



EMC[®] Watch4net[®]

Version 6.3u2

Variable Handling Filter Guide

P/N: pending

Rev : 45615

EMC²

Copyright ©2009 - 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
Intended Audience	iii
Prerequisites	iii
Where to get help	v
1 Product Overview	1
2 Architectural Overview	3
3 Configuration	5
1 Collecting service configuration	5
2 Filter configuration	6
4 Things to do after configuration	7
5 Appendix A	9
1 Configuration Example	9
6 Appendix B: Matcher Implementations	11
1 APG Filter Matcher	11
2 Property from File Matcher	12
7 Appendix C: Handler Implementations	15
1 BlockAllHandler	15
2 PushAllHandler	16
3 CombinedValueHandler	16
4 TimestampOrderingHandler	17
5 DecimalConverterHandler	17
6 TimestampOffsetHandler	18
7 TimestampValidationHandler	18
8 DeltaRateCalculationHandler	19

9 PropertyExtractionHandler	20
10 CharArrayToStringHandler	24
11 IntegerArrayToStringHandler	25
12 TimestampFromPropertyHandler	27
13 VlanPortIsVlansAllowedRangeExtractorHandler	32
14 PropertyArithmeticHandler	34
15 HexadecimalCounterHandler	35
16 ReplaceRegexHandler	36
17 PropertyMonitorHandler	37
18 VariableStateMonitorHandler	38
19 SendLastValueHandler	40
20 SpatialAggregationHandler	42
21 PropertyStringCaseHandler	46
22 DefinedLengthByteSequenceSplitHandler	47
23 HexaDecimalToDecimalConverterHandler	48

Preface

The purpose of this guide is to provide information about the Variable Handling Filter for APG. It contains all the necessary information to install and configure the filter.

Intended Audience

This guide is intended to be read by any *APG* administrator who will configure the Variable Handling Filter in order for variables to be processed in a particular way when they match a given filter.

Prerequisites

It is assumed that the *APG* application has been properly installed and configured on the server.

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:

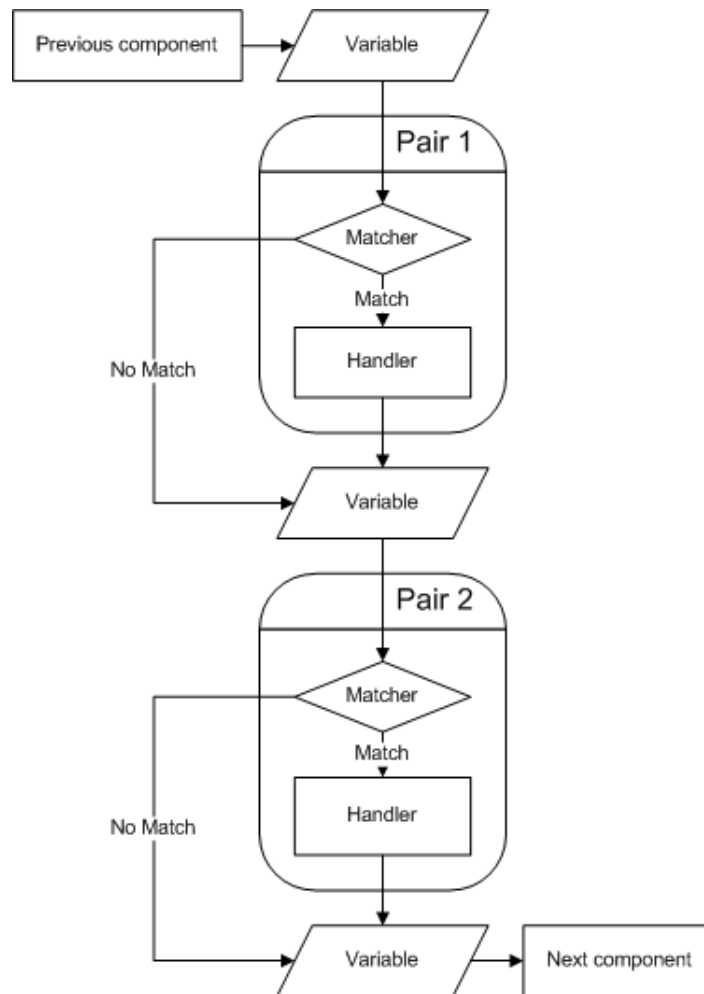
<https://support.emc.com>

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

Administrators sometimes need to modify variables before they are pushed into the *APG* backend. The processing that these variables must go through often depends on different properties, such as the source, the device type, etc. The Variable Handling Filter is designed to allow variables to be handled in a specific way depending on the criterias on which they match.

Basically, the user defines a series of Matcher/Handler pairs. Each incoming variable passes through the chain (composed of all Matcher/Handler pairs) in the order that they were declared in the configuration file. For each pair, if the variable matches on the matcher's criterias, it is handled by its corresponding handler. The variable then proceeds to the next pair, whether it was a match or not. The following diagram illustrates the process:



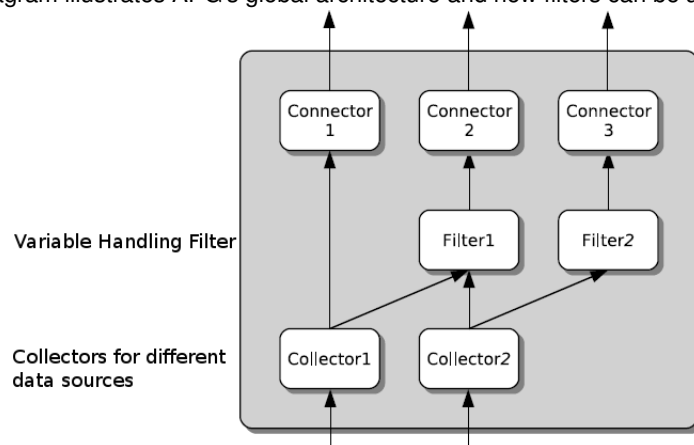
This filter is available for:

- APG 4.3 or higher

2

Architectural Overview

The Variable Handling 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.



3

Configuration

This section describes the different configuration files for the filter.

1 Collecting service configuration

The Variable Handling Filter for APG must be added in the main collecting configuration file (<APG>/Collecting/Collector-Manager/<INSTANCE>/conf/collecting.xml). Add the following section if it is not present, and make sure that the *enabled* attribute is set to *true*:

```
<filters>
  ...
  <filter name="Variable-Handling-Filter" next="Backend" config="Variable-Handling-Filter/
Default/conf/variable-handling-filter.xml" />
  ...
</filters>
```

2 Filter configuration

This file is the main configuration file for the filter. It contains information about the way variables will be matched and how they will be handled if a match occurs. It is up to the administrator to decide how to name this file and to make sure that the selected name is used to configure the filter in `collecting.xml` (see [Collecting service configuration](#) section page 5).

Global structure Here the global structure of the configuration file:

- `<configuration end-policy=end-policy>`: The main container for the collector configuration. Every other tag must be nested inside this one. The optional *end-policy* attribute represents the handler that will be used once a variable has passed through all matcher/handler pairs. The default value is the *PushAllHandler*.
 - `<handling-configuration id=id>` Defines a new handler/matcher pair. The *id* attribute is used to uniquely identify this pair.
 - * `<matcher class=class>`: Defines the matcher that will be used. The *class* attribute is optional, and it represents the java class that will be used as a matcher. If omitted, the default value is the *APGFilterMatcher*. For more information on implemented matcher classes, see [Appendix B: Matcher Implementations](#) section page 11.
 - `<parameter [type=string|file] name=name>`: Defines a parameter that will be passed to the matcher. The *type* parameter is optional and is used to specify the parameter type. A string will be used as is, while a file will resolve relative paths from the collector's working directory. The default value is string.
 - * `<handler class=class>`: Defines the handler that will be used if a match occurs. The *class* attribute is required, and it represents the java class that will be used as a handler. For more information on implemented handler classes, see [Appendix C: Handler Implementations](#) section page 15.
 - `<parameter [type=string|file] name=name>`: Defines a parameter that will be passed to the handler. The *type* parameter is optional and is used to specify the parameter type. A string will be used as is, while a file will resolve relative paths from the collector's working directory. The default value is string.
 - `<parameter [type=string|file] name=name>`: Defines a parameter that will be passed to the pair (not used very often). The *type* parameter is optional and is used to specify the parameter type. A string will be used as is, while a file will resolve relative paths from the collector's working directory. The default value is string.

4

Things to do after configuration

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

1. Start the APG collecting service.
2. Look at the collecting logs to make sure that everything is starting up as it should.

Appendix A

1 Configuration Example

The following example has 2 handling configurations.

The first one filter variables with *devtype* property equal "Array" and with *name* property containing "High". The matching variables are handled by the `BlockAllHandler` class.

The second one filter variables with *devtype* property equal "Array" and with *name* property containing "Low". The matching variables are handled by the `BlockAllHandler` class.

```
<?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="HighRemoval">
    <matcher>
      <parameter name="filter">devtype=='Array'&name='%High%'</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
BlockAllHandler" />
  </handling-configuration>
  <handling-configuration id="LowRemoval">
    <matcher>
      <parameter name="filter">devtype=='Array'&name='%Low%'</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
BlockAllHandler" />
  </handling-configuration>
</configuration>
```

6

Appendix B: Matcher Implementations

1 APG Filter Matcher

Description

Matches variable against a filter that is formatted exactly as the ones that are used in the APG user interface (i.e.: `device=='W4N-RTR1'`).

Filter syntax:

Predicates	
property='pattern'	Variable selection based on a property SQL pattern
property=='value'	Variable selection based on an exact property match
property	Selects only the variables that have the specified property
Operators	
&	AND Operator
	OR Operator
!	NOT Operator

Note: SQL pattern may contain '%' and '_' wildcard characters. Do not forget to escape characters that have a special XML meaning such as '&', '<', '>' and so on. You may encapsulate the filter within `<![CDATA[...]]>` to avoid having to escape every single special character.

Filter example:

```
device=='MyDevice' & ((parttype=='Interface' & (part='Gigabit%' | part='Ether%')
& name=='ifInOctets') | (!part & name=='Availability'))
```

This matches raw values that have a device property set to 'MyDevice' AND ((have a parttype set to 'Interface', have either part values starting with 'Gigabit' or 'Ether' and have name set to 'ifInOctets') OR (do not have the part property and have name set to 'Availability')).

Java class

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

Parameters

- filter (required): The APG filter that will be used for matching.

2 Property from File Matcher

Description

Matches variables for which the specified property is set to a value that is listed in a text file.

Java class

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

Parameters

- property (required): The name of the property to look for in each variable.
- file (required): The path of the file inside which accepted values are listed one per line. **This parameter must be of type=file.**
- refresh (optional): The interval (in seconds) between each time the property file is scanned for changes. Default value is 0, which means no refresh.
- invert (optional): Invert the match. If set to true, all values that are NOT present in the file will match. Values that ARE present will not match. Default value is false.

Appendix C: Handler Implementations

1 BlockAllHandler

Description

Drops the variable, making sure that it will not go any further in the collecting chain.

Java class

`com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.BlockAllHandler`

Parameters

- no parameters needed...

2 PushAllHandler

Description

Pushes the variable the the next component without applying any modification to it.

Java class

`com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.PushAllHandler`

Parameters

- no parameters needed...

3 CombinedValueHandler

Description

This handler aggregates metrics that have the same variable id and the same timestamp.

Java class

`com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.CombinedValueHandler`

Parameters

- *operation*: The aggregation operation. Possible values are: average, sum, min, max. Default value is sum.
- *tolerance*: If the timestamp difference between two metrics is smaller than this value, they will be considered as having the same timestamp. Default value is 60 seconds.
- *cached-data-sets*: The number of data sets to keep before pushing. When the number of data sets in memory exceeds this value, the oldest data set will be pushed. Default value is 2.

4 TimestampOrderingHandler

Description

This handler re-orders metrics in chronological order. It should only be necessary when dealing with a collector that cannot guarantee that order.

Java class

`com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.TimestampOrderingHandler`

Parameters

- *cached-data-sets*: The number of data sets to keep before pushing. When the number of data sets in memory exceeds this value, the oldest data set will be pushed. Default value is 2.

5 DecimalConverterHandler

Description

This handler formats the value of one or more properties into different representations. The property value can optionally be split on a given separator, and all tokens MUST be decimal values.

Java class

`com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.DecimalConverterHandler`

Parameters

- *converter*: The java class to use to convert each decimal value into another representation. Currently supported classes are:
 - `com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.decimalconverter.HexDecimalConverter`
 - `com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.decimalconverter.CharDecimalConverter`

- *property*: A comma-separated list of the properties for which values must be converted.
- *input-separator*: The separator to use to tokenize the input property value into decimal numbers. (Optional)
- *output-separator*: The separator to use join the formatted tokens together after they are converted. (Optional)
- *zero-padding*: For hexadecimal conversion only. Define width and pad with 0s. If number has more digits than specified padding, the digits will be preserved. (Optional)

6 TimestampOffsetHandler

Description

This handler subtracts the value of a metric from its timestamp, allowing the timestamp to be shifted. This is sometimes useful when calculating the beginning of something, given its end time and its duration.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.TimestampOffsetHandler

Parameters

- *multiplier*: The factor by which the value will be multiplied. This value can be an integer or a decimal value. The default value is 1.

7 TimestampValidationHandler

Description

This handler drops variables with a timestamp over the tolerance added to the current time. (`variable_timestamp > curent_time + tolerance`)

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.TimestampValidationHandler

Parameters

- *tolerance*: The tolerance in seconds to be added to the current time for comparison with the variable timestamp. This value must be an integer.

8 DeltaRateCalculationHandler**Description**

This handler computes a rate, positive rate, delta or positive delta on metrics that have different timestamps but identical variable ids.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.DeltaRateCalculationHandler

Parameters

- *operation*: The operation. Possible values are: rate, delta, ratep, deltap, ratep0, deltap0. There is no default value.

Operations definition

- *delta*: Variation of the value. (`currentValue - lastValue`)
- *deltap*: Same as delta but drops negative result values.
- *deltap0*: Same as deltap but send the `currentValue` if the result is negative, as if the `lastValue` was 0.
- *rate*: Variation of the value in function of the time. (`((currentValue - lastValue) / (lastTime - currentTime))`)
- *ratep*: Same as rate but drops negative values.
- *ratep0*: Same as ratep but set the `lastValue` to 0 if the result is negative.

9 PropertyExtractionHandler

Description

This handler extracts strings from properties to create new properties. There are three ways of extraction:

- List extraction : extracts multiple properties using a defined separator.
- Simple extraction : extracts one property using start and end delimiters.
- Regex extraction : extracts multiples properties using regular expression.

Note: Since version 1.4u1 the configuration of the property extraction handler has changed. To convert an older version configuration to the new one, just replace the `property` elements name by `list-property`.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.PropertyExtractionHandler

Parameters

- *file*: The path to the configuration file. Must use `type="file"` attribute.

Example:

```
<?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="Example">
    <matcher>
      <parameter name="filter">*</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
PropertyExtractionHandler">
      <parameter name="file" type="file">conf/property-extraction-handler.xml</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

```

    </handler>
  </handling-configuration>
</configuration>

```

Configuration

```

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/
VariableHandlingFilterPropertyExtractionHandler_./property-extraction-handler.xsd"
  xmlns="http://www.watch4net.com/VariableHandlingFilterPropertyExtractionHandler">
  <properties>

    <list-property name="inputProperty1" separator="." keep="true">
      <part name="outPutProperty1" size="1" />
      <part size="1" />
      <part name="outPutProperty2" />
      <part name="outPutProperty3" size="4" />
      <part name="outPutProperty4" size="1" />
    </list-property>

    <simple-property name="inputProperty2" keep="false">
      <before>
        <delimiter>id:</delimiter>
        <delimiter>identifier:</delimiter>
        <delimiter></delimiter>
      </before>
      <after>
        <delimiter>firstChoice</delimiter>
        <delimiter>secondChoice</delimiter>
        <delimiter>thirdChoice</delimiter>
        <delimiter></delimiter>
      </after>
      <part name="outPutProperty5" />
    </simple-property>

    <regex-property name="inputProperty3" keep="false">
      <regex>.*SQZ:([0-9]{5}).*</regex>
      <part name="outPutProperty6" value="{1}" />
      <part name="outPutProperty7" value="some_text{1}_around{1}_and_around" />
    </regex-property>
  </properties>
</config>

```

```
</regex-property>
</properties>
</config>%$
```

The `list-property` allows to extracts properties by splitting an existing property using a separator.

Note: If the number of extracted fields is bellow the number of output properties, no properties will be created.

Note: If the number of extracted fields is over the number of output properties, additional fields will be ignored.

- *property name*: The name of the property to split. (mandatory)
- *property separator*: The separator used to split the property. Can be one or more characters long. (mandatory)
- *property keep*: Set to `true` to not delete the original source property immediately after use. (optional, default is `false`)
- *part name*: The name of a new property to create (or overwrite). If none is set, the corresponding extracted value will be skipped. (optional)
- *part size*: The number of extracted values to place in the new property. If the size is more than one, the specified separator will remain between the values. If none is set, it will take what remains. Only one part per property can have unknown size. (optional)

The `simple-property` allows to extracts a property from an other by defining a start and an end delimiters.

It is possible to define more than one delimiters for the start and end.

- *property name*: The name of the property to perform extraction. (mandatory)
- *property keep*: Set to `true` to not delete the original source property immediately after use. (optional, default is `false`)
- *before delimiter*: The delimiters to search. Uses the first found from the order of the list. A delimiter with no value means the begining of the string. (at least one required)
- *after delimiter*: The delimiters to search. Uses the nearest delimiter found after the `before` delimiter. A delimiter with no value means the end of the string. (at least one required)

- *part name*: The name of a new property to create (or overwrite).

Example:

```
<simple-property name="in" keep="true">
  <before>
    <delimiter>id:</delimiter>
    <delimiter>identifier:</delimiter>
  </before>
  <after>
    <delimiter></delimiter>
    <delimiter> </delimiter>
    <delimiter></delimiter>
  </after>
  <part name="out" />
</simple-property>
```

For the input property value ... identifier:123(...) the output value will be 123.

For the input property value ... identifier:123 (... id:456 the output value will be 456 because id: is the first in the list of start delimiter.

The `regx-property` allows to extracts properties by applying an regular expression on an existing property.

Note: If a group number is over the number of extracted groups, the output property will not be created.

Note: Using a backslash (\) before the dollar sign (\$) will escape it. Double the backslash to print one.

- *property name*: The name of the property to split. (mandatory)
- *property keep*: Set to `true` to not delete the original source property immediately after use. (optional, default is `false`)
- *regex*: The name of a new property to create (or overwrite). If none is set, the corresponding extracted value will be skipped. (required)
- *part name*: The name of a new property to create (or overwrite). (required)
- *part value*: The value of a new property to create. `${<number>}` is used to insert in the value the same number extracted group from the regular expression. (required)

10 CharArrayToStringHandler

Description

This handler generates a string from an array of ASCII characters encoded as integers and separated by a specific character. The following character array types may be handled by this filter:

- "32.65.66.67.68" which translates to " ABCD";
- "0x20 0x41 0x42 0x43 0X44" which also translates to " ABCD";

The separator character may be specified as well as a replacement character for resulting non-printable characters. The resulting string will replace the original array of encoded characters if all of the array's encoded characters are identified successfully. If not, the original character array will remain in the property value.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.CharArrayToStringHandler

Parameters

- *file*: The path to the configuration file. Must use type="file" attribute.

Example:

```
<?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="Example">
    <matcher>
      <parameter name="filter">*</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
    CharArrayToStringHandler">
      <parameter name="file" type="file">conf/char-array-tostring-handler.xml</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/VariableHandlingFilterCharArrayToStringHandler"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.watch4net.com/VariableHandlingFilterCharArrayToStringHandler char-array-to-string-handler.xsd">
  <property>property</property>
  <asciiFormat>decimal</asciiFormat>
  <separator>
    <separatorString>.</separatorString>
  </separator>
  <nonPrintableCharReplacement> </nonPrintableCharReplacement>
</config>
```

- *property*: The property which contains the character array to transform. One or more must be supplied.
- *asciiFormat*: The format of the ASCII character. May be decimal (supports only decimal integers) or hexadecimal (supports hexadecimal, octal and decimal integers). Mandatory.
- *nonPrintableCharReplacement*: The character to use in order to replace unprintable characters (control characters, tab and delete). Optional.
- *separator*: A choice of separatorString or separatorCharacter. This is the encoded character array separator. Use the separatorString element when this character is known to be printable; use the separatorCharacter (encoded ASCII integer: i.e. A = 65) if the separator character is unprintable (control characters or delete). Mandatory.

11 IntegerArrayToStringHandler

Description

This handler generates a string from an array of integers, separated by a specific character. The following integer array types may be handled by this filter:

- "32.65.66.67.68"
- "0x20 0x41 0x42 0x43 0X44"

The separator character may be specified. The resulting string will replace the original array of integers if all of the array's integer values are identified successfully. If not, the original integer array will remain in the property value.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.IntegerArrayToStringHandler

Parameters

- *file*: The path to the configuration file. Must use `type="file"` attribute.

Example:

```
<?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="Example">
    <matcher>
      <parameter name="filter">*</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.IntegerArrayToStringHandler">
      <parameter name="file" type="file">conf/integer-array-to-string-handler.xml</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/VariableHandlingFilterIntegerArrayToStringHandler"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/VariableHandlingFilterIntegerArrayToStringHandler">
  <property>ifdescr</property>
  <asciiFormat>decimal</asciiFormat>
  <separator>
    <separatorString>.</separatorString>
  </separator>
</config>
```

```
</separator>
<alternateOutputFormat>
  <hexadecimalValue leading0x="false" valueSeparator="" toUpperCase="true"/>
</alternateOutputFormat>
</config>
```

- *property*: The property which contains the character array to transform. One or more must be supplied.
- *asciiFormat*: The format of the integer. May be decimal (supports only decimal integers) or hexadecimal (supports hexadecimal, octal and decimal integers). Mandatory.
- *separator*: A choice of separatorString or separatorCharacter. This is the value array separator. Use the separatorString element when this character is known to be printable; use the separatorCharacter (encoded ASCII integer: i.e. A = 65) if the separator character is unprintable (control characters or delete). Mandatory.
- *alternateOutputFormat*: Mandatory.
- *hexadecimalValue*: Output format specification. Set leading0x="true" if "0X" is required in the output string. Set valueSeparator="string" if a separator is required between values. Set toUpperCase="true" if uppercase values are required (A0 F0, instead of a0 f0). Using the options in this example, output format will be: "A018EF".

12 TimestampFromPropertyHandler

Description

This handler replaces the raw value timestamp with one provided in a property. Supported formats include:

- SNMP DateAndTime;
- Any date which may be parsed by a SimpleDateFormat instance;

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.TimestampFromPropertyHandler

Parameters

- *file*: The path to the configuration file. Must use `type="file"` attribute.

Example:

```
<?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="Example">
    <matcher>
      <parameter name="filter">*</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
    TimestampFromPropertyHandler">
      <parameter name="file" type="file">conf/timestamp-from-property-handler.xml</
      parameter>
    </handler>
  </handling-configuration>
</configuration>
```

Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns="http://www.watch4net.com/VariableHandlingFilterTimestampFromPropertyHandler"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/
  VariableHandlingFilterTimestampFromPropertyHandler_.../src/conf/com/watch4net/apg/v2/
  collector/plugins/variablehandlingfilter/timestamp-from-property-handler.xsd">
  <property>date</property>
  <convertFormatFrom>String</convertFormatFrom>
  <dateFormatPreProcessor>
    <matchOn>([0-9]{4}-[0-9]{1,2}-[0-9]{1,2} [0-9]{1,2}: [0-9]{1,2}: [0-9]{1,2}[+-]{1}[0-9]{1,2})
    : ([0-9]{1,2})</matchOn>
    <replaceWith>$1$2</replaceWith>
  </dateFormatPreProcessor>
  <dateFormat>yyyy-MM-dd HH:mm:ssZ</dateFormat>
```

```
</Config>
```

- *property*: The property which contains the data to use. One or more may be provided but the handler will set the date using the first usable one.
- *convertFormatFrom*: Either String or SNMP_DateAndTime. Mandatory.
- *dateFormatPreProcessor*: Use this to reformat a date that cannot be parsed directly by a SimpleDateFormat. The “matchOn” element contains the date format regex groups, while the “replaceWith” element contains the date format output. See java’s String.replaceAll method for more information about the preprocessor behavior. In this example, the input date format is the following: “2011-09-23 16:45:00+00:00”, which cannot be parsed with a SimpleDateFormat because there is a : in the timezone offset. This preprocessor configuration matches on the valid parts of the date format (group 1 and 2) and simply concatenates them into a valid date format “2011-09-23 16:45:00+0000”, which can be parsed by a SimpleDateFormat. The “dateFormatPreProcessor” is optional. More than one may be used in sequence.
- *dateFormat*: Supply this only when convertFormatFrom is set to String.

The SNMP_DateAndTime format specification is as follows:

A date-time specification.

field	octets	contents	range
		-----	-----
1	1-2	year*	0..65536
2	3	month	1..12
3	4	day	1..31
4	5	hour	0..23
5	6	minutes	0..59
6	7	seconds	0..60
		(use 60 for leap-second)	
7	8	deci-seconds	0..9
8	9	direction from UTC	'+' / '-'
9	10	hours from UTC*	0..13
10	11	minutes from UTC	0..59

* Notes:

- the value of year is in network-byte order

- daylight saving time in New Zealand is +13

For example, Tuesday May 26, 1992 at 1:30:15 PM EDT would be displayed as:

1992-5-26,13:30:15.0,-4:0

Note that if only local time is known, then timezone information (fields 8-10) is not present.

The SimpleDateFormat parses the date using a user specified data pattern. The pattern is defined as a string, using the following attributes:

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time
z	Time zone	General time zone	PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

Examples:

Date and Time Pattern	Accepted Input
yyyy-MM-dd HH:mm:ss z	2011-12-31 23:59:59 -0400
yyyy-MM-dd HH:mm:ss z	2011-12-31 23:59:59 EST
yyyy.MM.dd G 'at' HH:mm:ss z	2001.07.04 AD at 12:08:56 PDT
EEE, MMM d, ''yy	Wed, Jul 4, '01
h:mm a	12:08 PM
hh 'o'clock' a, zzzz	12 o'clock PM, Pacific Daylight Time
K:mm a, z	0:08 PM, PDT
yyyyy.MMMMM.dd GGG hh:mm aaa	02001.July.04 AD 12:08 PM
EEE, d MMM yyyy HH:mm:ss Z	Wed, 4 Jul 2001 12:08:56 -0700
yyMMddHHmmssZ	010704120856-0700
yyyy-MM-dd'T'HH:mm:ss.SSSZ	2001-07-04T12:08:56.235-0700

Local time is assumed if the time zone is not provided.

13 VlanPortIsVlansAllowedRangeExtractorHandler

Description

This handler specifically handles SNMP values found in the table column vlanPortIsVlansAllowed (1.3.6.1.4.1.9.5.1.9.3.1.5). Please check the schema file for details about the processing being done by this handler.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.
handlers.VlanPortIsVlansAllowedRangeExtractorHandler

Parameters

- file: The path to the configuration file. Must use type="file" attribute.

Example:

```
<?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="Example">
<matcher>
  <parameter name="filter">*</parameter>
</matcher>
<handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
VlanPortIsVlansAllowedRangeExtractorHandler">
  <parameter name="file" type="file">conf/vlan-port-is-lvlans-allowed-range-extractor.
xml</parameter>
</handler>
</handling-configuration>
</configuration>

```

Configuration

```

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/vlan-port-is-lvlans-allowed-range-extractor" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.watch4net.com/vlan-
port-is-lvlans-allowed-range-extractor_.../src/conf/com/watch4net/apg/v2/collector/plugins/
variablehandlingfilter/vlan-port-is-lvlans-allowed-range-extractor.xsd">
  <property>lvlans</property>
  <output-property-prefixname>outlvl</output-property-prefixname>
  <no-lvlan-string>No lvlans</no-lvlan-string>
  <bit-order>false</bit-order>
  <offset>0</offset>
</config>

```

- *property*: The property which contains the value of the vlanPortIsVlansAllowed row. Mandatory.
- *output-property-prefixname*: The name of the property which will contain the IVlan allowed ranges. Mandatory, limited to 7 characters.
- *bit-order*: indicates the bit order. false for decreasing bit order (default, true for increasing. Leave this to the default value when handling vlanPortIsVlansAllowed values.
- *offset*: The bit position offset. Leave this to 0 when handling vlanPortIsVlansAllowed values.

14 PropertyArithmeticHandler

Description

This handler is designed to perform arithmetic operations, using a raw value's property, and store the result in a new or existing property on the same raw value.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.PropertyArithmeticHandler

Parameters

- *file*: The path to the configuration file. Must use `type="file"` attribute.

Example:

```
<?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="Example">
    <matcher>
      <parameter name="filter">*</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
    PropertyArithmeticHandler">
      <parameter name="file" type="file">conf/property-arithmetic-handler.xml</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/property-arithmetic-handler"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.watch4net.com/property-arithmetic-handler_.../src/conf/com
/watch4net/apg/v2/collector/plugins/variablehandlingfilter/property-arithmetic-handler.xsd_...">
  <output-property>bandlim</output-property>
  <input-property>bandpct</input-property>
  <input-property>ifspeed</input-property>
  <expression>(bandpct / 100) * ifspeed</expression>
</config>

```

- *output-property*: The property which contains the result. (Optional, 0 or more)
- *output-value*: Will put the result in the value field of the raw value. (Optional, 0 or 1)
- *output-timestamp*: Will put the result in the timestamp field of the raw value. (Optional, 0 or 1)
- *input-property*: The property used in the calculation. (Optional, 0 or more)
- *input-value*: The value field of the raw value. The given name will be used in the expression. (Optional, 0 or 1)
- *input-timestamp*: The timestamp field of the raw value. The given name will be used in the expression. (Optional, 0 or 1)
- *expression*: The arithmetic expression. See <http://commons.apache.org/jexl/reference/syntax.html> for reference. this filter only supports simple, one line expression.

15 HexadecimalCounterHandler

Description

This handler is used to process metrics that have their value stored in a property as an hexadecimal counter. The metric value is replaced by the property's hexadecimal value converted to a float. This handler can perform rate, positive rate, delta or positive delta computations on metrics that have different timestamps but identical variable ids. It can also simply transform the hexadecimal value to a float.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.HexadecimalCounterHandler

Parameters

- *hex-value-property*: The name of the property which contains the hexadecimal value to convert / compute.
- *operation*: The operation. Possible values are: rate, delta, ratep, deltap, none. There is no default value.
- *drop-property*: Drop the hex-value-property after handling it. true or false. There is no default value.

Operations definition

- *delta*: Variation of the value. (`lastValue - currentValue`)
- *deltap*: Same as delta but drops negative values: negative values are not sent to the next handler.
- *rate*: Variation of the value in function of the time. (`(lastvalue - currentValue) / (lastTime - currentTime)`)
- *ratep*: Same as rate but drops negative values: negative values are not sent to the next handler.
- *none*: Set the value with the converted hexadecimal value of the property.

16 ReplaceRegexHandler

Description

This handler is a simple handler that replaces a regex into a property value with specified replacement.

Java class

`com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.ReplaceRegexHandler`

Parameters

- *property*: The name of the property to match and to modify according to the provided regex and replacement.
- *regex*: The regular expression to apply to the property.

- *replacement*: The replacement value to insert according to the regex. Will replace all occurrence of matched regex.

Example: remove redundant commas

```
<?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="Example">
    <matcher>
      <parameter name="filter">*</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
ReplaceRegexHandler">
      <parameter name="property">desc</parameter>
      <parameter name="regex">,</parameter>
      <parameter name="replacement">,</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

If we have in input a value with the property desc with this value Too,, ,many,,,,,commas.. The output for the property desc will be Too ,many ,commas ..

17 PropertyMonitorHandler

Description

This handler is a simple handler that monitor changes in raw values properties and adds a +r when a change is detected. Note that when the handler first sees a metric (e.g. upon restart), it also forces a +r.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.PropertyMonitorHandler

Parameters

- *includes*: An optional comma separated list of properties to consider when detecting changes. If not specified or empty, all the properties will be considered.
- *excludes*: An optional comma separated list of properties to ignore when detecting changes. If not specified or empty, no property will be ignored.

Example: detect changes in the `bunit` property

```
<?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="Example">
    <matcher>
      <parameter name="filter">*</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
PropertyMonitorHandler">
      <parameter name="includes">bunit</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

In this example, the handler will force a `+r` when a change in the property `bunit` is detected.

18 VariableStateMonitorHandler

Description

This handler monitors the state of raw values and allows tagging them according to various criteria.

Java class

`com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.VariableStateMonitorHandler`

Parameters

- *state-value*: The expression which computes the state of the current value. The raw value is available in the expression as the variable `value`. It exposes the usual raw values methods, including `getValue()` and `getTimestamp()` to get the actual value and its corresponding timestamp. See <http://commons.apache.org/jexl/reference/syntax.html> for reference.
- *property-name*: The name of the property which will be used to tag values.
- *property-value*: The expression which computes the property value. As in *state-value*, the raw value is available in the expression as the variable `value`, and the current state as the variable `state`. The `state` exposes only two methods: `getLastChangedAt()` which returns the timestamp of the last state change and `getCurrentValue()` which returns the current state value. See <http://commons.apache.org/jexl/reference/syntax.html> for reference.

Example: tag components inactive for more than 1 hour

```
<?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="Example">
    <matcher>
      <parameter name="filter">name=='CurrentUtilization'</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
VariableStateMonitorHandler">
      <parameter name="state-value">value.getValue() == 0</parameter>
      <parameter name="property-name">inactive</parameter>
      <parameter name="property-value">value.getTimestamp() - state.getLastChangedAt() >= 3600
&amp;&amp; state.getCurrentValue() == true</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

19 SendLastValueHandler

Description

This handler is devised to let through only the last 'n' values for a specific variable. It assumes that values are sent in chronological order.

Handling values polled from a SNMP table, which contains more than one result for an instance, is a typical case for using this filter.

If we have a table like this one:

```
.2011.5.25.111.4.3.1.20.2.65.70.1192 = Counter32: 74745
.2011.5.25.111.4.3.1.20.2.65.70.1193 = Counter32: 58909
.2011.5.25.111.4.3.1.20.2.65.70.1194 = Counter32: 119715
.2011.5.25.111.4.3.1.20.2.65.70.1195 = Counter32: 82278
.2011.5.25.111.4.3.1.20.2.65.70.1196 = Counter32: 61713
```

The variable id is stored in the raw value's variable while the instance id is stored in a distinct property. Typically: resultid.

- The variable id is the fixed part: .2.65.70
- The instance id is the last part: 1192, 1193, 1194, 1195, 1196

In this case, there is no timestamp linked to the result values, only the polling timestamp.

This filter can handle such case by providing several alternative processes:

- Send only the last value (with result id 1196) with its polled timestamp.
- Send all latest values but adjust their polled timestamps in the past, by assuming a fixed time interval.
- If the the result time interval is available in a property, perform same timestamp calculation as in 2, using this value instead of assuming one.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.sendlastvalues .SendLastValue-sHandler

Parameters

- *file*: The path to the configuration file. Must use `type="file"` attribute.

Example:

```
<?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="Example">
    <matcher>
      <parameter name="filter">name='Metric%</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
    sendlastvalues.SendLastValuesHandler">
      <parameter name="file" type="file">conf/send-last-values-for-probes.xml</parameter>
    </handler>
  </handling-configuration>
</configuration>
```

Example

Send up to the last 6 values for selected metrics; calculate the timestamp on the raw values by assuming that the interval between values is 10 seconds. The variable instance id is stored in the "resultid" property. Assume that the instance id was reset if the difference between the latest instanceid value and the current one is more than 10.

```
<?xml version="1.0" encoding="UTF-8"?>
<config idProperty="variable" instanceIdDeltaResetThreshold="10" instanceIdProperty="resultid"
  xmlns="http://www.watch4net.com/send-last-values-handler"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/send-last-values-handler../../../../src/conf/
  com/watch4net/apg/v2/collector/plugins/variablehandlingfilter/send-last-values-handler.xsd">
  <timestamp-correction calculateTimestamp="true" maxInstancesSent="6">
    <assumed-time-interval>10</assumed-time-interval>
  </timestamp-correction>
</config>
```

20 SpatialAggregationHandler

Description

This handler spatially aggregates values. There are three ways of aggregation:

- inventory : commits aggregation of all different last values.
- timestamp-change : commits aggregation based on value timestamps.
- flush : commits aggregation values when flush a received. (for specific cases)

Note: The flush aggregation will only outputs consistent values when only one supported collector gives data to the variable handling filter. Supported collectors are the ones which throw a flush after each polling. Collectors with multiple polling or throwing multiples flushes in one polling are not supported. It is better to assume that most collectors are not compatible with this option.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.SpatialAggregationHandler

Parameters

- *file*: The path to the configuration file. Must use `type="file"` attribute.

Example:

```
<?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="Example">
    <matcher>
      <parameter name="filter">*</parameter>
    </matcher>
    <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
      SpatialAggregationHandler">
```

```

    <parameter name="file" type="file">conf/spatial-aggregation-handler.xml</parameter>
  </handler>
</handling-configuration>
</configuration>

```

Configuration

```

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/spatial-aggregation-handler"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/spatial-aggregation-handler spatial-
aggregation-handler.xsd">

  <aggregation id="ID-of-conf-block" forward-input="true">

    <filter>name=='metric1'</filter>
    <group-by>source</group-by>
    <group-by>device</group-by>
    <group-by>parttype</group-by>
    <group-by>prop</group-by>

    <compute spatial="avg">
      <output-timestamp use="first" rounding="ceiling" rounding-base="1" />
      <output-group>group</output-group>
      <flush timeout="240" />
    </compute>

    <properties>
      <add name="name">metric1Avg</add>
      <add name="type">aggregate</add>
      <remove name="prop" />
      <merge name="part" separator="_" />
    </properties>

  </aggregation>

  <aggregation id="ID-of-conf-block" forward-input="true">

    <filter>name=='metric2'</filter>
    <group-by>source</group-by>

```

```
<group-by>device</group-by>
<group-by>parttype</group-by>

<compute spatial="count">
  <output-timestamp use="max" rounding="floor" rounding-base="60" />
  <inventory period="300" removal-timeout="300" />
</compute>

<properties>
  <add name="name">metric2Count</add>
</properties>

</aggregation>

<aggregation id="ID-of-conf-block" forward-input="true">

  <filter>name=='metric3'</filter>
  <group-by>source</group-by>
  <group-by>device</group-by>
  <group-by>parttype</group-by>

  <compute spatial="max">
    <output-timestamp use="last" rounding="ceiling" rounding-base="1" />
    <timestamp-change period="300" timeout="240" max-aggregation-units="3" />
  </compute>

  <properties>
    <add name="name">metric3Max</add>
    <add name="type">aggregate</add>
  </properties>

</aggregation>

</config>
```

Description of aggregators.

- *inventory*: Keeps the last value of each metric to aggregate them every defined period. Metrics no longer collected are removed according to the removal-timeout.
- *timestamp-change*: Put metrics in the right time block according to their timestamp. Can have multiple time blocks active at the same time (aggregation unit).

- *flush*: Uses the flush signal from collector to aggregate input. Should not be used except in specific cases as most of collectors are not compatible with this feature.

Description of configuration:

- *aggregation id*: The id for this aggregation. Used as a part of the output variable ID. (mandatory)
- *aggregation forward-input*: If set to true, all metrics used to process aggregation will be sent to the next processing element. If set to false, they will not. (mandatory)
- *filter*: APG filter to restrict which metrics are used for this aggregation. (mandatory)
- *group-by*: Properties to use to group aggregation. All metrics sharing the same values for these properties will be aggregated together. These are the default properties for the output variable. (mandatory, from 1 to many)
- *compute spatial*: Type of aggregation : sum, avg, min, max, count. (mandatory)
- *output-timestamp use*: Which timestamp from metrics to use: first, last, min, max (optional, default is first)
- *output-timestamp rounding*: Type of rounding : ceiling, floor. (optional, default is floor)
- *output-timestamp rounding-base*: Base for rounding. If value is set to 5, output timestamp will be rounded to the 5 seconds. (optional, default is 1)
- *output-group*: The output group name. (optional, default is the same of input metrics)
- *flush timeout*: Aggregation timeout in seconds. If no flush occurs before timeout, metrics already received are aggregated and sent. (mandatory)
- *inventory period*: Period of aggregation in seconds. Aggregate last value of each metric for each period. (mandatory)
- *inventory removal-timeout*: If no new value is received for a metric before the timeout in seconds, it is removed from inventory. (mandatory)
- *timestamp-change period*: Period of aggregation in seconds. Check the timestamp of received metrics. (mandatory)
- *timestamp-change timeout*: Aggregation timeout in seconds. Time to wait before sending aggregated data after time block creation. (mandatory)
- *timestamp-change max-aggregation-units*: Maximum number of concurrent aggregation processing units for each aggregation. (mandatory)

Note: The variable ID is constructed based on all group-by properties values combined with the aggregation id.

Note: Metrics which enter this handler but not matching any aggregation filter are forwarded automatically to the next processing element.

Note: Only group-by properties are included by default for the output variable. All other properties from input variables are ignored except when configured in properties configuration element.

21 PropertyStringCaseHandler

Description

This handler puts to upper and lower case specified properties.

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.PropertyStringCaseHandler

Parameters

- *to-upper-case*: Space separated list of properties to upper case.
- *to-lower-case*: Space separated list of properties to lower case.

Example:

```
<?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="Example">
  <matcher>
    <parameter name="filter">*</parameter>
  </matcher>
  <handler class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.
PropertyStringCaseHandler">
    <parameter name="to-upper-case">device part</parameter>
    <parameter name="to-lower-case">devtype unit name</parameter>
  </handler>
</handling-configuration>
</configuration>

```

22 DefinedLengthByteSequenceSplitHandler

Description

This is used to split sequence of bytes into subsequences of bytes. The format of the byte sequence handled by this handler is as follows:

$n \cdot ((\text{decimal byte value: } 0-255) \cdot (n \text{ times}) \cdot ((\text{decimal byte value: } 0-255) +$

Where “n” is the number of bytes in the sub-sequence.

Example:

The byte sequence “3.65.66.67.2.66.67.4.67.68.69.70” is split using the following logic:

- The first part of this sequence is 3 bytes long. It contains bytes “65.66.67” (which could be translated to “ABC”).
- The second part is 2 bytes long. It contains bytes “66.67” (which could be translated to “BC”).
- The last part is 4 bytes long. It contains bytes “67.68.69.70” (which can could translated to “CDEF”).

Java class

com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.DefinedLengthByteSequenceSplitHandler

Configuration

- *input-property-name* The name of the property which contains the byte sequence to split.

- *delimiter* The delimiter character which isolate bytes input sequence.
- *output-property name* The name of the property which contains the output sub-sequence.

Example: extract 2 strings from the property named “input” to properties named output1 and output2

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/defined-length-byte-sequence-split" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.watch4net.com/defined-length
-byte-sequence-split_defined-length-byte-sequence-split.xsd">
  <input-property-name>property</input-property-name>
  <delimiter>.</delimiter>
  <output-property name="output1" />
  <output-property name="output2" />
</config>
```

Using the sample configuration on the input byte sequence “5.72.69.76.76.79.3.69.77.67” sets the following properties on the handled raw value:

- output1=“72.69.76.76.79”
- output2=“69.77.67”

The CharArrayToStringHandler can be used to translate this filter’s output subsequences to actual ASCII characters. In the case of this example, applying the CharArrayToStringHandler to the resulting subsequence would result:

- output1=“HELLO”
- output2=“EMC”

One instance where the DefinedLengthByteSequenceSplitHandler may be used is with SNMP OIDs which represent strings, expressed as byte ASCII characters, where the length of substrings is specified at the beginning of the substring. The pingCtlTable (.1.3.6.1.2.1.80.1.2) is an example of that case.

23 HexaDecimalToDecimalConverterHandler

Description

This handler formats the value of one or more hexadecimal value properties to decimal format. The property value can optionally be split on a given separator, and all tokens MUST be hexadecimal

values. Also the supported format of the hexadecimal is '0x..' or '0X..' or '#..' hex digits. If the number is not a valid hexadecimal number, then the property value is dropped.

For example,

- *hostlun=0xa*: converts to *hostlun=10*
- *hostlun=N/A*: converts to *hostlun=*
- *hostlun=0xa,invalid* converts to *hostlun=10* (assuming input separator is comma)

Java class

`com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.HexaDecimalToDecimalConverterHandler`

Parameters

- *input-separator*: The separator to use to tokenize the input property value into hexadecimal numbers.
- *output-separator*: The separator to use join the formatted tokens together after they are converted. If not supplied, but the input is delimited, the output values would be joined without any separator.
- *property*: A comma-separated list of the properties for which values must be converted.