MIB Explorer 5.0

The user-friendly SNMP MIB Browser for Java SE

Table Of Contents

1 MIB Explorer Manual Overview

The MIB Explorer manual is organized into the following main topics. If you are new to MIB Explorer and SNMP/SMI, start with the following sections:

- **System Requirements**
- **Setup**
- **MIBs**
- **MIB Sets**
- **Targets**

As SNMP versed user you may prefer the following section order:

- **Setup**
- **Preferences**
- MIB Repository
- Compiling MIBs
- Loading MIB Modules
- **MIB Sets**
- **Targets**

Any MIB Explorer user should read at least how the MIB Tree view and the Table view panel are used:

- **MIB Tree Panel**
- **Table View**

Supplementary but yet very useful features are described in:

- Trap Receiver
- Protocol Data Units (PDUs)
- Packet Analyzer
- Scripts
- TFTP
- Snapshots
- Discovery of Network Elements
- SNMPv3 User Administration

2 System Requirements

The system requirements for MIB Explorer are:

- Java SE Runtime Environment (JRE) version 9 or later.
- UDP transport and/or TCP transport
- 256 MB RAM
- You can use the -Xmx option of the JRE the increase or decrease the maximum memory used by MIB Explorer. For example
  `java -Xmx512M -jar mxp.jar`
  will allow MIB Explorer to use up to 512MB of RAM.
- ~60 MB free hard disk space (JAR) distribution, with installer package that includes a Java Runtime Environment (JRE), space requirements are around 200MB.

3 Setup

Please read the section "System Requirements" on page 3 for prerequisites needed to install and run MIB Explorer. Then follow the below steps to install and setup MIB Explorer below.

3.1 Installation

MIB Explorer can be installed as *native* application or standard Java application.
The latter requires a Java Runtime already installed on the target operating system whereas the native application installs a stripped OpenJDK Java Runtime bundled with MIB Explorer.
The native installation provides operating system integration, such as icons on the desktop and start menu integration.

To install MIB Explorer for an already installed Java Runtime (JRE), first make sure you have installed a JRE version **9** or later (see "System Requirements" on page 3). Then follow the steps set forth in section "Any Platforms - Plain Java Installation" on page 4.

3.1.1 Native Application Installation

1. Log on to your system as the user who is supposed to use MIB Explorer.
2. Download one of the native application packages of MIB Explorer compatible with your operating system from *https://agentpp.com/download.html*.
3. Run the installer and start the application.
4. Follow the instructions to setup MIB Explorer in section "Setup" on page 5.

3.1.2 Any Platforms - Plain Java Installation

Download the `mxp-<version>.jar` file in a folder of your choice. Start the MIB Explorer application by

- double clicking it from your system's file explorer, or
- running the following command from the command line:

```
java -jar mxp-<version>.jar
```

Java Runtime Environment (JRE) Installation

1. Download the latest JRE from *https://java.com.*
2. Install the Java Runtime Environment (JRE) 9 or later on your system and add the `bin` directory of the JRE (or SDK) installation to your `PATH` environment variable (this is often already done by the installation process).

3.2 Starting MIB Explorer

If you have used a native installer to install MIB Explorer, then you can start it from your systems application start menu or by clicking its icon on your desktop.
Otherwise double click the downloaded `mxp-<version>.jar` file or run

```
java -jar mxp-<version>.jar
```

from the command line (see previous section). When MIB Explorer is started for the first time, you will be prompted for your license information.
If you are using a restricted or evaluation license you can upgrade it later without reinstalling MIB Explorer by choosing Help>License… from the main menu.

Please enter your license key including blanks! The license key is case sensitive.

3.3 Setup

Once you have started MIB Explorer and entered your license information, choose File>Install... to install MIB Explorer MIB files and repository as well as other accompanied files on your system.
At first application start, you will be automatically asked to specify an empty installation directory for MIB Explorer accompanied files.
Every time MIB Explorer is updated and the structure or version of the accompanied file set has changed, you will be asked to install/update those files again. You have then the choice to install the files to a new location or update the files in the existing location.

Note: When updating accompanied files, modifications on that files may get lost. Thus, it is recommended to use a new installation directory and delete the old manually if you have changed any of those files. You can then delete the old directory, if it is not needed any more.

3.3.1 Install Templates, Examples, and MIBs

Once you have started MIB Explorer and entered your license information, choose File>Install... to install MIB Explorer MIB files, repository of precompiled MIB files, example scripts and monitors as well as other accompanied files on your system.

3.4 Updates and Upgrade

MIB Explorer checks on startup if there is a new version available if it can reach the update server through the Internet.
You can run that check manually from the Help>Check for Updates menu.
If a newer version is available, MIB Explorer prompts you to start the Update process.
As there are two different installation types (native and plain java), there are two different upgrade/update methods:

By default MIB Explorer uses the system proxy (or no proxy if there is not any system proxy defined). You can specify your proxy as described in section "Preferences" on page 8.

1. In-place JAR update
2. Native OS Installer

3.4.1 In-place JAR Update

For plain Java installations, you can update your MIB Explorer application by replacing the mxp-<version>.jar at the location it is installed. This in-place update will

1. download the new version,
2. extract a small Java application within the same directory that is going to do the actual update,
3. start the Java update application and exits MIB Explorer
4. move the existing JAR to a backup file and then move the downloaded new version to the existing JAR filename.
5. then restart MIB Explorer using the new JAR.
6. verify the MIB Explorer setup and will remove any backup files during restart - if still present.

3.4.2 Native OS Installer Update

If the MIB Explorer update function detects, that it is currently running from a native OS installer built setup, then it will offer to use that update method. It will run the update as follows:

1. Download the new OS installer for the existing application.
2. Run the OS installer.
3. Exit MIB Explorer.

Please follow the steps of the installer to either install updated MIB Explorer in a new location or overwrite the existing installation. It depends on your operating system which options you actually have.

3.4.3 Manual Update

In any case, you can switch between the installation and update methods by doing a manual update. Simply follow the steps in “Setup” on page 5. MIB Explorer will find the configuration file in your home directory and reuse existing settings and accompanied files.

3.4.4 Updating Accompanied Files

If a newer version of the accompanied file set is available with the new version, MIB Explorer will ask you to install them in the current installation location. If you confirm the installation, MIB Explorer will overwrite existing files with the newer version.
If the newer version has a higher major release number, then it is recommended - but not required - to install the files into a new directory.

MIB Explorer will contact a service on https://updates.snmp.app to check if new updates are available for the installed version and license. When you confirm the update, the new version will be downloaded from https://agentpp.com from the same location you would use for a manual download.

3.4.5 Upgrade License

To upgrade to a new major release version, purchasing an upgrade license is an alternative to purchasing a regular new license. To upgrade your existing software you can download the new software and start it.
It will then prompt for a license key (because the stored license key is not valid for the new version). Now enter your upgrade or regular license key.
If the configuration file of the old version cannot be found then you will be prompted for entering the license key of the previous version too in order to validate the upgrade license. For a regular license this step is not necessary.

3.5 Uninstall

For an installation with one of the native installer packages, please use the platform specific uninstall mechanisms to remove the software itself. Otherwise it is sufficient to remove the `mxp-<version>.jar` file.
In any of both cases, you may manually remove the accompanied files installed by MIB Explorer during initial startup at the location you had then chosen.
MIB Explorer holds its configuration data in the `mxp4.cf` file in your home directory. To completely uninstall MIB Explorer, this file has to be removed manually. By removing it, you will have to reenter your license information - as well as other configurations - when you reinstall MIB Explorer.

4 Preferences

With the preferences dialog accessible from the tool bar () or with Edit>Preferences from the menu bar, MIB Explorer's settings are configured.
The preferences are divided into the top level areas of configuration that are described by following sub-sections.

4.1 General

The general preferences can be used to configure overall settings of MIB Explorer. These are:

- **Maximum number of undo/redo steps**

  Specifies how many undo steps should be stored by MIB Explorer. This value applies to each table or PDU frame independently.

- **Maximum number of instances retrieved for MIB tree**

  Specifies the maximum number of MIB object instances that will be retrieved for MIB tree and browse operations. Too many objects will unnecessarily consume memory and performance. A good choice is a limit of a few hundred or thousand objects. The limit can be disabled at all, by setting a zero value.

- **Load a target's associated MIB set (if present) when the target is selected**

  If checked, MIB Explorer will automatically load a target's MIB set (if there is any associated MIB Set) when a new target is selected.

- **Enable SET for variables with MAX-ACCESS read-only**

  Enabling this option allows you to use the Set Dialog on OBJECT-TYPES with MAX-ACCESS read-only and write or create "read-only" table columns using the Table View or Grid View. This option might be useful for testing purposes or when configuring an AGENT++ simulation agent, but this option should be disabled by default.

- **Confirm overwriting a file**

  If checked, MIB Explorer will always ask before it overwrites a file (except its configuration file).

- **Support obsolete RFC 1442 BIT STRING syntax**

  Enable this option to communicate with SNMP entities that implement and use the obsolete RFC 1442 BIT STRING syntax.

- **Use always GETNEXT instead of GETBULK**

  By enabling this option, MIB Explorer will never send GETBULK PDUs. Instead GETNEXT PDUs are used in a compatible way. This option is useful in an environment where agents exists, that are capable of SNMPv3 but do not support the GETBULK PDU type.

- **Multi-threaded column retrieval for wide tables**

  When checked and when a table has more columns than fitting in a single request PDU (limited by "PDU Size" on page 10), the additional requests for the off limit columns will be send immediately after the first without waiting for the response of the initial request. This can speed up table retrieval because agents fetch typically fetch (and cache) complete rows. Using multi-threaded column retrieval is the default.

4.2 MIB Compiler

The MIB Compiler preferences define where to store compiled MIB modules and other settings about the integrated MIB compiler.

- **MIB Repository Path**

  The MIB repository directory must be specified before any MIB modules can be compiled (see MIBs). It must exclusively contain compiled MIB modules and the `MODULE.IDS` file which stores module IDs.

- **Maximum Errors per MIB File**

  The maximum parse errors specify the number of errors the MIB compiler should collect before bailing out and reporting the found errors.

- **Record file name of the imported MIB file in the MIB repository**

  Activate this option, if you want to later append the original file name to the file name of exported MIB modules. The MIB compiler will store the file name but not any path information in the compiled MIB module file. See also section "Exporting MIB Modules" on page 29.

- **Compress MIB module files in the MIB repository**

  To save disk space, the compiled MIB modules can be stored in the MIB repository using GZIP compression. Deactivate this option if

you are using AgenPro or MIB Designer with the same repository if either version is less than 2.5.

- **Compile MIBs Leniently**

  If the MIB files you want to use with MIB Explorer contain many errors you may use this option to compile those MIBs anyway with a minimum syntax error checking. Although this might work in most cases, lenient compiled MIB modules might cause problems - therefore they are marked with '(!)' in the module list.

- **When compiling „new“ MIB modules update existing if LAST-UPDATED is newer**

  If disabled, only MIB modules will be compiled and added to the current repository, that do not exists in the repository yet. If enabled, only MIB modules will be compiled as „new“ that do not exists, or have a LAST-UPDATED time stamp which is newer that those of the existing MIB module with the same name. SMIv1 modules will be always overwritten by SMIv2 modules of the same name.

4.3 PDU Size

The PDU Size settings define indirectly the maximum size of request PDUs sent by MIB Explorer. Because of implementation specific restrictions such as message buffer sizes not all SNMP agents are able to process arbitrary sized SNMP messages. Bigger values for the below parameters provide best

performance but result in a higher risk of timeouts, because of non-responding agents.

The maximum VBs per PDU restriction limits the number of variable bindings sent in a PDU on behalf of table retrieval operations. The maximum repetitions for GETBULK operations limits the number of 'rows' in a response PDUs to be sent by the target agent on behalf of table operations.

- **GET BULK repetitions**

  Specifies the maximum number of variable bindings an agent may return on a GETBULK request sent by MIB Explorer during a Browse or Get operation with SNMPv2c or SNMPv3.

- **Maximum VBs per PDU**

  Specifies how many variable bindings should be sent by MIB Explorer in a GETNEXT or GETBULK request to retrieve table data. To optimize table operations, normally all column or scalar OIDs of a table

are put together into one PDU. If the table has more columns or scalars than specified by this value, then MIB Explorer will send the following number of packets to retrieve a single row with SNMPv1:

$$\frac{\text{number of columns}}{\text{maximum VBs per PDU}}$$

and with SNMPv2c or SNMPv3:

$$\frac{\text{number of columns}}{\text{max. VBs per PDU} \bullet \text{GETBULK repetitions}}$$

4.4 Trap Receiver

In order to be able to receive SNMPv1 traps or SNMPv2c notifications and INFORM PDUs, MIB Explorer has to be configured to listen on at least one UDP or TCP port on a specific or all local address(es). The default is UDP port 162 on all local IP addresses ("0.0.0.0"). You can specify listen addresses and ports in the following subtopics:

- UDP Trap Addresses
- TCP Trap Addresses
- TLS Trap Addresses

To receive SNMPv3 notifications and INFORM PDUs, MIB Explorer also needs its own authoritative SNMP engine ID specified and which management targets are allowed to send notifications or INFORMs to MIB Explorer.

Once you are able to receive traps and notifications, you might want to prioritize them according to their trap/notification ID. You can do this using the Trap Priorities subtopic.

The general trap receiver settings specify MIB Explorer's behavior when a new trap or notification is received:

To Add a Trap Listen Port:

1. Choose Add from the Notification Listen Addresses area. A trap listen address configuration dialog will be shown.
2. Choose one of the local system's IP addresses from the drop down list. `0.0.0.0` represents all local addresses.
3. Choose the UDP port to listen on.
4. Press OK to activate the listen address.

To Specify Authoritative Engine ID:

1. Either press the Default button or enter a 5 to 32 bytes long Authoritative Engine ID as a hexadecimal string into the corresponding text field. If the entered engine ID has less than 5 or more than 32 bytes or is otherwise invalid, the default authoritative engine ID will replace the entered value when preferences are saved.

To Specify Principals for Trap Reception:

If you want to enable trap reception for some management targets:

1. Choose one or more targets from the "Available Principals" list for which you want to enable trap reception.
2. Press the Add button.

If you want to disable trap reception for some management targets:

1. Choose one or more targets from the "Enabled Principals" list for which you want to disable trap reception.
2. Press the Remove button.

Actions on New Trap:

- **Beep**

  If checked, an audio beep is emitted whenever a new trap or notification is received.

- **Bring trap receiver to front**

  If checked, the trap receiver frame is brought to front whenever a new trap or notification is received, even if the trap receiver frame has not been visible.

- **Auto-Inhibition**

  The auto-inhibition hides new traps in the trap receiver dialog if more than the specified amount of traps per second is received for within a interval of the specified number of seconds.

- **INFORM Response Delay**

  To be able to test INFORM request command responder implementations, simulating a slow INFORM response sending could be necessary. Specifying a value greater than zero (and less than 300000ms = 5min) activates INFORM response sending for all incoming INFORM requests. Zero or any out-of-range value disables the delay.

If you specify a notification history file by checking the "Use persistent notification history" box, notifications and traps received by MIB Explorer will be stored on (normal) application exit to the specified file. At application restart, this file is read again to fill the trap receiver table with the previously received traps.

4.4.1 Trap Priorities

With the Trap Priorities settings you can specify logging severities for categories of incoming traps, notifications, and inform messages. The severity is determined by analysis of the notification ID. For each incoming trap/notification, the trap severities table will be searched for the entry (category) whose subtree object identifier (OID) is the longest possible match. The severity for this message will then be set to the severity specified for the matched category.

The trap severities configuration table has the following columns:

COLUMN	DESCRIPTION
Subtree	The OID of the subtree for which a trap/notification severity (=priority) is defined. Matching entries with longer (subtree) OIDs override entries with a shorter OID.
Severity	The severity defines the trap/notification priority. The severity FATAL has the highest priority and INFO the lowest.
Script	The path of a MIB Explorer script (see below) that has to be executed when this entry matches an incoming trap or notification.
Comment	An arbitrary comment that can be referenced by the above script.

Table 1: Trap severities configuration table.

If there has been assigned a MIB Explorer Script for the matched category, then the corresponding script will be executed with the `snmp`, `utils`, and `smi` contexts and additionally the following special context values:

CONTEXT	DESCRIPTION
severity	The assigned severity for the received notification as one of the following strings: FATAL, ERROR, WARN, and INFO.
content	The comment string assigned to the category the received notification matches or null if the comment is left empty.
sourceAddress	The complete source address of the notification.
sourceHost	The host (IP address) of the notification source.
sourcePort	The UDP or TCP port of the notification source.

Table 2: Special context values for Trap Recveiver scripts.

To Open the Trap Severities Editor:

1. Open Preferences from the Edit menu.
2. Choose Trap Receiver from the preferences tree.
3. Add or remove categories by either using the Add orRemove buttons respectively or alternatively using the context menu of the shown table.
4. Press OK to save your changes.

To Configure a Script for a Notification Category:

1. Select the category row by clicking on the row's Script column cell.
2. Open the context menu by pressing the right mouse button.
3. Choose Script... and choose or enter the file name of the script to run for notifications of this category.
4. Press OK to save your changes to the category.

There is an example for sending an email when receiving a trap in the `examples/script` directory of the MIB Explorer installation named `email_on_trap.vm`.

4.4.2 UDP Trap Addresses

To receive traps, notifications, and inform requests over UDP, you can specify the local UDP listen address(es) and port(s) here. If the status of a listen port is '*Unavailable*' then the port is used by another application or MIB Explorer has insufficient system rights to bind the port. On UNIX system, for example, super-user rights are needed to bind ports below 1024.

When you add a listen address, MIB Explorer immediately tries to bind the address. The status column of the configuration table indicates then whether binding the address and port was successful (status *Available*) or not (status *Unavailable*). When closing the preferences dialog, MIB Explorer will update its configuration again, depending on whether you saved your changes or not.

4.4.3 TCP Trap Addresses

To receive traps, notifications, and inform requests over TCP, you can specify the local TCP listen address(es) and port(s) here. If the status of a listen port is '*Unavailable*' then the port is used by another application or MIB Explorer has insufficient system rights to bind the port. On UNIX system, for example, super-user rights are needed to bind ports below 1024.

When you add a listen address, MIB Explorer immediately tries to bind the address. The status column of the configuration table indicates then whether binding the address and port was successful (status *Available*) or not (status *Unavailable*). When closing the preferences dialog, MIB Explorer will update its configuration again, depending on whether you saved your changes or not.

4.4.4 (D)TLS Trap Addresses

To receive traps, notifications, and inform requests over TLS or DTLS, you can specify the local (D)TLS listen address(es) and port(s) here. If the status of a listen port is '*Unavailable*' then the port is used by another application or MIB Explorer has insufficient system rights to bind the port. On UNIX system, for example, super-user rights are needed to bind ports below 1024.

When you add a listen address, MIB Explorer immediately tries to bind the address. The status column of the configuration table indicates then whether binding the address and port was successful (status *Available*) or not (status *Unavailable*). When closing the preferences dialog, MIB Explorer will update its configuration again, depending on whether you saved your changes or not.

4.5 SNMPv3

The SNMPv3 settings local engine ID and engine boots counter do not need to be changed manually in general. However, if you encounter connectivity problems then a probable cause might be an agent that is using the same engine ID as MIB Explorer's default engine ID. In this case, you should change the local engine ID (or the remote - if feasible) to make both unique again.

Note: Changes to these settings take effect after restarting MIB Explorer.

The engine boots counter should be strictly monotonic increased. It is automatically increased by one on each restart of MIB Explorer.

- **Authoritative Engine ID**

  The engine ID that uniquely identifies the authoritative SNMP entity MIB Explorer. All SNMPv3 applications and services running in a network must have such unique identifier. It is therefore recommended to include IP address and SNMP port in the authoritative SNMP engine ID. MIB Explorer by default includes the local host name into the default engine ID if available.

- **Engine Boots**

  The engine boots counter should not be decreased because this could cause interoperability problems with running SNMPv3 entities. The boots counter can be manually increased to force time resynchronization with SNMPv3 entities contacted after increasing the counter value.

- **Report Security Level Strategy**

  Reports send by a SNMPv3 entity should be send with the same security level of the request that triggered the report. This the standard behavior of MIB Explorer and complies to the SNMPv3 standard.
  This standard configuration makes it difficult to detect problems with an authentication or privacy configuration, because the report cannot be sent based on the configuration error. In such a case, it might be reasonable to relax the setting to noAuthNoPrivIfNeeded.
  The setting neverNoAuthNoPriv provides maximum security, but also limited error detection.

- **Non-Standard Privacy Protocol OID Mapping**

  There are some privacy protocols which have never been propagated to an RFC. Those protocols have no assigned standard object identifier (OID) to identify them in USM operations. SNMP4J and MIB Explorer use OIDs defined in the OOSNMP-USM-MIB created by AGENTPP.
  Non-AGENTPP based products, might use different OIDs for the non-standard AES protocols with 192 and 256bit keys.
  In the provided table, any other OID can be defined for the listed protocols. The mapping is needed to run the USM key exchange operations to change passphrases/keys with Edit>Create/Modify SNMPv3 User on target systems that use custom OIDs.
  For any other SNMPv3 operations the mapping is not relevant.

4.6 Transport

The supported transport protocols are:

- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)
- Transport Layer Security (TLS)
- Datagram Transport Layer Security (DTLS)

At least one transport protocol needs to be enabled in order to be able to communicate with a SNMP entity. The default transport protocol is UDP, but there are also SNMP agents that support TCP as well as the new (D)TLS transport mappings. TCP and TLS provide better performance for bulk data retrieval than UDP and DTLS respectively. (D)TLS in addition simplifies security deployment by using established certificate infrastructure.

4.6.1 UDP

The UDP transport mapping is the default transport mapping for SNMP. By default the wildcard IP address '0.0.0.0' with the wildcard port '0' will be used to send SNMP requests and receive responses. You may choose other values although ports below 1024 may require system administrator privileges on some operating systems.

When you first run MIB Explorer, the default UDP transport mapping will be added to the configuration.

The elements that define a transport mapping are:

- **Enabled**

  If checked UDP is made available as transport mapping. If UDP is disabled you will not be able to access most SNMP devices, since UDP is the default transport mapping for SNMP.

- **IP Address**

  The local IP addresses and the wildcard address 0.0.0.0 are listed here and can be chosen as source address for SNMP packets sent by MIB Explorer.

- **UDP Port**

  The source UDP port for outgoing SNMP packets. The outgoing UDP port is different from the incoming - well known - SNMP port. It is highly recommended to use the wildcard port 0 the allow MIB Explorer to choose any free port above 1024.

- **Maximum Inbound Message Length**

  The maximum inbound message length defines the maximum allowed number of bytes SNMP response PDUs returned to MIB Explorer may have. It is recommended for most situations to use 65535 bytes. A smaller value may cause `tooBig` errors if an agent tries to sent a bigger response than specified here.

4.6.2 TCP

The TCP transport mapping is an alternative transport mapping for SNMP which can increase performance when retrieving bulk data from a few devices. By default the wildcard IP address '0.0.0.0' with the wildcard port '0' will be used to send SNMP requests and receive responses. You may choose other values although ports below 1024 may require system administrator privileges on some operating systems.

The elements that define a transport mapping are:

- **Enabled**

  If checked then TCP is made available as transport mapping.

- **IP Address**

  The local IP addresses and the wildcard address 0.0.0.0 are listed here and can be chosen as source address for SNMP packets sent by MIB Explorer.

- **TCP Port**

  The source TCP port for outgoing SNMP packets. The outgoing TCP port is different from the incoming - well known - SNMP port. It is highly recommended to use the wildcard port 0 the allow MIB Explorer to choose any free port above 1024.

- **Maximum Inbound Message Length**

  The maximum inbound message length defines the maximum allowed number of bytes SNMP response PDUs returned to MIB Explorer may have. It is recommended for most situations to use at least 65535 bytes. A smaller value may cause `tooBig` errors if an agent tries to sent a bigger response than specified here.

- **Connection Timeout (TCP and TLS only)**

  The connection timeout specifies the amount of time in seconds, a connection (TCP and TLS) remains open while no data has been exchanged. After the timeout, MIB Explorer will close the connection and reopen it, once data has to be sent again.
  The default timeout is one minute. Keeping the connection open for longer time may reduce connection initialization overhead, especially for TLS.

4.6.3 (D)TLS

The basic Transport Layer Security settings are the same as for “TCP”.

4.6.4 (D)TLS Security

With the (D)TLS Security settings, you can choose the (D)TLS version to use and other security properties. Changing the TLS version applies after restarting MIB Explorer.

- **(D)TLS Version**

  TLS v1.0, 1.1, and 1.2 are supported for TLS and DTLS v1.0 or v1.2 are supported for DTLS.

- **Key Store File**

  A Java key store file that contains X.509 certificates for the (D)TLS transport mapping according to RFC 5953.

Note: The password will be stored in plain text in the MIB Explorer configuration file unless you have set a master password.
With a master password set, all USM passphrases, community names, and key store passwords are AES128 encrypted.

- **Key Store Password**

  The password for the key store file.

- **Test**

  Test the loading of the key store file with the specified password and displays success or failure in a message box.

4.6.5 (D)TLS Security Name Mapping

Because the transport layer security model does not exchange a security name on the wire, the security name used by the View Access Control Model (VACM) to authorize access to a SNMP entity, has to be derived from authentication identities. As those identities are certificates, a mapping specification has to be provided that assigns a security name to a fingerprint, subject distinguished name (DN), issuer DN, and other attributes of a X.509 certificate.
Those mappings are used for notifications by MIB Explorer and can be specified here.
The Fingerprint column specifies the certificate's fingerprint that is mapped to a security name. The Type column defines the mapping type (see RFC 5953) and the Data column defines the security name.
If type is Specified then Data contains the security name whereas for all other mapping types, the security name will be derived from a certificate attribute.

4.6.6 (D)TLS Accepted SubjectDN

The (D)TLS protocol requires client and server authentication by exchanging X.509 certificates. The remote SNMP entity then needs to accept the presented certificate based on key attributes of the certificate. Such an attribute is either the subject distinguished name (DN) or the issuer DN of the certificate. You can here specify a list of accepted certificate DNs.

4.6.7 (D)TLS Accepted IssuerDN

The list of Distinguished Names (DN) identifying remote certificates provided here, accept those certificates as authorized to access this SNMP entity.

4.7 View

The view preferences tab can be used to configure the appearance of MIB Explorer:

- **SMI Definition Font Size**

  Specifies which font size should be used for the SMI definition text area.

- **Default OCTET-STRING Display Mode**

  The default OCTET-STRING display mode is used when there is no DISPLAY-HINT defined for a variable binding.

- **Enable syntax highlighting for SMI Definition pane**

  Specifies whether the SMI definition text should be displayed colored or not.

- **Enable syntax highlighting for MIB Editor**

  Enables syntax highlighting for the MIB file editor. Disabling syntax highlighting may slightly improve performance and also disables font styles when printing a MIB file.

- **Split MIB tree and Tools pane horizontally**

  Splitting the main window horizontally between tree and tools panel is the default. Nevertheless, for some purposes it might be useful to split them vertically.

- **Resolve OIDs to object names if feasible**

  If checked, OIDs will be displayed as the last known object name in the path denoted by the OID (last name) plus the remaining OID suffix in dot notation. For example, the OID `1.3.6.1.2.1.1.1.0` will be displayed as `sysDescr.0` if the SNMPv2-MIB is loaded or as `mib-2.1.1.0` if only SNMPv2-SMI MIB module is loaded.

- **Update Browse pane while retrieving instances (slow)**

  For better performance, MIB object instances are not displayed in the browse tab as while they are being received. Instead, they are displayed at once when all instances have been received or the operation has been canceled. By checking this box, immediate update of the browse view is enforced.

- **Enable cell delta highlighting in Table View by default**

  If checked, the Table View will use an orange background for cells that have changed their content between the last and the actual refresh (or manual update).

- **Enable auto-save (restore) of changed column widths in Table Views**

  If checked, changed column widths are automatically saved and restored when the same table (MIB object) is viewed again. As this option saves changed column width for each opened table node, it requires more disk space for the MIB Explorer configuration file, if enabled.

4.7.1 Look and Feel

Choose the Look&Feel you want to use for MIB Explorer. There are three Look&Feels built-in. The selected Look&Feel will be loaded when MIB Explorer is restarted.

Experts may enter the Java class name (including package name) of a third party Look&Feel if the appropriate JAR file is added to the classpath of MIB Explorer.

Some look & feels may cause exceptions on certain platforms, if you encounter such an exception and you cannot start MIB Explorer to change the look & feel to the default again, then remove the row starting with `LookAndFeel` from the `mxp4.cf` file in your home directory. MIB Explorer will then use the default look and feel which does not cause the exception.

4.8 Internet Proxy

With MIB Explorer 5.0 and later, an Internet proxy can be configured for:

- Viewing MIB Explorer help using with the built-in JavaFX browser. See also “View” on page 21.
- Updating MIB Explorer and notifying about the availability of new (free) updates and upgrades, see “Updates and Upgrade” on page 6.

By default, MIB Explorer uses the same settings as your operating system for Internet proxy. There might be cases where the proxy settings of the operating system are wrong, incompatible with Java, or otherwise not accessible.
Then please switch off the check box Use system proxy and provide the following parameter manually:

- **Proxy Host:Port** - The IP(v4/v6) address or the fully qualified domain name of the Internet proxy host is configured by the first field.

In the second field, the TCP port is configured, which is typically 80, 8080, or 3128.

- **No Proxy Hosts** - A list of domain names or IP addresses, separated by a pipe symbol (|) for which no proxy should be used. This should include the at least `localhost|127.*|[::1]` to allow MIB Designer's Java Runtime services on the local host.
- **Proxy server requires password** - Check this option if the proxy you want to use requires authentication using user name and password.
- **Proxy User** - The user name for the proxy authentication.
- **Proxy Password** - The password for the proxy authentication.

Note: The proxy password will be stored in clear text in the MIB Explorer configuration file in your home directory.

5 MIBs

SNMP Management Information Base (MIB) specifications are documents containing definitions of management information so that network systems can be remotely monitored, configured, and controlled. MIB Explorer makes extensive use of all machine readable information within MIBs. This information is available as so called MIB modules.

MIB Explorer is a generic tool for managing any SNMP system. It has only very limited built-in knowledge of MIB information (see "SNMPv3 User Administration" on page 140). As a consequence, it is essential to MIB Explorer to be able to parse MIB modules and compile them into an internal format.

Once MIB module information is available, this Structure of Management Information (SMI) facilitate accessing, formatting, organizing, and modifying MIB object values.

The most important operations on MIBs are:

- Getting MIB Modules.
- Compiling MIBs.
- Creating a MIB Repository.
- Loading MIBs.
- Deleting MIBs.

5.1 Getting MIB modules

MIB specifications developed by the IETF working groups contain prose descriptions and references to other documents that enclose the actual MIB module(s). MIB Explorer compiles SMIv1 (RFC 1155) and SMIv2 (RFC 2578-2580) conforming MIB modules. However, MIB modules have to be extracted from RFC specifications before they can be compiled.

Whereas extracting MIB modules from RFC documents can be done manually by removing any prose descriptions, page headers, and footers from a RFC MIB text document, it is much easier to use the Tool "Extracting SMI from RFC documents" on page 150 to create SMI MIB specification files that can be parsed by MIB Explorer.

If you are looking for an enterprise specific MIB module that did not came along with your SNMP device, then you might want to search for it on the Internet.

5.2 MIB Repository

A MIB repository is a directory that MIB Explorer exclusively uses to store compiled MIB modules in an internal format. Before a MIB module can be loaded into MIB Explorer's MIB tree, it has to be compiled and stored into a MIB repository.

To Create a MIB Repository:

1. From the file menu choose Set MIB Repository. A File Open menu dialog box will appear.
2. Navigate through the file system to the directory where you want to create the MIB Repository.
3. Within that directory, create a new folder by clicking on the Create New Folder () button. The new folder can be renamed by double-clicking it.
4. Choose the new (or any other *empty* folder) by selecting it. Click Open.

Note: Do not double-click the new folder! Otherwise you cannot select the folder itself.

As long as a MIB Repository directory is used by MIB Explorer, it must not be altered outside MIB Explorer - except by other AGENTPP tools, like AgenPro or MIB Designer. Once a valid MIB Repository has been set, you may compile MIB files to store them in the repository.

To Select a MIB Repository:

1. From the file menu choose Set MIB Repository. A File Open menu dialog box will appear.
2. Navigate through the file system and select the MIB Repository directory you want to use.
3. Click Open.

Note: Do not double-click the new folder! Otherwise you cannot select the folder itself.

The MIB repository will be verified. If any inconsistent or corrupted MIB modules are found, a dialog will be displayed with instructions to repair the repository.

5.3 Compiling MIBs

Before you can compile MIB modules into MIB Explorer's internal format, a MIB repository has to be created where the compiled MIBs are

stored. During the first startup of MIB Explorer you will be asked to specify a MIB repository.

Precompiled MIBs

MIB Explorer comes with a set of precompiled SMIv2 MIBs which are located in the repository directory of the MIB Explorer installation. MIB Explorer uses that directory as its initial default repository.

To Compile MIBs

1. From the File menu, choose Compile MIBs (or  from the main toolbar). A file open dialog will appear.
   Alternatively you may also use the Compile New MIBs menu item to compile only MIBs that are newer than the existing or not available yet in the repository. With the MIB Compiler settings, you can fine-tune the behavior of the Compile New MIBs operation.
2. Choose a MIB file, ZIP file or a directory and click Open. If you choose a file, then that file will be compiled and all contained MIB modules (typically one) are stored into the MIB repository

   If you choose a directory or a ZIP file, then recursively all contained files will be parsed. All successfully parsed MIBs will be automatically sorted by their dependencies and then compiled into the MIB repository. Directories may also contain ZIP files.
3. After compilation a message dialog with summary information is shown.
4. Press Details to open the Compiler Log window (see Figure 1). It lists status information for each MIB file compiled. A MIB file that failed to compile has more than zero errors Errors column. To view the errors detected for that file, click on that row. The right pane will then show the file's content and the list of errors below. By clicking on an error description, the error location text is selected in the editor.
5. The MIB modules of the successfully compiled MIB files are automatically stored in the MIB repository. From there, the MIB modules can be loaded into the MIB Explorer application.
   Existing MIB modules will be overwritten (updated). If you do not want to change any MIB modules that already exist in the current MIB repository, then use Compile New MIBs from the File menu.

5.3.1 Compiler Log

The Compiler Log dialog lists the status of all MIB files of a compilation run. If the compilation of a MIB file failed, the Status column displays the text *Failed*. The error messages for that file can be expanded by clicking on the + sign of the file's row. By selecting an error message, its description text will be displayed in the text pane on the bottom of the dialog.
The file name of the MIB file is displayed in the File column, whereas its complete path is displayed in the Path column.

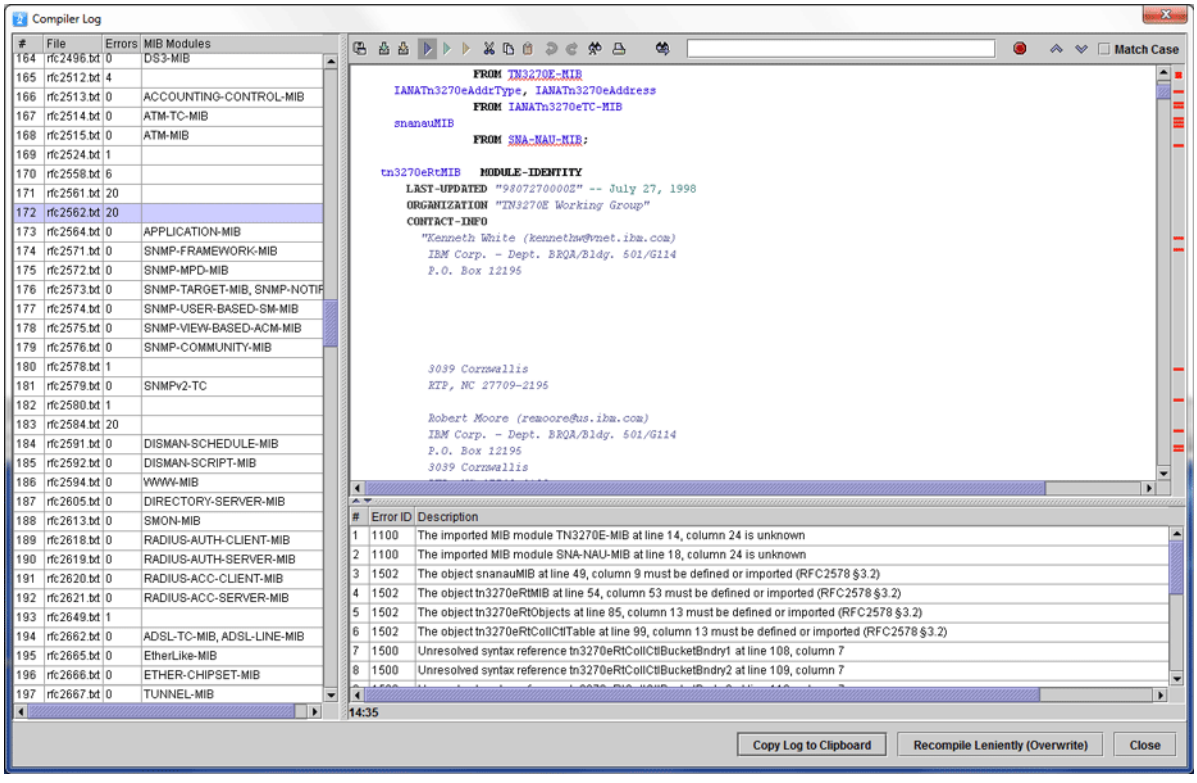


Figure 1: Compiler Log window with compilation errors displayed per MIB module.

To Correct a MIB File

Double click on the row corresponding to the MIB file you want to edit. The MIB file editor window will appear (see section “MIB File Editor” on page 31). Alternatively, you may double click on an error message to directly jump to that error location in the file.
If the error message selected includes location information about the error's line and column, then the editor's cursor will be placed at that location in the MIB file. When you have clicked on the file, the first error

will be located. Otherwise, the first occurrence of the object name corresponding to a semantic error will be searched. The next occurrence of the object name may be found with the Find Again button  .
After having fixed the error, the MIB file can be saved and compiled again by using the Import button  .
If the compilation was successful, the editor window will be closed. Otherwise the cursor of the editor will be positioned on the new error.

Tip: You can print the expanded compiler log's content from the context menu.

5.4 Loading MIB Modules

MIB Explorer needs to load MIB modules from a MIB Repository into its memory to be able to display and use the contained information. For a better overview and performance, it is recommended to not load unneeded MIBs.

To Load MIBs:

1. From the File menu, choose Open/Close MIB (or  from the main tool bar). A shuffle dialog will appear. It contains two lists of MIB modules. The left list shows all MIB modules currently not loaded but available from the MIB repository. The right list shows the MIB modules currently loaded.
2. Select any MIB modules you want to load from the left list of available MIB modules. Click on the Add button to move the selected modules to the right list of MIBs to be loaded. If a MIB that is moved to the right list depends on another MIB module that is currently not loaded, then that MIB (and all MIBs it depends on) will be also moved to the right list. This ensures that MIB Explorer has always a consistent view on MIB data.
3. Select any MIB modules you want to unload (close) from the right list. Click on the Remove button to move the selected modules to the left list of available MIBs. Loaded MIB modules that depend on the removed (unloaded) MIB modules will also be unloaded and thus moved to the left list.
4. Click on the OK button to execute the changes made. Depending on the number of MIB modules that need to be loaded, it may take a while until all modules are loaded and the MIB tree is refreshed.

5.5 Deleting MIB Modules

Deleting a MIB module from a MIB Repository cannot be undone. A MIB module can only be deleted together with those MIB modules that depend on it by importing any MIB objects from it.

1. From the File menu, choose Delete MIB (or  from the main tool bar). A shuffle dialog will appear. It contains two lists of MIB modules. The left list shows all MIB modules available from the current MIB repository. The right list shows the MIB modules that are to be deleted.
2. Select any MIB modules you want to delete from the left list of available MIB modules. Click on the Add button to move the selected modules to the right list of MIB modules that should be deleted. Any MIB modules that depend on a MIB that is moved to the right list will be moved to the right list too. This ensures that MIB Explorer has always a consistent view on MIB data.
3. Select any MIB modules you want to preserve from deletion in the right list. Click on the Remove button to move the selected modules to the left list of available MIBs. Any MIB modules that preserved MIB module depends on will also preserved from deletion.
4. Click on the OK button to execute the changes made.
5. Confirm the deletion of the displayed number of MIB modules by choosing the Yes option.

5.6 Exporting MIB Modules

MIBs can be exported from the current MIB repository to:

1. Plain text files
2. HTML files
3. XML files that are using the SMI DTD v0.1
4. XML Schema files (XSD)
5. PDF

To Export MIBs:

1. Choose Export MIBs from the File menu.
2. Choose the file format for the exported MIB modules.
3. Select the MIBs to export from the list of available modules and press the Add button to add them to the list of modules to be exported.
4. Choose the destination directory.

Any files that already exist in that directory might be overwritten!

5. Press OK to start the export operation. Each MIB module will be exported to a file, whose name will be the MIB modules name concatenated with one of the suffixes `.txt`, `.html`, `.xml`, `.xsd` or `.pdf`.
6. When exporting to PDF, you will be now prompted by an additional dialog for page layout and other document settings. You can choose the page size, footer, outline structure and font size. Press OK to export the selected MIBs with the selected settings.

With the MIB Explorer configuration file entry `mibexplorer.compile.storeFilenameSeparator=-` *you can specify the hyphen character as separator, for example. Other characters or strings can be defined as separator as well, including the empty string.*

The option Append the original file name separated by „_“ can be used to append the original file name of the MIB specification file the exported MIB module was imported from. The resulting file name will then be

```
<module-name><sep><orig-fname-w/o-suffix>.<suffix>
```

By default MIB Explorer will not record the file name of the imported MIB specification file in the compiled MIB modules. The original file name cannot be appended in that case. To let MIB Explorer record the file name, activate this option as described in section “MIB Compiler” on page 9.

5.7 Importing a MIB Module

In contrast to “Compiling MIBs” on page 25, *Importing a MIB* compiles a single MIB file and directly loads the contained MIB modules into the MIB tree.

To Import a MIB:

1. Choose Import MIB () from the File menu.
2. Select the MIB file to import.
3. Press the Open button to compile the selected file, add it to the current MIB repository, and load the contained MIB modules. If there is a syntax error in the MIB file, then the MIB File Editor will open with that file and the cursor will be positioned at the error location in that file. None of the contained MIB modules will be loaded nor added to the MIB repository if the parser detects a syntax error.

6 MIB File Editor

The MIB file editor has the usual capabilities of a text editor including undo and redo. The status bar displays row and column position of cursor. The table below the editor displays error messages from the integrated MIB compiler.
The annotation bar highlights the location of SMI syntax errors in the text. The text is rechecked at least one second after each change. During the update of the annotation bar the overall status on the top is gray.
The background validation of the text can be disabled using the Enable Background Validation toggle menu item of the editor‘s File menu.

6.1 Save, Compile, and Load a MIB File at Once

By choosing Import MIB from the editor's File menu the edited file is saved, compiled, and loaded into the MIB tree. If compilation fails, then the edited MIB module(s) will not be imported into MIB Explorer. Instead an error text will be displayed in the text area below the editor's tool bar. On successful compilation, the MIB module(s) are stored in the MIB Repository and loaded and the editor window gets closed.

6.2 Search and Replace Function

A powerful way to make modifications to a MIB file is searching and replacing by regular expressions.
To search a MIB file by a regular expression, choose Find from the Edit menu. Enter the expression to search for in the opened dialog. The combo box will remember ten expressions used last.
To search and replace found matches, choose Replace from the Edit menu. Enter the search expression and the substitution expression and press OK.
The first matched region in the MIB file will be selected and a confirmation dialog for the replacement operation will be shown. Each substitution can be confirmed individually or all substitutions can be confirmed at once.
The substitution string may contain variable interpolations referring to the saved parenthesized groups of the search pattern. A variable interpolation is denoted by `$1`, or `$2`, or `$3`, etc. It is easiest to explain what an interpolated variable does by giving an example:

Suppose you have the pattern `b\d+:` and you want to substitute the `b`'s for `a`'s and the colon for a dash in parts of your input matching the pattern. You can do this by changing the pattern to `b(\d+):` and using the substitution expression `a$1-`. When a substitution is made, the `$1` means *„Substitute whatever was matched by the first saved group of the matching pattern“*. An input of `b123:` after substitution would yield a result of `a123-`.

6.3 Regular Expression Syntax

A regular expression (or RE) specifies a set of strings that matches it. Thus, a regular expression can be used to check whether an input string is matched by that expression.
Regular expressions can be concatenated to form new regular expressions; if `A` and `B` are both regular expressions, then `AB` is also a regular expression. If a string `p` matches `A` and another string `q` matches `B`, the string `pq` will match `AB`. Thus, complex expressions can easily be constructed from simpler primitive expressions like the ones described here.

A brief explanation of the format of regular expressions borrowed from the Python Library Reference follows.
Regular expressions can contain both special and ordinary characters. Most ordinary characters, like `A`, `a`, or `0`, are the simplest regular expressions; they simply match themselves. You can concatenate ordinary characters, so `last` matches the string '`last`'. (In the rest of this section, we will write RE's in this special style, usually without quotes, and strings to be matched '`in single quotes`'.
Some characters, like "`|`" or "`(`", are special. Special characters either stand for classes of ordinary characters, or affect how the regular expressions around them are interpreted.

The special characters are shown by Table 3 on page 32:.

EXPRESSION	DESCRIPTION
.	(Dot.) In the default mode, this matches any character except a newline. If the DOTALL flag has been specified, this matches any character including a newline.
^	(Caret.) Matches the start of the string, and in MULTILINE mode also matches immediately after each newline.

Table 3: Regular expression syntax characters with special meaning.

EXPRESSION	DESCRIPTION
`$`	Matches the end of the string and in MULTILINE mode also matches before a newline. `foo` matches both '`foo`' and '`foobar`', while the regular expression `foo$` matches only '`foo`'.
`*`	Causes the resulting RE to match 0 or more repetitions of the preceding RE, as many repetitions as are possible. `ab*` will match '`a`', '`ab`', or '`a`' followed by any number of '`b`' s.
`+`	Causes the resulting RE to match 1 or more repetitions of the preceding RE. `ab+` will match '`a`' followed by any non-zero number of '`b`'s; it will not match just '`a`'.
`?`	Causes the resulting RE to match 0 or 1 repetitions of the preceding RE. ab? will match either '`a`' or '`ab`'.
`*?,+?,??`	The `*`, `+`, and `?` qualifiers are all *greedy*; they match as much text as possible. Sometimes this behavior is not desired; if the RE `<.*>` is matched against '`<H1>title</H1>`', it will match the entire string, and not just '`<H1>`'. Adding `?` after the qualifier makes it perform the match in *non-greedy* or *minimal* fashion; as *few* characters as possible will be matched. Using `.*?` in the previous expression will match only '`<H1>`'.
`{m,n}`	Causes the resulting RE to match from *m* to *n* repetitions of the preceding RE, attempting to match as many repetitions as possible. For example, `a{3,5}` will match from 3 to 5 `a` characters. Omitting *n* specifies an infinite upper bound; you can't omit *m*.
`{m,n}?`	Causes the resulting RE to match from *m* to *n* repetitions of the preceding RE, attempting to match as *few* repetitions as possible. This is the non-greedy version of the previous qualifier. For example, on the 6-character string '`aaaaaa`', `a{3,5}` will match 5 `a` characters, while `a{3,5}?` will only match 3 characters.

Table 3: Regular expression syntax characters with special meaning.

EXPRESSION	DESCRIPTION
`\`	Either escapes special characters (permitting you to match characters like `*`, `?`, and so forth), or signals a special sequence; special sequences are discussed below.
`[]`	Used to indicate a set of characters. Characters can be listed individually, or a range of characters can be indicated by giving two characters and separating them by a "`-`". Special characters are not active inside sets. For example, `[akm$]` will match any of the characters "`a`", "`k`", "`m`", or "`$`"; `[a-z]` will match any lowercase letter, and `[a-zA-Z0-9]` matches any letter or digit. Character classes such as `\w` or `\S` (defined below) are also acceptable inside a range. If you want to include a "`]`" or a "`-`" inside a set, precede it with a backslash, or place it as the first character. The pattern `[]]` will match '`]`', for example. You can match the characters not within a range by *complementing* the set. This is indicated by including a "`^`" as the first character of the set; "`^`" elsewhere will simply match the "`^`" character. For example, `[^5]` will match any character except "`5`".
`|`	`A|B`, where `A` and `B` can be arbitrary REs, creates a regular expression that will match either `A` or `B`. This can be used inside groups (see below) as well. To match a literal "`|`", use `\\|`, or enclose it inside a character class, as in `[|]`.
`(...)`	Matches whatever regular expression is inside the parentheses, and indicates the start and end of a group; the contents of a group can be retrieved after a match has been performed (for example in a substitution expression), and can be matched later in the string with the `\number` special sequence, described below. To match the literals "`(`" or "`)`", use `\(` or `\)`, or enclose them inside a character class: `[(] [)]`.

Table 3: Regular expression syntax characters with special meaning.

EXPRESSION	DESCRIPTION
`(?...)`	This is an extension notation (a "`?`" following a "`(`" is not meaningful otherwise). The first character after the "`?`" determines what the meaning and further syntax of the construct is. Extensions usually do not create a new group; `(?P<name>>...)` is the only exception to this rule. Following are the currently supported extensions.
`(?imsx)`	(One or more letters from the set "`i`", "`L`", "`m`", "`s`", "`x`".) The group matches the empty string; the letters set the corresponding flags for the entire regular expression: `i` - Do case-insensitive pattern matching. `m` - Treat string as multiple lines. That is, change "`^`" and "`$`" from matching the start or end of the string to matching the start or end of any line anywhere within the string. `s` - Treat string as single line. That is, change "`.`" to match any character whatsoever, even a newline, which normally it would not match. The `/s` and `/m` modifiers both override the `$*` setting. That is, no matter what `$*` contains, `/s` without `/m` will force "`^`" to match only at the beginning of the string and "`$`" to match only at the end (or just before a newline at the end) of the string. Together, as `/ms`, they let the "`.`" match any character whatsoever, while yet allowing "`^`" and "`$`" to match, respectively, just after and just before newlines within the string. Extend your pattern's legibility by permitting whitespace and comments.
`(?:...)`	A non-grouping version of regular parentheses. Matches whatever regular expression is inside the parentheses, but the substring matched by the group *cannot* be retrieved after performing a match or referenced later in the pattern.
`(?#...)`	A comment; the contents of the parentheses are simply ignored.

Table 3: Regular expression syntax characters with special meaning.

EXPRESSION	DESCRIPTION
`(?=...)`	Matches if `...` matches next, but doesn't consume any of the string. This is called a look-ahead assertion. For example, `Isaac(?=Asimov)` will match '`Isaac`' only if it's followed by '`Asimov`'.
`(?!...)`	Matches if `...` does not match next. This is a negative look-ahead assertion. For example, `Isaac(?!Asimov)` will match '`Isaac`' only if it's *not* followed by '`Asimov`'.

Table 3: Regular expression syntax characters with special meaning.

7 MIB Sets

A MIB set is a named group of MIB modules. MIB sets are stored in the MIB Explorer configuration. Users can define their own MIB sets. MIB sets do not contain any MIB information. Instead they contain references to MIB module names only.

MIB sets can facilitate loading MIB modules for different targets or purposes:

- A MIB set can be associated with a target to ensure that (only) these MIB modules are loaded whenever that target is selected.
- A MIB set can be used to store references to the MIB modules supported (implemented) by a target. In conjunction with the above feature, this can be used to automatically load the MIBs supported by a target when selecting it.
- A MIB set can be created to combine MIBs for any purpose. Using MIB sets can reduce memory consumption, increase performance, and improve overview.

Operations on MIB Sets:

- Creating a New MIB Set
- Editing a MIB Set
- Associating a MIB Set With a Target
- Determining an Agent's MIB Set
- Deleting a MIB Set
- Loading MIB Sets

The following operations are available from the context menu of the MIB Set tab:

OPERATION	DESCRIPTION
New	Creates a new mib set. You will be asked for a name of the new MIB set. The name may include blanks. If the name already exists, an error message will be displayed. By confirming the MIB set name, an empty MIB set will be created in the MIB set tree.
Edit	Edits the content of the selected MIB set. MIB modules can be added to or removed from the MIB set by using a shuffle dialog. Analogous to the "Loading MIB Modules" on page 28 dialog, dependent MIBs are automatically added or removed respectively.
Delete	Deletes the selected MIB set. Since MIB sets only store references to MIB modules, the MIB modules themselves are not deleted. The deletion of a MIB set is not undoable.
Export	Exports some or all of the MIB Set's modules as plain text, HTML, XSD, PDF, or XML files. See "Exporting MIB Modules" on page 29.
Load	Unloads all MIBs from the MIB tree and loads the MIB modules referenced by the selected MIB set from the MIB repository.
Add	Adds the MIB modules referenced by the selected MIB set to the MIB tree. Already loaded MIB modules remain unchanged.

Table 4: Operations of the MIB Set context menu.

7.1 Creating a New MIB Set

To create a new MIB Set:

1. Choose New MIB Set () from the MIB Sets menu.
2. Enter a name for the MIB new MIB set and press

3. OK. The name may include any printable characters and spaces.
4. If the entered name is already used by another MIB set, an error dialog will be display.
5. Otherwise, a new MIB set will be created and selected in the MIB Sets tab. You may then edit this MIB set.

Once a MIB set has been created its name cannot be changed.

Alternatively to step 1, you may choose New... from the context menu of the MIB Set panel.

7.2 Editing a MIB Set

A MIB set consists of references to MIB modules. The contents of a MIB set can be edited at any time.

To Edit a MIB Set:

1. Select the MIB Sets tab from the MIB Explorer's MIBs panel.
2. Select the MIB set you want to edit and press the right mouse button to activate the context menu.
3. From the context menu choose the Edit... item.
4. A shuffle dialog will be displayed that shows all available MIBs on the left side and the MIBs referenced by the selected MIB set on the right. Similar to the Open/Close MIBs dialog you can add (remove) MIBs to (from) the MIB set. By adding a MIB module, MIB Explorer makes sure that all MIB modules imported by the MIB are also added.
5. Pressing the OK button will save the MIB set.

Please note: All MIB modules that are not available in the current MIB repository will be removed from the MIB set when the MIB set is saved.

7.3 Associating a MIB Set with a Target

By associating a *target* with a MIB Set, its MIB modules are automatically loaded whenever the target is selected.

To Associate a Target with a MIB set:

1. Choose Edit->Targets from the main menu.
2. Select the Targets tab.
3. Choose a target from the target table by clicking on its name (first column).
4. Choose the MIB set you want to associate with the target from the drop-down list in the MIB Set column. Choosing the empty entry will remove any association previously made.
5. Save your changes with Save.

You can let MIB Explorer determine the MIB set supported by the agent (See "Determining a Target's MIB Set" on page 40.) and then assign it to the target used to access the agent.

7.4 Determining a Target's MIB Set

Making sure MIB Explorer's MIB tree contains only those MIBs actually supported by the currently selected target, improves overview, increases performance, and reduces memory footprint. Thus, whenever you have created a new target, it is recommended to let MIB Explorer scan the agent for supported MIBs.

Note: MIB Explorer can only detect MIB objects currently available for read access. Empty tables, cannot be detected and if a MIB module implementation in an agent has empty tables only, then MIB Explorer will not include that MIB module in the MIB Set. You can add it manually later or add it by a scan later, when at least one of the tables contain data.

Before a target can be scanned, the following prerequisites have to be met:

- All available MIBs (or at least those possibly supported by the target) have to be compiled into the MIB repository.
- A target has to be configured for the agent.
- That target should have been successfully used to browse the system group of the target agent.

To Determine a Target's MIB Set:

1. Select the target you want to scan from the target tool bar.
2. Choose Determine Target's MIB Set from the MIB Sets menu. MIB Explorer will then immediately start scanning the target as follows:

   ⏩ The scalar and tabular MIB objects of each MIB module in the current MIB repository are requested from the agent.

   ⏩ If MIB Explorer gets a valid response (no error and no exception value) for an object, then that MIB module will be added to the MIB set. This MIB module will not be processed any further. Instead, the next one will be loaded and processed.

   ⏩ If a timeout occurs or an engine ID discovery fails a corresponding error dialog will be displayed and the scan will be stopped.
3. When the scan is complete or has been canceled, a shuffle dialog will be displayed. The right list contains the MIB modules that are at least partially implemented by the target agent. The left list contains all other MIB modules available in the MIB repository. You can add MIBs to the MIB set or remove MIBs from it as described in "Loading MIB Modules" on page 28.
4. Press the OK button to save the MIB set under the target's name. An already existing MIB set with the same name will be overwritten. Press the Cancel button to abandon any changes.

5. After having saved the MIB set you will be prompted to replace the currently loaded MIBs by the MIB modules from the MIB set and associate the MIB set with the current target. If you confirm the dialog, the MIBs modules of the determined MIB set will be loaded whenever you select the current target. Otherwise the MIB set will be saved only.

7.5 Loading MIB Sets

The main purpose of MIB sets is to facilitate MIB loading by grouping MIB modules in named units. A MIB set can be loaded exclusively or additively. When exclusively loading a MIB set, all already opened MIBs are closed before the modules of the MIB set are loaded. Loading a MIB set additively will load those modules of the MIB set, that are not already opened.

To Load a MIB Set:

1. Select the MIB Sets tab from the MIB Explorer's MIBs panel.
2. Select the MIB set you want to load and press the right mouse button to activate the context menu.
3. From the context menu choose the Load item.

Or

1. Choose Open MIB Set... from the MIB Sets menu.
2. Choose the MIBs set you want to load from the shown list and press OK. If there is a MIB set associated with the current target, then this MIB set will be selected as default.

To Add a MIB Set to the MIB Tree:

1. Select the MIB Sets tab from the MIB Explorer's MIBs panel.
2. Select the MIB set you want to add to the MIB tree and press the right mouse button to activate the context menu.
3. From the context menu choose the Add item.

If a MIB set has been associated with the current target, that MIB set can be (exclusively) loaded by choosing Load MIB Set for Target from the MIB Sets menu.

7.6 Saving a MIB Set

The MIB modules loaded can be saved as a new MIB Set or to replace an existing one.

To Save a MIB Set

1. Choose Save MIB Set as from the MIB Set menu.
2. Select the MIB set you want to replace with the currently loaded MIB modules from the drop down list. If the current target is associated with a MIB set, then this set will be selected as default. If you want to save the set of loaded MIB modules as a new MIB set, then enter a new name for the set.

7.7 Deleting a MIB Set

When a MIB set is deleted, any references from targets to this MIB set are also removed. Deleting a MIB set cannot be undone.

To Delete a MIB Set:

1. Select the MIB Sets tab from the MIB Explorer's MIBs panel.
2. Select the MIB set you want to delete press the right mouse button to activate the context menu.
3. From the context menu choose the Delete item.

8 Targets

The set of information that describes where and how to send a SNMP message is called a '*Target*' and consists of three kinds of information:

- Destination information, consisting of the network transport protocol, an IP address or host name, and the port.
- Message processing parameters, consisting of timeout value, number of retries, and SNMP version (message processing model).
- SNMP parameters, consisting of security model (community based or USM), security level, and security name information. For SNMPv3 targets there are additional parameters like engine ID, context, and context engine ID which may be optionally configured.

Targets may be configured for the following purposes:

1. To manage a SNMP agent or proxy agent (also known as command responder).

   ⏩ These targets have typically a host IP address and the default SNMP port 161.

2. To send traps/notifications or Inform requests to a trap receiver application (also known as command generator).

   ⏩ These targets have typically a host IP address and the SNMP trap port 162.

3. To discover SNMP agents in a (sub) network.

   ⏩ These targets may have a broadcast IP address.

Since version 5.0, MIB Explorer allows to define SNMPv3 targets using user based security (USM) in two ways:

- **SNMPv3 /USM**

  User and Target are independently specified and target uses such an USM user by reference

- **SNMPv3 /direct**

  Target directly defines any necessary SNMPv3 user information like security name and protocols and (localized) keys. Using this new definition type, targets are fully independent and can be easier maintained

because passphrase (and resulting key) changes do not need to be executed concurrently on all associated targets.

- **SNMPv3 /(D)TLS**

  With DTLS or TLS targets no user information are exchanged over the wire - even no security name. To be able to associate View Based Security Model access restrictions to such SNMP operations, the notion of a security name is essential.
  Thus, certificates, host-names, fingerprints etc. can be used to map from a certificate/peer to a SNMPv3 security name. This mapping is done independently from any particular target as described by "(D)TLS Security Name Mapping" on page 20.

Figure "Three ways to define a SNMPv3 target using different Version options." on page 44 illustrates which of the three different target types relate to the USM User list (SNMPv3 /USM only) and why the new types SNMPv3 /direct and SNMPv3 /(D)TKS do not need/has any relation to it.

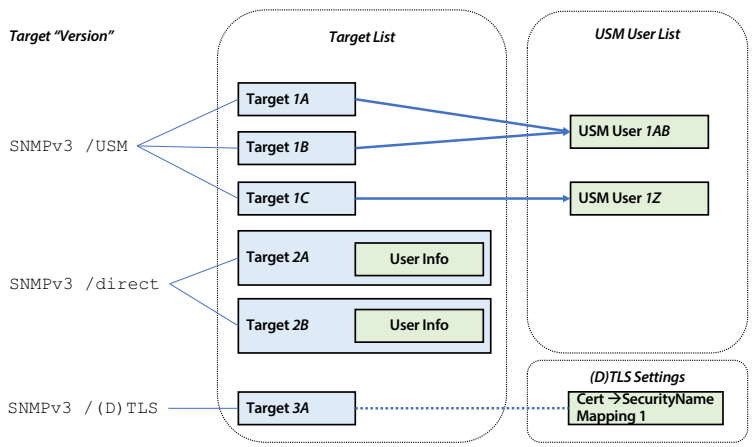


Figure 2: Three ways to define a SNMPv3 target using different Version options.

8.1 Target Configuration

Targets are configured by using target and optionally user editor. The target configuration therefore consists of two tabs with two tables. The first

table is used to configure targets including direct user information and the other is used to configure USM users.
Users in the USM user list may have two possible usages:

1. Provide centrally SNMPv3 USM user credentials for target configurations (with version "SNMPv3 /USM").
2. Define USM users as so called "principals" that are used by the Trap Receiver to receive SNMPv3 user based traps and notifications.

Both tables (Targets and USM Users) can be sorted by arbitrary columns by clicking on the column headers.
By setting a master password with Edit->Master Password all passphrases and community names of the target and user configuration as well as the keystore passwords of the TLS transport are stored AES 128 encrypted in the configuration file within the users home directory.

To Edit Targets:

1. Choose Targets from the Edit menu or  from the tool bar.
2. Edit the targets as described by "Adding a New Target" on page 45 and "Removing a Target" on page 49.
3. Save your changes by pressing OK.

8.2 Selecting the Active Target

To select the target you want to work with:

1. Select the target from the combo-box in the Targets tool bar.
2. Open the Targets editor by choosing Edit->Targets, select the target you want to use, and then choose Set Active from the context menu or select the check-box in the first column named "Active".

8.3 Adding a New Target

Although targets may be used for different purposes, they are created in the same way. Only the used address/port distinguishes between agent, trap, or discovery targets.
The target tab is divided into a upper and lower pane where the upper contains the list of all configured targets and the lower pane provides a form to edit the selected target.
A changed form value is applied to the table whenever the focus is moved from the edited field. If a value is not valid, the field background will change to pink and the table value is not updated until the value has been corrected.

Tip: The name column of the table is not editable and can be therefore used to easily select a row.

To Add a New Target:

1. From the Edit menu choose Targets (). A modal dialog will be shown.
2. Choose the Targets tab.
3. Press the New Target button.
4. A new row will be inserted into the target configuration table and the target name editor is activated. You can now edit the entry using the form on the lower half of the split pane.
5. Enter an unique name for the target.
6. Select the transport mapping (UDP, TCP, TLS, or DTLS) with which the target can be accessed from the Transport column. Which target Version is available, depends on the transport mapping selection. Version SNMPv3 (D)TLS is available with TLS and DTLS only, but not with any other transport mapping.
7. Enter the IP address and port of the target separated by a slash (/). You can also enter a host-name if the target uses Dynamic Host Configuration Protocol (DHCP) to determine its IP address. To access the SNMP of your local system, enter `127.0.0.1/161`, `127.0.0.1`, `localhost/161`, or `localhost`.
   Additional examples are:

   - `switch01:1161` - Switch on non-standard port 1161
   - `92.168.0.1:162` - A trap target using standard trap port 162

   255.255.255.255 - Discovery target using an IPv4 broadcast address - works with UDP only.

   92.168.0.255:4700 - Discovery target for port 4700 in class C network 192.168.0

   080::8:800:200C:417A - An IPv6 address.

   FFFF:129.144.52.38/161 - An standard IPv4 address in IPv6 format.
8. Select the SNMP version for the target:

- **SNMPv1**
  - Community based weak security
  - No GETBULK
- **SNMPv2c**
  - Community based weak security

- **SNMPv3 /direct**

  ⏩ Strong security using SNMPv3 user information directly embedded within the target configuration. Only localized keys are stored - no plain text passwords.

- **SNMPv3 /USM**

  ⏩ SNMPv3 user credentials are not included in target configuration. Instead they are references by name from the MIB Explorer's USM.

  ⏩ Use this version, to manage multiple targets with the same credentials - requires using synchronous multi-target USM key changes!

- **SNMPv3 /(D)TLS**

  ⏩ Mandatory with DTLS or TLS targets.

  ⏩ No security information in the target configuration, instead security name and trusted certificates are configured in the (D)TLS Security preferences.

9. Choose the Timeout value and the number of Retries.
10. Choose the MIB set you want to associate with the target from the drop down list. Select the empty entry, if you do *not* want to associate any MIB set with the target.
11. If you have chosen SNMPv1 or SNMPv2c as SNMP version then enter the community to be used with the target.
12. Otherwise, if you have chosen SNMPv3 /USM then select an USM user from the drop-down list. If you need to add a new user then create it using New User.
13. Otherwise, if you have chosen SNMPv3 /direct then configure the embedded user credentials as described in
14. You can continue to specify the optional SNMPv3 security parameters engine ID, context name, and context engine ID as described below.
15. Save the new target into MIB Explorer's configuration by pressing OK.

To Configure Optional SNMPv3 Security Parameters:

1. Use the Engine ID label button to discover the targets Engine ID. Leaving the engine ID field empty will let MIB Explorer discover the target's engine ID automatically, once for each session.
2. Enter the Context to be used with the target as plain text. The default is an empty string.

3. Enter the Context Engine ID which selects the subsystem or proxy as a plain text or hexadecimal string, for example `0f:ab:12:A:g5` (use the context menu Format to change the input format). The default is an empty string. In this case, MIB Explorer will use the entered or discovered engine ID as context engine ID.

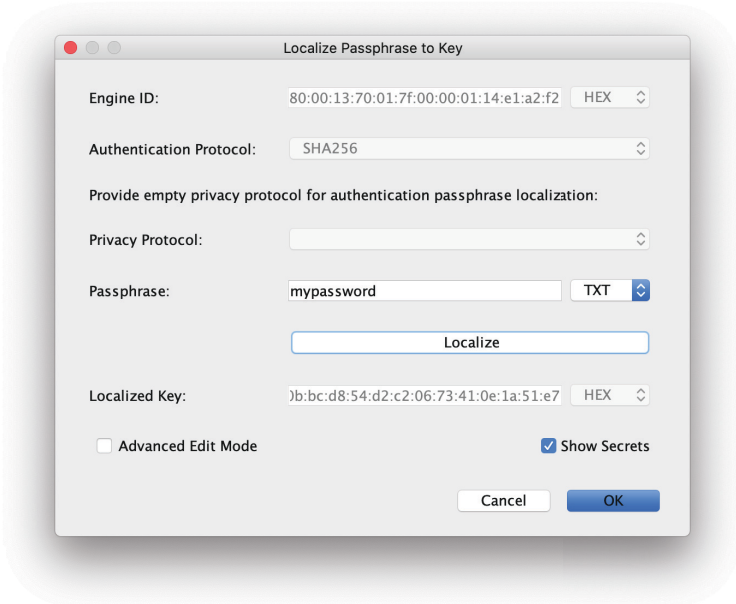


Figure 3: Dialog for localizing a passphrase to USM key

To Configure Embedded User Credentials:

1. Enter security name.
2. Select authentication protocol - or leave it empty if security is not needed.
3. Select privacy protocol - or leave it empty if security is not needed.
4. If you have chosen an authentication protocol:
   Click on the Authentication Key label and a dialog will be shown to enter a clear text password in the Passphrase field.
5. Click on Localize to compute and preview the Localized Key (use the Show Secrets checkbox to actually show key and password on the screen).
6. Press OK to save the computed localized key into the corresponding key field of the target's embedded user configuration.

If there was already a key in the corresponding embedded user field, then its content will be shown here in the pass-phrase field as hexadecimal (HEX) representation. Then select the format "__" to clear the existing key and switch to the text (TXT) format immediately.

7. Repeat steps 4-6 for the privacy key.
   The only difference is here, that now authentication *and* privacy protocol have to by set in the localization dialog.

8.4 Removing a Target

1. From the Edit menu choose Targets (). A modal dialog will be shown.
2. Choose the Targets tab.
3. Select the target to delete from the by clicking on it (using the Name column).
4. Press the Delete Target button.
5. Select a target you want to work with and choose Set Active from the context menu or select the corresponding check-box in the column Active.
6. Press the Save button.

Removing a target will not invalidate monitor configurations using that target, however the removed target will no longer available for the discovery configuration after restarting MIB Explorer.

8.5 Communities

A SNMPv1/SNMPv2c community typically consists of at least one agent and one or more managers. A community is named by a string of octets which is called a community string. Although many SNMP developers and users believe that a community string is a password, its originally intended use was not that simple. Nevertheless, many agent implementations are using a community string as password for read-only access and another for read-write access.
With MIB Explorer you can specify a single community for each target that is used for all request types. But of course, you may define more than one target definitions for the same agent (i.e., command responder application).

Community strings are send as plain text over the wire.

8.6 USM Users

The User based Security Model (USM) associates a user name with security information and is defined in RFC 2574. A USM user consists of:

- **User Name**

  An internal name for the user. In most cases this name would match the security name. The user name must be unique within a MIB Explorer configuration.

- **Security Name**

  Identifies the user. The security name is used to refer to an user in many MIBs, in particular the SNMP-VIEW-BASED-ACM-MIB maps security model/name combinations to groups in the VACM. Without such a mapping a USM user cannot access any MIB information in an agent.

- **Authentication Protocol**

  Determines the authentication protocol to be used for this user:

  - no authentication
  - MD5, SHA-1 (deprecated, because safer protocols with SHA-2 are available now)
  - SHA-2 algorithms:
    - SHA-224
    - SHA-256
    - SHA-384
    - SHA-512

- **Authentication Passphrase**

  If an authentication protocol is selected, an authentication passphrase has to be entered too. That will be combined with the target's SNMP engine ID to form the localized authentication key by using the selected hashing algorithm.
  Instead providing the passphrase, you can also directly compute the localized key and enter it (see below).

  If you do not provide a Localization Engine ID, which is recommended here for an USM user, then the target's engine ID will be used to localize passphrases on-the-fly. The USM user can thus be used securely for several SNMP targets.

  If you provide a Localization Engine ID then this user can only be used with a target whose authoritative engine ID equals the used localization engine ID.

To enter a passphrase in hexadecimal format, use the format button on the right of the input field.

- **Privacy Protocol**

  Determines which privacy protocol to be used for this user:

  - no privacy
  - DES, 3DES
  - AES128, AES196, or AES256 is used with this user.

  The nonstandard privacy protocols AES192-KeyExt3DES and AES256-KeyExt3DES are provided to ensure interoperability with some devices that implemented AES 192 and 256 privacy with a key extension algorithm specified for 3DES. Although, that combination was never specified by an IETF RFC or draft, it has been implemented by some manufactures and is necessary for interoperability if using inappropriate authentication algorithms that produce too short keys for those AES types.

Note: As long as you use SHA-256 or higher, key extension is not needed and both AES implementations are compatible on the wire.

- **Privacy Passphrase**

  If the privacy protocol DES is selected then the entered privacy passphrase is localized with the selected authentication protocol (analogous to localizing an authentication passphrase) and then used to encrypt/decrypt SNMP messages.

- **Localization Engine ID**

  The localization engine ID can be left empty by default. However, if two targets use the same security name with different passphrases and/or authentication/privacy protocols then you need to localize each user for its specific engine ID to avoid clashes. You can localize the passphrases of an user easily by using Localize User from the context menu. It prompts for the target engine ID used for the localization in hexadecimal format. Once you press OK, the passphrases are localized and the entered localization engine ID is stored with the USM user security credentials.

MIB Explorer abstracts from security names by using User Profiles. The User Profile Name is independent from the user's security name. Nevertheless, it makes sense to choose the profile name according to an user's security name for better readability.

8.7 Adding an USM User

Please note that adding a new user to MIB Explorer's configuration does not create that user in the USM MIB of the target for which you added the user. To create a new user in the USM MIB of one or more targets, use the SNMPv3 user administration (See "SNMPv3 User Administration" on page 140.).

To Add a USM User Profile:

1. Select Targets from the Edit menu.
2. Press the Add User button from the tool bar. The USM User tab will be shown and a new row at the bottom of the table will be inserted. The user name editor in the form in the bottom split view is then activate to enter the name of the new USM user profile.
3. Enter a unique name for the user profile. If possible, the profile name should be equal to the security name of the user.
4. Enter the properties of the USM user (See "USM Users" on page 49.).
5. Press the Save button.

8.8 Deleting an USM User

A user profile can be deleted from MIB Explorer's configuration if there is not any target using that profile any more. Otherwise a different user has to be selected for those targets first. Deleting a user profile does not delete the corresponding USM user in the SNMP agent associated with the target. In order to delete a user from the USM MIB of an agent use the SNMP Tables function on the `usmUserTable` MIB object.

To Remove a User Profile:

1. From the Edit menu choose Targets (). A modal dialog will be shown.
2. Choose the USM Users tab.
3. Select the user profile to be deleted.
4. Press the Remove User button to delete the profile. If other targets are also referencing that user the button is disabled and the user cannot be deleted.
5. Press the Save button to finally commit your change.

8.9 Target Statistics

In target selection tool bar there is a button named Statistics. It opens a dialog which displays a sortable read-only table that provides for each target address/port combination statistics about the number of messages totally sent to the SNMP entity, as well as the number of messages needed per request (one is optimal here). In addition, the number of retries needed to fulfill the request as well as the runtime of the successful message exchange is counted.
The columns of the target statistics table are described in the below table:

COLUMN	DESCRIPTION
Address	The IP address and port of the target that was subject to SNMP requests that expected a response PDU.
PDU Size	Classifies the PDU size in number of variable bindings in the received response. The value * represents any PDU size. Otherwise the PDU size range is provided, whereas he lower bound is inclusive and the upper is exclusive.
Total Messages Sent	The total number of confirmed messages sent to this target since the last start of the application.
Sent/Request MIN	The minimum number of messages sent per confirmed SNMPv3 request. SNMPv1/v2c messages are not counted.
Sent/Request MAX	The maximum number of messages sent per confirmed SNMPv3 request. SNMPv1/v2c messages are not counted.
Sent/Request AVG	The average number of messages sent per confirmed SNMPv3 request. SNMPv1/v2c messages are not counted.

Table 5: Column descriptions of the Target Statistics table.

COLUMN	DESCRIPTION
Retries MIN	The minimum number of retries that where needed to receive a response for a SNMPv3 confirmed request. *Note: More retry messages could have been actually sent than this value indicates.*
Retries MAX	The maximum number of retries that where needed to receive a response for a SNMPv3 confirmed request. *Note: More retry messages could have been actually sent than this value indicates.*
Retries AVG	The average number of retries that where needed to receive a response for a SNMPv3 confirmed request. *Note: More retry messages could have been actually sent than this value indicates.*
Runtime MIN	The minimum number of milliseconds elapsed between sending the request message (initial request or retry) and receiving the corresponding response PDU (SNMPv3 only).
Runtime MAX	The maximum number of milliseconds elapsed between sending the request message (initial request or retry) and receiving the corresponding response PDU (SNMPv3 only).
Runtime AVG	The average number of milliseconds elapsed between sending the request message (initial request or retry) and receiving the corresponding response PDU (SNMPv3 only).

Table 5: Column descriptions of the Target Statistics table.

9 MIB Tree Panel

The MIB tree panel displays the MIB objects defined by the *loaded* MIBs as a tree. Each node represents an object. All objects below the Objects node have an object identifier assigned which is shown as tooltip. Type assignments and textual conventions do not have an object identifier assigned and thus they are displayed below the Textual-Conventions node.

By clicking on a node, the SMI definition of the corresponding object is displayed in the SMI Definition tab of the Info/Tools panel. In addition, the MIB module(s) defining the object will be selected in the MIB Modules tab of the MIBs panel.

All MIB Explorer's basic functions can be accessed from the tree node's context menu which is displayed by clicking with the right mouse button on a node.

Instances of MIB objects (variables) can be displayed in the MIB tree as well.
In contrast to MIB objects, instance nodes are displayed as two parts:

See also the below sample MIB tree cutout.

1. The instance's index objects' values enclosed in brackets (`[ ]`).
2. The value of the variable.

The OID of a MIB object instance is composed by appending its index OID to the OID of the corresponding object type. MIB Explorer displays the index OID of an instance within two brackets `[`and`]`. Scalar instances like `usmUserSpinLock.0`, have `[0]` as their index value if the corresponding MIB object is known. If not, the index denotes the whole path down from the last known MIB object's OID to the instance's OID.

The complete OID of an instance or any other node is shown by the node's tooltip.

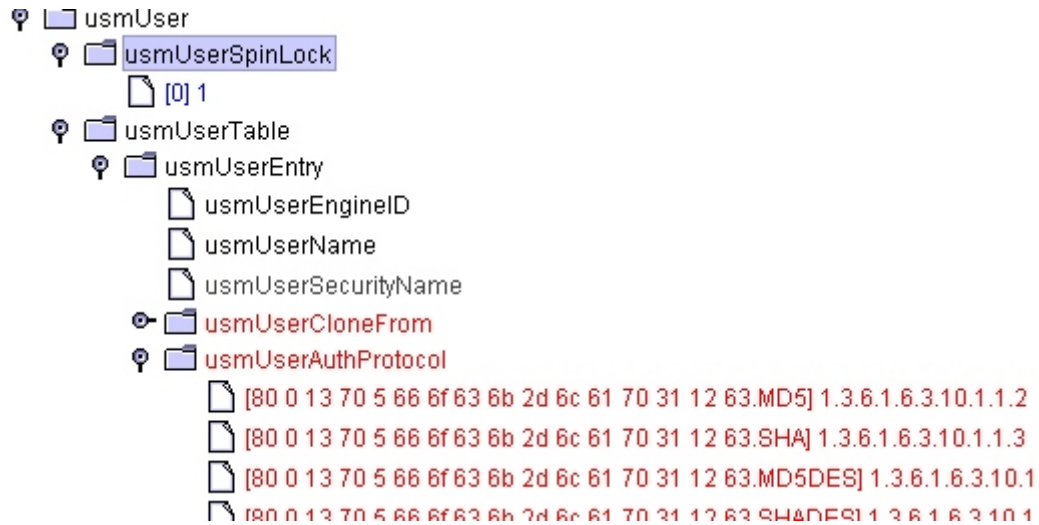


Figure 4: Sample MIB tree cutout with MIB object nodes and instances.

9.1 Browse Tab

The browse tab shows MIB object instances in a table. The object instances are retrieved from a target agent using the Browse function. You can directly change browsed values within the browse tab by simply editing them. All values with a blue background can be edited.

By changing the general preferences to allow changing read-only values, all browsed values may be edited even if the corresponding MIB definition specifies it as read-only.

Whether a value is editable or not, is determined by their corresponding MIB specification. If such a specification is not available (i.e., not loaded) then the value is assumed to be editable. However, the agent may reject a change in any case because of its own constraints.

The result table of the Browse Tab can be searched for object names and values using the ?? Search Panel if the Browse Tab is selected.

To Browse a Target:

1. Select the target you want to browse from the Target tool bar.
2. Select the subtree (node) you want to browse in the MIB tree.
3. Choose Browse ( ) from the Edit or the tree's context menu.

During the browse operation, the status bar shows the instances received from the agent so far. Retrieved instances are displayed immediately, when

the option "Refresh browse view during operation" is enabled in Preferences. However, this option drastically slows down the operation.

The browse operation can be canceled by pressing the status bar's Stop button (). Any instances received so far will then be displayed in the browse table.

The Progress Bar displays the portion of the timeout value passed for the current request. If the displayed value reaches 100% during a browse operation you should consider increasing the timeout value for the target.

9.1.1 Result Table

Each row in the table represents a MIB object instance. The columns are:

- **OID**

  The object identifier of the object instance. Typically the OID of an instance is displayed as the corresponding MIB object's last name and the instance sub-identifiers. The complete OID is displayed as tool-tip or if the option "Resolve OIDs to object names" is deactivated in Preferences.

  If the corresponding MIB object for an instance is not defined in the loaded MIB set, then the row will be displayed with orange background.

Tip: By clicking on an OID, the corresponding object (denoted by the OIDs last name) will be selected in the MIB tree.

- **Syntax**

  The SNMP syntax of the MIB instance as received from the agent. If the syntax is not compatible with the syntax defined for the corresponding MIB object in a loaded MIB module, then the row will be displayed with red background. A red background may also indicate that there are two MIB modules loaded that define the same MIB object with different syntax. If you are unsure whether a returned value does not match its MIB definition, check with the Node Info window whether all syntax definitions for that object are consistent.

- **Value**

  The value of the MIB instance. Values are formatted according available MIB information such as DISPLAY-HINT and enumerations. If there is no formatting information for an OCTET STRING type, then its value is displayed as a printable string where non-printable characters are displayed as dots. The tool-tip then displays the hexadecimal string representation.

Tip: You can copy the content of a single cell or a range of cells to the clipboard by pressing <Ctrl-C>*. The format of the copied content is compatible with most spreadsheet applications.*

If the value has a blue background then you can directly edit it. When you press <Enter> after having changed the value it will be directly changed in the agent by sending a SET PDU to the active target. If the SET request fails a corresponding error message is displayed in the status bar. Changes can be undone and redone by using the context menu of the value column. For a description of the input formats supported, please see the Set Dialog documentation.

9.2 Colors

The node label colors in the MIB tree have the following meaning:

- Black denotes a not-accessible or accessible-for-notify MIB object as well as textual conventions or type assignments.
- Gray denotes a read-only MIB object type or instance.
- Light-Gray denotes any MIB object that is obsolete or deprecated.
- Blue denotes a read-write MIB object type or instance.
- Red denotes a read-create MIB object type or instance.
- Orange denotes a trap or notification type.

The background color green denotes a bookmarked MIB object.

9.3 MIB Tree Context Menu

The MIB tree's context menu can be accessed by clicking with the right mouse button on a node. It provides the following functions that are also partially available from the main menu:

- **Browse**

  Selects the browse tab and sends a series of GETNEXT (SNMPv1) or GETBULK (SNMPv2c/SNMPv3) requests to the target agent in order to get all instances that are available in the selected node's subtree.

- **Get**

  Displays all MIB object instances whose OID starts with the OID of the selected node in the MIB tree. If the SNMP protocol for the current target is SNMPv1 a series of GETNEXT request PDUs is sent to the target until an error or an instance with an OID outside the node's subtree is returned. If the protocol is SNMPv2c or SNMPv3 a series of GETBULK requests is sent until an error, exception, or an OID outside the node's subtree is returned by the target agent. Getting a single instance, for example the instance of a scalar MIB object, can be inef-

fective, depending on the max repetitions value set in Preferences. In such cases it might be more efficient to issue a Get on a parent node in order to get more instances at once.

A double click on a node is a shortcut for this function.

- **Table**

  Opens a table view that displays scalar or tabular instances as a table for the currently selected target agent.

- **Table** (*)

  Opens a table view that displays scalar or tabular instances as a table. The table content is sorted by *instance index* + *target* by default. Thus, row instances from different targets but with same index can be directly compared.
  With the Copy... function from the table's context menu, you can copy the writable column's values to one or more target rows of the same or a different target agent.

- **Grid**

  Opens a grid view that displays one or more related SNMP tables as a hierarchy of tables. The Grid View is not fully supported for Nimbus look & feel.

- **Set**

  Opens a set dialog to modify or create a single MIB object instance.

- **Subtree**

  - Clear

    Remove all instance nodes from the subtree. MIB object nodes are not affected by this operation.

  - Expand All

    Expand all nodes (including MIB object instances) in the subtree.

  - Collapse All

    Collapse the subtree.

  - Hide Absent

    Remove those nodes from the subtree that denote a subtree which (currently) does not contain any MIB instances available from the target agent. In particular, conformance statements, object groups,

and notification definitions will be hidden from the tree. To show all objects again, choose Refresh from the Edit menu.

- Dump Subtree

  Dump the subtree to a given text file as a tree. MIB instances contained in the subtree will be dumped too. If you do not want instances to be dumped, remove them from the subtree first by using the Clear button. This function is especially useful for documentation purposes.

- **Node Info**

  Shows the Node Info window that displays all SMI definitions for the selected node's OID. Normally there is only one MIB module that defines an OID.

- **Bookmarks**

  - Toggle

    Toggle the bookmark for the selected node. Only MIB objects with OID can be bookmarked.

  - Previous

    Go to the closest lexicographic smaller MIB object (if available) that is bookmarked.

  - Next

    Go to the next lexicographic greater MIB object (if available) that is bookmarked.

- **Copy OID**

  Copies the object identifier in dot-notation into the clipboard.

9.4 Set Dialog

The set dialog can be used for setting (modifying or creating) a single MIB object instance at once. If you only need to modify existing values, you can also use the Browse panel.

To Perform a Set:

1. Select a node in the MIB tree representing

- the MIB object instance you want to modify or
- the MIB object for which you want to create/modify an instance.

2. Choose Set Instance from the Edit menu or Set from the nodes context menu.
3. If you have selected a non-scalar MIB object, enter the index value(s) for the instance you want to modify/create. The format for the entered index objects depend on the type of the respective object (see below). Each given index value will be checked for size constraints. If an object value violates such a restriction, then the corresponding object name will be reported in an error message when OK is pressed, the PDU will not be sent, and you will then be able to adjust the value.
4. Enter the new value in the Value field. If the underlying node is an instance, its value and index will become the preset values, otherwise the DEFAULT value specified for the MIB object (if present).
   The value format for OCTET STRING syntaxes is as specified in the DISPLAY-HINT clause of the OBJECT-TYPE specification, if present. Otherwise, the default format as specified in the View preferences (see section "View" on page 21) applies if that is not MIB. In the latter case, the string has to be entered in hexadecimal format.
5. Press OK to send the SET request PDU to the target agent.
6. If the request was successful, the corresponding instance node will be created or updated. Otherwise the Status Bar will provide information about the error occurred.

Depending on the object's Type the value has to be entered as follows:

- **OCTET STRING**

  If there is a DISPLAY-HINT format defined for the MIB object, the value has to be entered according to that format. For example, DisplayString values (format "255a") are entered as plain text.

  If there is no DISPLAY-HINT defined and no default format given in the View preferences then the value is entered as an hexadecimal string, for example: '`ab 8 2f 16`' without the quotes.

  For convenience, ASCII characters may be entered directly when they are enclosed by double quote characters (`"`). For example, if you enter ' `"publi"1c` ' this will result in the hex string '`70 75 62 6c 69 1c`'.

- **OBJECT IDENTIFIER**

  Enter an object name and press enter to convert the name into an OID and append then any instance suffix. For example, enter `ifIndex` and press Return and the displayed value will become '`1.3.6.1.2.1.2.2.1.1`'. Now append '`.2`' to reference the second

interface. The resulting OID ('1.3.6.1.2.1.2.2.1.1.2') can then be send to the target agent.

- **Enumerated INTEGER**

  Select one of the enumerated values from the provided list or enter an integer value to set a value not defined in the MIB.

- **TimeTicks, Counter32, Counter64, Unsigned32, Integer 32, INTEGER**

  Enter a numeric value. You can use the spin box to browse through possible values. Range restrictions defined for the MIB object are not enforced by MIB Explorer, because the agent will reject out of range values anyway.

  If there is a DISPLAY-HINT format defined for the MIB object, no spin box will be displayed and the value has to be entered according to that format.

- **BITS**

  The value may be entered as a series of `1` and `0` characters that are grouped in packets of eight bits separated by spaces. For example to set the second, fourth, and thirteenth bits enter '`01010000 00001`' without quotes.

- **IpAddress, NetAddress**

  Either enter the IP address in raw IP address format (e.g. "`192.168.0.1`") or enter the host name (e.g. "`www.mibexplorer.com`") and press Enter to resolve it into a raw IP address.

If a value does not comply with the input format, a beep will be emitted.

9.5 Node Info

The node info window is opened by choosing the Node Info menu item ( ) from the MIB tree context menu. It shows detailed information about the SMI definitions for a node in the MIB tree. The info window can be opened for MIB object nodes under the Objects root node but not for instances.

The tab title denotes the MIB module that contains the shown definition which is:

- The Name of the MIB object. The object name may differ for OBJECT-IDENTIFIER objects between MIB modules. For example the OID "`0.0`" is named "`null`" in RFC1155-SMI and "`zeroDotZero`" in SNMPv2-SMI.

- The object identifier (OID) of the MIB object. The OID value is the same for all module tabs.
- By clicking on the Path value, the named sub-identifiers from the current node up to the root node are displayed in a list.
- If the node is an OBJECT-TYPE, the Effective Syntax represents the node's base syntax with all refinements. This base syntax is used by MIB Explorer (in combination with a possibly defined DISPLAY-HINT) to render and edit instances of the object.
- The Definition text area shows the complete definition from the corresponding MIB module.

9.6 Search

You can search the whole MIB Tree including possibly retrieved MIB object instances by regular expressions. A node whose specified property matches the given regular expression will be selected. With the Find Again () menu item or button you are then able to find the next node that matches the expression.

To Find a Node:

1. Choose Find from the Edit menu or press from the main tool bar. The search dialog will be displayed.
2. Enter the Search Expression in regular expression syntax.
3. Select whether case should ignored or not. If selected, this will insert `(?i)` at the beginning of the used search expression.
4. Select what type of properties of a node you want to be matched against the search expression. Choosing All will match the whole SMI text of a MIB object node, including key words, or the value part of a MIB instance node against the given search expression.

To Find a Node Again:

1. Choose Find Again from the Edit menu or press from the main tool bar. The next node in depth first search order from the currently selected node will be searched matching the previously specified search expression and options.

10 Table View

MIB Explorer provides two kinds of table views:

1. SNMP Tables

   Displays a table with rows and columns as defined in a special OBJECT-TYPE object definition whose name ends with `Table` by convention. The instances of a row are identified by an uniform indexing scheme that is defined by a row object definition by its INDEX clause. This index value is displayed for each row in the first column, the Index column. It has a gray background and is frozen.

2. Scalars

   Displays a list of scalar object instances. Since scalar instances are always identified by a zero OID suffix ".0", there is no Index column.

If the table has a `RowStatus` column, the keys <INS>, <DEL>, and <Ctrl>-<INS> can be used to modify the `RowStatus` column of the current row.
Pressing <INS> changes the `RowStatus` to `createAndGo(4)`, <Ctrl>-<INS> to `createAndWait(5)`, and <DEL> to `destroy(6)`.

MIB Explorer can display both kinds also in a *multi-target* mode. Here the same table information of one or more agents can be displayed (and compared) in a single table view. This is accomplished by appending the target address to the row indexes.

10.1 Context Menu

A context menu is available for each column of a SNMP table (for each row of a Scalars view) which provides additional information like the MIB object's description and its effective syntax.

In addition, the display and input format for columns with OCTET STRING syntax can be chosen. The chosen format overrides the default format chosen in the View Settings. The available formats are:

FORMAT	DESCRIPTION	SAMPLE VALUE DISPLAY OF „ABC“
ASCII	Plain text formatting.	`aBc`
Decimal	Formats each byte as a decimal value with a dot (".") as separator.	`97.66.99`
Hexadecimal	Formats each byte as a hexadecimal value with a colon (":") as separator.	`61:42:63`
Octal	Formats each byte as a octal value with a colon (":") as separator.	`141:102:143`
Binary	Formats each byte as a binary value with a colon (":") as separator.	`1100001:1000010:1100011`

Table 6: The available column formats of the table view.

If the view setting "Enable auto-save (restore) of column widths in Table View" is checked in Preferences, then any manual column width change will be stored for a table node and restored when it is opened again.

With the Copy... menu item, the writable cells of a row (or column in transposed view) can be copied to one or more other existing rows.

10.2 Tool Bars

The main tool bar contains the following elements (items marked with * are not available for scalars):

Figure 5: Table View's main tool bar.

- **New Row** (*)

  Add a new row to the table. Before the new row can be inserted into the table the row's index has to be specified. You will be prompted to enter a value for each index object. The presented input fields depend on the type of the index object. For a description of the different input fields see “Set Dialog” on page 60.

In a multi-target table view, you need also to select the targets for which a row with the given index should be created.

Duplicate Row

Duplicates the selected row and prompts for its new index, because an index must be unique within the same agent (and context). If the table view had been opened for multi-target view, the user is also prompted for the target for which the new row should be created.
The duplicated row is not committed to the target agent. Thus it can be edited before committing it with Commit button.

Apply Filter (*)

Note: You may create a new row in the table even if its index does not match the filter criteria, but if you refresh the table contents, that row will not be displayed anymore. Also if there were rows added to the table that have not been committed (sent) to the agent yet, then those rows will be removed from the table when applying a filter. There is no undo available for applying a filter!

Restricts the requested and viewed rows in the table to those rows whose index value matches a given range. In the filter dialog you may specify a lower and/or an upper bound. If you do not want to specify a lower bound, then uncheck the "Use lower bound" check box. If you do not want to specify an upper bound, then uncheck the "Use upper bound" check box. If you want to disable the filter, then uncheck both.

Toggle Cell Delta Highlighting

Enables or disables the highlighting of cell data changes with orange background color. When the data in a cell changed after a refresh or by manually modifying its value, the cell gets a orange background instead a blue (writable column) or white background (read-only column).

Refresh

Invalidates the table contents, resets (deletes) the undo/redo history, and then retrieves the table data from the target agent. The operation can be canceled by pressing the Cancel button of the progress window.

Each GETNEXT (SNMPv1) or GETBULK (SNMPv2c/v3) PDU is filled up with column/instance OID's up to the maximum number of variable bindings per PDU as set in “Preferences” on page 8.

Thus, more than one PDU may be sent to the agent per row. If an agent returns a `tooBig` error, MIB Explorer splits the original request PDU into two PDUs and (re)sends those PDUs to the agent.

Redo

Redo last undone change.

- **Undo**

  Undo last change. The maximum number of steps that can be undone is set in MIB Explorer's “Preferences” on page 8.

- **Transpose** (*)

  Swaps rows and columns. This might be useful for tables with many columns.

- **Find**

  Finds the next cell in the table whose value (as displayed) matches a given regular expression (See “Regular Expression Syntax” on page 32). The search starts at the current position and continues row by row from left to right.

- **Find Again**

  Finds the next cell in the table whose value (as displayed) matches the last search expression.

- **Replace**

  Replaces occurrences of a given regular expression with a substitution (expression) in any writable cells.

  Note: In contrast to the Find *and* Find Again *functions, the* Replace *function does not match the cell values as they are displayed, rather than it compares the search expression against the representation of a cell's value as shown by the cell editor.*

- **Preferences**

  Configures whether OID values in this table are displayed with last-name and instance identifier or numerically. In addition, one can specify whether the cell tool tip should display the cells value or its type. Displaying the type is useful when modifying cells to determine the required type of a value.

- **Print**

  Open a dialog to print the table on a printer or save the output to a PDF, PCL, or PS file. Before actually printing the table, the result can be previewed on the screen. The table can be wrapped or shrunk to fit on the selected paper size.

10.2.1 Refresh Tool Bar

With the refresh tool bar you specify the refresh interval in seconds and whether the Table View is periodically refreshed or not. You can either use the predefined values or enter your own value. By pressing Enter,

periodically refresh starts with the given refresh interval. 0 deactivates periodical refresh.

Figure 6: Table View's periodic refresh tool bar.

- **Start**

  Starts periodically refresh, if the refresh interval value is not zero (or disabled).

- **Pause**

  Stops the periodical refresh.

- **Monitor**

  Shows the time passed until the next refresh.

- **Export Data**

  By pressing the toggle button, you can specify a comma separated value (CSV) text file, an Excel (XLS), or an XML file to save the table data to whenever the table is refreshed. The CSV as well as the XLS file can be later opened by a spreadsheet application. As long as the toggle button is pressed, refreshed table data will be written or appended to the file. If a file already exists you can choose whether new data should replace existing data in the file or if the new data should be appended. If a file does not exists, then the new data will be appended. This is also the case, if the periodic refresh is enabled.

  If the XLS file format has been selected as output format, data is appended sheet by sheet

10.3 Table

The table area is composed of:

- **Column Labels**

  The column table texts are the names of the corresponding object types. If there can be identified a common prefix for all columns in a table (including its index columns), then the labels will be displayed without that common prefix.
  By clicking on a column header the rows in the table are sorted on that column in ascending order. Another click on that column sorts in

descending order. A third click resets sorting on that column. If the table is sorted by more than one column, the table is sorted by those columns from left to right.

- **Index Columns** (*)

  The index columns represent the index objects defined in the INDEX clause definition of a table row definition. Index columns have a gray background and are frozen (not scrolled). Index columns cannot be edited.

- **Read-Only Columns**

  Columns that cannot be edited have a white background.

- **Read-Write Columns**

  Columns that can be edited have a blue background.

  The object instances are displayed and edited within the cells of the table. You may *copy a range of cells* to the clipboard using <Ctrl-C>. The columns of the copied cells will be separated by a <Tab> character and the rows by a newline. Cell values may also be *pasted* from the clipboard into writable cells (those with blue background) by pressing <Ctrl-V>.

- **Status Bar**

  Displays the active filter expression (if present) and error status messages. In addition, the progress of the current request is shown. With the Stop button  , the current request can be canceled.

10.4 Buttons

- **Commit**

  Send a series of SET requests to the target agent. For each modified cell, the cell's value is combined with the cell's OID (resulting from the column OID + index OID). OID and value together build a variable binding. For each row that has been modified, the variable bindings representing the modifications are sent as a single SET PDU to the agent. Since SNMP assures that a SET request is atomic, either all variables will be successfully set to the new values or none of the variables will be modified. Set PDUs are sent row by row, starting with the row with the lowest index value.

  If a SET fails for a row, a dialog box will be shown that reports the error occurred and displays the row index of the failed row in its title. MIB Explorer will then continue to commit modifications made on

following rows. If the error index is zero, all modified cells in the affected row will get a red background. If the error index is not zero, the failed cell will be marked with red background. Failed rows will not be marked as successfully committed. Thus, by committing changes again, only the modified cells of failed rows and any newly modified cells will be sent to the agent.

- **Commit and Verify**

  Same as Commit, but sends a GET request to the agent after each SET request response it receives to verify the value of the modified cells after the SET. This is particularly useful in conjunction with row creation using the `RowStatus` textual convention, since values set for creation remain not the same after creation.

- **By Row**

  If committing changes with this option checked all writable columns of each changed row are sent to the target within a SET request per changed row. Otherwise, only the changed columns are SET.

- **Close**

  Close the table view and abandon any changes.

10.5 Scalars

The scalars table view displays all scalar MIB objects defined under the selected node in a two column list. The left column represents the object name of the corresponding scalar object and the right column contains the actual value of its instance.

To Open a Table View for Scalars:

1. Select an OBJECT-IDENTIFER node (no leaf) in the MIB tree that is the closest parent of the scalars you want to view or modify.
2. Choose Table from the node's context menu or from the Edit menu.

Modifications will be committed atomically by sending a single SET PDU. See “Table View” on page 64.

10.6 SNMP Tables

The SNMP table view displays all columnar MIB objects defined under the selected node and optionally from augmenting tables as well as other depending tables. The leftmost columns with gray background represent the index of the table. Columns with a white background are read-only.

To Open a Table View for SNMP Tables:

1. Select an OBJECT-TYPE node (no leaf) with a SYNTAX definition starting with "SEQUENCE OF". Typically, the object names of those nodes end with "Table".
2. Choose Table from the node's context menu or from the Edit menu.

If the table is augmented or extended by other tables of currently loaded MIB modules, a shuffle dialog will be displayed. You can add the tables with which you want to extend the master table from the left list to the right one. By pressing the OK button you add the columns of the tables in the right list to the table view. By pressing the Cancel button the table selected in the MIB tree is opened as it is.

10.6.1 Cells

MIB object instances are displayed (and edited) according to the corresponding MIB object's effective syntax:

- **OCTET STRING**

  If there is a DISPLAY-HINT format defined for the MIB object, the value has to be entered according to that format. For example, `DisplayString` values (format "`255a`") are entered as plain text.

  If there is no DISPLAY-HINT defined, the value is entered as hexadecimal string, for example: "`ab 8 2f 16`".

  For convenience, ASCII characters may be entered directly when they are enclosed by double quote characters (″). For example, if you enter '`"publi"1c`' this will result in the hex string '`70 75 62 6c 69 1c`'.

  Alternative display and edit formats for string objects can be set by the context menu available for each cell. See “Table View” on page 64.

- **OBJECT IDENTIFIER**

  Whether OIDs are displayed with last name or as numeric only OIDs depends on the preferences set. To edit an OID you may enter an object name and press `<Enter>` to convert the name into an OID. Then any instance suffix may be appended. For example, enter "`ifIndex`" and press `<Enter>` and the displayed value will become "`1.3.6.1.2.1.2.2.1.1`". Now append "`.2`" to reference the second interface. The resulting OID ("`1.3.6.1.2.1.2.2.1.1.2`") can then be saved into the cell by pressing `<Enter>` or `<Tab>` again.

- **Enumerated INTEGER**

  Select one of the enumerated values from the provided list or enter an integer value to set a value not defined in the MIB.

- **TimeTicks, Counter32, Counter64, Unsigned32, Integer 32, INTEGER**

  Enter a numeric value. You can use the spin box to browse through possible values. Range restrictions defined for the MIB object are not enforced by MIB Explorer, since the agent will reject out of range values.

  If there is a DISPLAY-HINT format defined for the MIB object, no spin box will be displayed and the value has to be entered according to that format.

- **BITS**

  The value may be entered as a series of '1' and '0' digits that are grouped in packets of eight bits separated by spaces. For example, to set the second, fourth, and thirteenth bits enter "01010000 00001" without quotes. Alternatively, BITS can also be entered by enumerating their names in the following form:

  ```
  { bitName1, bitName<n>, ... bitName<k> }
  ```

- **IpAddress, NetAddress**

  Either enter the IP address in raw IP address format (e.g. "192.168.0.1") or enter the host name (e.g. „www.mibexplorer.com") and press <Enter> to resolve it into a raw IP address.

Cells with white background are read-only cells. Cells with blue background can be modified. See “Set Dialog” on page 60.

11 Grid View

The grid view provides an intuitive structured visual representation of related SNMP tables. MIB Explorer recognizes relationship between tables according to their indexes. The following types of table relationships are recognized by MIB Explorer:

- **One-To-One Dependent Relationships**

  Two tables have a one-to-one relationship if one table augments the other. When a table augments another, it shares the same index columns with its base table and contains the same number of rows than its base table.

- **Sparse Dependent Relationship**

  A table is sparse dependent related to its base table, if it extends only some of the base table's rows. The dependent table then shares the index columns of the base, but in contrast to the one-to-one dependent relationship, the sparse dependent table does contain less rows than its parent. Typically, the sparse dependent table conceptually extends the base table with additional columns for those rows that have a specific attribute value.

- **Dependent Expansion Relationship**

  A table is dependent expansion related to its base table, if it shares the same index than its base but adds additional index columns to its index. An expansion table thus may contain more than one row for each row in its base table.

Reordering relationships where the index objects in two tables are the same, but in a different order cannot be viewed directly in a MIB Explorer grid view. Reordering relationships are often used to improve lookup performance for specific management tasks. In most cases it is not useful to view reordered tables in the same view.

To Open a Grid View for SNMP Tables

1. Select an OBJECT-TYPE node (no leaf) with a SYNTAX definition starting with "`SEQUENCE OF`". Typically, the object names of those nodes end with "`Table`".
2. Choose Grid from the node's context menu or from the Edit menu.

3. If the table is augmented or extended by other tables of currently loaded MIB modules, a shuffle dialog will be displayed. You can add the tables with which you want to extend the master table from the left list to the right one. By pressing the OK button you add the columns of the tables in the right list to the table view. By pressing the Cancel button the table selected in the MIB tree is opened as it is.

The SNMP grid view displays all columnar MIB objects defined under the selected node and optionally from augmenting tables as well as other depending tables.
The leftmost columns with gray background represent the index of the table. Columns with a white background are read-only. For each row in the base table, there may be zero or more rows of dependent tables. To view these rows, click on the Plus sign ('+') on the left side of the row or choose Expand Current from the context menu to expand all rows of the current table. The headers of the different tables will be shaded in different blue tones.

11.1 Editing and Cell Navigation

The editing model of the grid view is different from the model of the table view. While the table view allows undo and redo of editing steps, the grid view allows to cancel the changes made to the current row only.
If the value of a cell is changed and the focus leaves that row, the changes to the row are automatically committed to the target agent. Changes may also be committed manually by clicking on the edit symbol on the left side of the edited row.

In the Table View there is an option „By Row", which allows to commit all writable columns of a row too.

Even if only a single value has been changed in a row, all writable columns of that row will be set which is another difference to the table view's commit policy where only changed values are sent to the agent when committed.
The editing cell format equals to the format of the Table View as described by the section "Cells" on page 71, but with the following limitations:

- Enumerated values cannot be chosen from a list by using a combo box, only the numeric representation can be entered.
- OIDs can only be entered as integer values separated by dots, thus object names cannot be used to lookup OID values in grid views.
- BITS can only be entered by their binary representation.

11.2 Tool Bar

Figure 7: Grid View's tool bar.

- **Find**

  Find the next cell in the table whose value (as displayed) matches a given regular expression. The search starts at the current position and continues row by row from left to right.

- **Find Again**

  Find the next cell in the table whose value (as displayed) matches the last search expression.

- **Refresh**

  Refresh the whole table. The top level table is retrieved from the agent and then for each received row the dependent and augmented table rows are retrieved.

11.3 Context Menu

- **Insert Row**

  Insert a new row into a table of the selected type. Before the new row is inserted into the table, you will be prompted to specify the new row's index by entering a value for each index object. The presented input fields depend on the type of the index object. For a description of the different input fields see "Set Dialog" on page 60.
  If a row with the entered index already exists, an error dialog will be displayed and the insert operation will be aborted. Otherwise, the new row is inserted in a table with the selected type whose base table's index is lexicographic less than the index of the new row. If such a table does not exists yet, then it will be created.

- **String Format**

  Selects the display and input format for the selected/current column with OCTET STRING syntax. The available formats are described in "Column syntax formats of the PDU editor." on page 79.

- **Cancel**

  Cancel the changes made to the current row and restores the values of the row of the last refresh or successful commit.

- **Requery**

  Requery the specified rows by sending GETNEXT (SNMPv1) or GETBULK (SNMPv2c/v3) requests to the target agent.

- **Update**

  Update the agent by committing row changes to the target agent. A row is always completely updated which means that all writable elements are sent with a SET request message to the target.

*Tip: If the option "*Allow editing of objects with MAX-ACCESS read-only*" is selected in Preferences/ SNMP, then committing rows on regular SNMP entities will fail. Either use the table view instead or deselect that option.*

- **Select**

  Select the specified range of rows.

- **Expand Current**

  Expand all rows in the current table.

- **Print**

  Directly print the selected parts of the grid to the default printer.

- **Print Preview**

  Open a dialog to print the table on a printer. Before actually printing the table, the result can be previewed on the screen. The page layout and the printer can be selected.

12 Protocol Data Units (PDUs)

MIB Explorer provides means to easily compose SNMP PDUs with the PDUs tab from the info panel. A variable binding can be appended to a PDU by simply dragging the corresponding MIB tree node into the PDU editor table. PDUs created with the PDUs tab can be used for the following purposes:

- Monitoring an arbitrary set of MIB objects of the current target.
- Atomically modifying an arbitrary set of MIB objects of the current target.
- Sending any type of SNMP PDU to the one or more targets (see also Script and Monitor tools).
- Save retrieved object instances into a XML file for further processing or later reuse. See "To Save a PDU:" on page 77.

To Create a New PDU:

1. Select the PDUs tab from the main window's tools panel.
2. Press the New button to create a new PDU. If you want to create a trap, notification or INFORM PDU, then press the New Trap button instead. A new PDU will be created and displayed on the right hand side.
3. Edit the PDU.

To Open a PDU:

1. Select the PDUs tab from the main window's tools panel.
2. Press the Open button to open a previously saved PDU from disk. If the extension of the file is xml then it has to follow the XML schema `MIBExplorerPDU.xsd`. Otherwise, the file will be opened with MIB Explorer's internal PDU file format.
3. Select the PDU file in the file open dialog and press Open. The corresponding PDU will be displayed on the right hand side.

To Save a PDU:

1. Select the PDU to save in the PDUs tab.

2. Press the Save As button. A file selection dialog will be displayed. Select a file or enter a new filename to save the PDU to. To save the PDU as a XML file following the XML schema `MIBExplorerPDU.xsd`, use a file name with the extension `xml` (case insensitive). The `MIBExplorerPDU.xsd` located in the `xsd` directory of the MIB Explorer installation. For all other extensions, MIB Explorer's internal PDU file format is used.
3. Press OK. When the PDU is saved successfully, the change list will be cleared.

To Close a PDU:

1. Select the PDU to save in the PDUs tab.
2. Press Close. If there are any unsaved changes, you will be prompted for saving them.

12.1 Editing PDUs

The PDU editor can be used to create a SNMP PDU by specifying the variable bindings contained in the PDU. Each row in the editor represents a variable binding. The row positions correspond to the variable binding positions of the edited PDU. A variable binding is specified by supplying values for the following columns

1. Object ID

   Specifies the object identifier of the variable binding. This value must be given for each row.

2. Syntax

   Specifies the SMI syntax of the value supplied in the third column. If syntax Null is chosen, a value must not be specified.

3. Value

   The actual value of the variable binding. The format of the value depends on the chosen syntax in the second column. The formats are the same as for the Set Dialog.

To create correct trap and inform PDUs the first variable binding has to be a `sysUpTime.0` instance with a `TimeTicks` syntax and the second binding has to be a `snmpTrapOID.0` instance with an OID syntax. For SNMPv1, the required enterprise, `genericID`, and `specificID` fields are automatically determined from these two variable bindings if the target uses SNMPv1 and the PDU type is TRAP. If you want to specify the `agentIpAddress` field of the SNMPv1 trap, then include a

`snmpTrapAddress.0` (`1.3.6.1.6.3.18.1.3.0`) variable binding with syntax `IpAddress` as third variable binding in the trap PDU.

12.2 Context Menu

With the Unlock option of the PDU table's context menu, you can unlock the Object ID and Syntax columns of variable bindings that have been dropped from the MIB tree into the PDU. Unless such a variable binding has not been unlocked, you cannot edit its object ID and syntax.

In addition, the display and input format for variable bindings with OCTET STRING syntax can be chosen. The available formats are listed in the table below.

FORMAT	DESCRIPTION	SAMPLE VALUE DISPLAY OF „ABC“
ASCII	Plain text formatting.	`aBc`
Decimal	Formats each byte as a decimal value with a dot (".") as separator.	`97.66.99`
Hexadecimal	Formats each byte as a hexadecimal value with a colon (":") as separator.	`61:42:63`
Octal	Formats each byte as a octal value with a colon (":") as separator.	`141:102:143`
Binary	Formats each byte as a binary value with a colon (":") as separator.	`1100001:1000010:1100011`
MIB	Uses the format defined by the DISPLAY-HINT clause of the corresponding OBJECT-TYPE definition (if available) otherwise the Hexadecimal format will be used if non-printable characters occur.	*Depends on the MIB format!*

Table 7: Column syntax formats of the PDU editor.

12.3 Tool Bars

12.3.1 Main Tool Bar

Figure 8: PDU Editor's main tool bar.

- **Add New Variable Binding**

  Adds a new row representing a variable binding before the selected row. If no row is selected, the row will be added at the end of the PDU.

  - Duplicate Variable Binding

    Duplicate the selected row (variable binding).

  - Remove Variable Binding

    Remove the selected rows (variable bindings) from the PDU.

  - Move Variable Binding Up

    Move the selected variable binding(s) one position to the beginning of the PDU.

  - Move Variable Binding Down

    Move the selected variable binding(s) one position to the end of the PDU.

  - Set Values

    Sends the PDU's variable bindings as a single SET request PDU to the current target agent. If the target agent returns an error, then the corresponding error message and error index will be displayed in the status bar.

  - Send PDU to Targets

    Open a dialog for choosing one or more targets to send the variable bindings of the current PDU to.

  - PDU Type Selection

    Defines the PDU type to be used by any Refresh operation (manual or periodic) and defines the pre-selected PDU type for "Send PDU to Targets".

- **Send PDU to Current Target**

  Sends the current PDU as a PDU of the selected type (see "PDU Type Selection") request to the current agent. The values of the variable bindings in the sent request are set to the SNMP `Null` syntax for GET, GETNEXT and GETBULK PDU types. The values of each variable binding will be replaced with the value returned by the agent.

If the agent returns an exception value (SNMPv2c or SNMPv3 targets only) that corresponding value will be displayed with orange background. If the agent returns an error, the corresponding message and error index will be displayed in the status bar and no value will be changed.

- **Redo Last Change**

  Redo the last change made to the PDU (if there is any in the redo stack).

- **Undo Last Change**

  Undo the last change made. Changes made by PDU processing can be undone too, but only all changes of a PDU response at once.
  The undo stack is reset when the PDU is saved.

- **Detach PDU**

  Detach the PDU from the main window and show it in its own window.

- **Attach PDU**

  Attach a previously detached window to the main window again.

12.3.2 Periodic Refresh Tool Bar

Figure 9: PDU editor's periodic refresh tool bar.

- **Refresh Interval**

  Specifies the refresh interval in seconds. You can either use the predefined values or enter your own value. By pressing `<Enter>`, periodically refresh starts with the given refresh interval. `0` deactivates periodical refresh.

  - Start

    Starts periodically refresh, if the refresh interval value is not zero (or disabled).

  - Pause

Stops the periodical refresh.

- Progress Bar

  Shows the percentage of the refresh period left until next refresh.

- **Export Data**

  By pressing the toggle button, you can specify a comma separated value (CSV) text file to save the PDU data to whenever it is refreshed. The CSV file can be later opened by a spreadsheet application. As long as the toggle button is pressed, refreshed table data will be written or appended to the file. If the file already exists you can choose whether new data should replace existing data in the file or if the new data should be appended. If a file does not exist, then the new data will be appended. This is also the case, if the periodic refresh is enabled.

12.4 Sending a PDU

MIB Explorer is able to send any type of SNMP PDU to one or more targets at once. Because trap (SNMPv1 only), notification, and INFORM PDUs have a slightly different format than other SNMP PDUs, they always require two variable bindings which are as follows:

1. The first variable binding in a trap, notification or INFORM PDU has to be a `sysUpTime.0` instance (`1.3.6.1.2.1.1.3.0`) with syntax `TimeTicks`.
2. The second variable binding in a trap, notification or INFORM PDU has to be a
3. `snmpTrapOID.0` instance (`1.3.6.1.6.3.1.1.4.1.0`) with syntax OBJECT-IDENTIFIER.

To specify the agent IP address of a SNMPv1 trap PDU, you may add a `snmpTrapAddress.0` instance (`1.3.6.1.6.3.18.1.3.0`) as the third variable binding. If the third variable binding is not `snmpTrapAddress.0`, then the agent address will be set to "`0.0.0.0`".

To send a PDU

1. Create or open a PDU. If you want to send a trap, notification, or INFORM PDU, then create a new PDU using the New Trap button in the PDUs tab.
2. Add all variable bindings you want in the PDU to the variable bindings table. See “Editing PDUs” on page 78.

3. Press the Send PDU () button in the PDUs tool bar. A dialog window will be shown, where you can specify the targets to send the PDU to.
4. Add those targets to the Selected Targets panel on the right side. The current target will be added to the selected targets by default.
5. Select the PDU Type. If the PDU does not contain the two required variable bindings for TRAP and INFORM PDUs listed above, then the GET, SET, GETNEXT, and GETBULK PDU types will be available only.
6. If you have chosen GETBULK as PDU type, specify the number of non-repeaters in the PDU and the maximum repetitions for the repeatedly requested variable bindings.
7. Press OK to send the PDU to the selected targets.

Note: If you select GETBULK for a SNMPv1 target, then MIB Explorer will send a GETNEXT request instead.

After pressing OK, MIB Explorer will display a Result Window. It contains a table, which displays for each target the status of the respective SNMP request. If a request is successfully finished, the status column will be "Success". Since a TRAP request is not responded, its status will be "Success" when it has been sent successfully. This does not imply that the target has received it! For all other PDU types where the request has not timed out, you can click on the row in the status table to show the response PDU of the respective target.
The Response PDU table displays the variable bindings of the selected target's response (if any response is available) in the same manner as described for the Browse Tab.
If a target responded with an error, the error status will be displayed in the Status column. Is the error index greater than zero, then the corresponding variable binding will be displayed with a red background.
Variable bindings with OIDs that are not known (cannot be resolved with the currently loaded MIB modules) are displayed with orange background.

13 Trap Receiver

The trap receiver pane displays SNMPv1 traps, SNMPv2c/v3 notifications, and INFORM messages received on specified trap listener addresses. UDP, TCP, TLS and DTLS trap listener addresses and ports are configured using the Trap Receiver section of the Preferences. See "Trap Receiver" on page 11.

The background color of the Notification ID field of a trap, notification, or inform message reflects the severity assigned for the object ID subtree the notification ID belongs to:

- Red indicates a message with a severity of FATAL,
- Orange indicates an ERROR, and
- Yellow indicates a WARNING.

SNMPv3 traps and INFORMS are received on behalf of principals which can be configured by setting up a target.

To setup SNMPv3 INFORM request reception:

In contrast to SNMPv3 trap reception or sending of SNMPv3 requests, MIB Explorer is the authoritative entity when receiving SNMPv3 INFORM requests. To enable reception of such requests on behalf an USM user, one has to configure an user as follows:

- The Localization Engine ID (if specified at all) has to be set to engine ID of MIB Explorer which can be found in SNMPv3 section of the Preferences dialog.
- The Principal check box has to be selected for that USM user.

13.1 Tool Bars

Figure 10: Trap Receiver's main tool bar.

- **Acknowledge**

  Moves selected traps from the list of new traps to the notification history. The notification history contains acknowledged traps for further reference.

- **Save As**

Saves the selected trap as a MIB Explorer PDU file. This PDU file can then be loaded and modified with the PDU editor or used in MIB Explorer Scripts.

- **Delete**

  Deletes the selected traps/notifications from the list of new traps or the notification history respectively.

- **Redo**

  Redo last change.

- **Undo**

  Undo last change. You can undo trap deletions and acknowledgments.

- **Properties**

  Open the Trap Severities dialog for specifying logging severities for incoming traps, notifications, and inform messages. In addition, scripts can be specified for each notification category to be executed when receiving a notification whose ID matches that category.

- **Notification History**

  Toggles the display between list of new traps and notification history. If selected (dark background), the notification history is shown, otherwise the list of new traps is displayed.

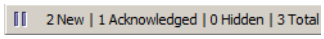


Figure 11: Trap Receiver's acknowledgment tool bar.

- **Suspend Trap Table Update**

  Suspend the update of the trap table. Press this button to save processor time or to stabilize the trap list while MIB Explorer is receiving a lot of traps. The new traps received while the Pause button is selected will be counted by the Hidden traps counter (see below). By reactivating the trap table update, any hidden traps will be inserted into the table according to the current sort settings.

Note: If the trap receiver receives more than given number of traps per second, MIB Explorer will automatically suspend update of the trap table. Suspension has to be disabled manually. The number of traps per second to be tolerated by the auto-inhibition can be configured in Preferences.

- **New**

  The total number of traps received that have not yet been acknowledged.

- **Acknowledged**

  The total number of traps in the notification history list.

- **Hidden**

  The total number of traps received while the trap table update is suspended.

- **Total**

  The total number of traps, new traps and acknowledged traps, available in the trap receiver.

13.2 Traps/Notifications Table

The traps table displays new and acknowledged traps/notifications depending on the status of the Notification History toggle button. Each row represents an event. The row label indicates the time and date when the event has been received. The other columns are as follows:

- **Notification ID**

  The notification ID is the OID identifying the event. Although, SNMPv1 traps are not identified by OIDs, but identified by a combination of an OID and an integer value, SNMPv1 traps can be easily mapped to an unique OID.
  The "generic traps" are mapped to the notification IDs defined under the `snmpTraps` (`1.3.6.1.6.3.1.1.5`) node. The "specific traps" are mapped to an OID value by using the `ENTERPRISE` OID value of the trap, adding a zero OID sub-identifier, and adding a final sub-identifier value corresponding to the specific trap value. See RFC 2576 for details on this mapping.

The background color of the Notification ID field reflects the assigned severity for this notification ID:

```
FATAL, ERROR, WARN, INFO
```

- **Originator**

  The originator address denotes the IP address (host name) and UDP port of the SNMP entity that sent the event.

- **Destination**

  The destination address value denotes trap listener address on whose behalf the trap has been received.

- **System Up Time**

  The system up time value represents the up time of the system (e.g. SNMP agent) that sent the trap.

- **Security Name**

  The security name denotes the security user on whose behalf the event was sent (SNMPv3) or the community of the trap/notification (SNMPv1 and SNMPv2c).

- **Version**

  The SNMP version of the event message.

- **Context**

  For SNMPv3 events, the context value denotes the event generating subsystem.

- **Context Engine ID**

  For SNMPv3 events, the context engine ID value denotes the engine ID of the subsystem that sent the event.

Note: If there is no UDP port specified, then the displayed address is not the originator's UDP transport address, but the IP address value of the agent address field of the SNMPv1 trap received.

By clicking a row, the variable bindings contained in an event message are displayed in the below described Trap Payload Table.

13.3 Traps Payload Table

The trap payload table displays the variable bindings contained in the first selected trap (event) message of the Traps Table. The variable bindings are displayed in the same manner as described for the Browse Tab. See "Browse Tab" on page 56.

13.4 Trap Severity Editor

The Trap Severities Editor dialog that lets you specify (logging) severities for categories of incoming traps, notifications, and inform messages. The severity is determined by analysis of the notification ID. For each incoming trap/notification, the Trap Severities table will be searched for the entry (category) whose subtree OID is the longest possible match. The severity for this message will then be set to the severity specified for the matched category.

If there has been assigned a MIB Explorer Script for the matched category, then the corresponding script will be executed with the snmp, utils, and mib contexts and additionally the following special context values:

CONTEXT	DESCRIPTION
severity	The assigned severity for the received notification as one of the following strings: `FATAL`, `ERROR`, `WARN`, and `INFO`.
comment	The comment string assigned to the category the received notification matches or `null` if the comment is left empty.
sourceAddress	The complete source address of the notification.
sourceHost	The host (IP address) of the notification source.
sourcePort	The UDP or TCP port of the notification source.

Table 8: Special context values for MIB Explorer Scripts triggered by traps.

To Open the Trap Severities Editor

1. Choose Trap Receiver from the Tools menu.
2. Click on the Properties () button.
3. Add or remove categories by either using the Add or Remove buttons respectively or alternatively using the context menu of the shown table.
4. Press OK to save your changes.

To Configure a Script for a Notification Category

1. Select the category row by clicking on the row's Script column cell.
2. Open the context menu by pressing the right mouse button.
3. Choose Script... and choose or enter the file name of the script to run for notifications of this category.
4. Press OK to save your changes to the category.

There is an example for sending an email when receiving a trap in the `examples` directory of the MIB Explorer installation directory named `email_on_trap.vm`.

14 Scripts

MIB Explorer can execute scripts based on VTL, the *Velocity Template Language* from Apache. Also VTL, in the first place, is designed for generating text or HTML output with dynamic content, its clear differentiation between model, view, and controller (MVC) makes it also a first choice for scripting.

The control structures provided by VTL are limited, but they also make it easy for users with little or no programming experience to write scripts based on VTL. Supported control statements are `#if..#else..#end` to conditionally execute statements and `#foreach..#end` to execute statements for each element of a given list. With the `#macro` statement parts of script that are frequently used can be combined into a function that can called in the script with `#<macro_name>`. Please refer to the VTL user guide for a complete description of the VTL language.

A MIB Explorer script differs from any other Velocity Macro (VM) only by the *model* MIB Explorer provides for the script. The model is accessed from a VM through *contexts*. To access the methods (services) provided by a context, a VTL reference with the context's name is used, for example

```
$utils.sleep(1000)
```

will cause the script to cease execution for one second by calling the `sleep` method of context `utils`. The contexts supported by MIB Explorer are shown in the table below. The Scope column indicates in which type of scripts the corresponding context can be used. The methods implemented

by the contexts are documented in the JavaDoc HTML documentation also available in the `api` subdirectory of the `doc` directory in the MIB Explorer installation folder:

CONTEXT	SCOPE	DESCRIPTION
`snmp`	any (except monitor-data-servlet)	The `snmp` context provides services to create SNMP request, load PDUs, modify their variable bindings, send SNMP requests and notifications synchronously and inspect SNMP responses. The context has several internal members which can be accessed through this interface: ▸ Current Target - The target to be used for subsequent requests. ▸ Default Target - The target set for MIB Explorer. ▸ Request PDU - The PDU to be used for subsequent SNMP operations. ▸ Response PDU - The PDU received from the last request operation. ▸ PDU directory - The directory the search for PDU files, if their file name is relative.
`smi` ~~`mib`~~	any (except monitor-data-servlet)	The `mib` context provides services to ▸ load/unload MIB modules, ▸ retrieve and exploit MIB object definitions, and ▸ manipulating OID values.
`utils`	any	The `utils` context provides utility services to ▸ temporarily execute a system command, ▸ send email using (authenticated) SMTP, ▸ cease script execution, ▸ stop script execution, ▸ create random integers, and to ▸ create an empty `Vector` (e.g., to collect response PDUs).

Table 9: Velocity Macro (VM) language contexts available for MIB Explorer scripts.

CONTEXT	SCOPE	DESCRIPTION
`gui`	script-tool	The `gui` context provides means to get input from the user when the script is executed. The user can be prompted for a string value, or for a variable binding. *Note: This context is only available for scripts executed on behalf of the script tool pane of the MIB Explorer GUI.*
`alarm`	monitor-alarm-script	The `alarm` context provides information about the alarm event that triggered the script's execution. *Note: This context is exclusively available for scripts executed on behalf of a monitor's alarm configuration.*

Table 9: Velocity Macro (VM) language contexts available for MIB Explorer scripts.

14.1 Script Editor

The script editor is divided into three areas: the Tool Bar area, the Script area, and the Output area. The Script tab is used to edit the script whereas the Output tab shows the script's output from its last run.

14.1.1 Tool Bars

Figure 12: Script tab's main tool bar.

- **Run**

  Executes the script once. The Output tab is selected and the script's output is shown as it is generated.

- **Stop/Abort**

  Stop the execution of the script.

- **Redo**

  Redo last change made to the script.

- **Undo**

  Undo last change made to the script.

- **Detach**

  Detach this script panel from the main window and show it in its own window.

- **Attach**

  Attach a previously detached window to the main window again.

Figure 13: Script refresh tool bar.

- **Run Interval**

  Specifies the run interval in seconds. You can either use the predefined values or enter your own value. By pressing `<Enter>`, periodically script execution starts with the given interval. `0` deactivates periodical refresh.

  During periodically script execution, the script is scheduled for repeated *fixed-delay execution*. Subsequent executions take place at approximately regular intervals, separated by the specified period.

  In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as garbage collection or other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period.

- **Start**

  Start periodically script execution, if the interval value is not zero (not "disabled").

- **Stop**

  Stop the periodical script execution.

- **Progress Bar**

  Shows the percentage of the execution pause period left until the next scheduled execution.

- **Export Output**

  By pressing the toggle button, a text file can be specified to store the script's output. While the export toggle button is selected the script's output is saved to the specified file whenever it is updated.

15 TFTP

The Trivial File Transfer Protocol (TFTP) as specified in RFC 1350 provides means to transfer files over an IP network with very little protocol overhead. Files can be transferred to a remote TFTP server (PUT) or retrieved from a remote server to a local file (GET).

MIB Explorer can provide a TFTP server and client which are described in the following sections.

15.1 TFTP Client

To put a file on a TFTP server

1. Select the TFTP tab from the main window's Tools panel.
2. Select/configure the target which represents IP address of the remote TFTP server.
3. Choose the local file to be transferred by pressing the Choose button. You could also drag&drop a file from a file explorer to the local file text field.
4. Enter the remote file name or drag&drop a MIB instance containing a file name string from the MIB tree.
5. Press the Put button to transfer the local file to the specified remote file using binary transfer mode. The bytes transferred so far as well as encountered errors are displayed in the status bar.

To get a file from a TFTP server

1. Select the TFTP tab from the main window's Tools panel.
2. Select/configure the target which represents IP address of the remote TFTP server.
3. Choose the local file for the transferred data by pressing the Choose button. You could also drag&drop a file from a file explorer to the local file text field. Please note that the local file is overwritten by pressing the Get button in step 5.
4. Enter the remote file name or drag&drop a MIB instance containing a file name string from the MIB tree.

5. Press the Get button to transfer the remote file to the specified local file which will be overwritten. The bytes transferred so far as well as encountered errors are displayed in the Status Bar.

15.2 TFTP Server

A TFTP server is a non-secure (no password required to login) trivial file server. *It should not be run unattended*. It can be used to transfer BULK data such as system dumps from or to a (SNMP) device.

To start a TFTP server

1. Select the TFTP tab from the main window's Tools panel.
2. Use Choose from the TFTP Server area of the TFTP pane to specify a directory that contains the file(s) to be served by the TFTP server. Files in the specified directory and below.
3. Specify the UDP TFTP Port that should be used by the TFTP server. The default port is 69.
4. Press the Start button to start the TFTP server.

To stop a TFTP server

1. Select the TFTP tab from the main window's tools panel.
2. Press the Stop button in the TFTP Server area of the TFTP pane to stop the TFTP server.

16 Snapshots

Snapshots can be taken either from the MIB Tree or from the Browse tab and then stored to a file. A snapshot file contains the MIB object instances (values) that were in the MIB Tree or Browse panel when the snapshot has been taken. A snapshot can be loaded from file at any time thereafter and displayed within a snapshot browser dialog. In addition, snapshots can be compared to identify differences between snapshots.

16.1 Use Cases

Snapshots can be used inter alia to:

- Save the state of an agent or of a sub-tree of the same for future reference.

  How-To: Browse the `iso` sub-tree of the agent and create a snapshot file from it. Then you can load and browse the snapshot (file) at any later time.

- Save the state of an agent or of a sub-tree of the same to restore the values at a later time.

  How-To: Browse the `iso` sub-tree of the agent and create a snapshot file from it. At a later time re-select the target agent's target and then load the snapshot (file). By using the Restore function of the Snapshot Browser, the values can be set on the target agent.

- Compare the state of two agents or of two states of the same agent at different points in time.

  How-To: Browse the `iso` sub-tree of the agent and create a snapshot file from it for both agents or at two different points in time respectively. Then compare the two snapshot files by using the Snapshot Comparison Wizard.

- Copy the values of one agent or any sub-tree of its MIB to another agent.

  How-To: Select the source agent as target and browse the `iso` sub-tree of that agent. Create a snapshot file from the Browse panel. Then load select the target agent's target and open the snapshot. By using the Restore function of the Snapshot Browser, the values can be set on the target agent.

16.2 Snapshot Operations

To Create a Snapshot

1. Select the target from which you want to create the snapshot.
2. Select the sub-tree you want to create a snapshot from in the MIB tree. To create a snapshot from the complete MIB of an agent, select the `iso` node.
3. Either use Browse or Get from the selected node's context menu to retrieve the instances in the selected sub-tree. Using Get is slower than Browse if the sub-tree contains many instances. However, the first allows to retrieve several sub-trees one after the other so that the resulting snapshot contains the union set of the retrieved sub-trees.
4. Choose File>Snapshots>Save from Tree if you used Get to retrieve the values or File>Snapshots>Save from Browse tab if you used Browse. Select a file or specify a new file name and press Save to save the snapshot file.

To Open a Snapshot

1. Choose File>Snapshots>Open from the main menu and select the snapshot file to open.
2. Press Open to load and display the snapshot in a Snapshot Browser window.

To Compare Two Snapshots

1. Choose File>Snapshots>Compare from the main menu.
2. The Snapshot Comparison Wizard is displayed. It has three steps. The first two steps are required and specify the snapshots to be compared. The third (optional) step specifies which syntaxes should be compared and whether only values that are present in both snapshots should be compared.

16.3 Snapshot Browser

A Snapshot Browser is divided into a tool bar and a tree-table area. The tree-table has four columns:

1. MIB Tree

   The MIB tree column contains the MIB object identifiers of the MIB objects and instances in the snapshot. By default the MIB tree column display objects and instances as an expanded tree. By clicking on the

button in the upper left corner of the table's header, the tree is flattened and the objects and instances in the snapshot are displayed as a table.

2. Syntax

   If the MIB object in a row is an OBJECT-TYPE definition or instance, then its syntax is displayed in this column. The syntax of MIB OBJECT-TYPE definitions is displayed as defined in the corresponding MIB module, whereas the syntax of MIB object instances is displayed as received from the agent.

3. Value

   The value of an object instance. The value is displayed by using the DISPLAY-HINT specification applicable for that object definition, if such a definition is available in the currently loaded set of MIB modules.

4. Status

   The status column shows the operation status for Restore or Refresh operations per value instance. The status values `Restored` and `Refreshed` indicate operation success, whereas `Not Restored` and `Not Refreshed` indicate that the restore or refresh operation did not succeed on the particular instance, because there was an error on another instance which has been sent in the same PDU. For restore operations, values are sent within the same PDU, when they are part of the same row. For refresh operations, up to the maximum number of variable bindings allowed per PDU are sent within one PDU. An error status is indicated by a lower letter starting error status description.

16.3.1 Tool Bar

- **Refresh**

  Refresh the values in the snapshot from the current target. The target that will be used for any operations in a snapshot browser window is displayed in the window's title. While the refresh operation is performed, a progress bar is being displayed. After completion, a message box is shown and the status column contains detail information about the operation success for each value instance.

- **Save**

  Saves the content of the snapshot browser to a snapshot file. With this function snapshot files can be copied or kept up-to-date with the latest agent state (in conjunction with the Refresh operation).

- **Restore**

  The restore operation sends the values in the snapshot as SET requests to the current target (as displayed in the title). In other words, the restore operation assigns the values in the snapshot to the corresponding MIB object instances of the target agent. By pressing the Restore button an option dialog is shown which allows to specify whether

  Note: This option is only available from the Snapshot Comparison Browser.

  - only mismatched instances are restored;
  - rows with a `RowStatus` column should be set into `notInService(2)` before updating them or set to `createAndWait(5)` before creating them;
  - only those objects are restored whose OBJECT-TYPE definition (which has to be present in the loaded MIB modules) has a MAX-ACCESS value of `read-write` or `read-create`.

By pressing OK, the values that match the selected criteria are set in the target agent. Each scalar value will be set by itself, which means that each scalar variable binding instance is sent in its own PDU to the target. Columnar object instances (if the corresponding table definition is contained in the loaded MIB modules) will be sent row-by-row to the agent.
If the second option (use `RowStatus` states) is selected rows will be created in the target agent by setting the `RowStatus` column to `createAndWait(5)` and then sending a SET PDU with all matched values for that row. Then the status of the `RowStatus` column is set to `active(1)` or it is left unchanged if the status is not specified by the snapshot or it is specified as any other value than `active(1)`.

Note: If the tool bar is part of the Snapshot Comparison Browser *and the second option is active and the values restored do not contain a row that is contained in the opposite snapshot, then the row is deleted from the target by sending a SET request on the corresponding* `RowStatus` *column with a value of* `destroy(6)`.

During the restore operation a progress bar is shown. After completion, a message box is displayed and the status column contains detail information about the operation success for each value instance.

16.4 Snapshot Comparison Browser

The Snapshot Comparison Browser is similar to the Snapshot Browser described above, except that it contains a second tool bar for the second snapshot and the following three additional columns:

- *Difference*

  The Difference column shows the comparison result for each value pair. Possible values are:

  -  Equal Sign

    The equal sign indicates that both values have been compared, have the same syntax, and that they equal each other.

  -  Less Than (Yellow)

    The yellow less-than-sign indicates that both values have the same syntax, have been compared, and the left value is less than the right value.

  -  Greater Than (Yellow)

    The yellow greater-than-sign indicates that both values have been compared, have the same syntax, and the left value is greater than the right value.

  -  Less Than (Red)

    The red less-than-sign indicates that only the right snapshot contains a value for this instance.

  -  Greater Than (Red)

    The red greater-than-sign indicates that only the left snapshot contains a value for this instance.

  -  Error

    The stop/error sign indicates that the values of the both snapshots have different syntax and therefore cannot be compared.

- **Right Syntax**

  The syntax of the right snapshot's value.

- **Right Value**

  The right snapshot's value.

17 Monitors

The Monitor tool is used to collect numerical and other values from one or more targets and store them in a round robin database. The collected numerical values can be recomputed based on spreadsheet like formulae. The collected and computed values can then be displayed graphically as 2D charts or 3D charts.

Data is collected on demand or periodically in the, so called, primary round robin database with configurable number of samples, start time, and refresh interval. Each monitor has exactly one primary round robin database.

The collected data can be consolidated based on consolidation functions like AVERAGE, MAX, MIN, LAST, and SUM into secondary round robin databases. A monitor can be configured to have an arbitrary number of such consolidation round robin archives.

For each primary data point, regardless whether it is a collected or a computed one, an alarm configuration can be specified. Each alarm configuration consists of a boolean expression which raises the alarm and optionally another boolean expression which clears the alarm. Alarms are logged in the MIB Explorer logging system with a configurable severity. In addition, MIB Explorer may be configured to execute scripts to generate traps, notifications, and informs, as wells as to send emails or execute system commands if one of the configured raise or clear alarm conditions becomes true.

Primary and secondary round robin databases as well as the corresponding charts may be exported to XLS and CSV files (tabular data) or JPEG, PNG, PDF, PS, and PCL graphic files (charts) whenever data is updated periodically. The monitor configuration and data can be automatically saved as XML or an internal binary file whenever its data has been updated.

Monitor examples can found in the examples/monitor *directory of the MIB Explorer installation.*

17.1 Basic Monitoring Operations

MIB Explorer provides means to easily create Monitors with the Monitors tab from the info panel. Monitors can be used for the following purposes:

- Collecting numerical SNMP data over a long time period and optionally compute additional values based on the collected data.

- Displaying collected or computed data as plot, bar, area, and pie charts. Alternatively, collected data can be also displayed as 3D bar, surface, and scatter plot charts.
- Exporting data and charts to XLS, CSV and JPEG, PNG (as well as 2D charts to PDF, PS, and PCL) files.
- Generating SNMP traps/ informs and log entries on configurable alarm conditions.
- Run MIB Explorer scripts to spawn external processes, send emails, or send SNMP requests.

To Create a New Monitor

1. Select the Monitors tab from the main window's Tools panel.
2. Press the New button to create a new Monitor. If you want to create a monitor with 3D chart display like 3D bars, surface, or scatter plot, then press the New 3D button instead. A new Monitor will be created and displayed on the right hand side.
3. Configure the Monitor

To Open a Monitor

1. Select the Monitors tab from the main window's Tools panel.
2. Press the Open button to open a previously saved Monitor from disk.
3. Select the PDU file in the file open dialog and press Open. The corresponding Monitor will be displayed on the right hand side.

To Save a Monitor

1. Select the Monitor to save in the Monitors tab.
2. Press the Save or Save As button. A file selection dialog will be displayed in the latter case or if the monitor has not been saved yet or the file is no longer accessible.
   Select a file or enter a new filename to save the Monitor to. The monitor will be saved in XML format if the file name's suffix is "`.xml`" or "`.XML`". Otherwise the internal binary format will be used.
   The XML schema of the monitor configuration file can be found in the `xsd` directory of the MIB Explorer installation.
3. Press OK. When the monitor is saved successfully, the change list will be cleared.

To Close a Monitor

1. Select the Monitor to save in the Monitors tab.
2. Press Close. If there are any unsaved changes, you will be prompted for saving them.

To Convert a Monitor from 2D to 3D Charts or Vice Versa

1. Select Monitors tab from the main window's Tools panel.
2. Press the Convert... button to open an existing monitor which will then be automatically converted to from 2D to 3D or from 3D to 2D charts depending on the type of the opened monitor.
3. The converted monitor will be displayed on the right hand side. Check the properties of the monitor in order to adjust any settings.

To Restart Open Monitors When MIB Explorer is Run Next Time

1. Select Monitors tab from the main window's Tools panel.
2. Open all monitors that should be run the next time MIB Explorer or MIB Explorer Server is started following the instructions above.
3. Select the "Restart Next Time" checkbox.
4. Exit MIB Explorer at any later time without opening or closing monitors meanwhile.
5. If MIB Explorer Server is run then, the monitors will be started from the location where they were opened under (2). If MIB Explorer Pro is started, then you will be prompted at startup whether you want to run the monitors. If you confirm, the monitors are loaded and started otherwise they are loaded only.

17.2 Monitor Configuration

The monitor panel is divided into three regions: the tool bars at the top, the view navigation tree on the left, and the view itself on the right (configuration, table, or chart).

To configure a Monitor

1. Create a new monitor as described here or open one as described here.
2. Select the target you want to monitor and make sure that all MIB objects (modules) you want to monitor are loaded.

3. Select the Configuration node from the navigation tree of the monitor panel.
4. For each object to be monitored, select it in the tree (preferably the corresponding instance) and drag it into the configuration table. A new row will be created with the name of the MIB object. In case you have dragged an OBJECT-TYPE definition, its OID will be filled into the OID/Formula column of the new series. If the dragged object is an instance of an OBJECT-TYPE, then the Index column will be filled with the index portion of the instance's OID.
5. Add additional rows to the table if you need to compute any values from the collected ones above. Enter a formula beginning with an equals sign ("=") into the OID/Formula column. After the equals sign, enter an arithmetic expression.
6. Choose the properties button () from the tool bar. From the properties dialog choose the Data tab and configure data coverage, update interval, and number of primary samples. Save your changes by pressing OK.

The above steps are sufficient to setup basic monitors. However, in many situations you may need to use some of the advanced monitoring features:

- Index Calculation
- Customized Legend
- Alarm Configuration

How to configure these features is described in the following sub-sections.

17.2.1 Monitoring Series Configuration Matrix

The Configuration node of the Monitor navigation pane displays the monitoring series configuration matrix on the content pane. This table

specifies which data is monitored and which series of the monitored data is displayed in the associated charts. The table below describes the columns of the series configuration matrix.

COLUMN	DESCRIPTION
Name	The name of the series. If the Legend column is empty, then the series name will be used in the chart legend instead.
Target	The target from which data is retrieved for this series. If the target field is empty, then data is retrieved from the current target configured for MIB Explorer.
OID/Formula	The OID of the OBJECT-TYPE definition (without any index portion) that defines the data to be collected. The OID may be specified as dotted string of positive integers, for example `1.3.6.1.2.1.2.2.1.10`, or as a combination of object name and a numerical dotted string, for example `ifInOctets` or `ifTable.1.10`. Instead of collecting the series from a target it may also be computed based on other values in the primary round robin database by specifying a formula instead of an OID. A formula begins with an equals sign ("="). After the equals sign, an arithmetic expression has to be entered. The evaluation of formulas is done for a row left to right when all "OID" data points have been retrieved (even if an error occurred).
Index	The index portion of the OID that identifies the OBJECT-TYPE instance to retrieve data from, for example `1` (index has `Integer32` syntax) or `127.0.0.1` (index has `IpAddress` syntax). In addition to simple OID string any type of OID string expression can be entered as formula (thus, the leading equals sign ("=") is required), for example: `="1"+indexof(ifName, "eth0.*").` If the OID/Formula column contains a formula, then this column will be ignored.
Display	Specifies whether this series is displayed in chart views. Series with data retrieved from OBJECT-TYPES with a syntax other than `Integer`, `Counter`, `TimeTicks`, `Gauge`, and `Counter64` must not be displayed. Such series may only be used to reference them from chart name or legend templates.

Table 10: Column description of the monitoring series configuration matrix.

COLUMN	DESCRIPTION
Legend	Specifies the text to be displayed for this series in the chart's legend. The text may include an arbitrary number of formatted arithmetic expressions enclosed in brackets { and }. For example, "Foo Average {FORMAT(AVERAGE(Z?:Z0), "#,##0")}" prints the average over all data points in the series Foo as Foo's legend.
Alarm	The alarm column indicates whether alarm checking is enabled, disabled, or not present (no alarm) for this series. To configure an alarm for this series, select the corresponding row by clicking on the row's label and then pressing the right-mouse button to open the context menu. There choose Configure Alarm.

Table 10: Column description of the monitoring series configuration matrix.

17.3 Monitor Alarms

It is often useful to generate an alarm when a value crosses a certain boundary, for example if the free disk space is lower than a predefined value or percentage. MIB Explorer provides means to raise and clear an alarm dependent on respective boolean expressions for each series of a monitor configuration. An alarm can be simply logged or a script can be executed, which may send a trap/inform SNMP message when the alarm is triggered.

To Configure an Alarm

1. Within the monitor configuration table, select the row (series) that should be watched for an alarm condition by clicking on the row label.
2. Press the right mouse button to open the context menu.
3. Select Configure Alarm to open the Alarm Configuration dialog, which is described below.
4. Select Log Severity and enter at least a Raise Condition.
5. Press OK to save your changes. The alarm is then enabled.

To Disable an Alarm Watch

1. Select a row (series) in the monitor configuration with an enabled alarm status.
2. Press the right mouse button to open the context menu.
3. Select Disable to disable the alarm watch. No alarms will be generated for this series until it is enabled again.

To Enable an Alarm Watch

1. Select a row (series) in the monitor configuration with an disabled alarm status.
2. Press the right mouse button to open the context menu and select Enable to enable the alarm watch. From now on, alarms will be generated for this series if the corresponding conditions are met.

To Remove an Alarm

1. Select a row (series) in the monitor configuration with a disabled or enabled alarm status.
2. Press the right mouse button to open the context menu and select Remove to remove the alarm watch.

17.3.1 Alarm Configuration Dialog

- **Target**

  The alarm target for scripts executed on behalf of this alarm. This field has no effect, if Type is set to Log Only. The special target "`MIB Explorer`" sends alarms to the MIB Explorer's trap receiver using UDP transport. The alarm target which will be configured when "`MIB Explorer`" is selected, is the first entry in the list of trap listener addresses/ports.

- **Severity**

  Specifies the log priority (severity) to be used for logging events generated on behalf of this alarm. This is also the severity returned within the alarm context of alarm scripts.

- **Raise**

  - Condition

    The boolean expression which raises the alarm when true. Unless a Clear Condition is also provided, the alarm is transient. Thus, it will be raised whenever the raise condition evaluates to true. When a Clear Condition is given, the alarm will be raised only once. Before it could be raised again, the clear condition has to be true at least once.

  - Script

    Specifies a MIB Explorer script to be executed when this alarm is raised. Besides the normal script contexts `snmp`, `mib`, and `utils`

there is also the `alarm` context supported, which provides an interface to access alarm and monitor data from within the script.
There are two example alarm scripts available from the examples directory: `alarm.vm` and `alarm_email.vm`. The first can be used to send a SNMP notification when an alarm is raised or cleared whereas the second can be used to send an email using SMTP (authenticated or not). See the MIB Explorer Script API documentation for details.

Note: The gui context is not supported in alarm scripts!

If the Embed Script check box is selected, the supplied script file will be copied into the alarm configuration. This is particularly useful when a monitor should be run on a server or distributed to other MIB Explorer instances, because one need not to deploy the script file separately. A disadvantage of using embedded scripts is the fact that embedded scripts cannot be changed by changing a single file per MIB Explorer instance.

With Edit, an embedded script can be viewed and changed.

To test a script, click on the Test button and the supplied script will be executed as if the given condition would be true. The values presented to the script are taken from the most recent row of the primary values table.

- **Clear**
  - Condition

    Specifies the boolean expression which clears the alarm when true. By providing a clear condition an alarm is non-transient. Clear Condition and Raise Condition have to be mutually exclusive.

  - Script

    Specifies a MIB Explorer script to be executed when this alarm is cleared. In most cases, this will be the same script as for the Raise Condition. Besides the normal script contexts `snmp`, `mib`, and `utils` there is also the `alarm` context supported, which provides an interface to access alarm and monitor data from within the script.
    See the MIB Explorer Script API documentation for details.

    Note: The gui context is not supported in alarm scripts!

    If the Embed Script check box is selected, the supplied script file will be copied into the alarm configuration. This is particularly useful when a monitor should be run on a server or distributed to other MIB Explorer instances, because one need not to deploy the script file separately. A disadvantage of using embedded scripts is the fact that embedded scripts cannot be changed by changing a single file per MIB Explorer instance.

    With Edit, an embedded script can be viewed and changed.

To test a script, click on the Test button and the supplied script will be executed as if the given condition would be true. The values presented to the script are taken from the most recent row of the primary values table.

- **Value**

  Specifies the alarm text or alarm value that provides further information about the alarm. The alarm text may include formatted arithmetic expressions placed between brackets `{` and `}`. The brackets will not be part of the output text but replaced by the result of the entered expression. For example, an alarm text value

  ```
  "This is the series average: {AVERAGE(Z?:Z0)}"
  ```

  will result in the following text if the series average is `100.0`:

  ```
  "This is the series average: 100.0"
  ```

17.3.2 Monitor Chart Types

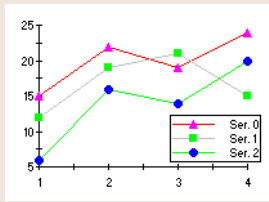
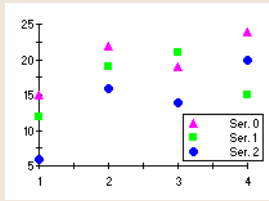
CHART TYPE	DESCRIPTION
PLOT 	Each series is drawn as connected points of data. ▸ X-axis annotated using time labels. ▸ Series appearance determined by line style and symbol style.
SCATTERED PLOT 	Each series is drawn as unconnected points of data. ▸ X-axis annotated using time labels. ▸ Series appearance determined by symbol style.

Table 11: Monitor 2D chart types.

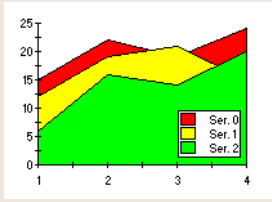
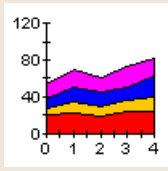
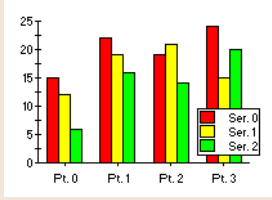
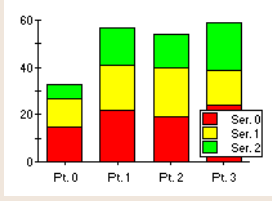
CHART TYPE	DESCRIPTION
AREA 	Each series is drawn as a bar in a cluster. The number of clusters is the number of points in the data. Each cluster displays the *n*-th point in each series. ▸ X-axis annotated using time labels. ▸ Series appearance determined by line style color
STACKING AREA 	Each series is drawn as connected points of data, filled below the points. Eeach series is placed on top of the last one to show the area relationships between each series and the total. ▸ X-axis annotated using time labels. ▸ Series appearance determined by line style color.
BAR 	Each series is drawn as a bar in a cluster. The number of clusters is the number of points in the data. Each cluster displays the *n*-th point in each series. ▸ X-axis annotated with time labels of the respective data points. ▸ Series appearance determined by line style color ▸ 3D effect available using depth, elevation, and rotation properties.
STACKING BAR 	Each series is drawn as a portion of a stacked bar cluster, the number of clusters being the number of data points. Each cluster displays the *n*-th point in each series. Negative Y-values are stacked below the X-axis. ▸ X-axis annotated with time labels of the respective data points. ▸ Series appearance determined by line style color. ▸ 3D effect available using depth, elevation, and rotation properties.

Table 11: Monitor 2D chart types.

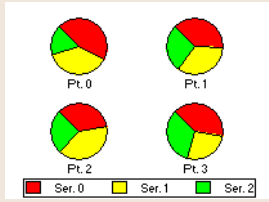
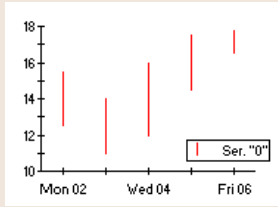
CHART TYPE	DESCRIPTION
PIE 	Each series is drawn as a slice of a pie. The number of pies is the number of points in the data. Each pie displays the *n*-th point in each series. ▸ Pies are annotated with time labels only. ▸ Series appearance determined by line style color. ▸ 3D effect available using depth and elevation properties.
HILO 	Two series are drawn together as a "high-low" bar. The points in each series define one portion of the bar: ▸ 1st series - points are the "high" value ▸ 2nd series - points are the "low" value The appearance is determined by line style color property in the first series of each pair.

Table 11: Monitor 2D chart types.

17.3.3 3D Chart Types

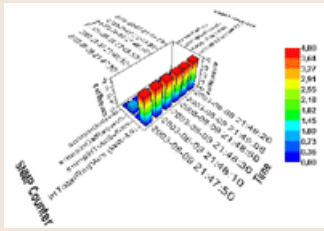
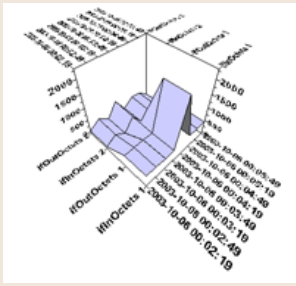
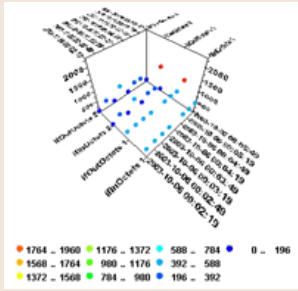
CHART TYPE	DESCRIPTION
BAR 	Each series is drawn as a series of 3D bars. Bars can be shaded, contoured, and zoned by elevation data (Z-axis). When zoned mode is chosen, the zone color ranges are displayed in the charts legend.
SURFACE 	From the elevation data of all displayed series an (optionally shaded) surface is drawn. The surface's top and bottom color can be defined independently.
SCATTER 	Each series is drawn as a series of points in the `x,z` plane, where `x` denotes the time and `z` the collected or computed value. Z values can be zoned. The zone colors and their corresponding value ranges are then display in the chart's legend.

Table 12: Monitor 3D chart types.

17.3.4 Monitor Expressions

There are four types of expressions that can be used for monitor configuration: *arithmetic*, *formatted arithmetic*, *boolean*, and *OID string* expressions.

Within arithmetic expressions, values from the primary round robin database may be referenced. Like a spreadsheet application, this can be done by specifying simple references or regions:

- **Reference**

  A reference consists of a letter and an optional positive integer or question mark ("?"). The letters A–S refer to the columns of the primary data table (thus they refer to the rows of the configuration table). The integer value refers to the row in the data table, where 0 refers to the last row, 1 to the second to last row and so on. The question mark refers to the *first* row.

  The special letter T refers to the time column which contains the number of milliseconds since January 1, 1970, 00:00:00 GMT.

Please note: Z may be used for legend and alarm configuration only!

  The special letter Z refers to that column of the primary data table for which the expression has been entered.

  Examples:

  ⏩ A1 refers to the second to last value of first series (if such a value exists).

  ⏩ Z refers to the last value of the series for which the expression has been entered.

- **Region**

  A region consists of two references where the first reference denotes the upper left corner of the region in the primary data table and the second reference denotes the lower right corner. Both references are concatenated by a colon (":"). Regions where the first and second reference do not follow the above rule are invalid.

  Examples:

  ⏩ A?:A0 refers to all values of the first series.

  ⏩ Z?:Z0 refers to all values of the series for which the expression has been entered.

  ⏩ T1:T0 refers to the time values of the last two collected rows.

  ⏩ A?:E0 refers to all values of the first to fifth series.

Arithmetic Expressions

Arithmetic expressions evaluate to a scalar numerical value, for example "(1+2)*3^2" and "SUM(Z?-Z0)". The following unary and binary

operators are supported and take precedence in the shown order (from highest to lowest):

```
^
+,- (unary)
%
/
*
+,-
```

The following functions are supported (operands might be scalar values, expressions, references, or regions):

OPERATION	DESCRIPTION
`ABS(s)`	Return the absolute value of scalar operand.
`AVERAGE(r1[,r2])`	Returns the sum of all elements divided by the number of elements.
`CEILING(r)`	Return the least integer greater than or equal to the operand, which may be a scalar or a list of values (region).
`COUNT(r1[,r2])`	Return the total number of elements in its operands.
`COUNTWHEN(sumOID[,selOID, regex[,displayHint]])`	Return the number of elements found in the subtree (i.e., column of a table) denoted by *sumOID*. If *selOID* and a regular expression *regex* are also given, only those objects in subtree of sumOID are counted where the corresponding value in subtree of selOID matches the given regular expression. For example, The OID(s) may be specified as "1.3.6.1.2.1.2.2.1.3" or "ifTable.1.3". With the optional displayHint string the display format defined for *selOID* can be overridden. For example, by specifying a display hint of "255a" OCTET STRING values will be compared as ASCII strings, by specifying "1x:" values will be compared as hexadecimal strings where each character is separated by a colon. The syntax of *displayHint* is the same as for the DISPLAY-HINT clause defined for SMIv2. Note: Use this function carefully since it could cause heavy load if used on large tables.

Table 13: Arithmetic expressions of the monitor configuration.

OPERATION	DESCRIPTION
`DELTA(r,s)`	Return the delta between the last and second to last element of the value list *r*, which is `r[n]-r[n-1]`. If *r* contains only one element, then zero will be returned. If `r[n]-r[n-1]` is negative, a counter wrapping is assumed and the delta is computed as `(s+(r[n]-r[n-1]))+1`. If the region *r* contains cells for which no values could be retrieved (because of a timeout or an error returned by the agent) then those cells will be ignored.
`DELTA32(r)`	Return the delta between the last and second to last element of the value list *r*, which is `r[n]-r[n-1]`. If *r* contains only one element, then zero will be returned. If `r[n]-r[n-1]` is negative, a counter wrapping is assumed and the delta is computed as : $(2^{32} + (r[n] - r[n-1])) + 1$ If the region *r* contains cells for which no values could be retrieved (because of a timeout or an error returned by the agent), then those cells will be ignored.
`DELTA32(s1,s2)`	Return *s2-s1* if this difference is not negative. If it is negative, then will be returned. $2^{32} + (s2 - s1)$
`DELTA64(r)`	Return the delta between the last and second to last element of the value list *r*, which is `r[n]-r[n-1]`. If *r* contains only one element, then zero will be returned. If `r[n]-r[n-1]` is negative, a counter wrapping is assumed and the delta is computed as : $(2^{64} + (r[n] - r[n-1])) + 1$ If the region *r* contains cells for which no values could be retrieved (because of a timeout or an error returned by the agent), then those cells will be ignored.
`DELTA64(s1,s2)`	Return *s2-s1* if this difference is not negative. If it is negative, then will be returned. $2^{64} + (s2 - s1)$
`FLOOR(s)`	Return the greatest integer less than or equal to the scalar operand.
`GEOMETRICMEAN(r1[,r2])`	Return the *n*-th root of the product of a set of *n* numbers.

Table 13: Arithmetic expressions of the monitor configuration.

OPERATION	DESCRIPTION
`IF(cond,s1,s2)`	Return *s1* if the boolean expression *cond* evaluates to true, otherwise s2 is returned.
`MAX(r1[,r2])`	Return the largest element of one or two regions or the largest element of two scalars.
`MEDIAN(r1[,r2])`	Return the middle element of a sorted list, or the average of the two middle values if the list has an even number of elements.
`MIN(r1[,r2])`	Returns the smallest element of one or two regions or the largest element of two scalars.
`POWER(s1,s2)`	The exponentiation (^) operation. It takes two scalar values or expressions. The left operand (*s1*) is the base and the right operand (*s2*) is the exponent: $(s1)^{s2}$
`PRODUCT(r1,r2)`	A product can be performed on a pair of elements or across a list. The product of a region is the product of its individual members. Multiplication order is left-to-right, and first element of a list to last element. The result of a matrix multiplication may depend on the order of the operands.
`ROOT(s)`	Return the positive square root of its operand:. $\sqrt{s}$
`ROUND(s)`	Return the nearest integer to the operand. Rounding is done to an even number if the operand is exactly midway between two integers.
`SORT(r1[,r2])`	Return a sorted list of the given elements. Any secondary or nested lists are flattened.
`STDDEVIATION(r1[,r2])`	The sample standard deviation, given by $\sqrt{\frac{1}{1-n}\sum_{i=1}^{n}\left(r_i-\left(\frac{1}{n}\sum_{i=1}^{n}r_i\right)\right)}$ where *n* is the number of samples.

Table 13: Arithmetic expressions of the monitor configuration.

OPERATION	DESCRIPTION
`SUM(r1[,r2])`	Return the sum of two scalars or the sum across one or two regions: $\sum_{i=1}^{n} r_i$
`SUMWHEN(sumOID[,selOID, regex[,displayHint]])`	Return the sum of all (numerical) values found in the subtree (i.e., column of a table) denoted by *`sumOID`*. If *`selOID`* and a regular expression *`regex`* are also given, only those objects in subtree of *`sumOID`* are add up where the corresponding value in subtree of *`selOID`* matches the given regular expression. With the optional *`displayHint`* string the display format defined for *`selOID`* can be overridden. See also `COUNTWHEN`. *Note: Use this function carefully since it could cause heavy load if used on large tables.*
`TRUNC(s)`	Returns the integer part of a number. Equivalent to rounding to the nearest integer closer to zero. Example: `trunc(-3.5) = -3`

Table 13: Arithmetic expressions of the monitor configuration.

Formatted Arithmetic Expressions

A formatted arithmetic expression is an arithmetic expression enclosed in an optional `FORMAT` operation:

```
<formatted arithmetic expression>:
      [FORMAT( <arithmetic expression>,
                <decimal format string>
      [, <minimum width>] )] | <arithmetic expression>
```

where `<decimal format string>` is a pattern string as described in the Java `DecimalFormat` class documentation, for example `"#,##0"` formats a positive number with no decimal digits and a separator. The optional `<minimum width>` positive integer specifies the minimum width in characters of the formatted string. If the formatted string's length is less than the specified number, spaces will be inserted at the beginning of the result string until it has the specified length.

Boolean Expression

A boolean expressions evaluates to a truth value: `true` or `false`. MIB Explorer supports several comparison operators that take one or two

arithmetic expressions. These boolean operators take the following precedence (from highest to lowest):

```
<, >, <=, >=
==, !=
!
||, &&
```

All arithmetic operators take precedence over these boolean operators, for example the boolean expression

```
5 * 10 < 50 || (5 == 2+3)
```

evaluates to `true`.

OID String Expression

An OID string expression evaluates to an OID string, which is a dotted string of positive integers (e.g., "`1.3.6.1.2.1.2.2.1.3.4976`"). There is only one operator supported for OID string expressions:

```
+
```

The plus sign concatenates two OID strings by inserting a dot ("`.`") between them. Thus, the expression

```
"1.3.6"+"1.2.1.2.2.1.3.4976"
```

will evaluate to "1.3.6.1.2.1.2.2.1.3.4976". In addition, MIB Explorer supports two very powerful operations to retrieve OID strings from a target:

OPERATION	DESCRIPTION
INDEXOF(*oid*, *regex*[,*n*,[*displayHint*]])	Searches all instances of the OBJECT-TYPE represented by *oid* for the *n*-th matching of the given regular expression *regex* with the value of the respective instance. If n is not specified, then the index of the first instance of oid that matches regex will be returned. The parameter *oid* may be an OID string ("1.3.6.1.2.1.2.2.1.3") or an object name with an OID string suffix ("ifTable.1.3"). In the latter case, the MIB module needed to resolve the object name must be loaded whenever the monitor is used! The optional parameter *displayHint* can be used to specify how the values of the instances of *oid* are converted to strings for matching with *regex*. For example, by specifying a display hint of "255a" OCTET STRING values will be compared as ASCII strings, by specifying "1x:" string values will be compared as hexadecimal strings where each character is separated by a colon.The syntax of *displayHint* is the same as for the DISPLAY-HINT clause defined for SMIv2.
VALUEOF(*oid*)	Returns the value of the MIB object instance represented by *oid*. The parameter *oid* may be an OID string ("1.3.6.1.2.1.2.2.1.3.4976") or an object name with an OID string suffix ("ifType.4976").

Table 14: OID string operations of the monitor configuration.

17.3.5 Monitor Properties

To configure the properties of a monitor, click on the Properties () button of the monitor's main toolbar.
The following categories of settings are available (*italic* categories are available for 3D chart monitors only):

1. Chart
2. Chart Area
3. X Axis
4. Y Axis
5. Z Axis

6. Legend
7. Titles
8. Data

Chart Properties

- **Chart Type**

  Selects the chart type from eight supported basic types: Plot, Scatter Plot, Area, Stacking Area, Bar, Stacking Bar, Pie, and Hi-Lo.

- **Chart Name**

  Specifies the chart's name which is shown by the chart's legend (if visible). You may enter a single line of plain text or HTML formatted text. The HTML text has to start with `<html>` and end with `</html>`. Formatted arithmetic expressions may be placed within plain text as well as in HTML text between brackets `{` and `}`. The brackets will not be part of the output text but replaced by the result of the entered expression. For example, the chart name

```
<html><center>Chart Name<p>Maximum of A=
{FORMAT(MAX(A?:A0), "#,##0")}</center><tml>
```

will produce the following output:

```
    Chart Name
Maximum of A=1.234
```

- **Foreground Color**

  Defines the foreground color for the chart.

- **Background Color**

  Defines the background color of the chart

- **Preferred Width**

  Specifies the width of the chart image in pixels when this monitor is run on a headless server. If a width of `0` pixels is given, a server will use either its built-in default value of `640` pixels or the width specified by the server's command line option.

- **Preferred Height**

  Specifies the height of the chart image in pixels when this monitor is run on a headless server. If a height of `0` pixels is given, a server will use either its built-in default value of `400` pixels or the width specified by the server's command line option.

Chart Area Properties

- **Font**

  The font for text printed within the chart area.

- **3D Effect (not available for 3D charts)**

  - Depth

    The depth property controls the apparent depth of a graph.

  - Elevation

    The elevation property controls the distance above the x axis for the 3D effect.

  - Rotation

    The rotation property controls the position of the eye relative to the y axis for the 3D effect.

- **3D Settings (not available for 2D charts)**

  - Axis Scaling

    The three axes (x,y,z) of a 3D chart can be individually scaled relative to each other.

  - Cube

- **Transparency**

  If selected, all lines of the objects (e.g. bars) within the 3D cube will be displayed.

- **Zoned**

  If selected, the distribution of the elevation data is shown.

- **Contoured**

  Displays data distribution by drawing contour lines demarcating each of the contour levels.

- **Meshed**

  If selected, the X-Y grid projected onto the 3D surface is displayed in a 3D view with a Z-axis.

- **Shaded**

  If selected data will be displayed as a flat shaded surface.

- **Shaded With Series Color**

  If selected, the color assigned to the respective chart series are used for shading.

Axis Properties

- **General**
  - Title

    The title for the axis. It can be specified as a plain text string or a HTML string, which has to start with `<html>` and end with `</html>`. In order to make the title visible you have to select the Visible check box.

  - Rotation

    The rotation of the title.

  - Placement

    The placement for the title relative to the axis.

  - Grid

    The grid tab specifies the axis grid and whether it will be shown or not. The grid settings will have no effect on 3D charts.

- **Font**

  The font to be used for point labels and axis title.

Legend Properties

The legend of 2D charts is created from the Legend column of the Monitor Configuration. The legend of 3D charts is computed from the elevation data (`z` series) of the monitor.

- **General**
  - Visible

    If selected the legend will be displayed, otherwise it will be not visible.

  - Anchor

    The anchor property specifies the location of the legend relative to the chart area.

  - Orientation

    Set the orientation of the chart.

- **Font**

  Set the font used to print the legend.

- **Colors**

  In order to specify a background color for the legend the Opaque check box has to be selected. The foreground color can be specified in either way.

Titles Properties

- **Header**

  - General

    Set the header text, which can be specified as a single line of plain text or a multi-line HTML text. Select the Visible check box to make the header visible. Formatted arithmetic expressions may be placed within plain text as well as in HTML text between brackets `{` and `}`. The brackets will not be part of the output text but replaced by the result of the entered expression. For example, the chart name

    ```
    <html><i>Chart Title</i>
      <p>Total Average is {FORMAT(AVERAGE(A?:F0),
        "#,##0", 10)}</center><html>
    ```

    will produce the following output:

    ```
    Chart Name
    Total Average is    10.234
    ```

  - Font

    Select the font for the header.

  - Colors

    In order to specify a background color for the header the Opaque check box has to be selected. The foreground color can be specified in either way.

- **Footer**

  - General

    Set the footer text, which can be specified as a single line of plain text or a multi-line HTML text. Select the Visible check box to

make the footer visible. As with header text, also the footer supports formatted arithmetic expressions.

- Font

  Select the font for the footer.

- Colors

  In order to specify a background color for the footer the Opaque check box has to be selected. The foreground color can be specified in either way.

Data Properties

The data properties define how SNMP data is collected and consolidated by the monitor and how that data is exported and saved to disk.

- **General**

  - Data Coverage

    The collected values can be stored as Absolute or as Delta values. Delta data coverage should be chosen for monitors that exclusively collect data from `Counter`, `Counter32`, and `Counter64` objects. In delta mode, counter wrappings will be detected if a delta value is negative. Then the delta will be computed as described for the `DELTA32` and `DELTA64` operations (`DELTA64` is used if syntax of collected object is `Counter64`).

    All other monitors should use absolute data overage. If you want to collect absolute data, like temperature, and counter based values, like incoming packets, with the same monitor, then you should use absolute data coverage and add a computed series for each counter based value that computes its delta value using one of the available DELTA operations.

    If delta coverage is selected, then the raw delta value can be put into ratio with the time delta by specifying a factor f unequally zero. A zero factor will disable the time ratio and the raw delta values will be stored (`dv=(v0-v1)`). Otherwise the delta will be computed according to the following formular:

$$dv = \frac{(v0 - v1)}{\left(\frac{1}{1000}(t0 - t1)\right) \cdot f}$$

    where `v0` denotes the last value collected and `v1` the second to last value, `t0` the time in milliseconds when the last value has been col-

lected, `t1` the time when the second to last value has been collected, and `f` denotes the specified factor (`!=0`). Thus, to collect delta values on a per second basis, use a factor of `1`.

- Data Source

  By default, data is being collected by sending GET, GETNEXT or GETBULK requests to the target(s) configured for a monitor and collecting the values from the corresponding responses.

  If the data that needs to be collected and monitored is sent by notifications, then a monitor can be configured to use passive data collection. In passive mode, a monitor listens for notifications on the listen address(es) configured for MIB Explorer's trap receiption. When it receives a notification, its ID is compared to the notification ID prefix configured for the monitor. If they match and the source IP address of the notification matches the target configured for the first series (`A`), then a new primary data row is appended to the primary data. Those values of variable bindings will be assigned whose OID match the OID prefix defined for a series while the series' target is empty or equals the source IP address of the notification.

- Start Time

  If given, the start time specifies when data collection for this monitor starts.
  By specifying a date and time (`yyyy-MM-dd HH:mm:ss`) in the future and closing the monitor properties dialog with OK, the monitor will switch into run mode. Leaving the start time empty will allow a manual start of the monitor. When a monitor with a start time in the future is deployed to a server, then the server will start data collection for that monitor at the given time.

- Round Start Time

  Check this option, if you want to MIB Explorer to use a round start time when starting the monitor. A round start time is the next full minute, five minutes, ten minutes, fifteen minutes, half an hour, hour, six hours, twelve hours, or 24 hours - depending on the interval specified.

- Interval

  The update interval specifies the period in seconds between two consecutive data collections while the monitor is in run mode.

- Fixed Rate

If checked, updates are scheduled for repeated *fixed-rate execution*, beginning at the specified time. Subsequent updates take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period.

If Fixed Rate is not checked, updates are scheduled for repeated *fixed-delay execution*. Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as garbage collection or other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period.

- Primary Samples

  The maximum number of data points (rows) for all series in the primary round robin database.

- Index Calculation

  Specifies whether calculated index values should be cached for subsequent data collections or whether index values should be determined for each collection separately.

- **Consolidation**

  For each monitor an arbitrary number of consolidation round robin archives can be configured. To add a consolidation archive to the monitor press the Add button and a new row will be added at the end of the table. To remove a consolidation archive, select it and press the Remove button. Changes will not take effect until the monitor properties dialog is closed using the OK button. However, if the changes are saved, no undo is available and all data in any removed consolidation archives will be lost!

  The properties of a consolidation archive are:

  - Name

The name of the consolidation archive. This name will be appended to the monitor name when primary and consolidation tables are exported and if the corresponding charts are exported. Thus, if you intend to export any data, you should avoid any special characters and use letters and digits only.

- Function

  Set the consolidation function to be used to create consolidation data points. The consolidation function takes a vector of primary data points, whose size is specified by Stepping, and computes a consolidated data point from it. Available functions are: `AVERAGE`, `MAX`, `MIN`, `LAST`, and `SUM`.

- Samples

  The (maximum) number of rows in the consolidation archive.

- Stepping

  The number of primary data points needed to create a consolidated data point.

Export

- Export Directory

The directory where data table and charts should be exported to. If the specified directory is invalid or empty, data or chart export will not work. Please check the log, category "Monitor", for any errors.

Note: Even if the Enable *button is selected no export takes place until the* Export *toggle button is selected.*

- Data File Type
  Specifies whether data tables (primary as well as consolidated) should be exported as comma separated values (CSV) or XLS files whenever data is updated while the Export ( ) button is selected. Check the corresponding Enable button to enable export of data tables.

Note: Even if the Enable *button is selected no export takes place until the* Export *toggle button is selected. When running the monitor on a server, the export of the monitor configuration is always activated.*

- Chart File Type
  Specifies whether charts (primary as well as consolidated) should be exported as `JPEG`, `GIF`, `PNG`, `PDF`, `PS`, or `PCL` files whenever data is updated while the Export ( ) button is selected. Check the corresponding Enable button to enable export of charts.

- Monitor

  Enable the "Auto save monitor after data update" check box to save the monitor (including any uncommitted configuration changes) whenever new data is collected and (a) the Export ( )

toggle button is selected or (b) the monitor is being run on a server. With the option "Auto Save Interval" you can specify how many updates have to have been processed before an auto save is actually performed. A value of zero will save the monitor data each time new data has been collected.
With the option "Number of Backup Files" the number of backup files of the auto saved monitor configuration can be configured. No backup file is saved if the number is zero. Otherwise, up to the specified number backup files will be saved, where the file with the suffix `.001` contains the next to last version.

Monitor Series Styles

To change the style of a series:

1. Select the Configuration node in the navigator tree.
2. Select the row representing the series to change by clicking on the row label.
3. Open the popup menu with the right-mouse button.
4. Choose Style... from the popup menu. A dialog with the tabs Line Style and Symbol Style will be opened.
5. Change the settings and press OK to save the changes.

- **Line Style**

  Set the color, width, and style of lines drawn for a series in the chart view. The selected line color will also be used for filling bars and areas.

- **Symbol Style**

  Set the color, shape, and size of symbols drawn by plot and scattered plot charts.

Monitor Toolbars

Figure 14: Monitor main tool bar.

- **Add New Data Point**

  Add a new row representing a series before the selected row. If no row is selected, the row will be added at the end of the monitor configura-

tion (in order to remove any selection in the configuration table, press <ESC>).

- **Duplicate Series**

  Duplicates the selected row in the monitor configuration table.

- **Remove Selected Series**

  If the current view is the monitor's configuration, then remove the selected rows (series) from the monitor. The corresponding rows in the round robin databases, where collected and computed data is stored, will also be deleted when you commit your configuration changes by either selecting a data table or chart in the navigator tree or by choosing from this tool bar.

  If the current view is one of the data table views, then it removes the selected rows from the viewed data table. In contrast to the row deletion in the configuration view, this operation cannot be undone.

- **Move Series Up**

  Move the selected series one position up in the configuration table. This will also move the corresponding columns in the data tables one position to the left when configuration is committed. Accordingly, this will also change the order of series in the charts' legends.

- **Move Series Down**

  Move the selected series one position down in the configuration table. This will also move the corresponding columns in the data table one position to the right when configuration is committed. Accordingly,, this will also change the order of series in the charts' legends.

- **Delete Collected Data**

  Empty all data tables of the round robin databases and charts.

- **Collect Data Once (Not Available for Remote Monitors)**

  If there have been made any changes to the configuration of this monitor, you will be prompted to commit the changes to the data tables by

deleting all data or by preserving existing data. Choosing the latter will have the following affect:

- All columns referring to series that have been removed from the configuration will be removed from the series.
- New series will be filled with null (no data) values.
- Series that have been reordered in the configuration will also be reordered in data tables and charts.

Once changes have been committed, a new primary data row will be created in the primary data table and the corresponding series values will be collected and computed.

- **Redo Last Change**
- **Undo Last Change**
- **Detach Monitor**

  Detach the monitor from the main window and show it in its own window. This can be useful if you want a custom size for the created chart(s).

- **Attach Monitor**

  Reattach a previously detached window to the main window.

- **Save As**

  Saves the displayed table, chart, or configuration (depending on the selected view) to a file. When the configuration view is selected, this function is the same as the Save As function from the basic monitor operations ("Basic Monitoring Operations" on page 102).

  If a chart is displayed, it is saved as a `JPEG`, `GIF`, `PNG`, `PDF`, `PS`, or `PCL` file. If a data table is displayed, the table is saved as a `XLS` or `CSV` file. Which format is used, depends on the file extension for the target file.

- **Monitor Properties**

  Configures the monitor properties, for example chart type and appearance, data coverage, update interval, number of samples, and data consolidation. Although, monitor properties may be changed while it is

running, it is not recommended to change any data properties while a monitor is running.

Figure 15: Refresh and update monitor tool bar.

Caution: If the monitor is a client monitor connected to a MIB Explorer server, then the server's monitor instance will be automatically updated with the new monitor configuration when committed.

- **Start Monitor**

  Start the monitor. If there are any uncommitted changes to this monitor's configuration pending, then you will be prompted to commit the changes to the data tables by deleting all data or by preserving existing data.
  Then the monitor collects one new primary data row from the configured target(s). This operation will repeated for each update interval as specified by the monitor properties.

- **Stop Monitor**

  Stop the monitor. Data will no longer be updated regularly. If a client monitor is stopped, the server's monitor instance will be not affected. If you want to stop a monitor on the server, use the Connect dialog as described in “Basic Monitoring Operations” on page 102.

- **Progress Status**

  The progress status shows date and time when the next update will occur. At the same time, the progress bar shows the percentage of time elapsed for this interval waiting for the next update. If a client monitor is started, **`"Waiting for next remote update"`** will be displayed in the progress status bar until the next update on the primary data of the remote monitor has been propagated to the client.

- **Export**

  Enable the export of data tables and charts. The export directory as well the export file types have to be specified with the monitor data export properties. Otherwise, enabling this option has no effect.
  Table data can be exported to comma separated value (`CSV`) files and XLS files. Chart images can be exported as `JPEG`, `GIF`, `PNG`, `PDF`, `PS` or `PCL` files. If the auto save mode is enabled and export is selected, the monitor file will be saved whenever data is collected.

Important note: The auto save mode will also save any configuration changes you make to the monitors configuration while it is running!

17.4 Interactive Chart Customization

A monitor's chart view can be interactively customized by the user. The following interactions are provided:

- moving the chart by holding down the <Shift> key while dragging the mouse with the left mouse button pressed
- zooming into or out of the chart by holding down the <Alt> key while dragging the mouse with the left mouse button pressed
- rotation of 3D charts or 2D bar or pie charts with 3D effect by holding down the <Ctrl> key while dragging the mouse with the left mouse button pressed

To reset a chart to its default view settings, select the chart view and press <Alt>+<R>.

18 Packet Analyzer

To enable the packet analyzer on application start, press the <Shift> key while pressing the "Capture Packets" toggle button.

The SNMP Packet Analyzer can be used to analyze all SNMP packets sent and received by MIB Explorer and in addition packets you provide as a hex string. By default the packet capturing is disabled to save resources and increase overall performance.
The Packet Analyzer panel is the second to right tab of the Tools panel. It is divided into three areas:

1. The top most pane contains a list of all captured packets with their source and destination addresses, their transport, size and content (as a hex string). Packets you have entered manually for analysis have an all zero source and target address with UDP as transport.
2. The left pane contains the SNMP message structure of the selected message as a tree. The tree reflects the SNMPv1, v2c, or v3 message format defined using the Abstract Syntax Notation 1 (ASN.1). The tool tips of the tree nodes and the node text provide you with information about the message's encoding according to the *Basic Encoding Rules* (BER).
3. The right pane displays the selected packet as a hex dump with a view of the printable characters on the right. The above selected BER element (node) is highlighted through bold text within the message's hex dump.

Captured packets can be saved into a capture XML file. The XML schema for the capture file format can be found in the `xsd` directory of the MIB Explorer installation. Capture files can be opened later at any time to continue analysis.
MIB Explorer can also parse packet dumps from log files. With that feature you can easily analyze the complete packet flow from a log file including the send and receive times if logged at the beginning of each packet line.

The packet analyzer tries to decrypt the scoped PDU with the user information provided in the Targets configuration on the fly when displaying the messages BER structure. When the security credentials have been changed or are not identical with those used by the message sender, then the structure tree may contain a "BER error..." node below the "Encrypted Scoped PDU:..." node. This node does not actually indicate an error - it is just caused by an incorrect decryption of the scoped PDU due to non-matching security credentials.

When you click on a node within an encrypted scoped PDU then the scoped PDU will be displayed decrypted in the hex dump at the appropriate place. Because decrypted PDUs may have less payload bytes than their encrypted counter part, it may contain superfluous bytes at its end.

18.1 Operations

To Start Packet Capturing

1. Select the Packets tab from the tools panel.
2. Click on the Capture Packets toggle button. MIB Explorer will start to capture all SNMP packets send and received via the configured transport mappings.

To Stop Packet Capturing

1. Select the Packets tab from the tools panel.
2. Click on the Capture Packets toggle button to deselect it. MIB Explorer immediately stops capturing packets.

To Clear Packet List

1. Select the Packets tab from the tools panel.
2. Click on the Clear () button and the packets list will be cleared.

To Save Packet List

1. Select the Packets tab from the tools panel.
2. Click on the Save as () button.
3. Specify a (new) XML file to store the captured packets (those currently in the packet list).
4. Click Save to save the file.

To Open a Packet List

1. Select the Packets tab from the tools panel.
2. Click on the Open () button.
3. Specify a previously saved captured packets XML file.

4. Click on Open to load the packets into the packets list. Any already listed packets will be removed and replaced by the packets in the loaded file.

To Manually Analyze (Decode) a Packet

1. Select the Packets tab from the tools panel.
2. Click on the Analyze () button.
3. Enter a complete SNMP message in hexadecimal format (bytes separated by a colon) and press OK.
4. The message will be added to the packet list. Click on it in the list to analyze its structure and content.

To Analyze Packets from a Log File

1. Select the Packets tab from the tools panel.
2. Click on the Analyze Log... (2nd ) button.
3. Enter the file name of the log file to extract the packets from. The packets must be dumped in the log file in hexadecimal format (bytes separated by a colon).
4. Enter a format of date and time information at the beginning of a log line with a dumped packet as Java `SimpleDateFormat`. The format for SNMP4J is the default (`yyyy-MM-dd HH:mm:ss`). For AGENT++ log files, use `yyyyMMdd.HH:mm:ss`.

19 Discovery of Network Elements

With MIB Explorer, SNMPv1/v2c/v3 command responder, generators and other network elements can be discovered in a Local Area Network (LAN) by using broadcast targets and other targets (e.g. routers or bridges) as seeds. To perform a discovery you need to specify at least one seed target. That target should use a community or SNMPv3 user common to most (possible) targets in your network.
Besides active discovery, source addresses of notifications are also covered by the discovery process.

To Discover Network Elements

1. Choose the Discovery tab from the Tools panel.
2. If you have not yet defined targets to be used as seeds, choose Discovery Preferences () from the tool bar. A configuration dialog with three tabs will be opened:

   ⏩ In the first tab you specify the seed targets, by adding them to right list of the shuffle lists.

   ⏩ The refresh interval and the scan interval in seconds are specified in the second tab. The refresh interval defines the number of seconds MIB Explorer will wait until it refreshes the state of each discovered network element. The scan interval must be less or equal to the refresh interval. The scan interval defines the number of seconds MIB Explorer will scanning for new network elements and updating existing ones.

   ⏩ With the third tab additional variables may be defined that should be collected by the discovery process. The given OIDs will be added to the GET request sent to discovered network elements. The default variables requested are:

   ```
   ifNumber.0, sysName.0, sysUpTime.0, sysLocation.0,
   sysContact.0, sysDescr.0, sysObjectID.0, and sysSer-
   vices.0
   ```

19.1 Tool Bar

Figure 16: Discovery tool bar.

- **Start**

  Starts the discovery process. All items in the discovery table will be removed and newly discovered network elements will be added to the table in background.

- **Stop**

  Stops the discovery process. This may take a few seconds.

- **Refresh**

  Discovers agents using the selected target and add them to the result table. Already discovered targets will be updated. A GET PDU is send to the IP address and UDP port specified in the discovery target. The PDU's variable bindings are. If a target responds with an error status other than `0` (no error), the target will not be displayed in the result table. Instead, the error counter will be incremented by one and displayed in the Status Bar.

- **Add Selected Network Elements to Target Configuration**

  Adds the selected targets to MIB Explorer's configuration. Each target will be named by its discovered system name (`sysName.0`). The selected targets will be contacted with the configured seed target security information (by order of their position in the seed configuration list). The new target gets the configuration of the first seed whose security information results in a successful contact.

- **Discovery Preferences**

  Opens the target configuration window, where you can make adjustments to any target, in particular discovery targets. The target selected in the configuration when you save your changes using the Save button will become the next discovery target.

19.2 Table of Discovered Network Elements

The table of discovered network elements may contain SNMP command responder (typically agents) as well as simple network addresses discovered from notifications and routers. If a network element has been discovered through evaluation of a system's address table then that source system's IP address will be displayed in the Source Address column and its Domain Name System (DNS) name in the Hostname column.

The Last Contact column shows the time of the last update of the discovered variables of the network element. If there is not any time shown, then no variables has been discovered for the network element and it could not be contacted with a ping. The refresh interval for the variables can be configured with the Discovery Preferences.

20 SNMPv3 User Administration

An USM User associates SNMPv3 security parameters with a user name. RFC 3414 describes how the use of the *User Security Model (USM)* protects SNMPv3 communication against classic threads against network protocols.

A resource is commonly secured by password protection. However, the administrative overhead for using a different password for each network device is big. On the other hand, using a single password for all network devices is very dangerous; because once an attacker has deciphered the password it compromises all devices in the network. As a consequence, the USM security model localizes a plain text password with a SNMP entity's engine ID using a hashing algorithm. The resulting key is no longer human readable and even if it is deciphered it provides only access to a single SNMP entity. Nevertheless, changing the secrets of a USM user on a regular basis is required to protect the secrets against disclosure, as stated in RFC 3414 §11.1 Recommended Practices:

> The frequency with which the secrets of a User-based Security Model user should be changed is indirectly related to the frequency of their use. Protecting the secrets from disclosure is critical to the overall security of the protocols. Frequent use of a secret provides a continued source of data that may be useful to a cryptanalyst in exploiting known or perceived weaknesses in an algorithm. Frequent changes to the secret avoid this vulnerability.

MIB Explorer provides all necessary operations to:

- create a new USM user by cloning it from an existing user on the agent
- modifying the secret keys (i.e. changing passwords) of an USM user

on a single target and optionally also on more than one target at once.
The following two sections “Key Change” and “Multi-Target: Create or Modify USM User” refer each to one of these two key vs. password management approaches.
The first section is about key change, which includes creating new users by cloning from existing users. But different to the multi-target approach, here localized keys are used. Key change is only available if the active target’s version is “SNMPv3 /direct”.

When dealing with multiple targets, localized keys cannot be used. Instead the active user and the new/modified user's secrets need to be provided as passphrases. Therefore, this option is available for targets with version "SNMPv3 /USM" without localized key only.

20.1 Key Change

The key change menu is available if the active target is a "direct user target" with version "SNMPv3 /direct".

20.1.1 USM Key Change

The USM key change uses the active target (see "Selecting the Active Target" on page 45) to modify the same or another USM user in the target's user based security model (USM).
As shown by Figure 17, there are the following attributes required to do a key change:

- **Security Name**

  The security name of an existing user (= modifying) in the active target or the new user (= cloning). By pressing the label button, you can select a security name from existing users of the active target.

- **Engine ID**

  The engine ID for the new user. Typically, this is the engine ID of the target which is the authoritative engine ID for command responder. The engine ID differs from that target engine ID, if the user is supposed send requests to other command responders, i.e. INFORM, GET, GETNEXT, GETBULK, or SET requests.
  To select an engine ID from the engine IDs in the target already associated to any security name (using `usmUserSecurityName` OID) use the label button.
  Providing an empty engine ID will use the engine ID of the target (or discover it, if it is not known yet).

- **Authentication**

  - Protocol

    The authentication protocol for the new user which is mandatory.

  - Passsphrase

    Any character sequence with more than 8 bytes length. This sequence will be localized with the provided Engine ID.

- **Privacy**
  - Protocol

    The optional privacy protocol. If the user must not use privacy, leave it empty.

  - Passphrase

    Any character sequence with more than 8 bytes length. This sequence will be localized with the provided Engine ID using the selected authentication protocol. For best interoperability, make sure that the key length produced by the authentication protocol is greater or equal to the required key length of the selected privacy protocol. For AES256, for example, use at least SHA256.
- **Access**

  If a new USM user is created in an agent within the `usmUserTable`, that user is not mapped to a view group in the View Access Control Model (VACM) of the agent. Therefore the user cannot be used with SNMP requests. To solve this issue, you can check this option, to copy the VACM group of the active target's user (= the operational user) and assign the new user to that group.
  With this option checked, the new user will be allowed to access the same objects than the operational user. Otherwise, no access will be granted.
- **New Target Name**

  Specifies the name of the new target in MIB Explorer that this key change operation will create for you. The name must not be assigned to any other target yet.

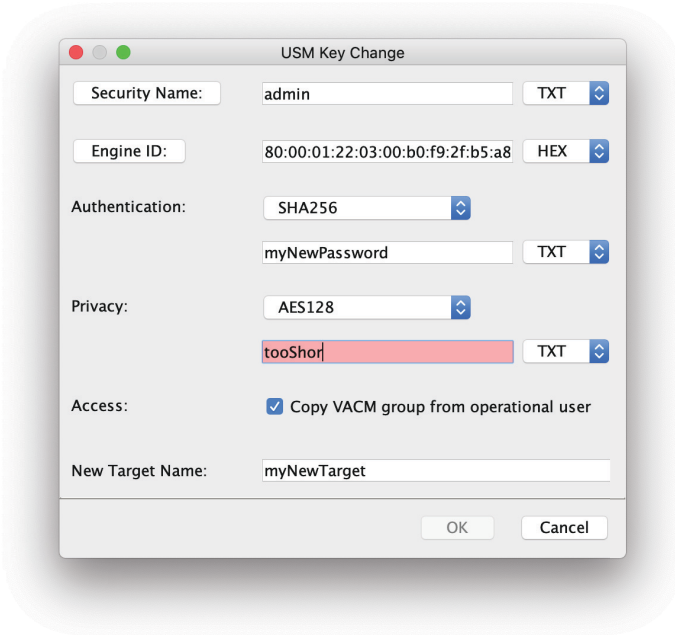


Figure 17: USM Key Change dialog with sample data

20.1.2 Diffie Hellman (DH) Key Change

By using the Diffie Helllman (DH) key change, MIB Explorer will itself compute the new authentication and optionally privacy keys based on shared keys computed on agent and MIB Explorer side individually. That means that the new keys are not send over the wire, instead each side can compute the shared keys based on public keys received from the peer.
To run a DH key change, the following prerequisites must be met:

- Agent must support DH Key Exchange by RFC 2786, for example SNMP4J-Agent 3.3.4 or later.
- Agent contains a user with the desired authentication and (optional) privacy protocol with a known keys to be modified/updated. The user whose keys should be exchanged can be different to the user running the operation (which is the user of the active target).

With the label buttons of input fields you can select available values from the target agent. Entering or selecting other values might only make sense when debugging an agent.

After having select the security name, authentication protocol and optionally the privacy protocol of the user to update, press the button Execute Diffie Hellman Key Change to run the change on the agent.
Within MIB Explorer, no configuration data will have been updated so far. To update the active target (or create a new target) with the new keys, you need to press the button Create Target or Update Target.
Pressing Cancel after the key change, might result in not being able to access the agent anymore unless you have copied/saved the new keys by other means than updating the active target or creating a new one.
To not change, MIB Explorer's configuration, press Cancel.

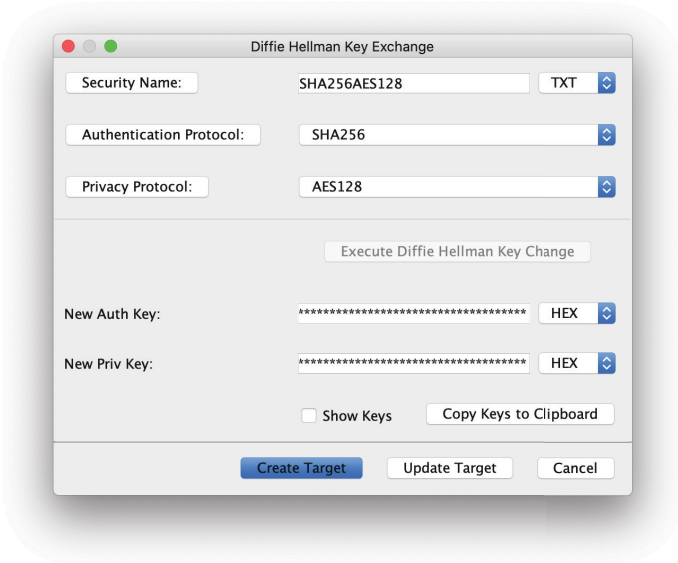


Figure 18: Diffie Hellman Key Exchange dialog

20.2 Multi-Target: Create or Modify USM User

A new USM user is created via SNMP by cloning it from an existing user. Thus, an initial user has to be configured for each SNMP command responder (agent) by other means than SNMP, for example a configuration file. The authentication and privacy passwords of an USM user should be always changeable.

To Create an USM User

Note: A user cannot provide a higher security level than the user it has been cloned from.

1. If not done yet, configure the target(s) for which the new user should be created (see "Adding a New Target" on page 45). For each of the

targets, configure the USM user you want the new user to be cloned from (see "Adding an USM User" on page 52).

2. Choose Create/Modify SNMPv3 User from the Edit menu.
3. Select as User for Operation the user configured in step 1, thus the user you want the new user to be cloned from.
4. Specify all necessary values for the user to be created in the User to Be Created/Modified pane. Fill in a SNMP engine ID, if the user should be created on behalf of an engine ID different from the targets engine ID. This is necessary for setting up a user for enabling a target to send INFORM messages or to proxy SNMP requests to other targets. In all other cases, the engine ID field can be left empty.
5. For very specific tasks it could be useful to check the Do not modify operational user option. When checked, the user information of MIB Explorer will not be changed trough the operation, thus the cloned user will not be added to the user repository of MIB Explorer.
6. Check the Copy VACM group for new user from operation user option, if you want to assign the same access rights of the operational user to the new user. Otherwise leave it unchecked. Then the new user will have no access rights unless the agents VACM is updated as needed.
7. Press the Next button to get to the next step of the wizard.
8. Select the targets you want the new user to be created for from the Available Targets table. It shows all targets for which the selected operational user (specified in step 3) is configured. Press the Add button to add the selected target to the list of targets to be changed.
9. Press the Finish button to start the creation of the new user.
10. The status of the operation will be shown in step 3 of the wizard. You cancel the operation by pressing the Stop button. For each target the status is shown in a table.

    The new user will be added to MIB Explorer's user configuration. The configuration of the changed targets will not be changed.
11. Close the wizard by pressing the Close button.

To Modify an USM User

1. If not done yet, configure the target(s) for which a user should be modified (see "Adding a New Target" on page 45). Configure for each of the targets the USM user you want modify (see "Adding an USM User" on page 52).

2. Choose Create/Modify SNMPv3 User from the Edit menu.
3. Select as User for Operation the user configured in step 1, thus the user to be modified.
4. Change the properties of the user to be modified in the „User to Be Created/Modified“ pane. Select the same user in both User fields! Fill in a SNMP engine ID, if the user is modified on behalf of an engine ID different from the targets engine ID. This is necessary for enabling a target to send INFORM messages or to proxy SNMP requests to other targets.

Warning: When checking the Do not modify or add local user *option, make sure that you have configured a user with the new credentials or otherwise you will not be able to access the agent(s) any more!*

5. For very specific tasks it could be useful to check the Do not modify operational user option. When checked, the user information of MIB Explorer user specified on the left side will not be changed trough the operation! As a consequence, after having successfully changed the users security credentials, you might not be able to access the agent(s) with the user you have chosen for the operation.
6. Press the Next button to get to the next step of the wizard.
7. Select the targets for which you want the user to be modified from the Available Targets table. It shows all targets for which the selected operational user (specified in step 3) is configured. Press the Add button to add the selected target to the list of targets to be changed.
8. Press the Finish button to start the creation of the new user.
9. The status of the operation will be shown in step 3 of the wizard. You cancel the operation by pressing the Stop button. For each target the status is shown in a table.

   If the operation failed or was canceled for any of the selected targets, MIB Explorer will add a new user to MIB Explorer's configuration with the current date and time appended to the user profile name of the modified user. That new user will be an exact clone of the original (unmodified) user profile. Each failed target will then be automatically configured to use the clone user, whereas each successfully updated target will use the modified user.
10. Close the wizard by pressing the Close button.

20.3 Deleting an USM User

To delete an USM user:

1. Open the `usmUserTable` with the MIB Tree Panel’s context menu Table.

2. Select the row of the user you want to delete.
3. Set the value of the RowStatus column to `destroy(6)`.
4. Press Commit & Verify.

21 Logging

MIB Explorer 5 or later uses Java logging as internal logging framework. By default logging is enabled. Logging can be disabled or enabled by using the Log panel of MIB Explorer's user interface (see below). Logged events are shown in the logging text area of the Log panel. They can be exported to a text file using the Save As button.

21.1 Configuration

1. Select the Log tab from the tools panel.
2. Press the Properties button. The logging properties window will be displayed.
3. Enter the maximum number of log records to be held by MIB Explorer in the log area. Zero will disable logging.
4. Browse through the event tree and assign priorities other than FATAL to the events you want to monitor. Assigning FATAL to the root priority will disable logging for all subtrees in the event hierarchy that do not override that priority.
5. Press Save to save the settings. The logging properties will be restored when MIB Explorer is started for the next time.

22 Tools

22.1 Incremental Search

To start the incremental search, you can press <Ctrl->-<I>.

Many dialogs and screen elements support instance search if you select the element (by mouse or keyboard actions) and then starting to type the term you are looking for.
The first matching element in the list, combo box, or table will be selected then and a popup menu will open where the search term can be entered or modified.
Using the up and down keys navigates through the found matches. Matching is case-insensitive and searches for sub-strings. The search term is a (Java) regular expression.

22.2 Searching the MIB Tree

MIB Explorer's MIB Tree can be searched by *regular expressions*. The standard Search Panel help you to find MIB nodes and MIB object instances by text and regular expression.

Figure 19: Standard Search Panel.

The left most button provides access to the search history which saves the last ten search expressions. The search expression is entered into the text field and search immediately starts from the root of the MIB Tree after a character has been entered.

A node whose SMI text (or instance value) matches the given regular expression will be selected. With the Find Again menu item or button you are then able to find the next node that matches the expression.

To Find a Node:

1. Enter a search term or regular expression into the text field of the “Standard Search Panel.” on page 149. The search operation starts immediately when a character is entered from the root node (and top level row of the “Browse Tab” on page 56).

2. Alternatively, choose Find from the Edit menu or press  from the main tool bar. The search dialog will be displayed.
3. Enter the search expression in regular expression syntax (see "Regular expression syntax characters with special meaning." on page 32 for details).
4. Select whether case should be matched or not.
5. Only available with the Find dialog:
   Select what type of attributes of a node you want to be matched against the search expression. Choosing All will match the whole SMI text of a MIB object node, including key words, or the properties rendered as "`key= value`" node against the given search expression.

To Find the Next Node:

Choose Find Again from the Edit menu or press  from the main tool bar. The next node in depth first search order from the currently selected node will be searched, that matches the previously specified search expression and options.
Using the down arrow button from the Search Panel will also find the next occurrence.

To Find the Previous Node:

Choose the Up arrow button on the Search Pane

To Clear Search Expression:

Press the red button right of the search text field.

22.3 Identifying Duplicate OIDs

It could be problematic and it is not desirable for the code generation if an object identifier (OID) is not unique within the set of generated MIB objects. To avoid such a situation, MIB Explorer can list the duplicate OIDs of the loaded MIB modules in a table. From the Tools menu, choose Duplicate OIDs to open this list.

22.4 Extracting SMI from RFC documents

SMI MIB module definitions are embedded in IETF RFC documents which also includes page headers within the module text. This extraction tool can read a RFC file or a directory of RFC files to extract any embedded SMI modules and save them into new files.

To Extract SMI Modules from a RFC document:

1. Choose Extract SMI from RFC from the Tools menu.
2. Choose a source file or a source directory.
3. Choose a target file if you have chosen a source file or choose a target directory if have chosen a source directory.
4. Press the Ok button to run the extraction. A progress dialog will open where you can also cancel the operation if more than one file is being processed.

If two directories are specified, then the target file name is build from the source file name by appending „.smi". If such a file exists already, then „-<n>.smi" is appended where <n> is counted up from 1 to 999 until such a file does not exists.

23 MIB Compiler Error Messages

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
0000	File open error: <X>.	The file <X> could not be read, please check access rights.
0010	The length of identifier <X> exceeds 64 characters (RFC 2578 §3.1, §7.1.1, §7.1.4).	It is recommended to use only identifiers with a length of less than 32 characters for interoperability issues. Identifiers that exceed 64 characters in length must be avoided.
0050	Encountered lexical error at …	The encountered character is not allowed in a SMI MIB module.
1000	Syntax error: Encountered „token1" at row r, column c, expected one of the following: ...	The parser encountered a string it did not expect. Please look at the list of expected tokens carefully in order to determine the trouble cause. If the parser complains about a SMIv2 keyword like MAX-ACCESS, please check whether the first statement after the IMPORTS clause is a MODULE-IDENTITY definition. This is a requirement for a SMIv2 MIB module (RFC2578 §3).
1001	The DISPLAY-HINT clause value „token1" at row r, column c is invalid (RFC 2579 §3.1).	The DISPLAY-HINT clause does not correspond to any of the allowed formats for INTEGER or OCTET STRING base types.

Table 15: MIB Explorer SMI compiler error messages.

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
1002	The UTC time value "token1" at row r, column c does not match the mandatory format `YYMMDDhhmmZ` or `YYYYMMDDhhmmZ` (RFC 2578 §2)	The UTC time value does not correspond to the format `YYMMDDhhmmZ` or `YYYYMMDDhhmmZ` where `YY` - last two digits of year (1900-1999 only) `YYYY` - last four digits of the year (any year) `MM` - month (01 through 12) `DD` - day of month (01 through 31) `hh` - hours (00 through 23) `mm` - minutes (00 through 59) `Z` - denotes GMT (the ASCII character Z)
1020	Identifier <X> is ambiguous (RFC 2578 §3.1).	The identifiers (descriptors) in a MIB module must be unique.
1050	The clause <X> is not allowed within this context.	There are several clauses in SMI that are optional, but if specified those clauses need to be consistent with other clauses in the object definition. Examples for such clauses are the ACCESS, MIN-ACCESS, and SYNTAX clauses in MODULE-COMPLIANCE constructs, which must not be present for variations of NOTIFICATION-TYPEs.
1100	Imported MIB module <X> unknown.	The MIB module <X> could not be found in the MIB repository and neither in the MIB modules being compiled. Make sure that the MIB module name is not misspelled (this is often the case for older SMIv1 MIBs).
1101	Imported MIB module <X> contains a circular import.	The MIB module <X> imports from a module that either imports itself from <X> or any other module in the import chain imports from a preceding module.
1102	MIB module <X> is imported more than once.	The ASN.1 rules about IMPORTS that SMI is based on require that an import source is defined not more than once in a module.

Table 15: MIB Explorer SMI compiler error messages.

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
1110	<X> imported from MIB module <Y> must be imported from <Z> instead.	For historical reasons, SMI requires to import the MACRO definitions SMI is based on from some ASN.1 modules. For SMIv1 and SMIv2 it is defined which MACRO (construct) is imported from which ASN.1 module. Since those ASN.1 modules (e.g. SNMPv2-SMI) are not SMI themselves, the MACRO definitions have to be removed in order to be able to compile them.
1111	Missing import statement for <X> (RFC 2578 §3.2).	To reference an external object, the IMPORTS statement must be used to identify both the descriptor and the module in which the descriptor is defined, where the module is identified by its ASN.1 module name.
1112	Imported object <X> is not defined in MIB module <Y>.	Use the Edit>Search MIB Repository to search for the MIB module that defines <X>.
1113	Object <X> is imported twice from MIB module <Y>.	An object definition shall only be imported once from a MIB module.
1114	<X> cannot be imported (RFC 2578 §3.2).	Notification and trap type definitions as well as SEQUENCE constructs cannot be imported by other MIB modules.
1150	Wrong module order within file.	The MIB file that failed to compile contains more than one MIB module and the order of those MIB modules does not correspond with their import dependencies.
1200	The SYNTAX clause of the columnar OBJECT-TYPE definition <X> does not match with the SYNTAX clause of the corresponding SEQUENCE definition.	The object <X>'s syntax differs in a SEQUENCE definition from its OBJECT-TYPE definition.

Table 15: MIB Explorer SMI compiler error messages.

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
1202	The OBJECT-TYPE <X> has inconsistent maximum access (RFC 2578 §7.3).	An object <X> has a MAX-ACCESS or ACCESS clause that does not match its context (RFC 2578 §7.3). For example, a columnar object must not have a MAX-ACCESS value of "read-write" if any other columnar object in the table has a MAX-ACCESS value of "read-create".
1210	The conditionally GROUP clause <X> must be absent from the corresponding MANDATORY-GROUPS clause (RFC 2580 §5.4.2).	A conditionally group cannot be mandatory at the same time!
1211	OBJECT variation <X> must be included in a GROUP or MANDATORY-GROUPs reference (RFC 2580 §5.4.2).	The object reference <X> must be part of any object group specified as conditionally or mandatory for this compliance module.
1212	Only 'not-implemented' is applicable for the ACCESS clause of the notification type variation <X> (RFC 2580 §6.5.2.3).	If the notification has to be implemented, then the ACCESS clause should be removed.
1220	The CREATION-REQUIRES clause of variation <X> must only be present for conceptual row definitions (RFC 2580 §6.5.2.4).	The CREATION-REQUIRES clause must not be present unless the object named in the correspondent VARIATION clause is a conceptual row, i.e., has a syntax which resolves to a SEQUENCE containing columnar objects.
1221	Only columnar object type definitions with ‚read-create' access may be present in the CREATION REQUIRES clause of variation <X> (RFC 2580 §6.5.2.4).	Other objects and columns cannot be created and thus they cannot participate in a row creation.
1500	Undefined syntax(es): <X>[,…]	The syntax (data type) <X> is not defined in the parsed MIB module and it is not imported from another MIB module. Use the Edit>Search MIB Repository function to search the MIB repository for object name <X> and add the corresponding IMPORT FROM clause for <X>.

Table 15: MIB Explorer SMI compiler error messages.

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
1501	Undefined object(s): <X>[,…]	The object name <X> is not defined in the parsed MIB module and it is not imported from another MIB module. Use the Edit>Search MIB Repository function to search the MIB repository for object name <X> and add the corresponding IMPORT FROM clause for <X>.
1502	The object <X> must be defined or imported (RFC 2578 §3.2).	The object <X> is not defined in the parsed MIB module and it is not imported from another MIB module. Use the Edit>Search MIB Repository function to search the MIB repository for object name <X> and add the corresponding IMPORT FROM clause for <X>.
1600	The object definition <X> references a <Y> definition, expected a reference to an OBJECT-TYPE conceptual row definition instead.	The AUGMENTS clause, for example, requires that the referenced object definition is a conceptual table definition, i.e., has a syntax which resolves to a SEQUENCE containing columnar objects.
1601	The GROUP clause <X> references a <Y> definition, expected a reference to an OBJECT-GROUP or NOTIFICATION-GROUP instead (RFC 2580 §5.4.2).	The GROUP clause requires a reference to an object group definition.
1602	The object reference <X> points to a <Y> definition, expected a reference to an OBJECT-TYPE or NOTIFICATION-TYPE definition instead.	The VARIATION clause, for example, requires a reference to an OBJECT-TYPE or a NOTIFICATION-TYPE definition.
1700	Object reference(s) with wrong type: <X> (expected <Y> but found <Z>) [,…]	The referenced to object <X> must be of type <Y> but it is of type <Z>.

Table 15: MIB Explorer SMI compiler error messages.

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
1800	The SEQUENCE clause of the table entry definition <X> does not match the order or number of objects registered for that table at entry <Y>.	The column references in the SEQUENCE definition of a table must be lexicographically ordered by their object-identifiers. The object name Y is the name of the first object reference in the SEQUENCE definition that does not match the order of columnar objects of that table.
1810	The OBJECT-TYPE <X> has an invalid index definition (RFC 2578 §7.7).	The OBJECT-TYPE <X> has an invalid INDEX clause, i.e., an empty clause.
1811	The OBJECT-TYPE <X> has invalid index definition because <Y> may be negative (RFC 2578 §7.7).	Index values have to be encoded as OID suffixes on the wire. Since OID sub-identifiers are 32-bit unsigned integer values, negative values cannot be encoded over the wire. See RFC 2578 §7.7 for more details.
1812	The OBJECT-TYPE <X> has an invalid index definition (RFC2578 §7.7) because the minimum total index length exceeds 128 which is the maximum SNMP OID length.	Use length limitation in OCTET STRING and other variable length index objects that limits the total size to less than 128 sub-identifiers.
1813	The OBJECT-TYPE <X> has an invalid index definition (RFC2578 §7.7) because the sub-index with the IMPLIED length can have a zero length.	Index objects with IMPLIED length, must have a non-zero length. Use explicit length with a extra length sub-identifier to represent objects which have to support zero length, instead of IMPLIED length.
1850	The OBJECT-TYPE <X> has invalid index definition, because <Y> is not a columnar object (RFC 2578 §7.7).	The OBJECT-TYPE <X> has an invalid INDEX clause, because <Y> does not refer to a columnar OBJECT-TYPE definition. An OBJECT-TYPE is columnar object, if it is part of a table definition. See RFC2578 §7.7 for more details.
1851	OBJECT-TYPE definition <X> is a scalar and therefore it must not have an INDEX clause (RFC 2578 §7.7).	Scalar objects have a fixed instance identifier ("index") of '0', thus an INDEX clause must not be specified.
2000	Duplicate object registration of <X> after <Y> for the object ID <Z> (RFC 2578 §3.6).	Once an object identifier has been registered* it must not be reregistered.

Table 15: MIB Explorer SMI compiler error messages.

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
2010	Illegal object registration of <X> under <Y> for the object ID <Z>.	For example, it is not legal to register objects in the sub-tree of an OBJECT-TYPE registration.
3000	The default value of OBJECT-TYPE <X> is out of range (RFC 2578 §7.9).	The values specified in a DEFVAL clause have to be valid values for the corresponding data type syntax.
3001	The size of the default value of OBJECT-TYPE <X> is out of range (RFC 2578 §7.9).	The length of the specified octet string exceeds the SIZE constraints defined for the corresponding data type syntax.
3002	The format of the default value of OBJECT-TYPE <X> does not match its syntax (RFC 2578 §7.9).	The value <X> is not properly defined for the corresponding syntax.
3003	A DEFVAL clause is not allowed for OBJECT-TYPE <X> which has a base syntax of Counter (Counter32 or Counter64) (RFC 2578 §7.9).	Either change the syntax type to non Counter type (if the MIB has not been released yet) or remove the DEFVAL clause.
4000	The syntax definition of the object <X> is not a valid refinement of its base syntax (RFC 2578 §9).	A refinement must not extend the range of valid values for a data type.
4010	The range restriction is invalid because …	The lower bound (first value) of range restriction must be less or equal than the corresponding upper bound (second value). In addition, bounds for unsigned values cannot be negative.
4100	The TEXTUAL-CONVENTION definition <X> must not have a DISPLAY-HINT clause because its SYNTAX is OBJECT IDENTIFIER, IpAddress, Counter32, Counter64, or any enumerated syntax (BITS or INTEGER) (RFC 2579 §3.1)	Only textual conventions for INTEGER and OCTET STRING base types may have a DISPLAY-HINT clause.
4101	The DISPLAY-HINT clause value „token1“ of the TEXTUAL-CONVENTION definition <X> is not compatible with the used SYNTAX (RFC 2579 §3.1)	The integer DISPLAY-HINT format must be used with the INTEGER base type only whereas the string DISPLAY-HINT format must be used with OCTET STRING base type only.

Table 15: MIB Explorer SMI compiler error messages.

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
5000	The object definition <X> must be included in an OBJECT-GROUP or a NOTIFICATION-GROUP definition respectively (RFC 2580 §3.1 and §4.1).	This requirement ensures that compliance statements for a MIB module can be written.
5100	Object group <X> must not reference OBJECT-TYPE <Y> which has a MAX-ACCESS clause of not-accessible (RFC 2580 §3.1).	Only accessible objects and notifications may be included in object groups.
5101	The OBJECTS clause of NOTIFICATION-TYPE <X> must not reference OBJECT-TYPE <Y> which has a MAX-ACCESS clause of 'not-accessible' (RFC2578 §8.1)"	It is impossible for an agent to implement View Access Control Model (VACM) correctly and sending an object which has a maximum access of 'not-accessible'.
6000	The PIB-INDEX clause of OBJECT-TYPE definition <X> does not reference a columnar object with an 'InstanceId' syntax (RFC3159 §7.5)	Create an columnar object with SYNTAX `InstanceId` and reference it in this PIB-INDEX clause or change the SYNTAX of the referenced OBJECT-TYPE to `InstanceId`.
6001	The PIB-TAG clause present in <X> must be absent because the SYNTAX is not 'TagReferenceId' (RFC3159 §7.11)	Remove the PIB-TAG clause or change the OBJECT-TYPE SYNTAX clause to `TagReferenceId`.
6002	The PIB-REFERENCES clause present in <X> must be absent because the SYNTAX is not 'ReferenceId' (RFC3159 §7.10)	Use the PIB-REFERENCES only if the SYNTAX is `ReferenceId`.
6003	A PIB-TAG clause must be present in <X> because its SYNTAX is 'TagReferenceId' (RFC3159 §7.11)	The PIB-TAG clause is mandatory if the SYNTAX is `TagReferenceId`.
6004	The PIB-REFERENCES must be present in <X> because its SYNTAX is 'ReferenceId' (RFC3159 §7.10)	The PIB-REFERENCES is mandatory if the SYNTAX is `ReferenceId`.
6005	The UNIQUENESS clause of OBJECT-TYPE definition <X> must not contain the attribute <Y> referenced in the PIB-INDEX clause (RFC3159 §7.9)	Do not include the PIB-INDEX attribute in an UNIQUENESS clause.

Table 15: MIB Explorer SMI compiler error messages.

ERROR #	MESSAGE	DESCRIPTION/SOLUTION
6006	The UNIQUENESS clause of OBJECT-TYPE definition <X> must not contain the attribute <Y> more than once (RFC3159 §7.9)	There must not be duplicate attributes in an UNIQUNESS clause.
6007	The INSTALL-ERRORS clause of OBJECT-TYPE definition <X> has an invalid error number <N> for label <L> which is out of the range 0-65535 (RFC3159 §7.4)	Error numbers in a INSTALL-ERRORS clause must be between 0 and 65535.

Table 15: MIB Explorer SMI compiler error messages.

*. An object registration is any object definition other than OBJECT-IDENTIFIER.

24 Trouble Shooting

Although MIB Explorer has been designed to make dealing with SNMP and MIBs straightforward, there are some situations where MIB Explorer cannot solve a problem without help from the user. The following hints should help you to cope with such exceptional incidents.

PROBLEM	SOLUTION
MIB files cannot be compiled although other SNMP tools accept them.	In almost all such cases, the MIB module that fails to compile has a serious syntax error, thus, an error that clearly violates a SMI rule. If you are not able to fix the MIB module with the built-in MIB editor, you may switch the MIB compiler into lenient mode and recompile the module. Caution: Using lenient MIB compilation may cause problems later because incorrect or inconsistent data may have been read. If you encounter problems with MIB Explorer, please make sure that all your MIBs have passed normal compilation.
The chart view of a monitor does not update correctly when data collection is performed by expressions only.	*Workaround:* Add a (dummy) row that collects an arbitrary numerical value (e.g. `sysUpTime`) and uncheck the Display box in the configuration of the new row to avoid displaying the dummy value. The chart update will be then correct.
The axis labels of 3D charts are not correctly rendered when saved as `PDF`, `PS`, or `PCL` files.	This is a known issue and there is currently no workaround other than using the `JPEG` or `PNG` data types instead.
Timeout in SNMP Table view or a `tooBig` error status is returned when refreshing a table, but single columns of the table can be retrieved without problems.	The packet size of the response might be too big. Try reducing the "Max VBs per PDU" value or increasing the "Maximum inbound message size" value of the transport mapping used in Preferences. If the target is not a SNMPv1 target, then reducing the "Get Bulk repetitions" value will also help to reduce the size of the response PDU send back by the target.

Table 16: MIB Explorer trouble shooting hints.

PROBLEM	SOLUTION
Determining a target's MIB set detects MIB modules that are not implemented by the agent.	The agent does not implement lexicographic ordering correctly. Enable logging with a log table size of at least 100 entries and set the logging priority for SNMP to DEBUG. Verify the responses got from the target for correct lexicographic ordering. Alternatively, you can also use the Packet Analyzer to inspect the responses sent by the agent.
Sometimes timeout on a specific target.	Timeout or retries value to small. First, try to increase the timeout value in Target Preferences. If this does not help, try to increase the "Number of retries" value.
Timeout on any type of request for a specific target.	Wrong community or USM user specified for the target. Change the community or the user settings in Target Preferences.

Table 16: MIB Explorer trouble shooting hints.