



EMC[®] Watch4net[®]

Version 6.4u2

MIB Browser User Guide

P/N: pending

Rev : 42070

EMC²

Copyright ©2010 - 2014 EMC Corporation. All rights reserved. Published in the USA.

Published October, 2014

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC², EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to EMC Online Support (<https://support.emc.com>).

Contents

Preface	iii
Intended Audience	iii
Document Organization	iv
Where to get help	v
1 Installation	1
1 Prerequisites	1
1.1 License	1
1.2 Installation	2
1.3 Global Configuration	2
2 MIB Browser Interface	5
1 Overall View	6
2 Header Section	8
2.1 Export	8
2.2 Walkfiles	9
2.3 Options	11
2.4 MIB Files	17
3 Tree Section	22
3.1 Structure	22
3.2 Search	24
4 MIB Information	25
5 SNMP Agent	26
5.1 Agent Configuration	26
5.2 Ping	29
5.3 SNMP Commands	30
5.4 Real time SNMP Polling	32
6 Content Section	34
6.1 Value Translation	34
6.2 Available Views	34

6.3 Graphing Values	38
3 Polling Configuration	41
1 Configurations	42
2 SNMP Poller	44
2.1 Communication	45
2.2 Threads	47
2.3 Miscellaneous	49
3 Polling Groups	49
3.1 Settings	52
3.2 SNMP Agents	52
3.3 SNMP Masks	54
4 SNMP Masks	55
4.1 SNMP Masks Import	58
4.2 Masks list	58
4 Table polling configuration	67
1 Basic edit mode	68
2 Advanced edit mode	71
5 Agents configuration	83
1 SNMP Agents	86
2 IP Pattern Grouping	89
3 Explicit Grouping	90
6 Synchronization	95
1 SNMP Poller	96
2 Configuration Archival	99
3 MIB Files	100
A Context File Sample	101

Preface

The purpose of this guide is to provide information about *Watch4Net Solutions's MIB Browser* installation and usage, including the collector configuration if the required license has been installed.

Intended Audience

This guide is intended for all *MIB Browser* end-users, who will use the application to browse MIB files and send SNMP queries to devices. Targeted audience also includes system integrators and administrators who will install and set-up the *MIB Browser* application and use the Polling configuration to manage SNMP Collectors and add, edit or remove polled metrics from the collecting configuration.

Document Organization

1. Installation	This section shows the <i>MIB Browser</i> installation procedure and environment configuration.
2. <i>MIB Browser</i> Interface	This section describes the <i>MIB Browser</i> graphical user interface and all of its components.
3. Polling Configuration	This section describes the SNMP poller configuration interface.

Where to get help

EMC support, product, and licensing information can be obtained as follows:

PRODUCT INFORMATION - For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

<https://support.emc.com>

TECHNICAL SUPPORT - Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

1

Installation

1 Prerequisites

1.1 License

Two separate valid licenses must be installed to use the *MIB Browser* and *SNMP Poller* configuration components.

- The “*MIB Browser*” license allows access to the *MIB Browser* section of this application.
- The “*SNMP Poller*” license allows access to the Polling configuration section of this application.

Typically, both licenses are sold as a unit but, in some cases, only one is provided, therefore allowing access to a specific section of the application.

1.2 Installation

The Mib Browser / SNMP Polling configuration application is installed by default when installing APG, using the "Default" installation mode.

However, the application may also be installed manually by installing the mib-browser package, along with its required dependencies which normally include the following packages:

- tomcat;
- license-manager;
- webapps-resources.

Note: The *MIB Browser* is normally used in conjunction with the SNMP Collector, as it can manage its configuration and create/edit polling configurations (SNMP masks). Installing a specific SNMP Collector test instance is highly desirable as it allows testing trial configurations without impacting data collection on production systems.

1.3 Global Configuration

The *MIB Browser* application may require some configuration in order to work properly. Some default values have already been setup, but they might require customizing to fit APG's installation.

The `APG/Web-Applications/Mib-Browser/mib-browser/conf/common.properties` contains the variables which may be set for this application. The mib file repository directory may be changed in this file by specifying a valid path for the `snmp.repository.dir` variable.

The `APG/Web-Servers/Tomcat/Default/conf/Catalina/localhost/mib-browser.xml` file contains other settings such as the user database access configuration, the walk file storage path and the ping command path. By default, the `PING` system command is assumed to be in the user executable `PATH`. If that is not the case, the absolute path to the executable, including its name (normally: `PING`) must be provided here.

The `mib-browser.xml` file becomes available when the application is deployed for the first time.

WARNING: The APG Tomcat service must be stopped before editing the `mib-browser.xml` file.

Note: Make sure their ownership, group and/or permissions remain the same on configuration files. These files must be owned and/or readable and writable by the *APG* user.

2

***MIB Browser* Interface**

The *MIB Browser* is a web application which only requires a web browser in order to be used. Supported browsers depend upon the package release but typically include:

- Internet Explorer (8 and later);
- Firefox (18 and later);
- Chrome.

Although there are only a few noticeable differences between browsers, we recommend using Firefox for a better user experience.

Note: Firefox users who have installed debugging plugins such as `firebug` are advised to disable them as they may significantly slow down the *MIB Browser* execution.

1 Overall View

The login page is the same as *APG's*, and switching from *APG* to the *MIB Browser* will be transparent if single-sign on has been enabled. The *MIB Browser* may be accessed from either *APG* or directly, at *http://your_apg_host:58080/mib-browser*.

After the login process, the main interface of the *MIB Browser* will load and appear as displayed on figure 1. The *Tree Section*, *Content Section* and *MIB Information* section should be empty on initial load.

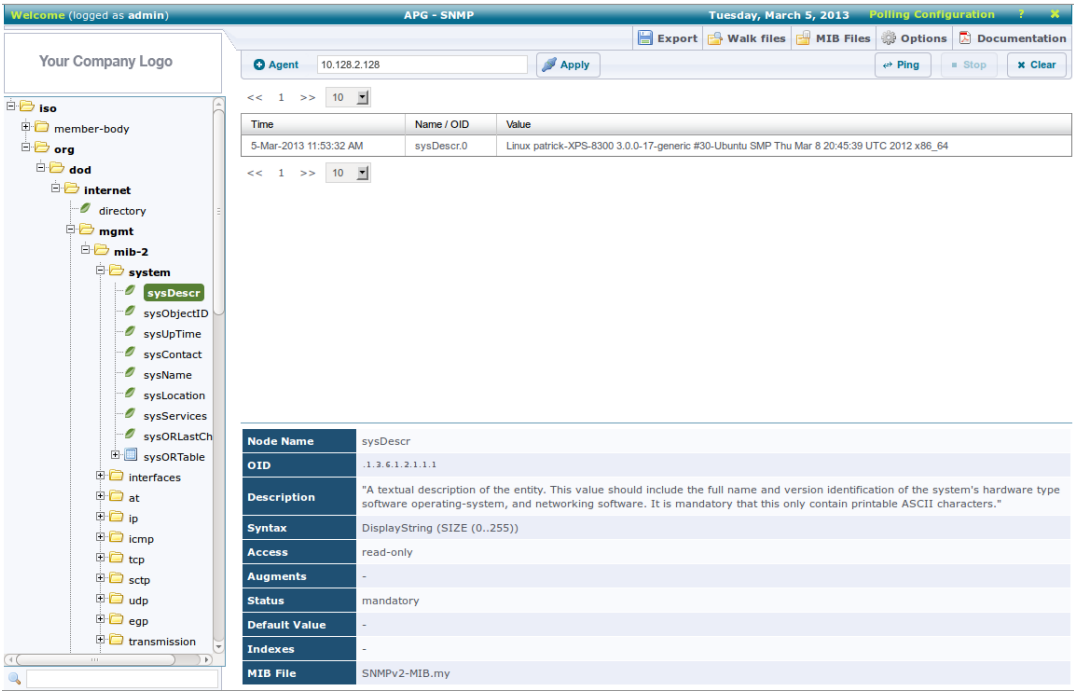


Figure 1: Main interface

There are five major sections in the page, each one with a specific role.

- The [Header Section](#) provide general information and access to specific functionality;
- The [Tree Section](#) displays loaded MIB's oids and tables and features a quick search facility;
- The [SNMP Agent](#) is used to choose/specify the target for SNMP requests / ICMP pings sent from the *MIB Browser*;
- The [Content Section](#) displays SNMP requests completion results;
- The [MIB Information](#) section displays the selected node (OID) attributes.

2 Header Section

The header section is located at the top of the interface and provides the following information and functionality:

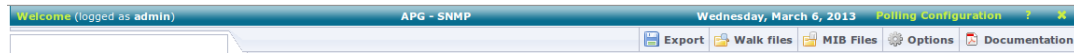


Figure 2: Header Section

The header's first row provides:

- The logged in user;
- The current date;
- A link to access the (SNMP) Polling Configuration, if available (see note);
- A link to display license information;
- A logout link (this will log you out of APG as well).

Note: The SNMP Polling configuration is accessible only if the required license is installed and the user role has access to this application.

The header's second row provides access to:

- The [Export](#) functionality;
- [Walkfiles](#) management;
- [MIB Files](#) management;
- Various *MIB Browser* configuration [Options](#);
- Documentation.

2.1 Export

The *Export* functionality is used to download the information available in the [Content Section](#) after a SNMP request was completed on the chosen agent. A file download dialog will be triggered upon activation. The downloaded file will contain the request result as well as the directory of MIB files available to the *MIB Browser* application.

2.2 Walkfiles

This dialog displays the available walk files, generated by the various "Walk on ... to Walk file" actions (see [SNMP Commands](#) and figure 3).

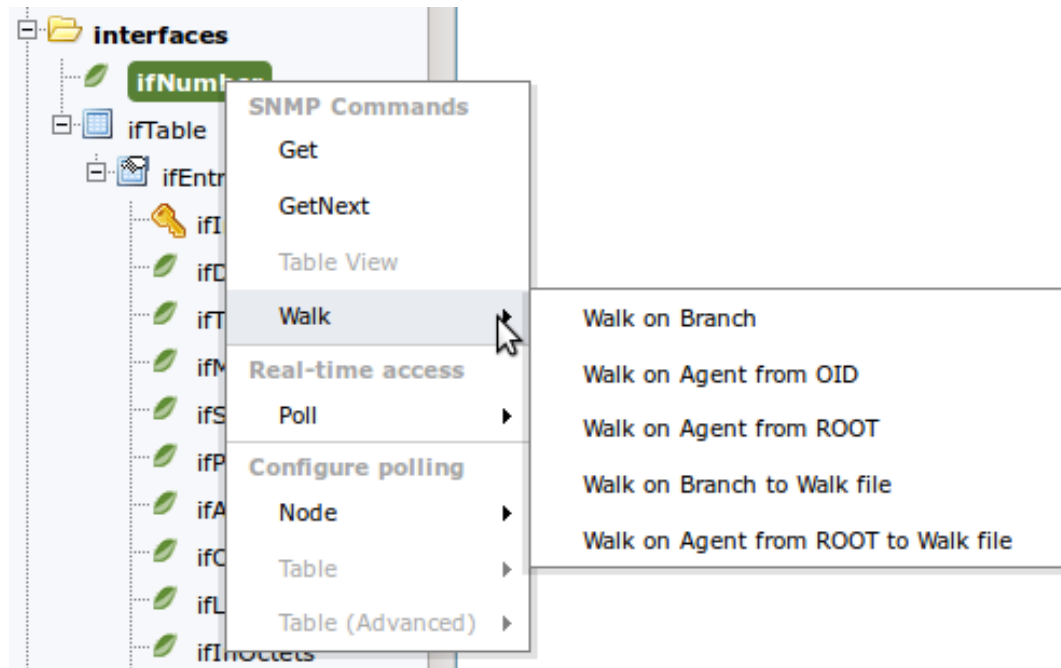


Figure 3: Walk options

These actions are done in the background and their result, instead of being displayed directly on screen, are stored as walk files.

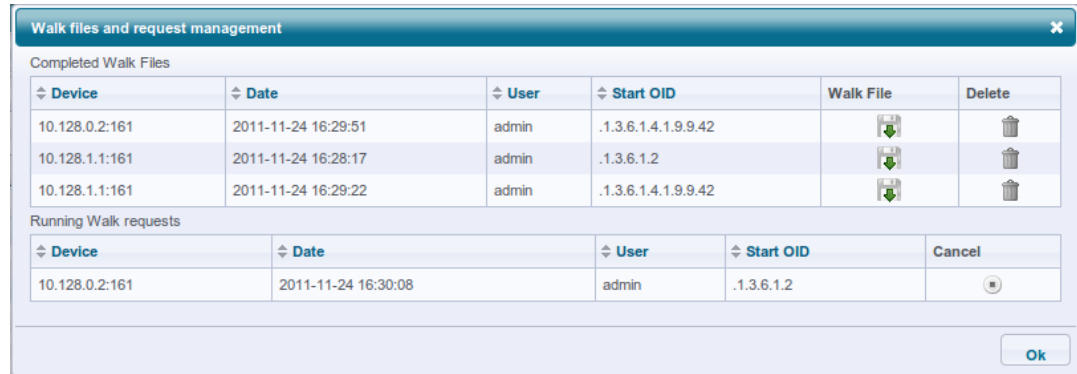


Figure 4: Walkfiles dialog

This Walkfiles dialog (4) features the following actions:

- Download a completed walk file;
- Delete a completed walk file;
- Cancel an ongoing walk action.

Note: Downloaded walk files also contain the directory of MIB files available to the *MIB Browser* application.

2.3 Options

The *MIB Browser*'s behavior can be customized through this menu.

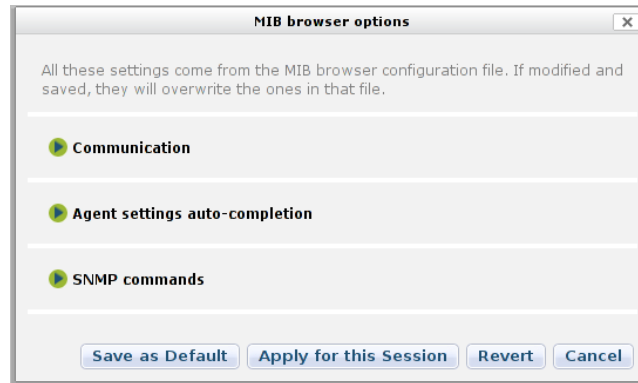


Figure 5: *MIB Browser* Options

This menu contains three major sections, that are collapsed by default as shown on figure 5 page 11. Each one deals with a specific aspect of the *MIB Browser* behavior.

The communication section deals with network interface binding and communication mode.

The "Bind to Interface" list is used to choose a specific interface to be used when communicating with SNMP agents or SNMP Collectors. The default server network interface is used when none is specified. In any case, this interface is used to communicate with SNMP agents (direct polling) or SNMP Collectors (remote SNMP agents polling, send control commands).

The *MIB Browser* can communicate with SNMP agents either directly or through an optional SNMP Collector.

The preferred mode for MIB browsing is direct communication as it reduces network bandwidth and response time. But there are cases where this is not possible:

- Server host is not registered in the SNMP agent's ACL;
- Server host does not have a route to the SNMP agent;
- SNMP agent is behind a firewall or is accessible only through a SNMP Collector.

Contacting a SNMP Collector, which will in turn poll the SNMP agent and send the back the results to the *MIB Browser* is the way to go in these cases.

See figure 6 page 13 for a graphical view of the possible communication modes.

The *Direct Connection* mode is used to poll SNMP agents directly by the *MIB Browser* while the *This Collector* mode is used to poll SNMP agents via a SNMP Collector. The address of the collector must be provided in order to use for remote polling.

Note: The collector's address format is "IP:port"

In Direct Connection mode, the Tomcat server host will be sending the SNMP commands. This means the server running Tomcat needs network access to the target SNMP Agents. Make sure a firewall or other security mechanism does not block SNMP packets (SNMP usually uses port 161).

If the Tomcat server cannot access the Agents, the *slave mode* will have to be used. In *slave mode*, a remote SNMP poller will perform SNMP queries on the target agent. This remote SNMP poller is a SNMP Collector instance, which is in fact an optional APG data collection adapter that is used to poll SNMP Agents. This collector is highly versatile and can be used to complement or even replace EMC SMARTS when doing highly specialized SNMP data collection. It also provides the means, through its API, for the *MIB Browser* to poll SNMP agents which would otherwise be unreachable from the server host. This component is required in order to use this method of communication. It is well adapted for distributed network polling.

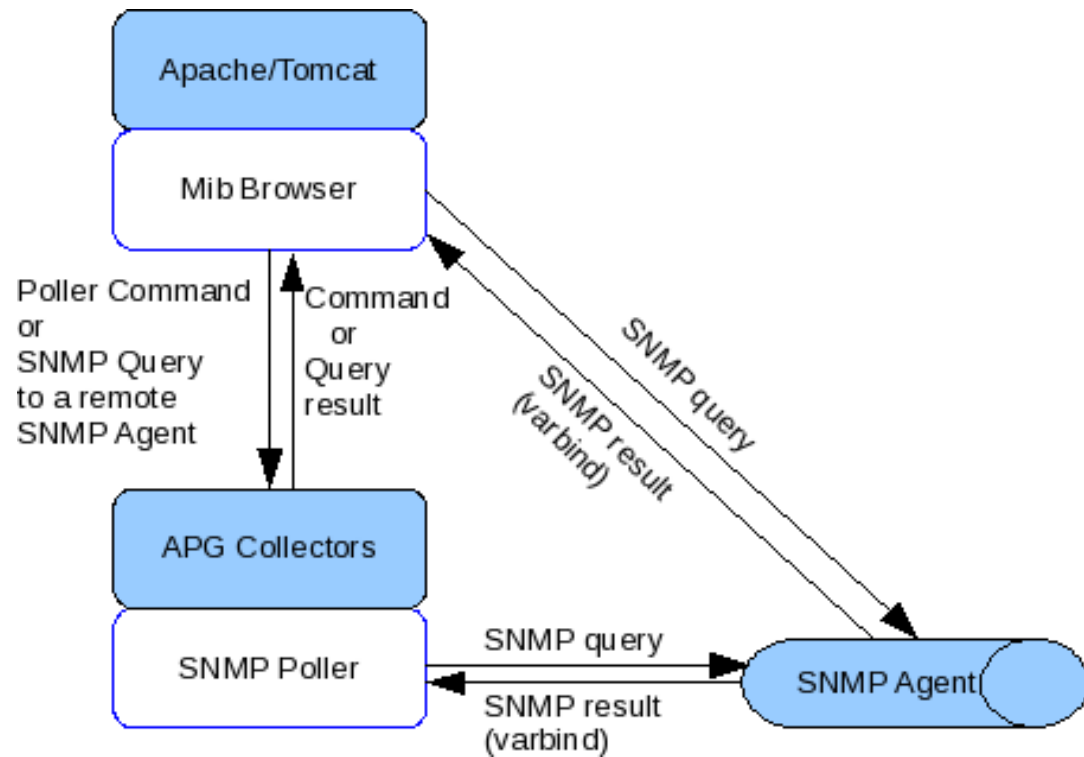


Figure 6: Communication Modes

If the SNMP configuration tool is enabled, and some SNMP slave pollers have been defined in the corresponding file, then they will be available in a suggestion box, which is displayed when the field is modified.

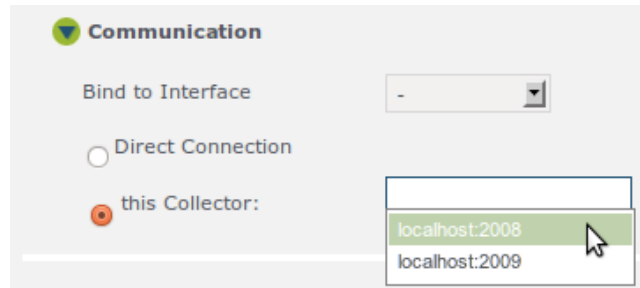


Figure 7: *MIB Browser* Communication

The SNMP Agents auto-completion section (see figure 8 on page 15) configures the location where the agent configuration helper will look for agents settings. It suggests available SNMP Agents as you type their IP addresses or hostnames in the input box. The location where SNMP Agent settings will be gathered can be setup here by checking or unchecking the different options.

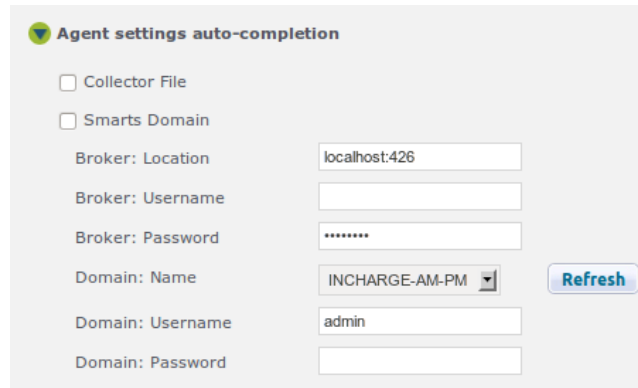


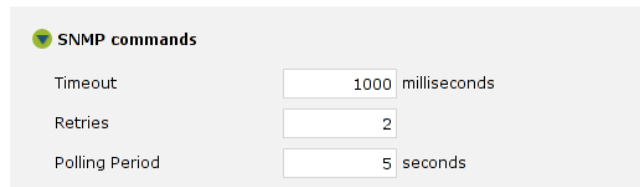
Figure 8: MIB Browser Agents Autocompletion

If `Collector File` is selected, then agents settings from the SNMP polling distribution file will be used and showed in the suggestion list.

If a `Smarts Domain` is available and well configured, then all available agents will be displayed in the suggestion list. In order to get this working, a broker needs to be defined, then a domain selected, and finally the credentials need to be provided. Please note that the *MIB Browser* support broker authentication, but if the broker to contact is not configured to use it, you simply need to let the corresponding fields blank.

Note: The `Refresh` button will query the provided broker (format is 'IP:port' and then update the list of available domains.

The specific settings (see figure 9 on page 16) used with SNMP requests are defined in the following section. It includes the `timeout`, in milliseconds, before the connection is considered unsuccessful, the number of `retries` before giving up while trying to reach an SNMP agent, and the polling period, in seconds, for any polling process launched from the GUI.

A screenshot of a dialog box titled "SNMP commands" with a green icon. It contains three settings: "Timeout" set to 1000 milliseconds, "Retries" set to 2, and "Polling Period" set to 5 seconds. Each setting has a text input field and a unit label.

Setting	Value	Unit
Timeout	1000	milliseconds
Retries	2	
Polling Period	5	seconds

Figure 9: MIB Browser SNMP Commands

Settings can be saved in 2 fashions:

- *saved as default values* : settings are persisted as default;
- *saved for the current user session* : settings are used until logging out;

Revert resets all settings to the default configuration; *Cancel* simply closes the dialog without saving changes.

2.4 MIB Files

The Management Information Base (MIB) files are an essential part of the MIB browser. No OID tree may be created without them. Furthermore, corrupted MIB files, if parsed successfully, may have unexpected effects on that same tree.

On application startup, all MIB files found in the mib repository will be parsed and valid ones will be loaded. This process will automatically generate the OID tree.

MIB validation and parsing is not overly strict, meaning that missing dependencies for a MIB file is not a fatal parsing error if they are not mandatory. However, some translations might not work as expected in such cases.

An extensive set of MIB files is provided with the application. These MIBs have been tested and successfully displayed in the *MIB Browser*. By default, they are located in `Custom/WebApps-Resources/Default/mib-files`.

The MIB file management tool displays all available MIBs, highlighting the number of loaded MIBs and the invalid ones (see figure [10](#)).

Available MIB Files

You can add MIB files using the form below, and delete or download them using the menu available on each line.

Search:

Name	Size	Last Modified	Status
ALLOT-COMPANY.mib	2.8 KB	7-Mar-2013 10:54:56 AM	
APACHE-MIB.mib	5.1 KB	5-Mar-2013 9:24:45 AM	
APENT-MIB.mib	4.8 KB	5-Mar-2013 9:24:45 AM	
APPLICATION-MIB.mib	117.4 KB	5-Mar-2013 9:24:45 AM	
ATM-MIB.mib	122.4 KB	5-Mar-2013 9:24:45 AM	
ATM-TC-MIB.my	26.9 KB	5-Mar-2013 9:24:45 AM	
avayagen.mib	3 KB	5-Mar-2013 9:24:45 AM	
BGP4-MIB.mib	36.1 KB	5-Mar-2013 9:24:45 AM	
BLACKBERRYSERVER-MIB.mib	40.7 KB	5-Mar-2013 9:24:45 AM	
BRIDGE-MIB.mib	45 KB	5-Mar-2013 9:24:45 AM	
BROCADE-SW.mib	33.2 KB	5-Mar-2013 9:24:45 AM	
CHARACTER-MIB.mib	20.5 KB	5-Mar-2013 9:24:45 AM	
checkpoint.mib	3.6 KB	5-Mar-2013 9:24:45 AM	
CISCO-ATM-CONN-MIB.mib	40.8 KB	5-Mar-2013 9:24:45 AM	
CISCO-ATM-IF-MIB.mib	19.4 KB	5-Mar-2013 9:24:45 AM	
CISCO-ATM-IF-PHYS-MIB.mib	22.2 KB	5-Mar-2013 9:24:45 AM	
CISCO-ATM-PVC-MIB.mib	8.6 KB	5-Mar-2013 9:24:45 AM	

There are 336 MIB files loaded out of 337.

Upload a MIB file

Reload MIBs

Figure 10: MIB Files Management

The safest way to update MIB files is through the provided interface. However, the import must be done one file at a time. One need to select the MIB to import using the *Browse* button, and then

click on the *Upload* button to send it to the server. This method has the advantage of telling the user whether the MIB is correct or not, and highlight non-defined element in the MIB, mostly because of a missing dependancy.

Note: Invalid MIB files are uploaded to the server but marked as invalid. OIDs defined in these files will not appear in the OID tree.

There are two ways to reload these invalid MIBs: upload them again (the recommended one) or reload all available MIB files if uploading failed.

MIB files can also be deleted or downloaded from this view. A simple context menu with those two options will appear when clicking the MIB file entry.

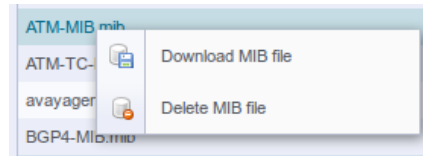


Figure 11: MIB Files Context Menu

Once MIB files have been modified, added or deleted, changes will not be taken into account immediately. Click the *Reload MIBs* button to reload all MIB files and update the MIB tree.

There is no *Cancel* button on this page, as no cancellation can be done here. In order to close the window, click on the *X* at the window's top right corner.

Note: The quickest way to rename, delete or modify MIB files in the Custom/WebApps-Resources/Default/mib-files folder is to edit them on the server filesystem, and reload them through the Mib Files list dialog. However, missing dependancies will not be notified and the guilty MIB will simply be flagged as invalid.

Make sure file permissions are correctly set after manually adding/editing a MIB file. If the server is not able to read or delete it, then the file may not be parsed or updated/removed.

Note: If the MIB files dialog is not accessible, due to a missing license, it is still possible to copy MIB files directly to Custom/WebApps-Resources/Default/mib-files : this directory is checked for changes in content every 5 minutes and MIB files will be reloaded automatically if new ones are detected. This allows using the Polling Configuration section of the application to setup report packs which uses new MIB files, without having access to the MIB Browser section.

WARNING: EMC does not recommend to edit the provided MIB files, unless it is absolutely necessary. If a new version of the MIB file is required to support newer equipment, then the old MIB file should be backed-up somewhere else before it is overwritten by the new one.

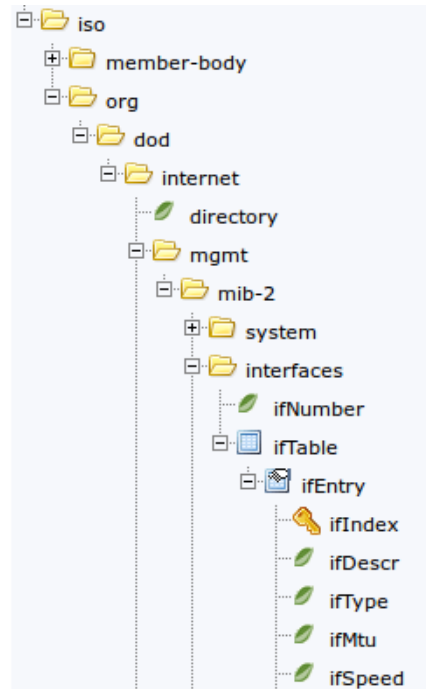
Sub directories to Custom/WebApps-Resources/Default/mib-files are ignored when loading MIB files. So it is safe to create backup directories there.

3 Tree Section

All available MIB files are loaded into memory and merged into one single tree structure. This structure can be browsed by expanding and selecting any of its nodes, each node being a specific object identifier (OID) that has been defined in the MIBs.

3.1 Structure

The tree nodes have many possible representations depending on their type. All of them are explained below:



- the *folder* icon represents a container that can be expanded in order to reveal its child nodes;
- the *table* icon highlights any SNMP table node;
- followed by the *entry* node icon of a table one;
- the *key* icon represents an index of the table it belongs to;
- the *leaf* icon is a final OID that cannot be expanded any further. When a leaf is inside a table, then this leaf represents one of this table's column.

Note: The tree panel can be resized by sliding it's right edge. However, limits have been set for such a resize. It cannot be reduced under a certain width, and the content size at its left also cannot be reduced under a certain width as well, thus preventing the tree to expand anymore. Double-clicking its edge hides the tree panel. Double-clicking it again brings it back into view.

3.2 Search

A search box, located at the bottom of the tree section, allows searching for a specific OID, by specifying either by it's numerical OID (.1.3.6.1.X) or node textual name.

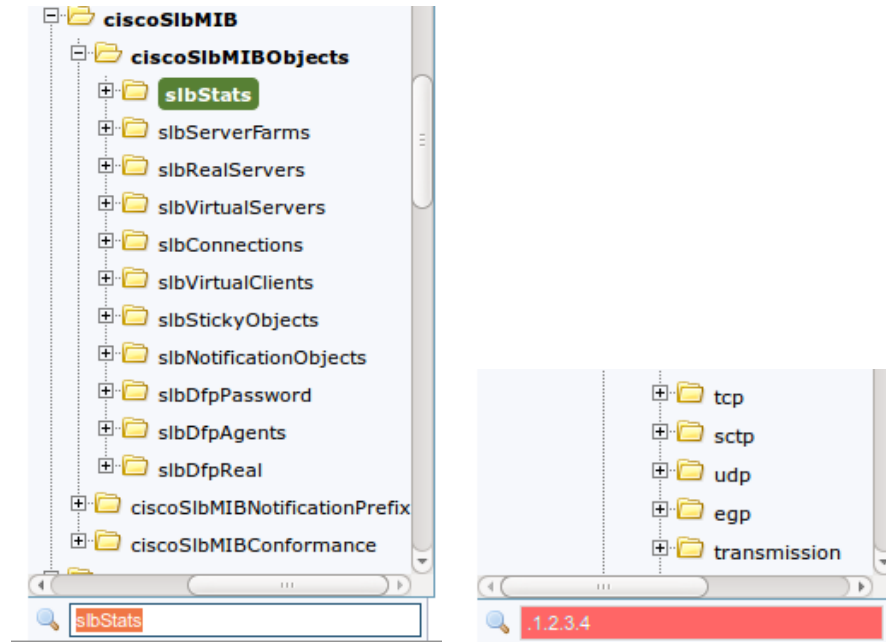


Figure 12: MIB Tree Search

Last but not least, the tree panel includes a quick search function which can be used in two different ways. The search box at the bottom of the tree accepts both names and OIDs but its behavior changes depending on which one is used.

Searching for a specific OID is done by specifying it in the search box. This OID *MUST* start with a "." (dot) and be fully specified from the tree's root OID. Most OIDs supported by SNMP Agents start with the .1.3.6.1 prefix.

This is a *search-as-you-type* mode which automatically highlights any matching node and warns the user if no matching OID was found, as shown on figure 12.

If the entered value does not start with a dot, then searching is performed on a specific name. In that case, the input name is case insensitive and wildcards such as * or ? (in a *Unix-like* fashion) may be used. The search will be performed after hitting the `Enter` key or clicking the search icon. Repeating a search with the same parameter will highlight the next matching node.

Note: When the MIB tree shows inconsistencies due to invalid/corrupted MIB files, the search function might not find the guilty OIDs as they may not be in the tree.

The figure on page 24 describes the two search modes and their associated visual representations. A successful search highlights the matching node and displays it in the middle of the tree. An unsuccessful search is indicated by a red background in the search text box.

4 MIB Information

The currently selected MIB tree node's information is displayed at the bottom of the request result pane.

Node Name	sysDescr
OID	1.3.6.1.2.1.1.1
Description	"A textual description of the entity. This value should include the full name and version identification of the system's hardware type software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters."
Syntax	DisplayString (SIZE (0..255))
Access	read-only
Augments	-
Status	mandatory
Default Value	-
Indexes	-
MIB File	SNMPv2-MIB.mib

Figure 13: MIB Tree Information

This information is updated every time a new node is selected. Information is retrieved from the MIB file which defines the node. However, this only applies for the following fields: name, OID, syntax, access, augments, status, defval, indexes, description and the last loaded MIB file that defines that OID.

Note: This information panel may be resized by sliding its top horizontal edge.

5 SNMP Agent

5.1 Agent Configuration

The SNMP Agent must be specified before attempting to communicate with it. Access information such as its ip address, port, community string must be provided before queries may be performed. Before retrieving any information from a reachable SNMP agent, this one has to be configured: hostname, port, community string...

When no agent is configured, a disconnected plug will appear near the end of the line, meaning that no SNMP command may be sent yet. See figures 14 or 15 where this case is shown.

If the auto-completion feature has been enabled in the configuration, then a list of suggested agents, taken from the configured sources will appear as the agent IP or hostname is typed in the input field. The matching entries will be highlighted in the drop down list. Any agent selected in this list will be automatically configured using the settings provided by an EMC Smarts domain or the snmp-polling-distribution.xml distribution file.

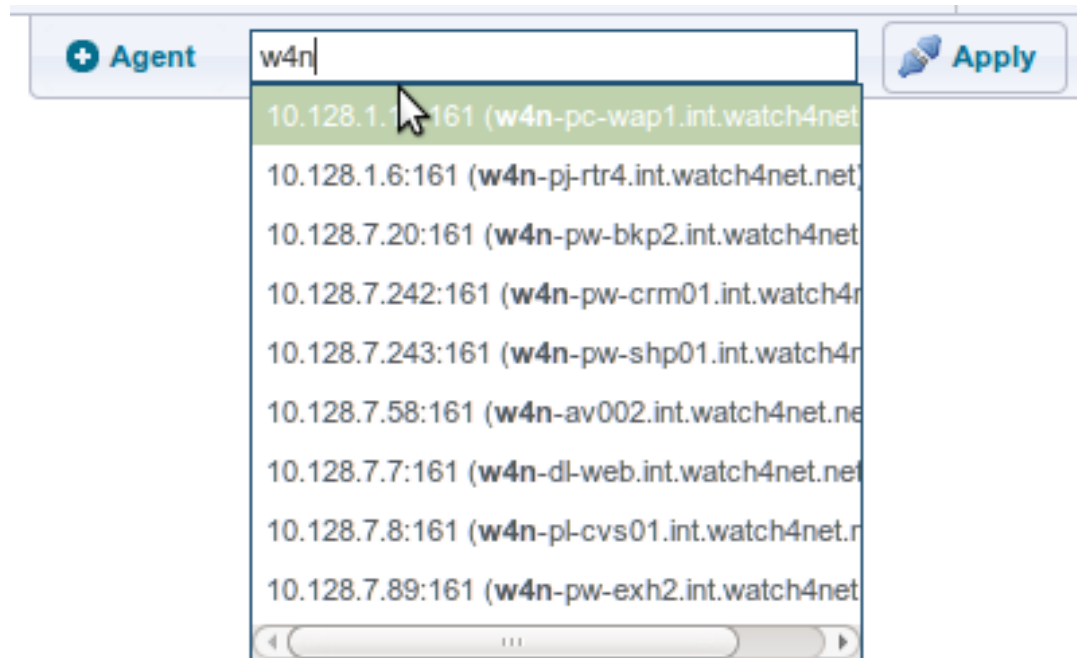
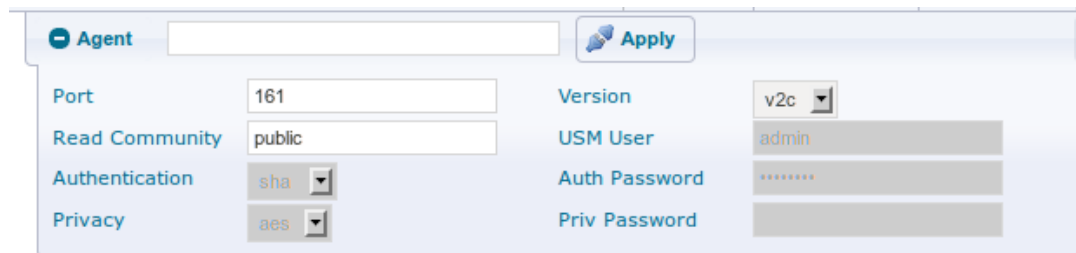


Figure 14: SNMP Agent Auto-Completion

The target SNMP Agent may also be specified by entering manually all required information. In order to display this panel (see figure 15), click on the *Agent* or on the + icon at its left. There, device IP, SNMP port, SNMP version, Read Community String and SNMPv3 settings can be filled. The main input box expects either the device IP or *hostname* if it can be resolved. All other fields have been filled with standard default values, except for SNMPv3 settings.



The figure shows a web interface for configuring an SNMP Agent. At the top, there is a label 'Agent' with a minus icon and a text input field. To the right of the input field is an 'Apply' button with a rocket icon. Below this, the settings are organized into two columns. The left column contains: 'Port' with a text input field containing '161'; 'Read Community' with a text input field containing 'public'; 'Authentication' with a dropdown menu showing 'sha'; and 'Privacy' with a dropdown menu showing 'aes'. The right column contains: 'Version' with a dropdown menu showing 'v2c'; 'USM User' with a text input field containing 'admin'; 'Auth Password' with a text input field containing masked characters; and 'Priv Password' with a text input field containing masked characters.

Figure 15: *SNMP Agent* Settings

After the agent has been selected or manually setup, hitting the **Enter** key or the **Apply** button will send the configuration to the server. If all fields have been filled, the SNMP agent will be set in *configured* mode (see figure 16). SNMP queries may now be sent to this SNMP Agent.



The figure shows a web interface for a configured SNMP Agent. It features a label '+ Agent' with a plus icon, followed by a text input field containing the IP address '10.128.2.128'. To the right of the input field is an 'Apply' button with a rocket icon.

Figure 16: Configured *SNMP Agent*

Note: When the agent has been set in *configured* state using the auto-completion feature, manual settings will not be updated.

The connected plugs mean that all settings have acceptable values. However, this does not mean that the provided settings are correct for the specified SNMP Agent.

Queries will timeout on SNMP agents if:

- The community string is incorrect (v1, v2c only; v3 agents will reply with an error varbind);
- The port is incorrect (agent is not listening on this port);
- The agent is unresponsive due to internal device problems;
- v2c queries are sent to a v1 agent;
- v3 queries are sent to a v1 or v2c agent.

5.2 Ping

The Ping command (figure 17) may be used to check reachability, from the server standpoint.



Figure 17: Ping Command

If device is up and reachable, then the device's SNMP Agent might be down if not responding to correct credentials, or its configured access lists may reject SNMP queries coming from the server. Firewall rules may also prevent the server from communicating directly with the SNMP Agent.

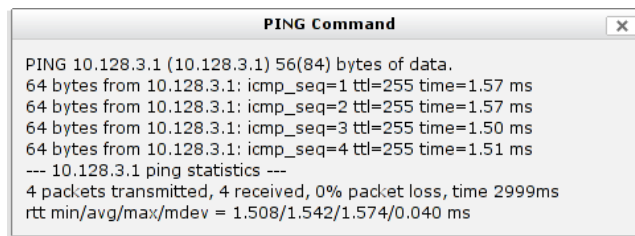


Figure 18: Ping results

Note: The ping command is executed on the server itself, targetted at the specified SNMP agent. Alternately, reachability may also be checked from the SNMP Collector by clicking the reachability test button displayed on the agent configuration summary line. See section about [86](#) for details. In many cases, agents ACL are setup to allow SNMP queries coming from specific hosts running collectors, not front ends, so using the indirect reachability check, through a collector, is sometimes the only way to check the device reachability status.

5.3 SNMP Commands

Once the agent has been properly configured, SNMP commands can be sent via the contextual menu available on each tree node (see figure [19](#) for a detailed view). In this section, we will focus on SNMP commands only.

- **GET** retrieves a single value. It can only be performed on a terminal leaf, which is not a table column.
- **GET NEXT** retrieves the next OID's value. The base OID is the one taken from the previous GET or GET NEXT command. In order to reset it to a specific OID, a GET command on that OID has to be issued.
- **TABLE VIEW** retrieves all data from the selected table. This command can be performed on any *table*, *table entry* or *table column*, with the same effect on any of these nodes. The data will then be displayed in a table format which mimics the SNMP Agent table's content at the request's time.
- **WALK** performs a SNMP walk on the agent values from the given starting point. By definition, a SNMP walk retrieves a subtree of management values using SNMP GET NEXT requests. The following modes are supported:
 - **ON BRANCH** will start from the node on which the context menu has been called and returns the values from all its sub-nodes.
 - **FROM OID** will start from the node on which the context menu has been called, and walk until the end of the MIB is reached
 - **FROM ROOT** will start at the beginning of the MIB (OID .1) and walk until the end of the MIB is reached

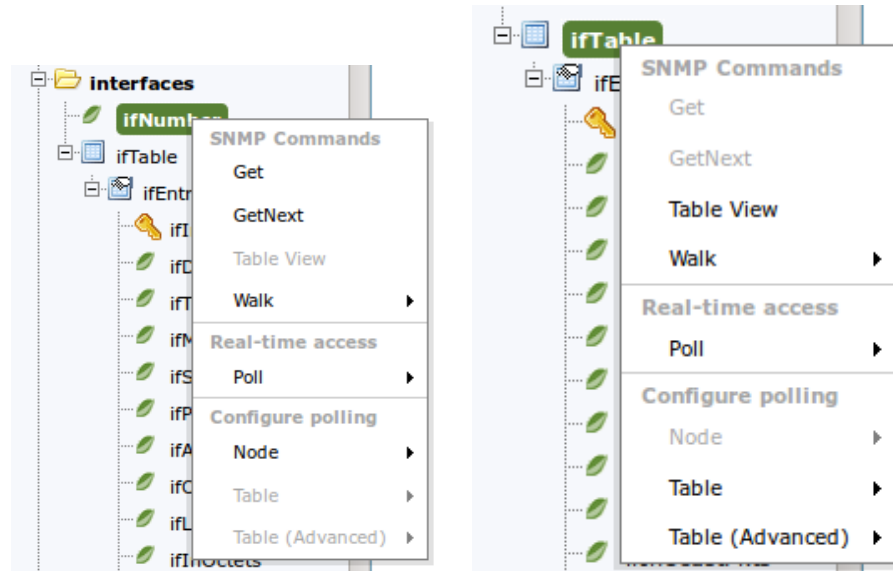


Figure 19: Contextual Menu on the Tree: node oid and table

- **ON BRANCH TO WALKFILE** will perform a background walk, starting for the selected node and retrieving all of its subnode values. The result of this action will be stored in a walk file, accessible through the Walkfile dialog (See 2.2).
- **ON AGENT FROM ROOT TO WALKFILE** will perform a background walk, starting from the SNMP agent root node all the way to its last available node. The result of this action will be stored in a walk file, accessible through the Walkfile dialog (See 2.2).

Note: Command results will appear in the main *MIB Browser* panel (see section 6 for more details).

The *Stop* button in the SNMP settings bar will stop the current command, if it is not already finished.

5.4 Real time SNMP Polling

Quick troubleshooting can be performed on a specific device by polling some of its OIDs at a high frequency to check the value's status and evolution. This *MIB Browser* can perform this type of polling. Click the MIB tree node to poll in real time, right-click it to display the contextual menu (figure 20) and choose the polling mode.

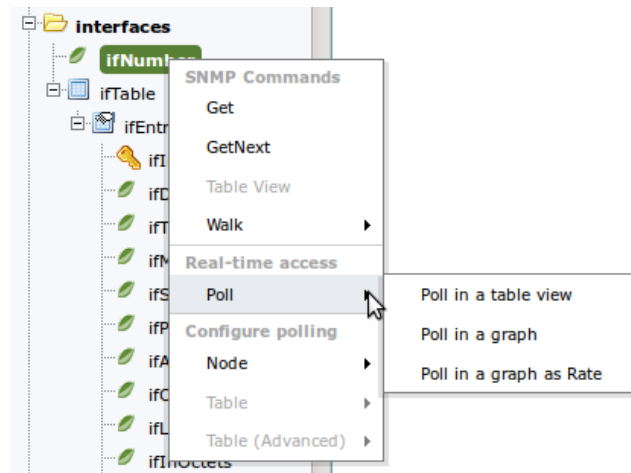


Figure 20: Real time polling mode

The OID node's contextual menu offers several SNMP Agent polling actions:

- **TABLE VIEW** retrieves all the values under the selected OID (same behavior as a walk on the chosen branch) and displays them in a table. This table is updated, the most recent values appearing at the top.
- **GRAPH** retrieves either all the values of the chosen node (for instance, all the values of the selected column) or a single value when the row id is specified. These values are displayed in a graph.
 - **COUNTER** the polled value will not be modified;
 - **RATE** a rate will be computed from the polled value and the previous one.

Note: Polled results will appear in the main *MIB Browser* panel (see section [6](#) for more details). The polling period defaults to 5 seconds but may be changed in the *MIB Browser* configuration.

The *Stop* button, located in the SNMP settings bar, will stop the current polling process when clicked.

6 Content Section

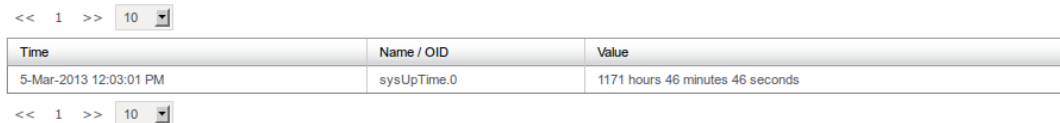
SNMP query results and other information are displayed in the interface main panel as soon as they are received from the Web Server.

6.1 Value Translation

SNMP query results are usually numerical values. However, numerical values may also be used to represent a state, a type or something else. The MIBs provide explanations for the query results as well as a text translation when required. For instance, numeric values whose syntax is known, such as `ifAdminStatus`, will automatically be translated into the corresponding descriptions: `up`, `down` and `testing` instead of 1, 2 and 3.

The same behavior also applies to OIDs, which will be transformed to human-friendly strings. For instance, `.1.3.6.1.2.1.2.2.1.7.1.1` will be translated into `ifAdminStatus.1`.

Some specific numeric types ones may benefit this as well. This is the case for `sysUpTime` values that are transformed into hours, minutes and seconds, instead of being displayed as a number of hundredths of seconds.



Time	Name / OID	Value
5-Mar-2013 12:03:01 PM	sysUpTime.0	1171 hours 46 minutes 46 seconds

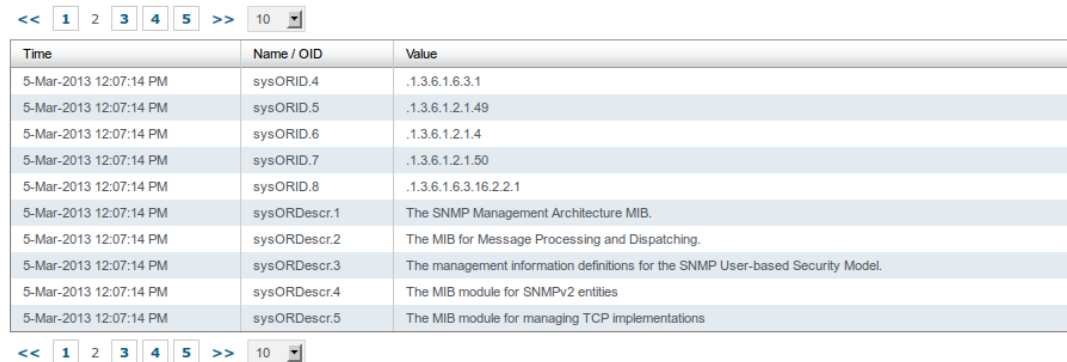
Figure 21: Recovering a set of Values

6.2 Available Views

Some of the SNMP queries may be performed asynchronously, depending on the mode (direct or via an external SNMP Collector) and the amount of data to recover. In such cases, the data will appear progressively, as it arrives from the device, and a little indicator at the top right corner of the page will inform of the current status (when the command will end, it will disappear). Large query results, spanning on more than the specified number of lines, will be displayed on several pages.

Paging can be controlled through the options displayed above the table. The first figures are the available pages, for quick access, with *first* and *last* page shortcuts. The drop-down list enables the

selection of the number of results to display in a page. Default is 10, but it can be extended up to 500. This is especially useful when requesting SNMP walks on agents as shown on figure 22 page 35, or when polling a specific value at a fixed rate in table mode (figure 23 page 36).



Time	Name / OID	Value
5-Mar-2013 12:07:14 PM	sysORID.4	.1.3.6.1.6.3.1
5-Mar-2013 12:07:14 PM	sysORID.5	.1.3.6.1.2.1.49
5-Mar-2013 12:07:14 PM	sysORID.6	.1.3.6.1.2.1.4
5-Mar-2013 12:07:14 PM	sysORID.7	.1.3.6.1.2.1.50
5-Mar-2013 12:07:14 PM	sysORID.8	.1.3.6.1.6.3.16.2.2.1
5-Mar-2013 12:07:14 PM	sysORDescr.1	The SNMP Management Architecture MIB.
5-Mar-2013 12:07:14 PM	sysORDescr.2	The MIB for Message Processing and Dispatching.
5-Mar-2013 12:07:14 PM	sysORDescr.3	The management information definitions for the SNMP User-based Security Model.
5-Mar-2013 12:07:14 PM	sysORDescr.4	The MIB module for SNMPv2 entities
5-Mar-2013 12:07:14 PM	sysORDescr.5	The MIB module for managing TCP implementations

Figure 22: Performing a SNMP Walk

When querying a large table, a standard SNMP walk will not display the columns as one could expect to see them. In fact, there will be one line per value, but the table columns and lines will not be easily noticeable. As a consequence, another mode exists to query tables, and tables only.

The *table view* request (see figure 24 page 37) will display the table using its exact representation in the MIB, one column for each one defined in the corresponding MIB file and one line per real line of the table.

This view is very useful to get a quick look at a specific row in a table, such as an interface, without having to browse through the results of a large SNMP walk query.

The results for SNMP queries can also be exported to a walk file by clicking the *Export* link located in the header toolbar (see section 2).

<< 1 2 3 4 5 >> 10 ▾

Time	Name / OID	Value
6-Mar-2013 11:27:09 AM	ifIndex.1	1
6-Mar-2013 11:27:09 AM	ifIndex.2	2
6-Mar-2013 11:27:09 AM	ifIndex.22	22
6-Mar-2013 11:27:09 AM	ifDescr.1	lo
6-Mar-2013 11:27:09 AM	ifDescr.2	eth0
6-Mar-2013 11:27:09 AM	ifDescr.22	cscotun0
6-Mar-2013 11:27:09 AM	ifType.1	softwareLoopback
6-Mar-2013 11:27:09 AM	ifType.2	ethernetCsmacd
6-Mar-2013 11:27:09 AM	ifType.22	other
6-Mar-2013 11:27:09 AM	ifMtu.1	16436

<< 1 2 3 4 5 >> 10 ▾

Figure 23: Polling values in a Table

<< 1 >> 10 ▾

ifName	ifInMulticastPkts	ifInBroadcastPkts	ifOutMulticastPkts	ifOutBroadcastPkts	ifHCInOctets	ifHCInUcastPkts	ifHCInMulticastPkts	ifHCInBroadca
Gi0/0	287943	1222824	40493	233317	10028998023776	9575988757	287943	1222824
Gi0/1	1131624	20300	3076103	172	1996945257348	7803490461	1131624	20300
Nu0	0	0	0	0				
Lo0	0	0	0	0	0	0	0	0
NV0	0	0	0	0	0	0	0	0
Tu0	0	0	0	0	7804895642990	6161340247	0	0
Tu1	0	0	0	0	0	0	0	0
Tu2	0	0	0	0	0	0	0	0
Tu10	0	0	0	0	174474008818	1083480426	0	0
Lo6	0	0	0	0	0	0	0	0

<< 1 >> 10 ▾

Figure 24: Table View Mode

6.3 Graphing Values

Graphing values dynamically may sometimes be more useful than simply displaying them in a table. This mode of visualization actually allows to observe the behaviour and variation of the variable value over a short time range

Graphs appear in the same section as tables but will automatically adapt to the viewport. The polled OID will appear at the top of the panel, and the time axis will be displayed as HH:MM:SS, in a 24 hours format.

There are two different modes to graph values as well as the possibility to apply a `rate` transformation to the retrieved values.

The first graphing mode allows polling every value under the selected node. This is the default behavior on every node, and will only display the first few ones in the resulting drawing.

On table columns, a simple dialog box will appear (cf figure 25) where a specific row in the table can be specified. If not blank, then this row, if it exists, will be selected. Then its value, and its value only, will be polled and displayed in the graph. Specifying the row should be done when graphing data columns as it is the only way to get meaningful results, without having too much data displayed on the graph.

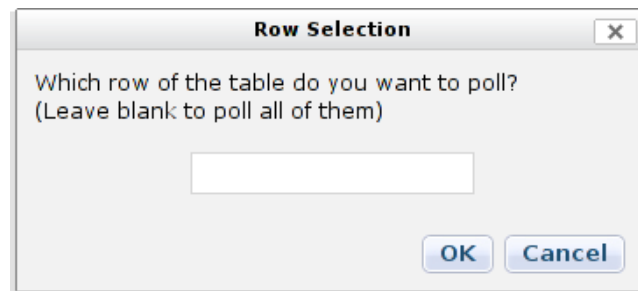


Figure 25: Table Column Polling

For cases where a `rate` is needed, the rate will be computed between the received values, using the real polling time to perform the operation.

For instance, in the `ifTable`, issuing a polling of `ifInOctets` as `rate` on the first row of the table will result in the figure 26.

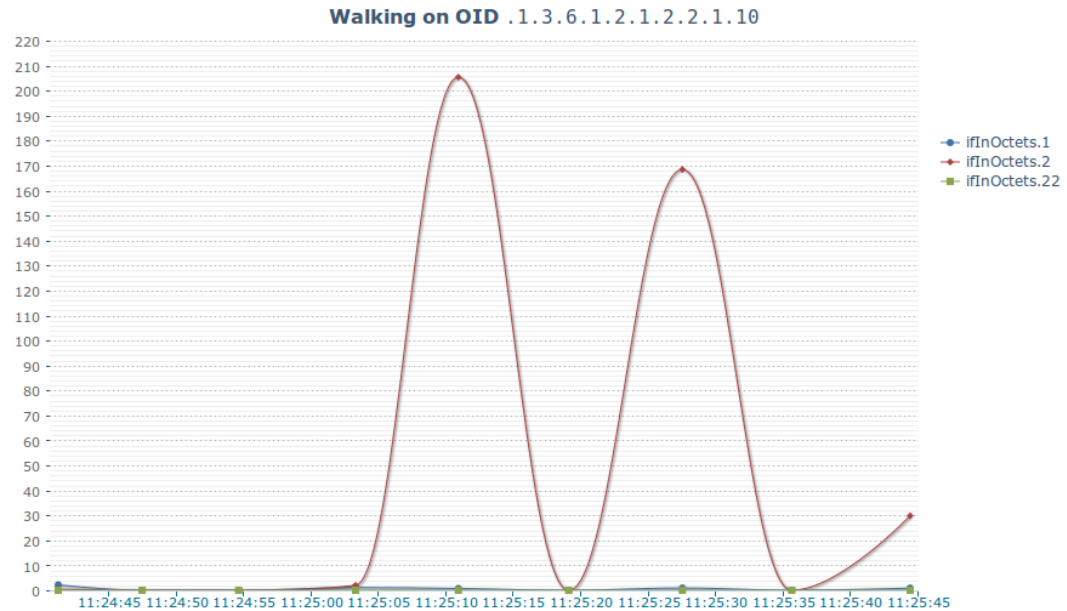


Figure 26: Graph Polling

Note: Rates normally apply to COUNTER value types. Please note that Counter32 values may be polled via SNMP v1, v2c and v3. While Counter64 values are supported only by SNMP v2c and v3. Polling Counter64 values with SNMP v1 will not yield any result for this value.

Polling Configuration

SNMP Collectors may be configured and controlled remotely via the *MIB Browser* application. This feature is available only when the *SNMP Poller* licence is installed. Functionalities include:

- Control over the SNMP Collector state (Running or Stopped);
- Configuration of the collector behaviour;
- Synchronization of the collector configuration files;
- Management of polled SNMP Agents;
- Customization of polled OIDs, tables and their associated properties (snmp-masks management).

This remote control interface will work with SNMP Collectors already installed and running on either local or remote hosts. This interface can be reached through the Polling Configuration link in the *MIB Browser* interface header (see section [2](#) page [8](#)).

Note: The remotely controlled SNMP Collector must already be installed and running on its host machine in order to be accessible through the Polling Configuration interface. The installation of the SNMP

Collector is normally done in default Watch4Net installation, when an idle poller is setup as one of the preconfigured collectors. In this state, the SNMP Collector is ready to accept commands and configuration files but it does not collect any data yet.

WARNING: The SNMP Collector version must be compatible with the Mib Browser in order to be able to send SNMP requests or control commands to it. The Mib Browser application and SNMP Collector must be from the same release to guarantee compatibility.

1 Configurations

Each configuration contains the files a SNMP Collector requires in order to work properly. As a consequence, any configuration is mapped to one SNMP Collector only. The interface allows visualization and modification of almost all properties that can be set in those files.

Note: Some advanced and less frequently used properties must be manually edited in the configuration file. Please refer to APG-SNMP-Collector-Admin-Guide for further information.

The first step in creating a working configuration is to define the communication settings of the associated SNMP poller. By default, the `Polling Config` node will be selected, displaying the list of available configuration.

New configurations can be added here, as shown on figure 27 page 43. Several settings can be defined in this page:

- **Name:** The name of the SNMP Collector;
- **Hostname/IP address:** The name or IP of the host on which the collector is running;
- **Communication Port:** The port number used for communication purposes. The collector listens for commands on this port. (default port: 2008).

The screenshot displays the 'Collectors' configuration page in the MIB Browser. On the left, a sidebar shows the navigation menu with 'Collectors' selected. The main content area has a title 'Collectors' and a descriptive paragraph. Below this is a table with three columns: 'Name', 'Hostname / IP address', and 'Port'. The table lists two collectors: 'Watch4NetSnmpCollector-1' (port 2008) and 'Watch4NetSnmpCollector-2' (port 2009). A 'New Collector' button is located in the top right corner of the table area. At the bottom center, there is a 'Save' button with the text 'save modifications' below it.

Name	Hostname / IP address	Port
Watch4NetSnmpCollector-1	localhost	2008
Watch4NetSnmpCollector-2	localhost	2009

Figure 27: SNMP Poller Configurations

In this example, SNMP Collectors are running on localhost. Watch4NetSnmpCollector-1 listens on port 2008 while Watch4NetSnmpCollector-2 is configured to listen on port 2009.

This page also displays collector availability: Watch4NetSnmpCollector-1 is reachable (and ready to accept commands and queries) while Watch4NetSnmpCollector-2 is unavailable.

A SNMP Collector may be unavailable due to a number of reasons:

- Its collector manager service is not started;
- No route exists between the collector host and the Mib Browser;
- Firewall rules may be blocking TCP traffic on port 2008 between the Mib Browser and the collector;
- It may be extremely overloaded and unable to answer connectivity requests within 5 seconds;

- It may be configured to listen for commands on a specific interface on its host, which is not the one configured in the Mib Browser;
- It may be configured to accept commands only from specific hosts and the Mib Browser's IP is not included in the list;
- ...

2 SNMP Poller

When selecting a specific configuration in the tree in the left window pane, detailed information about the associated SNMP poller will be displayed.

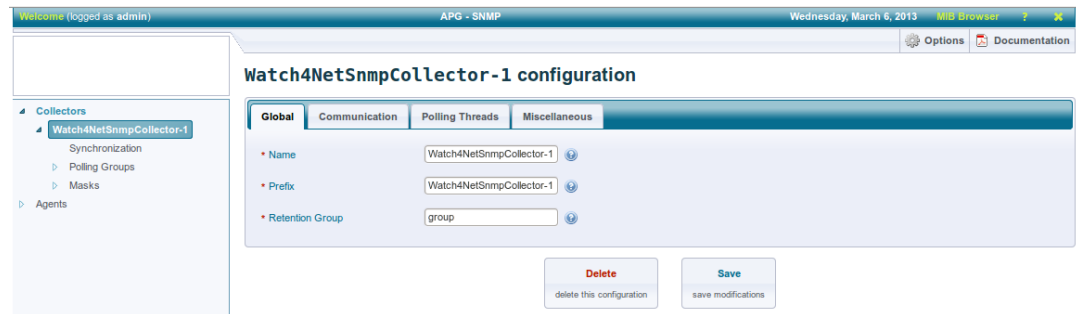


Figure 28: SNMP Poller Configuration

Note: Contextual help is available for most attributes by mousing over the help icon (?). A pop-up will appear, describing its meaning and usage.

Four other settings tabs are also accessible from this page.

2.1 Communication

Basic communication parameters are setup when creating a SNMP Collector, other more specific options may be set here.

Watch4NetSnmpCollector-1 configuration

The screenshot shows the 'Communication' tab of the 'Watch4NetSnmpCollector-1 configuration' window. The window has four tabs: 'Global', 'Communication', 'Polling Threads', and 'Miscellaneous'. Below the tabs, a blue bar contains an information icon and the text: 'These properties will setup any communication with the collector, especially the interfaces for SNMP polling and the IPs allowed to send commands.' The configuration fields are as follows:

- Hostname / IP address:** A text box containing 'localhost' with a help icon.
- Local communication address:** A text box containing '0.0.0.0' with a help icon.
- Port:** A text box containing '2008' with a help icon.
- Interface to use for Polling:** A text box containing '0.0.0.0' with a help icon.
- Allowed IPs for Communication:** A list box that is currently empty, with an 'Add' button (plus icon) and a 'Remove' button (minus icon) to its right.

At the bottom of the window, there are two buttons: 'Delete' (with the text 'delete this configuration' below it) and 'Save' (with the text 'save modifications' below it).

Figure 29: Communication Settings

In this example, the collector uses any/all suitable interface (0.0.0.0) to poll its SNMP agents, listens to commands on any/all interface (0.0.0.0) and allows communication from any *MIB Browsers* server.

Note: Communication restrictions apply as follows:

- Leaving the *Allowed IPs for Communication* field empty lets the collector accept commands from all hosts;
 - Specifying at least one address here orders the collector to reject all commands, except the ones coming from the listed addresses;
 - An unlimited number of IP addresses may be specified.
-

New Allowed IPs for communication are added and removed through the two buttons at the right of each list.

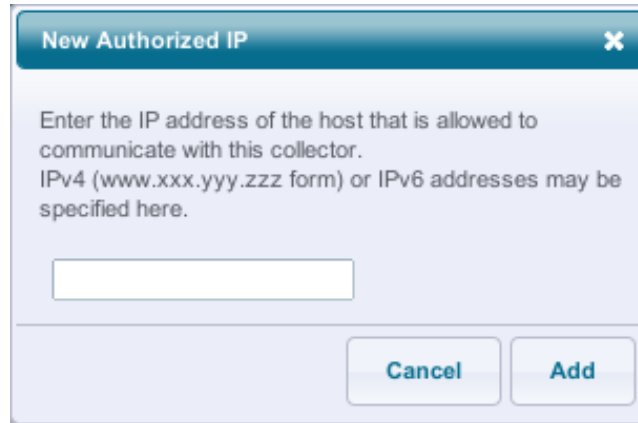
A dialog box titled "New Authorized IP" with a close button (X) in the top right corner. The text inside reads: "Enter the IP address of the host that is allowed to communicate with this collector. IPv4 (www.xxx.yyy.zzz form) or IPv6 addresses may be specified here." Below the text is a single-line text input field. At the bottom right are two buttons: "Cancel" and "Add".

Figure 30: Adding a New IP

The add button displays an input popup (see figure 30), which accepts only valid IPs. IP addresses can be removed by selecting them and clicking the *Removal* button.

2.2 Threads

The number of threads is an advanced set of properties that manages the number of threads that will be allocated for the SNMP Collector. Please refer to APG-SNMP-Collector-Admin-Guide for further information about this topic.

Default values should be suitable for a limited application, such as polling specific OIDS and tables on a limited number of SNMP Agents.

Note: The number of concurrent sessions *MUST* be less than the number of threads dedicated to both the polling of OIDs and the polling of tables. The default values for threads and sessions are suitable for

Watch4NetSnmppCollector-1 configuration

Global

Communication

Polling Threads

Miscellaneous

ⓘ

One must note the importance of setting the following attributes with adequate values as they will directly impact the collector's performance.

• Max Concurrent Sessions

98

?

• Max OID polling threads

100

?

• Max Table polling threads

100

?

Delete

delete this configuration

Save

save modifications

Figure 31: Threads and Sessions Settings

relatively light data collection processes.

2.3 Miscellaneous

This tab comprises other settings which modify the SNMP Collector behavior.

Watch4NetSnmCollector-1 configuration

The screenshot shows the 'Miscellaneous' tab of the 'Watch4NetSnmCollector-1 configuration' window. The window has four tabs: 'Global', 'Communication', 'Polling Threads', and 'Miscellaneous'. The 'Miscellaneous' tab is active. It contains four settings, each with a checkbox and a help icon (a blue circle with a question mark):

- 'Always include properties' is checked.
- 'Handle SNMP Counter overflow' is checked.
- 'Include OID as property' is unchecked.
- 'Use collector prefix in variables' is unchecked.

At the bottom of the window, there are two buttons: 'Delete' (with the text 'delete this configuration' below it) and 'Save' (with the text 'save modifications' below it).

Figure 32: Miscellaneous Settings

3 Polling Groups

Polling groups are logical units. They bind SNMP Agents and polled OIDs to a polling period. [SNMP Agents](#), [IP Pattern Grouping](#) and/or [Explicit Grouping](#) need to be defined along with [SNMP Masks](#) so that they can be added to the new polling group. See figure 33, page 50, for a Polling Group logical representation.

In the global example we are following, none of the previously mentioned parts have been defined. So we will switch to the [Agents configuration](#) section first to define some SNMP agents. Then, we will move on to the [SNMP Masks](#) section to specify OIDs and tables to be polled.

Once SNMP Agents and SnmpMask have been created, a new polling group can be created in the *polling group overview panel*. There, some basic properties must be set, such as the polling group name, its period in seconds, the time of the day at which polled properties will be refreshed and the retention group to use.

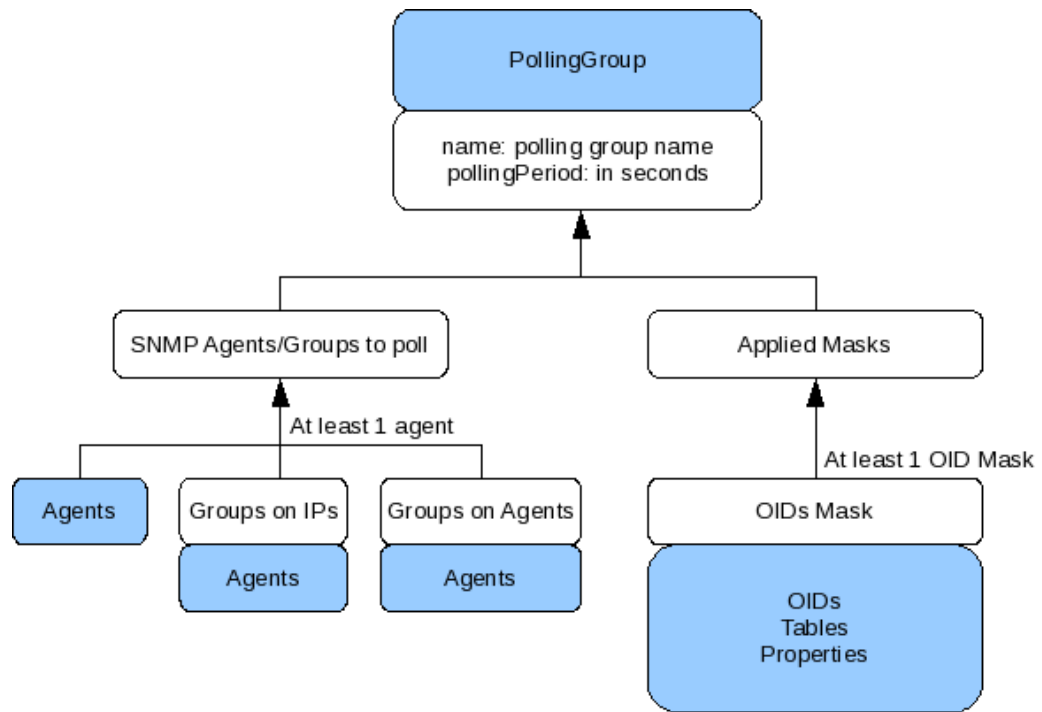


Figure 33: Polling Group structure

Groups can also be merged with an external `slave-snmppoller` file and thus quickly imported from other configurations. The file input field on that particular screen will accept any XML file that matches the requested format.

When the *overwrite* option is selected, any already existing polling group whose name is present in the import file will be silently overwritten. If not checked, then the merge process will interrupt itself on such collisions.

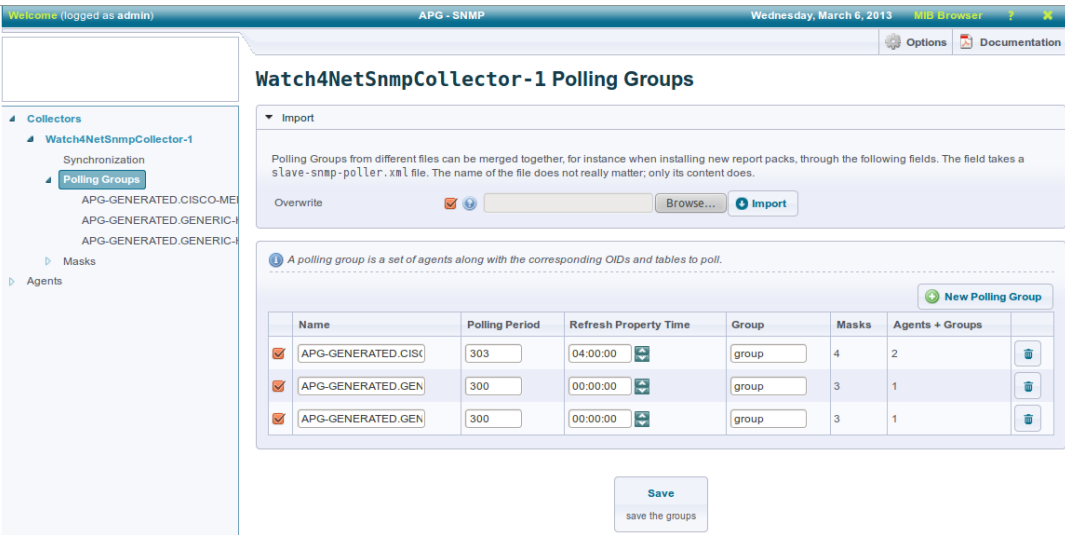


Figure 34: Defined Polling Groups Overview

Note: In order to enhance backend performance, properties are refreshed only once a day. However, they are polled on every cycle as some filters require those properties.

Note: The retention group is a *backend*-related notion. For more details about its meaning, please refer to the Backend Administration Guide.

3.1 Settings

This panel contains the main settings of a polling group. The retention group name and the *Enable this group* check box (leaving this box unchecked tells the collector to ignore this polling group and not collect any data for it) refer to the collection and the retention processes. *Uniform Timestamps*, *Start at Defined Period* and *emphStart* time offset deal with the timestamps associated with the polled OIDs and polling group delay before starting their first polling. They may only be useful when specific starting times are required or when a better polling load distribution is desirable.

Configuration

* Name: APG-GENERATED.CISCO-M

* Polling Period: 303

Enable this Group: ☒

▼ Advanced

* Refresh Property Time: 04:00:00

Retention Group Name: group

Uniform Timestamps: ☐

Start at Defined Period: ☐

Start time offset, in seconds.: 0

Figure 35: Polling Group Settings

3.2 SNMP Agents

SNMP agents can be selected and/or removed through the three following lists. Each one selects a specific group as defined in the [Agents configuration](#) section.

Individual SNMP agents, complex IP-based groups and explicit groups may be associated to this polling group. Then, for any of the agents contained in those groups, all specified OIDs applied through the section below will be polled at the specified period.

Agents & Groups

The available agents and/or groups are those defined in the corresponding SNMP Agent's section available at the bottom of the tree in the left panel. They need to be defined there under the same name as the one of this specific collector, before being selected in that screen.

Explicit Groups

CISCO-DEVICES (1 agents)

IP Groups

Agents

Figure 36: Polling Group Agents

Agent groups and/or Agents assigned to the Polling Group will appear in their corresponding lists. Clicking the *Edit* button displays a popup which allows the selection of items to add or remove from the list. Those lists allow multiple selection in order to remove or add more than one item at a time.

Note: Agents selected through their IP:port setting here will not be selected anymore if those values change afterwards. This is where Agent groups' main advantage is. If a SNMP Agent is added to a Agent group, then the SNMP Agent will be polled by all Polling Groups which use the Agent group. And the reverse is also true: once an Agent is removed from an Agent group, none of the Polling Group which

use the Agent group will poll the removed Agent anymore.

3.3 SNMP Masks

The selection of SNMP masks works like the Agent selection. The only difference here is that there is only one kind of mask, so there is only one list.

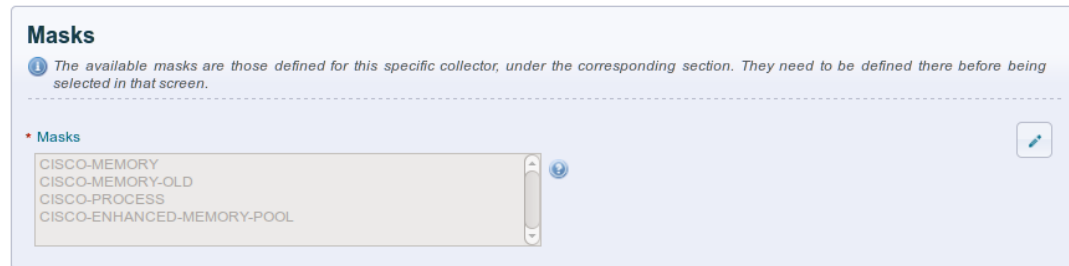


Figure 37: Polling Group Masks

Any mask added here will then be applied on all selected agents and/or Agent groups at each polling cycle. If OIDs are added to the SNMP mask, or any change (except for the name) is applied on the SNMP mask after this one has been selected inside a polling group, then the changes will be taken into account when the SNMP poller reads its configuration and starts polling.

Note: If a mask is used in a polling group and the mask name changes, it will be changed in the polling group as well.

4 SNMP Masks

Snmp masks (mask) contain oids (scalar) and tables polled for a specific purpose. Typically, individual mask contain polling definition for a given technology, be it proprietary or not. For instance, a mask may contain polling definitions for Host information (Cpu, Memory, Disk usage percentage, etc). The mask name should reflect this, as is the case with the ones provided by default.

Masks management (Create/Edit/Delete) may be done via the mask management page (figure 38). This list is displayed by clicking the “Collector -> <Collector Name> -> Masks” link.

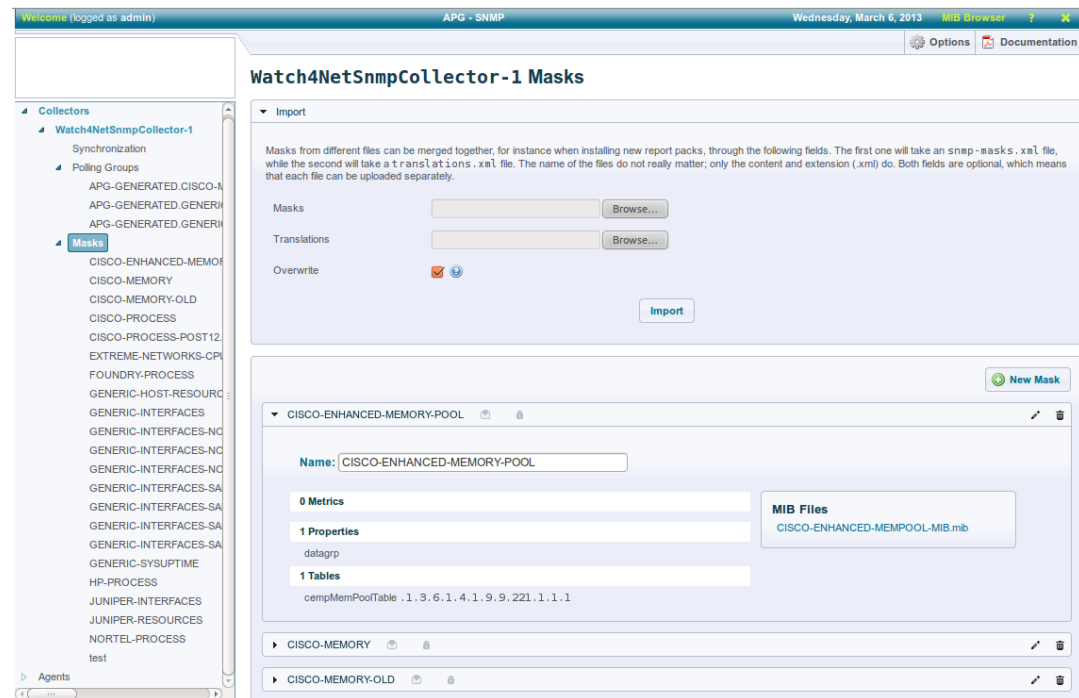


Figure 38: Masks management

This page has 2 distinct sections:

- The *Import* section (collapsed by default, displayed in in figure [38](#));
- The *Masks* list.

About raw values and properties Before continuing with this section description, here is a brief reminder about Watch4Net's data model, using SNMP terminology:

- The variable is the metric class (polled scalar OID or a table "cell", identified by its column suffix and row index);
- The raw value is the metric instance (with variable name, timestamp, group, polled value and other properties).

The SNMP Collector provides a set of universal properties for every raw value it is producing:

- `source`: This is the Prefix value, defined in the Collector configuration.
- `pollgrp`: This is the name of the Polling Group which generated the value.
- `devtype`: The device's devtype property. This is set up in the Agent configuration.
- `device`: The device name.
- `devdesc`: The device description. This is the sysDescr value polled on the device.
- `model`: The model OID. This is the device's sysObjectID value. This OID may be translated to a human readable string via a filter installed in the backend collecting.xml file.
- `ip`: The device's ip address.
- `contact`: The device's sysContact value, if available.
- `location`: The device's sysLocation value, if available.

Properties really play a crucial role in Watch4net: The report tree is built on their presence and possible values. As a consequence, the more properties the more flexibility. However, some limitations have to be kept in mind when adding properties to the polling.

- Property names length are limited to 8 characters, while their values are limited to 128 (this can be changed but it requires altering the database schema, which is out of this documentation scope);
- Make sure that the `Optimized Flat Property Table` has been enabled in your *APG* configuration as it will greatly improve performance when a lot of properties are polled;
- Property values are not the ones appearing in the reports as graphs: they are only used to identify variables displayed in them;

- Property history is not tracked through time in Watch4Net: reports only show their last value recorded in the backend.

Some further properties can be added to this universal set, depending upon the data that is actually polled. These can be added as:

- Mask property, where their values will be applied to all metrics generated by the mask (see section 4.2);
- Single oid metric properties (part and parttype only, see section 4.2);
- Table metric properties (values usually come from the same row as the polled variable, or in a joined table);

Polled oids or table columns must have values that can be parsed as a floating point value. Supported types include:

- Integer, Unsigned, Counter, Gauge;
- Integer enum values, translated back to a floating point value (up:100.0;down:0.0);
- Floating point values formatted as strings;
- String values that can be translated to a floating point value.

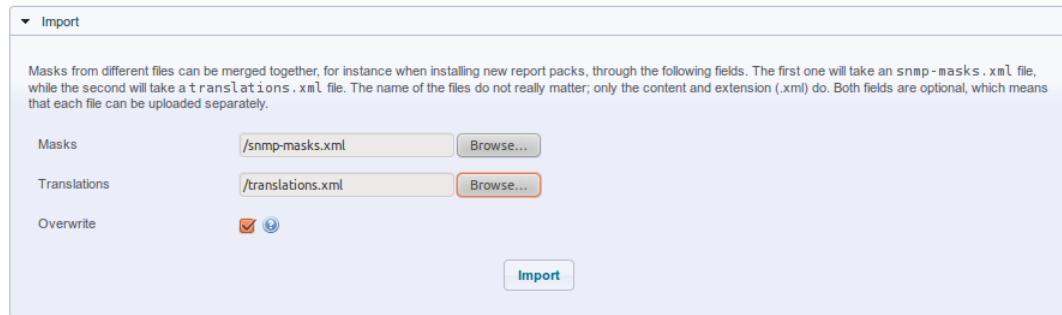
On a last note, polled values can be transformed through calculations:

- Counter or —: No calculation is performed on the polled value. It is sent as is to the backend.
- Delta: This is the difference the current and last polled value: $\text{Value}(t2) - \text{Value}(t1)$. Counter overflow is handled on SNMP Counter type values. High values may appear when counters are reset.
- Positive Delta: This is the difference the current and last polled value: $\text{Value}(t2) - \text{Value}(t1)$. Counter overflows are not handled: when they occur, $\text{Value}(t1)$ is considered to be 0. While this is not mathematically accurate, it is better than getting extremely large erroneous values on counter resets.
- Rate: This is the calculated rate $(\text{Value}(t2) - \text{Value}(t1)) / (t2 - t1)$. Counter overflow is handled on SNMP Counter type values. High values may appear when counters are reset.
- Positive Rate: This is the calculated rate $(\text{Value}(t2) - \text{Value}(t1)) / (t2 - t1)$. Counter overflows are not handled: when they occur, $\text{Value}(t1)$ is considered to be 0. While this is not mathematically accurate, it is better than getting extremely large erroneous values on counter resets.

4.1 SNMP Masks Import

Typically, this functionality is used to import SNMP masks XML files provided by EMC. These masks are usually required to perform data collection for specific report packs. Value translations XML files (if provided) may also be imported through this functionality.

Note: Masks files are typically named *snmp-masks.xml* while translations filenames is usually *translations.xml*.



▼ Import

Masks from different files can be merged together, for instance when installing new report packs, through the following fields. The first one will take an `snmp-masks.xml` file, while the second will take a `translations.xml` file. The name of the files do not really matter; only the content and extension (.xml) do. Both fields are optional, which means that each file can be uploaded separately.

Masks

Translations

Overwrite ☒ 

Figure 39: Masks import

4.2 Masks list

The masks list (figure 40) displays the masks information summary (collapsed by default) and offers management functionalities such as create, delete and edit.

The following information is provided by this list:

- The mask name;
- Envelope icon, which indicates the mask encryption status. Encrypted masks cannot be edited and are usable only on SNMP Collectors that have the required license for the mask;
- Padlock icon: locked means that the mask cannot be edited;
- The mask details can be viewed or edited by clicking the edit (pencil) button;

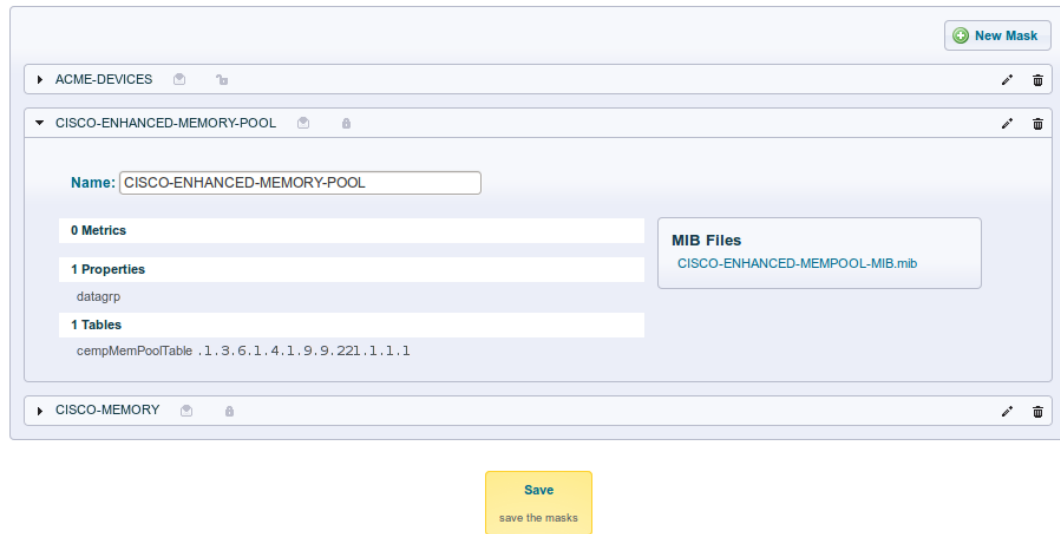


Figure 40: Masks list

- The mask can be deleted by clicking the delete (trash can) button;
- The mask summary information (when expanded).

The list also allows creating a new mask by clicking the “New Mask” button. Modifications to the list must be saved by clicking the “Save” button, which is located at the bottom of the list. Not saving the list after modifying its content and navigating to another page will simply cancel the changes.

Mask edit page The mask edit page (figure 41) allows basic mask editing as well as viewing the polled oids (scalar and table) configuration.

Note: Locked masks configuration details can be viewed in this page, but not edited.

A few actions may be performed here:

Name

EXTREME-NETWORKS-CPU-MEMORY

Version

20130131

MIB Files

EXTREME-BASE-MIB
EXTREME-SOFTWARE-MONITOR-MIB
EXTREME-SYSTEM-MIB

Cleanup List

Delete Mask

delete this mask

Save Mask

save this mask

Metrics

These metrics will be directly polled on the agent.

New Metric

Name	OID	Transformation	Unit	Part	Part type	Type	Actions
TestMetric	.1.2.3.4.5.0	RATEP	test/s	Test	TestPart	Unknown	

Properties

These properties (OID or static value) will be attached to every metrics collected by the mask.

New Property

Name	Static Value / OID	Actions
datagrp	EXTREME-NETWORKS-CPU-MEMORY	

Tables

extremeCpuMonitorSystemTable

.1.3.6.1.4.1.1916.1.32.1.4

extremeMemoryMonitorSystemTable

.1.3.6.1.4.1.1916.1.32.2.2

Figure 41: Masks edit page

- Mask name editing;
- Version editing (version number is optional but useful for change management purposes);

- Required MIB file list edit: mask required MIB files may be added or removed through this list. The cleanup button regenerates the required MIB list by scanning the polled oids (scalars and tables) and identifying the MIB files where those oids are defined. No deep type dependency is performed however.
- Delete the mask (click the Delete button, then confirm);
- Save the mask (click the Save button);
- Add, edit or remove polled metrics (click the “New Metric” button, click the “pencil” button on the metric row, click the “trash can” button on the metric row);
- Add, edit or remove properties (click the “New Property” button, click the “pencil” button on the metric row, click the “trash can” button on the metric row);
- View polled table metrics details (figure 42 and 43 collapsed by default, expand to display them);
- Delete a polled table.

Note: Single oid (scalar) based metrics and properties configuration may be edited from this page. Table based metrics polling configuration must be performed through the Mib Browser.

MIB files list The list of the MIB files which apply to this mask should contain at least the MIBs in which scalar OIDs and tables are declared. The information contained in these MIB files is used to translate the data polled from the SNMP Agent.

Note: If a dummy MIB is provided, the translation might not work properly because the collector will lack the lookup tables required to perform data translation. The polled value will then be pushed as is. This is not usually desirable when polling Integer values which models status information.

One such example lies in the ifTable, is the ifAdminStatus column, where polled values are either 1, 2 or 3. These values are status information which are: up, down, testing. If no MIB translation is done, values pushed by the collector will be 1, 2 or 3. These values are of limited use in a network management context. If the MIB translation is available (the RFC1213-MIB file has been loaded), polled values will be translated to the textual status: up, down, testing. Then, via the user provided translations, these status may be converted to 100, 0 and NaN (for not a number: testing is neither up

▼ stdIfTable2

Summary

Table Index

.1.100

Index Entry Mask

oid.____.100

Part

.1.2

Part Type

Interface

Next Reference Table Column

.1.100

Index OID

Keep OID order in output

☐

Cache reference tables

☐

Table row specification file name

Match on 'n' first suffixes only

Metrics

Name	Column	Type	Unit	Property Name	Property Column	Use in Variable
ifInDiscards	.1.13	RATEP	Pkts/s	ifindex	.1.100	
ifInErrors	.1.14	RATEP	Pkts/s	ifindex	.1.100	
ifInUnknownProtos	.1.15	RATEP	Pkts/s	ifindex	.1.100	
ifOutDiscards	.1.19	RATEP	Pkts/s	ifindex	.1.100	
ifOutErrors	.1.20	RATEP	Pkts/s	ifindex	.1.100	

Properties

Name	Column	Reference table	Index column	Joint column positions	Index Entry Mask	Use in Variable	Nested table joint col.	Hex Value
part	.1.2	ifTable.ifDescr	.1.100	1	oid.____.100	☒		☐
iftype	.1.3	ifTable.ifType	.1.100	1	oid.____.100	☐		☐
maxspeed	.1.5	ifTable.ifSpeed	.1.100	1	oid.____.100	☐		☐
ifacemac	.1.6	ifTable.ifPhysAddress	.1.100	1	oid.____.100	☐		☒

Figure 42: Table details

or down. NaN signals the collector not to push this value when it is encountered). Such values are

Others Tables Properties								
Name	OID	Reference table	Index column	Joint column positions	Index Entry Mask	Use in Variable	Nested table joint col.	Hex Value
ifaceip	.1.3.6.1.2.1.4.20.1.1	ipAddrTable.ipAdEntAddr	.1.2	1		☐		☐
ifname	.1.3.6.1.2.1.31.1.1.1.1	ifXTable.ifName	.1.100	1	oid._._.100	☐		☐
ifalias	.1.3.6.1.2.1.31.1.1.1.18	ifXTable.ifAlias	.1.100	1	oid._._.100	☐		☐
mxhspeed	.1.3.6.1.2.1.31.1.1.1.15	ifXTable.ifHighSpeed	.1.100	1	oid._._.100	☐		☐

Joint specifications (between the table and filters on external tables)

Joint Name	Applies to Table Columns
No joint specification defined.	

Applied Filters

Name	OID	Value	Index Entry Mask	On Table Column Name	On Table Name	On Table OID	...
	.1.7	up			stdIfTable2	.1.3.6.1.2.1.2.2	

Figure 43: Table details

very useful when graphing interface availability status.

The MIB file list can be manually updated, using a quick switch list (see dialog [44](#) page [64](#)), but it also can be automatically managed by the application. In fact, each time a new OID (property, single or table) is added through the tree interface, the MIB file which defines it will be added as well.

The required MIB file will not be removed upon metric, property or table deletion. This is because it involves a global cleanup, which will remove MIB files that were added manually. However the *Cleanup List* button will trigger this cleanup, removing every non-essential MIB file from the list.

Metric edit dialog Single oid (scalar) based metric polling configuration dialog is invoked by clicking the “New Metric” button or the “Pencil” (edit) button on the metric row. A dialog similar to figure [45](#) is displayed upon invocation. Fill in the required information and click save to persist the changes or cancel to void them.

Single OIDs are device-based OIDs that do not come from a table (scalar). They are the time-

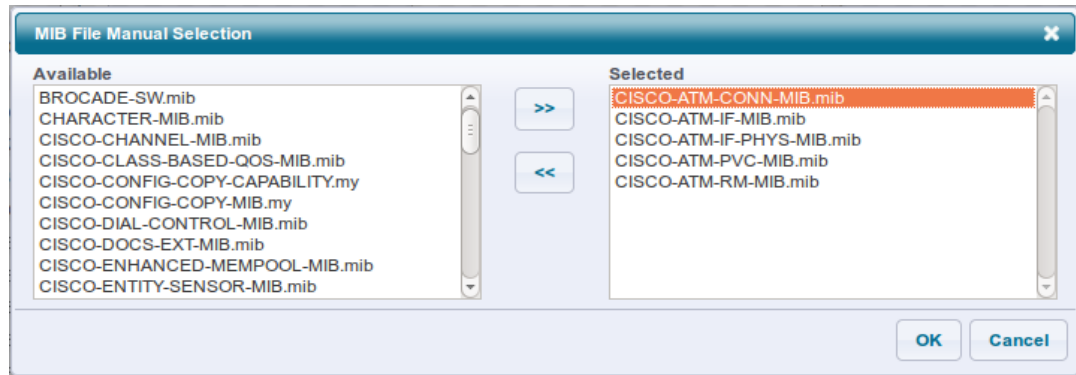


Figure 44: Mask MIB Files Management

based values that will be available in Watch4Net's frontend for any computation or display.

Note: Even if the interface allows it, string values cannot be polled as values: *APG* needs numeric values for its time-based series. The only exception to this rule are string values that can be parsed as floating point numbers, or translated to a floating point value.

Single OIDs (scalar) may also be configured through the MIB tree by selecting them, right-clicking, then choose "Configure Polling" and either create the metric or edit an existing one.

Note: When editing an existing polled single oid, a new variable will be created if its transformation type or unit is changed, as it alters the signification of the metric.

The two last fields are optional and may be omitted if not required. This is often the case as single OIDs are mostly device-related values that do not need *part* or *parttype* information.



The image shows a 'Metric' dialog box with a title bar containing a close button. The dialog contains several input fields, each with a help icon (question mark in a circle) to its right. The fields are: 'Name' with the value 'AvailableMemory', 'OID' with the value '.1.3.6.1.4.1.9.9.99999', 'Type' (empty), 'Transformation' (empty with up/down arrows), 'Unit' with the value 'MBytes', 'Property: part' with the value 'RAM', and 'Property: parttype' with the value 'Memory'. At the bottom right are 'OK' and 'Cancel' buttons.

Field	Value
Name	AvailableMemory
OID	.1.3.6.1.4.1.9.9.99999
Type	
Transformation	
Unit	MBytes
Property: part	RAM
Property: parttype	Memory

Figure 45: Metric edit dialog

Property edit dialog Single oid (scalar) based property, as well as static ones (with a hardcoded value) are edited by invoking the property edit dialog. This is done by clicking the “New Property” button or the “Pencil” (edit) button on the property row. Alterating between oid (scalar) based property

value and a static (hardcoded) one is done by selecting the type in the dialog. See figure 46.



Figure 46: Property edit dialog

Table polling configuration Table polling configurations can be viewed from the mask edit page. Tables may be removed from the mask as well. The Mib Browser is however required in order to create table polling configuration. Please refer to section 4 for more information about this.

4

Table polling configuration

Polling table columns is more complex than polling scalar OIDs. Tables are often linked in unpredictable ways, their index structure may not be static or are simply identifiers unrelated to any relevant information.

Non-accessible columns are sometimes used as index, preventing their direct use. Other times, the table does not even have index columns defined in the MIB. So, while there are general guidelines to establish an accurate polling model, configurations are usually created on a case by case basis.

The following is a list of guidelines generally applicable when building a table polling configuration:

- Typically, tables contain only one *part type*.
- Tables typically have several parts (rows). These rows are identified by values found in 1 or more of its columns, or in columns found in tables which extend it.
- Normally, the raw value variable uses the value from the same column as the one used to set the *part* value. But this not always applicable: the variable can also be formed by combining several part values.
- Table row index values should be stable over time to avoid the creation of an infinite number of metrics in the backend. Index using information such as timestamps, session numbers and combination of IP addresses and ports should be *AVOIDED* for this very reason.

Example:

Let's suppose that we want to poll ifTable's statistics. Using the guidelines stated above, we would:

- Set the parttype property to "Interface";
- Set the part property to use column ifDescr (we assume the ifDescr is unique among interfaces);
- Set the part property as the variable. The ifDescr column effectively becomes the index in this case (ifIndex is the table index, not the variable);
- We assume that ifDescr stays stable over time: if the interface description changes, it means that new metrics are generated as the interface probably changed its purpose.

Now, one might say that ifDescr is the same for all interfaces on a particular device. In this case, we would use ifXTable.ifName instead of ifTable.ifDescr for the part and variable. In the end, the idea is the same: use an index that is unique among rows and is stable over time.

Table polling configuration must be performed through the Mib Browser.

Two configuration modes are now provided: a basic and advanced one. The basic mode can be used to create and edit polling table configuration; the advanced mode can only be used to edit an existing configuration.

1 Basic edit mode

Accessing the table configuration mode is done by right-clicking on a MIB tree table node, as shown in figure 47. Then select either "Create new..." to create a new mask or "Edit mask 'name' (collector 'collector name') to edit an existing one (on the named collector).

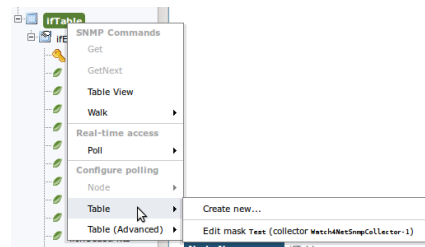
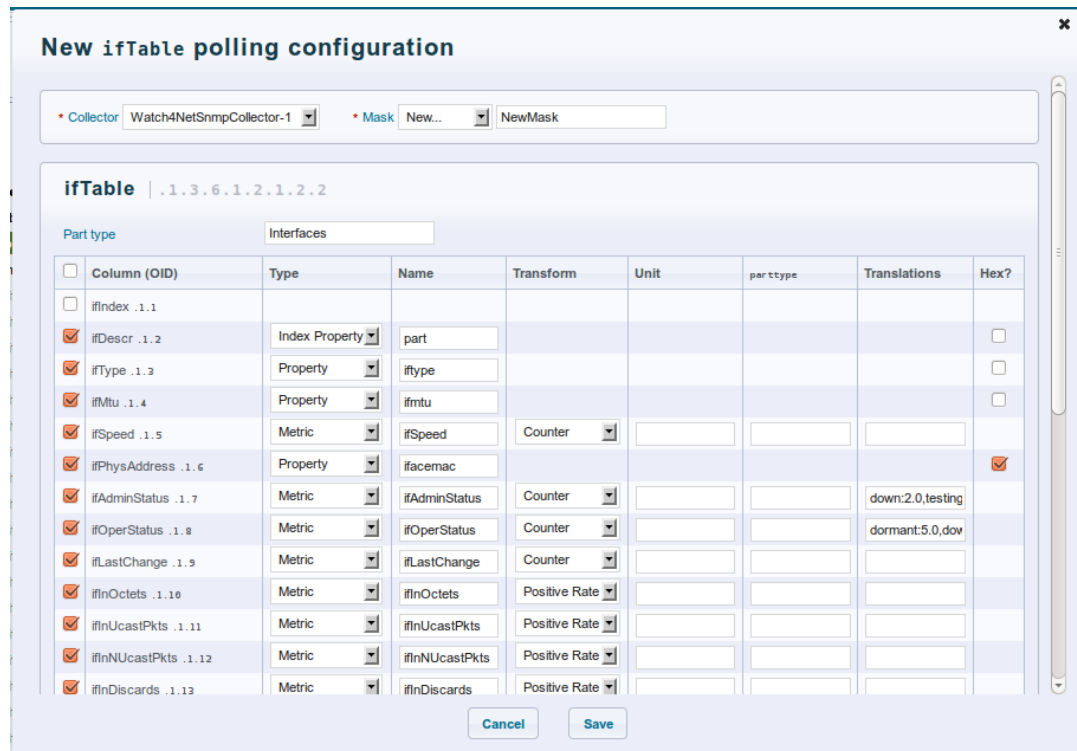


Figure 47: Table edit context menu

The table configuration will appear in the contextual dialog as shown in figure 48.



New ifTable polling configuration

* Collector: Watch4NetSnmpCollector-1 * Mask: New... NewMask

ifTable | .1.3.6.1.2.1.2.2

Part type: Interfaces

☐ Column (OID)	Type	Name	Transform	Unit	part type	Translations	Hex?
☐ ifIndex .1.1							
☒ ifDescr .1.2	Index Property	part					☐
☒ ifType .1.3	Property	iftype					☐
☒ ifMtu .1.4	Property	ifmtu					☐
☒ ifSpeed .1.5	Metric	ifSpeed	Counter				
☒ ifPhysAddress .1.6	Property	ifacemac					☒
☒ ifAdminStatus .1.7	Metric	ifAdminStatus	Counter			down:2.0,testing	
☒ ifOperStatus .1.8	Metric	ifOperStatus	Counter			dormant:5.0,dow	
☒ ifLastChange .1.9	Metric	ifLastChange	Counter				
☒ ifInOctets .1.10	Metric	ifInOctets	Positive Rate				
☒ ifInUcastPkts .1.11	Metric	ifInUcastPkts	Positive Rate				
☒ ifInNUcastPkts .1.12	Metric	ifInNUcastPkts	Positive Rate				
☒ ifInDiscards .1.13	Metric	ifInDiscards	Positive Rate				

Cancel Save

Figure 48: Basic table edit dialog

The collector and mask names must be specified when editing a new table configuration. The mask name may also be new: in this case, select “New...” and input the mask name in the field.

Setting up the Part Type for the table is also recommended, although it is possible to set a different value for every Metric column.

By default, all columns are selected (all columns are checked) and will be polled either as “Metric”, “Property”, “Index Property” or “Index”. Each type has its own purpose and, based on the SNMP

provided type, the Mib Browser will try its best to guess which type should be used in the configuration. Type definitions are:

- *Metric* type columns are polled as values (in the output raw value). There must be at least one Metric per polled table.
- *Property* type columns are polled as properties (in the output raw value). Having properties is optional, though highly recommended to ensure data is more usable.
- *Index Property* type columns as polled as properties (in the output raw value) and are also use (in combination) to build the raw value variable. Having at least one *Index Property* table is highly recommended, otherwise the variable will be generated by combining the actual SNMP table index column values, which are not always meaningful or stable over time.
- *Index* type is special: it is only available for columns that are declared in the table but cannot be polled directly (not-accessible).

WARNING: Polling not-accessible columns as Index or Index Property type may yield unpredictable variable if combined with Index Property type columns. Please review the configuration in advanced mode if this is required as some combinations may not be generated correctly by the basic edit mode. Also, polling non-declared columns as Index or Index Property is not possible in this mode. Please use the advanced mode to edit this type of configuration.

Possible column configuration options include:

- *Name* is the value of the name property for “Metric” columns (128 characters long, mixed case). It is the name of the property for “Property” and “Index Property” columns (8 characters long, lowercase only).
- *Transform* is the transformation type to apply on the polled value. See `Referencessec:snmp-masks` for transform types.

Then, columns of type “Property” and “IndexProperty” can be added from other columns. Simply scroll down the table (see figure 49) and input the table name or OID in the field and click the “Add ” button.

Once again, columns may be set to “Property” or “Index Property” (see figure 50. Metrics cannot be polled from a property table: a new table must be configured for this.

Note: Tables that are already configured in other masks polling are prepopulated with “Index Property” and “Property” columns as used elsewhere for convenience and performance enhancement.

The screenshot shows a configuration window for table polling. It contains a table with three rows of properties: #OutErrors (Metric, Positive Rate), #OutQLen (Metric, Counter), and #Specific (Property, specific). Below the table is a section for 'Advanced properties: editable in advanced mode only' and a section for 'Add properties from another table' with a text input field containing '#XTable' and an 'Add' button. At the bottom are 'Cancel' and 'Save' buttons.

☒	#OutErrors .1.20	Metric	#OutErrors	Positive Rate				
☒	#OutQLen .1.21	Metric	#OutQLen	Counter				
☒	#Specific .1.22	Property	specific					☐

Advanced properties: editable in advanced mode only

Add properties from another table

Table name or OID: #XTable Add

Cancel Save

Figure 49: Adding a table of properties

2 Advanced edit mode

The advanced edit mode can be used to edit table polling configuration to a finer degree. But be advised that table edited in this mode may not be editable in the basic mode afterwards as it does not support all configuration features due to its simplified layout and configuration logic.

Do not edit a table configuration in basic mode if the following fields are changed in advanced mode:

- table index;
- index entry mask;
- inner joint name;
- inner joint columns;

Edit mask NewMask (collector Watch4NetSnmCollector-1)

☒

#OutNUcastPkts .1.18

Metric

☒

#OutDiscards .1.19

Metric

☒

#OutErrors .1.20

Metric

☒

#OutQLen .1.21

Metric

☒

#Specific .1.22

Property

specific

#OutNUcastPkt

Positive Rate

#OutDiscards

Positive Rate

#OutErrors

Positive Rate

#OutQLen

Counter

Advanced properties: editable in advanced mode only

ifXTable | .1.3.6.1.2.1.31.1.1

☐	Column (OID)	Type	Name	Hex?
☒	#Name .1.1	Property	ifname	☐
☐	#InMulticastPkts .1.2			
☐	#InBroadcastPkts .1.3			
☐	#OutMulticastPkts .1.4			
☐	#OutBroadcastPkts .1.5			
☐	#HCInOctets .1.6			
☐	#HCInUcastPkts .1.7			
☐	#HCInMulticastPkts .1.8			
☐	#HCInBroadcastPkts .1.9			

Cancel

Save

Figure 50: Editing a table of properties

Main Attributes

The Main Table Tab (figure 51) displays the basic table attributes such as its Id, name, base table OID, index column, an optional index structure definition (Index entry Mask), the column where the values for the part property will be polled (Table Column Name OID) and part type value.

On the right side of the dialog, a panel displays some information about the table entry and its description, especially the indexes of the table.

Note: Technical note about the table index and indexEntryMask field:

In many cases, the index column exists in the table and is accessible. So it can be used as is in the Table Index field with the format as shown: .1.X (X being the column number).

However, in order to improve polling performance, it is not desirable to poll this index column as its values are accessible as part of the table row's index. Therefore, it is much better to declare the column in the indexEntryMask structure, as displayed in this example, where the indexEntryMask is oid.....100. Using this structure ensure that the SNMP Collector will not try to poll this column.

In the case of table index columns that are declared but not accessible, the same principle can be used except that the actual column number may be declared in the indexEntryMask. For instance, if polling table that has its index in column .1.1 and this column is not accessible, the table index field may be set to .1.1 and the indexEntryMask=oid.....1

Table Columns to be Polled

Main TableColumnsFiltersReferences

?

ID

ifTable-136258389705

*

?

Name

ifTable

*

?

OID

.1.3.6.1.2.1.2.2

?

Table Index

.1.100

*

Index Entry mask

oid_...100

?

Part Column

?

Parttype (Static)

Interface

?

Multiple Rows

Yes

No

?

SNMP Repeaters

?

Cache reference tables

Yes

No

?

Keep OID order in output

Yes

No

?

Row filter file

?

Match on 'n' first suffixes only

?

Inner Joint name

?

Inner Joint columns

Table Entry

Entry Name

ifEntry

Entry OID

.1.3.6.1.2.1.2.2.1

Table Indexes

ifIndex

Description

"An interface entry containing objects at the subnetwork layer and below for a particular interface."

OK

Cancel

Figure 51: Main Table Tab

Columns to Poll

This section contains the columns of the table (figure 52). Polled columns are highlighted in green. In order to trigger the polling of a specific column, there are two easy steps: selecting the column and checking the corresponding box in the *Polling* tab. On each column, transformations and other properties can be applied at will. All of them are explained in APG-SNMP-Collector-Admin-Guide.

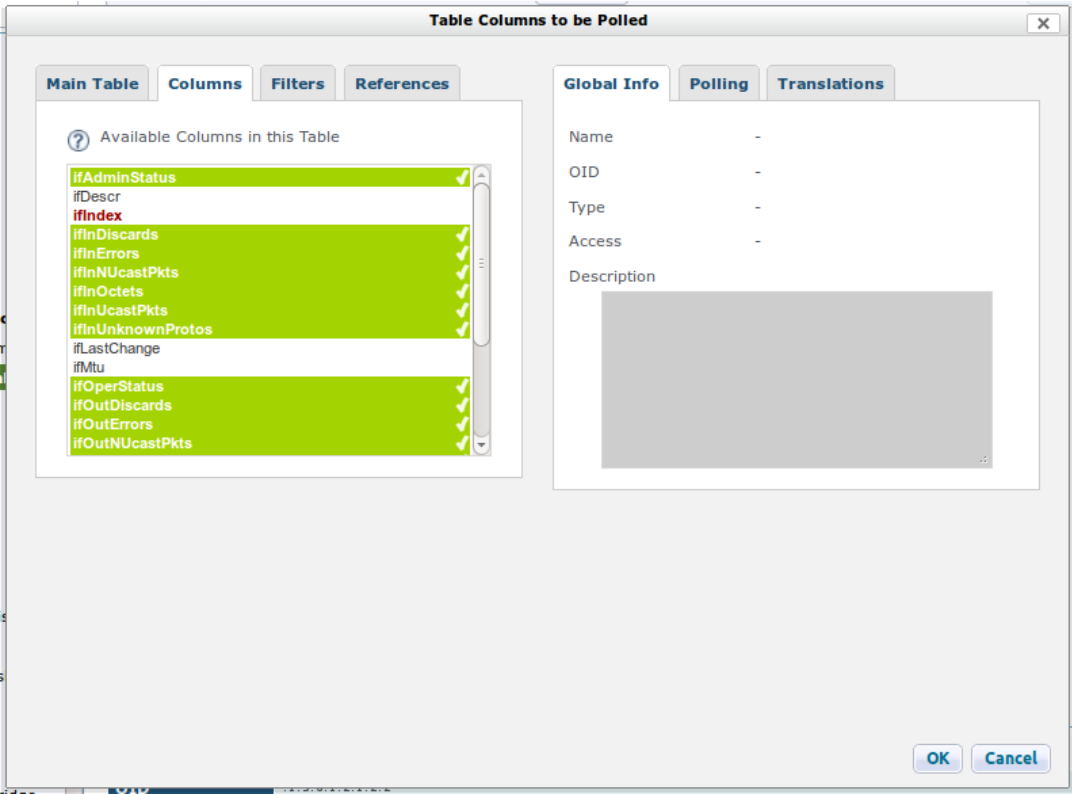


Figure 52: Table Columns Tab

If required, specific translations can be applied on a column value. For instance, if the column is an Integer that represents different states, those states are likely to have String translations in the MIB file that defines them. The *Translations* tab is the place where these translations can be modified so that they fit the needs. For instance, a “up” string might be translated to 100 while the “down” string translates to 0 (instead of 2, as provided by the MIB file).

Filters on the Table

Filters (figure 53) are used to filter out table rows which do not match the specified criteria. Each filter is displayed by a single line with its name on it. Filters on the same page will use a `AND` link, while different pages will use a `OR` link between them. Clicking on the filter displays the complete filter information and allows editing. The corresponding data will then be displayed in the left panel.

Filters are regular expressions that can be applied on every value of the table. The syntax in the one commonly used in either PERL or Java. Please refer to the [java regular expression pattern documentation](#) for more details.

The given example contains such a regular expression that only keeps values if AdminStatus is 'up'.

Other fields for this filter are filled so that this filter applies to the table we are polling (ifTable). In this case, the *Table Name*, *Table OID*, *IndexEntryMask* **MUST** remain empty.

Filtering on related tables is also possible; please refer to APG-SNMP-Collector-Admin-Guide for guidelines on this type of configuration. The provided default configurations can also be used as an example, especially tables contained in the GENERIC-INTERFACES mask.

Reference Tables

Reference tables are used to get properties for the table's polled values.

It basically matches two column types in the basic edit mode:

- Index Property: this type is configured in the advanced mode by setting the *Use Property Value in Variable Name* to *yes*.
- Property: this type is configured in the advanced mode by setting the *Use Property Value in Variable Name* to *no*.

The reference table can be an external table, which base OID is not the same as the polled table. It can also specify the same base OID as the polled table (in this case, the reference is the same as the polled table).

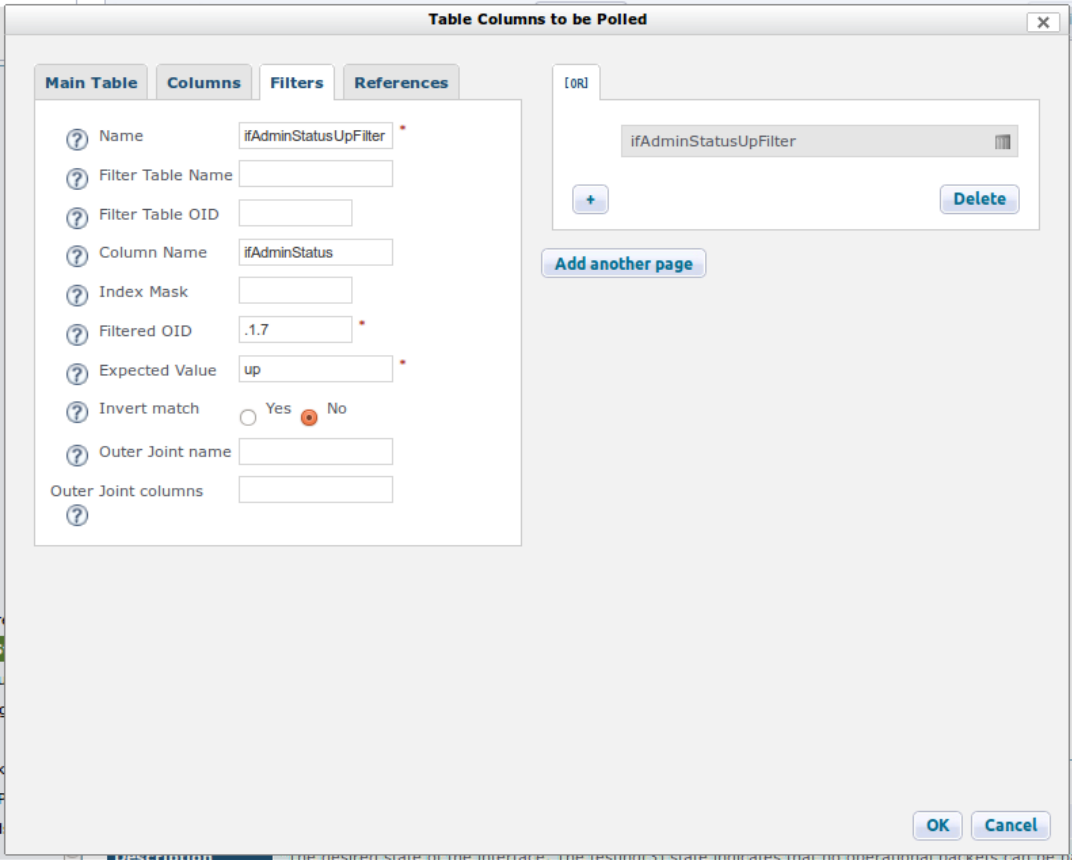


Figure 53: Table Filters Tab

All properties set in the reference table apply to all polled values in the table.
In this example case (figure 54), we are linking properties from the ifTable and ifXTable. The column

we are using to link these properties to the polled values is set in the *Main Table's next Reference OID*. In this case, we are using the *.1.100* column on the *ifTable* (remember: this index is not polled directly on *ifTable*). We are using several columns in this example:

- *ifTable.ifMtu*;
- *ifTable.ifDescr*;
- *ifTable.ifPhysAddress*;
- *ifTable.ifType*;
- *ifXTable.ifName*;

In order to get the *ifName* property from the *ifXTable* (as shown):

- *Table Name* must be set. A good naming convention is based on the table and column name, such as *ifTable.ifType*. Alternately, generated configuration names include the table name, the index column, the corresponding position in the parent table and the column name;
- *Table OID* is the reference table's base OID. In this case, it is *ifXTable's* base OID;
- *Parent Col Idx Position* refers to the position of the index column in the *Main Table's next Reference OID* field; 1 in this case;
- *Table Index OID* is the reference table index column used to link it with the base polled table (*ifTable*). Since *ifXtable* extends *ifTable* and does not have a declared index, *.1.100* is used here;
- *Index Entry Mask* is the index specification for the *ifXTable*. It is required because *ifXTable* does not have a declared index. It would also be required if the index column was declared but not accessible;
- *Property Name* should be set to a meaningful value (maximum length: 8 characters, lowercase. This is not enforced here but is standard in most report packs);
- *Property Value* is the column, in *ifXTable*, where the *ifType* property gets its value. In this case, this is the 1st column: *.1.1*
- *Hex Value* is used when recovering strings that have are hexadecimal values expressed as characters. *ifTable.ifPhysAddress* is such a string. Use the Mib Browser (get, walk or table view) to check if this is required.
- *Use Property Value in Variable Name* is normally set to no. When enabled, this tells the SNMP Collector to use this properties' value as part of the raw value's variable. It overrides the polled table's *Table Index* value.
- *Next Ref Table Cold Idx OID* is set only when a nested reference table, linked to this one, is used. This nested reference table is declared in the "Nested Reference 1" tab.

Table Columns to be Polled

Main Table **Columns** **Filters** **References**

Main Table's Next Reference OID: .1.100

Current References:

- ifTable..1.100.1.ifMtu
- ifTable.ifDescr
- ifTable.ifPhysAddress
- ifTable.ifType
- ifXTable.ifName**

New Delete

Reference Properties **Nested Reference-1** **Nested Reference-2** **Nested Reference-3**

Table Name: ifXTable.ifName

Table OID: .1.3.6.1.2.1.31.1.1

Joint column positions: 1

Table Index OID: .1.100

Index Entry Mask: oid.100

Property Name: ifname

Property Value: .1.1

Hex Value: ☐ Yes ☒ No

Use Property Value in Variable Name: ☐ Yes ☒ No

Next Ref Table Col Idx:
OID:

Cache table: ☐ Yes ☒ No

OK Cancel

Figure 54: Table References Tab

- *Cache table* is false by default. Setting this to true will cache the reference table until properties are refreshed for the polling group. This is enhance polling performance as it caches information

that is not updated frequently. It is not recommended to cache information used in building the raw value's variable however.

Note: About *Use Property Value in Variable Name*: when set to *yes*, the polled values use the property value for backend metric instance naming. Therefore, the value will not be pushed to the backend if this property does not contain any value as no proper identification can be generated for it.

References can be nested up to 3 times in a reference table, in order to reach tables that are four hops away from the polled table.

5

Agents configuration

The Agents node displays the Agents summary page (figure [56](#)).

Note: The SNMP agents configuration can be found in the `snmp-polling-distribution.xml` file, located in the Mib Browser configuration directory. All agents configurations are declared in this file, which is used by all collectors managed by the Mib Browser instance.

Then the Agents -> Collector name node displays this collector configuration overview.

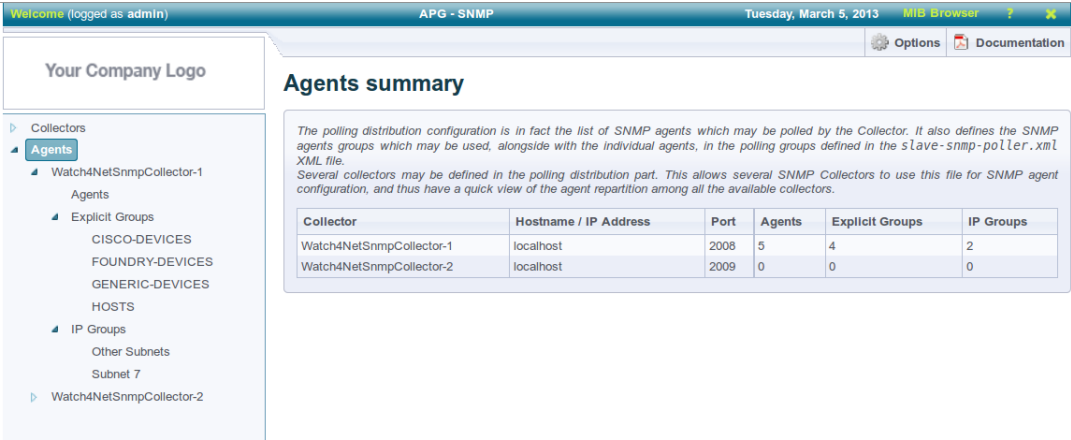


Figure 55: Polling Distribution Overview

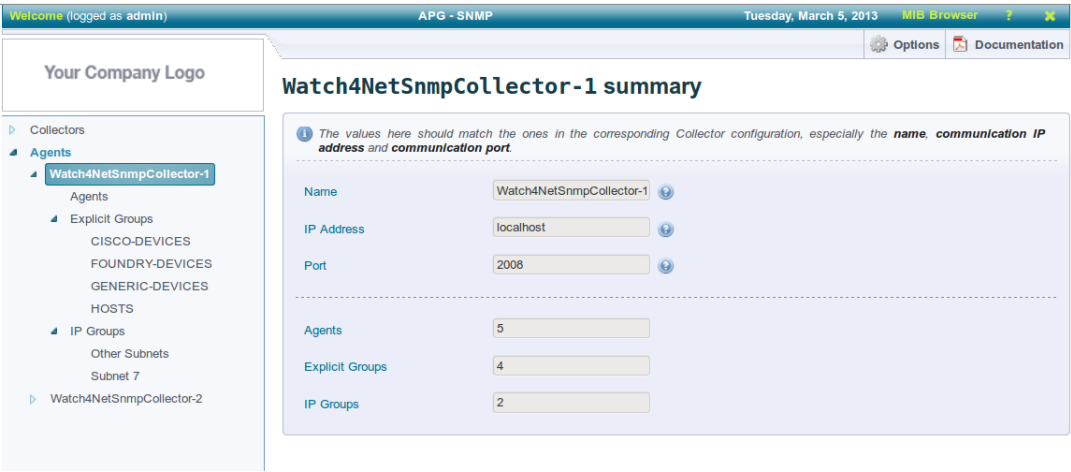


Figure 56: Agent distribution overview

1 SNMP Agents

This section lists the SNMP agents that are available the the collector.

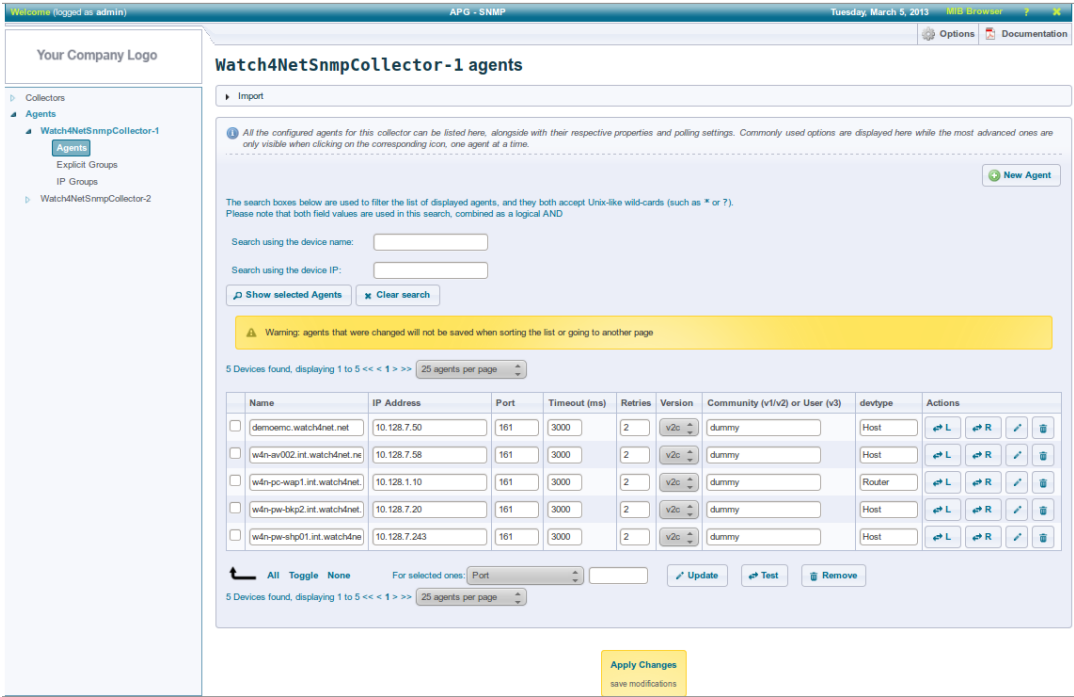


Figure 57: No Configured SNMP Agents

This page offers several agent management functionality:

- Create a new SNMP agent by clicking the “New Agent” button;
- Agent search, by its Name or IP address (glob expression is supported);
- Agent edit, by clicking its “Pencil” (edit) button;

- Multiple agent modifications, available for port, timeout, retries, community string and devtype;
- Agent reachability test (from the server and collector);
- Agent delete (through its “trash can” button or by selecting them and clicking the “Delete” button);

The most commonly modified agent settings are directly available in the table:

- **Agents / Devices** which is the name of the device (this will fill the `device` property in *APG*)
- **IP Address** which is the IP to use when communicating with the agent (this will fill the `ip` property in *APG*)
- **SNMP Port** number on which the device’s SNMP Agent is listening to
- **Timeout** is the number of milliseconds before giving up on a connection attempt (1000ms is 1 second)
- **Retries** is the number of retries before giving up when trying to contact an agent
- **SNMP Version** which is the version of the SNMP protocol the agent answers to (usually v1 or v2c)
- **SNMP Community** is the community string to use for authentication purposes (on SNMP v1 and v2c agents. It is the user name for SNMP v3)
- **devtype property** which sets the `devtype` property in raw values collected from the agent

Adding a new agent This is done by simply clicking the “New Agent” button. An empty line is then displayed in the agent list. All fields must be filled with their correct value. Settings that do not appear on the line, such as SNMP v3 and other advanced parameters (figure 58) can be accessed by clicking the “Pencil” (edit) button, which displays the agent edit dialog.

All agent settings are documented in their help pop-up. Mandatory fields have been highlighted with a red asterisk at the right of the field.

Once done, save the list of agents by clicking the “Save” button.

Adding agents via a seedfile Agents may also be added by importing a correctly formatted seed file (figure 59).

Possible formats vary, but missing settings will need to be defined manually if default values do not fit. Valid seed file formats are:

- basic: IP,name
- standard: IP,name,[community string],[port],[SNMP version]

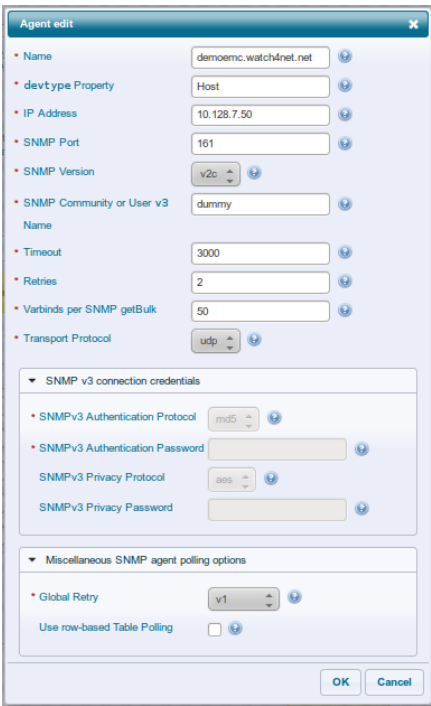
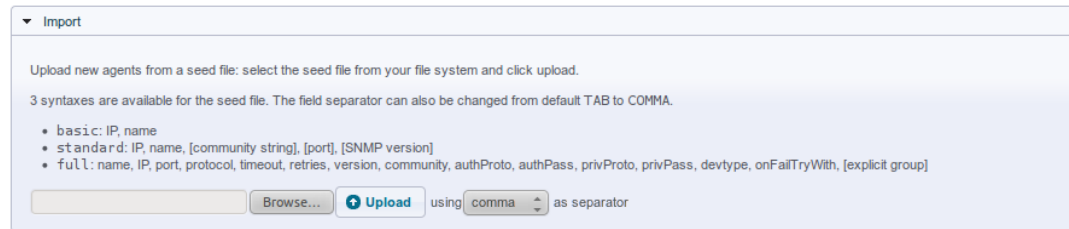


Figure 58: Agents edit dialog

- full: Name,IP,port,protocol,timeout,retries,version,community,authProto,authPass,privProto,privPass,devtype,[explicit-group]

Note: Standard field separator is COMMA; TAB is also supported.

Upon successful upload, a message with a quick summary of the operation is displayed, explaining



Import

Upload new agents from a seed file: select the seed file from your file system and click upload.

3 syntaxes are available for the seed file. The field separator can also be changed from default TAB to COMMA.

- **basic**: IP, name
- **standard**: IP, name, [community string], [port], [SNMP version]
- **full**: name, IP, port, protocol, timeout, retries, version, community, authProto, authPass, privProto, privPass, devtype, onFailTryWith, [explicit group]

using as separator

Figure 59: Agents import

any error if some were found in the file. For instance, a seed file containing 150 agents may have 3 invalid ones (invalid line numbers are displayed in the import confirmation message).

2 IP Pattern Grouping

Two types of grouping may be used to select the SNMP agents defined in the section above. The first one is based on IPs and ports of the agents.

This group only needs a name and a pattern. Any defined agent `IP:port` will then be matched against that pattern to see if it fits the group.

Click the IP groups node (figure 60) to display a quick summary defined IP groups with the number of matching SNMP agents. IP groups can be created and deleted from this page.

Click the IP Group -> Group node (figure 61) to display the IP group and its list of selected agents.

For IP Groups, the *expression type* setting can be modified. Two pattern syntax flavors are supported:

- **Unix-like** patterns are the ones commonly used on Unix systems. They use `*` as wildcard for an unknown number of unknown characters and `?` as wildcard for a single unknown character. One can also use syntaxes like `[2-7]` in order to specify a digit between 2 and 7 (included) They are quite simple and provide a quick means to group IPs.
- **Perl / Java** regular expressions are more complicated but also more powerful than the previous ones. They have been made available for advanced users who require more expression power.

Note: Selection Patterns may contain the port as well. For instance, `10.128.1.*:161` will only select agents

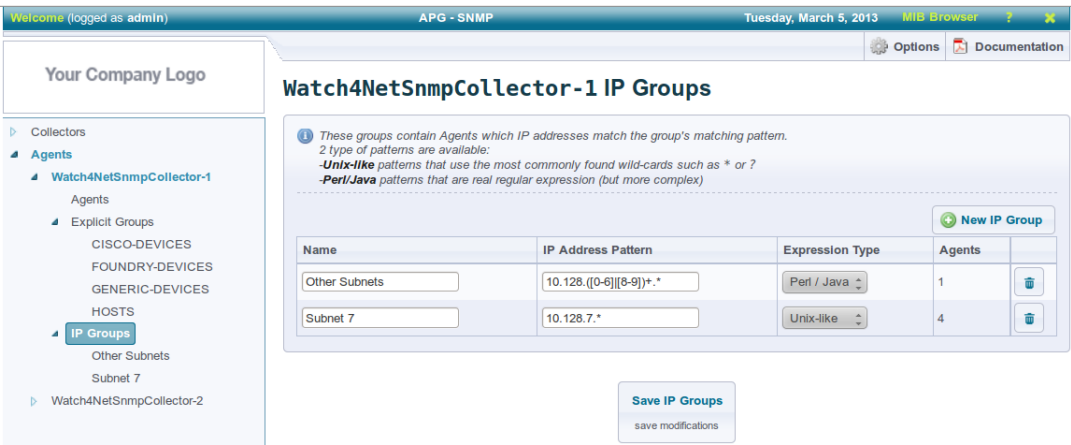


Figure 60: Groups Defined on IPs

on the sub-network 10.128.1.0 that listen on port 161 while all others (even the ones on the sub-network 10.128.1.0 but that listen on other ports) will not belong to the group.

3 Explicit Grouping

Explicit grouping is another way of creating groups. It works by manually specifying the ids of agents belonging to the group. The agent id is its IP address and port. Thus, if a device IP changes, or if the agent is set to listen on a different port afterwards, then the agent will no longer belong to the group.

The main page (figure 62) shows a list of configured groups along with the number of agents they select. This count includes both *valid* agents (that exist in the [SNMP Agents](#) section) and *invalid* ones.

On each explicit group section (see figure 63), a summary of all selected agents is displayed in a list box. There, invalid ones are highlighted so that they can be either fixed or removed from the list.



Figure 61: IP Grouping and Selection

Graphical display depends on the browser being used.

Modifying the list of agents has been made quite easy through a multiple selection dialog. This popup (figure 64 page 94) displays both available and selected agents list, with their names as tooltip and provides a quick means to switch them from one list to the other.

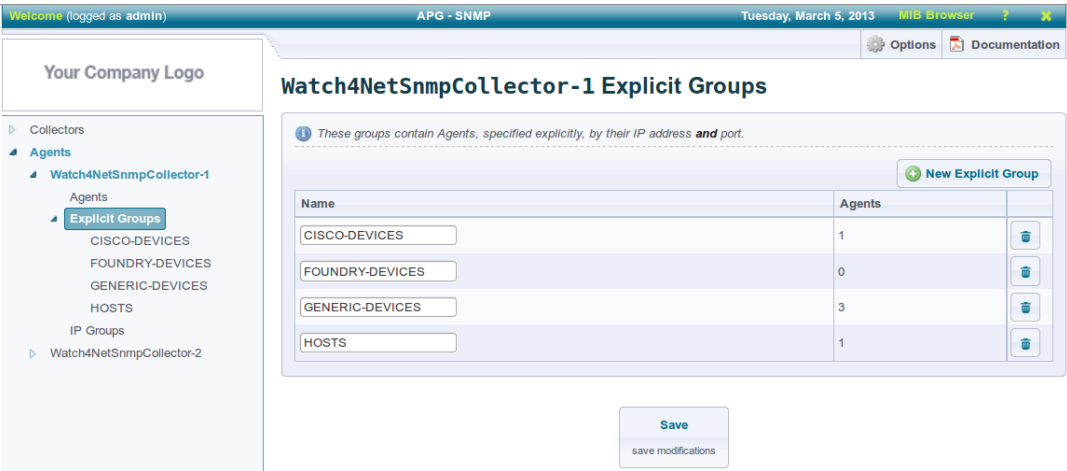


Figure 62: Explicit Groups



Figure 63: Explicit Grouping

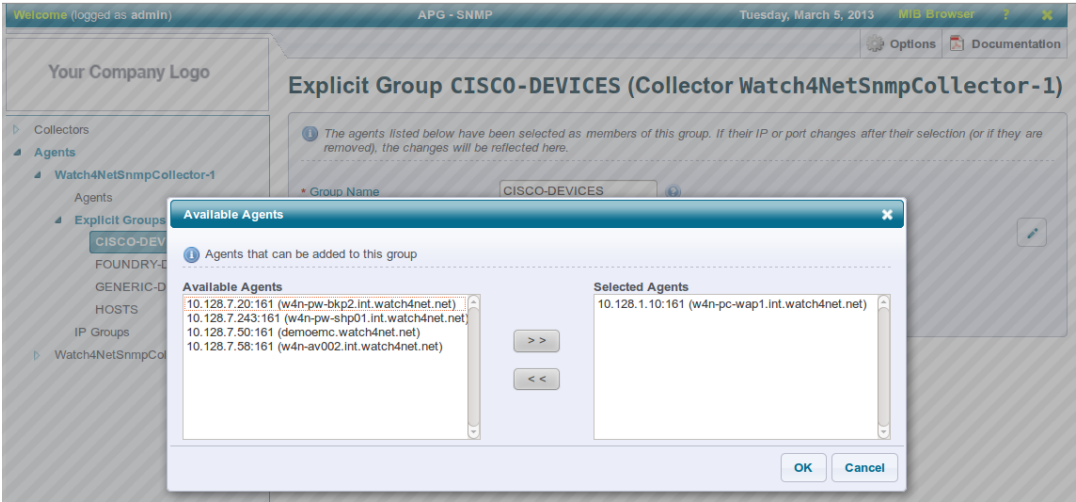


Figure 64: Explicit Grouping and Selection

6

Synchronization

Synchronization between the SNMP Polling Configuration and a running SNMP Collector is done via the Synchronization page. Basically, this means that any changes listed below will take effect on the collector only once the configuration is pushed to it:

- Collector configuration settings (Global, Communication, Polling Threads, Misc);
- Masks (properties, single oid metrics, tables);
- Polling groups;
- Agents groups (IP or Explicit groups);
- MIB files (added, updated or removed);

Before trying to synchronize a configuration, there are a few requisites to be met.

- The SNMP Collector *must* be up and running, even if it is not polling anything at all. By default, without any configuration, the collector will simply standby, waiting for orders. It should also be bound to the specified interface if needed.
- The SNMP Collector must accept configuration commands from the server.

- The configuration, including polling groups, masks and MIB files must be valid.

WARNING: Be careful when sending a configuration which included IP addresses that are allowed to communicate with a given collector. Make sure that the server used to send the configuration has its IP included in this list, otherwise you will lock yourself out as the collector will no longer accept commands from the server once the new configuration has taken effect.

The configuration panel contains three major sections, and the first one will give a quick status of the SNMP poller, which becomes quite useful when a communication error occurs for some reason.

1 SNMP Poller

If the poller cannot be reached for a specific reason, this reason will be explained here, at the end of the *SNMP Poller Configuration* section. Common errors include:

- `no route found` if the given IP cannot be reached
- `connection refused` if the poller is not answering on the given IP or port

Watch4NetSnmppCollector-1 synchronization

Status

If the collector is reachable, you can start it, stop it and restart it from here.

Could not reach the collector

Possible causes for this problem are:

• collector manager is not started

• web server host IP is not in the collector Allowed IPs for Communication list

• collector listening port is blocked by a firewall or is otherwise unreachable from web server

Archived Configurations

When a new save is made, the configuration is archived using the current time of the archival.

Archived configurations can be downloaded as a ZIP file, uploaded to the collector or set back as current edited configuration.

New Save Point

Archival Date	Status	Actions
2013-03-06 11:10:13		
2013-03-06 11:09:57		
2013-02-12 09:48:50		

Figure 65: Communication error while trying to reach the SNMP poller.

Figure 65 shows an error instance. In this case, the SNMP poller has not been started, and thus cannot answer the server's request. The configuration can still be archived and downloaded, but not uploaded to the poller.

If started and reachable, the SNMP poller will answer the server and provide some relevant information about its status (see figure 66).

The Snmp Poller Status section indicates whether the poller is collecting data (meaning the collecting process has been started) or is simply idle (dormant) and waiting for the start command. The poller can be stopped (*Stop* icon), started (*Play* icon) or restarted (*Stop-Play* icon) from here. A restart causes the Snmp Poller to stop, read its configuration files and start collecting again.

Note: It is preferable to stop a heavily loaded collector before pushing a new configuration to it.

MIB Browser User Guide

97

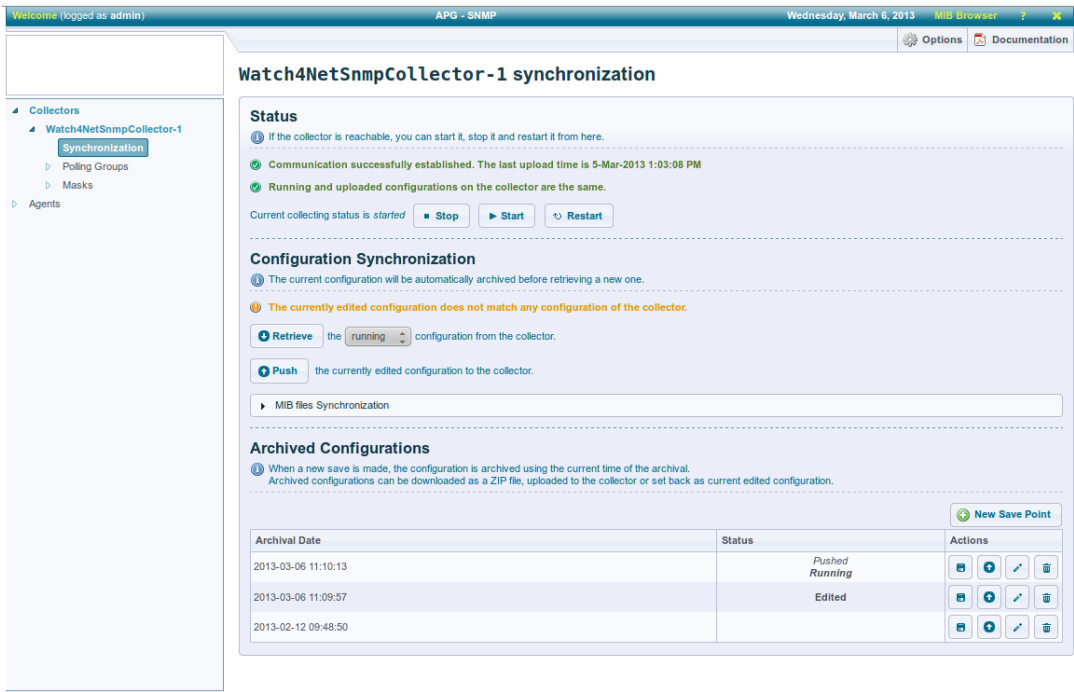


Figure 66: Poller Status Overview

The Configuration Synchronization section allows retrieval of the running or stored configurations (loaded by the poller or stored in its configuration files). The current configuration can also be pushed to the collector in this section.

The collector must be restarted (or started) for the new configuration to take effect.

The currently edited configuration (the one selected for edition in the web interface) can be downloaded to the user computer or uploaded to the SNMP poller. Optionally, this configuration can be archived as reference. It will then appear in the following section.

2 Configuration Archival

Older configurations may be archived for rollback purposes, or simply to get an history of their evolution. They are listed in that section, the most recent one being at the top of the list.

Archived Configurations		
 When a new save is made, the configuration is archived using the current time of the archival. Archived configurations can be downloaded as a ZIP file, uploaded to the collector or set back as current edited configuration.		
		+ New Save Point
Archival Date	Status	Actions
2013-03-06 11:10:13	<i>Pushed Running</i>	   
2013-03-06 11:09:57	<i>Edited</i>	   
2013-02-12 09:48:50		   

Figure 67: Available Configuration Archives

The *status* column contains information about the archive location. There are four different displays, each one with a specific meaning:

- No status at all means the archive has no relation with any of the available configurations
- *Edited* that means the archive files are the same as the ones currently edited
- *Pushed* for the configuration files that match the ones uploaded to the poller
- *Running* highlights the fact that this configuration is currently running on the SNMP poller

Archives act as save points. Each one can be deleted, downloaded to the user computer, directly uploaded to the SNMP poller (which will also trigger the next option), or set as current configuration. This last option will create a backup of the previously edited configuration if it does not match any of the already archived files.

3 MIB Files

MIB files can be synchronized between the SNMP poller and the server. In theory, the only MIB files required on the poller side are the ones specified in every mask used by this poller (as well as their dependencies, which are not listed in the masks). Synchronizing MIB files between the server and collector guarantees that both sides have access to exactly the same MIB files.

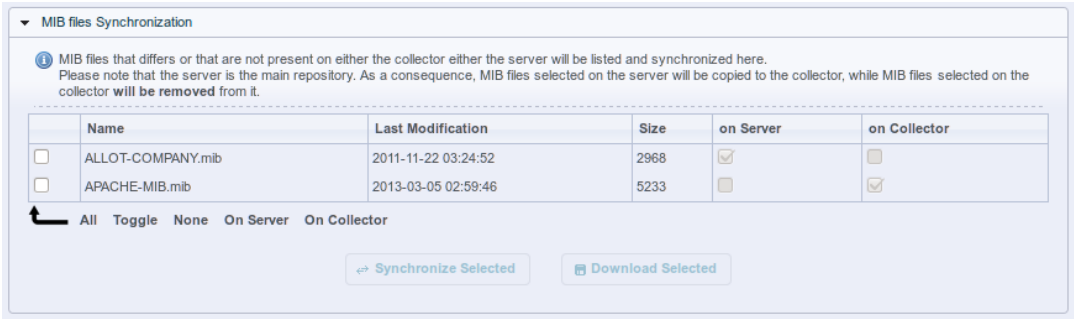
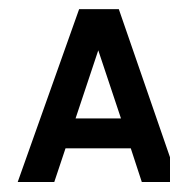


Figure 68: MIB files synchronization

The synchronization panel is displayed on figure 68 page 100. All files that do not match appear here along with their size and location. Files need to be selected here for the synchronize action to be applied.

The server MIB files directory is the master: MIB files present only on the collector are *deleted* when MIB file synchronization occurs. MIB files present only on the server side are *uploaded* to the collector upon MIB file synchronization.



Context File Sample

This file is located in Tomcat/conf/Catalina/localhost/ and named mib-browser.xml.

```
<Context>

  <!--
    Relative paths are based on the value of the 'apg.resources.webapp.dir' system
    property, except for ping which is run from Tomcat's home directory.
  -->
  <Resource auth="Container" name="bean/SettingsFactory"
    type="com.watch4net.apg.mibbrowser.config.SettingsFactory"
    factory="org.apache.naming.factory.BeanFactory"
    configPath="mib-browser" mibPath="mib-files"
    pingPath="ping" />

  <!-- Gives access to the users, profiles, roles and rights. -->
  <ResourceLink name="jdbc/master" global="jdbc/master" type="javax.sql.DataSource" />

  <!-- If SSO is disabled, uncomment the following element. -->
  <!--
```

```
<Realm className="org.apache.catalina.realm.DataSourceRealm"
dataSourceName="jdbc/master" localDataSource="true"
userTable="users" userNameCol="username" userCredCol="password"
userRoleTable="users" roleNameCol="rolename" />
-->

<!-- This valve MUST be the first one to fix encoding. -->
<Valve className="com.watch4net.apg.v2.gui.tomcat.EncodingFixValve" />

</Context>
```