



[MG-SOFT Corporation](http://www.mg-soft.com)

# **MIB Compiler 2010**

## **USER MANUAL**

(Document Version: 1.8.4)

Document published by MG-SOFT on 02-March-2010

Copyright © 1995-2010 MG-SOFT Corporation

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.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of MG-SOFT Corporation. Permission to print one copy is hereby granted if your only means of access is electronic.

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.

Copyright © 1995-2010 MG-SOFT Corporation. All rights reserved.

---

## TABLE OF CONTENTS

---

<b>1</b>	<b>Introduction</b> .....	<b>5</b>
1.1	Product Description.....	6
1.2	About This Manual .....	7
<b>2</b>	<b>Start MIB Compiler</b> .....	<b>8</b>
2.1	Windows Operating System.....	8
2.2	Linux Operating System.....	9
2.3	Mac OS X Operating System.....	10
<b>3</b>	<b>MIB Compiler Desktop</b> .....	<b>11</b>
3.1	Customizing Desktop .....	12
3.1.1	<i>Docking and Regular Windows</i> .....	13
<b>4</b>	<b>Compile MIB Definition Files</b> .....	<b>15</b>
4.1	Compiling MIB File.....	15
4.2	Compiling Group of MIB Files .....	17
4.2.1	<i>Registering MIB Definition Files</i> .....	17
4.2.2	<i>Compiling Group of MIB Definition Files in Modules Window</i> .....	19
4.2.3	<i>Compiling Multiple MIB Definition Files Using Batch Compile Command</i> .....	20
4.3	Compiling All MIB Modules .....	22
4.4	Specifying MIB Definition File Path or Alias If Prompted While Scanning .....	23
4.5	Saving Compiled MIB Files.....	24
<b>5</b>	<b>Resolve Compilation Problems</b> .....	<b>26</b>
5.1	Examining MIB Compiler Log Messages .....	26
5.2	Fixing Inconsistent MIB Definitions in MIB Editor .....	28
<b>6</b>	<b>Improve Compilation Performance</b> .....	<b>31</b>
6.1	Registering All MIB Definition Files.....	31
6.2	Adjusting MIB Compiler Preferences .....	31
6.2.1	<i>Defining Aliases</i> .....	31
6.2.2	<i>Setting Other MIB Compiler Preferences</i> .....	33
<b>7</b>	<b>(Re)register Compiled MIB Files</b> .....	<b>35</b>
<b>8</b>	<b>Organize Compiled MIB Modules in MIB Groups</b> .....	<b>37</b>
8.1	Create New MIB Group.....	37
8.2	Manage Existing MIB Groups .....	39
<b>9</b>	<b>Delete MIB Modules</b> .....	<b>41</b>
<b>10</b>	<b>Glossary</b> .....	<b>42</b>
<b>11</b>	<b>Index</b> .....	<b>47</b>

---

## TABLE OF FIGURES

---

Figure 1: Launching MIB Compiler from MIB Browser's submenu on Windows.....	8
Figure 2: Tip of the Day dialog box .....	9
Figure 3: Launching MIB Compiler from the Finder on Mac OS X.....	10
Figure 4: MIB Compiler desktop and the Modules window pop-up menu.....	11
Figure 5: An example of a customized MIB Compiler desktop .....	13
Figure 6: Select a MIB definition file to compile .....	16
Figure 7: A compiled MIB module displayed in the Modules and MIB Group windows.....	17
Figure 8: Scan For Source Files dialog box .....	18
Figure 9: Example of a scanning log in the Output window after using the Scan For Source Files command.....	18
Figure 10: Compiling several MIB modules.....	19
Figure 11: Batch-compiling a group of MIB modules .....	20
Figure 12: Batch Compile dialog box .....	21
Figure 13: Specifying a path to MIB definition file .....	23
Figure 14: Compiled MIB Modules dialog box.....	24
Figure 15: Save As dialog box .....	25
Figure 16: Viewing a compilation log.....	26
Figure 17: Debugging in MIB Compiler .....	27
Figure 18: Commenting out selected lines of SMI code.....	29
Figure 19: Defining a MIB module alias .....	32
Figure 20: Setting MIB Compiler preferences .....	33
Figure 21: Scan For Database Files dialog box .....	35
Figure 22: Newly registered MIB modules in the Modules window.....	36
Figure 23: The Modules tab of the MIB Group window.....	37
Figure 24: MIB Tree and MIB Node Properties panels .....	38
Figure 25: Specifying a desired name for the MIB Group .....	38
Figure 26: MIB Groups tab of the Modules window .....	39
Figure 27: Delete MIB Module dialog box .....	41

## 1 INTRODUCTION

---

Thank you for using MG-SOFT MIB Compiler.

MG-SOFT Corporation, established in March 1990, is the world's leading supplier of SNMP, SMI and general network management applications, toolkits and solutions for Windows and Linux 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 software development services and network management integration solutions based on our vast experience in SNMP, SMI and network management fields.

MG-SOFT has developed one of the first SNMPv3 implementations for Win32 platforms and took the initiative by designing extensions to the WinSNMP specification in order to accommodate the SNMPv3 protocol features. With that, MG-SOFT has gained the leading WinSNMP implementation supplier position in the industry. Today, all MG-SOFT's SNMP products fully support the SNMPv3 protocol. Some products also include the Diffie-Hellman key exchange model for management of DOCSIS-based SNMPv3 agents.

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](mailto:info@mg-soft.com)  
URL: <http://www.mg-soft.com/>

---

## 1.1 Product Description

---

MG-SOFT MIB Compiler is a tool that lets you compile any MIB definition file that conforms to the SMIv1 or SMIv2 specification.

MIB definition files are typically provided by vendors of SNMP manageable devices in form of ASCII files and written in MIB module definition language. A MIB definition file contains a description of MIB objects, their attributes, and hierarchy in the SNMP-manageable device, i.e., serving as a roadmap for managing that device. Before such a MIB file can be used, it must be converted to the format an SNMP application can understand and utilize. In case of MG-SOFT applications, this is the SMIDB binary file format.

While compiling, MIB Compiler engine checks the syntax of SMI code in MIB definition files and, if the code is error-free, compiles those files into the SMIDB binary format. Compiled MIB files can then be utilized by all MG-SOFT network management applications or by any other application using the industry standard WinMIB API. These applications use compiled MIB files to graphically represent the MIB objects defined in those files, to meaningfully interpret the results of SNMP operations related to managing these objects, to display properties of MIB objects defined in those MIB files, etc.

Besides several methods of compiling MIBs, MIB Compiler lets you scan MIB definition files to determine if all required MIB definition files are available, register new or moved MIB files, define aliases for MIB modules, etc.

MIB Compiler also incorporates the built-in MIB Editor, which can be used to fix inconsistent MIB definitions in case they cause compilation errors or warnings. MIB Editor window supports SMI syntax coloring, Find/Replace, Bookmarking and many other features that are helpful while editing or writing a MIB module.

Furthermore, MIB Compiler lets you open compiled MIB modules in MIB Group windows to view MIB trees, symbols and traps defined in these modules. In the tree view, you can see the MIB node properties, search the tree for particular MIB object, and print parts or the entire MIB tree along with various node information. You can also organize related MIB modules in MIB groups, to enable more efficient MIB module management in other applications supporting this feature (e.g., MG-SOFT MIB Browser Professional Edition can load or unload all MIB modules in a MIB group at once).

MIB Compiler is available for Microsoft Windows operating systems, for Linux operating systems running on Intel x86 architecture (RedHat, Mandriva, SuSE,...), as well as for Mac OS X operating systems (Intel x86 and PPC).

---


## 1.2 About This Manual

---

This manual will guide you step-by-step through the typical procedures of using MG-SOFT MIB Compiler. It is assumed that you are familiar with basic actions in a graphical desktop environment, such as handling windows, choosing menu commands, dragging and dropping items, etc.

Using MIB Compiler generally does not require any knowledge about the MIB module definition language in which MIB definition files are written. However, for resolving potential compilation problems related to inconsistent MIB definitions, it is an advantage if you are familiar with the MIB module definition language that is specified by the Structure of Management Information (SMI) specification.

Almost all MG-SOFT MIB Compiler operations can be accessed in several possible ways. You can either use:

- Main menu commands (e.g., **Edit / Copy** - the expression “Edit / Copy” means: expand the **Edit** menu in the menu bar and choose the **Copy** command from the menu).
- Toolbar buttons (e.g.,  - to copy selected text in a MIB Editor window, click the **Copy** toolbar button).
- Pop-up menu commands (e.g., **Copy** - to use this command, right-click the selected text in a MIB Editor window to display the pop-up menu and select the **Copy** command from the this menu). On some Mac OS systems you need to press and hold down the **Ctrl** key and left-click the relevant item to display the pop-up menu.
- Keyboard shortcuts (e.g., **Ctrl+C** – press and hold down the **Ctrl** key and press the **C** key; on Mac OS replace the **Ctrl** key with the **Command** key (⌘) on the Apple Macintosh keyboard, i.e., press and hold down the **Command** key and press the **C** key).

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

---

## 2 START MIB COMPILER

---

MIB Compiler is a specialized tool that is shipped and installed with several MG-SOFT products, like MIB Browser Professional, Trap Ringer Professional, SNMP MIB Query Manager, Net Inspector, etc.

MIB Compiler can be started either from the main application (e.g., MIB Browser), or separately as a stand-alone application. Starting MIB Compiler from other MG-SOFT applications is subject to their respective manuals and/or help documentation. This manual explains only the options of launching MIB Compiler as a stand-alone application.

### 2.1 Windows Operating System

---

Under Windows operating systems, start MIB Compiler by selecting the MIB Compiler icon from the Windows **Start** menu:

1. Select the **Start / Programs** command and point to the submenu of the application that includes MIB Compiler (e.g., MG-SOFT MIB Browser).
2. Select the **MIB Compiler** entry from the respective submenu, as shown in the picture below.

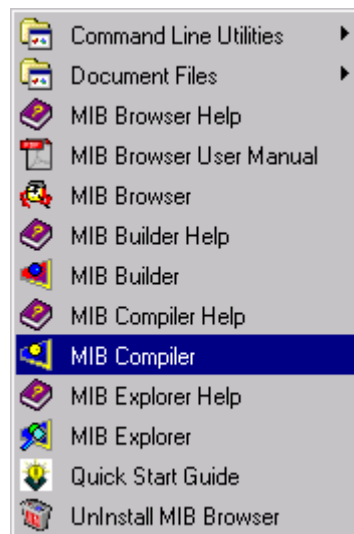


Figure 1: Launching MIB Compiler from MIB Browser's submenu on Windows

3. As the program starts, the MIB Compiler splash screen appears followed by the Tip Of The Day dialog box.

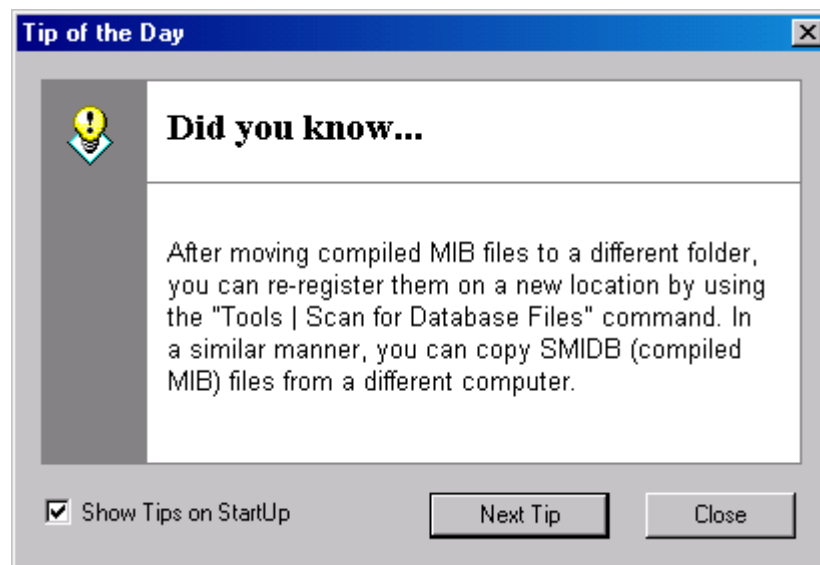


Figure 2: Tip of the Day dialog box

4. To view additional tips, click the **Next Tip** button. After reading the recommendations, click the **Close** button to start using MIB Compiler.
5. The MIB Compiler desktop appears and you can start using the software.

**Tip:** To view MIB Compiler tips after closing this dialog box, select the **Help / Tip of the Day** command.

**Tip:** There are several alternative ways of starting MIB Compiler on Windows. You can do this by double-clicking any of the following files in Windows Explorer:

- winMibC.exe (the MIB Compiler GUI executable file),
- A MIB module definition source file,
- An SMIDB file (a compiled MIB module).

In addition to starting MIB Compiler, option 2 also opens the double-clicked MIB module definition file in a new MIB Editor window, while option 3 opens the double-clicked SMIDB database in a new MIB Group window.

## 2.2 Linux Operating System

1. Under Linux operating systems:
  - ❑ With KDE desktop installed, display the **K** menu from the taskbar and select the MIB Compiler entry from the application submenu containing it, e.g., if MIB Browser is installed, select the **MG-SOFT MIB Browser / MIB Compiler** command.
  - ❑ With GNOME desktop installed, display the **Gnome** menu and use the same procedure as explained in the previous paragraph.
2. As the program starts, the MIB Compiler splash screen appears followed by the Tip Of The Day dialog box (Figure 2).

3. After reading the recommendations in the Tip Of The Day dialog box, click the **Close** button.
4. The MIB Compiler desktop appears and you can start using the software.

**Tip:** You can also start MG-SOFT MIB Compiler by typing `mibcgui` in a terminal window (e.g., `xterm` or `compatible`).

## 2.3 Mac OS X Operating System

Under Mac OS X operating systems:

1. Open the **Finder** and select the **Applications** entry in the panel on the left.
2. Expand the submenu of the application that includes MIB Compiler (e.g., MG-SOFT SNMP Lab) in the Finder and double-click the **MIB Compiler.app** entry to launch MIB Compiler.

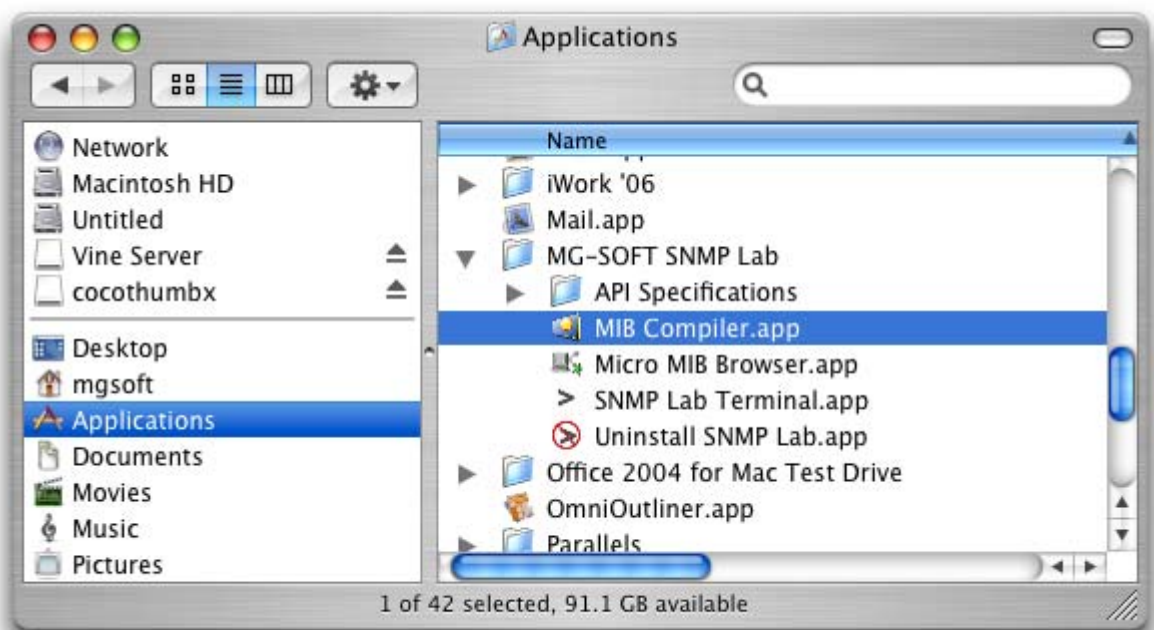


Figure 3: Launching MIB Compiler from the Finder on Mac OS X

3. As the program starts, the MIB Compiler splash screen appears followed by the Tip Of The Day dialog box (Figure 2).
4. After reading the recommendations in the Tip Of The Day dialog box, click the Close button.
5. The MIB Compiler desktop appears and you can start using the software.

### 3 MIB COMPILER DESKTOP

The MIB Compiler main window has a title bar, menu bar, toolbar, status bar, minimize, maximize and close buttons and some areas specific only to MIB Compiler.

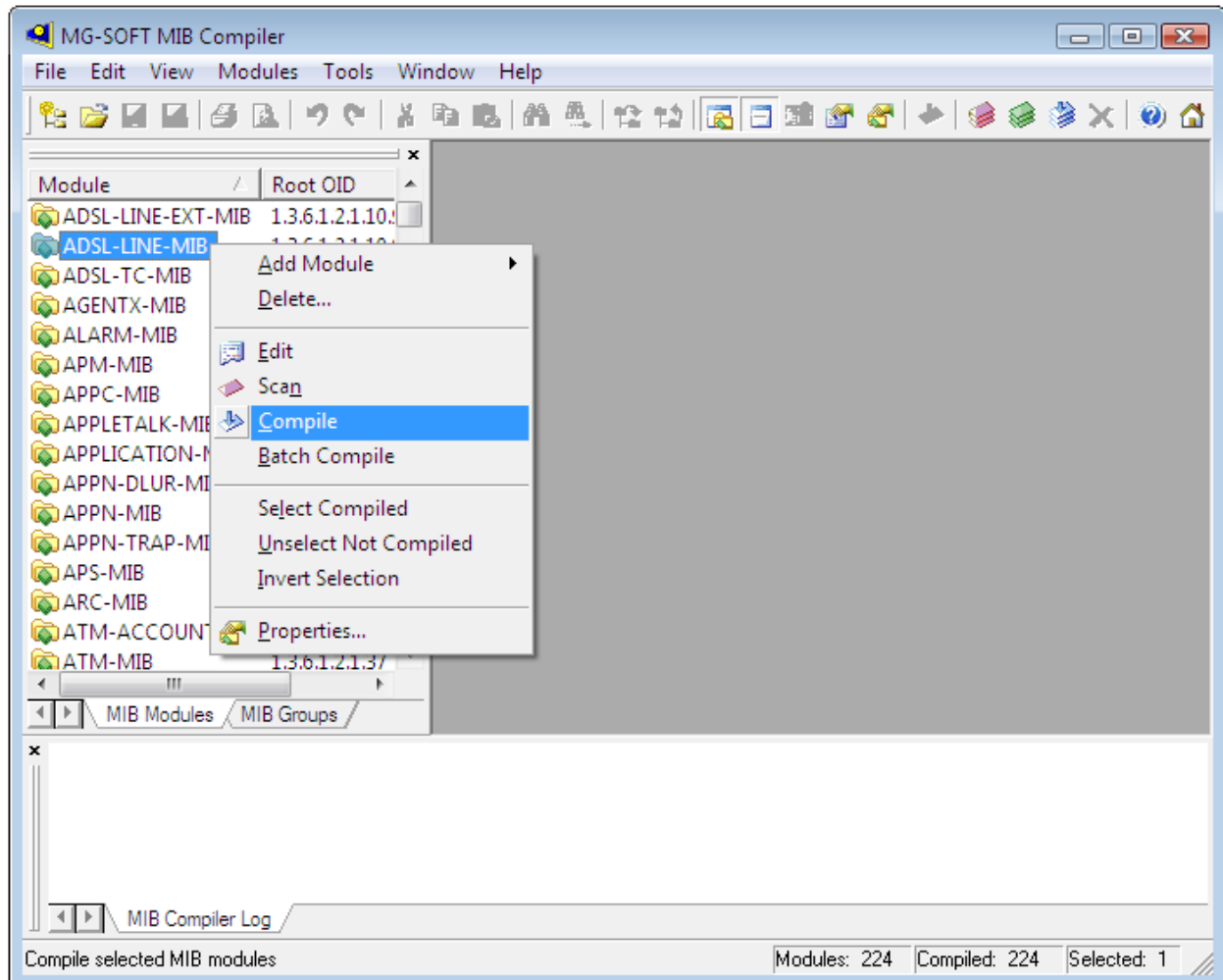


Figure 4: MIB Compiler desktop and the Modules window pop-up menu

After launching MIB Compiler from, it displays the Modules window and the Output (compiler engine) window. Additionally, you can open any number of MIB Editor or the MIB Group windows.

For a detailed description of these windows, please see the MIB Compiler documentation (*Help / Help Topics / MIB Compiler Main Windows*).

#### Menu bar

A menu bar is a bar near the top of the main window that contains names of the program menus, such as File, Edit, View,.... By clicking or activating a menu name via keyboard, a list of commands used to access various functions is displayed.

**Toolbar**

The toolbar contains a group of buttons that provide quick access to the most common commands. You can get a brief description of a command behind each toolbar button, either in a tooltip or in the status bar, by positioning the mouse cursor above the toolbar button (without clicking).

**Working area**

The working area, placed between the toolbar and the status bar, is the area where all application windows are displayed (Modules, Output, MIB Editor and MIB Group windows).

The Modules window shows all registered MIB modules and offers many features for managing them, like selecting, scanning and compiling MIB modules. Additionally, it provides an overview of MIB Groups in the MIB Groups tab.

The Output window contains a log generated by the MIB Compiler engine during a MIB compilation or scanning. You can step through log messages to determine if the operation was successful and, if not, locate the possible problems.

MIB Group windows are used for displaying MIB trees consisting of various MIB modules. In addition to that, they display MIB module and MIB node properties. Modules from MIB Group windows can be saved to MIB groups – logical MIB units that can be used by other applications.

MIB Editor windows are used for editing MIB module definition (source) files directly in the MIB Compiler environment. They provide SMI syntax coloring, bookmarking, compiling MIB definition files directly from the MIB Editor window, and other features that are helpful when editing inconsistent MIBs.

The working area can be fully customized to meet the user's specific needs, as described in the [Customizing Desktop](#) section.

**Status bar**

A status bar is a bar at the bottom of the main window that primarily shows the status of the currently performed operation. It is context sensitive, meaning that it displays different information, according to the object selected or operation performed in the main window. Depending on this, it also contains one or more status bar fields. For example, if a MIB module in the Modules window is selected, the status bar shows the following information in the status bar fields (from left to the right): the number of all, the number of compiled, and the number of selected MIB modules in the Modules window.

## 3.1 Customizing Desktop

---

All MIB Compiler windows can be re-arranged, re-sized, or hidden in order to produce a desktop layout that meets your requirements best.

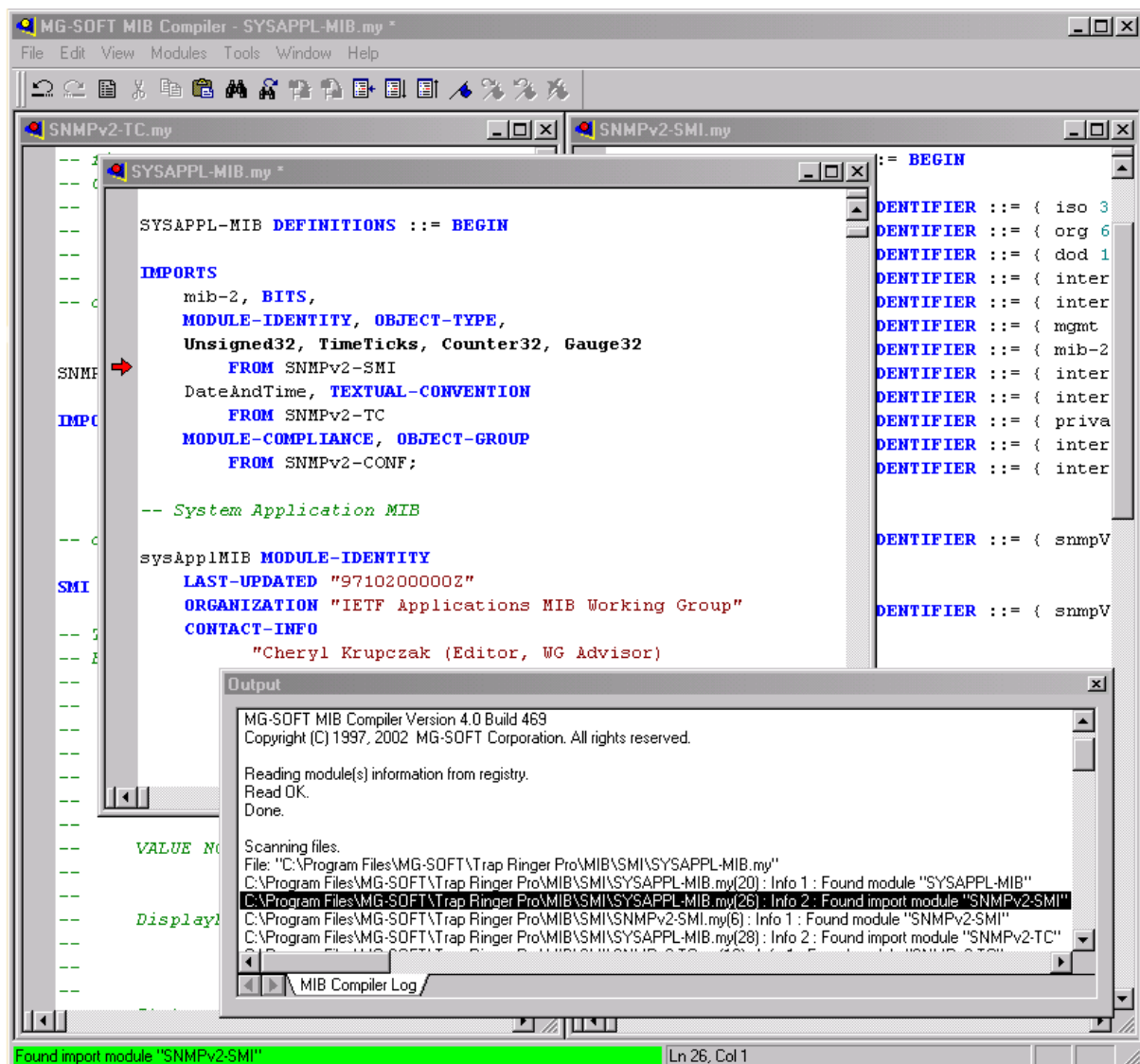


Figure 5: An example of a customized MIB Compiler desktop

### 3.1.1 Docking and Regular Windows

Generally, there are two types of application windows in MIB Compiler: the docking windows and the regular windows.

A docking window is a special window type that is always (when opened) on top of the regular windows and has two display modes: floating and docked. In the floating mode, a docking window can be placed to any position on the screen (also outside the borders of the main window); while in the docking mode, it is bound (docked) to the inner borders of the main window.

The Modules and the Output windows are docking windows, while the MIB Editor and MIB Group are regular windows.

Only one instance of each docking window can be opened in MIB Compiler, i.e., one Modules and one Output window. In contrast to that, you can open numerous instances

of MIB Editor and MIB Group windows. To do this, select a MIB module in the Modules window and select the **Edit** or the **Add Module / New MIB Group** pop-up command.

To move a docked window to a new position, do the following:

1. Double-click the gray title bar of a docked window to change its mode to floating.
2. The window undocks and the title bar changes its color.
3. Drag the floating docking window to the desired position on the screen. While dragging it, MIB Compiler will try to dock it to the currently nearest border of the main window. If you want to prevent the docking, hold down the **Ctrl** key while dragging it.

To close a docking window, click the **Close** (x) button in its title bar. You can re-display it by selecting the **View / Modules** or **View / Output** command.

---

**Note:** MIB Compiler automatically stores the properties of the main window and the docking windows (opened/close status, position, size). Therefore, they will reappear the same after restarting the application.

---

## 4 COMPILE MIB DEFINITION FILES

With MG-SOFT MIB Compiler you can compile any MIB definition (source) file compliant with the SMIv1 or SMIv2 language specifications. Such MIB definition files are typically supplied by the vendors of SNMP-manageable devices. MIB Compiler lets you compile MIB definition files into MG-SOFT proprietary format (SMIDB), which can be utilized by MG-SOFT applications or any other applications utilizing the industry standard WinMIB interface.

MIB Compiler offers several ways of compiling MIB module definition files. This section explains these compiling options. First, a procedure of compiling a single MIB definition file is described. Then, two methods of compiling a group of MIB definition files are explained: using the regular Compile command and using the Batch Compile command specially optimized for large compilation tasks. Next, the option of compiling all registered MIB modules using the Batch Compile command is described. This section also points out how to scan and register MIB module definition files before compiling them. The section ends with a short description of how to save compiled SMIDB files in case they are not saved immediately after the compilation.

**Note:** It is important to note the difference between a MIB module, which is a general term that stands for a definition of related managed objects, and the corresponding MIB files: a MIB module definition (source) file and a MIB module database (compiled) file.

A MIB module definition file is a plain ASCII text file written in MIB module definition language as specified by the SMI specification. It contains a definition of MIB objects, their attributes and hierarchy. In order to use such a MIB module in an SNMP application, it must be compiled to a data format, which this application understands. In our case, this is the SMIDB binary format. This means that there will be typically two physical files for every MIB module, a source and a compiled MIB file.

All bundled MIB modules come in both file versions, as compiled and as source MIB files. These MIB modules are listed in MIB Compiler Modules window, because their compiled and source MIB files were registered at the time of installing MIB Compiler.

### 4.1 Compiling MIB File

To compile a single MIB definition file (e.g., a MIB definition file provided by a vendor of an SNMP-manageable device), do the following:

1. Select the **File / Compile** command, which prompts you with the standard Open dialog box.
2. Use the Open dialog box (Figure 6) to browse to the MIB definition file you want to compile. Select the desired file and use the **Open** button.

**Tip:** Alternatively, you can compile a MIB definition file by right-clicking it in Windows Explorer and selecting the **Compile** pop-up command.

**Tip:** Choose the **All Files (\*.\*)** entry from the **Files of type** drop-down list if the extension of the MIB file you want to compile differs from the extensions indicated in this list by default.

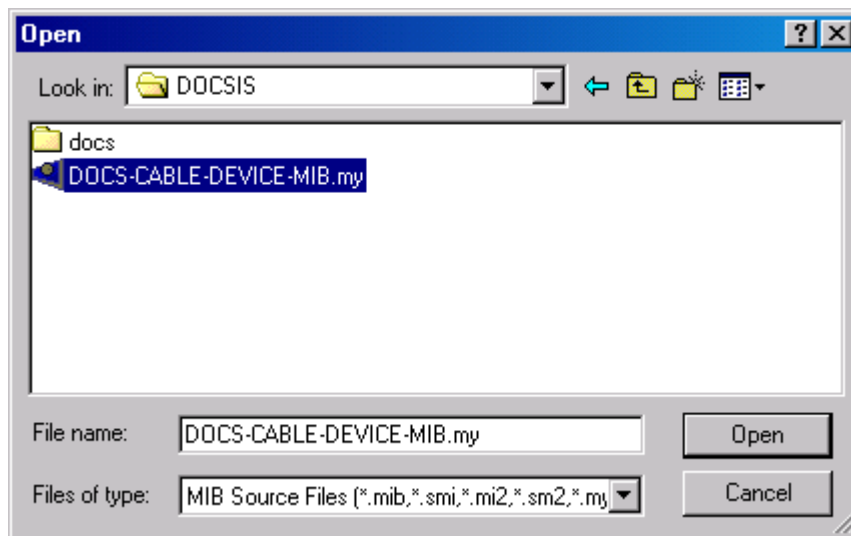


Figure 6: Select a MIB definition file to compile

3. By default, MIB Compiler first scans the selected MIB file. This operation scans the selected MIB module definition file for the modules referenced in its `IMPORTS` clause and recursively in the `IMPORTS` clauses of all imported modules. While scanning, MIB Compiler writes the scanning log to the Output window and stores the relevant information to the application data files.
4. If the Scan operation finishes successfully, the Compile operation starts. While compiling, MIB Compiler logs the compilation activities in the Output window.

**Tip:** In case the Edit Module Properties dialog box appears while scanning, follow the steps described in the *Specifying MIB Definition File Path or Alias If Prompted While Scanning* section.

**Tip:** In case the compilation process terminates due to an error (MIB Compiler log displays one or more Error messages), please refer to the [Resolve Compilation Problems](#) section for the guidelines on how to eliminate errors.

5. After a successful compilation, the compiled MIB module is displayed in a new MIB Group window. This operation also registers the MIB definition (source) file and displays it in the Modules window with a light-colored icon ([Figure 7](#)).

**Note:** Light-colored icons in the Modules window indicate the MIB modules, which are partly registered, i.e., having only the MIB module definition file registered. When the corresponding compiled MIB module file is saved, it is also automatically registered and the MIB module's icon turns from light-colored to normal.

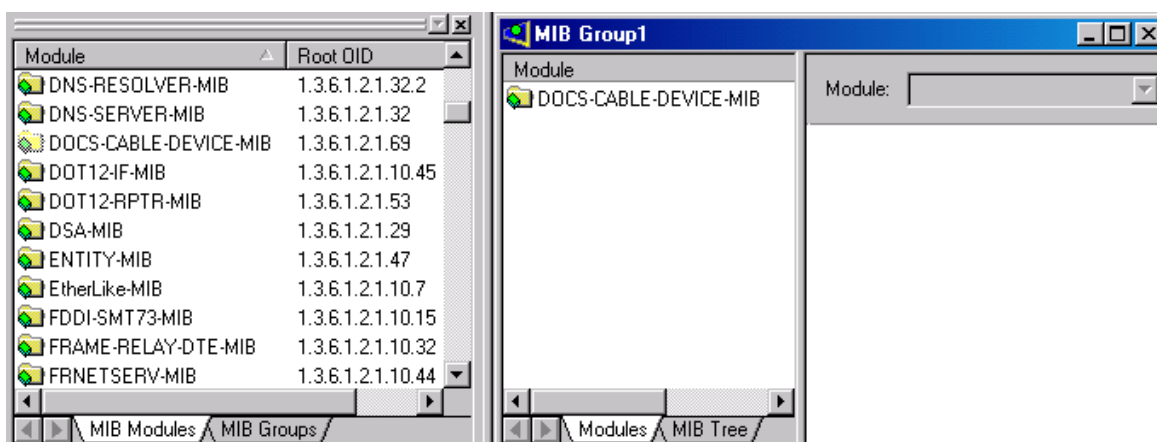


Figure 7: A compiled MIB module displayed in the Modules and MIB Group windows

- By default, after a successful compilation, MIB Compiler also displays the Compiled MIB Modules dialog box allowing you to save the compiled MIB file. To save it, follow the instructions specified in the [Saving Compiled MIB Files](#) section.

## 4.2 Compiling Group of MIB Files

To be able to compile a group of MIB definition files from the Modules window, you must scan and register them first (if not registered already). This operation associates MIB definition file names and paths with the names of MIB modules (as specified in those definition files). This information is stored in the application data files for future use. After this, the (partly) registered MIB modules appear in the Modules window and you can compile them using the **Compile** or the advanced **Batch Compile** command, as described in the following sections.

By default, the Compile operation includes the Scan operation, which scans all selected MIB definition files for their `IMPORTS` clauses and checks if all MIB modules referenced by these `IMPORTS` clauses are available. This is done recursively, i.e., also for all imported MIB modules. If the Scan operation finishes successfully, the MIB definition files are compiled.

**Note:** The **Scan** command can also be run independently, i.e., not as a part of the Compile operation.

This section ends with a description of using (both versions of) the **Batch Compile** command to compile a larger group of MIB definition files.

### 4.2.1 Registering MIB Definition Files

- To register a group of MIB definition (source) files, select the **Tools / Scan For Source Files** command.

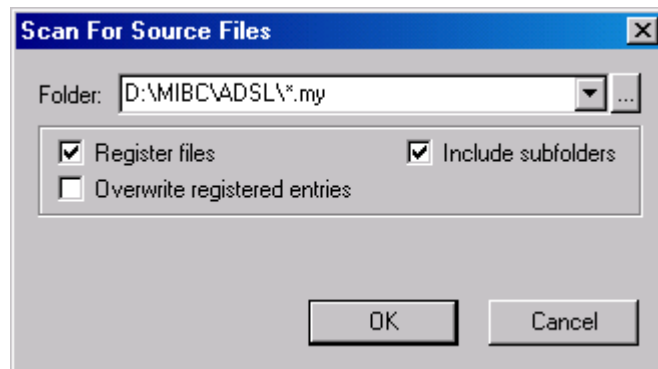


Figure 8: Scan For Source Files dialog box

2. Within the Scan For Source Files dialog box specify the following:

- ❑ Into the Folder drop-down list specify the full path of the folder containing MIB definition files. To narrow down the Scan operation to files with specific file names and extensions, change the default file mask (\*.\*) to a desired one (e.g., \*.my, \*.m\*, etc.).
- ❑ To register found MIB definition files, i.e., write the information regarding the MIB module names and associated file paths to corresponding application data files, check the **Register files** checkbox.
- ❑ To search for MIB definition files in all subfolders of the specified folder, check the **Include subfolders** checkbox.
- ❑ If MIB Compiler finds any MIB definition files that are already registered, it will overwrite the relevant information in application data files if you check the **Overwrite registered entries** checkbox.

**Tip:** Use the “...” button next to the **Folder** drop-down box to browse to the desired folder.

3. After specifying the necessary details, select the OK button.

4. While scanning and registering MIB definition files, MIB Compiler engine writes a scanning log to the Output window.

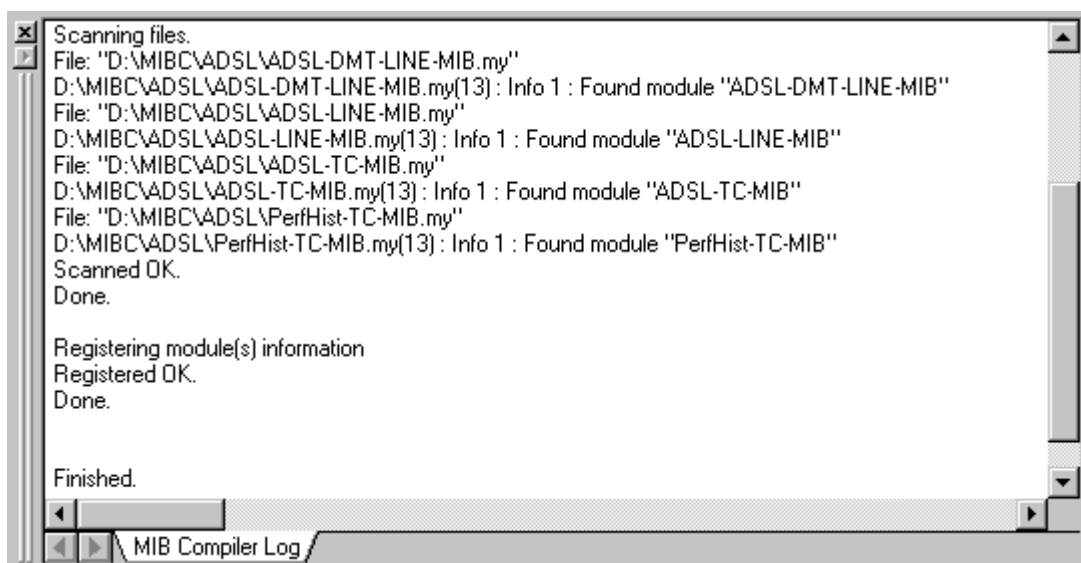
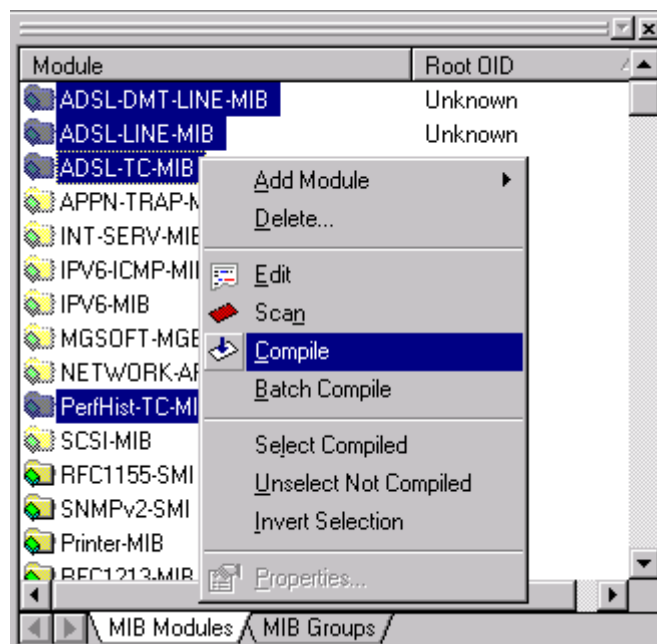


Figure 9: Example of a scanning log in the Output window after using the Scan For Source Files command

After successfully registering MIB definition files, they appear in the Modules window represented by light-colored MIB module icons (note that they have `Unknown` Root OID until compiled).

## 4.2.2 Compiling Group of MIB Definition Files in Modules Window

1. To compile a group of MIB definition files, select them within the Modules window and choose the **Modules / Compile** command (or the **Compile** pop-up command).



**Tip:** To quickly select all MIB modules, which are not compiled, first choose the **Modules / Select Compiled** and then the **Modules / Invert Selection** command.

Figure 10: Compiling several MIB modules

2. MIB Compiler engine starts scanning selected modules and logs its scanning activity in the Output window. This type of Scan operation scans the definition files of selected MIB modules for the modules listed in their `IMPORTS` clauses and in the `IMPORTS` clauses of all imported modules.

**Tip:** If MIB Compiler prompts you with the Edit Module Properties dialog box, follow the steps described in the [Specifying MIB Definition File Path or Alias If Prompted While Scanning](#) section.

3. If the Scan operation finishes successfully, the Compile operation starts. While compiling, MIB Compiler engine logs the compilation activities and results in the Output window.
4. By default, MIB Compiler prompts you with the **Compiled MIB Modules** dialog box after a successful compilation. This enables you to save compiled MIB files as described in the [Saving Compiled MIB Files](#) section of this document.

### 4.2.3 Compiling Multiple MIB Definition Files Using Batch Compile Command

To compile multiple MIB modules at once, you should use the Batch Compile command, which is optimized for large compilation tasks.

**Note: Compile vs. Batch Compile**

The **Modules / Compile** command invokes the MIB Compiler engine twice for every selected MIB module: once for the Scan operation, and if this completes successfully, again for the Compile operation. In every invocation, MIB Compiler engine also reads and updates the application data files.

Unlike the **Modules / Compile** command, the **Modules / Batch Compile** command compiles all selected MIB modules in one pass, i.e., the MIB Compiler engine is invoked only once. This significantly speeds up the compilation process when compiling a large number of MIB modules. Note also that the Batch Compile process does not include the Scan operation.

There are two versions of the Batch Compile command in MIB Compiler:

- ❑ The **Modules / Batch Compile** command
- ❑ The **Tools / Batch Compile** command

Both versions are designed for compiling larger groups of MIB definition files. However, the **Modules / Batch Compile** command can compile only registered MIB modules that are displayed within the Modules window, whereas the **Tools / Batch Compile** command can compile non-registered MIB modules as well. The latter also incorporates the Batch Compile dialog box, where you can set several compilation options, as described below.

#### Modules | Batch Compile

To use the **Modules / Batch Compile** command, follow these steps:

1. To batch-compile a group of registered MIB definition files, select them within the Modules window and choose the **Modules / Batch Compile** command (or the **Batch Compile** pop-up command).

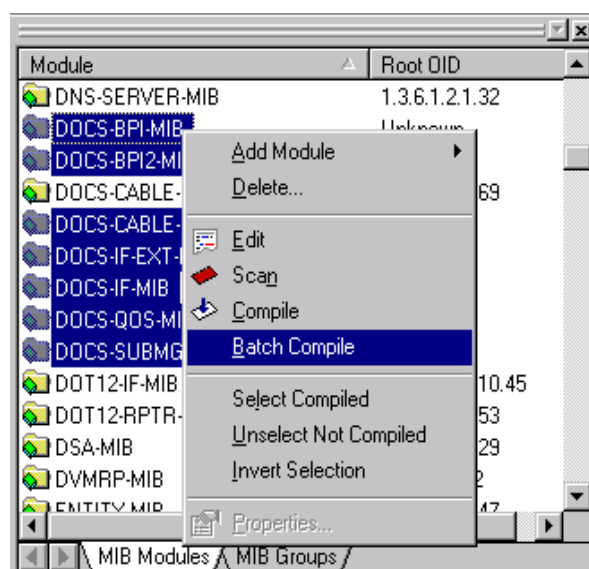


Figure 11: Batch-compiling a group of MIB modules

- The Batch Compile operation starts. This operation reads the information about selected MIB modules from the application data files and starts compiling them in a single pass. While compiling, MIB Compiler engine logs its activity in the Output window.

**Note:** The Batch Compile command starts compiling MIB definition files without scanning them first. If MIB Compiler encounters a MIB definition file requesting to import a definition from an unregistered MIB module, MIB Compiler generates an Error message and aborts compiling such MIB definition file. However, this does not terminate the entire batch-compilation process unless the number of compilation errors exceeds the maximal number set in the Preferences dialog box (default value: 100). To avoid such errors, select the desired set of MIB modules and run the **Modules / Scan** command before batch-compiling them.

- By default, the Compiled MIB Modules dialog box appears after finished compilation (i.e., if at least one MIB definition file has been compiled successfully). To save compiled MIB module files listed in this dialog box, follow the instructions specified in the [Saving Compiled MIB Files](#) section.

## Tools | Batch Compile

To use the **Tools / Batch Compile** command, which is the fastest method of compiling and registering a group of MIB definition files, do the following:

- To compile a large group of MIB definition files stored in the same folder (or its subfolders), select the **Tools / Batch Compile** command. This prompts you with the Batch Compile dialog box.

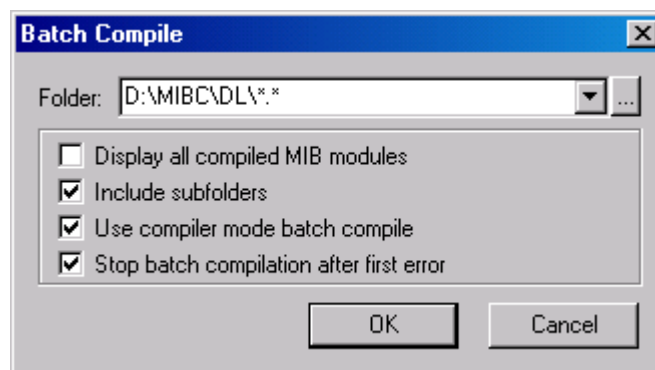


Figure 12: Batch Compile dialog box

- Into the **Folder** drop-down list enter the full path to the folder containing MIB definition files. To narrow down the Batch Compile operation to files with specific file names and extensions, change the default file mask (\*.\*) to a desired one (e.g., \*.mib, \*MIB.m\*, etc.).
- Tip:** Use the “...” button next to the **Folder** drop-down list to browse to the desired folder.
- Within the Batch Compile dialog box, you can check the following checkboxes to control the compilation process:
    - To display all compiled MIB modules in a new MIB Group window after compilation, check the **Display all compiled MIB modules** checkbox.

- ❑ To compile the files that match the given file mask in the specified folder and all its subfolders, select the **Include subfolders** checkbox.
  - ❑ Check the **Use compiler mode batch compile** to compile all MIB module definition files in one pass (same as the **Modules / Batch Compile** command). If this checkbox is not checked, MIB Compiler engine is invoked for every MIB module it compiles.
  - ❑ To stop the batch compilation process after the first error is encountered, check the **Stop batch compilation after first error** checkbox.
4. After specifying all information, click the **OK** button.
  5. The Batch Compile operation starts. While compiling, MIB Compiler engine logs its activities in the Output window.

**Note:** This checkbox can only be used in combination with the **Use compiler mode batch compile** checkbox.

**Note:** The Batch Compile command starts compiling selected MIB definition files without scanning them first. While compiling, MIB Compiler may encounter a MIB definition file demanding to import a definition from a MIB module, which is not registered. In this case the attempt to compile such a MIB definition file will fail and an error message will be generated. However, if the **Stop batch compilation after first error** checkbox is not checked, this will not terminate the Batch Compile operation altogether (unless the maximum number of permitted errors is reached). To minimize the chance of experiencing batch-compile errors, you should run the **Tools / Scan for Source Files** command before compiling MIB files (as described in the [Registering MIB Definition Files](#) section).

6. By default, the Compiled MIB Modules dialog box appears after finished compilation (i.e., if at least one MIB definition file has been compiled successfully). To save compiled MIB module files listed in this dialog box, follow the instructions specified in the [Saving Compiled MIB Files](#) section.

### 4.3 Compiling All MIB Modules

Sometimes it may be necessary to recompile all registered MIB modules (e.g., if compiled MIB files become somehow corrupted etc.):

1. Select the **View / Preferences** command to open the Preferences dialog box, where you can verify or set the compilation preferences for this operation.
2. Within the Compiler tab of the Preferences dialog box, set the following:
  - ❑ To force MIB Compiler to compile already compiled modules, uncheck the **Skip compiled MIB modules** checkbox.
  - ❑ To compile also all imported MIB modules, check the **Compile imported MIB modules** checkbox.
  - ❑ To modify the maximal number of errors that MIB Compiler tolerates before terminating the compilation process, enter a new value into the **Max errors** input line. It may be useful to increase this value if compiling a large number of MIB definition files.
3. Click the **OK** button to close the Preferences dialog box.

4. Select the **Modules / Compile All** command (or the **Compile All** pop-up command within the Modules window).
5. MIB Compiler starts batch-compiling all registered MIB definition files. While compiling, MIB Compiler engine logs the compilation activity and results in the Output window.

**Note:** While compiling, MIB Compiler may encounter some errors. However, this will not terminate the compilation process, unless the number of compilation errors exceeds the maximal number set within the Preferences dialog box (default value: 100).

6. By default, the Compiled MIB Modules dialog box appears after a compilation is finished. To save compiled MIB module files listed in this dialog box, follow the instructions specified in the [Saving Compiled MIB Files](#) section.

## 4.4 Specifying MIB Definition File Path or Alias If Prompted While Scanning

During the scanning phase of a compilation process or while only scanning, MIB compiler may encounter an `IMPORTS` clause requesting to import a definition from a MIB module, which is not registered. In such a case, MIB compiler prompts you with the Edit Module Properties dialog box to enter a MIB definition file path or to specify an alias for this module. If prompted with the Edit Module Properties dialog box ([Figure 13](#)), proceed as follows:

1. Carefully read the **Module identity** line, which states the requested MIB module name as found in the `IMPORTS` clause.
  - ❑ Into the **Module source path** input line enter a full path of the MIB definition file defining this module, or
  - ❑ Define an alias. That is, select another MIB module, which will be used instead of the original, whenever MIB Compiler comes across an `IMPORTS` clause requesting to import a definition from the original file. To define an alias, check the **Aliased** checkbox to activate the drop-down list and select proper MIB module from it.

**Tip:** Use the “...” button to browse to the desired file.

An alias is another name for a MIB module that substitutes the original.

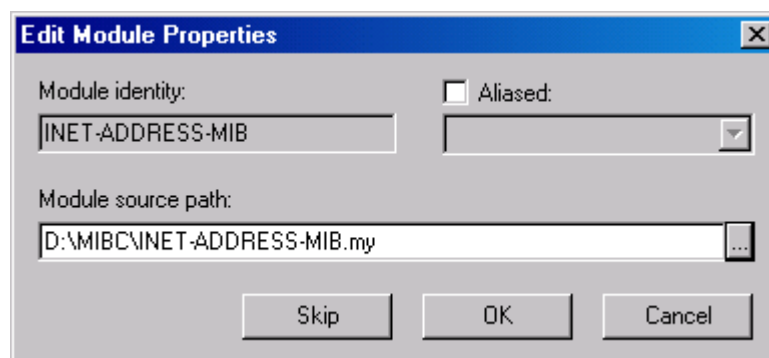


Figure 13: Specifying a path to MIB definition file

2. After specifying a proper file path or alias, click the **OK** button.
3. The Scan operation will be automatically restarted, taking the newly specified information into account.

**Note:** In case you do not enter a proper alias or a file path, or you select the **Cancel** or the **Skip** button in the Edit Module Properties dialog box, the scanning process fails. If the scanning process was a part of a compilation process, the compilation is terminated as well.

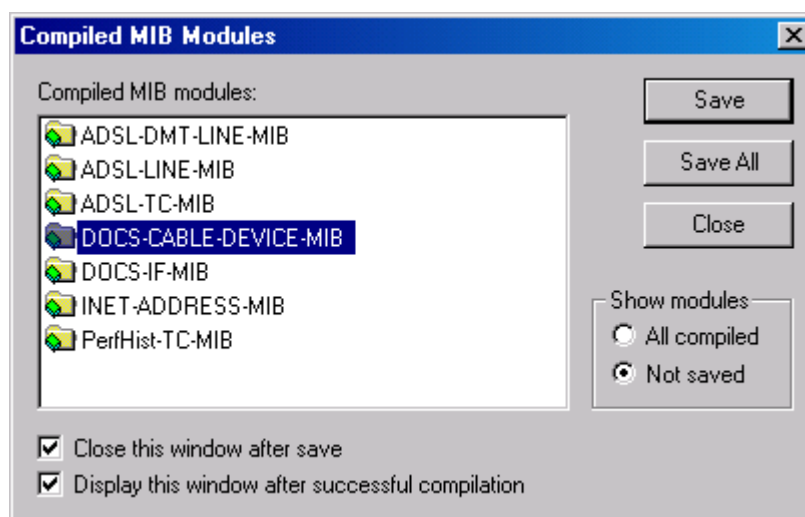
## 4.5 Saving Compiled MIB Files

In order to use MIB modules in WinMIB-based applications, you must save compiled (SMIDB) MIB files. While saving SMIDB files, MIB Compiler also registers them, i.e., it stores their file names and paths to the application data files in order for WinMIB-based applications (e.g., MG-SOFT MIB Browser) to be able to find and use them.

If not automatically prompted after a successful compilation (see the tip below), you should save compiled MIB files manually.

**Tip:** By default, MIB Compiler prompts you with the Compiled MIB Modules dialog box after every successful compilation in order to immediately save compiled MIB files. However, you can turn this feature on and off in MIB Compiler. To do this, select the **View / Preferences** command to display the Preferences dialog box. On the Compiler tab of this dialog box, check or uncheck the **Display save dialog** checkbox to enable or disable the automatic appearance of Compiled MIB Modules dialog box after compilation.

1. By default, MIB Compiler prompts you with the Compiled MIB Modules dialog box after a successful compilation in order to save compiled MIB files. If not prompted, you can manually open the Compiled MIB Modules dialog box by selecting the **File / Save MIBs** command.



**Tip:** Use the **Save All** button to save all MIB modules currently displayed in the **Compiled MIB modules** list.

Figure 14: Compiled MIB Modules dialog box

2. In the Compiled MIB Modules dialog box you can set the following:
  - ❑ To view all MIB modules compiled in the current MIB Compiler session, irrespective of whether they have already been saved or not, select the **All compiled** radio button. The default setting (**Not saved** radio button selected) displays only compiled MIB modules, which have not been saved yet.
  - ❑ To close Compiled MIB modules dialog box after saving MIB modules, check the **Close this window after save** checkbox.
  - ❑ If you want the Compiled MIB modules dialog box to appear automatically after every successful compilation, check the corresponding checkbox.
3. To save compiled MIB files, select them in the **Compiled MIB modules** list and select the **Save** button. This prompts you with the Save As dialog box.

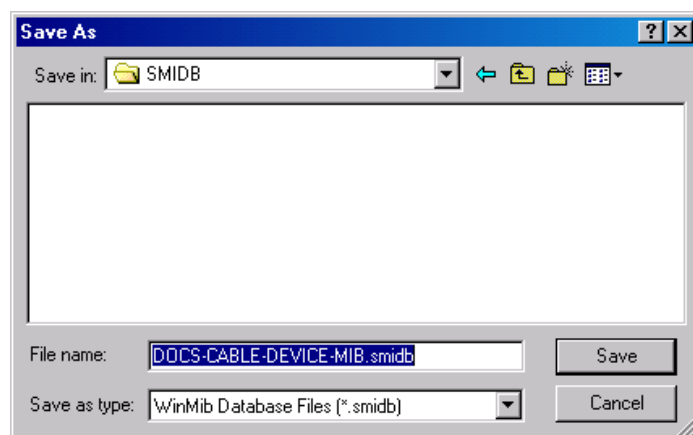


Figure 15: Save As dialog box

4. In the Save as dialog box (Figure 15) specify the save destination path and the file name of the first module to be saved. Then select the **Save** button, to actually write the SMIDB file(s) to disk.

**Note:** If you select more than one MIB file for saving, you will be prompted to specify the save destination path and file name for each of them. To save them to the same location as the first file and to accept the file names offered by MIB Compiler, simply use the **Save** button each time you are prompted.

5. When saving compiled MIB files, MIB Compiler also stores their file names and paths to the application data files. When saved, you can open these MIB modules in a new MIB Group window to view their MIB trees and properties or load and utilize them in other WinMIB based applications.

**Note:** The light-colored MIB module icons within the Modules window change to normal when the corresponding compiled MIB files are saved. Normal icons indicate MIB modules that have both, source and compiled MIB files properly registered.

6. If you have been automatically prompted to save MIB files after compilation, you should check the MIB Compiler Log in the Output window after saving compiled MIB files. When doing this, you can use the F11 and F12 keys to check if MIB Compiler generated any Warning or Error messages during the compilation process, as described in the [Examining MIB Compiler Log Messages](#) section.

## 5 RESOLVE COMPILATION PROBLEMS

Generally, MG-SOFT MIB Compiler will successfully compile all MIB definition files that are written according to the SMIv1 or SMIv2 specification rules. If a MIB definition file cannot be successfully compiled, it most probably means that it is not consistent (i.e., not properly written) and that it must be fixed.

This section describes the basic steps of resolving typical compilation problems related to inconsistent MIB definition files. The first step in this “debugging” process is to check the MIB Compiler log messages to detect problems and examine them. If compilation problems occur due to improperly written MIB definitions, these MIB definitions can be fixed in the built-in MIB Editor, as illustrated in the [Fixing Inconsistent MIB Definitions](#) section.

### 5.1 Examining MIB Compiler Log Messages

While compiling or scanning MIB files, MIB Compiler logs its activity in the Output window. The log may contain any number of Info, Warning, or Error messages as well as other information related to the compilation process. You should always check the output log for Error or Warning messages and, if present, examine them to determine their causes:

1. In the Output window, locate the line, which indicates the number of Error and Warning messages generated during the compilation process (as highlighted in [Figure 16](#)).

**Note:** This line is not logged if you use the Batch Compile command. In such case, proceed with step 2.

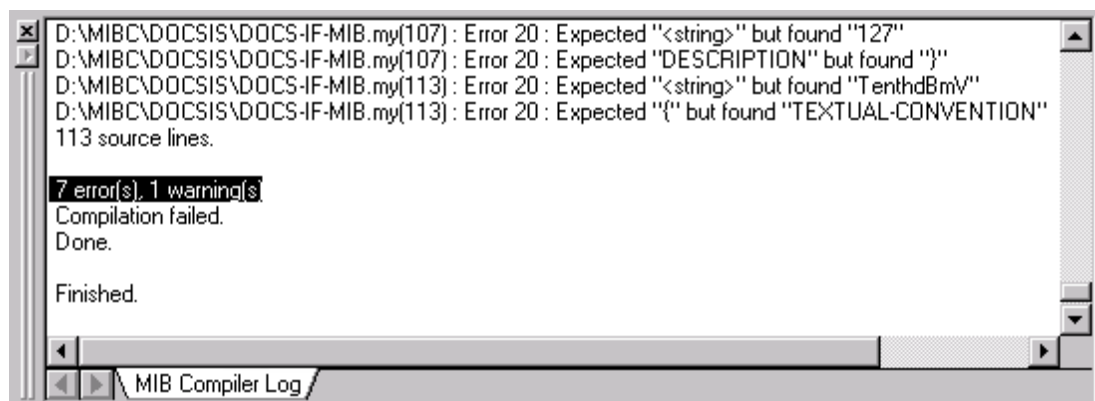


Figure 16: Viewing a compilation log

- ❑ If no errors and no warnings are indicated in this line, the compilation has succeeded without any problems and you can skip the following steps.
- ❑ If no errors, but one or more warnings are indicated in this line, the compilation has succeeded, however, MIB Compiler has found some inconsistencies, which should be eliminated. To do this, refer to the step 5.

- If one or more errors are indicated in this line, the compilation has failed. You should proceed with the following steps to locate and examine the logged Error/Warning messages.
- 2. To view and highlight the first Error message in the compilation log, press the **F12** key (on Mac OS: **⌘+F12**). By pressing the **F12** or **Shift+F12** keys (on Mac OS: **⌘+F12** and **Shift+⌘+F12**), you can jump to the next or previous Error message in the Output window.
- 3. When an Error message is highlighted, MIB Compiler opens the corresponding MIB definition file in the MIB Editor window and points out the problematic SMI line of that file (see the red arrow marker in [Figure 17](#)). This enables you to examine and fix the problem.

**Tip:** Use the **F11 / Shift+F11** (Mac OS: **⌘+F11 / Shift+⌘+F11**) keys to step through Warning and Error messages, or the **F4 / Shift+F4** keys to step through all log messages (Error, Warning and Info messages).

**Note:** The highlighted Error message is also displayed in the red colored MIB Compiler status bar field ([Figure 17](#)). The status bar also displays the (best guess of) line and column number of the MIB definition file, where the error occurred.

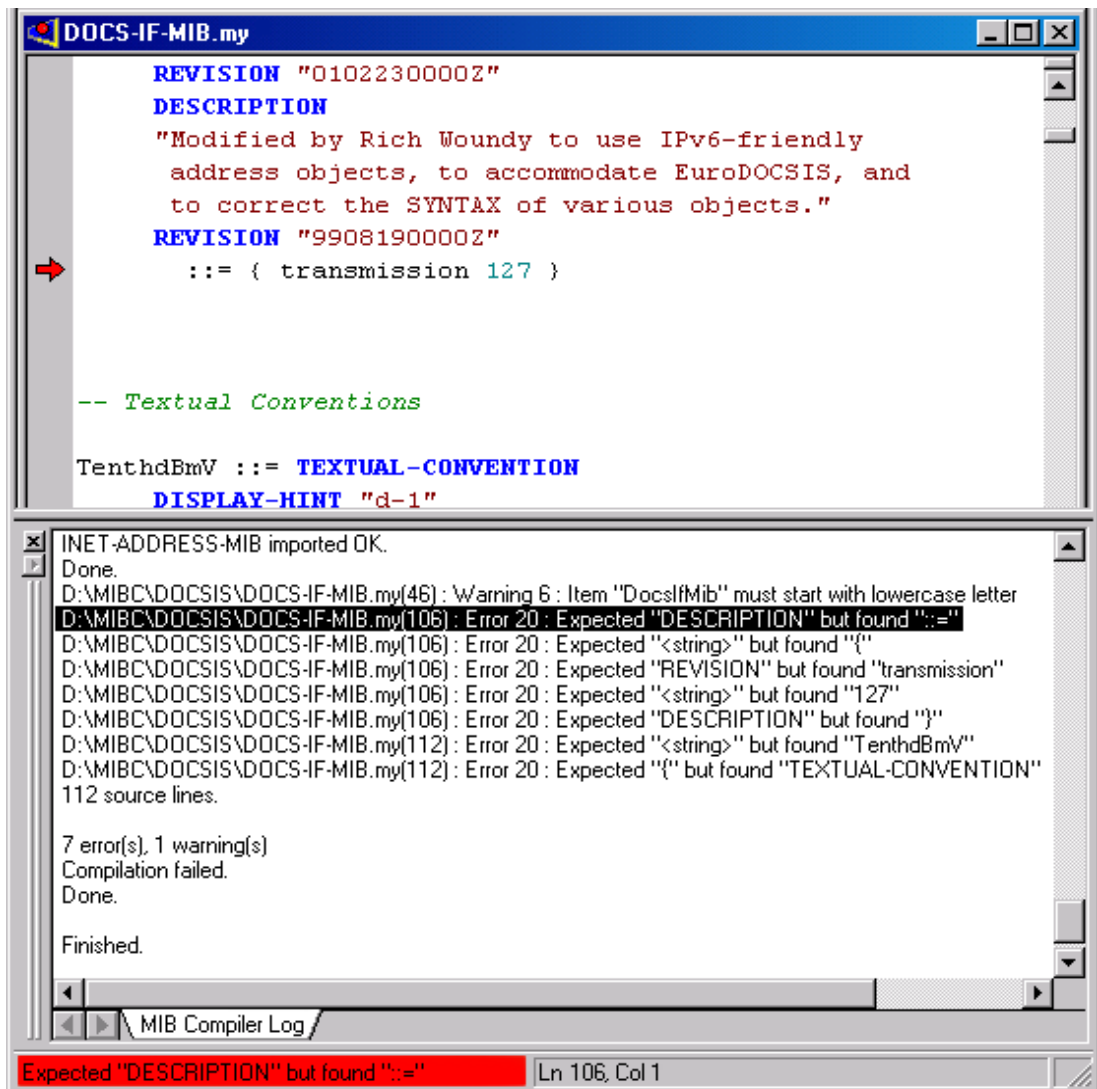


Figure 17: Debugging in MIB Compiler

**Tip:** For the meaning of the Error message codes generated by MIB Compiler, please refer to the MIB Compiler help documentation. To do this, use the **Help / Help Topics** command to open the Help dialog box and switch to the **Command Line MIB Compiler | MIB Compiler Messages | Error Messages** topic. This topic explains the meaning of all MIB Compiler Error codes.

### Example: Error 20

The Error Messages help topic states the following:

```
Error 20: Expected 's' but found 's'
```

```
MIB Compiler expected a different token than was found.
```

The Error message highlighted in the [Figure 17](#) therefore means that the expression in the line 106 of the DOCS-IF-MIB definition file is incorrect according to the SMI rules (this line should contain a DESCRIPTION clause instead of the ::= token).

4. If the encountered error is a consequence of improperly written MIB definition (as in the example above), you can fix this definition in the MIB Editor window, as illustrated in the [Fixing Inconsistent MIB Definitions](#) section.

**Tip:** If you do not know how to fix the MIB definition by yourself, you should contact the organization responsible for this MIB module and request the updated, properly coded version of the MIB definition file.

5. To view and examine the potential Warning messages, use the same procedure as described in steps 1-4. However, instead of using the **F12** key (on Mac OS: **⌘+F12**) in step 2 and referring to the Error Messages topic in MIB Compiler help in step 3, you should use the **F11** key (on Mac OS: **⌘+F11**) and refer to the description of the Warning message codes.

## 5.2 Fixing Inconsistent MIB Definitions in MIB Editor

This section illustrates how to fix errors in a MIB definition file. To fix inconsistent parts of SMI code within a MIB definition file, follow these steps:

1. To open the inconsistent MIB definition file in the MIB Editor window (if not already opened), press the **F12** or **F11** key (on Mac OS: **⌘+F12** or **⌘+F11**) or double-click an Error or a Warning message in the Output window. Note that MIB Compiler automatically locates the problematic SMI line within the opened MIB file and marks it with a red arrow marker.
2. Edit the inconsistent part of SMI code in the MIB Editor window so that it will comply with the SMIv1 or SMIv2 specification rules. When you modify a MIB definition file, an asterisk (\*) appears in the MIB Editor window title bar, indicating that the file has been changed ([Figure 18](#)).

**Note:** MIB Compiler has a built-in database of all SMIv1 and SMIv2 keywords. While writing or editing MIB definitions, MIB Compiler automatically recognizes parts of the SMI code and applies pre-defined colors and styles to them (syntax coloring). You can customize the syntax coloring preferences in the MIB Editor tab of the Preferences dialog box, or turn this option on and off (**SMI Syntax Coloring** pop-up command). If you notice that a part of SMI code is not formatted according to the syntax coloring settings, when this option is turned on, it means that this particular part of code does not comply with the SMI rules and that you should fix it (check the usage of lowercase and uppercase letters, spelling etc.).

**Tip 1:** Use **Bookmarks**, for example, to mark the problematic lines of a MIB definition file, so you can quickly find them thereafter.

**Tip 2:** To comment out a line in the MIB definition file, insert two adjacent hyphens in front of it or select it and use the **Comment Out** pop-up command, as illustrated in the figure below.

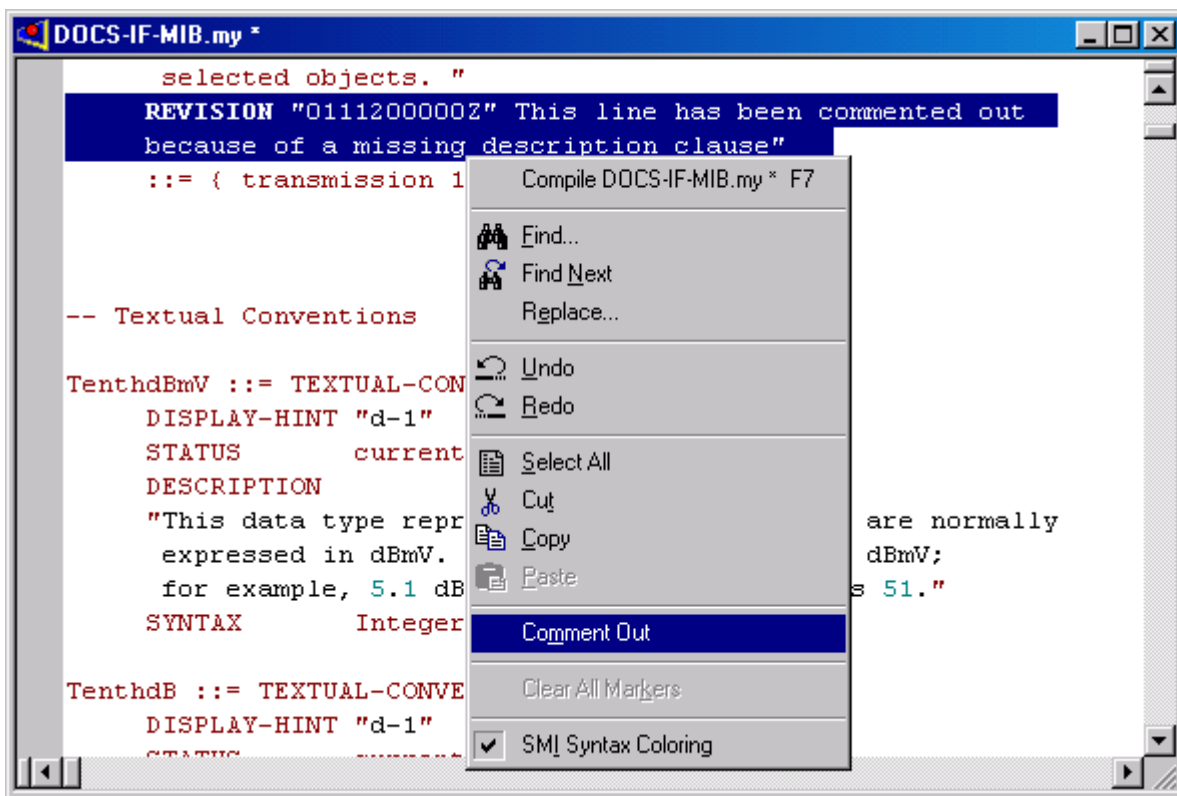


Figure 18: Commenting out selected lines of SMI code

- To save changes made in the MIB definition file, select the **File / Save Source** command. To save the file under another name use the **File / Save Source As** command and specify the file name and save destination into the Save as dialog box.

**Note:** When you save the modified MIB definition file, the asterisk mark disappears from the Editor window title

**Tip:** In the MIB Editor tab of the Preferences dialog box, you can check the **Auto save before compile** checkbox to let MIB Compiler automatically execute the **File / Save Source** command before it recompiles a changed MIB definition file.

4. Rescan/recompile the changed MIB definition file using the **Compile <name of the MIB file>** pop-up command or the **F7** key.
5. Check the MIB Compiler log for Error or Warning messages using the **F12** and **F11** (on Mac OS: **⌘+F12** and **⌘+F11**) keys as described in the [Examining MIB Compiler Log Messages](#) section.
  - ❑ If you still find any Error or Warning messages, repeat the procedure described in the steps 1-5.
  - ❑ If no more Error or Warning messages are present in the log, you should save the compiled MIB file(s), as specified in the [Saving Compiled MIB Files](#) section.

---

## 6 IMPROVE COMPILATION PERFORMANCE

---

To eliminate or reduce the need for user intervention while compiling, you should scan and register all third-party MIB definition files before compiling any of them. In addition, there are also several MIB Compiler preference settings, which can improve the compilation performance as explained in the following sections.

---

### 6.1 Registering All MIB Definition Files

---

During the scanning phase of a compilation process or while only scanning, MIB Compiler may prompt you with the Edit Module Properties dialog box to enter a MIB definition file path or an alias. This occurs if the path to the MIB module listed in the `IMPORTS` clause is not known (i.e., the required MIB definition file is not registered and no alias is defined for it).

**Tip:** For a description of the Edit Module Properties dialog box, please refer to the Specifying MIB Definition File Path or Alias If Prompted While Scanning section.

Therefore, before compiling any MIB module, it is recommended to register all available MIB definition files using the **Tools / Scan For Source Files** command, as described in the [Registering MIB Definition Files](#) section. This operation scans for MIB definition files in a specified folder or drive and stores the information about MIB definition files it finds to the application data files. More specifically, it writes the names and full paths of found MIB definition files to the corresponding application data files, in order for MIB Compiler and other MG-SOFT applications to be able to find and use them thereafter.

Registering all MIB definition files can significantly reduce the occurrence of prompting with the Edit Module Properties dialog box while compiling or scanning. It also diminishes the possibility that Batch Compile operations will fail

---

### 6.2 Adjusting MIB Compiler Preferences

---

---

#### 6.2.1 Defining Aliases

---

MIB module aliases enable you to compile MIB modules with non-standard or incorrect MIB module names specified in their `IMPORTS` clauses, without having to modify their SMI code. An alias is another name for a MIB module. To define an alias means to specify another MIB module, which will be used instead of the original, whenever MIB Compiler comes across an `IMPORTS` clause requesting to import a definition from the original module.

MIB Compiler comes with several pre-defined aliases, which let you bypass the most frequently found inconsistencies in MIB `IMPORTS` clauses.

You can also define new aliases or modify or remove the existing ones to enable a smooth compilation in any specific case.

To define a new alias, do the following:

1. Select the **View / Preferences** command to open the Preferences dialog box and switch to the MIB Modules and Aliases tab (Figure 19).
2. To activate the list of aliases, check the **Use additional MIB module source file paths and aliases** checkbox.
3. Use the **Add** button to open the Edit Module Properties dialog box, where you can define a new alias (Figure 19).
4. Into the **Module identity** input line enter the name of the MIB module for which you want to define an alias.
5. Check the **Aliased** checkbox to activate the drop-down list. In this list, specify the name of the MIB module that will be used instead of the module specified in the previous step. You can select among already registered MIB module names (recommended) or enter a new name.

**Note:** If you enter a name of not registered MIB module, you should register it before starting the compilation.

**Tip:** The option of defining the **Module source path** is described in the Specifying MIB Definition File Path or Alias If Prompted While Scanning section.

6. After specifying above details, click the **OK** button twice. This closes both dialog boxes and applies new settings. From this point on, MIB Compiler will use the aliased MIB definition file instead of the one specified in the **Module identity** input line (e.g., Figure 19: whenever MIB Compiler finds an `IMPORTS XY` from `RFC1493` clause, it will import the `XY` definition from the `BRIDGE-MIB` definition file).

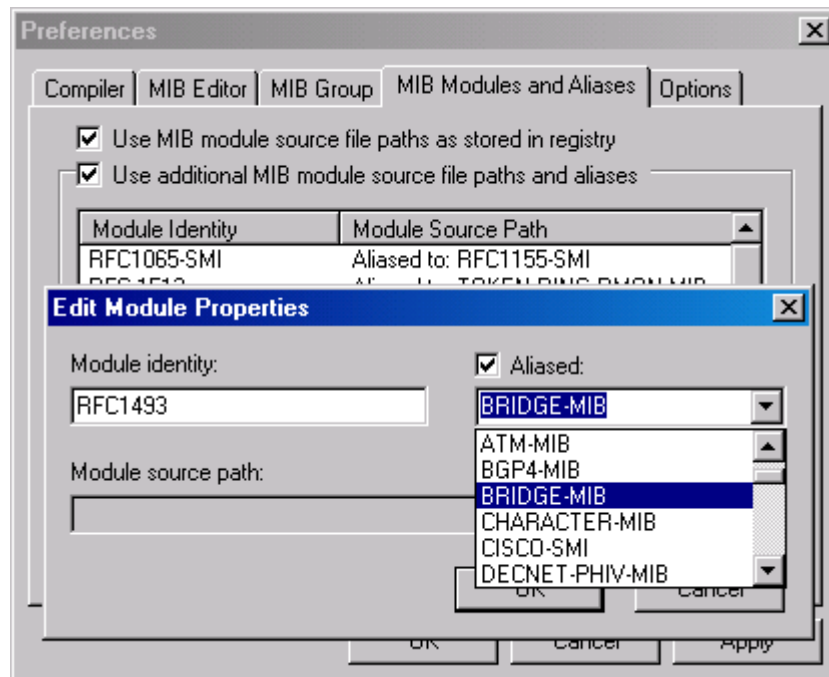


Figure 19: Defining a MIB module alias

## 6.2.2 Setting Other MIB Compiler Preferences

In the Compiler tab of the Preferences dialog box, you can set various compilation options before compiling MIB definition files. To display this dialog box, use the **View / Preferences** command.

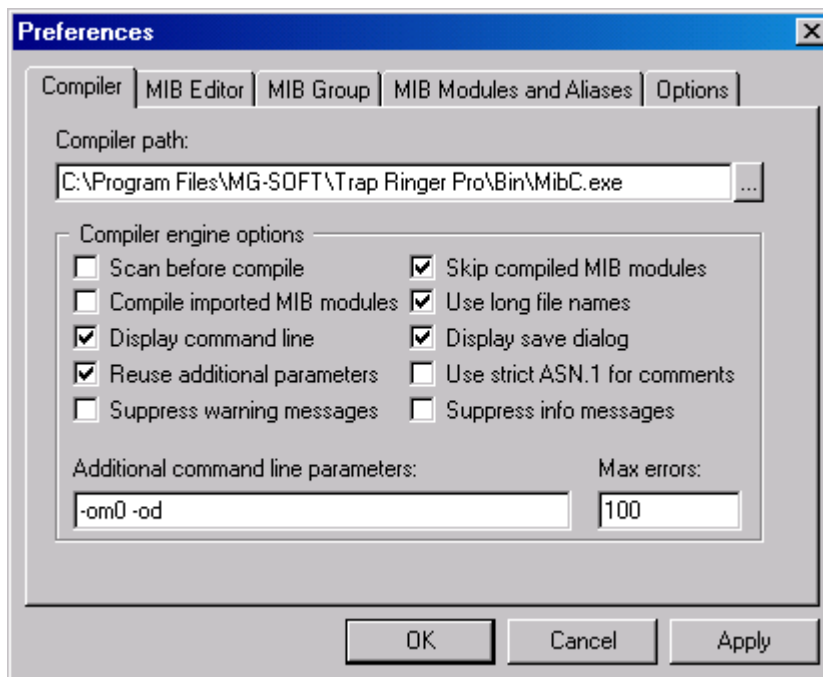


Figure 20: Setting MIB Compiler preferences

**Tip:** For a description of the Preferences dialog box options not covered in this section, please refer to MIB Compiler help documentation.

The following options can be set (separately or in combination) to speed up the compilation process:

### Deactivating Scan Before Compile Option

Unchecking the **Scan before compile** checkbox will speed up the compilation process if not using the Batch Compile command (which does not include the Scan operation).

However, this should be done only if all MIB module source files to be imported are properly registered, otherwise MIB compilation can fail.

**Tip:** Use this option if you have already executed the **Scan** command on the selected module(s) without any errors.

### Deactivating Compile Imported MIB Modules Option

Uncheck the **Compile imported MIB modules** checkbox if you do not need or want to compile all MIB modules which are to be imported (listed in the `IMPORTS` clauses). This can significantly speed up the compilation.

---

## Activating Skip Compiled Option

---

Checking the **Skip compiled MIB modules** checkbox can also accelerate the compilation process. If this option is activated, MIB Compiler will not compile those selected MIB modules, which are already compiled and whose source files have not been changed since the last compilation. In other words, MIB modules, which have older source than compiled files will not be compiled.

---

## Specifying Additional Command Line Parameters

---

Many advanced compilation options can be set via the **Additional command line parameters** input line. You can enter any number of additional switches and parameters, prefixed by “-” and separated by space (Figure 20). The content of this input line is added to the command that invokes MIB Compiler engine.

For information about the available command line switches and parameters, refer to MIB Compiler help documentation (topic: **Command Line MIB Compiler / Command Line Switches and Parameters**).

**Tip:** To view the actual command line used to invoke MIB Compiler engine, check the Display command line checkbox in the Preferences dialog box.

---

**Note:** The Save parameters that are automatically passed to the MIB Compiler engine cannot be overridden in MIB Compiler GUI.

---

**Tip:** To store additional parameters to the system registry, check the **Reuse additional parameters** checkbox. This way, specified parameters apply permanently for every compilation (i.e., also after restarting MIB Compiler).

---

## 7 (RE)REGISTER COMPILED MIB FILES

In the following cases it is necessary to manually register or reregister compiled MIB (.smidb) files, to be able to open and organize them in the MIB Group windows or to load and utilize them in any other WinMIB-based application:

- ❑ If you obtain .smidb files from other parties (e.g., copy additional .smidb files to your computer).
- ❑ If you move your existing (registered) .smidb files to other folder or drive.

To register or reregister compiled MIB files, follow these steps:

1. Select the **Tools / Scan For Database Files** command.

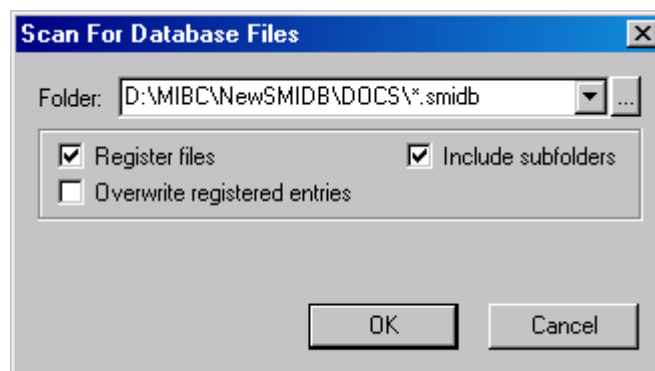


Figure 21: Scan For Database Files dialog box

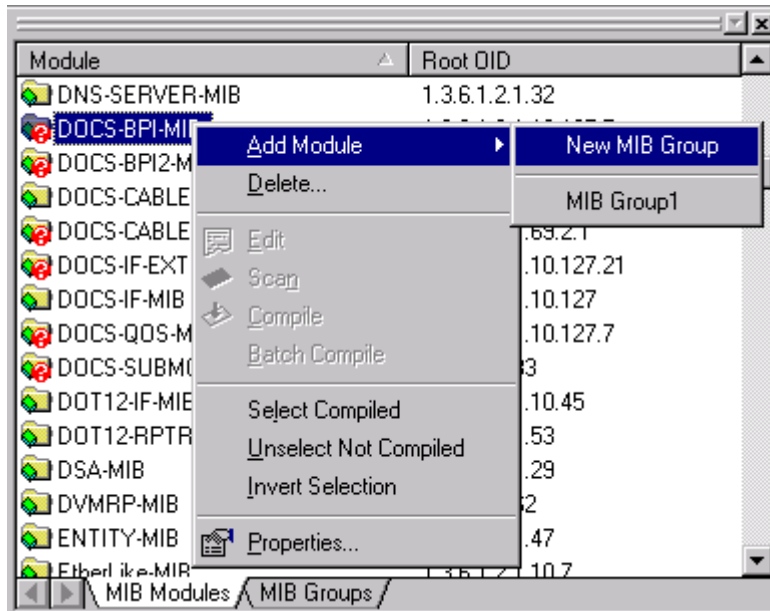
2. The Scan For Database Files dialog box appears (Figure 21), where you can specify the following:

- ❑ Into the **Folder** drop-down list specify the full path of the folder containing compiled MIB files. You can change the default file mask (\*.smidb) to narrow down the Scan operation to files with specific file names and extensions (e.g., \*-???.smidb).
- ❑ To register found compiled MIB files, check the **Register files** checkbox.
- ❑ To search for compiled MIB files in all subfolders of the specified folder, check the **Include subfolders** checkbox.
- ❑ If (some of the) found compiled MIB files are already registered and you want to reregister them, check the **Overwrite registered entries** checkbox.

**Tip:** Use the “...” button next to the **Folder** drop-down list to browse to the desired folder.

3. After specifying above details, select the **OK** button.
4. MIB Compiler scans and registers found compiled MIB files. This operation writes the names and file paths of the found .smidb files to the corresponding application data files, thus making them “known” to MIB Compiler and to other WinMIB-based applications.

After successfully registering compiled MIB files, they are listed in the Modules window. If the corresponding MIB definition files are not registered, such compiled MIB modules are displayed with icons with question mark (Figure 22), indicating that no corresponding MIB definition files are available.



**Note:** MIB modules represented by icons with question mark (?), cannot be edited, scanned or compiled.

Figure 22: Newly registered MIB modules in the Modules window

## 8 ORGANIZE COMPILED MIB MODULES IN MIB GROUPS

MIB compiler lets you organize MIB modules in MIB groups, enabling more efficient MIB database management in applications supporting this feature.

MIB Group windows display a graphical representation of MIB module definitions (MIB tree) and show properties of MIB module and MIB nodes. You can build a MIB tree consisting of nodes defined in different MIB modules by opening these modules in a MIB Group window.

MIB modules listed in a MIB Group window can be saved to a MIB group, representing a logical MIB unit (e.g., you can group all MIBs supported by the RouterX to “RouterX” MIB group etc.). WinMIB based applications utilizing this feature (e.g., MG-SOFT MIB Browser) can load/unload all modules of a MIB group at once. This simplifies MIB module management and makes it more effective.

### 8.1 Create New MIB Group

1. To open one or more compiled MIB modules in a new MIB Group window, select them in the Modules window and use the **Modules / Add Module / New MIB Group** command.

**Tip:** You can also use the corresponding pop-up command.

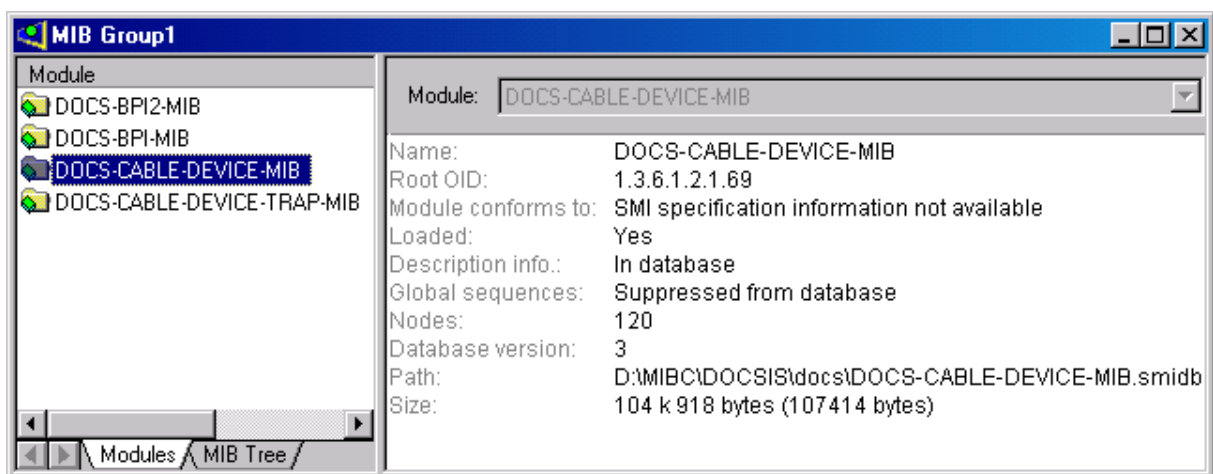


Figure 23: The Modules tab of the MIB Group window

2. The Modules tab of the MIB Group window displays a list of all added MIB modules, while the right panel displays general information about the selected module (Figure 23).

**Tip:** To view MIB tree consisting of nodes defined in MIB modules listed on the Modules tab, switch to the MIB Tree tab of the MIB Group window. Select a MIB tree node, SNMPv1 Trap, Textual Convention or Type Assignment in the left panel to view its properties in the right panel (Figure 24).

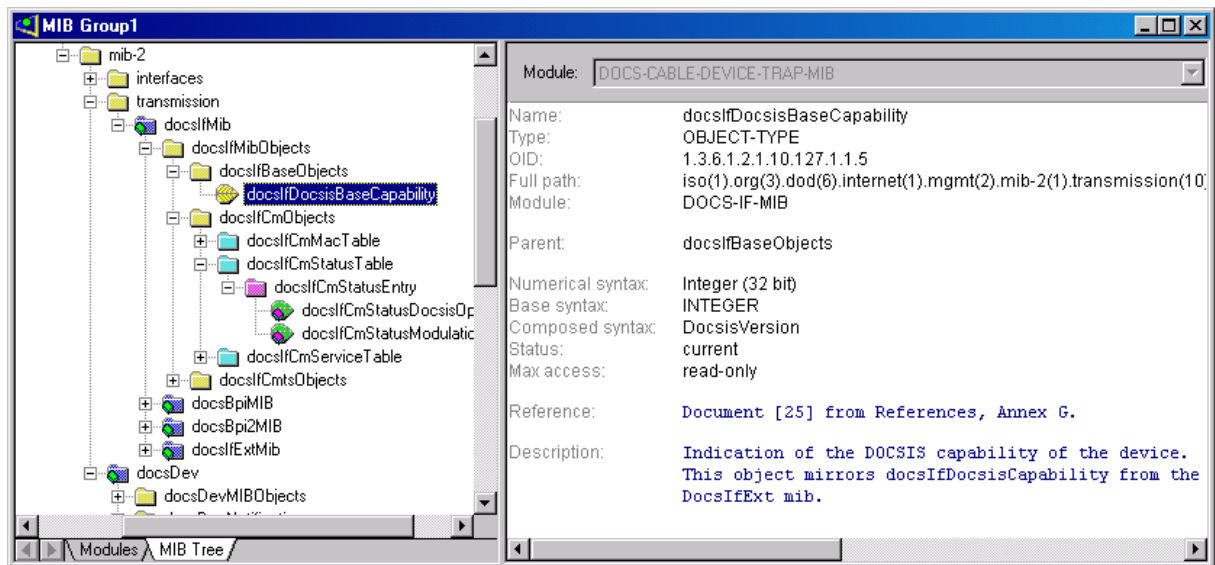


Figure 24: MIB Tree and MIB Node Properties panels

**Tip 1:** In the MIB Tree panel you can use the **Find** and **Find Next** commands to locate desired MIB objects in the MIB tree.

**Tip 2:** To view its SMI definition, select a MIB object (node) in the MIB tree and use the **Go To Source** pop-up command. MIB Compiler will open the corresponding MIB definition file (if available) and jump to the line defining selected MIB object.

**Tip 3:** You can also print parts of or the entire MIB tree together with various node attributes. Please refer to the MIB Compiler help documentation for more information about these features.

3. To add additional MIB modules to an opened MIB Group window, select them in the Modules window and choose the **Add Module / MIB GroupX** pop-up command (where X is the number of the target MIB Group window).
4. To remove a MIB module from the MIB Group window, select it and use the **Remove Module** pop-up command.
5. To save MIB modules opened in the MIB Group window, select them from the list and choose the **Store Selected To Group** pop-up command. To save all listed MIB modules to the same group, use the **Store All To Group** pop-up command.
6. The Group Name dialog box appears. Enter a name for this MIB group into the **Group** drop-down list and select the **OK** button.

**Tip:** The easiest way to add a module to a MIB Group window is to drag-and-drop it from the Modules window to the opened MIB Group window.

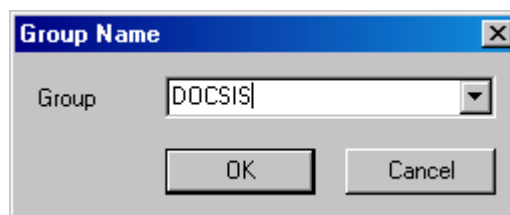


Figure 25: Specifying a desired name for the MIB Group

7. MIB Compiler writes information about the new MIB group to the application data files. Thereafter, this MIB group can be utilized by other WinMIB-based applications supporting this feature. You can view and change some properties of this and other existing groups in the MIB Groups tab of the Modules window (Figure 26).

## 8.2 Manage Existing MIB Groups

To view all defined MIB groups, click the MIB Groups tab in the Modules window.

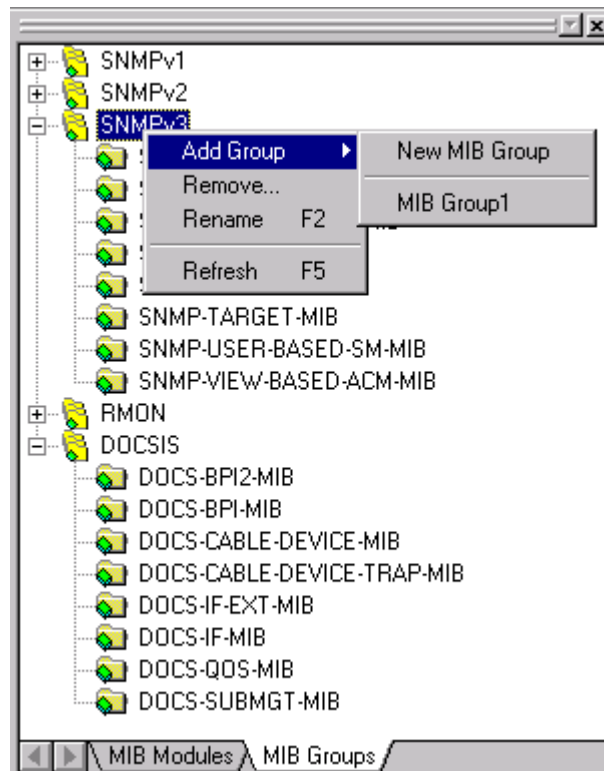


Figure 26: MIB Groups tab of the Modules window

You have the following options of managing existing MIB groups:

- ❑ Rename a MIB group (using the **Rename** pop-up command and specifying a new name for the MIB group)
- ❑ Delete a MIB group (using the **Remove** pop-up command and confirming the deletion)
- ❑ Open a MIB group in a new MIB Group window (using the **Add Group / New MIB Group** pop-up command)
- ❑ Add a MIB group to an opened MIB Group window (using the **Add Group / MIB GroupX** pop-up command)
- ❑ Refresh, the display of MIB groups (using the **Refresh** pop-up command or the **F5** key)

To add one or more modules to an existing MIB group, or to remove modules from the group, do the following:

1. Select a MIB group and open it in a new MIB Group window using the **Add Group / New MIB Group** pop-up command.
2. To add a module to this MIB group, drag-and-drop the module from the MIB Modules tab of the Modules window to the opened MIB Group window. To remove one or more modules from the group, select them in the opened MIB Group window and use the **Remove Module** pop-up command.

**Tip:** You can also remove a module from a group directly in the MIB Groups tab of the Modules window. To do this, select a module in this window and choose the **Remove** pop-up command

3. Save the changed MIB group by using the **Store All To Group** pop-up command in the MIB Group window and selecting the group's existing name from the **Group** drop-down list (Figure 25). Select **OK** button to confirm the overwriting.

## 9 DELETE MIB MODULES

MIB Compiler lets you physically delete registered MIB files from disk and/or delete the MIB module information from the relevant application data files (registry). To do any of these, follow these guidelines:

1. Select the MIB module(s) in the Modules window and use the **Modules / Delete** command.
2. The Delete MIB Module dialog box appears, where you can specify what will be deleted:

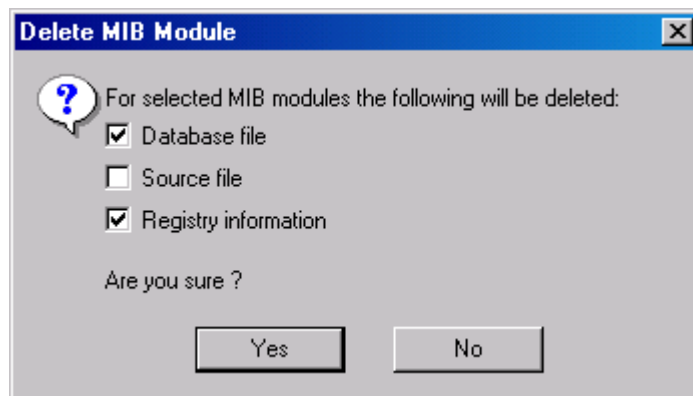


Figure 27: Delete MIB Module dialog box

3. To delete compiled (SMIDB) MIB files of selected MIB modules from the disk, select the **Database file** checkbox.
4. To delete MIB definition (source) files of selected MIB modules from the disk, select the **Source file** checkbox (not recommended).
5. To delete all information about selected MIB modules from the application data files, check the **Registry information** checkbox. This removes all information about MIB modules from the registry (i.e., application data files); therefore, no other application will be able to find and use them thereafter. If you choose to delete both files, the corresponding registration information will be deleted as well.
6. Select the **Yes** button to apply above settings, i.e., to delete MIB files and/or to delete the MIB module registration information from the application data files.

**Note:** If you choose to delete any of the MIB files, they will be permanently erased from the hard drive. It is recommended that you leave at least the **Source file** checkbox unchecked.

**Note:** If you select to delete the registry information for selected MIB modules, these modules disappear from the Modules window when you click the **Yes** button.

---

## 10 GLOSSARY

---

### Agent snapshot

A snapshot of SNMP agent's MIB (Management Information Base) represented in a tree structure. The snapshot displays all implemented object instances, their values and syntax.

### Authentication protocol

In the context of SNMPv3 User-based Security Model, authentication protocol provides a mechanism for ensuring SNMPv3 message integrity and origin. In other words, it provides a mechanism by which it can be reliably determined that the SNMP message has not been modified during its transmission through the network, that the message originates from the claimed party.

### Bookmark

A marker that identifies and saves the reference to the selected document or place in the document.

### Checkbox

A GUI control used to enable or disable an option in a program. An option is enabled when an X or a tick appears in the checkbox (square box), and disabled when the checkbox is empty. Click inside a checkbox to enable or disable the option.

### Client

The Net Inspector Client application connects to the Engine and provides the user interface. Several Clients can simultaneously connect to the same Engine. In this way, one or more network operators can manage network remotely.

### Clipboard

The operating system temporary storage area that is used to hold cut or copied information. The clipboard stores the information until it is pasted or replaced by another object that has been cut or copied.

### Collapse

A command that compresses hierarchical tree structures, like the MIB tree. When collapsed, only the roots of each branch are visible. The opposite of the collapse is the expand command.

### Columnar object

A MIB object that can have zero, one, or more instances and is a part of an SNMP table. In the table each row has one instance of the columnar object.

### Command

A control that, when selected or clicked, initiates an action in a program. Commands can be accessed either from a menu, by using keyboard shortcuts or by clicking toolbar buttons.

### Community

SNMPv1 and SNMPv2c protocols employ the community-based security, where community names included in SNMP messages serve as the only means of authentication. An SNMP community is specified by a string of octets called the community name or string. Typically, different community name is used for retrieval and modification SNMP operations (regarded to as the read and write community names).

### CSV format

CSV stands for Comma-Separated Values format that is a generic file format used for importing data into databases or spreadsheets (e.g., into Microsoft Access or Excel or other database systems).

### Default settings

Preset options built into the program.

### Desktop

The on-screen working area on which windows, icons, menus, and dialog boxes appear.

### Dialog box

A GUI component similar to window that is displayed when additional information is needed to perform an action or command. For example, when you choose the Open command to open a file, the Open dialog box appears, prompting you to specify the desired file.

### Diffie-Hellman key agreement

Diffie-Hellman (DH) key ignition and key change is a key agreement algorithm used by two parties to agree on a shared secret. It is specified as a requirement for cable modems and cable modem termination systems in the Data Over Cable Service Interface Specification (DOCSIS).

### DOCSIS

DOCSIS (Data Over Cable Service Interface Specification) defines interface requirements for cable modems involved in high-speed data distribution over cable television system networks.

### Drag-and-drop technique

To select an object and move it with the mouse while holding down the left mouse button (according to Windows default settings). Releasing the button completes the action.

### Drop-down list

A control that allows you to select from a list of built-in options. If the list cannot accommodate all available options, scroll bars are provided.

**Encrypt**

To convert plain text or data into unintelligible form by use of a code in such manner that conversion to the original form is possible.

**Engine**

The Net Inspector Engine (server) is the program running on the host (server) computer. The Engine itself does not have the user interface, it runs only as a background service application. Monitored devices and services are always contacted and managed through the Engine. The Engine takes care of all management operations as well sending notifications to network operators.

**Event**

Any significant occurrence in the managed device or service that requires network operators to be notified, or an entry to be added to a log. Events contain information about: time stamp, agent address, manager address and community.

**Expand**

A command that extends hierarchical tree structures. When expanding a root node, the entire sub tree becomes visible.

**Filter**

A pattern through which data is passed. Only the data that match the pattern is allowed to pass through the filter.

**Folder**

A named section of computer disk space used to store and organize your documents, programs, and other files.

**Graphic-object**

Graphic-objects are shapes and figures used in the MG-SOFT Net Inspector program. With graphic-objects the user can enhance a visual representation of the managed network.

**Grid**

A series of evenly spaced horizontal and vertical dots that are used for drawing and arranging objects. For greater accuracy, you can also have objects snap to the grid.

**GUI**

A Graphical User Interface to a computer. Today's major operating systems provide a GUI. Applications typically use the components of the GUI that come with the operating system and add their own GUI components. Components of a GUI include desktops, windows, dialog boxes, drop-down menus, buttons, scroll bars, icons, wizards, pointers, etc.

**HTML**

Stands for Hyper Text Markup Language. This is a scripting language used for creating hypertext documents.

**Icon**

A picture representing a tool, service, object, file, or other program items. An item is selected by clicking, or double-clicking its icon.

**Input line**

A line that allows you to type in a text.

**IPv6 address**

IPv6 stands for Internet Protocol Version 6. IPv6 has been designed to replace the current Internet Protocol Version 4, which has been so far most widely used for Internet communication. IPv6 fixes the limited number of available IPv4 addresses and adds many other improvements. An example of an IPv6 address (in shorter form with omitted zeros): fe80::2c1:27ff:fe00:2d9.

**Lexicographical order**

A hierarchical organization of OID values used for SNMP network management by which all object instances are uniquely identified with the object name and the corresponding suffix.

**Management information**

The information that is exchanged between an SNMP manager and SNMP agents and allows the monitoring and control of a managed device.

**Map**

Maps are used in the Net Inspector program to enable a hierarchical presentation of the managed system. A map can contain map-objects (icons of managed devices or managed services), graphic-objects (for example, graphical links between icons) and maps (sub maps). Maps can reflect the status of one or more map-objects according to the selected option, and propagate the status in the hierarchy upwards.

**Map-object**

Map-objects are icons used in the MG-SOFT Net Inspector program. Each icon added to a map represents a managed device or service.

**Menu**

A list of available commands or sub-menus that appear when you click a menu name in the Menu Bar.

**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.

**MIB**

A MIB (Management Information Base) contains a definition of management information that can be collected and controlled by the management application using the SNMP or RMON. Related management information is defined in MIB modules and stored in MIB files. A MIB file is usually supplied by the vendor of an SNMP manageable device and contains a description of the object hierarchy and

object attributes on the managed device and serves as a roadmap for managing that device.

### **MIB compilation**

An operation performed by MIB Compiler program that converts (compiles) MIB files from one format into another. MG-SOFT MIB Compiler compiles MIB files written in the SMI language into the SMIDB binary file format.

### **MIB expression**

MIB expressions are combinations of at least one operand and one or more operators. Operands are OID values, whereas operators are symbols that represent particular operations. In the MIB expression "{1.3.6.1.2.1.2.2.1.16} \* 8", the "1.3.6.1.2.1.2.2.1.16" and "8" are operands, and \* is an operator. In Query Manager, MIB expressions calculate values that cannot be queried by using an OID, because no such OID is implemented in the monitored SNMP agent.

### **Net Inspector session**

The Net Inspector Client Session is a lasting connection between a Net Inspector Client and Engine, usually involving the exchange of the management information between the Client and the Engine. The Layout of all desktop components is saved in the Session file. As you close the Session, Net Inspector saves all window positions, sizes, and other settings. When you re-open the Session at any later time, these properties will be restored.

### **Network operator**

Network operator is a person who runs network management applications and performs operations on the computer to monitor and control a device or service.

### **Object instance**

A MIB object instance, also called a MIB variable or SNMP variable, is a specific instance of an object that presents a class of management information. Each instance is identified by the identity of its class (OID value or name of the object) and by its identification within the class (suffix to the object OID value or name). SNMP operations can retrieve or modify values of object instances.

### **OID value**

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 MIB objects in MIB object instances. For example, OID value assigned to the sysUpTime MIB object is 1.3.6.1.2.1.1.3.

### **Path name**

Location of a folder or file on your system.

### **PDU**

See SNMP PDU

### **PFS module**

A proxy front-end server (PFS) module is a Net Inspector's plug-in module for a management application. Every PFS module contains one or more PFS components. Each PFS component specifies the type of the managed device or service and the type of the communication protocol to manage that device or service.

### **Pixels**

Short for picture element. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels that have a specific color.

### **Polling**

An operation that periodically queries a device for data.

### **Polling unit**

Polling unit is a basic unit of polling in the MG-SOFT Query Manager and Net Inspector programs. A polling unit can contain a single OID, a group of OIDs, or a MIB expression.

### **Pop-up menu**

Use the right mouse button (according to Windows default settings) to click inside a window, an object or icon. The pop-up menu that appears shows the most frequently used commands for that window, object or icon.

### **Privacy protocol**

In the context of SNMPv3 User-based Security Model, privacy protocol provides a mechanism that prevents transmitted SNMPv3 messages from being disclosed to unauthorized individuals, entities, or processes. This is achieved by message encryption.

### **Query**

A user's request for information from an SNMP agent or generally from any database.

### **Radio button**

A GUI control used to enable or disable an option. An option is enabled when a dot appears in the radio button circle, and disabled when the radio button is empty. Click inside a radio button to enable or disable the option. When you select one button all others are automatically deselected.

### **RMON**

RMON (Remote Monitoring) defines a set of SNMP MIB objects that instrument data collected on a network device.

**Scalar object**

A MIB object that has exactly one instance. The instance of a scalar object is always identified by the suffix 0. An example of a scalar object is a SysUpTime object, and its only instance is sysUpTime.0.

**Scope ID**

A scope ID identifies the network interface over which traffic is sent and received. The notation that is used to specify the scope ID with an address is Address%ScopeID.

Example:

```
fe80::280:48ff:fed7:59be%5
```

**Sibling node**

One of two or more nodes in a hierarchy that have the same parent node.

**SMI**

SMI stands for Structure of Management Information that defines a formal language for describing management information so that the information is retrievable and modifiable.

**SMIDB file**

SMIDB stands for Simple Management Information Database. SMIDB file is a binary file containing a compiled MIB module. The file is in MG-SOFT's proprietary format. It can be accessed from any application by utilizing MG-SOFT's WinMIB API.

**Snapshot**

See Agent Snapshot

**Snap to grid**

To force a map-object to align automatically to a point on the grid.

**SNMP**

SNMP stands for Simple Network Management Protocol that defines methods of communicating between a manager and an agent. Currently there are three major SNMP protocol versions: SNMPv1, SNMPv2c and SNMPv3.

SNMPv1 protocol defines basic SNMP operations, which are: SNMP Get, SNMP GetNext, SNMP Set and SNMP Trap.

SNMPv2c protocol introduces SNMP GetBulk and SNMP Inform operations that enhance performance and reliability.

SNMPv3 protocol introduces security enhancements in the area of the authentication and privacy.

**SNMP agent**

SNMP Agent is a piece of software implemented in a managed device. SNMP agent provides information about the managed device and allows the device to be monitored and/or controlled by a network management application via SNMP.

**SNMP Get**

An SNMP operation that retrieves the value of the object instance from an SNMP agent. It is used when the manager exactly knows the instance of the desired management information.

**SNMP GetBulk**

An optimization of the GetNext operation available only in the SNMPv2c and SNMPv3 protocols. It is used to minimize network interactions by allowing the agent to return large packets of information containing multiple instances of objects instead of just one, as in case of the GetNext operation.

**SNMP GetNext**

An SNMP operation that retrieves the value of the object instance that in lexicographical order follows the OID used in request.

**SNMP manager**

A SNMP manager is a software through which the network operator performs network management functions: queries agents, gets responses from agents, sets variables in agents and acknowledges events.

**SNMP message**

The SNMP (Simple Network Management Protocol) sends operation requests and responses in form of SNMP messages. An SNMP message consists of a SNMP protocol data unit (PDU) plus additional message header elements defined by the relevant specification.

**SNMP PDU**

SNMP PDU is a Protocol Data Unit that is encapsulated in every SNMP message and enables exchange of management information between the SNMP manager and the agent. Each PDU contains a variable bindings list and information about the PDU type. There are different types of SNMP PDUs, such as GetRequest-PDUs, GetNextRequest-PDU, GetBulkRequest-PDU, SetRequest-PDU, Response-PDUs, Trap-PDU, SNMPv2-Trap-PDU, Inform-Request-PDU, and Report-PDU.

**SNMP Set**

An SNMP operation used for controlling and configuring managed devices by modifying values of object instances in SNMP agents.

**SNMP Trap and Inform notifications**

SNMP Trap and Inform notifications are unsolicited reports generated by the SNMP agents. When an event is detected in the managed system, the SNMP agent sends either a Trap or an Inform notification to the SNMP manager. If a Trap notification is sent to a SNMP manager, a SNMP agent cannot determine whether the message was received or not. On the contrary, a SNMP manager that receives an Inform notification has to acknowledge the notification. Trap notifications are available in the SNMPv1, SNMPv2c and SNMPv3 protocols, while the Inform notifications

are available only in the SNMPv2c and SNMPv3 protocols.

#### SNMP variable

SNMP variable, also called a MIB variable, is a specific instance of an object that presents a class of management information. SNMP operations retrieve or modify values of variables.

#### SNMP Walk

An SNMP operation that is used to discover all object instances of management information implemented in the SNMP agent that can be accessed by the SNMP manager. The object instances are retrieved repeatedly querying an agent using the SNMP GetNext or SNMP GetBulk requests. Each SNMP GetNext request uses the OID returned in the response to the previous GetNext request.

#### Status bar

The Status bar is the on-line screen display area at the bottom of the main window that shows the status information about the ongoing operation and description of tasks behind the toolbar button and menu commands when placing the mouse cursor on the toolbar button or the menu command (without clicking).

#### Step-by-Step SNMP Walk operation

A variation of the SNMP Walk operation that unlike the 'one-step' SNMP Walk operation allows controlled 'instance-by-instance' traversing of agent's MIB tree. With the Step-by-Step Walk operation object instances in a MIB tree are traversed manually, while in the SNMP Walk operation the whole group of object instances is traversed automatically without stopping.

#### Task bar

The bar that contains the Start button and appears by default at the bottom of the desktop. You can click the taskbar buttons to switch between running programs. You can also hide the taskbar, move it to the sides or top of the desktop, and customize it in other ways.

#### Timeout

An interrupt signal a program generates after it has waited for a certain length of time to receive a response to a sent request. After the timeout the program either repeats or cancels the query of the agent.

#### Title bar

The bar along the top of the application window that contains the name of the window, the program icon, the Maximize, Minimize and Close button. Dialog boxes in MG-SOFT applications also have Title bars but not Maximize and Minimize buttons.

#### Toolbar

A group of buttons that provide quick access to a series of the most common menu commands.

#### UDP

UDP (User Datagram Protocol) is a connectionless transport-level protocol that runs on top of IP protocol and is the proffered transport protocol for SNMP. UDP is a part of TCP/IP suite of protocols.

#### UNC module

A user notification carrier module (UNC) is a Net Inspector's plug-in module used for forwarding the management information to network operators via e-mail and/or SMS. The network operators can choose which information they want to receive by applying appropriate filters for the UNC components.

#### USM

Stands for the SNMPv3 User-based Security Model, which defines the elements of procedure for providing SNMP message level security. This security model is based on traditional concept of a user (identified by a username) with which security information is associated.

To ensure security the User-based Security Model describes the use of HMAC-MD5-96 and HMAC-SHA-96 as the authentication protocols and the use of CBC-DES as the privacy protocol. For more information about USM, see RFC 3414.

#### Variable binding

A pairing of an object instance name and its corresponding value and syntax. A variable binding list is a series of variable binding entries included in a message protocol data unit (PDU).

#### Workspace

The Net Inspector workspace is maintained at the Engine side. It represents the environment for managing the system and contains all the information that the Engine receives from the managed devices.

#### XML

Stands for Extensible Markup Language. It is a flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, etc.

## 11 INDEX

### A

about this manual ..... 7  
alias ..... 23, 31

### B

Batch Compile dialog box ..... 21  
batch compiling ..... *See* compiling

### C

Compiled MIB Modules dialog box ..... 24  
compiling  
  all MIB definition files ..... 22  
  error ..... *See* resolving compilation problems  
  group of MIB definition files ..... 17  
    using *Compile* command ..... 19  
    using *Modules / Batch Compile* command .... 20  
    using *Tools / Batch Compile* command ..... 21  
  performance ..... *See* improving compilation performance  
  single MIB definition file ..... 15–17

### D

debugging ..... *See* resolving compilation problems  
Delete MIB Module dialog box ..... 41  
deleting  
  compiled MIB files ..... 41  
  MIB definition files ..... 41  
  registry information ..... 41  
docking windows ..... 13

### E

Edit Module Properties dialog box ..... 23, 32  
Error message ..... *See* resolving compilation problems

### G

Group Name dialog box ..... 38

### I

improving compilation performance ..... 31  
  adjusting preferences  
    Additional Command Line Parameters ..... 34  
    *Compile Imported MIB Modules* checkbox ..... 33  
    defining aliases ..... 31  
    *Scan before compile* checkbox ..... 33  
    *Skip compiled MIB Modules* checkbox ..... 34  
  registering all MIB modules ..... 31  
Info message ..... *See* resolving compilation problems

### M

MG-SOFT Corporation  
  about ..... 5  
  contacting information ..... 5  
MIB (Management Information Base) ..... 43  
MIB Compiler  
  about (product description) ..... 6  
  desktop  
    customizing ..... 12  
    description ..... 11  
  Help documentation ..... 11, 28, 34  
  log (Output window) ..... 26–28  
  starting the software  
    on Linux operating system ..... 9  
    on Windows operating system ..... 8  
MIB Editor window ..... 27, 29  
MIB files/modules  
  compile ..... *See* compiling  
  compiled MIB file ..... 15  
  delete ..... *See* deleting  
  edit MIB definition files ..... 28–30  
  MIB database file ..... *See* compiled MIB file  
  MIB definition file ..... 6, 15  
  MIB module ..... 15  
  MIB source file ..... *See* MIB definition file  
  register ..... *See* scanning and registering  
  save ..... *See* saving  
  scan ..... *See* scanning and registering  
  SMIDB ..... *See* compiled MIB file  
MIB group  
  about ..... 37  
  add MIB module to ..... 40  
  create ..... 37

open .....	39
remove .....	39
remove MIB module from .....	40
rename .....	39
save .....	38
MIB Group window	
MIB Tree tab .....	38
Modules tab .....	37
Modules window	
MIB Groups tab .....	39
MIB Modules tab .....	36

**O**

OID (object identifier) value .....	44
Open dialog box .....	16
Output (MIB Compiler Log) window .....	18, 26

**P**

Preferences dialog box .....	33
------------------------------	----

**R**

register MIB files .....	<i>See scanning and registering</i>
resolving compilation problems	
editing MIB Definition files .....	28–30
examining log messages .....	26–28
error messages .....	26, 27, 28
info messages .....	26

warning messages .....	26, 27, 28
------------------------	------------

**S**

Save As dialog box .....	25
saving	
compiled MIB files .....	24–25
edited MIB definition file .....	29
MIB group .....	38, 40
Scan For Database Files dialog box .....	35
Scan For Source Files dialog box .....	18
scanning and registering	
compiled MIB files .....	35
MIB definition files .....	17, 31
SMI	
code editing .....	28–30
language .....	7, 15
syntax checking .....	<i>See compiling</i>
syntax coloring .....	29
SMIDB .....	15, 24
sub-identifiers .....	44

**T**

Tip of the Day dialog box .....	9
---------------------------------	---

**W**

Warning message .....	<i>See resolving compilation problems</i>
-----------------------	---