



EMC® Watch4net®

Version 6.3u2

APG Storage Filter Administration Guide

P/N: pending

Rev : 40032

EMC²

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

Published April, 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
Document Organization	iv
Where to get help	v
1 Product Overview	1
2 Architectural Overview	5
3 Prerequisites	7
4 Installation	9
5 Configuration	11
6 Post-configuration Steps	13
7 Appendix A: Matchers Implementation	15
8 Appendix B: Handlers Implementations	17
1 UnixHandler	17
9 Appendix C: Configuration Example	21
1 storage-filter.xml	21
2 matching-configuration.xml	22
3 unix.xml	22
10 Appendix D: sdparm-tool.sh script	25

Preface

Purpose

This guide describes the configuration and usage of the APG Storage Filter.

Intended Audience

This guide is meant to be read by any person who will configure and/or use the APG Storage Filter.

Document Organization

1. Product Overview	This section provides an overall description of the filter.
2. Architectural Overview	This section is an overview of the architectural structure of the filter.
4. Installation	This section describes the installation of the filter.
5. Configuration	This section describes the configuration of the filter.
6. Post-configuration Steps	This section describes what to do after configuration of the filter.
7. Appendix A: Matchers Implementation	This section describes how matchers are implemented in the filter.
8. Appendix B: Handlers Implementations	This section describes how handlers are implemented in the filter.
9. Appendix C: Configuration Example	This section is an example of a full configuration of the filter, including a reference for all its configuration files.
10. Appendix D: sdparm-tool.sh script	This section describes using the sdparm tool to gather information about Unix hosts for the APG Storage Filter configuration.

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

Product Overview

The main task of the APG Storage Filter is to integrate data from a variety of modern storage solutions (NAS, SAN, DAS, virtualizes solutions, etc.) into *APG* in order to generate storage-specific reports independently of the storage type/manufacturer/technology.

The APG Storage Filter adds valuable information to raw values in order to generate more significant *APG* reports. The integration of disparate storage data from different sources, which would have otherwise been a tedious task, is thus organized and automated in *APG*.

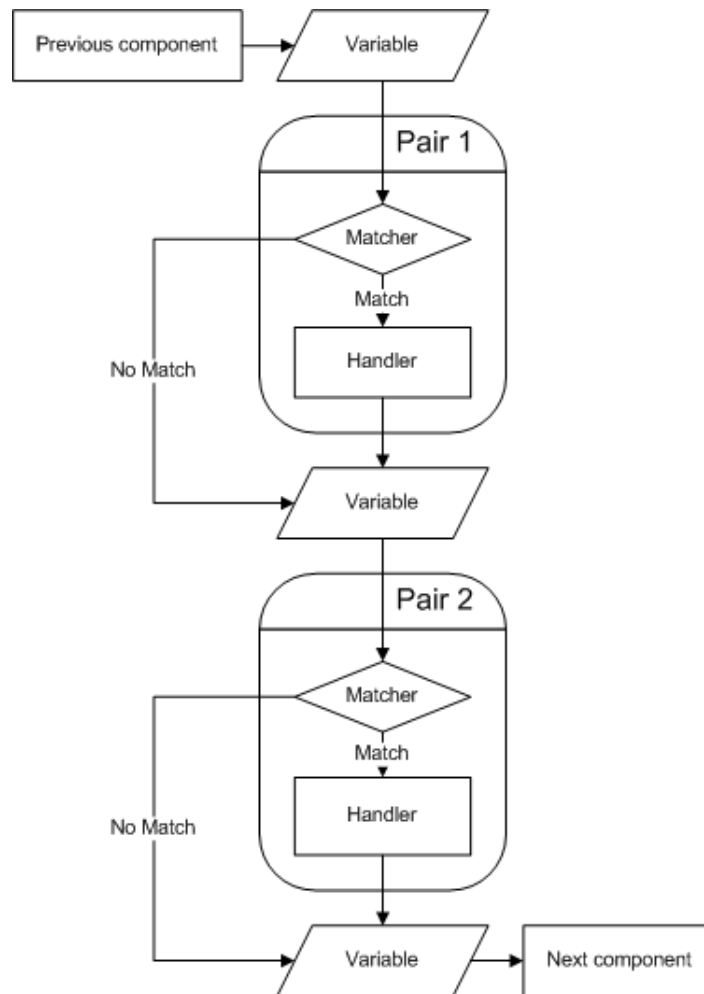
Some components of the *APG* ecosystem generate raw values related to storage, but these are sometimes incomplete. For example, consider the SNMP collector sending raw values regarding a particular host. On one hand, the raw values for the capacity and free space of each filesystem are sent, while on the other, no information about the various drives on the host is sent: how can hosts be managed without knowing which drives are present, what are their capacities, and which filesystems are on these drives? The APG Storage Filter solves these types of issues by generating the missing raw values or by appending the missing properties to the incoming raw values.

Most mainstream storage solutions are supported by the APG Storage Filter. Please contact *APG* support if you're unsure about the support of a storage solution after reading this document.

Architecturally, the APG Storage Filter is made of a series of matcher/handler pairs. Each raw value that enters the filter flows through the chain of matcher/handler pairs, in the same order that they were declared in the configuration file. For each matcher/handler pair, if the raw value meets the specific criteria defined by the matcher, it is then handled by the corresponding handler. The last handler defined is the next component reached by the filter (it is automatically configured by the filter, so there is no need to set it).

If you ever used and configured the *APG* Variable Handling Filter, the configuration of the APG Storage Filter should be familiar and so you can refer to its PDF for more information.

The APG Storage Filter is designed to only allow one raw value to flow through one handler (but obviously it can go through many matchers). The diagram on the next page represents the actual flow:



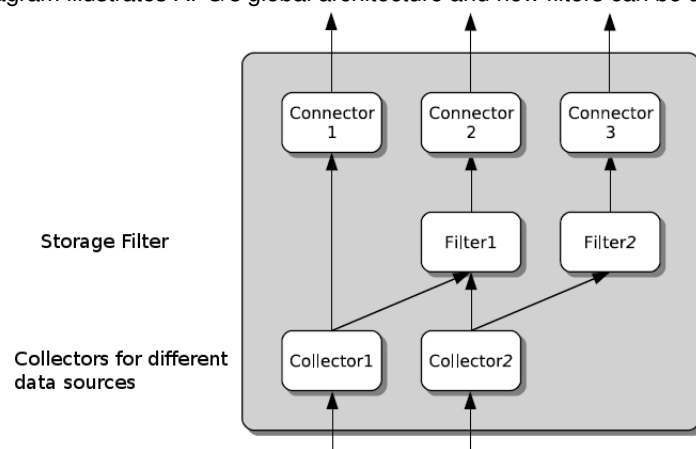
Currently, the APG Storage Filter supports a single storage type Unix hosts.

The APG Storage Filter is available since *APG 5.1*.

2

Architectural Overview

The APG Storage Filter is built to integrate seamlessly into the *APG* structure. The following diagram illustrates *APG*'s global architecture and how filters can be attached to it.



Like all the other filters in *APG*, the filter initially receives raw values, then applies the underlying logic of the filter to them, finally sending the raw values to the next component defined in the `collecting.xml` file.

Note: The APG Storage Filter can also generate some new raw values if necessary (for more information see *UnixHandler* on page [17](#)).

3

Prerequisites

The prerequisites for the APG Storage Filter depend on the type of handlers you use.

Unix Handler

The different hosts to be polled must be accessible via SSH.

4

Installation

The APG Storage Filter module ships with *APG* 5.1 but is not installed by default. You can use the Module Manager to install it. The APG Storage Filter module name is `Storage-Filter`.

To activate the APG Storage Filter, you must add the filter declaration to the `collecting.xml` configuration.

```
<filters>
  ...
  <filter enabled="true" name="StorageFilter" next="Backend" config="Storage-Filter/
Default/conf/storage-filter.xml" />
  ...
</filters>
```

Then, you can edit the filter configuration file in the *conf* folder of the `Storage-Filter` module. The default file name for this configuration file is `storage-filter.xml`.

5

Configuration

The following section outlines the APG Storage Filter configuration process.

The Storage Filter relies on a main configuration file, `storage-filter.xml`, located by default at *APG/Collecting/Storage-Filter/Default/conf/*, that points to the `matching-configuration.xml` file (by default located in the same directory). The `matching-configuration.xml` file is validated against the Variable Handling Filter XSD schema file.

The `matching-configuration.xml` file itself points to handler-specific configuration files.

For a full breakdown of the configuration steps involved in each of the different files, see [Appendix C: Configuration Example](#) on page 21.

Below is a simple example of a valid configuration for the APG Storage Filter's `matching-configuration.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.watch4net.com/VariableHandlingFilter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/VariableHandlingFilter_variable-handling-
```

```
filter.xsd">

    <handling-configuration id="UnixMatcher">
        <matcher class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.matchers.
APGFilterMatcher">
            <parameter name="filter">source=='SNMPCollector' & ((parttype=='FileSystem') &
& !(part==''))</parameter>
        </matcher>
        <handler class="com.watch4net.apg.v2.collector.plugins.storagefilter.handlers.
UnixHandler">
            <parameter name="config">./dist/conf/unix.xml</parameter>
        </handler>
    </handling-configuration>
</configuration>
```

The handlers are responsible for different operations on the raw values, according to their type. Each handler uses an internal cache mechanism to avoid multiple costly and long calls to entities that hold information. An internal mechanism forces the cache to be updated periodically: the frequency is configurable. Details about each handler are available in [Appendix B: Handlers Implementations](#).

You can use more than one handler of the same type. For example, you can use three UnixHandler for this filter each with their own configuration. In this case, each handler needs a different **id**, their own **filter** parameter argument, and also a relative path to their **config** file.

6

Post-configuration Steps

When the filter is installed and configured, *APG* is ready to handle variables as configured.

1. Restart the *APG Collector Manager* service.
2. Look at the collecting logs to make sure that everything is starting up as it should. Since the filter can rely on many configuration files, it is important to make sure that there is no errors in the logs once the filter is started.

Appendix A: Matchers Implementation

Currently, the APG Storage Filter only uses one type of matcher: the **APGFilterMatcher**, the same one that is used in the Variable Handling Filter. This matcher is available using the class:

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.matchers.APGFilterMatcher

The only mandatory parameter for this matcher is the **filter** parameter which specifies the APG filter to be used for the matching.

You do not use the other matchers defined in the Variable Handling Filter.

For more information on the matcher, see *Appendix B: Matcher Implementations* in the *APG-Variable-Handling* PDF. For more information on the syntax used to perform the matching, see *Appendix A: Filter Syntax Reference* in the *APG-Frontend-User-Guide* PDF.

8

Appendix B: Handlers Implementations

1 UnixHandler

Description

The UnixHandler handles raw values coming from Unix-based storage solutions. The necessary data is obtained using the **sdparm** tool and will be discussed later. The UnixHandler is responsible for adding new properties to incoming raw values, but will also generate some new raw values. Currently, it generates new raw values about the remaining capacity of drives on hosts and adds a property which contains all the LUN serial numbers (through the `lun-serial-number` property, which is configurable) for a particular drive. These raw values are sent every time that the internal cache is generated, which is a configurable value.

Since the number of hosts and their internal number of drives can change frequently, these parameters are not hardcoded in the configuration file. Instead, the filter relies on a tab-delimited configuration file

that defines all the hosts and their credentials to connect to them and gather the various information required by the filter. Currently, only SSH is supported. The output format will be discussed shortly. The benefit of this tab-delimited file is that it can be generated very easily by a script executed through a cron job, so the number of managed hosts and the changes made to them have no impact on the filter, and doesn't require it to be restarted. This file is used dynamically and is read every time the cache is refreshed.

By default, this tab-delimited file is called **hosts** and is located at *APG/Collecting/Storage-Filter/Default/conf*. The Unix handler configuration file points to the **hosts** file; for more information see [unix.xml](#) on page 22.

The tab-delimited file should have the following format, for every line:
Device Name<tab>**IP**<tab>**Port**<tab>**Username**<tab>**Password**<tab>**Command**

The following is an example of a valid tab-delimited file with the proper format. Please note that each host is separated by a line break:

W4N-Test-Storage0	12.34.56.100	22	root	pass100	sh /root/data.sh
W4N-Test-Storage1	12.34.56.101	22	root	pass101	sh /root/data.sh
W4N-Test-Storage2	12.34.56.102	22	root	pass102	sh /root/data.sh
W4N-Test-Storage3	12.34.56.103	22	root	pass103	sh /root/data.sh
W4N-Test-Storage4	12.34.56.104	22	root	pass104	sh /root/data.sh
W4N-Test-Storage5	12.34.56.105	22	root	pass105	sh /root/data.sh
W4N-Test-Storage6	12.34.56.106	22	root	pass106	sh /root/data.sh
W4N-Test-Storage7	12.34.56.107	22	root	pass107	sh /root/data.sh

Note: The password can be either in plain text or encrypted using the *APG* Password Encryption tool.

Note: Make sure that the text editor or that the program that will generate the file DOES NOT use soft tabs (spaces instead of the real tab character) but uses hard tabs instead.

Once the tab-delimited file is parsed, the cache is built according to the information collected in the file. For each device (host) specified in the file, the filter will remotely connect to it (using SSH) and will execute the command that was specified in the file. The command can either call a script or do a single

command line call, as long as the output generated by the execution match the expected format. The goal of this command is to gather some information about the different drives on the host. If you decide to use the default script provided (`sdparm-tool.sh`, it can be found in the same folder as all the default configuration file of the filter), the script must be placed on each host and accessible with the proper permissions for the user specified in the hosts definition file for the current host. More information is available in [Appendix D: sdparm-tool.sh script](#). The filter will read the command execution output from the standard output and will parse it. While you can use whatever tool you want to generate the output, but the **sdparm** tool is highly recommended.

The executed command on the server must return the content in the following format:

Drive<semicolon>**Capacity**<semicolon>**Lun Serial Number**<semicolon>**Filesystems**

Here's an example of a valid standard output command execution with the proper format:

```
/dev/sda;8589934592;q8Ui3013Jdq;/boot
/dev/sdb;8589934592;q8Ui3013Jdq;/DATA,/t
/dev/sdc;65802240;nBLskoQ6SIX7;
/dev/sdd;65802240;nBLskoQ6SIX7;
/dev/sde;65802240;nBLskoQ6SIX7;/usr,/tmp,/root,/boot
/dev/sdf;65802240;hg7Bqe0doc3;/usr,/tmp,/root,/boot
```

Note: If there are many filesystems attached to one drive, they have to be separated using commas as a separator, without any blank characters. If there is no filesystems related to a drive, the script must output an empty string (the line must end with the semicolon, or else it will be rejected).

Note: The capacity must be in bytes.

Java Class

`com.watch4net.apg.v2.collector.plugins.storagefilter.handlers.UnixHandler`

Parameters

- **config**: the path to the specific configuration file. The path can be relative or absolute.

Appendix C: Configuration Example

1 storage-filter.xml

The following configuration represent the default configuration file for the APG Storage Filter.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.watch4net.com/Storage-Filter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/Storage-Filter storage-filter.xsd">

  <matching-configuration>./dist/conf/matching-configuration.xml</matching-configuration>
  <property-refresh-frequency>43200</property-refresh-frequency>
</configuration>
```

- **matching-configuration:** relative or absolute path to the configuration file that defines the various matchers and handlers to be used by the filter. This parameter is mandatory.

- **property-refresh-frequency:** the APG Storage Filter implements a cache that keeps in memory the hashcode (signature) of the properties of each raw value that were sent to the next component (one cache entry per variable). This cache is useful in order to avoid forcing the backend to refresh uselessly the properties of a variable. However, it needs to be reset occasionally to avoid the cache becoming too large in memory size. The specified value is in seconds. This parameter is optional (default value is 43200 seconds (12 hours)).

2 matching-configuration.xml

The following is the default matcher/handlers configuration file.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.watch4net.com/VariableHandlingFilter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/VariableHandlingFilter_variable-handling-
filter.xsd">

  <handling-configuration id="UnixMatcher">
    <matcher class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.matchers.
APGFilterMatcher">
      <parameter name="filter">source=='SNMPCollector' & ((parttype=='FileSystem') &
& !(part==''))</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.storagefilter.handlers.
UnixHandler">
      <parameter name="config">./dist/conf/unix.xml</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

For more information on the matcher, see the *APG-Variable-Handling* PDF.

3 unix.xml

The following configuration is the default skeleton for a configuration file of a Unix Handler. The handler's configuration file is pointed to from the *matching-configuration.xml* file.

```
<?xml version="1.0" encoding="UTF-8"?>
<unix xmlns="http://www.watch4net.com/Storage-Filter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
xsi:schemaLocation="http://www.watch4net.com/Storage-Filter_unix.xsd">
  <source>StorageFilter</source>
  <group>group-Unix</group>
  <cache-update-frequency>3600</cache-update-frequency>
  <hosts-credentials-csv-file>./dist/conf/hosts</hosts-credentials-csv-file>
  <property-separator>|</property-separator>
</unix>
```

- **source**: the source to be set in the **source** property of the created raw values.
- **group**: the group to be set in the **group** property of the created raw values.
- **cache-update-frequency**: the frequency (in seconds) at which the internal cache should be updated.
- **hosts-credentials-csv-file**: the path (can be relative or absolute) to the hosts definition file.
- **property-separator**: the string separator to be used to separate elements into properties that can contains multiple elements.

Note: All these parameters are mandatory.

10

Appendix D: sdparm-tool.sh script

This script is the default Bash script provided to gather the required information about each Unix host and their drives. It is strongly recommended to use the **sdparm** tool since all the needed information is provided by the utility. However, you can customize the script or use your own one if you decide to use another tool to gather the necessary information.

Sources, .deb & .rpm files for **sdparm** are available at <http://sg.danny.cz/sg/sdparm.html>

```
#!/bin/bash

DELIM=';'
MPDELIM=', '
TMP_DISKS='/tmp/disks_output'
TMP_MOUNTS='/tmp/mounts_output'

fdisk -l | grep Disk > ${TMP_DISKS?}
while read line
do
    diskpath='echo ${line//:/} | awk '{ print $2 }''
    disksize='echo ${line//:/} | awk '{ print $5 }''
```

```

disksn='sdparm -q --page=sn $diskpath 2>/dev/null| tail -1 | awk '{ print $1}{'
mount | grep $diskpath > ${TMP_MOUNTS?}

mountpoint=""
while read line2
do
    mountpoint=${mountpoint}'echo $line2 | awk '{ print $3 }' '${MPDELIM?}
done < $TMP_MOUNTS
mountpoint=${mountpoint%${MPDELIM?}} #remove the trailing ,

if [ $? -ne 0 ]; then
    disksn=""
fi

echo -n $diskpath
echo -n ${DELIM?}
echo -n $disksize
echo -n ${DELIM?}
echo -n $disksn
echo -n ${DELIM?}
echo -n $mountpoint
echo -en "\n"
done < ${TMP_DISKS?}

rm ${TMP_DISKS?}
rm ${TMP_MOUNTS?}

```