



EMC® Watch4net®

Version 6.4u2

SNMP Device Discovery User Guide

P/N: pending

Rev : 49579

EMC²

Copyright ©2009 - 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
Purpose	iii
Intended Audience	iii
Prerequisites	iv
Where to get help	v
1 Product Overview	1
2 Definitions	3
3 Workflow	5
4 Graphical User Interface	7
1 Login Screen	7
2 Dashboard	9
3 Device Sources	10
4 Devices	13
5 Capability Discoverers	16
6 Collectors	18
7 Default Masks	21
8 Default MIBs	23
9 Event Log	24
5 Device Source Configuration Screens	27
1 Seed File Source	29
2 SNMP Discovery Source	31
3 SMARTS Source	34
6 Device edit	39
7 SNMP Capability Discoverer edit	43

8	<i>SNMP Collector Edit</i>	47
	Appendix	50
A	SeedFileSource Input File Format	51
B	SNMP Device Definition	55
C	Device Distribution Logic	57
D	SNMP Capability Discovery Actions Provided by Default	59
E	Controlling the GUI via a Command Line Interface	61
F	SNMP Capability Discovery Actions: Language Definition	63
G	Current Known Limitations	69
H	Application Properties	71
	1 Base configuration files directory	71
	2 Enable the distribution of devices that do not have capabilities	72
	3 Customize device list	72
	4 Customize the event log size	72
	5 Configuration auto-deployment	72
I	Polling periods priority	75

Preface

Purpose

The purpose of this guide is to provide configuration and usage information about the *SNMP Device Discovery* application.

This application can do the following:

- Import *Devices* from various sources.
- Find the *Devices* capabilities.
- Distribute the *Devices* among data collectors.
- Manage data collector configurations.

Intended Audience

APG administrators, system integrators, as well as users responsible for managing data collection should read this document.

The *SNMP Device Discovery* currently works exclusively with the *SNMP Collector*.

Prerequisites

- The *APG* master database must be installed and running in order to use this application.
- At least one user must have the rights to use this application (the “admin” user has such rights).
- A Device-Discovery license must be installed.

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.

Product Overview

The *SNMP Device Discovery* is a frontend application used to help manage pools of *Devices* which can be queried by one or more data collectors. One of its main functionalities is the ability to import *Devices* from external sources such as: Network Management Systems (EMC Smarts, HP OpenView, Proviso), separated-field seed files and other Watch4Net import modules (for example, the SNMP Discover Source).

The imported *Device* properties may be edited through this application. This application can also discover the capabilities supported by the *Devices*, as well as provide a means to specify them manually on each *Device*. Once capabilities are set on the *Device*, it becomes a *Device* which can be assigned to a suitable data collector, which will poll the *Device* periodically.

Currently only SNMP enabled devices are supported.

Definitions

This is a list of terms and definitions used in this document.

- *Capability*: This corresponds to the information (performance, statistics or other) which can be provided by a virtual or physical device. This definition is specific to every data collector type. For instance, with regards to Watch4Net's *SNMP Collector*, a *Capability* is polled by a <snmp-mask>. In this context, *Capability* and snmp-mask are equivalent.
- *Capability Discoverer*: This is a module used to find capabilities supported by a *Device*. There is only one such module per device *type* (for instance: SNMP). It uses *Discovery Actions* to query a *Device* for its capabilities.
- *Collector*: This is a system-representation of a real data collector (Such as Watch4Net's *SNMP Collector*, XML Collector or other) instance. It provides enough information to allow communication with its linked data collector.
- *Device*: This is a physical or virtual device or piece of equipment which can be polled by a data collector. The capabilities it supports have been identified (either manually or by a *Capability Discoverer*) or not. A *Device* can be assigned to a collector which will polls it for performance statistics and other information. The *Device* ID is typically (for SNMP) made of the device name and SNMP port, as well as the collectors to which it is assigned (if any) and the collector groups

to which it is assigned (if any).

- *Device Source*: This is a device source used to communicate with another system or external component in order to import *Devices*.
- *Discovery Action*: This contains the actions required to identify a capability on a *Device*. *Discovery Actions* may be recorded in an XML file (which is the case for *SNMP Devices*).
- *SNMP Collector* configuration files:
 - *slave-snmp-poller.xml*: This file contains the polling group definitions as well as other *SNMP Collector* specific parameters;
 - *snmp-polling-distribution.xml*: This file contains the list of SNMP agents assigned to the *SNMP Collector*, as well as the communication information (IP Address and port) used to reach it;
 - *snmp-masks.xml*: (optional) This file contains the snmp masks - equivalent to Capabilities in this case - which may be polled by the *SNMP Collector*;
 - *translations.xml*: (optional) This file contains a list of String-Value translations used by the *SNMP Collector* when polling integers mapped to strings.

Note:

The *SNMP Device Discovery* comes with a set of Capabilities and Translations. These configurations will be used by default unless the Collector locks the Capabilities, by providing its own *snmp-masks.xml* and *translations.xml* files. See section 8 for details on Collector addition.

- *Import Priority*: This applies to both a *Device* and *Device Source*. Lesser values mean a higher priority. Import Priority checking is done when importing *Devices* from a *Device Source* and the same *Device* already exists in the system. The “Update Action Policy” will occur only if the priority check passes, that is if the new *Device*’s priority value is higher or equal to the existing *Device* import priority. The highest Import Priority is -1. Priorities “higher” than this (-2,-3, etc.) are recorded as -1.

3

Workflow

This section describes typical *SNMP Device Discovery* workflows. The “Basic Configuration” and “Basic Operation” workflows are covered.

Basic configuration

1. Create a *Collector*: This should be the first step to perform when using the *SNMP Device Discovery*. This assumes that at least one supported data collector instance is installed and running somewhere on the network and is visible and accessible from the host on which the Device-Discovery application is running.

Note:

SNMP Collector configuration files may be retrieved automatically by the *SNMP Device Discovery* in order to easily create a *Collector* linked to it. The *Collector* may also be created manually, by specifying files directly on the host.

2. Create a *Capability Discoverer*: Creating a *Capability Discoverer* is optional, but it will save a lot of work when discovering capabilities on several devices. It is in fact possible to set the supported

capabilities manually on every *Device* but this is error-prone and very time-consuming.

Note:

An *SNMP Capability Discoverer* is currently provided by default.

3. Define a *Device Source*: This is another optional step as it is possible to manually create a *Device*. A *Device Source* will import *Devices* from an external component, saving a a lot of time in the process.

Basic operation

The steps in this section can be performed in any order once this condition has been met:

- A *Collector* has been created and it is ready to accept *Devices*.
1. Import *Device*: This will populate the Device List with *Devices*. All new *Devices* should appear in the list, unless an error occurred while importing them. Their status should be "Approved", unless another import policy action was set.
 2. Discover capabilities on the *Devices*. Once done, the number of capabilities discovered for the *Device* should be visible.
 3. Generate the configurations. This will assign every *Device* to a suitable *Collector*, and, by extension, to a suitable data collector.
 4. Apply the configurations so that the generated configurations are sent to data collectors.

The particularities of each *APG* deployment will determine the order with which you perform these steps. *Devices* may be edited, excluded or deleted at any time. They may also have their Capabilities rediscovered as required. Data collector configurations may be regenerated as needed.

4

Graphical User Interface

This section describes the main screens of the *SNMP Device Discovery*. These screens are accessible from the application's menu at the left of the screen.

Note: Please note that JavaScript need to be enabled in the browser for the application to work.

1 Login Screen

The login screen is similar to the one used with the *APG Frontend*.

An alert will be displayed if the database is not available or if there is a problem with the license.

Access this screen to log into the *SNMP Device Discovery* by entering the host address, followed by the Tomcat listening port and the *SNMP Device Discovery* instance name. For example, if the default installation is used on the localhost, the *SNMP Device Discovery* login URL would be: <http://localhost:58080/device-discovery>.

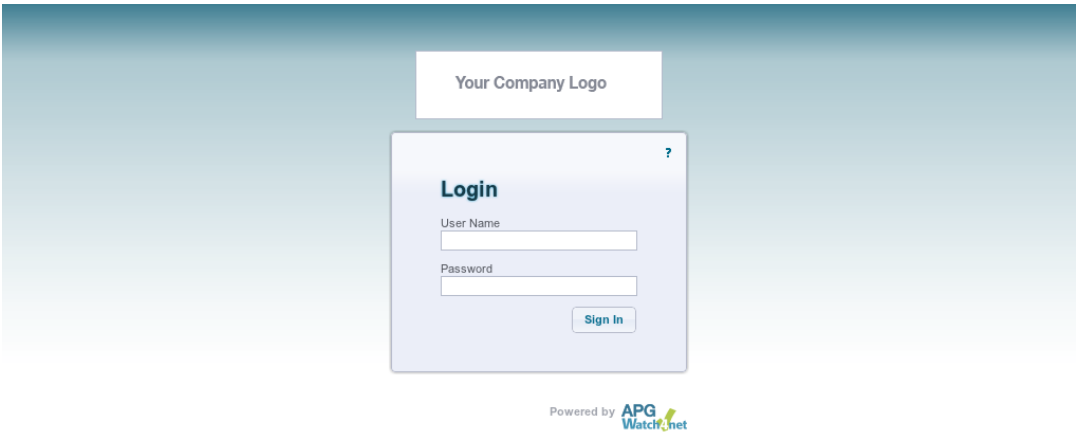


Figure 1: Login Screen

Any user with the required credentials (access rights to the Device-Discovery module) can log into the application from this link.

2 Dashboard

This screen shows the main actions relative to the *SNMP Device Discovery* use. It also shows basic information and alert messages about Capability Discoverers, Collectors, Device Sources and Devices.

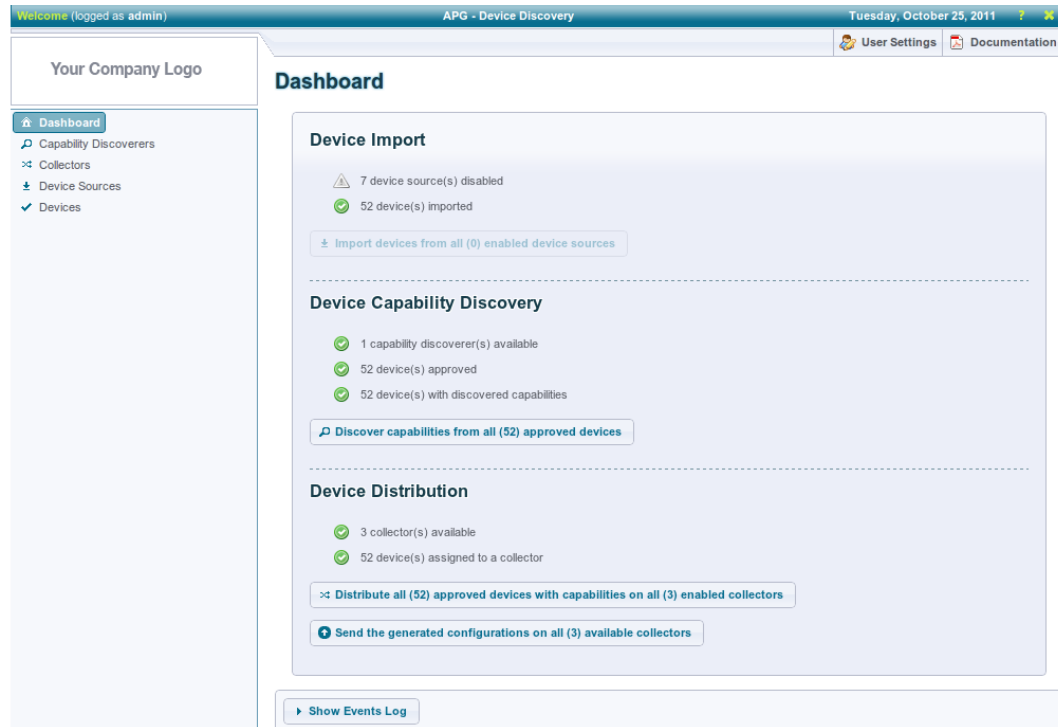


Figure 2: Dashboard

3 Device Sources

This screen shows the information relative to the configured *Device Sources*.

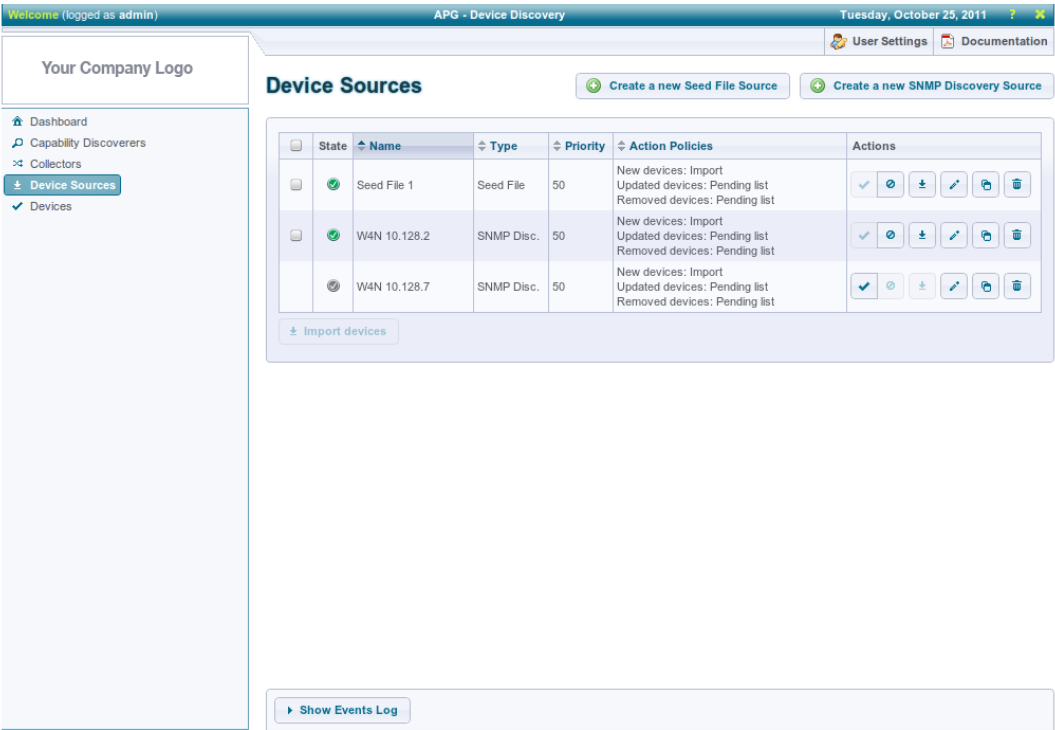


Figure 3: Device Sources

Note: Some Device Sources may not be available is they are disabled by the user or if no SNMP Collector is

available in case of an SNMP Discovery Source.

Information

The information displayed on the *Device Sources* screen include:

- State: The *Device Source* state.
 - A green check means that the *Device Source* loaded correctly and is ready to be used.
 - A gray check means than the *Device Source* is disabled.
 - A red X means that some errors occurred while loading the *Device Source*.
 - A spinning wheel means that the *Device Source* is currently importing devices.
- Name: The *Device Source* name. This must be unique among the *Device Sources*.
- Source type: This is the *Device Source* type.
- Priority: This is the imported *Devices* priority when they are imported through the source. A lower number means a higher priority. The priority has an impact on the resulting actions when importing a device.

Note:

If a *Device* already exists in the *SNMP Device Discovery* and it is imported again, 4 things can happen depending upon the *Device Source* “Updated devices” action policy and the existing *Device* import priority.

If the priority of the new *Device* is lesser (higher number) or equal to the existing *Device*, then nothing will happen.

Otherwise, the “Updated devices” action policy takes effect, either: updating the existing *Device*, updating the existing *Device* and put in the pending list or do nothing.

The same goes when the *Device* is 'deleted'.

- Actions Policies:
 - New devices: This is the action to be performed when a new *Device* is imported.
 - Updated devices: This is the action to be performed when an updated (existing) *Device* is imported.
 - Removed devices: This is the action to be performed when a *Device* is removed.

Functionality

The *Device Sources* screen provides the following functions:

- Import devices: this imports *Devices* from the selected *Device Sources*.
- Create a new Device Source: create a new *Device Source*.
- Enable/Disable the device source: this allows you to adjust which Device Sources are imported on a regular basis.
- Edit: an *Device Source* may be edited by clicking on the *Edit* button.
- Copy: click the *Copy* icon in this column to generate a copy of the *Device Source*. Provide a new *Device Source* name and save it to complete the process.
- Delete: click the *Trash* icon to delete the *Device Source*. A confirmation pop-up will appear before the *Device Source* is deleted.

4 Devices

The *Devices* screen lists all existing *Devices*.

Welcome (logged as admin) APG - Device Discovery Tuesday, October 25, 2011 User Settings Documentation

Devices

Add new Device

43 devices found, displaying 1 to 10

State	Name	Host	Type	Source	Import Priority	Cap.	Assigned to	Actions
✓	10.128.7.1	10.128.7.1	Node	W4N 10.128.7	50	2	W4N-SNMP1-P2008	[Icons]
✓	10.128.7.9	10.128.7.9	Host	W4N 10.128.7	50	3	W4N-SNMP1-P2009	[Icons]
✓	10.128.7.24	10.128.7.24	Node	W4N 10.128.7	50	2	W4N-SNMP2-P2010	[Icons]
✓	10.128.7.25	10.128.7.25	Node	W4N 10.128.7	50	2	W4N-SNMP1-P2008	[Icons]
✗	kb.watch4net.com	10.128.7.26	Host	W4N 10.128.7	50	3	W4N-SNMP2-P2010	[Icons]
✓	10.128.7.27	10.128.7.27	Node	W4N 10.128.7	50	2	W4N-SNMP2-P2010	[Icons]
✓	w4n-ll-appcluster1-vip.int.watch4net.net	10.128.7.40	Node	W4N 10.128.7	50	2	W4N-SNMP1-P2009	[Icons]
✓	w4n-ll-appcluster2-vip.int.watch4net.net	10.128.7.41	Node	W4N 10.128.7	50	2	W4N-SNMP2-P2010	[Icons]
✓	10.128.7.43	10.128.7.43	Node	W4N 10.128.7	50	2	W4N-SNMP1-P2008	[Icons]
✓	10.128.7.44	10.128.7.44	Node	W4N 10.128.7	50	2	W4N-SNMP1-P2009	[Icons]

43 devices found, displaying 1 to 10

Actions on selected device(s): Discover Capabilities Execute Action

Show Events Log

Figure 4: Devices

Note: It is possible for some very similar *Devices* to appear on this list, as the *Device* id uses the collector name and group collector names, which the *Device* is bound to, as well as the port. Both instances are different and are very likely to end up on different collectors after they have been distributed. This

is not a bug as it allows IP overlapping and having multiple client networks use the same IP address mappings for different devices.

Information

The *Devices* list provides a information about *Devices*.
It provides a way to differentiate *Devices*, which have a state bullet.

- A green check means that the *Device* is approved and it can be assigned to a collector.
- A yellow check means that the *Device* is approved but something is wrong.
- A tools icon means that the *Device* is waiting for approval in the pending list and won't be assigned to a collector.
- A gray check means than the *Device* is excluded from the management process and won't be affected by any management action.
- A red X means that the *Device* capability discovery failed.
This may be due to the device being physically down, unreachable or otherwise unresponsive. This may also be caused by misconfigured SNMP settings. If you don't want to use the *Device*, you can explicitly exclude it.
- A spinning wheel means that the *Device* is currently discovering capabilities.
To interrupt the capabilities discovery, see the *Capability Discoverers* page.

The *Device* status can be changed by using the Actions button.

- Click on the *Check* icon to *approve* the *Device* (green/yellow check icon or red X icon).
- Click on the *Wrench* icon to put the *Device* in the *pending* list (tools icon).
- Click on the *Cancel* icon to *exclude* the *Device* (gray check icon).

Other *Device* information includes:

- Name: the *Device*'s real name.
- Host: the *Device*'s IP Address or host name.
- Type: this describes the physical device type (default: Node).
Typical values include: *Host*, *Router*, *Switch*, *Node*.
- Source: the *Device Source* where the *Device* comes from or the user name for manually created *Devices*.

- Import Priority: the device import priority.
For more information see *Import Priority* on page 11.
- Cap.: the number of capabilities discovered on a *Device*.
- Assigned to: the collector name to which the *Device* is currently assigned to (in the *SNMP Device Discovery* generated configurations).

Functionality

The following actions may be performed on *Devices* from the *Devices* page.
The actions are also available in a drop-down list under the table for bulk operations.

- Approve: all selected *Devices* will be *Approved*.
- Send to pending list: all selected *Devices* will send to the *Pending* list.
- Exclude explicitly: all selected *Devices* will be excluded.
- Discover Capabilities: discover capabilities on all selected (and non-excluded) *Devices*.
- Add a new *Device*: show a dialog to manually add a *Device* to the *SNMP Device Discovery*.

Filtering and Sorting

The *Device* list may be filtered by clicking on one of the column header funnels and specifying a filtering expression.

To cancel the current filter, remove a filter criteria or click on the the *Cancel Filtering* link

Only 2 exceptions in behavior to this rule:

- State: filter on the device state by selecting one of the possible filter in the list.
- Capabilities: filter on the number of capabilities or on the capability names.

The *Device* list may be sorted by clicking on table header label.

5 Capability Discoverers

The *Capability Discoverers* page displays the configured *Capability* Discoverers. Only one *Capability* Discoverer can be configured for a given data collector type.

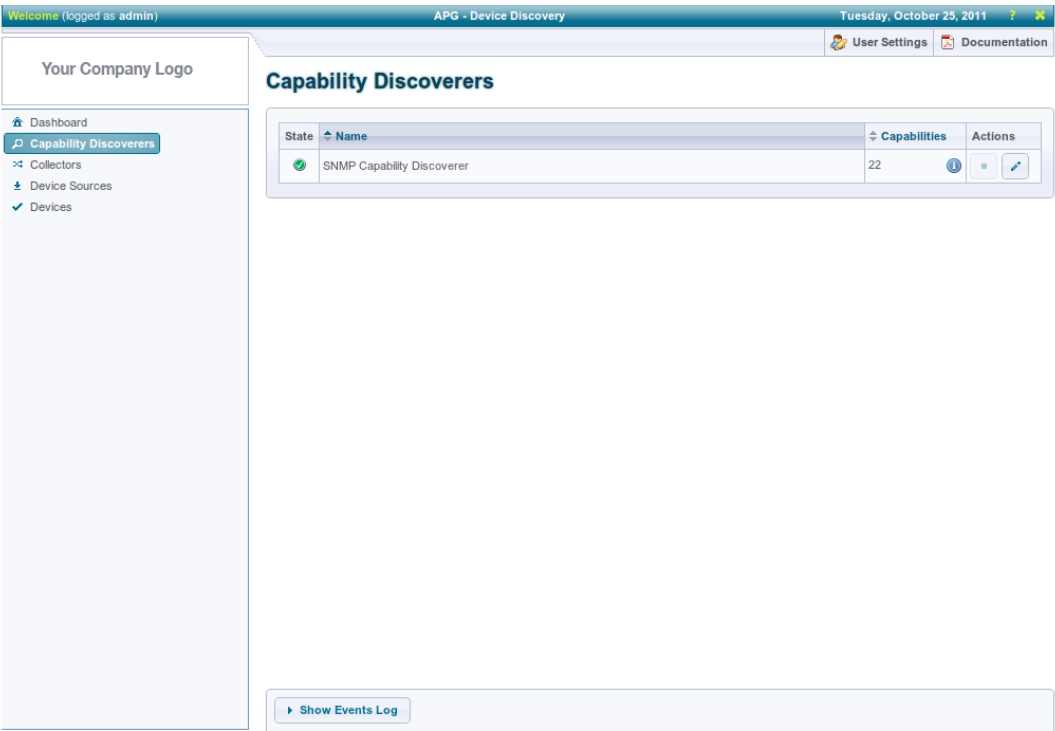


Figure 5: *Capability* Discoverers

Information

This *Capability Discoverers* list displays only 3 information fields:

- State: the module state.
 - A green check mark means that the module loaded correctly and is ready for use.
 - A red “X” means an error occurred while loading the module. Please check the Tomcat logs to find out more about the problem.
 - A spinning wheel means the the *Capability Discoverer* is currently being used. A *Capability Discoverer* discovery process may be interrupted by clicking the 'stop' icon in the 'Actions' column.
- Name: the name of the *Capability Discoverer* module. This name must be unique among *Capability Discoverer* modules configured in the application.
- Capabilities: the discovery actions capabilities of the *Capability Discoverer* module. Only the capabilities listed here can be managed in the *SNMP Device Discovery*.

Functionality

The *Capability Discoverers* provide the following functionality:

- Interrupt: Click on the 'stop' button to interrupt a capability discovery. Only available when the capability discoverer is currently used for the discovery of device capabilities.
- Edit: clicking on the 'edit' icon will display its edit screen.
- Add: Create a new *Capability Discoverer*. The button is visible only if it's possible to add a new *Capability Discoverer*. You can only add a *Capability Discoverer* if none exist for the same device type.

Note: As of version 2.0, it's not possible to remove a Capability Discoverer from the web interface anymore but it's still possible to remove the Capability Discoverer by editing the *SNMP Device Discovery* configuration file on the server.

6 Collectors

The *Collectors* page displays information about the configured collector. It also provides management functionality and information about Device distribution (assignment) as well as the data collector synchronization state.

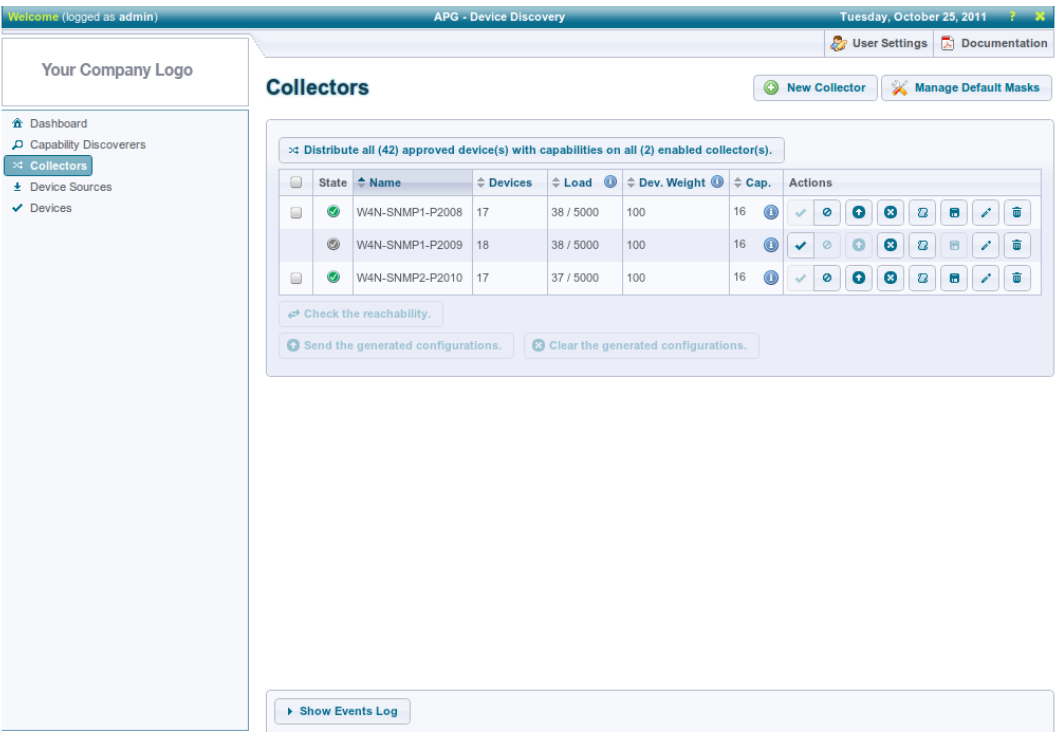


Figure 6: Collectors

Information

The *Collectors* page displays the following fields:

- State: the collector state.
 - A green check mark means that the collector is configured properly and reachable.
 - A gray check mark means that the collector is disabled. It doesn't accept *Device* in its configuration and its reachability is not checked.
 - A yellow '!' means that the collector configuration is not synchronized with the *SNMP Device Discovery* configuration.
 - A red 'X' means that the collector is unreachable or an error has occurred when loading the module. Please check the Tomcat logs for further details.
 - A wheel means that the collector is busy because a device distribution is in progress or a new configuration is being sent or its configuration is being cleared.
- Name: this is the name of the collector.
- Devices: the number of devices assigned to the collector.
- Load: these 2 values represent the current and maximum load the collector. The current load reflect the used resources (in terms of memory and CPU) on the collector. The maximum load is a user specified value which should reflect the available resources (in terms of memory and CPU) available to the collector. *The maximum load is not enforced.*
- Dev. Weight: this weight represents the data collector's share of *Devices* when assignment takes place. The example below illustrates distribution across 3 collectors.

Collector Name	Device Distribution Weight	Assignment share percent
A	100	33.33
B	50	16.66
C	150	50.00
- Cap.: the number of capabilities supported by the collector.
For *SNMP Collectors*, it is effectively the number of snmp-masks available to the collector.

Functionality

The *Collectors* provides the following functionality:

- Distribute Devices: assigns the *Device* to all enabled collectors.
Assignment takes place in accordance to a few distribution rules:

1. The *Device* will be assigned to a collector which is active (enabled).
 2. The *Device* will be assigned to a specific collector, if specified.
 3. The *Device* will be assigned a collector part of a Collector Group, if one is specified. This means that the *Device* may end up being by any collector which is part of this group.
 4. The *Device* will be assigned to a collector which supports all of its capabilities.
- Active state: enable or disable a *Collector*.
 - Sent Generated Configuration: apply the generated *Device* distribution to the selected collectors.
 - Clear Generated Distributions: delete the generated distributions in the *SNMP Device Discovery*. *This does not affect the collector.*
 - View generated configuration: click on the 'script' icon to view the current device distribution.
 - Download: click the 'download' icon to download of the generated configuration for this collector if it exists in the *SNMP Device Discovery*.
 - Edit: click on the 'Edit' button to display the collector edit dialog.
 - Delete: delete the collector (a confirmation pop-up will appear preventing accidental deletion).
 - New Collector: add a new collector to the *SNMP Device Discovery* from a running and reachable collector.
 - Manage Default Masks: open the management page for snmp-masks and translations used during the Capability Discovery process. These snmp-masks and translations applies for SNMP collectors whose configuration is not locked. See section [7](#) for details.

7 Default Masks

The *Default Masks* page displays information about the default snmp-masks along with controls to add or remove them. Default snmp-masks and translations are used during the Capability Discovery process for SNMP collectors whose configuration is not locked.

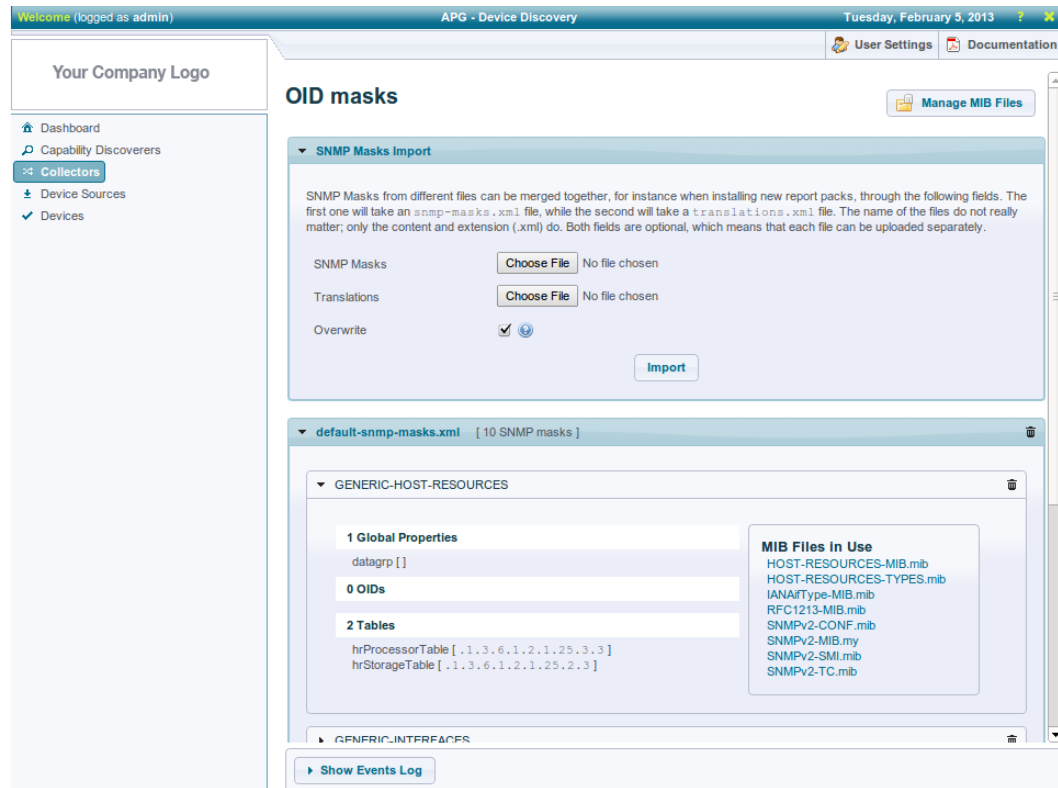


Figure 7: Default Masks

Information

The page lists all the snmp-masks files managed by the *SNMP Device Discovery*. Inside each file, the list of snmp-masks is displayed along with various information about the mask.

WARNING: Please make sure that the same snmp-mask does not appear in several files. In this case, only one definition of the mask will be used during the Capability Discovery.

Functionality

- **Manage MIB Files:** open the page containing the listing of MIB files used by the snmp-masks. See section [8](#) for details.
- **SNMP Masks Import:** the top section of the page allows the import of new snmp-masks and translations into the *SNMP Device Discovery*. The *Overwrite* setting controls the behavior when the imported file has the same name as an already existing one.
- **Mask Delete:** by clicking on a little trash of an snmp-mask section, a snmp-mask can be removed from the files that contains it.
- **File Delete:** by clicking on a little trash can of a file section, a complete snmp-mask file can be deleted.

8 Default MIBs

The *Default MIBs* page displays the list of MIB files managed by the *SNMP Device Discovery* and used by the Default snmp-masks.

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
RFC1229-MIB	21.1 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1231-MIB	38.6 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1232-MIB	35.6 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1233-MIB		9:54:14 AM	✓
RFC1253-MIB		9:54:14 AM	✓
RFC1253-MIB.mib		9:54:14 AM	✓
RFC1271-MIB	144.2 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1289-MIB	101.6 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1304-MIB	36.8 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1315-MIB.mib	21.3 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1315-MIB	20.7 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1316-MIB	16 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1317-MIB	18.3 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1318-MIB	8.7 KB	Jan 30, 2013 9:54:14 AM	✓
RFC1381-MIB	33.2 KB	Jan 30, 2013 9:54:14 AM	✓

There are 337 MIB files loaded out of 337.

[Upload a MIB file](#) [Reload MIBs](#)

Figure 8: Default MIB Files

Functionality

- Download MIB file: download the content of the MIB file.
- Delete MIB file: definitively remove the MIB file from the Device-Discovery.
- Upload MIB file: add a new MIB file on the Device-Discovery.
- Reload MIBs: Update the MIB files, in case some where modified by an other user or application.

9 Event Log

The *Event Log* is available at the bottom of all pages.

It contains information about the last 100 *SNMP Device Discovery* events which occurred since start up or since it was cleared by the user.

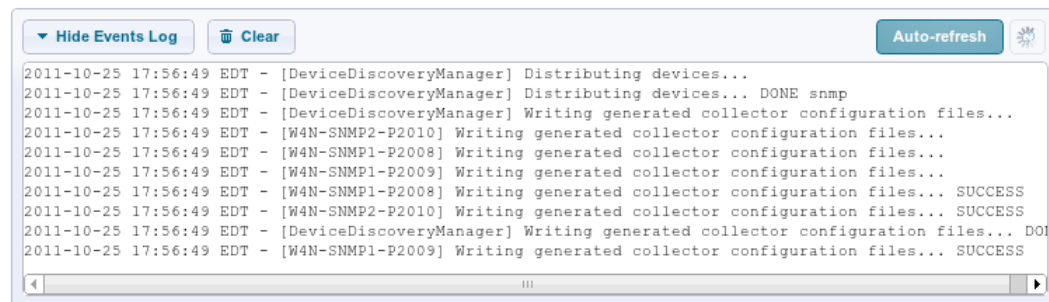


Figure 9: Event Log

Information

The information provided in these logs include:

- *Device* import devices operation success, failure and completion.
- *Device* capability discovery operation success, failure and completion.
- *Device* distribution success and failure.

Functionality

Logs may be cleared at anytime.

Logs can be automatically refreshed or the manually refreshed by the user.

5

Device Source Configuration Screens

There are currently three available *Device Source*:

- Seed File Source: this device source imports *Devices* by reading formatted seed files. The format of the file is provided in Appendix [A](#). Each line in this file defines a *Device*. All of the definition information, except the capabilities, must be provided for every *Device*.
- SNMP Discovery Source: this device source scans a network, searching for responding SNMP agents and importing them as *Devices*.
- SMARTS Source: this device source imports *Devices* by connecting to a SMARTS broker or a specific domain, retrieving its SNMP agents and importing them as *Devices*.

Functionality

General settings for all *Device Sources* are the `Import` priority and the `New Device`, `Updated Device` and `Deleted Device` action policies.

The Action Policies have different meanings, depending upon the context:

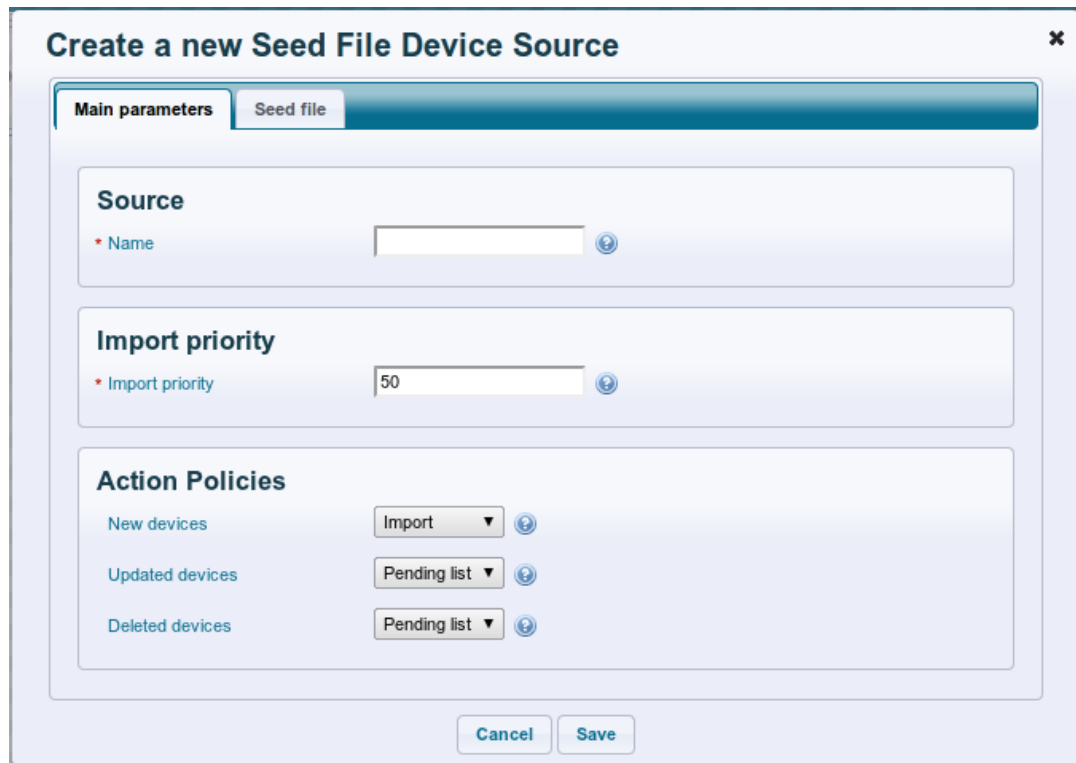
- New Device:

-
- `Import` means accept the new *Device* as an approved device.
 - `Send to pending list` means accept the new *Device* but send it to the pending list for later manual approval.
 - `Ignore` simply means to ignore the new *Device*: do not import it.
 - Updated Device:
 - `Update` means accept the new *Device* as an approved device, if the *Import Priority* check passes, and delete the existing *Device*.
 - `Send to pending list` means accept the new *Device*, if the *Import Priority* check passes, but send it to the pending list for later manual approval, and delete the existing *Device*.
 - `Ignore` means to ignore the new *Device*: do not import it and do not erase the existing *Device*.
 - Deleted Device:
 - `Delete` means delete an existing *Device*, if the *Import Priority* check passes.
 - `Send to pending list` means send the existing *Device* to the pending list, if the *Import Priority* check passes.
 - `Ignore` means ignore all delete *Device* requests from that Device Source.

Note: An *Import Priority* check is performed when updating or deleting a *Device* already present in the system. It is successful only if the new *Device*'s *Import Priority* value is less or equal to the existing *Device* *Import Priority* value. This priority check always fails if the existing *Device* is *Excluded*. The highest import priority is -1. Priorities *higher* than this (-2, -3, etc.) are recorded as -1.

1 Seed File Source

A seed file source is used to import *Devices* directly from a seed file. See Appendix [A](#) for further information about this file format.



The screenshot shows a configuration window titled "Create a new Seed File Device Source". It has two tabs: "Main parameters" and "Seed file", with "Seed file" being the active tab. The window is divided into three main sections: "Source", "Import priority", and "Action Policies".

- Source:** Contains a required field "Name" with a text input box and a help icon.
- Import priority:** Contains a required field "Import priority" with a text input box containing the value "50" and a help icon.
- Action Policies:** Contains three rows of settings:
 - "New devices" with a dropdown menu set to "Import" and a help icon.
 - "Updated devices" with a dropdown menu set to "Pending list" and a help icon.
 - "Deleted devices" with a dropdown menu set to "Pending list" and a help icon.

At the bottom right of the window are "Cancel" and "Save" buttons.

Figure 10: Seed File Source

Functionality

A seed file may be chosen or uploaded through the *Seed file* tab. The *Devices* defined in this seed file will be imported by the *Seed File Source*.

2 SNMP Discovery Source

This *Device Source* scans a network, searching for responding SNMP agents and importing them as *Devices*. It can also import *Devices* from a seed file (See Appendix A) and further discover their usable SNMP communication parameters (only v1 and v2c discovery are supported) as well as their device type, provided by their SNMP Agent.

Create a new SNMP Device Source [X]

Main parameters | SNMP: IP Address | SNMP: Discovery | SNMP: Credentials | SNMP: Collectors

Source

* Name ?

Import priority

* Import priority ?

Action Policies

New devices ?

Updated devices ?

Deleted devices ?

Figure 11: SNMP Discovery Source: common configuration

New SNMP Device Source

Main parameters | SNMP: IP Address | **SNMP: Discovery** | SNMP: Credentials | SNMP: Collectors

SNMP Device Discovery configuration

- * Threads: 10
- * Addresses per thread: 30
- * Device name provider: ☒ DNS ☐ SNMP Agent ☐ IP Address
- * Process request timeout (ms): 3000
- * Process request retries: 2
- * Agent polling request timeout (ms): 3000
- * Agent polling request retries: 2

Cancel Save

Figure 12: SNMP Discovery Source: discovery configuration

Functionality

The following functionality is provided by the *SNMP Discovery Source*:

- Import *Devices* from a seed file.
- Scan a network for responding SNMP agents by providing glob-based IP selection rules (simplified regular expressions).
- Multiple SNMP credentials are supported (only SNMP v1 and v2c).

- The imported *Devices* may be bound to *Collector Groups* or *Collectors*.
- The discovery requests timeout and number of retries is configurable.
- The *Device's* polling timeout and number of retries is configurable.
- The number of threads (concurrent connections to a SNMP Collector) is configurable.
- The number of IP addresses to scan per thread (connection to a SNMP Collector) is configurable.

3 SMARTS Source

This *Device Source* connects to a SMARTS broker or domain and retrieve its SNMP agents.

New Smarts Device Source

Main parameters | **Smarts: Domains** | Smarts: Capability exclusions | SNMP: Collectors

Smarts configuration

Connection timeout: 60

Use Smarts Broker: ☒

Smarts Broker

Host: 10.128.10.31

Port: 426

Broker user:

Broker password:

Domain Manager user:

Domain Manager password:

All domains: ☒

Cancel Save

Figure 13: SMARTS Source: using all domains from broker

Functionality

The following functionality is provided by the *SMARTS Source*:

- Query a SMARTS broker and retrieve *Devices* from all domains or specific ones.
- Query one or many domains explicitly and retrieve *Devices* from them.
- Assign capabilities which must not be discovered to retrieved *Devices*.
- The imported *Devices* may be bound to *Collector Groups* or *Collectors*.

New Smarts Device Source

Main parameters | **Smarts: Domains** | Smarts: Capability exclusions | SNMP: Collectors

Smarts configuration

Connection timeout: 60

Use Smarts Broker: ☒

Smarts Broker

Host: 10.128.10.31

Port: 426

Broker user:

Broker password:

Domain Manager user:

Domain Manager password:

All domains: ☐

Domains list:

- ☐ IC-ACM
- ☐ INCHARGE-MBIM
- ☐ INCHARGE-OI
- ☐ INCHARGE-SA
- ☐ INCHARGE-AM-PM
- ☐ INCHARGE-ESM
- ☐ BIGTOPO-AM-PM
- ☐ INCHARGE-AM-PM-DYN
- ☐ TRAP-INCHARGE-OI

Show domains

Cancel Save

Figure 14: SMARTS Source: not using all domains from broker

New Smarts Device Source

Main parameters

Smarts: Domains

Smarts: Capability exclusions

SNMP: Collectors

Smarts configuration

Connection timeout60

Use Smarts Broker☐

Smarts Domains

New domain

Name

Host

Port

User

Password

Cancel

Save

Figure 15: SMARTS Source: using specific domains

6

Device edit

Use the *Device* edit dialog to edit *Device* characteristics. The editable fields may differ depending upon the *Device* (only SNMP *Device* are available now):

Main parameters tab:

- Name: the name of the *Device*.
- Device Type: the device type (this is a free form field: any value is allowed here). Typical values include: `Node`(default), `Host`, `Router`, `Switch`.
- Host: the IP address or the host name of the *Device*.
- Timeout: the SNMP query timeout, in milliseconds.
- Number of retries: the number of retries in case of time out.
- Import Priority.
- Polling period: the *Device* polling period.

SNMP credentials tab:

- SNMP version: the version of the SNMP protocol to use in order to query this *Device* (SNMP v1, v2c and v3 are supported).

- SNMP Port: the port on which the SNMP Agent running on the *Device* can be accessed.
- SNMP v1 or v2c community name (if version is v1 or v2c)
- SNMP v3 user name (if version is v3)
- Authentication (v3) password.
- Authentication (v3) protocol.
- Encryption (v3) password.
- Encryption (v3) protocol.
- Context name (v3): Contexts are used to identify subsets of objects, within the scope of an engine, a contextName refers to a particular subset within an engine.
- Include context name in variables.
- V3 context engine id: the snmpEngineId
- Include context engine id in variables.

Note: RFC5343 states that: An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts (RFC3411). A context is identified by the snmpEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName).

Polling options tab:

- Varbinds per SNMP getBulk: the number of varbinds requested in getBulk requests. Default is 50 (good for many cases). Decrease this value if device times out due to large requests.
- Do not use getBulk requests: check this if a v2c / v3 device cannot handle getBulk requests correctly (typically times out).
- Line table polling (SNMP v1 only).

Collectors tab:

- Assign device to a given collector.
- Assign device to a given collector group.

Capabilities tab:

- Polls a capability on the device.

- Sets the polling period for a capability on the device in seconds.
- Disable a polled capability on a device.
- Exclude a capability from a device (do not perform this capability discovery on this device).

Edit the Device [X]

Main parameters | SNMP Credentials | Polling options | Collectors | Capabilities

* Name: W4N-FW-15 ⓘ

* Device Type: Firewall ⓘ

* Host (IP Address or host name): 192.168.0.1 ⓘ

* Timeout: 1000 ⓘ

* Number of retries: 2 ⓘ

* Import Priority: 50 ⓘ

Polling period: 0 ⓘ

Cancel Save

Figure 16: Device edit dialog

Collectors and *Collector Groups* can be assigned to a *Device* in the *Collectors* tab.

Note: Adding or removing a *Collector* (or a group of *Collector*) to a *Device* changes its unique identifier. If a *Device* already has the same unique identifier, it will be overwritten.

For SNMP *Devices*, index fields normally include: Name and SNMP Port.

Capabilities may also be assigned or excluded to a *Device* in the *Capabilities* tab.

Note: If a *Capability* is assigned (and enabled) to a *Device* but no *Collector* supporting this *Capability* is available, the *Device* will not be assigned. You can either make sure that a *Collector* supporting this *Capability* is available or disable the unsupported *Capability* on the *Device*.

7

SNMP *Capability Discoverer* edit

The SNMP *Capability Discoverer* edit dialog may be accessed either by creating a SNMP *Capability Discoverer*, via the *Capability Discoverers* page, or by clicking on the 'edit' button of the SNMP *Capability Discoverer*.

A suitable SNMP *Capability Discoverer* is provided by default.

The number of threads used for *Capability* discovery may be changed to suit large network requirements.

Functionality

The SNMP *Capability Discoverer* is used to discover Capabilities offered by SNMP agents.

A *Capability* is polled by the SNMP Collector using a specific *snmp-mask*.

The SNMP *Capability Discoverer* will execute a series of queries on an SNMP Agent in order to determine if the SNMP Agent offers the *Capability*.

The *Capability* is deemed to be supported when *ALL* queries are successful.

The SNMP *Capability Discoverer* will try to find every *Capability* that it can detect on any given SNMP Agent, unless the *Device* configuration explicitly prohibits discovery (The *Capability* is listed in the *Device* 'Do not discover *Capability*' list).

Edit SNMP Capability Discoverer

Base configuration

- Name: SNMP Capability Discoverer
- Number of threads: 20
- Device type mapping file: old2devtypes

Discovery Actions file directories

▼ snmp-capability-discovery-actions

Directory: snmp-capability-discovery-actions

Include sub-directories: No

Cancel Save

Figure 17: SNMP *Capability Discoverer* Edit Screen

The following parameters can be set on the SNMP *Capability Discoverer*:

- Number of threads: this is the number of concurrent connections to SNMP Collectors. More threads will create a higher workload for the available SNMP Collector but will speed up the discovery process.
- Number of devices in a request: this is the number of *Device* to process per connection to a SNMP Collector. A high number of devices means that the request will take longer to process.

The *Discovery Actions* tab shows the default polling period and load factor for each capability defined.

Note: Capability discovery should be done as a rather low priority task when using production SNMP Collectors. Having a higher number of threads (more than 2 per collector) will mean quicker discovery but will also impact the polling process. Setting a high number of devices per requests (over 100) usually means that the *Device* list will not be updated very frequently but also implies that connections will be used efficiently. The default values may be used on a production system but should be adjusted according to production requirements.

SNMP Collector Edit

The *SNMP Collector* edit dialog can be accessed either by adding an *SNMP SNMP Collector*, via the *Collectors* page, or by editing an existing one.

Adding a *Collector* from running *SNMP Collector*

Click the *Add from active Collector* button in the *Collectors* page.

This action will display the *Retrieve configuration from a running SNMP Collector* dialog, which contains network communication parameter fields as well as option check boxes used to specify further actions to perform with the retrieved *SNMP Collector* configuration files.

The following steps are required in order to add the *SNMP Collector*:

- Choose the Device-Discovery server network interface to use in order to reach the running *SNMP Collector* (default: *Auto-Select*).
- Specify IP address and port which can be used to reach a running *SNMP Collector* (default: 127.0.0.1 and 2008).

Possible errors while trying to add a running *SNMP Collector* are:

- a *Collector* with the same name already exist.

Retrieve configuration from a running SNMP Collector

✕

▼ **SNMP Collector information**

Network interface: Auto Select ⓘ

* Collector IP Address: 127.0.0.1 ⓘ

* Collector Port: 2008 ⓘ

▼ **Import options**

Extract Devices: ☐ ⓘ

Bind extracted Devices: ☐ ⓘ

Assign Capabilities to extracted Devices: ☐ ⓘ

Lock collector current capabilities: ☐ ⓘ

Cancel Save

Figure 18: Retrieve configuration from a running *SNMP Collector*

- the *SNMP Collector* is not reachable with the specified IP address and port from the *SNMP Device Discovery* host (may be related to Access control lists or firewall rules).

Other options are provided when retrieving a running *SNMP Collector*'s:

- *Extract Devices*: enabling this option will create a *Seed File Device Source* which contains the *Devices* currently assigned to the running *SNMP Collector*. The *Devices* will not, however, be bound to the *SNMP Collector* when *Device* distribution takes place, unless the *Bind extracted Devices* is checked.

- *Bind extracted Devices*: bind the extracted *Devices* the *SNMP Collector* when *Device* distribution takes place.
- *Assign Capabilities to extracted Devices*: enabling this option will assign the correct capabilities to the extracted *Devices*. Since a *Capability* is a snmp-mask, the retrieved polling groups definitions are used to assign the correct Capabilities to a specific *Device*.
- *Lock collector current capabilities*: When selected, the collectors snmp-masks and translations are not updated even when a modified configuration is pushed back. Otherwise the collectors capabilities are overridden by the *SNMP Device Discovery* ones.

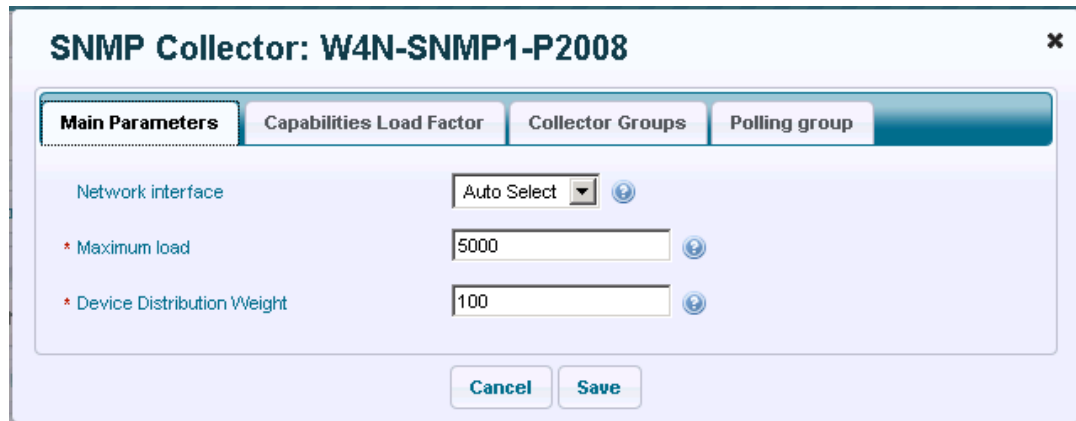
Once the running *SNMP Collector* configuration has been successfully retrieved, the *Collector* is created.

Note: Once set, the *SNMP Collector* configuration files may not be changed on the device source.

Editing an *Collector*

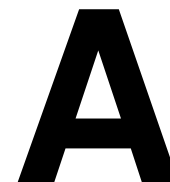
Click the *Edit* button of an existing *Collector* in the *Collectors* page.

This action will display the *SNMP Collector* edit dialog which contains settings about the collector, that capabilities and the polling group.



The image shows a dialog box titled "SNMP Collector: W4N-SNMP1-P2008". It has four tabs: "Main Parameters", "Capabilities Load Factor", "Collector Groups", and "Polling group". The "Main Parameters" tab is selected. It contains three settings: "Network interface" with a dropdown menu set to "Auto Select", "Maximum load" with a text box containing "5000", and "Device Distribution Weight" with a text box containing "100". Each setting has a help icon (a blue circle with a question mark) to its right. At the bottom of the dialog are "Cancel" and "Save" buttons.

Figure 19: SNMP Collector edit dilaog



SeedFileSource Input File Format

The Seed File Source is meant to read a file which contains one *Device* definition per line. The *Device* attribute fields are separated by a TAB character. The TAB character is invalid as the value itself. An attribute field contains the attribute name, followed by the “=” character, and a value. A *Device* definition must contain at least one required field for indexing purposes, while other fields are optional.

A minimal input file, based on the provided SNMP *Device* definition (see Appendix B), contains the following information:

```
name=name1 snmpPort=161 devtype=Router ipAddress=192.168.0.1 capability=ips1a
snmpCommunityOrUserV3Name=commString1
name=name5 snmpPort=161 devtype=Server ipAddress=192.168.0.5 snmpPort=161 capability=server
snmpCommunityOrUserV3Name=commString1
name=name2 snmpPort=161 devtype=Router ipAddress=192.168.0.2 collectorGroupName=net1
snmpCommunityOrUserV3Name=commString1
name=name6 snmpPort=161 devtype=Router ipAddress=192.168.0.6 collectorGroupName=net1
collectorGroupName=net2 snmpCommunityOrUserV3Name=commString1
name=name7 snmpPort=161 devtype=Node ipAddress=192.168.0.7 collectorName=W4nSnmp-1
snmpCommunityOrUserV3Name=commString1
```

SNMP *Device* attribute field descriptions The following are the attribute fields which may be used in an SNMP seed file.

Note: All fields are optional, except the `name` and `snmpPort` fields, which are part of the *Device* id.

- *name*: The name of the *Device*. This field is mandatory as it is an index field for the *Device*.
- *devtype*: The *Device*'s type. Common values include: "Host", "Node", "Router", "Switch". Any value may be set in this field. Use "Node" to indicate that the *Device* type is unknown.
- *doNotUseGetBulkRequests*: Specifies to use only snmp requests for polling SNMP v2c and v3 agents when set to true. Valid values: "true", "false".
- *ipAddress*: The *Device*'s IP Address. This may be an IPv4 or a fully defined IPv6 address. This field is *mandatory* when using the seed file with a SeedFileSourceAdapter.
- *onFailTryWith*: When polling agents, use the specified version specific requests for supplementary retry. Valid values are: "v1" (get request), "v2" (getBulk request), "doNotRetry" (do not perform supplementary retry).
- *transportProtocol*: The transport protocol used to communicate with the SNMP agent residing on the *Device*. This defaults to udp when not provided, although tcp may also be used.
- *snmpPort*: The port used to reach the SNMP agent residing on the *Device*. This value defaults to 161 when not specified. It is an index field for the *Device*.
- *timeout*: This is the SNMP request maximum time (in milliseconds) out when querying the SNMP agent residing on the *Device*. The default value is 1000.
- *retries*: This is the number of SNMP request retries to send to the SNMP agent if it does not answer within the allowed time out value. The default value is 2.
- *snmpVersion*: The SNMP version to use when communicating with the SNMP agent residing on the *Device*. Allowed values include: v1, v2c and v3. The default value is v2c.
- *snmpCommunityOrUserV3Name*: The SNMP community string (or SNMP v1 and v2c) or user name (SNMP v3). This is required.
- *snmpV3AuthenticationPassword*: The SNMP v3 authentication password. This is required when using SNMP v3.
- *snmpV3AuthenticationProtocol*: The SNMP v3 authentication protocol. Allowed values include: md5, sha. This is optional (noAuthNoPriv if not provided) when using SNMP v3.

- *snmpV3PrivacyPassword*: The SNMP v3 privacy (encryption) password. This is optional when using SNMP v3.
- *snmpV3PrivacyProtocol*: The SNMP v3 privacy (encryption) protocol. Allowed values include: aes, des. This is optional when using SNMP v3.
- *snmpV3ContextName*: The SNMP v3 context name. Default empty context name is used if not provided. Optional.
- *useV3ContextNameInVariable*: Set the SNMP v3 context name in the variable. This will create variables linked to the context name.
- *snmpV3ContextEngineId*: The SNMP v3 context engine id. Optional. Default to empty if not provided.
- *useV3ContextEngineIdInVariable*: Set the SNMP v3 context engine id in the variable. This will create variables linked to the context engine id.
- *delete* The delete attribute. Setting this attribute to true will cause the SeedFileSourceAdapter to delete this *Device* if the import priority is greater or equal than the existing *Device* and that the source action policy is set to DO_ACTION. It may also cause the *Device* to be sent to the pending list, depending on the source action policy.

collectorName and collectorGroupName fields A *Device* may contain one or more collector-Name values. The collectorName values are used when distributing the *Devices* among the collectors. A *Device* with a specified collectorName attribute can only be polled by a collector with the same name. Having several collectorNames means that the *Device* may be polled by anyone of the specified collectors. The same behavior applies to the collectorGroupName attribute: a *Device* assigned to a collector group may only be polled by a Collector that is part of this collector group.

pollingPeriod A *Device* may have its own polling period, which overrides its collector default period. This polling period must be set in seconds. Supported values include positive integers. Zero or negative values are ignored. The pollingPeriod field is used to set the *Device* polling period.

capability fields A *Device* may have one or more capability fields. A capability field consists of a capability name followed by an optional polling period string. The polling period string contains a polling period separator followed by a polling period which should be a non-negative integer. The polling period separator string can be user specified using the system property *devicediscovery.pollingperiodseparator*. A default value of "@" is used if it is not specified. Invalid or negative

values of polling period are reset to zero. The default capability (specified in the *Collector* configuration) is used when none are provided. The capability is used to determine whether a collector can poll a *Device* or not and for distribution purposes.

Note: In the case of the SNMP *Collector*, a capability corresponds to a valid *snmp-mask* defined in the collector's `snmp-masks.xml` configuration file.

doNotDiscoverCapability fields Some *Device* may not support specific capabilities, to the point that even trying to discover them may have adverse effects on the device behavior. Those capabilities which must not be searched for can be defined by adding one (or more) `doNotDiscoverCapability` value to the device definition. The `doNotDiscoverCapability` values may also be used when the provisioning system used to generate the seedfile can provide this information.

Note: All attribute fields, except `collectorName`, `collectorGroupName` and `capability` fields should be provided.



SNMP *Device* Definition

The *SNMP Device Discovery* provides the SNMP Agent *Device* definition as a valid type for import and export.

This definition is specified in this file:

<APG>/Custom/WebApps-Resources/<Instance>/device-discovery/target-device-defs.xml

The content of this file is listed below and is self-documented:

```
<?xml version="1.0" encoding="UTF-8"?>
<target-device-defs>
  <target-device-type type="snmp" collectorDomainInterface="NetworkCollectorDomain">
    <possible-properties>
      <property name="name" isIndex="true" valueType="string"/>
      <property name="devtype" value="Node" valueType="string"/>
      <property name="ipAddress" valueType="string"/>
      <property name="transportProtocol" value="udp" valueType="string"/>
      <property name="snmpPort" value="161" isIndex="true" valueType="int+"/>
      <property name="timeout" value="1000" valueType="int+"/>
      <property name="retries" value="2" valueType="int0+"/>
      <property name="snmpVersion" value="v2c" valueType="string"/>
      <property name="snmpCommunityOrUserV3Name" valueType="string0" encrypted="true"/>
    </possible-properties>
  </target-device-type>
</target-device-defs>
```

```
<property name="snmpV3AuthenticationPassword" valueType="string0" encrypted="true"/>
<property name="snmpV3AuthenticationProtocol" valueType="string0"/>
<property name="snmpV3PrivacyPassword" valueType="string0" encrypted="true"/>
<property name="snmpV3PrivacyProtocol" valueType="string0"/>
<property name="v1TableRowPolling" value="false" valueType="string"/>
<property name="onFailTryWith" value="v1" valueType="string"/>
<property name="doNotUseGetBulkRequests" value="false" valueType="string"/>
<property name="maxBulkSize" value="50" valueType="int+"/>
<property name="snmpV3ContextName" valueType="string0"/>
<property name="useV3ContextNameInVariable" value="false" valueType="string"/>
<property name="snmpV3ContextEngineId" valueType="string0"/>
<property name="useV3ContextEngineIdInVariable" value="false" valueType="string"/>
<possible-properties>
<target-device-type>
<target-device-defs>
```

Note: The devtype attribute can be set to *Node* when unknown. In this case, the *SNMP Capability Discoverer* module will poll the SNMP Agent and match its sysObjectId with the correct devtype, if found in the oid2devtypes file. This oid2devtype file can be edited to add missing entries. Simply save the *Capability Discoverer* after the oid2devtypes has been edited. The original oid2devtypes file packaged with this distribution is always available is <APG>/Custom/WebApps-Resources/<Instance>/device-discovery/current-version-standard-config directory.



***Device* Distribution Logic**

This section describes the algorithms used to perform the distribution of *Devices* among the configured collectors.

The following tests are performed to determine whether or not a *Device* can be polled by a given collector. Tests are performed in sequence; all tests must pass to confirm that a given *Device* may be polled by the collector.

- The *Device* type must match the collector type.
- The collector name must match one of the *Device* collectorName attributes, if some are provided. This condition is always satisfied when the *Device* does not have a collectorName attribute.
- The collector group name must match one of the *Device* collector group names, if some are provided. This condition is always satisfied when the *Device* does not have a collector group name attribute.
- The collector must support every one of the specified *Device* capabilities.

Device distribution can then take place once all tests have been performed for every *Device* on every collector. The *Device* distribution ratio is based on the collector distribution factor (snmp-agent-distribution-factor for the SnmpCollector). This factor is applied to a group of possible collectors for a group of *Devices* which may be polled by them.

As an example, let's suppose that 500 *Devices* were imported and that 4 collectors are available.

Available collectors:

- Collector A: distribution-factor = 10, capabilities: interface*, ipsla, qos
- Collector B: distribution-factor = 20, capabilities: interface*, qos
- Collector C: distribution-factor = 30, capabilities: interface*, ipsla, qos
- Collector D: distribution-factor = 40, capabilities: interface*, ipsla

Imported *Devices*:

- Group 1: 100 *Devices*, capabilities: qos
- Group 2: 100 *Devices*, no capability, collectorName = Collector A, collectorName = Collector D
- Group 3: 100 *Devices*, capabilities: ipsla, qos
- Group 4: 100 *Devices*, capabilities: ipsla
- Group 5: 100 *Devices*, capabilities: ipsla, collectorName = Collector B

The distribution algorithm result should be similar to this:

- Collector A: $17(1) + 20(2) + 25(3) + 13(4) = 75$
- Collector B: $33(1)$
- Collector C: $50(1) + 75(3) + 37(4) = 162$
- Collector D: $80(2) + 50(4) = 130$

The 100 remaining *Devices* could not be distributed because they did not match any of the collectors (conflicting criteria: Collector B does not support the ipsla capability).



SNMP *Capability* Discovery Actions Provided by Default

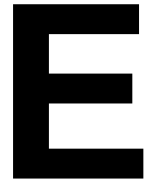
The following Discovery Actions definitions are currently provided with the *SNMP Device Discovery*:

- CISCO-MEMORY-discovery-actions.xml
- CISCO-PROCESS-discovery-actions.xml
- CISCO-PROCESS-POST12.2(3.5)-discovery-actions.xml
- FOUNDRY-PROCESS-discovery-actions.xml
- GENERIC-HOST-RESOURCES-PROCESSOR-discovery-actions.xml
- GENERIC-HOST-RESOURCES-STORAGE-discovery-actions.xml
- GENERIC-INTERFACES-discovery-actions.xml
- GENERIC-INTERFACES-NO-IFXTABLE-discovery-actions.xml
- GENERIC-INTERFACES-NON-UNIQUE-IFDESCR-discovery-actions.xml
- GENERIC-SYSUPTIME-discovery-actions.xml

- HP-PROCESS-discovery-actions.xml
- JUNIPER-MEMORY-discovery-actions.xml
- JUNIPER-PROCESS-discovery-actions.xml
- NORTEL-PROCESS-discovery-actions.xml

These Discovery Actions definitions cover the Capabilities of the provided default SNMP Masks.

You may create your own Discovery Actions (the script language specification is provided in Appendix [F](#)) or contact Watch4Net to provide you with custom Discovery Actions for your own SNMP Masks.



Controlling the GUI via a Command Line Interface

The *SNMP Device Discovery* may be controlled via a command line interface using the `device-discovery-cli` module.

Please refer to the module documentation for more details about the required setup modifications as well as how to use it.



SNMP *Capability* Discovery Actions: Language Definition

The following schema defines the XML syntax used to create new SNMP *Capability* Discovery Actions files.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.watch4net.com/
capability-discovery-action"
xmlns:tns="http://www.watch4net.com/capability-discovery-action" elementFormDefault="qualified
">

<element name="snmp-capability-discovery-actions-config">
  <complexType>
    <sequence>
      <element name="discovery-pass" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <sequence>
            <!--
              Define the capability-discovery-action-group. 1 or more are required here.
```

```

    The capability is deemed functional when all of the capability-discovery-action-group
    return positive results on a device.
    If one of the capability-discovery-action-group returns false, then fail fast:
    do not evaluate the next capability-discovery-action-group.
-->
<element name="capability-condition-group">
  <complexType>
    <choice minOccurs="1" maxOccurs="unbounded">
      <!-- Perform a snmpget on oid: returns true if the oid exists -->
      <element name="SingleOidExistsSnmpCondition" type="tns:condition-type"/>

      <!--
      Perform a snmpget on oid:
      returns true if the snmp varbind exists and its value matches the matchValue regex.
      invertMatch is used to negate the regex match result:
      return true when the varbind value does not match.
      -->
      <element name="SingleOidContainsSnmpCondition" type="tns:condition-type"/>

      <!--
      Perform a snmpgetnext at the table root oid.
      Returns true if the returned snmp varbind is part of the queried table.
      -->
      <element name="TableExistsSnmpCondition" type="tns:condition-type"/>

      <!--
      Perform a snmpgetnext on the table column oid.
      Return true if the returned varbind is part of the table column.
      -->
      <element name="TableColumnExistsSnmpCondition" type="tns:condition-type"/>

      <!--
      Perform a snmp walk on the table column oid.
      Return true if ANY of the returned varbind value matches the matchValue regex.
      If invertResult is set to true, return true if NONE of the varbind match the
      matchValue.
      -->
      <element name="TableColumnContainsSnmpCondition" type="tns:condition-type"/>
      <!--
      Perform a snmp walk on the table column oid.
      Return true if all values are unique in the column.
      -->

```

```

        <element name="TableColumnUniqueValuesSnmpCondition" type="tns:condition-type"/>
      </choice>
      <attribute name="snmpConditionBasePackage" type="string"
        fixed="com.watch4net.apg.devicediscoveryapi.capabilitycondition.snmp.impl." />
    </complexType>
  </element>
</sequence>
</complexType>
</element>
</sequence>

<!-- This is the capability name. It matches an existing snmp-mask. -->
<attribute name="capability-name" type="string" use="required" />

<!--
  This is the capability load factor.
  In other words, how much "time and effort is required to poll this capability, in terms of
  collector cpu usage or actual polling time.
  For instance, Servers are known to answer SNMP request rather slowly, therefore a high
  value, say 10 or more, might be required here.
-->
<attribute name="load-factor" type="positiveInteger" use="optional"/>
<!-- The default polling period for the capability. -->
<attribute name="default-polling-period" type="nonNegativeInteger" use="optional" default="0
" />
</complexType>
</element>

<simpleType name="oid-type">
  <restriction base="string">
    <pattern value="(\.[0-9]+)+" />
  </restriction>
</simpleType>

<complexType name="condition-type">
  <!-- The oid on which used by the action. -->
  <attribute name="oid" type="tns:oid-type" use="required" />

  <!--
    A regular expression, used for matching.
    This applies only to SingleOidContainsSnmpCondition and TableColumnContainsSnmpCondition.
  -->

```

```

<attribute name="matchValue" type="string" use="optional" default=""/>

<!-- Invert the Condition result. -->
<attribute name="invertResult" type="boolean" default="false" />
</complexType>
</schema>

```

As an example, below is the content of the GENERIC-INTERFACES SNMP *Capability* Discovery Action file, as provided in:

```

<APG>/Custom/WebApps-Resources/<Instance>/device-discovery/snmp-capability-discovery-actions/
GENERIC-INTERFACES-discovery-actions.xml

```

This discovery action is used to identify the GENERIC-INTERFACE capabilities on a queried SNMP device. It matches the GENERIC-INTERFACE snmp-mask distributed with the *SNMP Collector*. In this case, 2 passes are performed:

```

<?xml version="1.0" encoding="UTF-8"?>
<tns:snmp-capability-discovery-actions-config
  capability-name="GENERIC-INTERFACES" load-factor="5"
  xmlns:tns="http://www.watch4net.com/capability-discovery-action"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/capability-discovery-action_./././src/conf
/snmpp-capability-discovery-actions.xsd">
  <tns:discovery-pass>
    <tns:capability-condition-group>
      <tns:TableExistsSnmpCondition oid=".1.3.6.1.2.1.2.2"/>
      <tns:TableColumnUniqueValuesSnmpCondition oid=".1.3.6.1.2.1.2.2.1.2"/>
      <tns:TableExistsSnmpCondition oid=".1.3.6.1.2.1.31.1.1"/>
    </tns:capability-condition-group>
  </tns:discovery-pass>
  <tns:discovery-pass>
    <tns:capability-condition-group>
      <tns:TableColumnExistsSnmpCondition oid=".1.3.6.1.2.1.31.1.1.1.6"/>
      <tns:TableColumnExistsSnmpCondition oid=".1.3.6.1.2.1.31.1.1.1.10"/>
    </tns:capability-condition-group>
  </tns:discovery-pass>
</tns:snmp-capability-discovery-actions-config>

```

Pass 1:

1. Check that the table based at .1.3.6.1.2.1.2.2 exists (ifTable).
2. Check that the ifTable.ifDescription has unique values. All ifTable.ifDescription must be unique for this snmp-mask to work correctly.
3. Check that the table based at .1.3.6.1.2.1.31.1.1 exist (ifXTable).

If all criteria are met, go on with the second pass.

Pass 2:

1. Check that the column based at .1.3.6.1.2.1.31.1.1.1.6 exists (ifXTable.ifHCInOctets).
2. Check that the column based at .1.3.6.1.2.1.31.1.1.1.10 exists (ifXTable.ifHCOutOctets).

If all criteria are met on the last pass, then tag this capability as supported on the SNMP device.

Note: Discovery passes should always be defined to use the most discriminating factor first. For instance, if a *Capability* is supported only by devices made by a specific manufacturer, the first thing to check in the first pass should be the sysObjectId (.1.3.6.1.2.1.1.2.0) and match it against the manufacturer's IANA Private Enterprise number. Then, the second pass should look for polled tables.

The key in editing efficient SNMP *Capability* Discovery Action files is consistency: try to always test the same oids in the same pass. For instance: always perform manufacturer's IANA Private Enterprise number check on the first pass (if applicable), then go on finding the polled tables in subsequent passes.

The SNMP *Capability* Discovery Action files are located in:

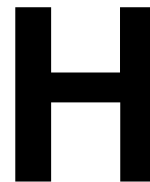
```
<APG>/Custom/WebApps-Resources/<Instance>/device-discovery/snmp-capability-discovery-actions
```

Note: The administrator may modify the SNMP *Capability* Discovery Action files as required and even delete them if they are not required. Simply keep in mind that a set of the standard SNMP *Capability* Discovery Action files is always available in the <APG>/.../device-discovery/current-version-standard-config/snmp-capability-discovery-actions directory.



Current Known Limitations

- Although multi-user support is provided, having several users edit the same component configuration can produce unexpected side-effects. Therefore, it's not recommend that multiple users access the application at the same time.
- There is currently no graphical editor for *Discovery Action* definitions.



Application Properties

1 Base configuration files directory

The *SNMP Device Discovery* configuration files are usually located in:
<APG>/Custom/WebApps-Resources/<Instance>/device-discovery.
This may be changed by specifying the `device.discovery.base.config.dir` property in the `apg-tomcat` service script in the <APG>/bin/apg.properties file.

Note: Make sure that the directory specified in the `device.discovery.base.config.dir` does not exist as the application will not deploy its base configuration files upon the first start up if the directory already exists.

You can either specify a relative path to the <APG>/Custom/WebApps-Resources/<Instance>/ directory or an absolute path.

2 Enable the distribution of devices that do not have capabilities

The default application behavior is to avoid distributing devices that have no capability to collectors. This behavior can be overridden by setting the `apg.devicediscovery.distribute.all.devices` to true in the `<APG>/bin/apg.properties` file.

Valid values for the `apg.devicediscovery.distribute.all.devices` property are:

- `apg.devicediscovery.distribute.all.devices=true` to enable the distribution of devices that have no capability to collectors, without having them polled by the collectors. In the case of SNMP Collectors, devices without capability will appear in the `snmp-polling-distribution.xml` file, but are not used in polling groups.
- `apg.devicediscovery.distribute.all.devices=false` (or not defined) is the default behavior.

3 Customize device list

While the *SNMP Device Discovery* can manage thousands of devices, you can only see 100 devices at the same time in the UI.

If you want to apply an operation to more devices, you can select all the devices matching the current table filter by selecting a check box under the table (this checkbox only appears when all visible devices are selected).

The user can select the number of displayed devices per page.

The default number of devices per page can be set using the `device.discovery.devicelist.pagelength` property (default: 10).

Possible values are: 10, 25, 50, 100.

4 Customize the event log size

Use the `device.discovery.eventlog.size` property (default: 100).

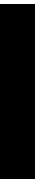
5 Configuration auto-deployment

Some configuration files are redeployed everytime upon tomcat startup. Those files are usually related to the *SNMP Device Discovery*'s version. Those files include:

- `smarts-mapped-properties.xml`

- source-adapter-device-validator.xml

In some rare cases, it may be desirable to avoid overwriting those files upon startup. The `devicediscovery.deployconfig=false` may be set to avoid this behavior.



Polling periods priority

There are several ways to set the polling periods for a *Device* managed by the Device-Discovery application. The order of priority in applying the polling period is as follows:

1. Polling period set on the *Device* for a specified Capability. This will override all other polling periods for that capability on that *Device* if set to a positive value.
2. Polling period set on the Capability (set in the SNMP *Capability* Discovery Action file). If set to a positive value, this will override the default polling period set on the *Collector* and/or the polling period set on *Device*.
3. Polling period set on the *Device* itself. If set to a positive value, this will only override the default polling period set on the *Collector*.
4. Polling period set on the *Collector*: this is the default polling period for *Devices*, if nothing else is set.

Setting different polling periods, using different priorities on a specific *Device* will affect the way it is distributed to a *Collector*.

In the case where:

- The default polling period set on the *Collector* is 300s;

- The polling period set on capability GENERIC-SYSUPTIME is 30s;
- The polling period set on capability CISCO-MEMORY is 600s;

Given the following example (seed file format is used here, for a *SNMP Device*):

```
name=Test6 devtype=Host ipAddress=192.168.0.6 snmpPort=161 timeout=1000 retries=1
snmpVersion=v2c snmpCommunityOrUserV3Name=public onFailTryWith=v1 maxBulkSize=25 pollingPeriod
=60 capability=GENERIC-INTERFACES capability=GENERIC-SYSUPTIME capability=CISCO-MEMORY@1800
```

The *Device* Test6 will be distributed to 3 different polling groups on its collector:

- APG-GENERATED-30.GENERIC-SYSUPTIME: polled every 30 seconds;
- APG-GENERATED-60.GENERIC-INTERFACES: polled every 60 seconds;
- APG-GENERATED-1800.CISCO-MEMORY: polled every 1800 seconds.