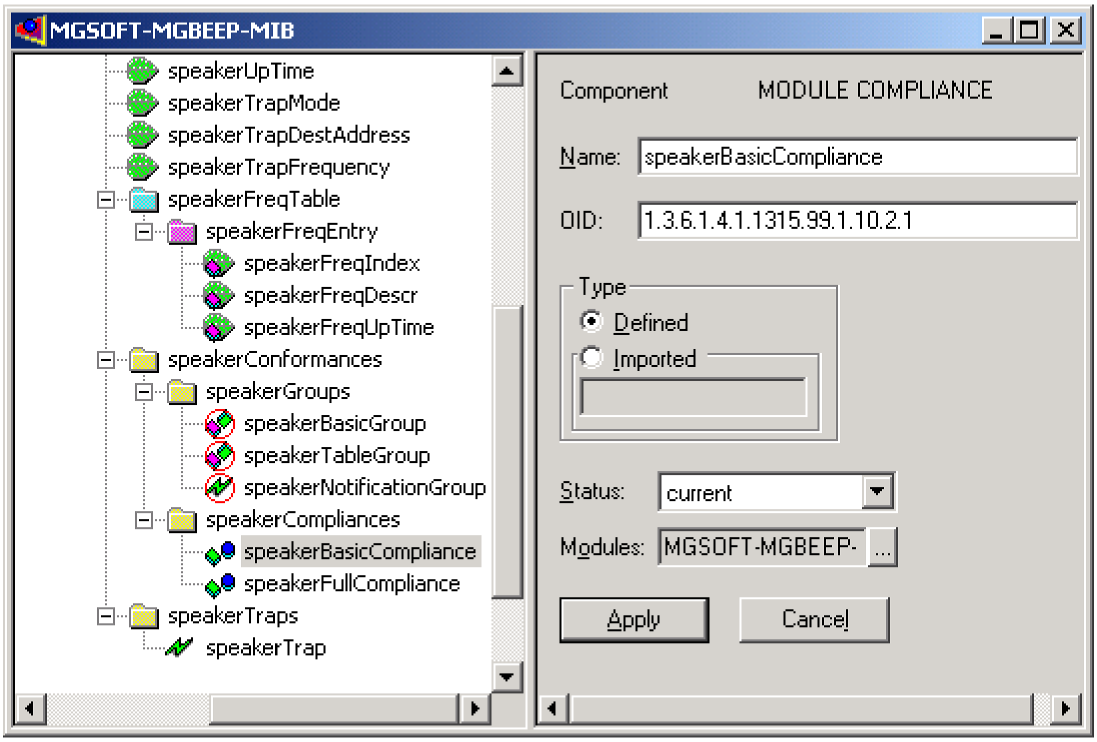
MODULE-COMPLIANCE Constructs

The MODULE-COMPLIANCE construct is an SMIv2 construct used for defining the implementation requirements specifications for an agent. The requirements specifications refer to the groups of object-types and/or events specified in MIB module(s) that the SNMP agent must implement to be compliant with the MIB module you are designing.

> **Note:** MODULE-COMPLIANCE construct may specify items from more than one MIB module.

1. Once the MODULE-COMPLIANCE node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.

   > **Tip:** For more information on how to add an MODULE-COMPLIANCE MIB object, read the Adding Constructs Specific to SMIv2 to MIB Modules section.



Component MODULE COMPLIANCE

Name: speakerBasicCompliance

OID: 1.3.6.1.4.1.1315.99.1.10.2.1

Type
● Defined
○ Imported

Status: current

Modules: MGSOFT-MGBEEP-

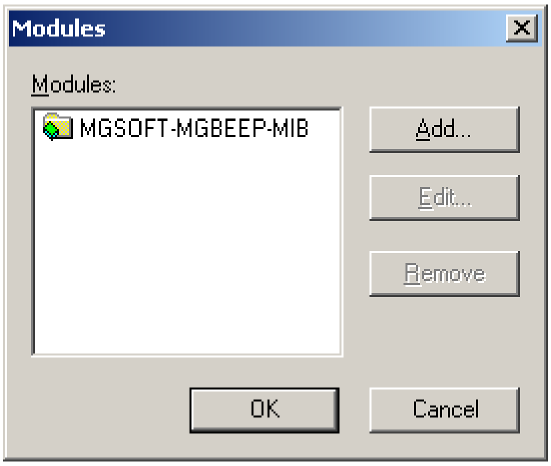
Figure 48: MODULE-COMPLIANCE Properties window panel

2. Into the ***Name*** input line enter the name of the MODULE-COMPLIANCE node. The name must start with a lower case letter (e.g., `basicCompliance`) and must be unique for the current MIB module.

3. From the ***Status*** drop-down list, select the value for the status clause.

   - `Current` - indicates that the object definition is valid.
   - `Deprecated` - indicates that the object definition is still valid in certain circumstances, but has been replaced by another.
   - `Obsolete` - indicates that the object definition is not valid

4. Next, click the ***Browse*** button (…) next to the ***Modules*** input line. The Modules dialog box appears (Figure 49). The Modules list specifies a list of MIB modules for which compliance requirements are being specified. To add or edit the details of compliance requirements, click the ***Add*** or ***Edit*** button and the Module Details dialog appears (Figure 50).

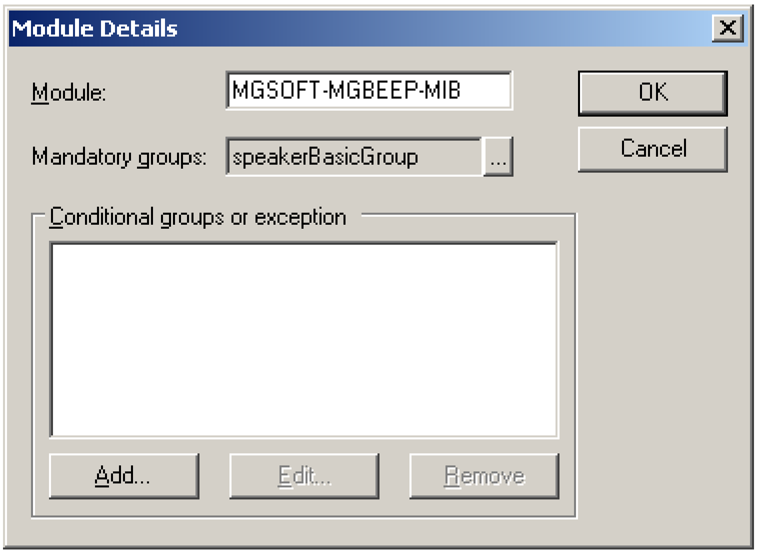


Modules: MGSOFT-MGBEEP-MIB

Figure 49: Modules dialog box

5. Into the ***Module*** input line specify the name of the MIB module in which groups of object-types and/or events are specified.

6. Click the ***Browse*** button next to the ***Mandatory groups*** input line to specify object and event groups, which are *unconditionally mandatory* for implementation. The Group Selection dialog box appears (Figure 56). Add the group(s) by selecting them in the ***Available groups*** list and clicking the right arrow button. When done, click the ***OK*** button to save the settings and close the dialog box.



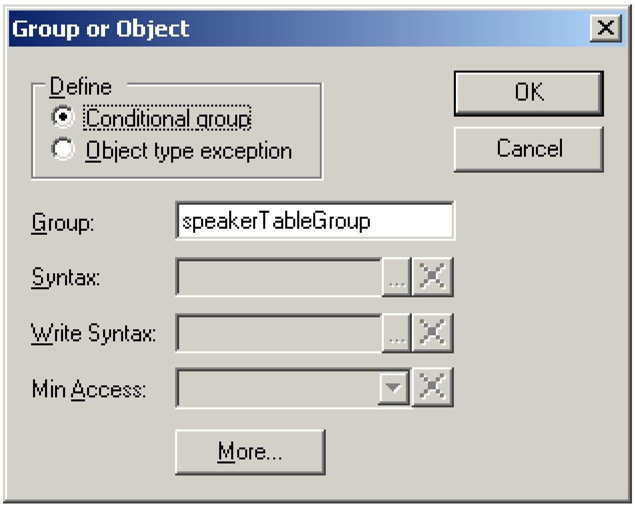
Module: MGSOFT-MGBEEP-MIB

Mandatory groups: speakerBasicGroup

Figure 50: Module Details dialog box

7. Click the ***Add*** button in the ***Conditional groups or exceptions frame*** to specify groups, which are conditionally required for implementation or/and to set the

restrictions in behavior of an object to meet the requirements specification. By clicking the ***Add*** button, the Group or Object dialog box appears (Figure 51).



Group or Object

Define
● Conditional group
○ Object type exception

Group: speakerTableGroup

Syntax:

Write Syntax:

Min Access:

Figure 51: Group or Object dialog box

Note: A group can be specified only in one Group clause, this means that the same group cannot be specified in the Mandatory groups clause and in a Conditional group clause nor can be the same group specified in two different mandatory or conditional group clauses.

8. In the ***Define*** frame, select:

   - ❑ `Conditional group` to specify object and/or event groups that are conditionally required for compliance to the MIB module.
   - ❑ `Object type exception` to specify the restrictions in behavior of an object to meet the requirements specification.

9. If specifying a conditional group, you have to enter the name of the group into the ***Group*** input line and describe the conditions that cause this group to be required in the Description dialog box (Figure 58). To open the ***Description*** dialog box, click the ***More*** button.

10. If specifying the restrictions in behavior of an object, enter the name of the object into the ***Object*** input line.

11. The purpose of ***Syntax*** and ***Write Syntax*** clauses is to specify a restriction in the implemented behavior of an object. The values specified for these clauses must be a reduction of the value for the Syntax clause in the object's definition (e.g., you can specify a subset of integer enumerations that must be implemented). Click the ***Browse*** button next to the ***Syntax*** (***Write Syntax***) input line to open the Syntax Details dialog box (Figure 52). From the ***Syntax*** drop-down list select the syntax type that matches the syntax with which the object is defined (e.g., to specify a restriction for the “scalarNodeX” object whose value of the ***Syntax*** clause is `Integer32`, you have to select the `Integer32` from the ***Syntax*** drop-down list in the Syntax Details dialog box).

Tip 1: The appearance of the Group or Object dialog box changes according to the selection made in the ***Define*** frame.

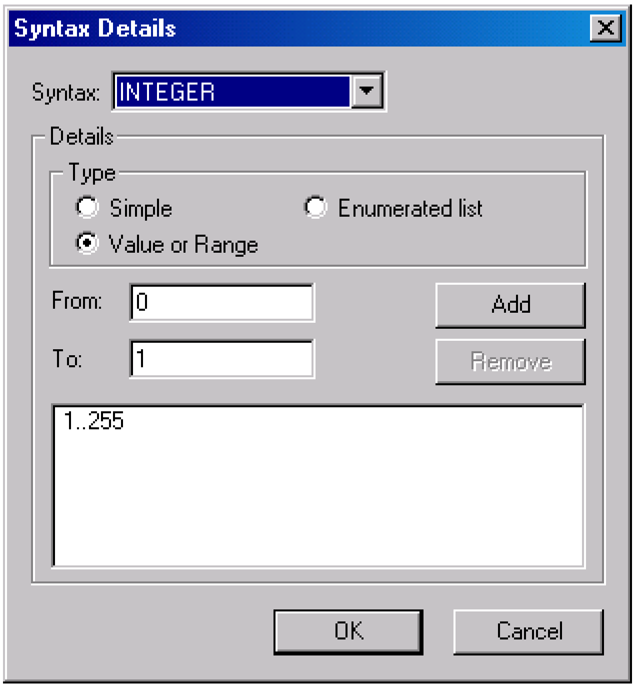
Tip 2: Use ☒ button to clear the set values from the ***Syntax***, ***Write Syntax*** and ***Access*** input lines.

Tip 3: The text in the Description dialog box can be checked for spelling errors by clicking the ***Spelling*** button. For more information on spell checking, read the Check Spelling section of this manual.

12. The appearance of the Details frame changes according to the selected syntax type.

   - ❑ If you select ***Simple***, the default integer range will be used (without restrictions).

- If you select ***Enumerated List***, you have to define a variable for a particular value. (e.g., `Value - 1, Label – on`). The enumerated list specified here can contain only a subset of enumerations that were originally defined for this object (no new enumerations can be defined).
- If you select ***Value or Range***, you have to specify the values and/or ranges, which are a subset of the values and/or ranges that are defined for this object's syntax.



Syntax: INTEGER

Type: ○ Simple ○ Enumerated list ● Value or Range

From: 0

To: 1

1..255

Figure 52: Syntax Details dialog box

13. After specifying the Syntax or Write Syntax clause, click the ***More*** button in the Group or Object dialog box to open the Description dialog box (Figure 58) where you have to describe restrictions in behavior of an object.
14. From the ***Min Access*** drop-down list select the value for the minimum access, which determines whether the object needs to be implemented and which part of the behavior can be omitted when the object needs to be implemented.
    - not-accessible – an object or object group does not need to be implemented.
    - access-for-notify – an object needs to be implemented in a way to be accessible only for event report operations (SNMP Trap or Inform).
    - read-only – an object needs to be implemented only to retrieve values of the instance(s).
    - read-write – an object needs to be implemented to retrieve and modify values of the instance(s).
15. To save the settings and close the dialog box, click the ***OK*** button.
16. After setting all parameters, click the Apply button in the Properties window panel or the changes will not take effect.

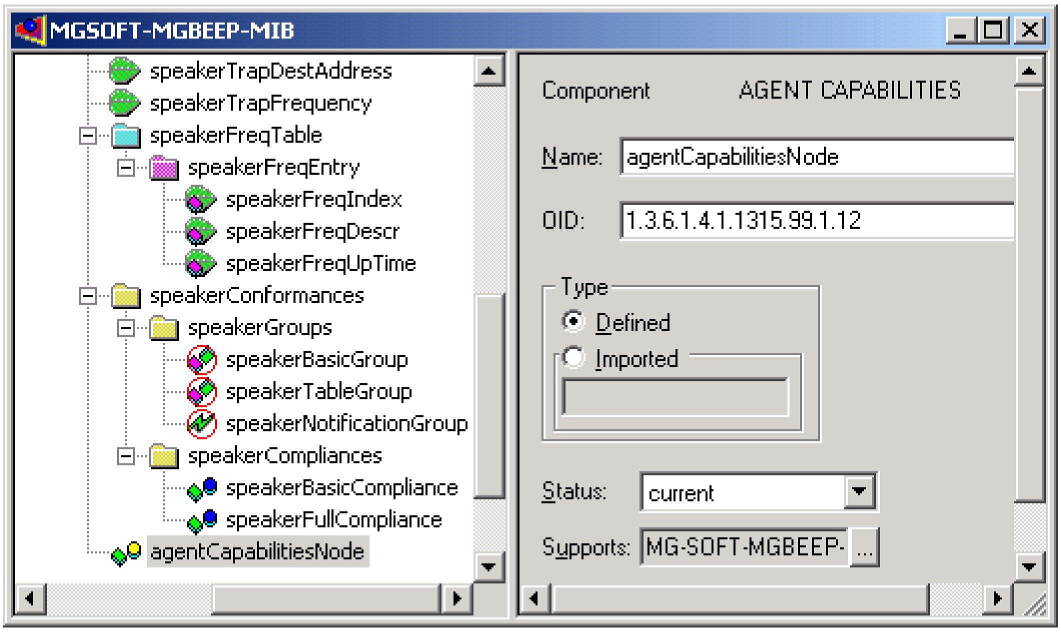
AGENT-CAPABILITIES Constructs

The AGENT-CAPABILITIES is an SMIv2 construct used for specifying the implementation characteristics of an SNMP agent sub-system with respect to object types and events.

> **Note:** One or more AGENT-CAPABILITIES constructs can be used to completely describe the implementation characteristics of an agent.

1. Once the AGENT-CAPABILITIES node is added to the MIB tree, click on the node and modify its default parameters in the Properties window panel.

> **Tip:** For more information on how to add an AGENT-CAPABILITIES MIB object, read the Adding Constructs Specific to SMIv2 to MIB Modules section.



Component AGENT CAPABILITIES

Name: agentCapabilitiesNode

OID: 1.3.6.1.4.1.1315.99.1.12

Type

● Defined

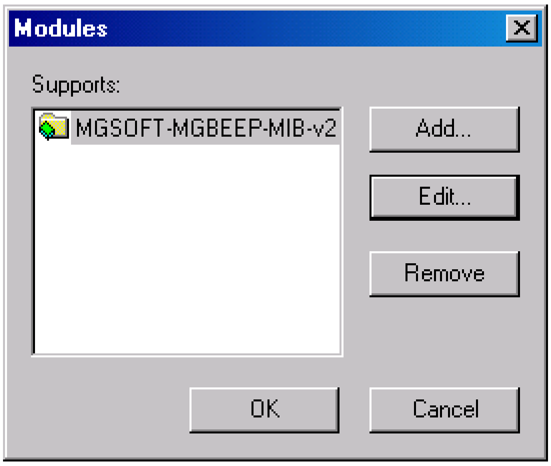
○ Imported

Status: current

Supports: MG-SOFT-MGBEEP- ...

Figure 53: AGENT-CAPABILITIES Properties window panel

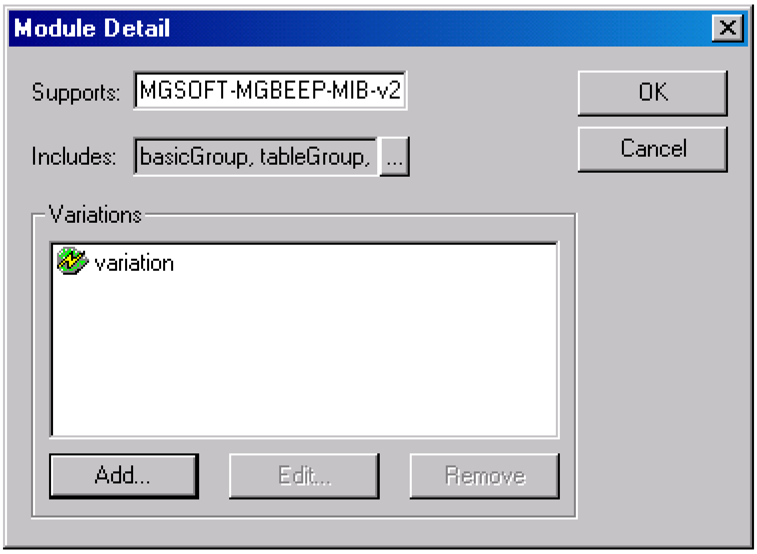
2. Into the ***Name*** input line enter the name of the AGENT-CAPABILITIES node. The name must start with a lower case letter and must be unique for the current MIB module.
3. From the ***Status*** drop-down list, select the value for the status clause.
   - `Current` - indicates that the object definition is valid.
   - `Deprecated` - indicates that the object definition is still valid in certain circumstances, but has been replaced by another.
   - `Obsolete` - indicates that the object definition is not valid
4. Click the ***Browse*** button (…) next to the ***Supports*** input line to specify the details of a module implementation. The Modules dialog box with an empty ***Supports*** list appears.



Supports: MGSOFT-MGBEEP-MIB-v2

Figure 54: Modules dialog box

5. The ***Supports*** list should list the MIB modules for which the agent claims a complete or partial implementation. To specify the MIB modules, click the ***Add*** button and specify the required module details in the Module Details dialog box. Into the ***Supports*** input line, enter the name of the MIB module. It should start with upper case letters. If no name is entered, Visual MIB Builder uses the name of the current MIB module.



Supports: MGSOFT-MGBEEP-MIB-v2

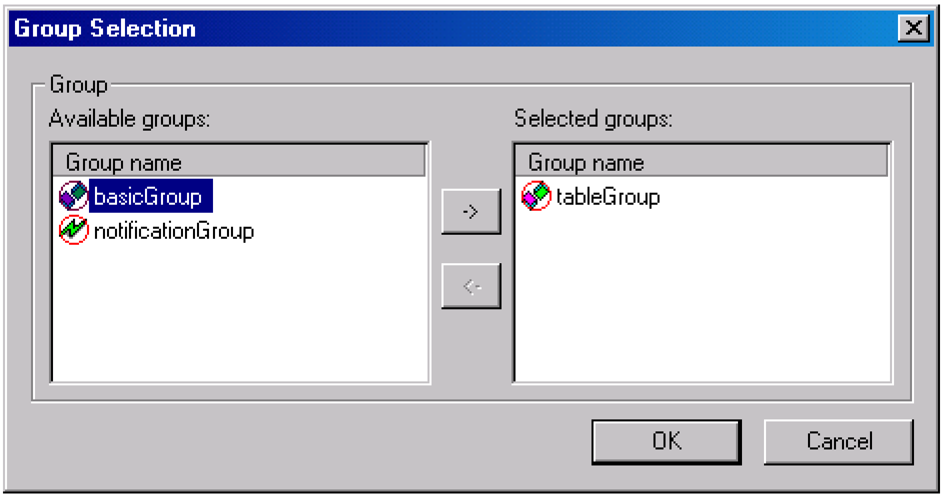
Includes: basicGroup, tableGroup,

Variations: variation

Figure 55: Module Details dialog box

Note: Unless specified in the ***Variation*** clause, an agent fully implements all the behaviors specified in the definitions of the objects and/or events that are members of the groups specified in the Includes clause.

6. Click the ***Browse*** button (…) next to the ***Includes*** input line to specify object and event groups associated with the MIB module that the agent claims to implement. The Group Selection dialog box appears. Add the group(s) by selecting them in the ***Available groups*** list and clicking the right arrow button. When done, click the ***OK*** button to save the settings and close the dialog box. The added groups will appear in the ***Includes*** input line.


Group Selection

Group

Available groups:

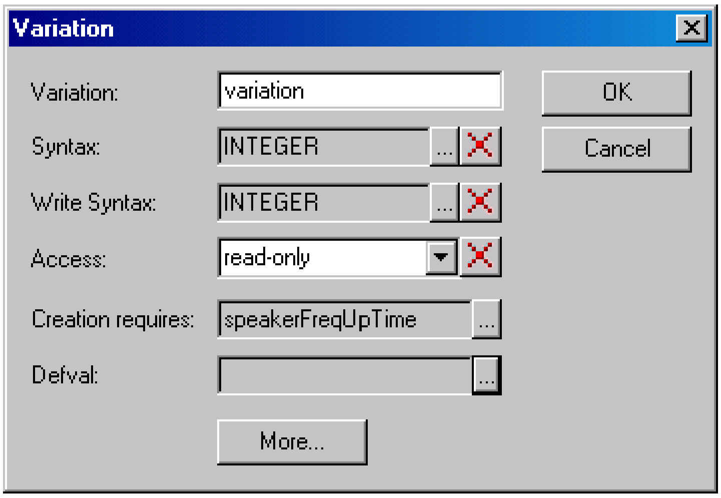
Group name
basicGroup
notificationGroup

Selected groups:

Group name
tableGroup

Figure 56: Group Selection dialog box

7. To specify that an event or object that is a member of a group listed in the ***Includes*** clause should behave differently than specified in its definition, or that an event or object is not implemented, use the Variation clause. Click the ***Add*** button in the ***Variation*** frame to open the Variation dialog box (Figure 57) where you can specify the details.
8. Into the ***Variation*** input line, enter the name of the object to which the variation refers.
9. The purpose of ***Syntax*** and ***Write Syntax*** clauses is to specify a restriction in the implemented behavior of an event or object. The values specified for these clauses must be a reduction of the value for the Syntax clause in the object's definition. Click the ***Browse*** button next to the ***Syntax*** (***Write Syntax***) input line to open the Syntax Details dialog box (Figure 52). From the Syntax drop-down list select the syntax type.


Variation

Variation: variation

Syntax: INTEGER

Write Syntax: INTEGER

Access: read-only

Creation requires: speakerFreqUpTime

Defval:

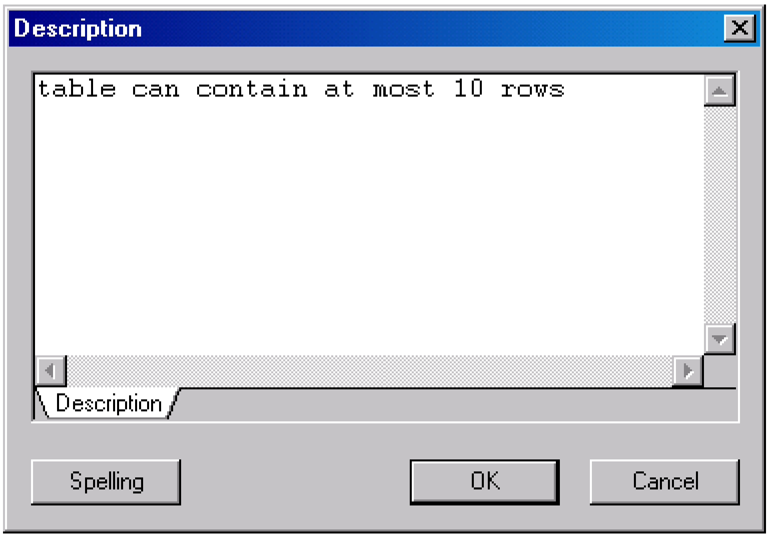
Figure 57: Variation dialog box

Tip: Use ☒ button to clear the set values from the ***Syntax***, ***Write Syntax*** and ***Access*** input lines.

10. The appearance of the ***Details*** frame changes according to the selected syntax type.
    - If you select ***Simple***, the default integer range will be used (without restrictions).

- If you select ***Enumerated List***, you have to define a variable for a particular value. (e.g., `Value - 1, Label – on`). The enumerated list specified here can contain only a subset of enumerations that were originally defined for this object (no new enumerations can be defined).
- If you select ***Value or Range***, you have to specify the values and/or ranges, which are a subset of the values and/or ranges that are defined for this object's syntax.

11. After specifying the Syntax clause, click the ***More*** button to open the Description dialog box (Figure 58) where you have to describe restrictions in behavior of an object.

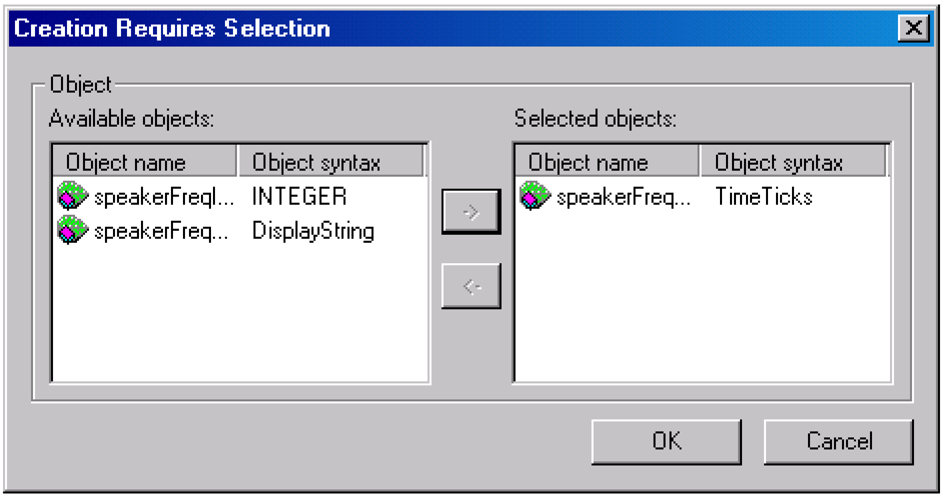


```
table can contain at most 10 rows
```

Figure 58: Description dialog box

Tip: The text in the Description dialog box can be checked for spelling errors by clicking the ***Spelling*** button. For more information on spell checking, read the Check Spelling section of this manual.

12. From the ***Access*** drop-down list select the value for the access, which determines which part of the behavior can be omitted and whether or not an event or object needs to be implemented.
    - `access-for-notify` – an object needs to be present only to return instance of an event.
    - `not-implemented` – an object needs not to be implemented.
    - `read-only` – an object needs to be present only to retrieve values of the instance.
    - `read-write` – an object needs to be present to modify and retrieve the existing values of the instance.
    - `write-only` – an object needs to be present only to modify the existing values of the instance.
13. The Creation requires clause is used to document the minimum set objects a manager must include in the single SNMP Set operation in order for agent to create an instance of a row in a table. Click the ***Browse*** button next to the ***Creation requires*** input line. The Creation Requires Selection dialog box appears. Add the object(s) by selecting them in the ***Available objects*** list and clicking the right arrow button. When done, click the ***OK*** button to save the settings and close the dialog box. The selected objects will appear in the ***Creation requires*** input line.



Creation Requires Selection

Object

Available objects:

Object name	Object syntax
speakerFreqI...	INTEGER
speakerFreq...	DisplayString

Selected objects:

Object name	Object syntax
speakerFreq...	TimeTicks

Figure 59: Creation Requires Selection dialog box

14. The Defval clause specifies a default value that an agent will use when an instance of a columnar object is created, instead of the default value that is specified in the definition of the object. Click the ***Browse*** button next to the ***Defval*** input line and the Default Value dialog box appears. Specify the details of the default value in ***Default Value*** input line and click the ***OK*** button.
15. To add a comment to the object, open the Additional Properties window (***View*** / ***Additional Properties***) and enter or edit the comment in the ***Comment Above*** and ***Comment Below*** tabs.
16. To add a description of the object or any information that is needed to understand a MIB object or module, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Description*** tab.
17. To add a reference, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Reference*** tab. Reference specifies the source of the definition. It may refer to a document from another standards organization, or an architectural for a proprietary system. The reference clause is optional.
18. The ***Product Release*** tab (***View*** / ***Additional Properties***) is used to describe the product release, which includes the implemented capabilities.
19. After setting all parameters, press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

Tip 1: The text in any tab of the Additional Properties window can be checked for spelling errors by using the ***Tools*** / ***Check Spelling*** commands. For more information on Spell checking read the Check Spelling section of this manual.

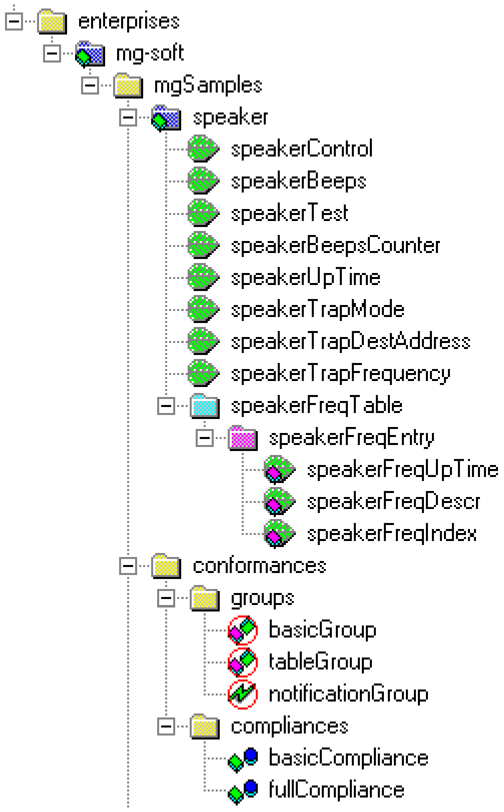
Tip 2: To view the selected node in the MIB module definition language, open the Node Preview window (***View*** / ***Node Preview***) and select the ***SMI Node Preview*** tab.

Tip 3: To remove a node from the MIB tree, select the node and use the ***Edit*** / ***Remove*** command. When removing a parent node, all its child nodes will be removed as well.

6 EDITING EXISTING MIB MODULES

6.1 Changing Node Location

As mentioned in previous sections, in Visual MIB Builder application, each MIB module is visually represented in hierarchically arranged tree structure (MIB tree).

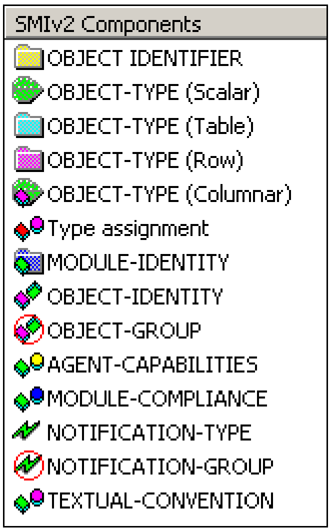


```
enterprises
  mg-soft
    mgSamples
      speaker
        speakerControl
        speakerBeeps
        speakerTest
        speakerBeepsCounter
        speakerUpTime
        speakerTrapMode
        speakerTrapDestAddress
        speakerTrapFrequency
        speakerFreqTable
          speakerFreqEntry
            speakerFreqUpTime
            speakerFreqDescr
            speakerFreqIndex
      conformances
        groups
          basicGroup
          tableGroup
          notificationGroup
        compliances
          basicCompliance
          fullCompliance
```

Figure 60: MIB tree

Tip: Use the ***File* / *Print*** command to print the MIB tree. For print preview, choose the ***File* / *Print Preview*** command.
By default, the printed MIB tree displays only names of MIB tree nodes. To specify additional information that the MIB tree nodes should display when printed, use the ***View* / *Preferences*** command, switch to the Options tab and enable desired options in the MIB Tree Printing frame.

Individual items (MIB objects) of the MIB module are in the MIB tree represented with different icons based on the type of the management information. Visually represented MIB objects are called nodes.



SMIv2 Components

- OBJECT IDENTIFIER
- OBJECT-TYPE (Scalar)
- OBJECT-TYPE (Table)
- OBJECT-TYPE (Row)
- OBJECT-TYPE (Columnar)
- Type assignment
- MODULE-IDENTITY
- OBJECT-IDENTITY
- OBJECT-GROUP
- AGENT-CAPABILITIES
- MODULE-COMPLIANCE
- NOTIFICATION-TYPE
- NOTIFICATION-GROUP
- TEXTUAL-CONVENTION

Figure 61: SMIv2 MIB object icons listed the Components window

Visual MIB Builder application lets you use the drag-and-drop technique to move a node within the MIB tree in order to change its position.

As you drag the node over the MIB tree, Visual MIB Builder indicates where the node will be inserted when you release the mouse button. There are two possible cases:

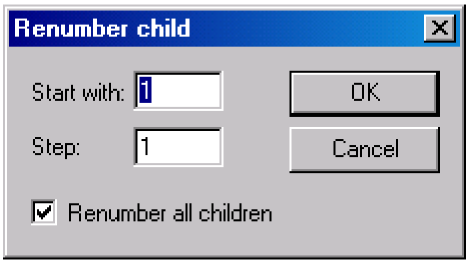
- ❑ The target node is selected. That means that the node you are about to drop will be added as the last child node of the selected node.
- ❑ A line is drawn between two nodes. That means that the node you are about to drop will be inserted between these two nodes.

Instead of using the drag-and-drop technique, you can also select a node and use the ***Edit*** / ***Copy*** command. This command will place the selected node to the clipboard. Then select a node under which you wish to insert the copied node and use the ***Edit*** / ***Paste*** command. Although this action behaves identically as the drag-and-drop technique, there are some limitations. First, there is no visual feedback on whether the node can be inserted at the desired location, and second, the node can only be inserted as the last child node of the target node.

When dropping the node to the desired position, Visual MIB Builder automatically assigns an OID value to that node based on the node location in the MIB tree. You can edit the last sub-identifier in the ***OID*** input line in the Properties window panel.

Modifying the last sub-identifier in the ***OID*** input line will update the object OID value and automatically move the node to the location that corresponds to the new OID (provided that no node with such OID value exists in the MIB tree yet). Use the ***Renumber child*** pop-up command to set all child nodes of a particular parent node in the right order after removing or moving a node.

1. First, select the parent node to which the new node was added and then use the ***Renumber child*** pop-up command. The Renumber Child dialog box appears.



Start with: 1

Step: 1

☑ Renumber all children

Figure 62: Renumber child dialog box

2. Into the ***Start with*** input line, enter the number for the first child node in the sub tree. This number indirectly determines the position of the child node, because it is the last sub-identifier of the object OID. Note that it is not necessary for the first child node to have the number 1, but it has to have the lowest number of all child nodes.
3. In the ***Step*** input line, specify the step size used for renumbering the last sub-identifiers of the child nodes. For example, if you enter number 1, then the last sub-identifier of each following child node will increase for one.
4. If the ***Renumber all children*** checkbox is checked, all child nodes of the selected node as well as their child nodes will be renumbered. If this checkbox is empty, only the child nodes of the selected node will be renumbered. For better understanding, please refer to the example below.

Example on how to renumber children of a particular node:

1. Open a new module by clicking the ***New*** toolbar button.
2. Select the OBJECT IDENTIFIER component in the Components window and drag-and-drop it onto the root node in the MIB tree.
3. Next, select the OBJECT-TYPE (Scalar) component and drag-and-drop it onto the OBJECT IDENTIFIER node (`node0`). Repeat the same procedure one more time, so that you will have two scalar nodes in the MIB tree. MIB builder will automatically assign their names and OID values.
4. Now lets crate a table. Select the OBJECT-TYPE (Table) component in the Components window and drag-and-drop it onto the IDENTIFIER node (`node0`). A table node with one row and one columnar node will be added as the last child node to the MIB tree. Add two more tabular nodes by selecting the OBJECT-TYPE (Tabular) in the Components window and then drag-and-drop the nodes onto the table node (`node4Table`). Your MIB tree should now look like this:

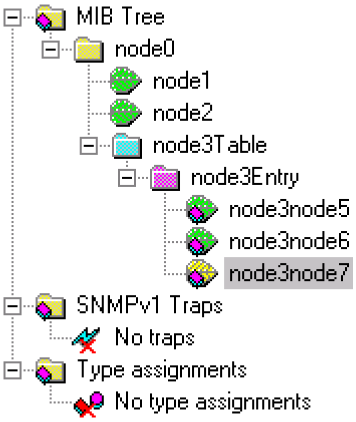
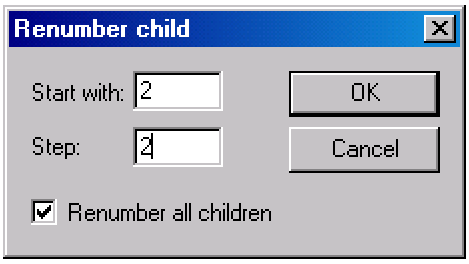


Figure 63: MIB tree

5. If you select a node in the MIB tree, you can view its properties in the Properties window panel. Select the `node1` and check its OID value. It should be `1.1` and `node2` should have OID value `1.2.` Check also the OIDs of columnar objects. `Node3node5` should have OID value `1.3.1.1` and OID value assigned to the `node3node7` columnar node should be `1.3.1.3.`

6. Now select the OBJECT IDENTIFIER node (`node0`) and use the ***Renumber child*** pop-up command. The Renumber child dialog box appears.

7. To renumber all children of the `node0` in the way that that the first child node has the number 2 and that OID numbers increase for 2, enter 2 into the ***Start with*** input line and 2 into the ***Step*** input line. Check the ***Renumber all children*** dialog box and click the ***OK*** button.



Start with: 2

Step: 2

☑ Renumber all children

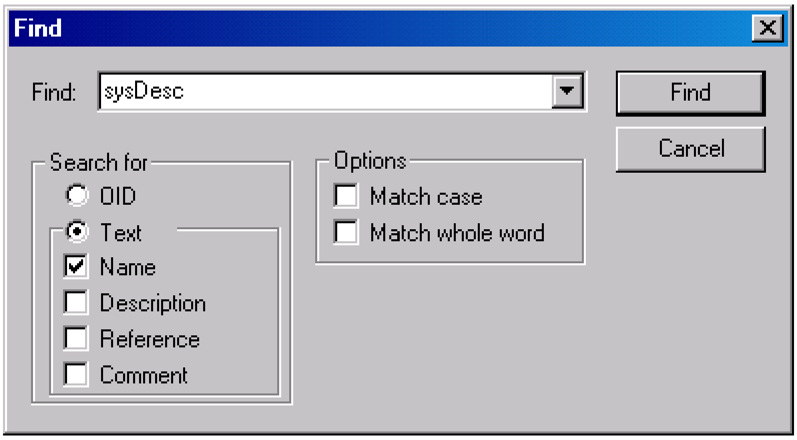
Figure 64: Renumber child dialog box

8. Now check how the OID values of MIB nodes have changed. As you can see the OID values of all nodes in the MIB tree were modified. The OID value of the first child node (`node1`) is now `1.2` and the OID value of the second child changed to `1.4.` Check also the OIDs of columnar objects. `Node3node5` should have OID value of `1.6.1.2` and columnar node `node3node7` should have OID value of `1.6.1.6.`

6.2 Finding Nodes

Visual MIB Builder lets you find nodes in the MIB tree that match the given conditions. A condition can be either OID or text (e.g., node name, description, etc.).

1. To specify the search conditions, open the Find dialog box by using ***Edit / Find*** command.



Find: sysDesc

Search for
○ OID
● Text
☑ Name
□ Description
□ Reference
□ Comment

Options
□ Match case
□ Match whole word

Figure 65: Find dialog box

Tip: You can also use the ***Ctrl+F*** keyboard shortcut or the **Find** pop-up command to open the Find dialog box.

2. In the ***Search for*** frame select whether you wish to search for a node by its ***OID***, ***Name***, ***Description***, ***Reference***, or ***Comment***.
3. After selecting an option in the ***Search for*** frame, enter the OID or text you want to search for into the ***Find*** drop-down list.
4. In the ***Options*** frame select
   - ***Match case*** if you wish Visual MIB Builder to match case while searching for the specified text.
   - ***Match whole word*** if you wish Visual MIB Builder to match the entire name while searching for the specified text.

Note: The ***Options*** frame is enabled only if the ***Text*** radio button is selected in the ***Search for*** frame.

5. After specifying the search conditions, click the ***Find*** button. Visual MIB Builder will search the MIB tree from the root node downwards and select the first matching node. The dialog box will be automatically closed.
6. The next matching node can then be found by using the ***Edit / Find Next*** command or by pressing the ***F3*** keyboard key. Note that ***Find Next*** command starts searching the MIB tree from the currently selected node.

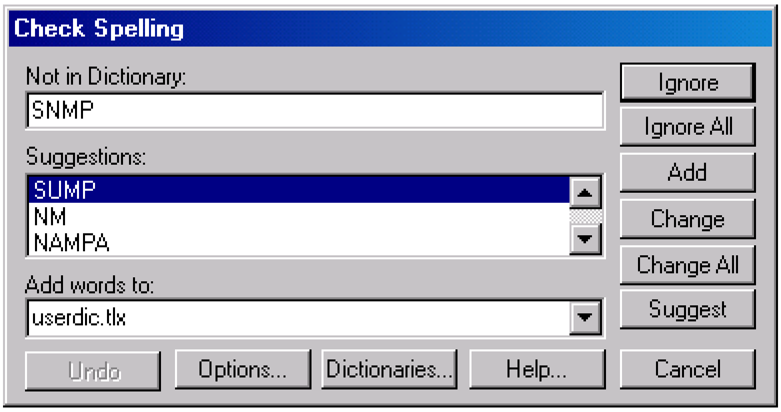
6.3 Spell Checking

The text in Additional Properties window tabs and in the Description dialog box can be checked for spelling errors.

1. To spell check the text in the Additional Properties tabs, use the ***Tools*** / ***Check Spelling*** command. You can select from the following two options:
   - ***Active tab.*** This option checks the active tab in the Additional Properties window for possible spelling mistakes and displays suggestions for correcting them in the Check Spelling dialog box.
   - ***All tabs.*** This option checks all tabs in the Additional Properties window for possible spelling mistakes and displays suggestions for correcting them in the Check Spelling dialog box.

Tip: Read the AGENT-CAPABILITIES Constructs and MODULE-COMPLIANCE Constructs sections to learn more about the Description dialog box.

2. The Check Spelling dialog box appears if a word requiring your attention is detected during the spell checking process. You can use the dialog box to specify whether the word should be ignored or replaced.



Check Spelling

Not in Dictionary: SNMP

Suggestions: SUMP, NM, NAMPA

Add words to: userdic.tlx

Figure 66: Check Spelling Dialog box

Note: The labels of some buttons and input lines in the dialog box change according to the context. For more information on that, refer to the Check Spelling dialog box section of the Visual MIB Builder help file, which can be accessed from the Help menu.

3. The misspelled word is displayed in the ***Not in Dictionary*** input line. The word is considered misspelled because it could not be located in any open dictionary, or was marked with an exclude action. You can edit the word in this input line or select a suggestion from the ***Suggestions*** drop-down list, then click the ***Change*** button to correct the word, or the ***Ignore*** button to skip the word. If you choose to skip the

Tip: To replace this and all following occurrences of the reported use the ***Change All*** button.

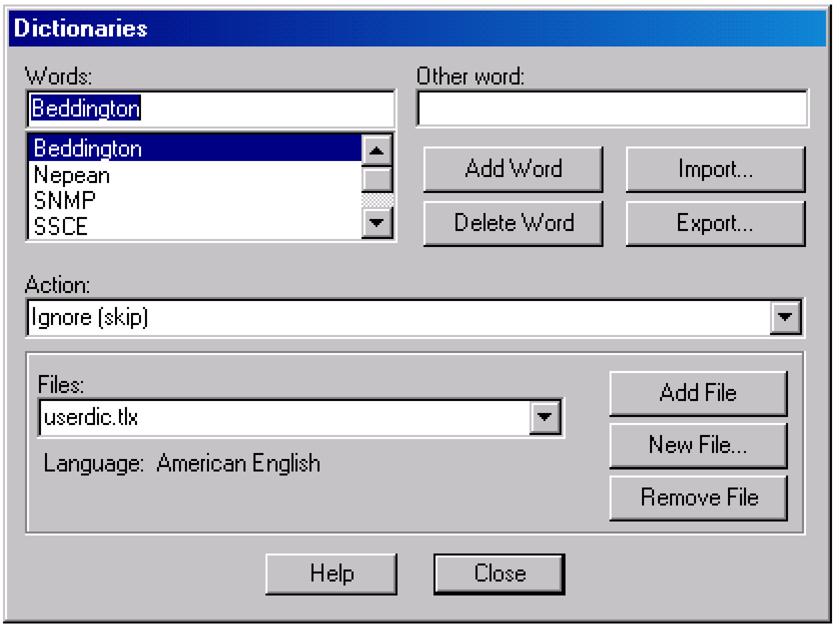
word and the same misspelled word appears later in the text, it will be reported again. To avoid this, use the ***Ignore All*** button instead.

4. The ***Suggest*** button will perform more thorough search for suggested replacements for the current misspelled word. Each time you press the ***Suggest*** button, a "deeper" search is made. The ***Suggest*** button is disabled once all possible suggestions have been located.

5. Use the ***Add*** button if a correctly spelled word you use often is reported as a misspelling (e.g., your family name). The reported word will be added to the dictionary selected in the ***Add Words To*** drop-down list. This button is enabled only if a user dictionary has been selected in the ***Add Words To*** drop-down list.

6. The ***Add Words To*** drop-down list shows all user dictionaries currently open. You can open or close other dictionaries via the Dictionaries dialog box (Figure 67), which is accessible by clicking the ***Dictionaries*** button.

Note: For more information on how to open and close user dictionaries, and how to edit the contents of an open user dictionary, please refer to the Dictionaries dialog box section of the Visual MIB Builder help file, which can be accessed from the Help menu.



Dictionaries

Words: Beddington

Beddington
Nepean
SNMP
SSCE

Other word:

Action: Ignore (skip)

Files: userdic.tlx

Language: American English

Figure 67: Dictionaries dialog box

7. You can use the ***Options*** button to open the Options dialog box where you can set the Check Spelling options.

8. Clicking the ***Undo*** button will remove the last change made. The ***Undo*** button can be clicked several times to remove the last several changes.

Note: For more information on how to modify spelling-checker options, please refer to the Options dialog box section of the Visual MIB Builder help file, which can be accessed from the Help menu or by clicking the ***Help*** button in the Options dialog box.

6.4 Revising MIB Modules

Once a MIB module has been published, it becomes a contract between agent developers and management application developers. Changing or modifying a MIB module can cause problems for both sides. However, all MIB modules need to be revised over a time. Whatever the reason, you may sometimes wish to change a MIB structure, delete or add some object to a MIB, or change the way objects in the MIB behave.

MG-SOFT Visual MIB Builder application enables you to modify MIB modules through the Revisions feature. You can perform any number of revisions, view all so far existing revisions, compare revisions and remove them.

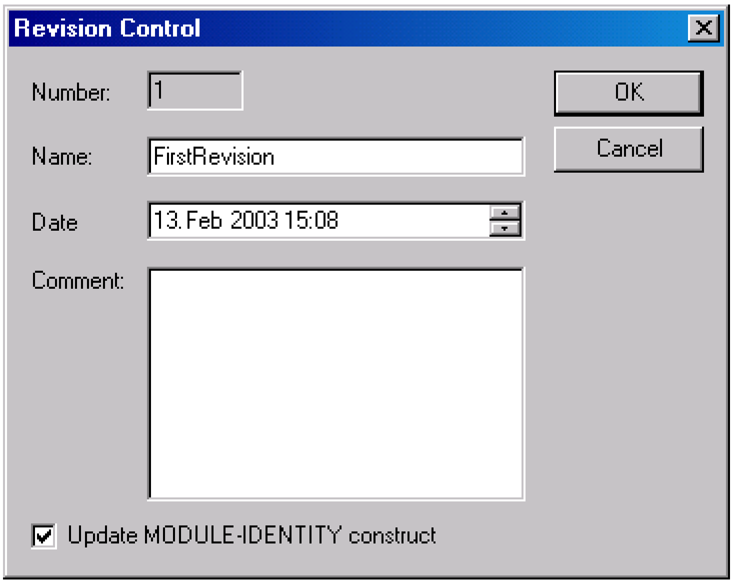
When a MIB module is in the revision mode, you can add a new object to a MIB module and modify some of its properties, but with limitations. For example, you can add new objects without any restrictions as long as you are not adding items to tables. You can also add or update optional clauses in object definitions, update the Description clause (only for editorial reasons), but you may not modify an item in the way that could change its behavior. For more information on module maintenance, see the Understanding SNMP MIBs book (Perkins & McGinnis, Prentice Hall, ISBN 0-13-437708-7).

6.4.1 New Revision

1. After modifying a MIB module, select the ***Tools*** / ***Revisions*** / ***Add*** command. The Revision Control dialog box appears (Figure 68). If no changes have been made to a MIB module since the last revision, the Add Revision message box appears, prompting you to confirm the ***Add*** command.

   Note: If this is the first revision, the ***Add Initial*** command is available instead of the *Add* command.

2. By default, Visual MIB Builder automatically updates the MODULE-IDENTITY construct properties with the current revision data and adds a new revision to the ***Revisions*** list. To disable this function, uncheck the ***Update MODULE-IDENTITY construct*** checkbox in the Revision Control dialog box.
3. In the Revision Control dialog box, specify the required information about the revision and click the ***OK*** button. A lock icon will appear next to each node in the MIB tree, indicating that the item is locked for editing.



Revision Control

Number: 1

Name: FirstRevision

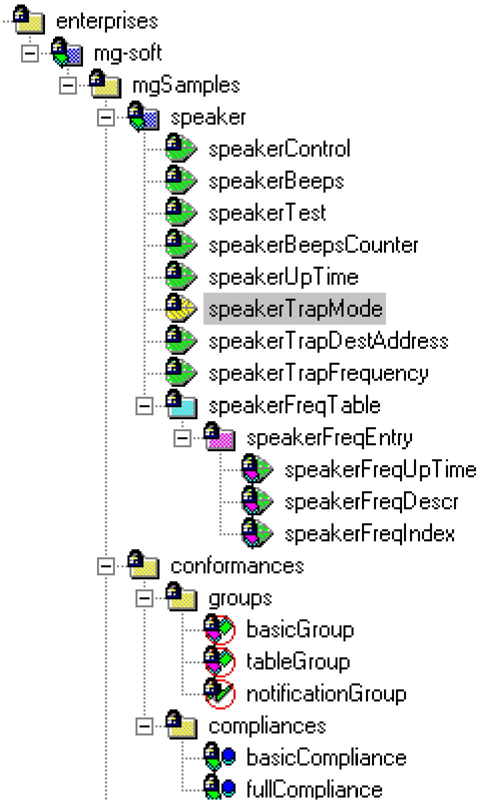
Date 13. Feb 2003 15:08

Comment:

☑ Update MODULE-IDENTITY construct

Figure 68: Revision Control dialog box

4. If you apply any changes to a module that has already been locked, the changed item will be colored green. Note that you can change a locked MIB module only in the currently active revision.

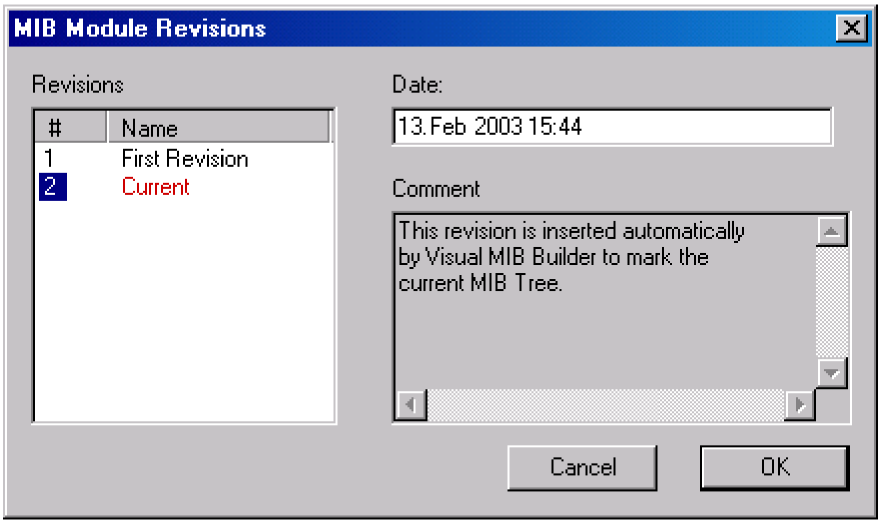


```
enterprises
  mg-soft
    mgSamples
      speaker
        speakerControl
        speakerBeeps
        speakerTest
        speakerBeepsCounter
        speakerUpTime
        speakerTrapMode
        speakerTrapDestAddress
        speakerTrapFrequency
        speakerFreqTable
          speakerFreqEntry
            speakerFreqUpTime
            speakerFreqDescr
            speakerFreqIndex
      conformances
        groups
          basicGroup
          tableGroup
          notificationGroup
        compliances
          basicCompliance
          fullCompliance
```

Figure 69: Locked MIB module

6.4.2 Switching Between Revisions

1. Select the ***Tools*** / ***Revisions*** / ***Switch to*** command. The MIB Module Revisions dialog box appears.



Revisions

#	Name
1	First Revision
2	Current

Date: 13.Feb 2003 15:44

Comment

This revision is inserted automatically by Visual MIB Builder to mark the current MIB Tree.

Figure 70: MIB Module Revisions dialog box

2. The ***Revisions*** list displays all so far existing revisions. The currently viewed revision is displayed in red. Select a revision from the list and click the ***OK*** button.
3. Visual MIB Builder will switch to the selected revision. The name and consecutive number of the selected revision will be displayed in the title bar of the MIB Builder window.
4. Optionally, you can also switch between revisions by changing the number of the revision in the drop-down list in the toolbar. The number that is currently displayed in this drop-down list shows which revision you are currently viewing.

Note: MIB modules can be edited only in the currently active revision. When viewing precedent revisions, modules are fully locked for editing and can only be viewed.

6.4.3 Comparing Revisions

1. Select the ***Tools*** / ***Revisions*** / ***Compare command***. The MIB Module Revisions dialog box appears (Figure 70). Make sure that you are currently viewing the revision that you wish to compare to another revision. To switch to another revision, read the section above.
2. The ***Revisions*** list displays all revisions that were performed before the currently viewed revision. A revision can namely be compared only to a precedent revision.
3. Select a revision from the list and click the ***OK*** button.
4. The compared revisions can be viewed in the MIB tree window panel in the MIB Builder window. Nodes that match are written in black color, nodes that have been

changed in the later revision are written in blue color and red color indicates the nodes that appear only in one of both compared revisions.

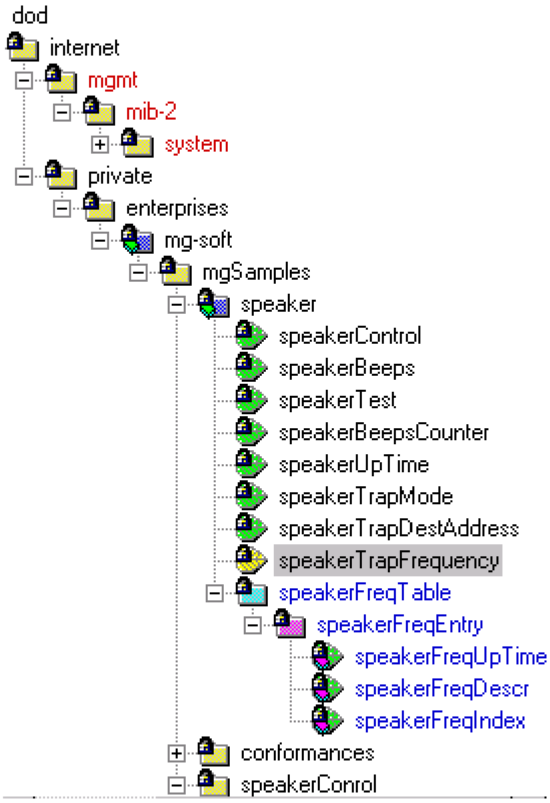


Figure 71: Compared Revisions

5. To exit this mode, select the ***Tools*** / ***Revisions*** / ***Exit compare*** command.

6.4.4 Removing Revisions

1. Select the ***Tools*** / ***Revisions*** / ***Remove command***. The Remove MIB module info message box appears prompting you to confirm the deletion.
2. The MIB Module Revisions dialog box appears (Figure 70). The ***Revisions*** list displays all so far existing revisions.
3. Select a revision you wish to remove from the list and click the ***OK*** button.

Note: Removing a revision will also remove all following revisions and all items that were locked by that revision will be unlocked.

7 CONVERTING MIB MODULES

Visual MIB Builder enables you to convert the SMIv1 MIB modules into the SMIv2 MIB modules and vice-versa.

7.1 Converting SMIv2 MIBs into SMIv1 MIBs

1. To convert a SMIv2 MIB into the SMIv1 MIB, first open the SMIv2 MIB module to be converted by using the ***File / Open*** command, and then use the ***Tools / SMI Conversion*** command.
2. The MIB module in the currently active MIB Builder window will be automatically converted according to the options set in the in the Preferences dialog box, Convert tab (**SMIv2 to SMIv1** frame and the **Imports** dialog box).
3. Converted MIB module will appear in a new MIB Builder window. By default, the converted MIB will get the name of the original MIB module, appended by the `-V1` suffix, which indicates the SMI version of the module (e.g., `MGSOFT-MGBEEP-MIB-V1`). To change the name of the module, enter a different name into the ***Module name*** input line in the Properties window panel.

> **Note:** The MIB module will be converted according to the options set in the Convert tab | SMIv2 to SMIv1 frame in the Preferences dialog box. For more information on how to set conversion options, refer to the Convert Tab section of this manual.

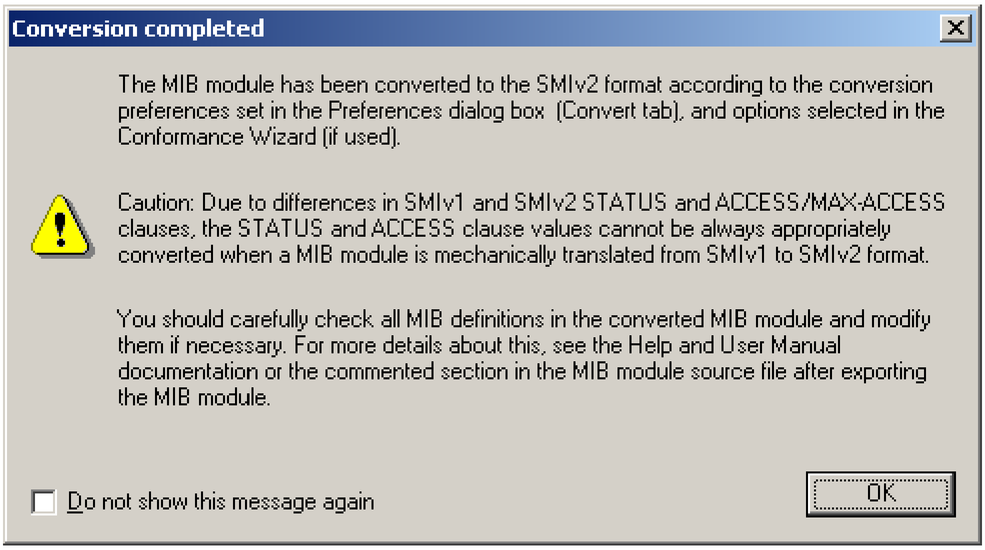
7.2 Converting SMIv1 MIBs into SMIv2 MIBs

1. To convert a SMIv1 MIB into the SMIv2 MIB, first open the SMIv1 MIB module to be converted by using the ***File / Open*** command and then use the ***Tools / SMI Conversion*** command.

> **Note 1:** By default, when converting SMIv1 MIBs into SMIv2 MIBs, the Conformance Wizard is started after the ***Tools / SMI Conversion*** command is used. To disable the conformance wizard, uncheck the ***Create conformance*** checkbox in the Preferences dialog box | Convert tab | SMIv1 to SMIv2 frame. For more information, read the Conformance Wizard section of this manual.
>
> **Note 2:** The MIB module will be converted according to the options set in the ***Convert tab | SMIv1 to SMIv2*** frame in the Preferences dialog box. For more information on how to set conversion options, refer to the Convert Tab section of this manual.

2. The MIB module in the currently active MIB Builder window will be automatically converted according to the options set in the Convert tab | SMIv1 to SMIv2 frame in the Preferences dialog box.
3. After conversion, the converted MIB module appears in a new MIB Builder window and, by default, the Conversion Completed dialog box is displayed. This message box informs you that due to differences in SMIv1 and SMIv2 STATUS and ACCESS/MAX-ACCESS clauses, the STATUS and ACCESS clause values cannot

be always appropriately converted when a MIB module is mechanically translated from SMIv1 to SMIv2 format.



Conversion completed

The MIB module has been converted to the SMIv2 format according to the conversion preferences set in the Preferences dialog box (Convert tab), and options selected in the Conformance Wizard (if used).

Caution: Due to differences in SMIv1 and SMIv2 STATUS and ACCESS/MAX-ACCESS clauses, the STATUS and ACCESS clause values cannot be always appropriately converted when a MIB module is mechanically translated from SMIv1 to SMIv2 format.

You should carefully check all MIB definitions in the converted MIB module and modify them if necessary. For more details about this, see the Help and User Manual documentation or the commented section in the MIB module source file after exporting the MIB module.

□ Do not show this message again

Figure 72: Conversion Completed dialog box

Note 1: The SMIv1 STATUS clause values specify two attributes of object definitions: the validity of the object definition and the implementation conformance requirements. The SMIv2 STATUS clause values specify only the validity of the object definition, while the conformance requirements can be defined with the MODULE-COMPLIANCE statements.

Note 2: In SMIv1, the ACCESS clause specifies the minimal required access, while in SMIv2, the MAX-ACCESS clause specifies the maximum allowed access. This should be considered when converting an ACCESS clause to a MAX-ACCESS clause.

4. You should carefully check all STATUS and MAX-ACCESS clause values of OBJECT-TYPE nodes in the converted module and modify them or add a note into their DESCRIPTION clause if necessary. For detailed rules for converting MIB modules from SMIv1 to SMIv2 format, see the RFC 3584 (section 2.1.1.), or "Understanding SNMP MIBs" book by D. Perkins and E. McGinnis (section 8.2).

   Especially, make sure the following rules are met:

   - If MIB module defines any columnar objects for which instances can be explicitly created by a SNMP Set operation, such objects shall have a MAX-ACCESS clause value of "read-create".
   - For every MIB object that originally had the ACCESS clause of "write-only", and which has been converted by Visual MIB Builder to "read-write" MAX-ACCESS value, a note shall be added into its DESCRIPTION clause, explaining that the reading this object will result in implementation-specific results.
   - For all objects whose value of the STATUS clause was "mandatory" or "optional" in the SMIv1 MIB module, the value in the SMIv2 MIB module must be replaced with "current", "deprecated", or "obsolete" depending on the current usage of such objects. Visual MIB Builder automatically replaces the "mandatory" value with "current", however, you can change this value to "deprecated", or "obsolete" depending on the current usage of such object. For all objects that initially had

STATUS value of "optional", the STATUS value in the converted module must be selected manually.

5. After checking and modifying MIB module definitions (if necessary), you can save the converted MIB module as described in the Saving MIB Modules section, or export it, as explained in the Exporting MIB Modules section of this manual.

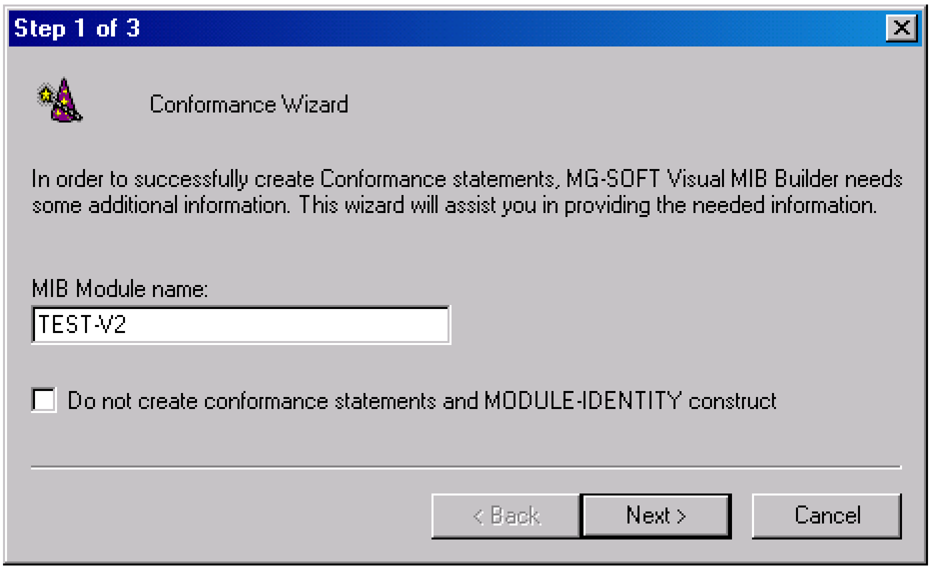
7.2.1 Conformance Wizard

The conformance wizard can be used only when converting MIB modules from SMIv1 into SMIv2 format.

1. To convert a MIB module from SMIv1 into SMIv2 format by using the Conformance wizard, first make sure that the ***Create conformance*** checkbox in the Preferences dialog box | Convert tab | SMIv1 to SMIv2 frame is checked.
2. Open the SMIv1 MIB module to be converted by using the ***File*** / ***Open*** command and then use the ***Tools*** / ***SMI Conversion*** command. The Conformance Wizard dialog box will appear (Figure 73). Follow the wizard's guidelines to provide the information needed for the conformance wizard to create conformance statements.

Note: To disable the conformance wizard, uncheck the ***Create conformance*** checkbox in the Preferences dialog box | Convert tab | SMIv1 to SMIv2 frame.

3. The first screen prompts you for the name of the converted MIB module. By default, the converted MIB module will obtain the name of the original MIB with the `-V2` suffix (e.g., `TEST-V2`). By entering a new into the ***MIB Module name*** input line, you can choose any other name.



In order to successfully create Conformance statements, MG-SOFT Visual MIB Builder needs some additional information. This wizard will assist you in providing the needed information.

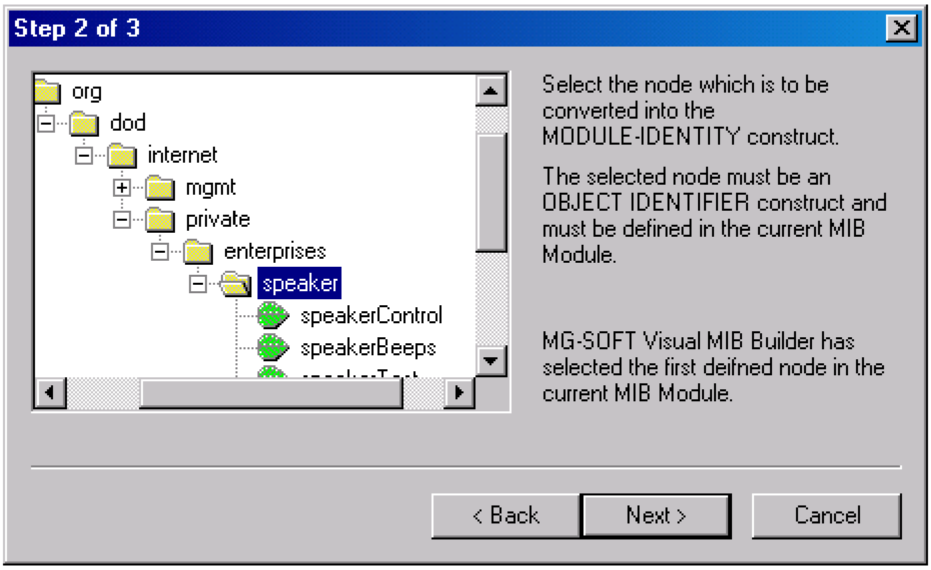
MIB Module name: TEST-V2

□ Do not create conformance statements and MODULE-IDENTITY construct

Figure 73: Conformance Wizard Screen 1

Tip: To move between the different steps of the Conformance Wizard, use the ***Next*** and ***Back*** buttons. If at any stage during the wizard you should decide to abort the configuration and thus lose all the information already entered, simply click the ***Cancel*** button.

4. As the SMI requires, each SMIv2 MIB must have exactly one MODULE-IDENTITY construct defined and it must be the first item defined. To disable this option, check the ***Do not create conformance statements and module-identity construct*** checkbox. If you check this checkbox, the name of the ***Next*** button changes into ***Finish*** and you can skip the next steps of the Conformance wizard. However, this means that no conformance statements will be generated and the module will not conform to SMI rules.
5. If you left the ***Do not create conformance statements and module-identity construct*** checkbox empty, click the ***Next*** button to proceed.



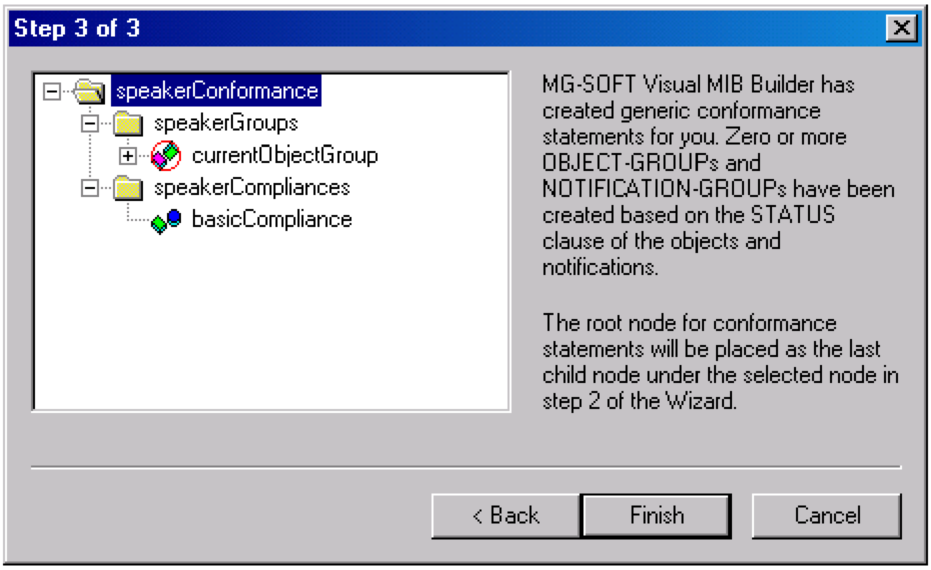
Select the node which is to be converted into the MODULE-IDENTITY construct.

The selected node must be an OBJECT IDENTIFIER construct and must be defined in the current MIB Module.

MG-SOFT Visual MIB Builder has selected the first deifned node in the current MIB Module.

Figure 74: Conformance Wizard Screen 2

6. The second step prompts you to select the node, which is to be converted into the MODULE-IDENTITY construct. By default, Visual MIB Builder selects the first defined node in the current MIB module. By clicking the node in the MIB tree, you can select the desired node. Selected node will be used as a parent node, under which the OBJECT-GROUP, NOTIFICATION-GROUP and MODULE-COMPLIANCE constructs will be created.
7. Please note that the selected node has to be an OBJECT-IDENTIFIER construct defined in the current MIB module, otherwise the ***Next*** button will be disabled and you will not be able to finish the conversion using the Conformance wizard.



MG-SOFT Visual MIB Builder has created generic conformance statements for you. Zero or more OBJECT-GROUPs and NOTIFICATION-GROUPs have been created based on the STATUS clause of the objects and notifications.

The root node for conformance statements will be placed as the last child node under the selected node in step 2 of the Wizard.

Figure 75: Conformance Wizard Screen 3

8. The last step displays how the conformance statements will look after the conversion. By clicking the plus or minus icons in the MIB tree, you can expand the MIB tree to view all its nodes.
9. Click the ***Finish*** button to convert the MIB module.

Note: The MIB module will be converted according to the options set in the Convert tab | SMIv1 to SMIv2 frame in the Preferences dialog box. For more information on how to set conversion options, refer to the Convert Tab section of this manual.

10. After conversion the converted MIB module appears in a new MIB Builder window and the Conversion Completed dialog box (Figure 72) is displayed by default. This message box informs you that due to differences in SMIv1 and SMIv2 STATUS and ACCESS/MAX-ACCESS clauses, the STATUS and ACCESS clause values cannot be always appropriately converted when a MIB module is mechanically translated from SMIv1 to SMIv2 format.

Note 1: The SMIv1 STATUS clause values specify two attributes of object definitions: the validity of the object definition and the implementation conformance requirements. The SMIv2 STATUS clause values specify only the validity of the object definition, while the conformance requirements can be defined with the MODULE-COMPLIANCE statements (these can be automatically created by the Conformance Wizard).

Note 2: In SMIv1, the ACCESS clause specifies the minimal required access, while in SMIv2, the MAX-ACCESS clause specifies the maximum allowed access. This should be considered when converting an ACCESS clause to a MAX-ACCESS clause.

11. You should carefully check all STATUS and MAX-ACCESS clause values of all MIB objects in the converted module and modify them or add a note into their DESCRIPTION clause if necessary. For detailed rules for converting MIB modules from SMIv1 to SMIv2 format, see the RFC 3584 (section 2.1.1.), or "Understanding SNMP MIBs" book by D.T. Perkins and E. McGinnis (section 8.2).

Especially, make sure the following rules are met:

- ❑ If MIB module defines any columnar objects for which instances can be explicitly created by a SNMP Set operation, such objects shall have a MAX-ACCESS clause value of "read-create".
- ❑ For every MIB object that originally had the ACCESS clause of "write-only", and which has been converted by Visual MIB Builder to "read-write" MAX-ACCESS value, a note shall be added into its DESCRIPTION clause, explaining that the reading this object will result in implementation-specific results.
- ❑ For all objects whose value of the STATUS clause was "mandatory" or "optional" in the SMIv1 MIB module, the value in the SMIv2 MIB module must be replaced with "current", "deprecated", or "obsolete" depending on the current usage of such objects. Visual MIB Builder automatically replaces the value "mandatory" with "current", however, you can change this value to "deprecated", or "obsolete" depending on the current usage of such object. For all objects that initially had STATUS value of "optional", the STATUS value in the converted module must be selected manually.

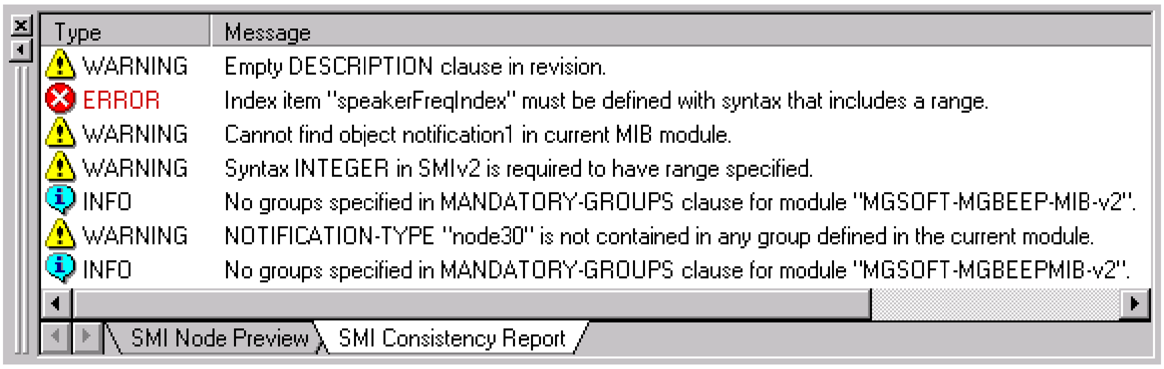
12. After checking and modifying MIB module definitions (if necessary), you can save the converted MIB module as described in the Saving MIB Modules section, or export it, as explained in the Exporting MIB Modules section of this manual.

8 CHECKING BUILT MIB MODULES FOR ERRORS

Visual MIB Builder provides the capability to check if the MIB module you have built conforms to the SMI rules. More precisely, Visual MIB Builder checks if the MIB definition source code that is generated from the visually built MIB module complies with the SMI syntax and semantic rules for writing MIB modules.

By default, Visual MIB Builder automatically checks consistency of the MIB module when you use the ***File*** / ***Export*** or the ***File*** / ***Export Preview*** command, as described in the Exporting MIB Modules section. This section describes how to check the consistency of a MIB module manually.

1. To check the consistency of a MIB module currently opened in the MIB Builder window, use the ***Tools*** / ***Check SMI Consistency*** command. If any inconsistencies are found during this operation, Visual MIB Builder reports them by displaying Error, Warning or Info messages (depending on the severity level of the inconsistency) in the SMI Consistency Report tab of the Node Preview window (***View*** / ***Node Preview*** / ***SMI Consistency Report tab***).
2. Double-click a message in the SMI Consistency Report tab and the node to which the message refers will be selected in the MIB tree, so you can adjust its properties in the Properties window panel or in the Additional Properties window and thus eliminate the reported inconsistency.



Type	Message
⚠ WARNING	Empty DESCRIPTION clause in revision.
⊗ ERROR	Index item "speakerFreqIndex" must be defined with syntax that includes a range.
⚠ WARNING	Cannot find object notification1 in current MIB module.
⚠ WARNING	Syntax INTEGER in SMIv2 is required to have range specified.
ℹ INFO	No groups specified in MANDATORY-GROUPS clause for module "MGSOFT-MGBEEP-MIB-v2".
⚠ WARNING	NOTIFICATION-TYPE "node30" is not contained in any group defined in the current module.
ℹ INFO	No groups specified in MANDATORY-GROUPS clause for module "MGSOFT-MGBEEPMIB-v2".

Figure 76: SMI Consistency Report tab

Visual MIB Builder uses three types of messages to report MIB definition inconsistencies. Each type of message indicates different severity level of inconsistency:

- The *ERROR message* (⊗) indicates that Visual MIB Builder has found a fatal syntactic or semantic error in the checked MIB definitions, which needs to be corrected before the resulting MIB definition source file will conform to the SMI specification rules and thus compile in any SMI compliant MIB Compiler.
- The *WARNING message* (⚠) warns you of a non-fatal syntactic or semantic error in the checked MIB definitions. To generate a MIB module that conforms to the SMI rules, you should correct the inconsistencies reported by the Warning messages.

Tip: Visual MIB Builder can be configured to suppress Warning and/or Info messages. To do that, check the ***Suppress Warning messages*** checkbox and/or ***Suppress Info Messages*** checkbox in the Preferences dialog box | Builder tab (***View*** / ***Preferences***).

- The *INFO message* (ℹ) informs you about a minor inconsistency in the checked MIB definitions. Although inconsistencies reported by the Info messages will most probably not cause compilation problems, it is recommended to eliminate these inconsistencies before generating a MIB definition source file.

Every message displays a short but meaningful description of the inconsistency.

Note 1: Regardless of number and type of reported messages, Visual MIB Builder will still let you use the ***File*** / ***Export*** command to generate the MIB module definition file, but definitions in such file may not conform to the SMI rules.

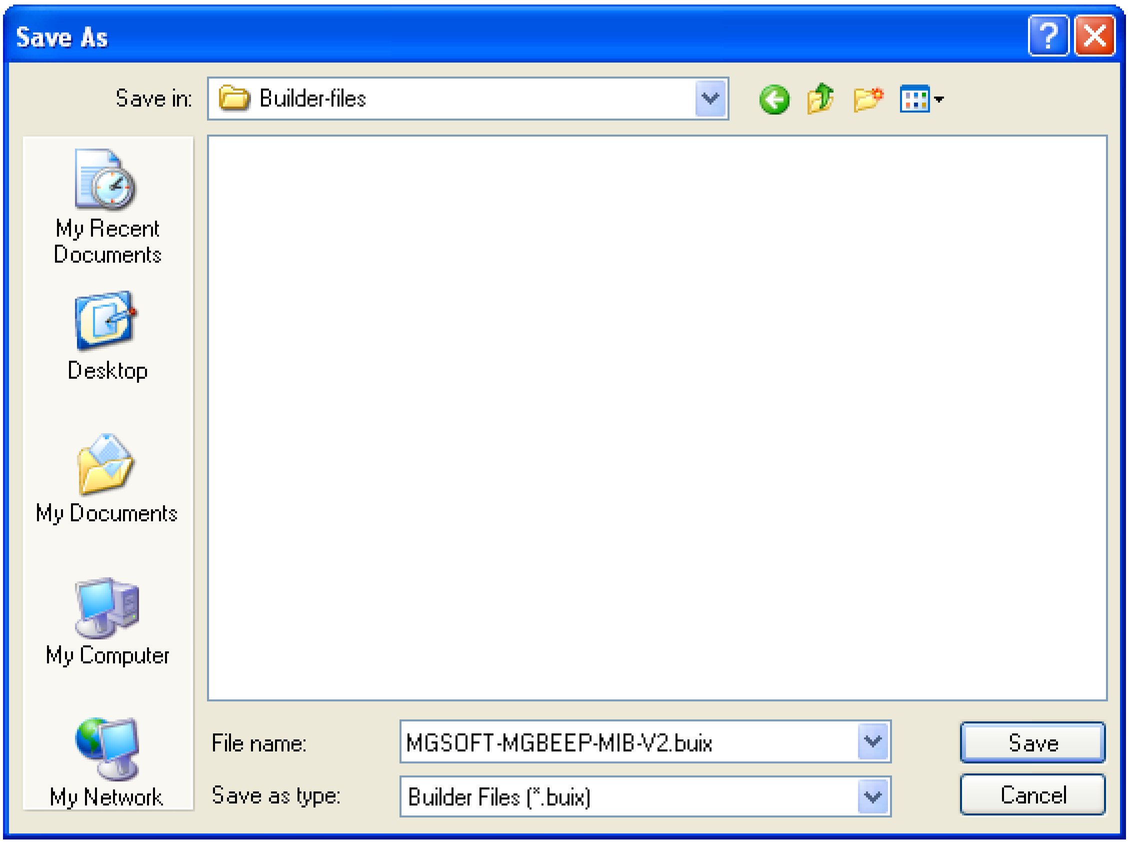
Note 2: For more detailed description of Error, Warning and Info messages please refer to the "Visual MIB Builder Messages" help topic (***Help*** / ***Help Topics***).

9 SAVING MIB MODULES

Save toolbar button

Each time you make a change to a MIB module, the MIB module has to be saved to store the changes into the MIB Builder file. Without saving, any modifications made to a MIB module will be lost when the MIB Builder window containing that MIB module is closed or when the application is closed.

1. To save a new MIB module to a file or to save an existing module under a new file name, use the ***File*** / ***Save As*** command. The standard Save As dialog box appears where you need to specify the file name and location whereto you want to store the file.



Save in: Builder-files

File name: MGSOFT-MGBEEP-MIB-V2.buix

Save as type: Builder Files (*.buix)

Figure 77: Save As dialog box

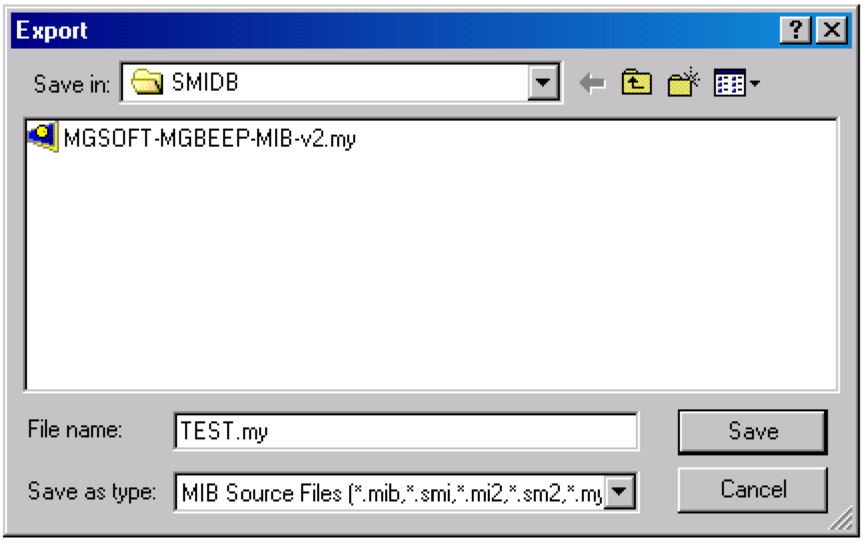
2. Click the ***Save*** button to save the built MIB module into Visual MIB Builder XML file format (.BUIX) and close the Save As dialog box.

3. Once the MIB module has been saved to a BUIX file, you can use the ***File*** / ***Save*** command or the ***Save*** toolbar button to save any further modifications to the same file.

10 EXPORTING MIB MODULES

In order to use a MIB module in any other application than Visual MIB Builder, it is necessary to export the MIB module into the MIB module definition file format. MIB module definition files are written in the MIB module definition language and saved in ASCII files. Such MIB files can be compiled by any SMI compliant MIB Compiler into a file format used by other applications. For example, MG-SOFT MIB Compiler compiles MIB module definition files into the MG-SOFT's proprietary SMIDB format. An application can access such compiled MIB files by using the industry standard WinMIB interface. When a compiled MIB module is saved to a SMIDB database file, a WinMIB-based application (like MG-SOFT MIB Browser Professional) can access and utilize the database file.

1. To export the MIB module currently opened in the MIB Builder window, choose the ***File*** / ***Export*** command. The Export dialog box appears. In the Export dialog box specify the save destination location and the file name for the MIB definition source file that will be generated. Click the ***Save*** button to start the exporting operation.



Save in: SMIDB

MGSOFT-MGBEEP-MIB-v2.my

File name: TEST.my

Save as type: MIB Source Files (*.mib,*.smi,*.mi2,*.sm2,*.my

Figure 78: Export dialog box

Export toolbar button

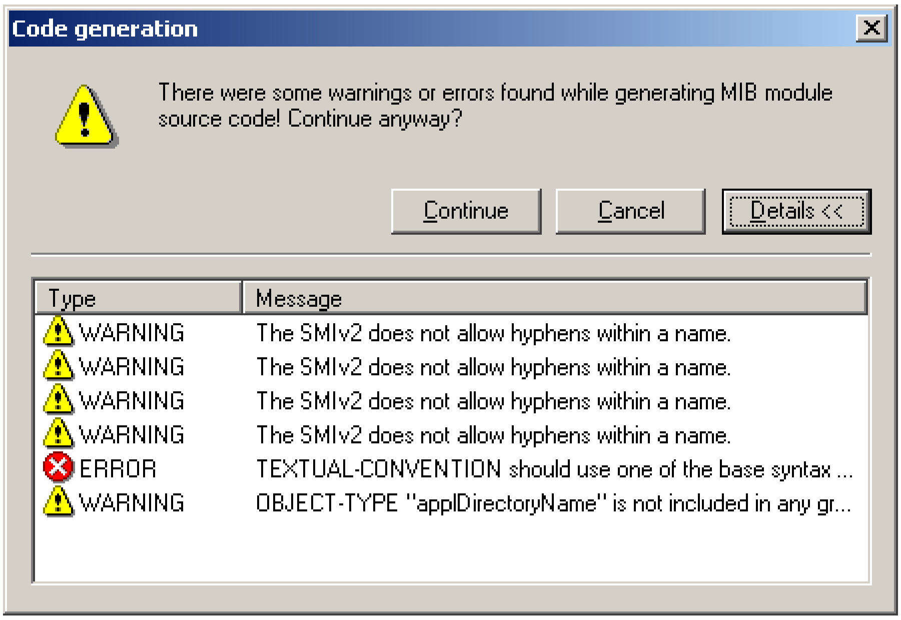
Export Preview toolbar button

Print toolbar button

Select All toolbar button

Copy toolbar button

2. By default, the MIB module is checked for inconsistencies before it is exported to a file (as described in the Checking Built MIB Modules for Errors section). If no inconsistencies are detected, the MIB module definition file is created, as described in step 5. In case any inconsistencies are found during the check operation, Visual MIB Builder displays a warning message box that informs you of detected inconsistencies and lets you view details of found inconsistencies, continue exporting operation, or cancel it.



Code generation

There were some warnings or errors found while generating MIB module source code! Continue anyway?

Type	Message
WARNING	The SMIv2 does not allow hyphens within a name.
WARNING	The SMIv2 does not allow hyphens within a name.
WARNING	The SMIv2 does not allow hyphens within a name.
WARNING	The SMIv2 does not allow hyphens within a name.
ERROR	TEXTUAL-CONVENTION should use one of the base syntax ...
WARNING	OBJECT-TYPE "applDirectoryName" is not included in any gr...

Figure 79: A warning message box informing you of inconsistencies in the MIB module

3. If you select the ***Continue*** button in the Code generation message box (Figure 79), the MIB definition source file is created and saved, despite the inconsistencies found. Note that definitions in such file will not conform to the SMI rules. If you select the ***Cancel*** button in the above message box (Figure 79), the export operation is aborted without exporting MIB definitions to a file. Instead, Visual MIB Builder displays Error, Warning or Info messages in the SMI Consistency Report tab of the Node Preview window (***View*** / ***Node Preview*** / ***SMI Consistency Report tab***). Use the ***Details*** button to view the messages before clicking the ***Continue*** or ***Cancel*** button.
4. You should examine these messages and remove all inconsistencies before performing the Export operation again. For more information on this topic, read the Checking Built MIB Modules for Errors section.

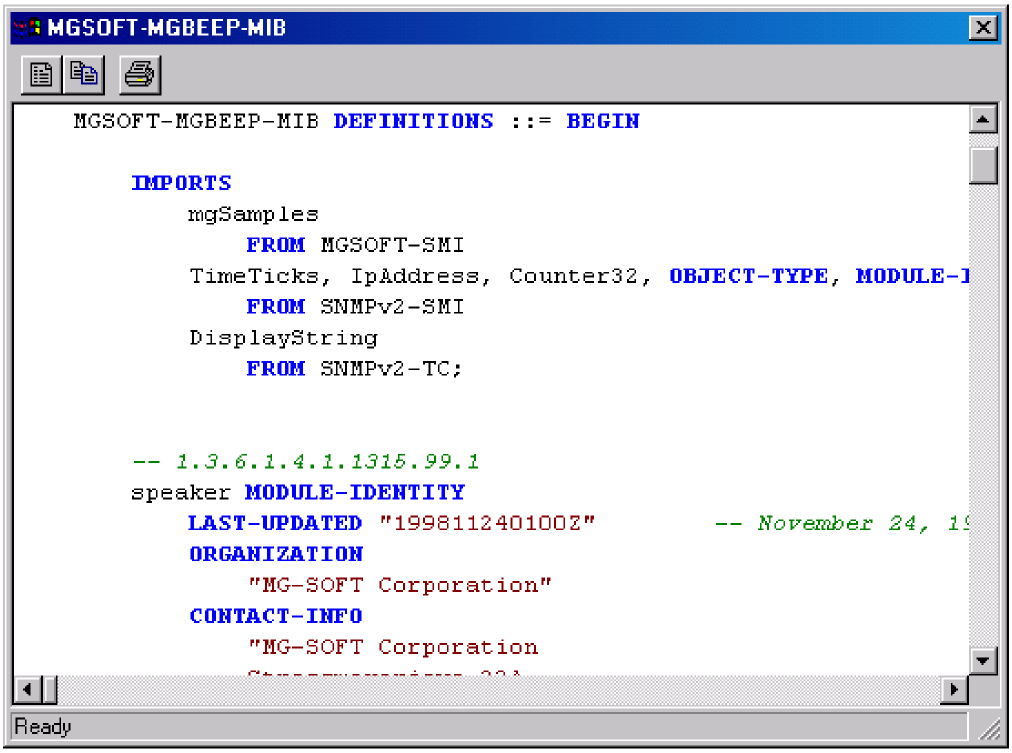
Tip: The option to check for errors before exporting MIBs can be disabled (although this is not recommended). To do this, uncheck the ***Check MIB tree before Export/Export Preview*** checkbox in the MIB Builder Preferences | Builder tab.

5. After the MIB module definition file is created and saved, it is automatically opened in MG-SOFT MIB Compiler, where you can view it or compile it by pressing the ***F7*** key. To disable the option to automatically open exported file in MIB Compiler, uncheck the ***Run MIB Compiler after export*** checkbox in the MIB Builder Preferences | Builder tab (***View*** / ***Preferences***). For more information on MIB Compiler, refer to the MIB Compiler help file and user manual.

Export Preview

Visual MIB Builder lets you preview all MIB definitions of the MIB module. You can use this option to view the MIB module definition source code before actually exporting it into a file.

1. To view definitions of the MIB module that is opened in the MIB Builder window, use the ***File*** / ***Export Preview*** command.
2. By default, the MIB module is checked for errors before its source code is displayed in the Export Preview dialog box. If no inconsistencies are found, the Export Preview dialog box appears displaying the generated MIB definition code with applied syntax-coloring (Figure 80). In case any inconsistencies are found during the check operation, Visual MIB Builder displays a warning message box that informs you of detected inconsistencies and prompts you to either examine or ignore them (Figure 79).



```
  MGSOFT-MGBEEP-MIB DEFINITIONS ::= BEGIN

      IMPORTS
          mgSamples
              FROM MGSOFT-SMI
          TimeTicks, IpAddress, Counter32, OBJECT-TYPE, MODULE-I
              FROM SNMPv2-SMI
          DisplayString
              FROM SNMPv2-TC;


      -- 1.3.6.1.4.1.1315.99.1
      speaker MODULE-IDENTITY
          LAST-UPDATED "199811240100Z"        -- November 24, 19
          ORGANIZATION
              "MG-SOFT Corporation"
          CONTACT-INFO
              "MG-SOFT Corporation
```

Figure 80: Export Preview

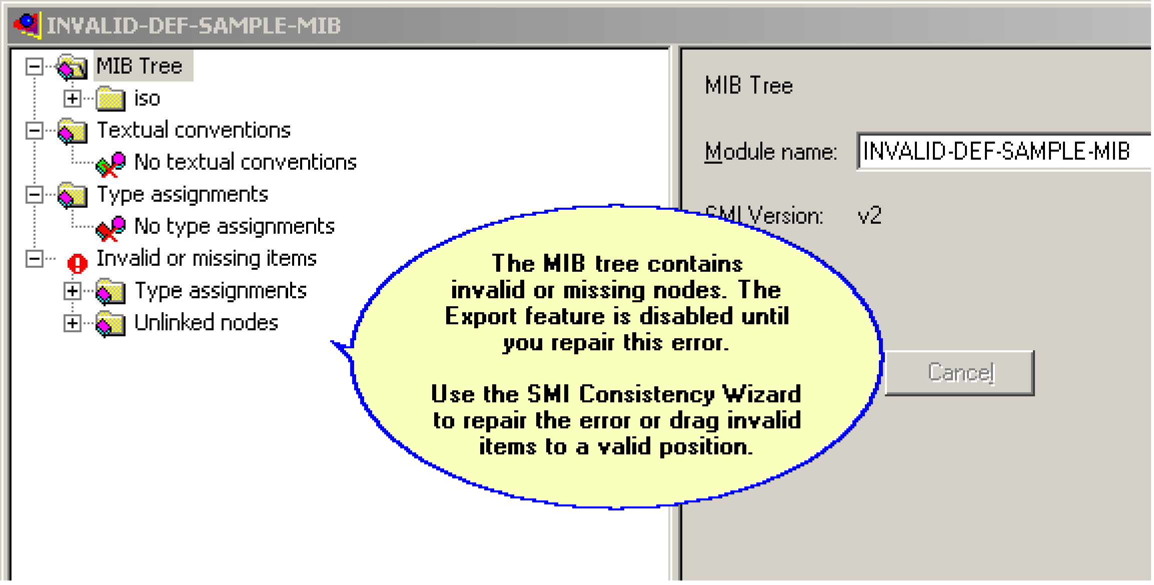
3. If you decide to ignore the inconsistencies, the Export Preview dialog box appears displaying the generated MIB definition code (Figure 80). Note that in such case displayed definitions will not fully conform to SMI rules. If you choose the ***Cancel*** option, the Export Preview operation is terminated without displaying MIB definitions. Instead, Visual MIB Builder displays Error, Warning or Info messages in the SMI Consistency Report tab of the Node Preview window (***View*** / ***Node Preview*** / ***SMI Consistency Report tab***). You should examine these messages and remove all inconsistencies before exporting the MIB module. For more information on this topic, read the Checking Built MIB Modules for Errors section.

Tip: The option to check for errors before exporting MIBs can be disabled. To do this, uncheck the ***Check MIB tree before Export/Export Preview*** checkbox in the MIB Builder Preferences | Builder tab.

4. The content of the Export Preview dialog box can be printed or copied to the clipboard by using the ***Print*** or ***Copy*** toolbar buttons. After clicking the ***Print*** button, standard Print dialog box appears where you can specify the printing parameters.

11 USING SMI CONSISTENCY WIZARD

Visual MIB Builder lets you open and repair MIB definition (source) files that contain some commonly found errors related to missing or invalid MIB object definitions. In such case, it informs you about invalid or missing nodes, when you open the file (Figure 81).



Module name: INVALID-DEF-SAMPLE-MIB

SMI Version: v2

The MIB tree contains invalid or missing nodes. The Export feature is disabled until you repair this error.

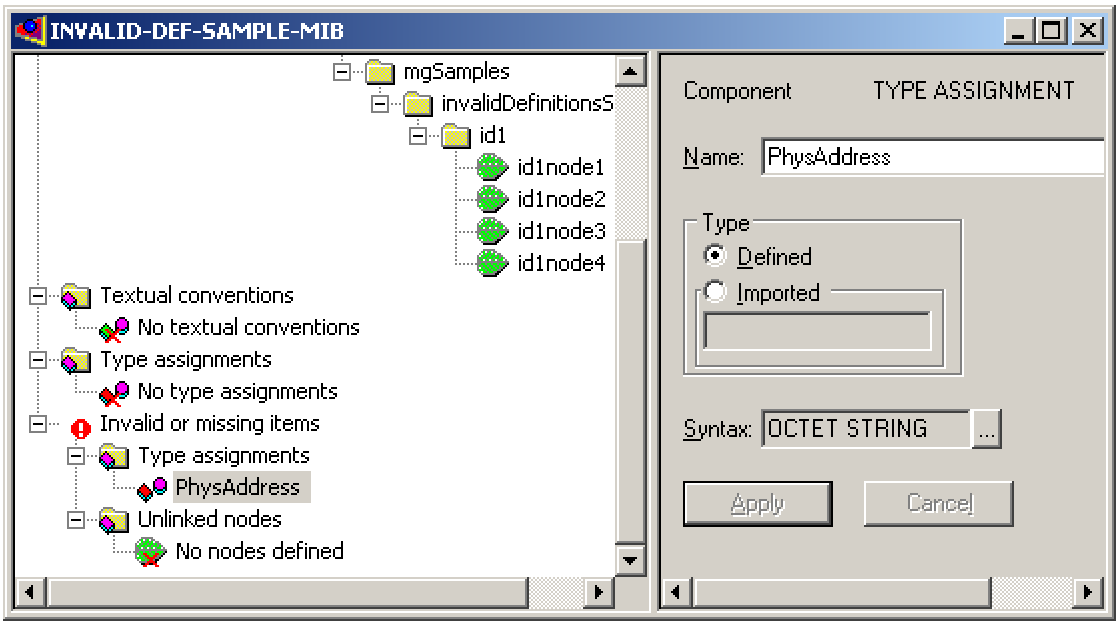
Use the SMI Consistency Wizard to repair the error or drag invalid items to a valid position.

Figure 81: Visual MIB Builder warns you about invalid or missing items in the MIB tree

Visual MIB Builder lets you repair two types of errors:

1. In case there are any nodes that cannot be linked to the MIB tree because of broken MIB definitions, they are displayed under the `Invalid or missing items / Unlinked nodes` subtree (Figure 81). In such case, you should drag the nodes to the appropriate position in the MIB tree to correct the problem.
2. In case the MIB definition source file references any Type Assignment or TEXTUAL-CONVENTION construct that is not imported or properly defined or there is a typing error in the construct that references it, the node representing such Type Assignment or TEXTUAL-CONVENTION construct is displayed under `Invalid or missing items / Type Assignments` in a MIB tree (Figure 81). You should run the SMI Consistency Wizard that will help you correct the broken or missing definition of the Type Assignment or TEXTUAL-CONVENTION objects.

Note: Until you correct the definition of broken MIB objects, the Export feature and the Revisions framework are not accessible.



Component TYPE ASSIGNMENT

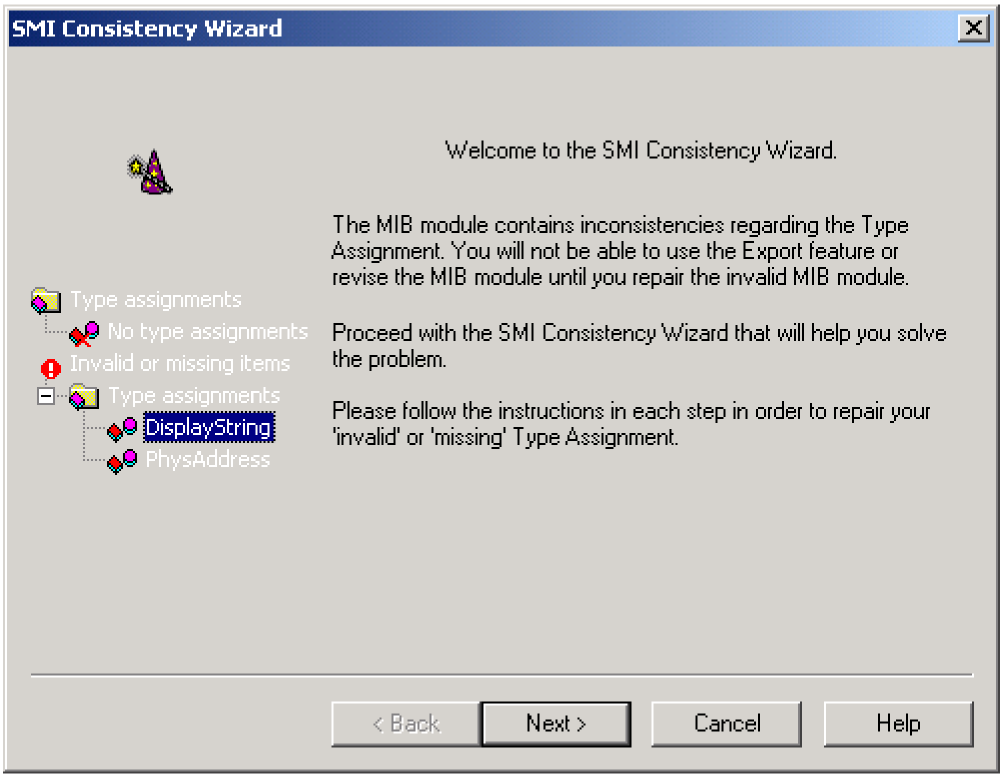
Name: PhysAddress

Type
● Defined
○ Imported

Syntax: OCTET STRING

Figure 82: Invalid or missing Type assignment or TEXTUAL-CONVENTION object

To repair the broken Type assignment definition, use the ***Tools / SMI Consistency Wizard*** command.



Welcome to the SMI Consistency Wizard.

The MIB module contains inconsistencies regarding the Type Assignment. You will not be able to use the Export feature or revise the MIB module until you repair the invalid MIB module.

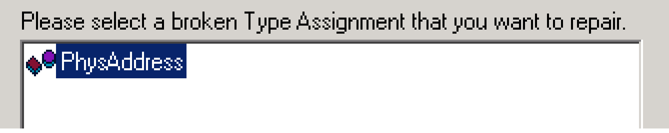
Proceed with the SMI Consistency Wizard that will help you solve the problem.

Please follow the instructions in each step in order to repair your 'invalid' or 'missing' Type Assignment.

Figure 83: SMI Consistency Wizard Welcome screen

1. The SMI Consistency Wizard Welcome screen is displayed (Figure 83).

2. Click the ***Next*** button and in the next screen select the Type Assignment that is to be repaired (Figure 84).

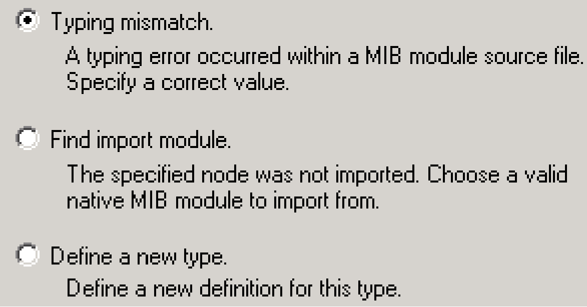


Please select a broken Type Assignment that you want to repair.

PhysAddress

Figure 84: Select the inconsistent Type Assignment to repair

3. Again, click the ***Next*** button to proceed to the following step (Figure 85).



● Typing mismatch.
A typing error occurred within a MIB module source file. Specify a correct value.

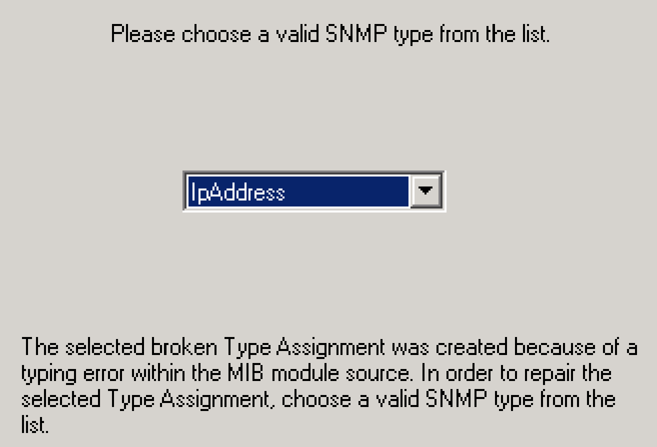
○ Find import module.
The specified node was not imported. Choose a valid native MIB module to import from.

○ Define a new type.
Define a new definition for this type.

Figure 85: Select repair action

4. Now choose an action that should correct the broken MIB definition. Select from:
   - Typing mismatch
   - Find Import module
   - Define a new type

5. Click the ***Next*** button to proceed to the following step.
   - If you have selected the `Typing mismatch` option, you will have to select the appropriate data type from the drop-down list containing valid data types (Figure 86). After specifying the details, click the ***Finish*** button.



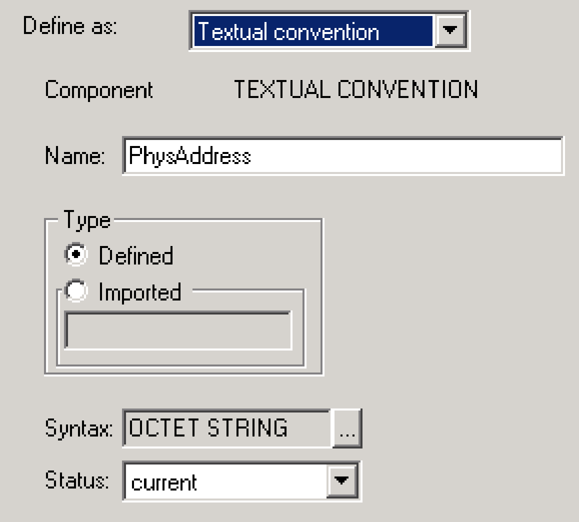
Please choose a valid SNMP type from the list.

IpAddress

The selected broken Type Assignment was created because of a typing error within the MIB module source. In order to repair the selected Type Assignment, choose a valid SNMP type from the list.

Figure 86: Repair typing mismatch

- If you have selected the `Define a new type` option, you will have to define a new data type by specifying properties for a Type Assignment or TEXTUAL CONVENTION construct. After specifying the details (Figure 87), click the ***Finish*** button.



Define as: Textual convention

Component TEXTUAL CONVENTION

Name: PhysAddress

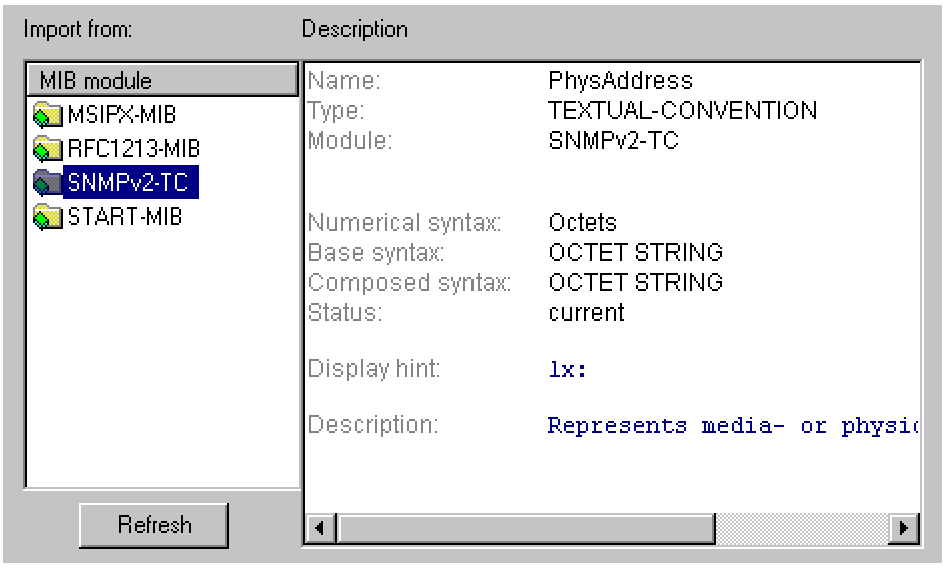
Type
● Defined
○ Imported

Syntax: OCTET STRING

Status: current

Figure 87: Define a new type

- If you have selected the `Find import module` option, the SMI Consistency Wizard will search all registered MIB modules for a definition of this MIB object and display the modules that this object can be imported from (Figure 88). Select the appropriate MIB module and click the ***Finish*** button.



Import from:

MIB module
MSIPX-MIB
RFC1213-MIB
SNMPv2-TC
START-MIB

Description

Name: PhysAddress
Type: TEXTUAL-CONVENTION
Module: SNMPv2-TC

Numerical syntax: Octets
Base syntax: OCTET STRING
Composed syntax: OCTET STRING
Status: current

Display hint: 1x:

Description: Represents media- or physic

Figure 88: Select a MIB module to import from

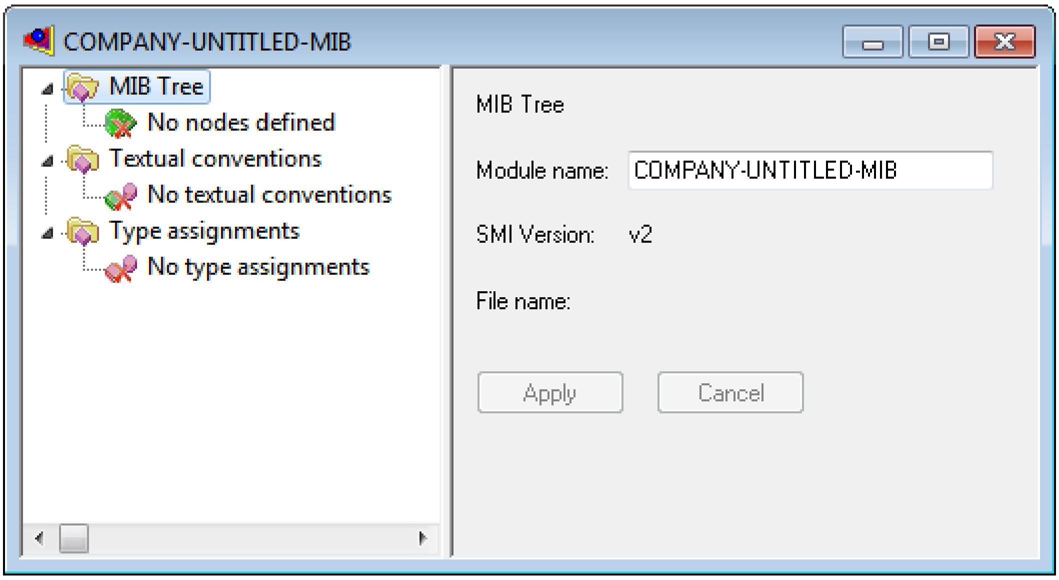
6. By performing the selected operation (e.g., define a new type), the SMI Consistency wizard repairs the broken Type Assignment construct and places its node on the right position in the MIB tree. Now you can export the repaired MIB module into MIB definition source file format as described in the Exporting MIB Modules section.

12 USAGE EXAMPLE OF CREATING AN SMIV2 MIB MODULE

This section contains a complete example of creating an imaginary MIB module with Visual MIB Builder. For practical reasons we have selected a device that is probably familiar to many readers: a cold drink vending machine. This example will guide you step-by-step through the process of building the SMIv2 MIB module that is supposed to be used to remotely manage that device. By using a similar approach, you can create any other MIB module definition file.

Importing Nodes

1. Start the Visual MIB Builder program.
2. Use the ***File / New / SMI2 MIB Module*** command, as we will build an SMIv2 MIB module. A new MIB Builder window appears, containing an empty (new) MIB Module called COMPANY-UNTITLED-MIB.



Module name: COMPANY-UNTITLED-MIB

SMI Version: v2

File name:

Figure 89: new MIB Builder window

3. Into the ***Module Name*** input line in the Properties window panel enter `MGSOFT-VENDING-MACHINE-MIB` and click the ***Apply*** button.
4. As we are going to build a sample MIB module for MG-SOFT Corporation, we will place this module under the `mgSamples` node, with the module root OID that has been assigned to us by the MG-SOFT MIB registration authority. Since the `MGSOFT-SMI` MIB file defines the `mgSamples` MIB object, we will import this node (including all its parent nodes) into the `MGSOFT-VENDING-MACHINE-MIB`.
5. Select the ***File / Import*** command. The Import dialog box appears. Navigate to the folder where the SMIv2 version of `MGSOFT-SMI` MIB definition source file is located. From the ***Files of type*** drop-down list select the `MIB Source Files.` All MIB source files within the given folder will be displayed in the Import dialog box. Choose

the `MGSOFT-SMI.my` file and click the ***Open*** button. This will open a new MIB Database window with all nodes defined in the `MGSOFT-SMI` MIB module.

6. Expand the MGSOFT-SMI MIB tree by right-clicking the root node and selecting the ***Expand*** pop-up command. Select the `mgSample node`, and drag it to the MGSOFT-VENDING-MACHINE-MIB window. Drop it onto the `enterprises` node. A message box will appear, informing you that you also have to import the parent nodes. Select ***Yes***. All nodes between the root node and the `mgSamples` node should appear in the MGSOFT-VENDING-MACHINE-MIB window. Your MIB tree should now look like this:

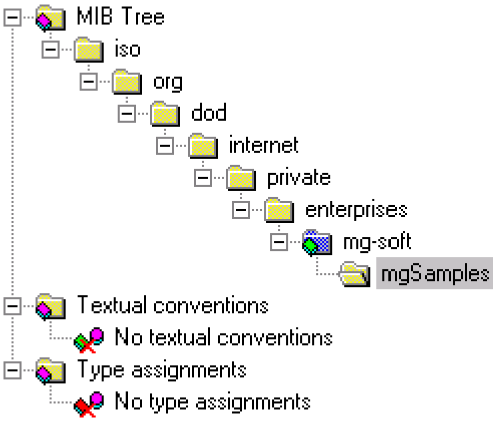


Figure 90: MIB tree

Adding MODULE-IDENTITY Construct

SMIv2 requires each SNMP MIB module to have exactly one MODULE-IDENTITY construct defined, which must precede all other definitions in the module. Although we have already imported one MODULE-IDENTITY construct from the MGSOFT-SMI MIB, we still have to define one in the current MIB module.

1. From the Components window (***View / Components***) select the MODULE-IDENTITY component and drag-and-drop it onto the `mgSamples` node. The new node will be placed below the `mgSamples` node as its child node.

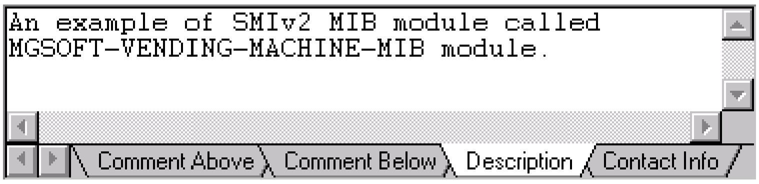
> **Note:** For more information on defining *MODULE-IDENTITY construct*, read the MODULE-IDENTITY Constructs section of this manual.

2. Into the ***Name*** input line in the Properties window panel enter `mgVendingMachine`.

3. The ***OID*** value will be automatically assigned, but we will change the last sub-identifier number of the OID to correspond to the OID value that has been assigned to us by MG-SOFT. The ***Last updated*** value will also be automatically updated.

4. Into the ***Organization*** input line, enter the name of our organization, which is `MG-SOFT`.

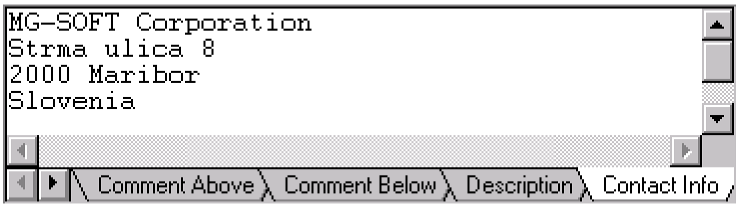
5. To add a description of the module or any information that is needed to understand the object or module, open the Additional Properties window (***View / Additional Properties***) and enter the text into the ***Description*** tab.



An example of SMIv2 MIB module called
MGSOFT-VENDING-MACHINE-MIB module.

Figure 91: Description tab

6. Into the ***Contact Info*** tab enter:



MG-SOFT Corporation
Strma ulica 8
2000 Maribor
Slovenia

Figure 92: Contact Info tab

7. After changing a value, press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

Defining Management Information

Now we will create the area of the MIB module in which the management information will be defined. As you probably remember from the first sections of this manual, only OBJECT-TYPE constructs are used for defining management information (*scalar* and *columnar objects*). OBJECT-TYPE constructs are also used for organizing related information into tables (*table* and *row objects*).

1. First, we will add an OBJECT IDENTIFIER node to identify a new branch in the MIB tree in which OBJECT-TYPE constructs will be defined. To do that, select the OBJECT-IDENTIFIER component from the Components window (***View / Components***) and drag-and-drop it onto the `mgVendingMachine` node. The new node will be placed below the `mgVendingMachine` node as its first child node. The ***OID*** values of this and all subordinated nodes will be automatically assigned according to their position in the MIB tree.
2. Into the ***Name*** input line in the Properties window panel enter `automatState`.
3. After changing a value, click the ***Apply*** button or the changes will not take effect.

Now let's define the management information. Our MIB module will be defined to contain the information about the number of containers in the machine (e.g., one container is for soda drink, the other for juice, the third for cola…) as well as number of cans in individual containers (e.g., the maximum number of all cans, the current number of cans…).

1. From the Components window (***View*** / ***Components***) select the OBJECT-TYPE (Scalar) component and drag-and-drop it onto the `automatState` node. The new node will be placed below the `automatState` node as its first child node. The ***OID*** value will be automatically assigned according to the position of inserted node in the MIB tree.
2. Into the ***Name*** input line in the Properties window panel enter `numberOfContainers`.
3. From the ***Max Access*** drop-down list, pick the `read-write` value.
4. Click the ***Browse*** ('...') button next to the ***Syntax*** input line. The Syntax Details dialog box will appear. From the ***Syntax*** drop-down list select `Integer32`. In the Type frame, check the ***Simple*** radio button.
5. Make sure the value of the ***Status*** clause is set to `current`, which indicates that the definition is valid.

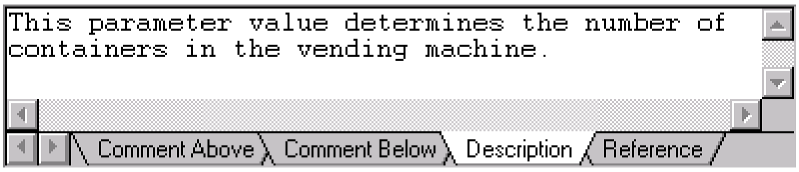

This parameter value determines the number of containers in the vending machine.

Figure 93: Description of the scalar node

6. To add a description of the object or any information that is needed to understand the object or module, open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the ***Description*** tab.
7. Leave all other parameters at their default value and click the ***Apply*** button or the changes will not take effect.

Now we will define objects that contain the information about the state of individual containers. There can be an optional number of containers in the machine and each container contains different number of cans, meaning that we will operate with multiple pieces of management data. In this case, it is very convenient to create an imaginary, tabular structure of an ordered collection of *scalar objects*. Each *table object* contains *rows*, and each *row* (*row object*) consists of one or more *scalar objects*, termed *columnar objects*.

Since a *table* contains a *row object* and *columnar objects*, creating a *table* takes three steps: defining a *table object*, defining a *row objects* and defining one or more *columnar objects*.

Note: Description clause, which is mandatory for all constructs in SMIV2 MIB modules, is used to describe the object or any information that is needed to understand the object or module. Open the Additional Properties window (***View*** / ***Additional Properties***) and enter the text into the Description tab, or just leave the default text (`Description`).

Defining Table and Row Objects

1. From the Components window (***View*** / ***Components***) select the OBJECT-TYPE (Table) component and drag-and-drop it onto the `automatState` node. By default, when adding a table node, one row node and one columnar node are automatically added to the table. The ***OID*** value will be automatically assigned according to the position of inserted nodes in the MIB tree.

2. Select the *table node* and enter `containerTable` into the ***Name*** input line in the Properties window panel.
3. The ***Syntax*** value is set automatically as `SEQUENCE OF….` This is a special value of the SYNTAX, which indicates that a *table object* is being defined. After you specify the *row object*, the syntax will change and conform to the *row object* definition.
4. Press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.
5. Now select the *row* node. The ***Name*** of the node (`containerEntry`) as well as the ***OID*** value will be chosen automatically.

Defining Columnar Objects

1. As already said, one *columnar* node has already been added to the *table*. To add more *columnar* nodes, select the OBJECT-TYPE (Columnar) component and drag-and-drop it onto the `containerEntry` node. The new node will be placed below the `containerEntry` node. In the same way, create four more *columnar* nodes. Their ***OID*** values will be assigned automatically according to their position in the MIB tree.
2. Select the first *columna*r node and enter `ItemIndex` into the ***Name*** input line. Each table needs to have an index column and every row of the table is identified by the index column value.
3. Click the ***Browse*** ('...') button next to the ***Syntax*** input line. The Syntax Details dialog box will appear. From the ***Syntax*** drop-down list select `Integer32`. In the Type frame, check the ***Value or Range*** radio button. Into the ***From*** input line enter `1` and into the ***To*** input line enter `8`. Click the ***Add*** button and then the ***OK*** button to close the dialog box.
4. Make sure the ***Max Access*** value is set to `read-only`. Set the ***Status*** value to `current`, this indicates that the definition is valid.
5. Leave all other parameters at their default values click the Apply button or the changes will not take effect.
6. Select the `containerEntry` node and click the ***Browse*** ('...') button next to the ***Index*** input line. The Index selection dialog box appears. From the ***Available indexes*** list select the `itemIndex` object and click the right arrow button. Selected object will appear in the ***Selected indexes*** list. Click the ***OK*** button to save the settings and close the window. The selected value will appear also in the ***Index*** input line.
7. Now select the second *columnar* node in the MIB tree and enter `itemName` into the ***Name*** input line.
8. Click the ***Browse*** ('...') button next to the ***Syntax*** input line. The Syntax Details dialog box will appear. From the ***Syntax*** drop-down list select `OCTET STRING`. In the Type frame, check the ***Simple*** radio button.
9. Set the ***Max Access*** value to `read-write` and the ***Status*** value to `current`. Leave all other parameters at their default values and click the ***Apply*** button or the changes will not take effect.
10. Select the third *columnar* node in the MIB tree and enter `maxItems` into the ***Name*** input line.

11. Set the ***Syntax*** to `simple Integer32.` Make sure the ***Max Access*** value is set to `read-write` and the ***Status*** value to `current`. Leave all other parameters at their default values and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.
12. Select the fourth *columnar* node in the MIB tree and enter `currentItems` into the ***Name*** input line. Follow the instructions in step 8 to set the ***Syntax*** of this object to `simple Integer32`.
13. Make sure the ***Max Access*** value is set to `read-only` and the ***Status*** value to `current`. Leave all other parameters at their default values and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.
14. Now select the last *columnar* node in the MIB tree and enter `thresholdItemValue` into the ***Name*** input line. Follow the instructions in step 8 to set the ***Syntax*** of this object to `simple Integer32`.
15. Make sure the ***Max Access*** value is set to `read-write` and the ***Status*** value to `current`. Leave all other parameters at their default values and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.
16. Your MIB tree should now look like this:

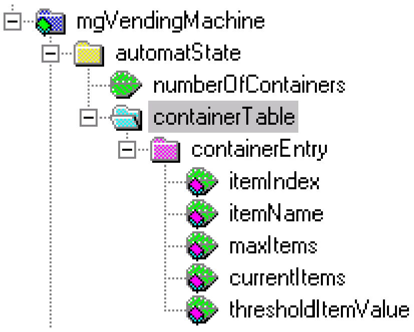


Figure 94: mgVendingMachine MIB tree

Defining NOTIFICATION-TYPE Constructs

If we wish to be notified about changes in the state of our vending machine, we have to specify when the agent implemented on the device should trigger an event to notify us about the change. We will define the first event message to be sent to the SNMP manager each time the number of cans in a container reaches the number specified, and the second event message should be triggered when the container is empty.

1. First, we will add another OBJECT IDENTIFIER node to identify a new branch in the MIB tree in which NOTIFICATION-TYPE constructs will be defined. To do that, select the OBJECT IDENTIFIER component from the Components window (***View / Components***) and drag-and-drop it onto the `mgVendingMachine` node. The new node will be placed in the MIB tree as a child of the `mgVendingMachine` node. The ***OID*** value will be automatically assigned according to the position of inserted node.
2. Enter `vendingMachineNotifications` into the ***Name*** input line and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

3. Now, select the NOTIFICATION-TYPE component from the Components window (***View / Components***) and drag-and-drop it onto the `vendingMachineNotifications` node. The new node will be placed in the MIB tree as the first child node of the `vendingMachineNotifications` node. The ***OID*** value will be automatically assigned according to the position of inserted node. Use the same procedure to add another NOTIFICATION-TYPE node.
4. Select the first NOTIFICATION-TYPE node and enter `outOfDrinkNotification` into the ***Name*** input line.
5. Now we will specify the object whose value will be sent in the event report message. Because this notification will be sent to the manager when a container is empty, it is quite logical for the event report message to identify the name of the container that is out of drink. Therefore, we will specify the `itemName` object.
6. Click the ***Browse*** button next to the ***Objects*** input line to open the Variable Selection dialog box. From the ***Available variables*** list select the `itemName` object and click the right arrow button. Selected object will appear in the ***Selected variables*** list. Click the ***OK*** button to save the settings and close the window. The selected value will appear also in the ***Objects*** input line.
7. From the ***Status*** drop-down list select `current` and press the ***Enter*** key or click the ***Apply*** button.
8. Next, select the second NOTIFICATION-TYPE node and enter `thresholdDrinkNotification` into the ***Name*** input line.
9. This notification will be sent to the manager when the number of cans in a container reaches the number specified. Therefore, it is quite logical for the event report message to identify the name of the container and the number of cans currently in the container. Values of those two MIB object will be sent as variable bindings of the event report message.
10. Click the click the ***Browse*** button next to the ***Objects*** input line to open the Variable Selection dialog box. From the ***Available variables*** list select the `itemName` object and click the right arrow button. Repeat this step to add the `currentItems` object. Selected objects will appear in the ***Selected variables*** list. Click the ***OK*** button to save the settings and close the window. The selected values will appear also in the ***Objects*** input line.
11. From the ***Status*** drop-down list select `current` and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

Each NOTIFICATION-TYPE node must be a member of at least one notification group. To define a collection of related NOTIFICATION-TYPEs, the NOTIFICATION-GROUP construct is used.

1. First, we will add another OBJECT IDENTIFIER node to identify a new branch in the MIB tree in which groups of MIB objects and notifications will be defined. To do that, select the OBJECT IDENTIFIER component from the Components window (***View / Components***) and drag-and-drop it onto the `mgVendingMachine` node. The new node will be placed in the MIB tree as a child of the `mgVendingMachine` node. The ***OID*** value will be automatically assigned according to the position of inserted node.
2. Into the ***Name*** input line, enter `vendingMachineGroups` and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

3. From the Components window (***View*** / ***Components***) select the NOTIFICATION-GROUP component and drag-and-drop it onto the `mgVendingMachine` node. The new node will be placed below the `mgVendingMachine` node as its first child node. The ***OID*** value will be automatically assigned according to the position of inserted node in the MIB tree.
4. Click the ***Browse*** button next to the ***Notifications*** input line to open the Notifications Selection dialog box. From the ***Available notifications*** list, select both notifications and click the right arrow button. The selected notifications will appear in the ***Selected notifications*** list.
5. From the ***Status*** drop-down list, select the `current` value and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

Defining OBJECT-GROUP Construct

SMI requires every object type that has the ***Max. Access*** clause value set to any other value than `not-accessible` to be a member of at least one object group. All objects that we have defined so far in our MIB have either `read-write` or `read-only` access; therefore, they all have to be added to a group.

1. From the Components window (***View*** / ***Components***) select the OBJECT-GROUP component and drag-and-drop it onto the `vendingMachineGroups` node. The new node will be placed below the `notificationGroup` node as the second child node of the `vendingMachineGroups` node. The ***OID*** value will be automatically assigned according to the position of inserted node.
2. Into the ***Name*** input line, enter `objectGroup.`
3. To specify the MIB objects to be added to the group, click the ***Browse*** button next to the ***Objects*** input line. The Object Selection dialog box appears. From the ***Available objects*** list select an object and click the right arrow button. Repeat this procedure to add all objects listed in the ***Available objects*** list. The selected objects will appear in the ***Selected objects*** list and. Click the ***OK*** button and the selected objects will appear also in the ***Objects*** input line.
4. From the ***Status*** drop-down list, select the `current` value and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

Defining MODULE-COMPLIANCE Construct

In order to be able to query all the information that we defined in our MIB module from the SNMP agent that is implemented on the vending machine, the agent must be compliant with the MIB module. This means, that the agent must implement all required definitions. Which definitions are mandatory and which conditional can be specified in the MODULE-COMPLIANCE construct.

1. First, we will add another OBJECT IDENTIFIER node to identify a new branch in the MIB tree in which MODULE-COMPLIANCE constructs will be defined. To do that, select the OBJECT IDENTIFIER component from the Components window (***View*** / ***Components***) and drag-and-drop it onto the `mgVendingMachine` node. The new node will be placed in the MIB tree as a child of the `mgVendingMachine` node. The ***OID*** value will be automatically assigned according to the position of inserted node.

2. Enter `vendingMachineCompliances` into the ***Name*** input line and press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.
3. From the Components window (***View*** / ***Components***) select the MODULE-COMPLIANCE component and drag-and-drop it onto the `vendingMachineCompliances` node. The new node will be placed below the `vendingMachineCompliances` node as its child node. The ***OID*** value will be automatically assigned according to the position of the inserted node.
4. Enter `vendingMachineCompliance` into the ***Name*** input line. Make sure the ***Status*** clause value is set to `current`.
5. Next, click the ***Browse*** button (…) next to the ***Modules*** input line. The Modules dialog box appears. The ***Modules*** list is currently empty but will later list the MIB modules for which compliance requirements are specified. To add the details of compliance requirements, click the ***Add*** and the Module Details dialog appears.
6. Into the ***Module*** input line enter the name of the module we are currently building i.e., `MGSOFT-VENDING-MACHINE-MIB.`
7. To specify the groups of objects that are unconditionally mandatory for implementation in order for the SNMP agent to be compliant with the MIB module, click the ***Browse*** button (…) next to the ***Mandatory groups*** input line. The Group Selection dialog box appears.
8. Select the `objectGroup` in the ***Available groups*** list and clicking the right arrow button. The group will appear in the ***Selected groups*** list. Repeat the same procedure to add the `notificationGroup` group. When done, click the ***OK*** button to save the settings and close the dialog box.
9. As you can se now the `MGSOFT-VENDING-MACHINE-MIB` module is listed in the ***Modules*** list in the Modules dialog box. Click the ***OK*** button to save the settings and close the dialog box.
10. Press the ***Enter*** key or click the ***Apply*** button in the Properties window panel or the changes will not take effect.
11. Your MIB tree should now look like this:

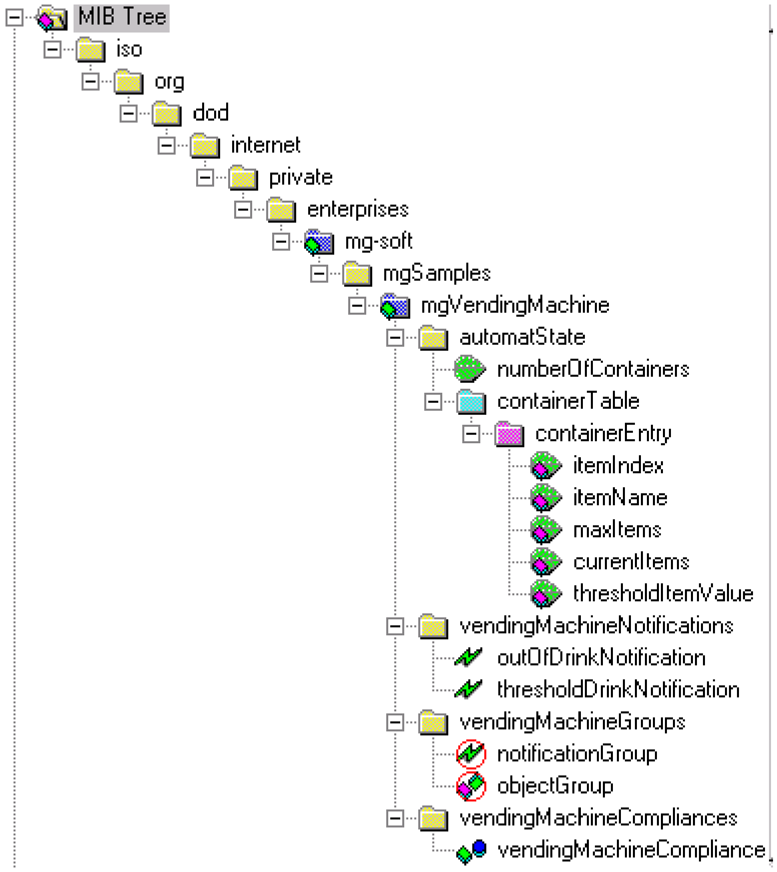


Figure 95: MGSOFT-VENDING-MACHINE MIB tree

Saving the MIB Module

At the end, the MIB module has to be saved to store the changes into a file. Without saving, any changes made to a MIB module will be lost when the MIB module is unloaded or when the application is closed. Before saving the MIB module, we will check if it conforms to the SMIv2 rules.

1. Select the ***Tools*** / ***Check SMI Consistency*** command or click the ☑ toolbar button to start the checking process. Visual MIB Builder will check all definitions in the MIB module. If any inconsistencies should be found during the checking operation, Visual MIB Builder displays corresponding Error, Warning or Info messages in the SMI Consistency Report tab in the Node Preview window (***View*** / ***Node Preview*** / ***SMI Consistency Report tab***). If you followed the instructions in this usage example precisely, no messages should be displayed.

> **Note:** Regardless of number and type of messages, Visual MIB Builder will still let you use the ***File / Export*** command to generate a MIB module definition file, but this definition may not conform to the SMI rules. Therefore, if any messages appear in the SMI Consistency Report tab, refer to the Visual MIB Builder help to learn more about how to deal with them. The Visual MIB Builder help file can be accessed via Help menu.

2. To save the MIB module to a file, select the ***File*** / ***Save As*** command. The standard Save As dialog box appears. By default, MIB files are saved under the name

specified for the MIB module. For convenience, the Visual MIB Builder automatically offers you a file name and destination to which the file can be saved. To specify another name or destination use the corresponding input lines. Note that MIB Builder stores the files in the BUIX file format.

3. Click the ***Save*** button. Visual MIB Builder writes the file to the disk and closes the Save As dialog box.

Exporting the MIB Module

In order to use a MIB module in any application other than Visual MIB Builder, it is necessary to export the MIB module into the MIB module definition file format. MIB module definition files are written in the MIB module definition language and saved in ASCII files. Such MIB files can be compiled by any SMI compliant MIB Compiler tool into a file format used by other applications. For example, MG-SOFT MIB Compiler compiles MIB module definition files into the MG-SOFT's proprietary SMIDB format. An application can access such compiled MIB files by using the industry standard WinMIB interface. When a compiled MIB module is saved to a SMIDB database file, a WinMIB-based application (like MG-SOFT MIB Browser Professional) can access and utilize the database file.

1. Before exporting the MIB module, let’s view all definitions of the current MIB module in the MIB module definition language. Select the ***File*** / ***Export Preview*** command. The Export Preview dialog box appears displaying the preview of the MIB module definition source file. The content of this dialog box can be printed by clicking the ***Print*** toolbar button in the Export Preview dialog box. The standard Print dialog box appears where you can specify the printing parameters.
2. To export the MIB module into MIB module definition source file format, choose the ***File*** / ***Export*** command. The Export dialog box appears. For convenience, the default file name and file type are already specified, so you can just click the ***Save*** button to actually start the export operation.
3. Once the MIB module definition file is created and saved, it is automatically opened in MG-SOFT MIB Compiler application. For more information on the MIB Compiler, refer to the MIB Compiler help file or user manual.

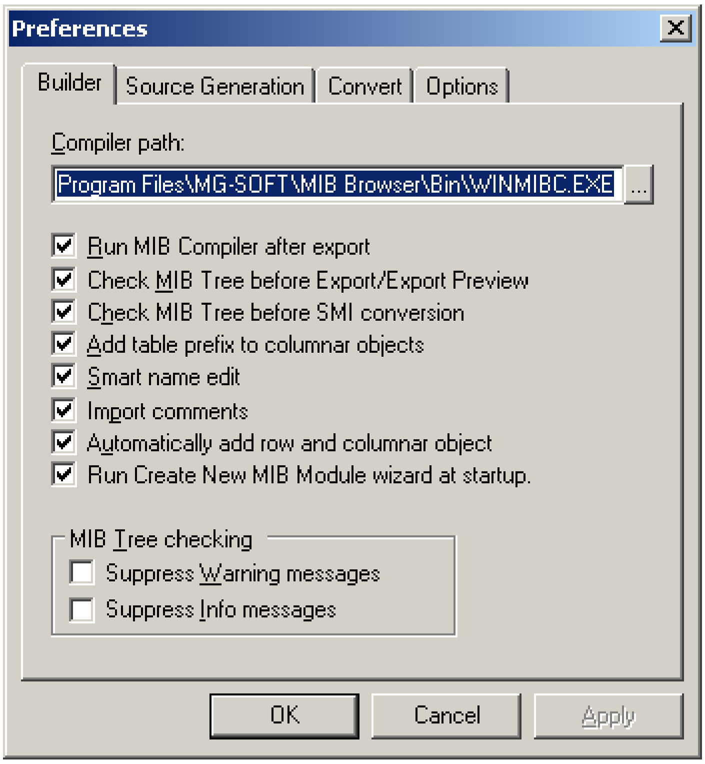
13 SETTING VISUAL MIB BUILDER PREFERENCES

This section will show you how to change the default Visual MIB Builder preference parameters that determine its behavior and thus adjust the application to your needs.

1. Open the Preferences dialog box by using the ***View*** / ***Preferences*** command.
2. The Preferences dialog box contains the following four tabs: Builder tab, Source Generation tab, Convert tab and Options tab. Click the tab containing the preference information you want to change. Specify new settings, and then click the ***Apply*** button to save the changes.

13.1 Builder Tab

1. To set the general MIB Builder options, select the ***View*** / ***Preferences*** command and switch to the Builder tab.



Compiler path: Program Files\MG-SOFT\MIB Browser\Bin\WINMIBC.EXE

☑ Run MIB Compiler after export
☑ Check MIB Tree before Export/Export Preview
☑ Check MIB Tree before SMI conversion
☑ Add table prefix to columnar objects
☑ Smart name edit
☑ Import comments
☑ Automatically add row and columnar object
☑ Run Create New MIB Module wizard at startup.

MIB Tree checking
☐ Suppress Warning messages
☐ Suppress Info messages

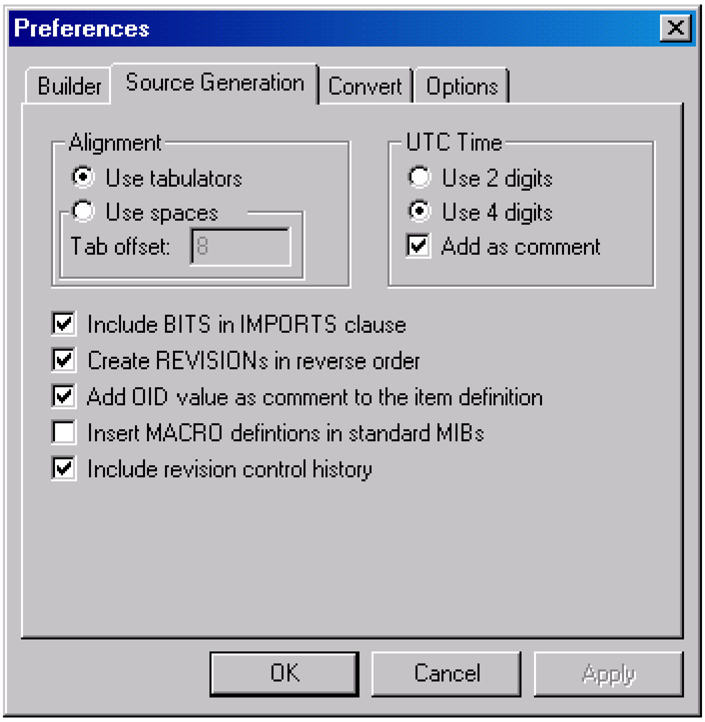
Figure 96: Builder tab

2. The ***Compiler path*** input line displays the path to MG-SOFT MIB Compiler GUI. To modify the path, click the ***Browse*** button ('...') next to the input line and navigate to the MIB Compiler GUI executable (`WinMibC.exe`).
3. If the ***Run MIB Compiler after export*** option is enabled, Visual MIB Builder will, after selecting the ***Export*** command, run MIB Compiler GUI front-end and open the currently created source file in a MIB Editor window where you can make changes or compile it to the WinMIB database.

4. When the ***Check MIB Tree before Export/Export Preview*** checkbox is checked, Visual MIB Builder will check the current MIB module for errors before the Export or Export Preview action is executed.
5. When the ***Check MIB Tree before SMI conversion*** checkbox is checked, Visual MIB Builder will check the current MIB module for errors before converting it from SMIv1 to SMIv2 or from SMIv2 to SMIv1 format.
6. If the ***Add table prefix to columnar objects*** checkbox is checked, Visual MIB Builder will automatically specify the first part of the columnar object name considering the name of the table that contains this column.
7. If the ***Smart name edit*** option is enabled, Visual MIB Builder automatically highlights the part of the node name that is to be changed (Name input line in the Property View panel) and enables you to enter a new name immediately.
8. If the ***Import comments*** checkbox is checked, Visual MIB Builder will, when opening a MIB module database file (*.smidb), also import the comments if they are available in the selected database file. If opening the MIB Module definition file (*.my), the file will be compiled by running MG-SOFT MIB Compiler in the background and the comments will be stored into the SMIDB database file.
9. When the ***Automatically add row and columnar object*** option is enabled, a row and a columnar node are automatically added to each newly added table node.
10. If the ***Run New MIB Module wizard at startup*** option is enabled, the New MIB Module wizard is launched automatically when the Visual MIB Builder application is started.
11. If the ***Suppress Warning messages*** checkbox is checked, MIB Builder will not display Warning messages in the Node Preview window | SMI Consistency Report tab after checking the MIB module for errors.
12. If the ***Suppress Info messages*** checkbox is checked, MIB Builder will not display Info messages in the Node Preview window | SMI Consistency Report tab after checking the MIB module for errors.
13. Note that if both options in the ***MIB Tree checking*** frame are enabled and if any errors are found during the checking process, only Error messages will be displayed in the Node Preview window | SMI Consistency Report tab.
14. After setting all parameters, press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

13.2 Source Generation Tab

1. To modify the contents of the generated MIB module definition file, select the ***View / Preferences*** command and switch to the Source Generation tab.



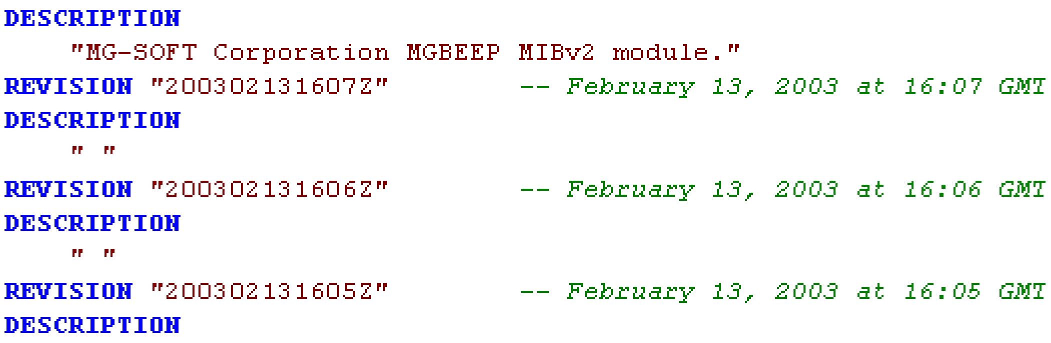
Alignment
● Use tabulators
○ Use spaces
Tab offset: 8

UTC Time
○ Use 2 digits
● Use 4 digits
☑ Add as comment

☑ Include BITS in IMPORTS clause
☑ Create REVISIONs in reverse order
☑ Add OID value as comment to the item definition
☐ Insert MACRO defintions in standard MIBs
☑ Include revision control history

Figure 97: Source Generation tab

2. In the ***Alignment*** frame, you can specify how the text in a MIB module definition file should be aligned.
   - If you select the ***Use tabulators*** radio button, Visual MIB Builder will use tabulators for text alignment.
   - If you select ***Use spaces*** radio button, Visual MIB Builder will use spaces for text alignment. Specify the number of spaces to be inserted for each alignment level in the ***Tab offset*** input line.
3. The MODULE-IDENTITY construct contains the information about the date and the time the MIB module was last updated and when the last revision was made. In the ***UTC Time*** frame, you can specify how the specified values should be displayed in the MIB module definition file.
   - If you select the ***Use 2 digits*** radio button, 2 digits will be used when displaying the year in the UTC time format: YYMMDDHHmmZ.
   - If you select the ***Use 4 digits*** radio button, 4 digits will be used when displaying the year in the UTC time format: YYYYMMDDHHmmZ (Figure 98).
   - If you select the ***Add as a comment*** radio button, the defined UTC time will be added as a comment to the generated MIB module definition file (Figure 98).



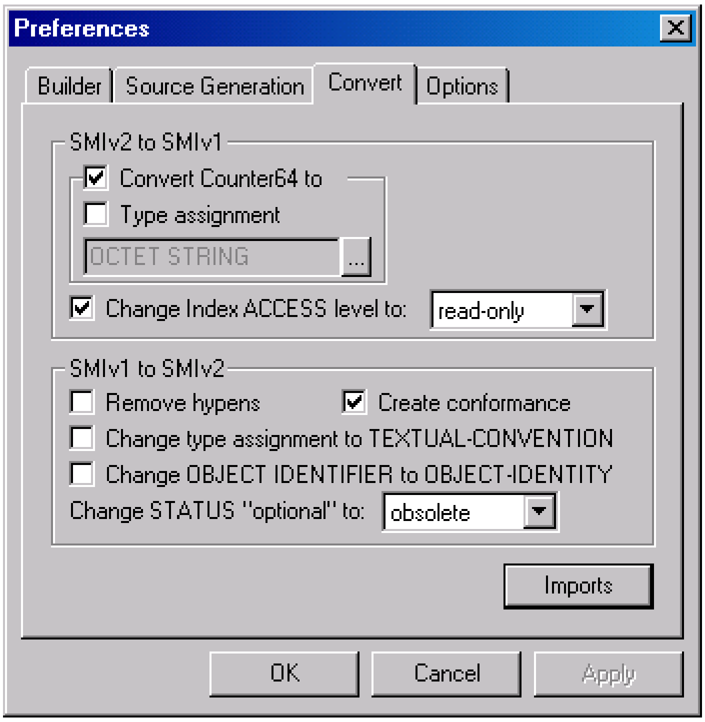
```
DESCRIPTION
    "MG-SOFT Corporation MGBEEP MIBv2 module."
REVISION "200302131607Z"        -- February 13, 2003 at 16:07 GMT
DESCRIPTION
    " "
REVISION "200302131606Z"        -- February 13, 2003 at 16:06 GMT
DESCRIPTION
    " "
REVISION "200302131605Z"        -- February 13, 2003 at 16:05 GMT
DESCRIPTION
```

Figure 98: UTC time in the MIB module definition file

4. If the ***Include BITS in IMPORTS*** option is enabled and the BITS syntax is defined for any imported MIB object, Visual MIB Builder will include BITS in the IMPORTS clause in the generated MIB module definition file.
5. If one or more revisions were made on the target MIB module, then the MODULE-IDENTITY section of the generated MIB module definition file contains all Revision clauses specified. By default, Revision clauses are listed starting with the initial revision toward the last revision made. If you check the ***Create Revisions in reverse order*** checkbox, the Revision clauses are generated in reverse order (Figure 98).
6. If the ***Add OID values as comment to the item definition*** option is selected, OIDs of individual MIB objects will be included as comments to the generated MIB module definition file. These comments will be inserted at the beginning of each MIB object definition.
7. If the ***Insert Macro definitions in standard MIBs*** option is enabled, Visual MIB Builder will insert MACRO definitions in the following MIBs: RFC1212, RFC1155, RFC1215, SNMPv2-SMI, SNMPv2-TC and SNMPv2-CONF.
8. If the ***Include revision control history*** checkbox is checked, and one or more revisions were made on the target MIB module, then the generated MIB module definition file contains a history of all revisions made on that module.
9. After setting all parameters, press the ***Enter*** key or click the ***Apply*** button or the changes will not take effect.

13.3 Convert Tab

1. To specify parameters according to which MIB modules will be converted from one SMI version to another, select the ***View*** / ***Preferences*** command and switch to the Convert tab.



SMIv2 to SMIv1

☑ Convert Counter64 to

□ Type assignment

OCTET STRING ...

☑ Change Index ACCESS level to: read-only

SMIv1 to SMIv2

□ Remove hypens ☑ Create conformance

□ Change type assignment to TEXTUAL-CONVENTION

□ Change OBJECT IDENTIFIER to OBJECT-IDENTITY

Change STATUS "optional" to: obsolete

Figure 99: Convert tab

2. In the ***SMIv2 to SMIv1*** frame, you can specify how the SMI conversion from the SMIv2 to SMIv1 syntax will be performed.

   - By default, the SMIv2 `Counter64` syntax is converted into the OCTET STRING syntax, size 8. If you want the `Counter64` syntax to be converted into any other syntax, check the ***Convert Counter64 to*** checkbox and click the ***Browse*** button to open the Syntax Details dialog box where you can specify the syntax details.
   - If you wish the `Counter64` syntax to be converted into a Type Assignments construct, check the ***Type assignment*** checkbox and specify the Type Assignments name into the input line below. Otherwise, `Counter64` will be converted into the syntax specified in the previous checkbox.
   - SMIv1 requires indexes to be accessible, while SMIv2 prefers that they are not-accessible. If the ***Change Index ACCESS level to*** checkbox is checked, the access level for index objects will be converted into read-only or read-write, according to the option set in the drop-down list.

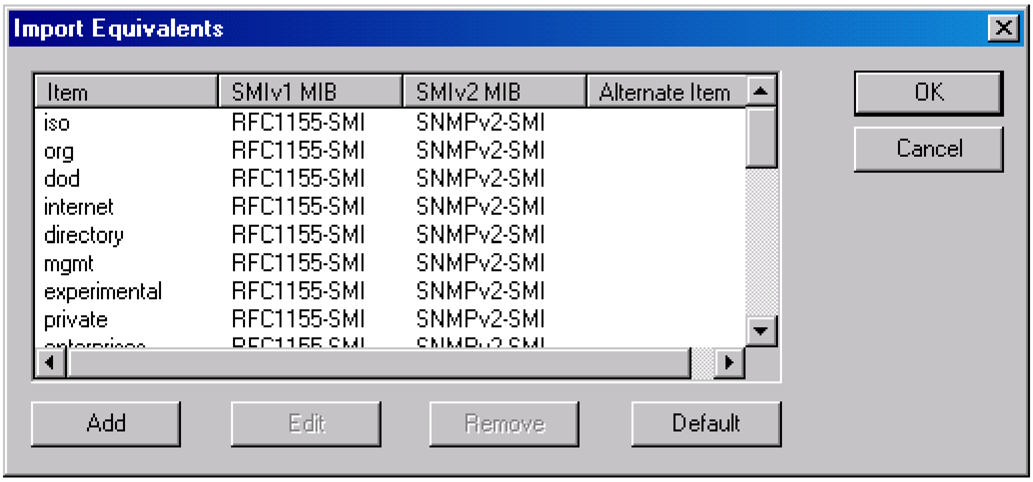
3. In the ***SMIv1 to SMIv2*** frame, you can specify how the SMI conversion from the SMIv1 to SMIv2 syntax will be performed.

- To remove hyphens from the names of SMIv2 MIB objects and enumerated values, check the ***Remove hyphens*** checkbox. Note that changing the name of an object also requires updating all other MIB modules that import those objects.
- If the ***Create conformance*** checkbox is checked, all clauses that are mandatory in the SMIv2 version but are not defined in the current SMI version will be automatically generated during the conversion process, using the Conformance wizard.
- If the ***Change Type Assignments to TEXTUAL-CONVENTION*** checkbox is checked, TYPE ASSIGNMENT constructs are converted into TEXTUAL-CONVENTION constructs.
- If the ***Change OBJECT IDENTIFIER to OBJECT-IDENTITY*** checkbox is checked, OBJECT IDENTIFIER constructs are converted into OBJECT-IDENTITY constructs.
- The status of the SMIv2 MIB objects cannot be set to 'optional'. Using the ***Change status optional to*** drop-down list, you can specify the status into which the 'optional' status will be converted in SMIv2 MIBs. You can choose between 'obsolete' and 'current' status.

4. Click the ***Imports*** button to open the Import Equivalents dialog box (Figure 100) where you can modify the Import statements.

Note: When converting a MIB module from one SMI version to another, Visual MIB Builder automatically changes the IMPORTS statements to refer to the appropriate version of MIB modules, as specified in the Import Equivalents dialog box.

5. The Equivalents dialog box lists all imported objects in the current MIB module and the MIB modules from which an object originates. If an item is defined in both SMIv1 and SMIv2, then both SMIv1 and SMIv2 MIB modules are listed.

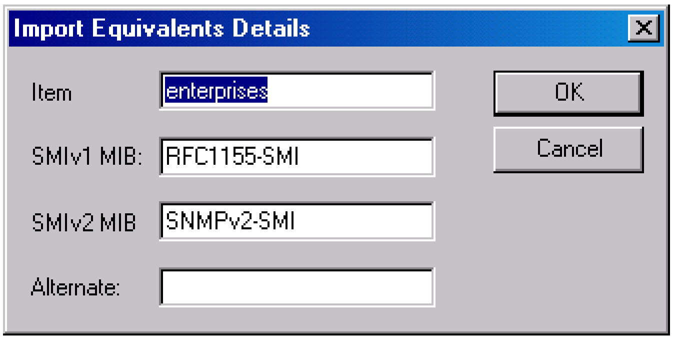


Item	SMIv1 MIB	SMIv2 MIB	Alternate Item
iso	RFC1155-SMI	SNMPv2-SMI	
org	RFC1155-SMI	SNMPv2-SMI	
dod	RFC1155-SMI	SNMPv2-SMI	
internet	RFC1155-SMI	SNMPv2-SMI	
directory	RFC1155-SMI	SNMPv2-SMI	
mgmt	RFC1155-SMI	SNMPv2-SMI	
experimental	RFC1155-SMI	SNMPv2-SMI	
private	RFC1155-SMI	SNMPv2-SMI	
enterprises	RFC1155-SMI	SNMPv2-SMI	

Figure 100: Import Equivalents dialog box

6. To modify the import statements, select a MIB object in the ***Item*** list and use the ***Edit*** button. Clicking the ***Remove*** button will delete the selected item from the list of import equivalents. Clicking the Add and Edit buttons will open the Import Equivalent Details dialog box where you can specify the information needed.

7. Into the ***Item*** input line, enter the name of the imported MIB item (e.g., `enterprises`).



Item: enterprises

SMIv1 MIB: RFC1155-SMI

SMIv2 MIB: SNMPv2-SMI

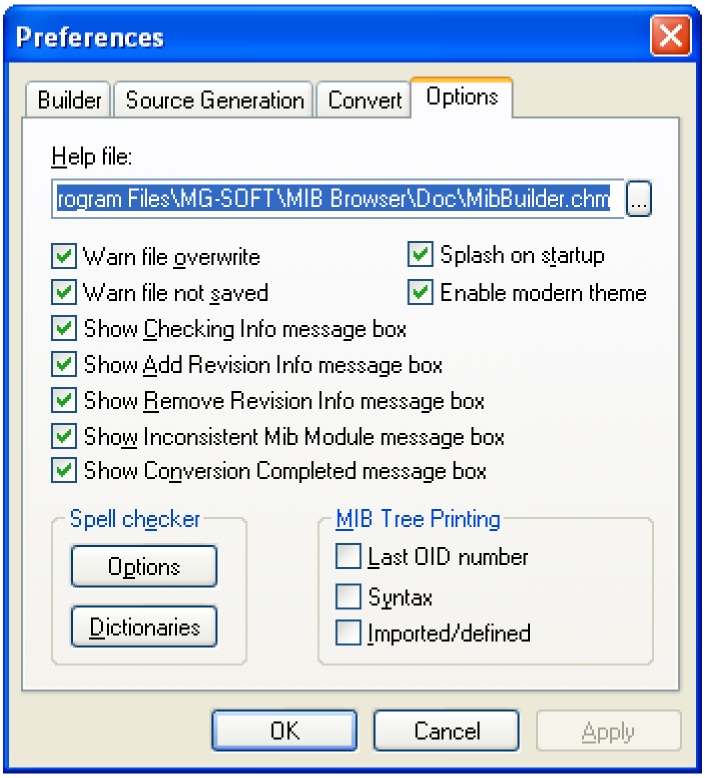
Alternate:

Figure 101: Import Equivalents Details dialog box

8. In the ***SMIv1 MIB*** input line specify the SMIv1 MIB module from which the selected item originates if creating an SMIv1 MIB module.
9. In the ***SMIv2 MIB*** input line specify the SMIv2 MIB module from which the selected item originates if creating an SMIv2 MIB module.
10. If you wish for the name of the imported item to be different after conversion, enter the alternate name into the ***Alternate*** input line.
11. After specifying all information, click the ***OK*** button to save the information and close the dialog box.
12. After setting all parameters, press the ***Enter*** key or click the ***Apply*** button in the Convert tab or the changes will not take effect.

13.4 Options Tab

1. To modify spell checker or MIB tree printing options or to define which pop-up message boxes should be displayed during your work with Visual MIB compiler, select the ***View*** / ***Preferences*** command and switch to the Options tab.



Help file:
rogram Files\MG-SOFT\MIB Browser\Doc\MibBuilder.chm

☑ Warn file overwrite
☑ Splash on startup
☑ Warn file not saved
☑ Enable modern theme
☑ Show Checking Info message box
☑ Show Add Revision Info message box
☑ Show Remove Revision Info message box
☑ Show Inconsistent Mib Module message box
☑ Show Conversion Completed message box

Spell checker

MIB Tree Printing
□ Last OID number
□ Syntax
□ Imported/defined

Figure 102: Options tab

2. The ***Help file*** input line displays the path to MG-SOFT Visual MIB Builder help file. To modify the path, click the ***Browse*** button ("...") next to the input line and browse for the Visual MIB Builder help file (MibBuilder.chm).
3. By default, when saving Visual MIB Builder files (*.buix) or creating new MIB module definition files, Visual MIB Builder displays a warning message box before overwriting an existing file with the same name. To disable this option, uncheck the ***Warn file overwrite*** checkbox.
4. By default, Visual MIB Builder prompts you to save all unsaved files before quitting the application. To disable this option, uncheck the ***Warn file not saved*** checkbox.
5. In case there are any inconsistencies found during the SMI consistency checking operation, Visual MIB Builder by default displays an Info message box informing you about the results of the checking process. To disable this Info message box, uncheck the ***Show Checking Info message box*** checkbox in the Options tab or check the ***Do not show this message again*** checkbox in the Info message box.
6. By default, after selecting the ***Edit*** / ***Revisions*** / ***Add*** command, Visual MIB Builder displays the Add Revision message box containing some basic information about the procedure and prompts you to confirm the selected command. To disable the Add Revision message box, uncheck the ***Show Add Revision Info message box***

checkbox in the Options tab or check the ***Do not show this message again*** checkbox in the Add Revision message box.

7. By default, after selecting the ***Tools / Revisions / Remove*** command, Visual MIB Builder displays the Remove Revision message box prompting you to confirm the deletion. To disable this message box, uncheck the ***Show Remove Revision message box*** checkbox in the Options tab or check the ***Do not show this message again*** checkbox in the Remove Revision message box.
8. When opening a MIB definition source file, Visual MIB Builder may encounter some inconsistent or missing definitions in the file and by default displays a message box informing you about the invalid or missing items in the MIB tree (i.e., broken Type Assignments constructs and/or unlinked nodes). To disable this message box, uncheck the ***Show Inconsistent MIB Module message box.***
9. By default, after converting an SMIv1 MIB module into SMIv2 format, Visual MIB Builder displays the Conversion Completed message box, informing you that some MIB definitions need to be manually reviewed and corrected if necessary. To disable this message box, uncheck the ***Show Conversion Completed message box.***
10. To modify the spell-checker options, which define the way the spell-checker operates, click the ***Options*** button and the Options dialog box will appear prompting you to specify the parameters. For more information on how to modify spell-checker options, please refer to the Options dialog box section of the Visual MIB Builder help file, which can be accessed from the Help menu or by clicking the ***Help*** button in the Options dialog box.
11. To open and close user dictionaries or to edit the contents of an open user dictionary, click the ***Dictionaries*** button in the Spell checker frame. The Dictionaries dialog box appears promoting you to specify some parameters. For more information on the Dictionaries dialog box, please refer to the Dictionaries dialog box section of the Visual MIB Builder help file, which can be accessed from the Help menu or by clicking the ***Help*** button in the Dictionaries dialog box.
12. To specify which information the nodes of printed MIB tree should display, check the checkbox(es) next to the desired options in the ***MIB Tree Printing*** frame. If none of the options is selected, printed MIB tree will display only names of the nodes. The below figure shows how the MIB tree should look when all three options in the MIB Tree printing frame are selected.

Note: After setting all parameters, press the ***Enter*** key or click the ***Apply*** button in the Options tab or the changes will not take effect.

13. By default, the Splash Screen is displayed on Visual MIB Builder startup. To disable this screen, uncheck the ***Splash on startup*** checkbox.
14. If the ***Enable modern theme*** checkbox is checked, MIB Builder graphical user interface displays the Modern visual theme (a set of bitmaps for GUI components such as toolbar buttons, MIB tree icons, menus, etc., that affect the appearance of the application). If this checkbox is not checked, the Classic visual theme is enabled. The Modern theme bitmaps use a significantly higher number of colors than the Classic theme bitmaps. A restart may be required for the change to take effect.

14 INDEX