Additional Policy Syntax Documentation

For Operations Bridge Manager (OBM) 2020.10

MICRO FOCUS

Contents

Introduction

1. Purpose of This Document

This document explains the policy syntax of 19 policy templates that must be edited using the RAW policy editor (and for which specific Flash-based editors existed in the past).

It is meant to be used together with the existing documentation for each policy template. The existing Policy Templates documentation explains all features of a policy template, various parts a policy consists of, possible options and values, etc.

If you are a new user and want to start developing a policy, please read the Policy Templates documentation first, then use this document to understand how to implement a certain policy feature using the correct policy syntax and the RAW policy editor.

Note, however, that we continue to include the Flash-based policy editor UIs in the product as a fallback. See Access Adobe Flash-based User Interface for details.

This document explains the syntax of each policy template by providing example policy templates. The examples are also available as a Management Pack that can be uploaded to Operations Bridge Manager (OBM). We recommend that you use the provided examples as a starting point when creating new or modifying existing policy templates.

This document does not explain the policy syntax of policy templates for which a specific HTML policy editor exists, such as Measurement Threshold or Windows Event Log policy templates. For those policy templates, please refer to the Policy Templates documentation and use the normal policy editor.

2. Usage Guidelines

Document structure and how to use this document

This document consists of three major parts: the *Introduction* part with usage guidelines, the *Policy Syntax Tables* part and the *Detailed Syntax Descriptions* part.

Please read *Introduction* including usage guidelines first, as these provide general information that is required for understanding the rest of the document.

In *Policy Syntax Tables*, you can find syntax tables for all 19 policy template types. A syntax table explains the structure of the policy template using an example and breaks it down into sections that are further explained in *Detailed Syntax Descriptions.*

Use the syntax table to understand the structure, then use the link to a section to understand the details and possible values of a certain section.

You can navigate back and forth between the syntax table and sections using the provided links.

Additionally, when you want to use a certain syntax part in your policy template, we recommend that you open the corresponding example policy template in OBM and copy and paste the corresponding part from the example policy template.

Note that an example policy template should not be deployed as it is, as it contains many placeholders like log file names or database column names that need to be replaced with real names, etc. to create a meaningful policy.

There is a lot of common syntax elements if you compare the syntax of two policy template types.
For example, the "Event from Structured Log File" policy template offers the same event options as the "Event from XML File" policy template as both create an event.

Similar, the "Metric from Structured Log File" policy template offers the same source options as the "Event from Structured Log File" policy template as both get data from the same source – a structured log file.

Therefore, if you understand the syntax of one policy template type, you can apply its parts when creating another policy template.

> NOTE
>
> To avoid duplication, syntax blocks that appear in multiple policy template types are described separately and referenced in the syntax table of a policy template type.

Colors used in syntax examples

Black text – do not change, keep as it is
Blue text – adjust according to your needs
Grey text – part of the syntax that is described in another section

Example:

```
NODE IP 0.0.0.0  "<$DATA:Node>"
APPLICATION "<$DATA:application>"
MSGGRP "Category"
```

UI references and links to the online documentation

To explain the syntax of a policy template, the corresponding (Flash-based) UI is displayed as a reference. However, this document does not explain every feature of a policy template in detail, as these are already described in the online documentation (part of the Policy Templates documentation).

Therefore, if you want to understand all features of a policy template, follow the provided links to the corresponding online documentation.

Please note that those links may refer you to a section of a policy template that is not the policy template you are currently working with but that has the same section. For example, the Options section is used in 10 policy templates and included in the documentation of 10 policy templates, but the provided link takes you to the Options section of one specific policy template.

Management Pack with example policy templates

The *Example Policy Templates* Management Pack containing all example policies used in the following sections is available as part of OBM.

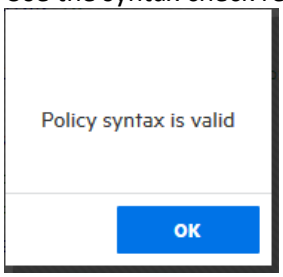
You select Management Packs to install in the Management Packs page of the OBM configuration wizard. To install the *Example Policy Templates* Management Pack after the first configuration, run the configuration wizard on the Data Processing Server and select the *Example Policy Templates* Management Pack. You can also use the opr-mp-installer Command-Line Interface to install the *Example Policy Templates* Management Pack on the Data Processing Server. For details, see opr-mp-installer Command-Line Interface.

We recommend that you use these examples as a starting point when developing a new or extending an existing policy template. Open the corresponding example, then delete unnecessary parts, copy and paste rule parts as needed and edit them for your specific needs.

Syntax check

The Raw policy editor provides a syntax check option.

Use the syntax check regularly to check if the policy syntax is valid.



Please note that the syntax check checks if the structure of the policy is correct. It does not check if the values are within limits or if the names of CUSTOM and PARAM fields (which will be explained later) are valid.

Therefore, it is recommended to test a policy regularly by deploying it to a node and to check corresponding log files for errors (see Troubleshooting Tips below).

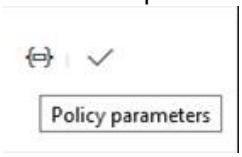
Passwords

Event, WMI and Database policy templates allow specifying users and passwords (for actions or as part of login credentials to databases or WMI).
It is a good practice to use policy parameters for those password values as this provides several advantages:

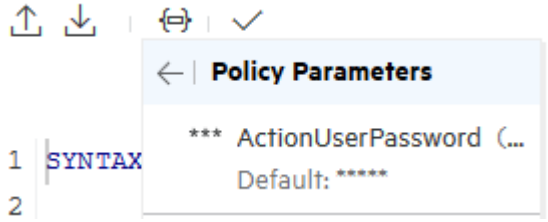
- The password can be easily changed (for example when it expired) without having to edit and re-deploy policies.
- The same policy can be used for different nodes, using different passwords when assigning the policy.

It is therefore recommended that you use password parameters wherever possible.

To create a password parameter in the RAW policy editor, click the Policy parameters icon:



Choose “String (password)” as a parameter type. Then use the parameter picker to insert the parameter into the policy syntax at the corresponding place:



The parameter picker will add the parameter using the %%<parameter name>%% syntax.
Add the password parameter at the correct location in the policy syntax:

Policy type	Usage of password	Example
Any event policy	Action user passwords	`USER "user" PWD "%%ActionUserPassword%%"`
WMI	Password to connect to WMI	`WMI_PASSWORD "%%WMIUserPassword%%"`
Event from DB Metric from DB Generic Output from DB	Password to connect to Database	`PARAM "password" "%%DBUserPassword%%"` **`ATT "encryption"  "ovcrypt"`**

NOTE

For DB policies, the ATT "encryption" "ovcrypt" line must follow the PARAM "password" line.
There is a known defect when editing DB policy types using the Adobe Flash-based editor: It does not add ATT "encryption" "ovcrypt" line to the policy syntax when a password parameter is used.

When switching between the RAW and Flash-based editors, make sure to add the ATT "encryption" "ovcrypt" line behind the Database password line when a password parameter is used using the RAW editor before deploying the policy.

<$DATA> references

When setting event or metric properties, you typically want to refer to data that you extracted from the source. For this, you can use the *<$DATA:SomeProperty>* notation. The Flash-based editors allowed loading source data and also allowed drag and drop of such properties. In this case, the dragged property was automatically converted into *<$DATA:>* notation. In the RAW policy editor, you must do that manually and refer to properties of the corresponding source according to the following table:

Policy template type	<$DATA:reference>	Example source-specific extraction of properties	Example reference
Structured Log File	`<$DATA:<variable>>`	`PARAM "struliPattern" "<#.EventId>,<*.Description>"`	`<$DATA:EventId>` `<$DATA:Description>`

Database	`<$DATA:<columnname>>`	`PARAM "sql" "SELECT count(*) FROM event.dbo.all_events WHERE state = 'IN_PROGRESS') AS in_progress"`
		or
		`PARAM "sql" "SELECT id,description FROM event.dbo.all_events WHERE state = 'IN_PROGRESS'"`
			`<$DATA:in_progress>`
			
			`<$DATA:id>`
			`<$DATA:description>`

XML File and REST Web Service Listener	`<$DATA:/<path to XML property from Root>/<XML Property>`	Example XML content:
		`<?xml version="1.0" encoding="UTF-8"?>`
		`<Event>`
		`<EventName>EndRequest</EventName>`
		`<Category>Administrative</Category>`
		`</Event>`
			`<$DATA:/Event/EventName>`
			`<$DATA:/Event/Category>`

Perl	`<$DATA:<AttributeName>>`	Example perl transfer data structure:
		`my %BASIC_METRIC_DATA = (`
		`"counter_name" => "<Value>",`
		`"counter_value" => "<Value>",`
		`"time_measured" => "<Value>"`
		`);`
			`<$DATA:counter_name>`
			`<$DATA:counter_value>`
			`<$DATA:time_measured>`

WMI	`<$WBEM:class property>` Note: Use $WBEM instead of $DATA!	WMI either returns a WMI instance or a WMI event with a TargetInstance. Properties of both can be accessed in the same way	`<$WBEM:Instance.Name>` `<$WBEM:TargetInstance.TrustedDomain>`

UUIDs & UUID generators

In various places within the policy syntax, it is necessary to use so-called UUIDs. A UUID is a universally unique identifier. Within policy templates, a UUID is required for example for a condition, an instruction text or a forwarding target. Flash-based and HTML editors generate those UUIDs automatically. When you use the RAW editor and add items requiring a UUID, you need to generate a new UUID manually. You can use UUID generators for this.

On Linux, uuidgen can be used to generate UUIDs.
On Windows, you can use the Windows Powershell guid object.

Below are examples how to create a new UUID:

Linux:

```
# uuidgen
7e0424e7-d84f-420c-bc4b-e3b3e08cc97f
```

Windows Powershell:

```
PS C:\>[guid]::NewGuid()

Guid
----
1b416870-c9b6-47d7-ab39-72e946b6ea14
```

Copy the new UUID (for example **`7e0424e7-d84f-420c-bc4b-e3b3e08cc97f`**) from the command output into your policy.

Troubleshooting tips

The Operations Agent writes warning and error messages and informational notifications in the System.txt file on the node. The contents of the System.txt file reveal if the agent is functioning as expected. Critical errors are also reported via events to the OBM server.

Therefore, when testing a policy template on a node, check if error events have been sent to OBM. Also, check the System.txt file on the node for errors or warnings, as it provides more details. For the location of the System.txt file, see https://docs.microfocus.com/itom/Operations_Agent:12.14/LoggingTaskTracing. Look for errors or warnings concerning the policy you just deployed.

Policy Syntax Tables

3. Syntax of Windows Management Interface policy template

Example policy template

```
SYNTAX_VERSION 11

WBEM "Example Windows Management Interface policy template"
DESCRIPTION "Example policy that uses many available features to
illustrate policy template syntax"
```

Do not change – can be changed in Policy Properties

```
NAMESPACE "\\\\node.example.com\\root\\MicrosoftActiveDirectory"
CLASS "__InstanceOperationEvent"
WITHIN "%%FREQ%%h0m0s"
WHERE_CLAUSE "TargetInstance ISA \"Microsoft_DomainTrustStatus\""
QUERY_LANGUAGE "WQL"
```

See **Windows Management Interface Source** section

```
MPI_AGT_COPY_MSG
HELPTEXT "hardcoded instruction text"
HELP "53373c12-f6fb-493c-bf3f-e83328290591"
SEVERITY Critical
NODE IP 0.0.0.0  "Node"
APPLICATION "app"
MSGGRP "Category"
OBJECT "obj"
SERVICE_NAME "id"
MSGKEY "key"
SEPARATORS ";"
```

Default Event Attributes section, see **Set Event Attributes** section

```
SUPP_DUPL_IDENT
    "0h30m0s" RESEND "8h0m0s"
LOGMATCHEDMSGCOND
LOGMATCHEDSUPPRESS
LOGUNMATCHED
```

See **Event Options** section (Windows Management Interface)

Following, there can be a mix of rules of 3 types:
Event generating rules, starting with `MSGCONDITIONS`
Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and
Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS`
There can be multiple blocks of each type.
Rules are processed from top to bottom until the first rule matches.

Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks.
Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks.
See **Condition** section.

MSGCONDITIONS	**Start of event generating Rules section**
DESCRIPTION "set all event attributes, default correlation, ca, hardcoded instruction, advanced, no action" CONDITION_ID "81582EC5-525B-8A4C-230B-673B873B7978" CONDITION "Severity" == "CRITICAL"	1. Rule **Rule section** Description, and condition of the first rule.
SET	Do not change, **start of Rule Set section**
SERVERLOGONLY "true" SEVERITY "Critical" TIMECREATED "<$DATA:Timestamp>" NODE IP 0.0.0.0 "<$DATA:Node>" APPLICATION "<$DATA:application>" MSGGRP "Category" OBJECT "<$DATA:object>" MSGTYPE "<$DATA:type>" SERVICE_NAME "<$DATA:HPOM service ID>" MSGKEY "<$DATA:Node>:<$DATA:EventId>" MSGKEYRELATION ACK "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS " " CUSTOM "Description" "<$DATA:Description>" CUSTOM "SubCategory" "Subcategory" CUSTOM "RelatedCiHint" "<$DATA:RelatedCI>" CUSTOM "NoDuplicateSuppression" "false" CUSTOM "EtiHint" "ETIName:Value" CUSTOM "SourceCiHint" "<$DATA:SourceCI>" CUSTOM "SubCiHint" "Sub Component" CUSTOM "SourcedFromExternalId" "<$DATA:EventId>" CUSTOM "CA_1" "2" CUSTOM "SourcedFromExternalUrl" "https://<DATA:url>" TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"	See **Set Event Attributes** section - this part sets the non-default attributes of the event generated by the first rule
SUPPRESSCONDITIONS	Suppress on matched rule section
DESCRIPTION "suppress on matched rule" CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206" CONDITION "SomeProperty" == "SomeValue"	2. Rule

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`
	3. Rule

`MSGCONDITIONS`	**Start of another event generating Rules block**
`DESCRIPTION …`	4. Rule **Rule section**
`SET …`	See **Set Event Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

4. Syntax of Event from Structured Log File policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Event from Structured Log File policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – can be changed in Policy Properties

`POLTYPE "struli"`
`  GROUP "ROOT"`
`  GROUP "content"`
	Do not change

`  GROUP "options"`
`  PARAM "isForwardUnMatchedMsgToSrv"  "1"`
`  PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"`
`  PARAM "fieldSep"  "&#x20;&#x09;"`
`  PARAM "caseSensitive"  "1"`
`  PARAM "isLogMatchedCond"  "1"`
`  PARAM "isLogMatchedSuppressCond"  "1"`
`  PARAM "isLogUnMatchedCond"  "1"`
`  GROUP_END`
	See the **Event Options** section

`  GROUP "sources"`
`  PARAM "logpath"  "/path/logfile"`
`  PARAM "interval"  "%%FREQ%%m30s"`
`  PARAM "chSet"  "69"`
`  PARAM "readMode"  "firstFromBegin"`
`  PARAM "noLogfileMsg"  "1"`
`  PARAM "closeAfterRead"  "1"`
`  GROUP "struliPatterns"`
`  PARAM "struliPattern" "^<@.Severity>,<@.SourceCI>,<@.RelatedCI>,<*.Timestamp>,<@.Node>,<#.EventId>,<*.Description>"`
`  GROUP_END`
`  GROUP "startPatterns"`
`  PARAM "startPattern"  "<*>,<*>,<#>/<#>/<#> <#>:<#>:<#> <2*>,<*>"`
`  GROUP_END`
`  GROUP_END`
	See the **Structured Log File Source** section

`  GROUP_END` `GROUP_END`	Do not change

`  MAP "map1"`
`  DESCRIPTION ""`
`  INPUT "input1"`
`  FROM "source1" TO "target1"`
`  FROM "source2" TO "target2"`
`  MAP "map2"`
`  DESCRIPTION ""`
`  INPUT "input2"`
`  FROM "source3" TO "target3"`
`  FROM "source4" TO "target4"`
	Optional section
	
	See **Mappings** section

`DEFAULTMSG`	Do not change **Start of Default event attributes**

`SERVERLOGONLY "false"`
`SEVERITY "Critical"`
`TIMECREATED "Time created"`
`NODE IP 0.0.0.0 "Node"`
`APPLICATION "DefaultApplication"`
`MSGGRP "Category"`
`OBJECT "DefaultObject"`
`MSGTYPE "DefaultType"`
`SERVICE_NAME "DefaultHPOMServiceID"`
`MSGKEY "DefaultEventKey"`
`MSGKEYRELATION ACK "<*>" SEPARATORS "  "`
`CUSTOM "Description" "Description"`
`CUSTOM "SubCategory" "Subcategory"`
`CUSTOM "RelatedCiHint" "Related CI"`
`CUSTOM "SourceCiHint" "Source CI"`
`CUSTOM "SubCiHint" "Sub Component"`
`CUSTOM "EtiHint" "ETI"`
`CUSTOM "SourcedFromExternalId" "Source Event ID"`
`CUSTOM "SourcedFromExternalUrl" "https://example.com"`
`CUSTOM "CA_0" "1"`
`CUSTOM "NoDuplicateSuppression" "true"`
`TEXT "Title"`
`AUTOACTION "DefaultAACmd" ACTIONNODE IP 0.0.0.0 "<$MSG_NODE_NAME>" ANNOTATE ACK`
`SEND_MSG_AFTER_LOC_AA SEND_FAILED_MSG`
`USER "user1" PWD "%%ActionUserPassword%%"`
`SIGNATURE ""`
`OPACTION "DefaultOperCmd" ACTIONNODE IP 0.0.0.0 "<$OPC_MGMTSV>"`
`SIGNATURE ""`
`MPI_AGT_COPY_MSG`
`HELPTEXT "DefaultinstructionText"`
`HELP "4EC0F729-6037-A577-8AFF-11A422242A70"`
`SUPP_DUPL_IDENT`
`"0h30m0s" RESEND "8h0m0s"`
	See **Set Event Attributes** section
	
	- this part sets the default attributes of the generated events

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types:
Event generating rules, starting with `MSGCONDITIONS`
Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and
Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS`
There can be multiple blocks of each type.
Rules are processed from top to bottom until the first rule matches.

Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks.
Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks.
See **Condition** section.

MSGCONDITIONS	**Start of event generating Rules section**
DESCRIPTION "set all event attributes, default correlation, ca, hardcoded instruction, advanced, no action" CONDITION_ID "81582EC5-525B-8A4C-230B-673B873B7978" CONDITION "Severity" == "CRITICAL"	1. Rule **Rule section** Description, and condition of the first rule
SET	Do not change, **start of Rule Set section**
SERVERLOGONLY "true" SEVERITY "Critical" TIMECREATED "<$DATA:Timestamp>" NODE IP 0.0.0.0 "<$DATA:Node>" APPLICATION "<$DATA:application>" MSGGRP "Category" OBJECT "<$DATA:object>" MSGTYPE "<$DATA:type>" SERVICE_NAME "<$DATA:HPOM service ID>" MSGKEY "<$DATA:Node>:<$DATA:EventId>" MSGKEYRELATION ACK "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS " " CUSTOM "Description" "<$DATA:Description>" CUSTOM "SubCategory" "Subcategory" CUSTOM "RelatedCiHint" "<$DATA:RelatedCI>" CUSTOM "NoDuplicateSuppression" "false" CUSTOM "EtiHint" "ETIName:Value" CUSTOM "SourceCiHint" "<$DATA:SourceCI>" CUSTOM "SubCiHint" "Sub Component" CUSTOM "SourcedFromExternalId" "<$DATA:EventId>" CUSTOM "CA_1" "2" CUSTOM "SourcedFromExternalUrl" "https://<DATA:url>" TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"	See **Set Event Attributes** section - this part sets the non-default attributes of the event generated by the first rule
SUPPRESSCONDITIONS	Suppress on matched rule section
DESCRIPTION "suppress on matched rule" CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206" CONDITION "SomeProperty" == "SomeValue"	2. Rule

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`
	3. Rule

`MSGCONDITIONS`	**Start of another event generating Rules block**
`DESCRIPTION …`	4. Rule **Rule section**
`SET …`	See **Set Event Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

5. Syntax of Event from XML File policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Event from Structured Log File policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – can be changed in Policy Properties

`POLTYPE "xml"`
`  GROUP "ROOT"`
`  GROUP "content"`
	Do not change

`  GROUP "options"`
`  PARAM "isForwardUnMatchedMsgToSrv"  "1"`
`  PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"`
`  PARAM "fieldSep"  "&#x20;&#x09;"`
`  PARAM "caseSensitive"  "1"`
`  PARAM "isLogMatchedCond"  "1"`
`  PARAM "isLogMatchedSuppressCond"  "1"`
`  PARAM "isLogUnMatchedCond"  "1"`
`  GROUP_END`
	See the **Event** Options section

`  GROUP "sources"`
`  PARAM "logpath"  "/path/logfile.xml"`
`  PARAM "interval"  "5m"`
`  PARAM "chSet"  "69"`
`  PARAM "readMode"  "fromLastPos"`
`  PARAM "noLogfileMsg"  "1"`
`  PARAM "closeAfterRead"  "1"`
`  GROUP "roots"`
`  GROUP "rootPair"`
`  PARAM "root"  "event"`
`  GROUP_END`
`  GROUP_END`
`  GROUP_END`
	See **XML File Source** section (event, metrics, generic output)

`  GROUP_END` `GROUP_END`	Do not change

`  MAP "map1"`
`  DESCRIPTION ""`
`  INPUT "input1"`
`  FROM "source1" TO "target1"`
`  FROM "source2" TO "target2"`
`  MAP "map2"`
`  DESCRIPTION ""`
`  INPUT "input2"`
`  FROM "source3" TO "target3"`
`  FROM "source4" TO "target4"`
	Optional section
	
	See **Mappings** section

`DEFAULTMSG`	Do not change **Start of Default event attributes**

`SERVERLOGONLY "false"`
`SEVERITY "Critical"`
`TIMECREATED "Time created"`
`NODE IP 0.0.0.0 "Node"`
`APPLICATION "DefaultApplication"`
`MSGGRP "Category"`
`OBJECT "DefaultObject"`
`MSGTYPE "DefaultType"`
`SERVICE_NAME "DefaultHPOMServiceID"`
`MSGKEY "DefaultEventKey"`
`MSGKEYRELATION ACK "<*>" SEPARATORS "  "`
`CUSTOM "Description" "Description"`
`CUSTOM "SubCategory" "Subcategory"`
`CUSTOM "RelatedCiHint" "Related CI"`
`CUSTOM "SourceCiHint" "Source CI"`
`CUSTOM "SubCiHint" "Sub Component"`
`CUSTOM "EtiHint" "ETI"`
`CUSTOM "SourcedFromExternalId" "Source Event ID"`
`CUSTOM "SourcedFromExternalUrl" "https://example.com"`
`CUSTOM "CA_0" "1"`
`CUSTOM "NoDuplicateSuppression" "true"`
`TEXT "Title"`
`AUTOACTION "DefaultAACmd" ACTIONNODE IP 0.0.0.0 "<$MSG_NODE_NAME>" ANNOTATE ACK`
`SEND_MSG_AFTER_LOC_AA SEND_FAILED_MSG`
`USER "user1" PWD "%%ActionUserPassword%%"`
`SIGNATURE ""`
`OPACTION "DefaultOperCmd" ACTIONNODE IP 0.0.0.0 "<$OPC_MGMTSV>"`
`SIGNATURE ""`
`MPI_AGT_COPY_MSG`
`HELPTEXT "DefaultinstructionText"`
`HELP "4EC0F729-6037-A577-8AFF-11A422242A70"`
`SUPP_DUPL_IDENT`
`"0h30m0s" RESEND "8h0m0s"`
	See **Set Event Attributes** section
	
	- this part sets the default attributes of the generated events

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types:
Event generating rules, starting with `MSGCONDITIONS`
Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and
Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS`
There can be multiple blocks of each type.
Rules are processed from top to bottom until the first rule matches.

Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks.
Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks.
See **Condition** section.

MSGCONDITIONS	**Start of event generating Rules section**
DESCRIPTION "set all event attributes, default correlation, ca, hardcoded instruction, advanced, no action" CONDITION_ID "81582EC5-525B-8A4C-230B-673B873B7978" CONDITION "Severity" == "CRITICAL"	1. Rule **Rule section** Description, and condition of the first rule
SET	Do not change, **start of Rule Set section**
SERVERLOGONLY "true" SEVERITY "Critical" TIMECREATED "<$DATA:Timestamp>" NODE IP 0.0.0.0 "<$DATA:Node>" APPLICATION "<$DATA:application>" MSGGRP "Category" OBJECT "<$DATA:object>" MSGTYPE "<$DATA:type>" SERVICE_NAME "<$DATA:HPOM service ID>" MSGKEY "<$DATA:Node>:<$DATA:EventId>" MSGKEYRELATION ACK "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS " " CUSTOM "Description" "<$DATA:Description>" CUSTOM "SubCategory" "Subcategory" CUSTOM "RelatedCiHint" "<$DATA:RelatedCI>" CUSTOM "NoDuplicateSuppression" "false" CUSTOM "EtiHint" "ETIName:Value" CUSTOM "SourceCiHint" "<$DATA:SourceCI>" CUSTOM "SubCiHint" "Sub Component" CUSTOM "SourcedFromExternalId" "<$DATA:EventId>" CUSTOM "CA_1" "2" CUSTOM "SourcedFromExternalUrl" "https://<DATA:url>" TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"	See **Set Event Attributes** section - this part sets the non-default attributes of the event generated by the first rule
SUPPRESSCONDITIONS	Suppress on matched rule section
DESCRIPTION "suppress on matched rule" CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206" CONDITION "SomeProperty" == "SomeValue"	2. Rule

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`
	3. Rule

`MSGCONDITIONS`	**Start of another event generating Rules block**
`DESCRIPTION …`	4. Rule **Rule section**
`SET …`	See **Set Event Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

6. Syntax of Event from Perl Script policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Event from Structured Log File policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – can be changed in Policy Properties

`POLTYPE "perl"`
`  GROUP "ROOT"`
`  GROUP "content"`
	Do not change

`  GROUP "options"`
`  PARAM "isForwardUnMatchedMsgToSrv"  "1"`
`  PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"`
`  PARAM "fieldSep"  "&#x20;&#x09;"`
`  PARAM "caseSensitive"  "1"`
`  PARAM "isLogMatchedCond"  "1"`
`  PARAM "isLogMatchedSuppressCond"  "1"`
`  PARAM "isLogUnMatchedCond"  "1"`
`  GROUP_END`
	See **Event** Options section

`  GROUP "sources"`
`  PARAM "interval"  "5m"`
`  PARAM "perlscriptname"  "/"`
`  PARAM "perlscript"  "IyEvdXNyL2JIyEvdXNyL2Jpbi9wZXJsCnVzZSB"`
`  ATT "encoding"  "base64"`
`  PARAM "perlsubname"  "mysub"`
`  PARAM "perldatarray"  "myarray"`
`  GROUP "datafields"`
`  PARAM "dataField"  "mydatafield1"`
`  PARAM "dataField"  "mydatafield2"`
`  PARAM "dataField"  "mydatafield3"`
`  GROUP_END`
`  PARAM "sourceTag"  "SomeTag"`
`  GROUP "inputparameters"`
`  PARAM "parameter"  "myuser"`
`  ATT "index"  "0"`
`  PARAM "parameter"  "86A0037691E47881D5CF"`
`  ATT "encryption"  "ovcrypt"`
`  ATT "index"  "1"`
`  GROUP_END`
`  GROUP_END`
	See **Perl Script Source** section

`  GROUP_END` `  GROUP_END`	Do not change

`  MAP "map1"`
`  DESCRIPTION ""`
`  INPUT "input1"`
`  FROM "source1" TO "target1"`
`  FROM "source2" TO "target2"`
`  MAP "map2"`
`  DESCRIPTION ""`
`  INPUT "input2"`
`  FROM "source3" TO "target3"`
`  FROM "source4" TO "target4"`
	Optional section
	
	See **Mappings** section

`DEFAULTMSG`	Do not change **Start of Default event attributes**

`SERVERLOGONLY "false"`
`SEVERITY "Critical"`
`TIMECREATED "Time created"`
`NODE IP 0.0.0.0 "Node"`
`APPLICATION "DefaultApplication"`
`MSGGRP "Category"`
`OBJECT "DefaultObject"`
`MSGTYPE "DefaultType"`
`SERVICE_NAME "DefaultHPOMServiceID"`
`MSGKEY "DefaultEventKey"`
`MSGKEYRELATION ACK "<*>" SEPARATORS "  "`
`CUSTOM "Description" "Description"`
`CUSTOM "SubCategory" "Subcategory"`
`CUSTOM "RelatedCiHint" "Related CI"`
`CUSTOM "SourceCiHint" "Source CI"`
`CUSTOM "SubCiHint" "Sub Component"`
`CUSTOM "EtiHint" "ETI"`
`CUSTOM "SourcedFromExternalId" "Source Event ID"`
`CUSTOM "SourcedFromExternalUrl" "https://example.com"`
`CUSTOM "CA_0" "1"`
`CUSTOM "NoDuplicateSuppression" "true"`
`TEXT "Title"`
`AUTOACTION "DefaultAACmd" ACTIONNODE IP 0.0.0.0 "<$MSG_NODE_NAME>" ANNOTATE ACK`
`SEND_MSG_AFTER_LOC_AA SEND_FAILED_MSG`
`USER "user1" PWD "%%ActionUserPassword%%"`
`SIGNATURE ""`
`OPACTION "DefaultOperCmd" ACTIONNODE IP 0.0.0.0 "<$OPC_MGMTSV>"`
`SIGNATURE ""`
`MPI_AGT_COPY_MSG`
`HELPTEXT "DefaultinstructionText"`
`HELP "4EC0F729-6037-A577-8AFF-11A422242A70"`
`SUPP_DUPL_IDENT`
`"0h30m0s" RESEND "8h0m0s"`
	See **Set Event Attributes** section
	
	- this part sets the default attributes of the generated events

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types:
Event generating rules, starting with `MSGCONDITIONS`
Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and
Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS`
There can be multiple blocks of each type.
Rules are processed from top to bottom until the first rule matches.

Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks.
Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks.
See **Condition** section.

MSGCONDITIONS	**Start of event generating Rules section**
DESCRIPTION "set all event attributes, default correlation, ca, hardcoded instruction, advanced, no action" CONDITION_ID "81582EC5-525B-8A4C-230B-673B873B7978" CONDITION "Severity" == "CRITICAL"	1. Rule **Rule section** Description, and condition of the first rule
SET	Do not change, **start of Rule Set section**
SERVERLOGONLY "true" SEVERITY "Critical" TIMECREATED "<$DATA:Timestamp>" NODE IP 0.0.0.0 "<$DATA:Node>" APPLICATION "<$DATA:application>" MSGGRP "Category" OBJECT "<$DATA:object>" MSGTYPE "<$DATA:type>" SERVICE_NAME "<$DATA:HPOM service ID>" MSGKEY "<$DATA:Node>:<$DATA:EventId>" MSGKEYRELATION ACK "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS " " CUSTOM "Description" "<$DATA:Description>" CUSTOM "SubCategory" "Subcategory" CUSTOM "RelatedCiHint" "<$DATA:RelatedCI>" CUSTOM "NoDuplicateSuppression" "false" CUSTOM "EtiHint" "ETIName:Value" CUSTOM "SourceCiHint" "<$DATA:SourceCI>" CUSTOM "SubCiHint" "Sub Component" CUSTOM "SourcedFromExternalId" "<$DATA:EventId>" CUSTOM "CA_1" "2" CUSTOM "SourcedFromExternalUrl" "https://<DATA:url>" TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"	See **Set Event Attributes** section - this part sets the non-default attributes of the event generated by the first rule
SUPPRESSCONDITIONS	Suppress on matched rule section
DESCRIPTION "suppress on matched rule" CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206" CONDITION "SomeProperty" == "SomeValue"	2. Rule

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`
	3. Rule

`MSGCONDITIONS`	**Start of another event generating Rules block**
`DESCRIPTION …`	4. Rule **Rule section**
`SET …`	See **Set Event Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

7. Syntax of Event from DB policy template

Example policy template

```
SYNTAX_VERSION 13

GENERIC_SOURCE "Example Event from Database policy template"
DESCRIPTION "Example policy that uses many available features to
illustrate policy template syntax"
```

Do not change – can be changed in Policy Properties

```
POLTYPE "eventdb"
   GROUP "ROOT"
      GROUP "content"
```

Do not change

```
         GROUP "options"
            PARAM "isForwardUnMatchedMsgToSrv"  "1"
            PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"
            PARAM "fieldSep"  "&#x20;&#x09;"
            PARAM "caseSensitive"  "1"
            PARAM "isLogMatchedCond"  "1"
            PARAM "isLogMatchedSuppressCond"  "1"
            PARAM "isLogUnMatchedCond"  "1"
         GROUP_END
```

See **Event** Options section

```
         GROUP "sources"
            PARAM "interval"  "5m0s"
            GROUP "configuration"
               PARAM "jdbcclasspath"  "/opt/OV/java/sqljdbc42.jar"
               PARAM "jdbcdriver"
"com.microsoft.sqlserver.jdbc.SQLServerDriver"
               PARAM "connectstring"
"jdbc:sqlserver://dbserver.example.com:1433;Database=Event;"
               PARAM "username"  "sa"
               PARAM "password"  "%%DBUSERPASSWORD%%"
               ATT "encryption"  "ovcrypt"
               PARAM "sqlinit"  "SELECT MAX(time_received) as timestamp
FROM event.dbo.all_events"
               PARAM "sql"  "SELECT (SELECT COUNT(*) FROM
event.dbo.all_events WHERE state =
                     'OPEN') AS openstate,
                       (SELECT count(*) FROM event.dbo.all_events
                       WHERE state = 'IN_PROGRESS')
                       AS in_progress,
                       (SELECT count(*) FROM event.dbo.all_events
            WHERE state = 'RESOLVED') AS
                       resolved,
                       (SELECT count(*) FROM event.dbo.all_events
            WHERE state = 'CLOSED' AND
                       time_received > '<$DATA:timestamp>') AS closed,
                       'OMi Prod' AS OMi"
               PARAM "resultsetsize"  "100"
               PARAM "fetchsize"  "100"
               GROUP "connectproperties"
                  GROUP "field"
                     PARAM "name"  "SomeProperty"
                     PARAM "value"  "someValue"
                  GROUP_END
               GROUP_END
            GROUP_END
            GROUP "data"
```

See the **DB Source** section

<pre>GROUP "field"
PARAM "name" "timestamp"
PARAM "value" "2020-09-01 10:00:00"
GROUP_END
GROUP_END
PARAM "sourceTag" "SomeTag"
GROUP_END</pre>	

<pre>GROUP_END
GROUP_END</pre>
	Do not change

<pre>MAP "map1"
DESCRIPTION ""
INPUT "input1"
FROM "source1" TO "target1"
FROM "source2" TO "target2"
MAP "map2"
DESCRIPTION ""
INPUT "input2"
FROM "source3" TO "target3"
FROM "source4" TO "target4"</pre>
	Optional section
	
	See **Mappings** section

<pre>DEFAULTMSG</pre>	Do not change **Start of Default event attributes**

<pre>SERVERLOGONLY "false"
SEVERITY "Critical"
TIMECREATED "Time created"
NODE IP 0.0.0.0 "Node"
APPLICATION "DefaultApplication"
MSGGRP "Category"
OBJECT "DefaultObject"
MSGTYPE "DefaultType"
SERVICE_NAME "DefaultHPOMServiceID"
MSGKEY "DefaultEventKey"
MSGKEYRELATION ACK "<*>" SEPARATORS " "
CUSTOM "Description" "Description"
CUSTOM "SubCategory" "Subcategory"
CUSTOM "RelatedCiHint" "Related CI"
CUSTOM "SourceCiHint" "Source CI"
CUSTOM "SubCiHint" "Sub Component"
CUSTOM "EtiHint" "ETI"
CUSTOM "SourcedFromExternalId" "Source Event ID"
CUSTOM "SourcedFromExternalUrl" "https://example.com"
CUSTOM "CA_0" "1"
CUSTOM "NoDuplicateSuppression" "true"
TEXT "Title"
AUTOACTION "DefaultAACmd" ACTIONNODE IP 0.0.0.0
"<$MSG_NODE_NAME>" ANNOTATE ACK
SEND_MSG_AFTER_LOC_AA SEND_FAILED_MSG
USER "user1" PWD "%%ActionUserPassword%%"
SIGNATURE ""
OPACTION "DefaultOperCmd" ACTIONNODE IP 0.0.0.0
"<$OPC_MGMTSV>"
SIGNATURE ""
MPI_AGT_COPY_MSG
HELPTEXT "DefaultinstructionText"
HELP "4EC0F729-6037-A577-8AFF-11A422242A70"
SUPP_DUPL_IDENT
"0h30m0s" RESEND "8h0m0s"</pre>
	See **Set Event Attributes** section
	
	- this part sets the default attributes of the generated events

Following the DEFAULTMSG section, there can be a mix of rules of 3 types:
Event generating rules, starting with MSGCONDITIONS
Suppress on matched rules, starting with SUPPRESSCONDITIONS and
Suppress on unmatched rules, starting with SUPP_UNM_CONDITIONS
There can be multiple blocks of each type.
Rules are processed from top to bottom until the first rule matches.

Event generating rules consist of DESCRIPTION, CONDITION and SET blocks.
Suppress rules consist of DESCRIPTION and CONDITION blocks.
See **Condition** section.

MSGCONDITIONS	**Start of event generating Rules section**
DESCRIPTION "set all event attributes, default correlation, ca, hardcoded instruction, advanced, no action" CONDITION_ID "81582EC5-525B-8A4C-230B-673B873B7978" CONDITION "Severity" == "CRITICAL"	1. Rule **Rule section** Description, and condition of the first rule
SET	Do not change, **start of Rule Set section**
SERVERLOGONLY "true" SEVERITY "Critical" TIMECREATED "<$DATA:Timestamp>" NODE IP 0.0.0.0 "<$DATA:Node>" APPLICATION "<$DATA:application>" MSGGRP "Category" OBJECT "<$DATA:object>" MSGTYPE "<$DATA:type>" SERVICE_NAME "<$DATA:HPOM service ID>" MSGKEY "<$DATA:Node>:<$DATA:EventId>" MSGKEYRELATION ACK "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS " " CUSTOM "Description" "<$DATA:Description>" CUSTOM "SubCategory" "Subcategory" CUSTOM "RelatedCiHint" "<$DATA:RelatedCI>" CUSTOM "NoDuplicateSuppression" "false" CUSTOM "EtiHint" "ETIName:Value" CUSTOM "SourceCiHint" "<$DATA:SourceCI>" CUSTOM "SubCiHint" "Sub Component" CUSTOM "SourcedFromExternalId" "<$DATA:EventId>" CUSTOM "CA_1" "2" CUSTOM "SourcedFromExternalUrl" "https://<DATA:url>" TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"	See **Set Event Attributes** section - this part sets the non-default attributes of the event generated by the first rule
SUPPRESSCONDITIONS	Suppress on matched rule section

DESCRIPTION "suppress on matched rule" CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206" CONDITION "SomeProperty" == "SomeValue"	2. Rule
SUPP_UNM_CONDITIONS	Suppress on unmatched rule section
DESCRIPTION "suppress on unmatched rule" CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC" CONDITION "SomeProperty" ~= "<1 -gt <#> -gt 50>"	3. Rule
MSGCONDITIONS	**Start of another event generating Rules block**
DESCRIPTION …	4. Rule **Rule section**
SET …	See **Set Event Attributes** section
Add as many rules as required in any sequence, using MSGCONDITIONS, SUPPRESSCONDITIONS and SUPP_UNM_CONDITIONS blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

8. Syntax of Event from REST Web Service Listener policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Event from REST Web Service Listener policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – can be changed in Policy Properties

`POLTYPE "xml-ws"`
`  GROUP "ROOT"`
`  GROUP "content"`
	Do not change

`  GROUP "options"`
`  PARAM "isForwardUnMatchedMsgToSrv"  "1"`
`  PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"`
`  PARAM "fieldSep"  "&#x20;&#x09;"`
`  PARAM "caseSensitive"  "1"`
`  PARAM "isLogMatchedCond"  "1"`
`  PARAM "isLogMatchedSuppressCond"  "1"`
`  PARAM "isLogUnMatchedCond"  "1"`
`  GROUP_END`
	See **Event** Options section

`  GROUP "sources"`
`  PARAM "logpath"  "/receiver"`
`  PARAM "chSet"  "69"`
`  GROUP "roots"`
`  GROUP "rootPair"`
`  PARAM "root"  "someXMLtag"`
`  GROUP_END`
`  GROUP_END`
`  GROUP_END`
	See
	
	REST Web Service Listener Source section

`  GROUP_END` `GROUP_END`	Do not change

`  MAP "map1"`
`  DESCRIPTION ""`
`  INPUT "input1"`
`  FROM "source1" TO "target1"`
`  FROM "source2" TO "target2"`
`  MAP "map2"`
`  DESCRIPTION ""`
`  INPUT "input2"`
`  FROM "source3" TO "target3"`
`  FROM "source4" TO "target4"`
	Optional section
	
	See **Mappings** section

`DEFAULTMSG`	Do not change **Start of Default event attributes**

`SERVERLOGONLY "false"`
`SEVERITY "Critical"`
`TIMECREATED "Time created"`
`NODE IP 0.0.0.0 "Node"`
`APPLICATION "DefaultApplication"`
`MSGGRP "Category"`
`OBJECT "DefaultObject"`
`MSGTYPE "DefaultType"`
`SERVICE_NAME "DefaultHPOMServiceID"`
`MSGKEY "DefaultEventKey"`
`MSGKEYRELATION ACK "<*>" SEPARATORS "  "`
`CUSTOM "Description" "Description"`
`CUSTOM "SubCategory" "Subcategory"`
`CUSTOM "RelatedCiHint" "Related CI"`
`CUSTOM "SourceCiHint" "Source CI"`
`CUSTOM "SubCiHint" "Sub Component"`
`CUSTOM "EtiHint" "ETI"`
`CUSTOM "SourcedFromExternalId" "Source Event ID"`
`CUSTOM "SourcedFromExternalUrl" "https://example.com"`
`CUSTOM "CA_0" "1"`
`CUSTOM "NoDuplicateSuppression" "true"`
`TEXT "Title"`
`AUTOACTION "DefaultAACmd" ACTIONNODE IP 0.0.0.0 "<$MSG_NODE_NAME>" ANNOTATE ACK`
`SEND_MSG_AFTER_LOC_AA SEND_FAILED_MSG`
`USER "user1" PWD "%%ActionUserPassword%%"`
`SIGNATURE ""`
`OPACTION "DefaultOperCmd" ACTIONNODE IP 0.0.0.0 "<$OPC_MGMTSV>"`
`SIGNATURE ""`
`MPI_AGT_COPY_MSG`
`HELPTEXT "DefaultinstructionText"`
`HELP "4EC0F729-6037-A577-8AFF-11A422242A70"`
`SUPP_DUPL_IDENT`
`"0h30m0s" RESEND "8h0m0s"`
	See **Set Event Attributes** section
	
	- this part sets the default attributes of the generated events

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types:
Event generating rules, starting with `MSGCONDITIONS`
Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and
Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS`
There can be multiple blocks of each type.
Rules are processed from top to bottom until the first rule matches.

Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks.
Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks.
See **Condition** section.

MSGCONDITIONS	**Start of event generating Rules section**
DESCRIPTION "set all event attributes, default correlation, ca, hardcoded instruction, advanced, no action" CONDITION_ID "81582EC5-525B-8A4C-230B-673B873B7978" CONDITION "Severity" == "CRITICAL"	1. Rule **Rule section** Description, and condition of the first rule
SET	Do not change, **start of Rule Set section**
SERVERLOGONLY "true" SEVERITY "Critical" TIMECREATED "<$DATA:Timestamp>" NODE IP 0.0.0.0 "<$DATA:Node>" APPLICATION "<$DATA:application>" MSGGRP "Category" OBJECT "<$DATA:object>" MSGTYPE "<$DATA:type>" SERVICE_NAME "<$DATA:HPOM service ID>" MSGKEY "<$DATA:Node>:<$DATA:EventId>" MSGKEYRELATION ACK "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS " " CUSTOM "Description" "<$DATA:Description>" CUSTOM "SubCategory" "Subcategory" CUSTOM "RelatedCiHint" "<$DATA:RelatedCI>" CUSTOM "NoDuplicateSuppression" "false" CUSTOM "EtiHint" "ETIName:Value" CUSTOM "SourceCiHint" "<$DATA:SourceCI>" CUSTOM "SubCiHint" "Sub Component" CUSTOM "SourcedFromExternalId" "<$DATA:EventId>" CUSTOM "CA_1" "2" CUSTOM "SourcedFromExternalUrl" "https://<DATA:url>" TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"	See **Set Event Attributes** section - this part sets the non-default attributes of the event generated by the first rule
SUPPRESSCONDITIONS	Suppress on matched rule section
DESCRIPTION "suppress on matched rule" CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206" CONDITION "SomeProperty" == "SomeValue"	2. Rule

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`
	3. Rule

`MSGCONDITIONS`	**Start of another event generating Rules block**
`DESCRIPTION …`	4. Rule **Rule section**
`SET …`	See **Set Event Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

9. Syntax of Metric from Structured Log File policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Metric from Structured Log File policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax."`
	Do not change – can be changed in Policy Properties

`POLTYPE "metricstruct"`
`  GROUP "ROOT"`
`  GROUP "content"`
	Do not change

`  GROUP "options"`
`  PARAM "isForwardUnMatchedMsgToSrv"  "1"`
`  PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"`
`  PARAM "fieldSep"  "&#x20;&#x09;"`
`  PARAM "caseSensitive"  "1"`
`  PARAM "isLogMatchedCond"  "1"`
`  PARAM "isLogMatchedSuppressCond"  "1"`
`  PARAM "isLogUnMatchedCond"  "1"`
`  GROUP_END`
	See **Metric Options** section

`  GROUP "sources"`
`  PARAM "logpath"  "/path/to/logfile"`
`  PARAM "interval"  "5m"`
`  PARAM "chSet"  "69"`
`  PARAM "readMode"  "fromLastPos"`
`  PARAM "noLogfileMsg"  "1"`
`  PARAM "closeAfterRead"  "1"`
`  PARAM "dataFieldSeparator"  ","`
`  PARAM "staticDataFields"  "time"`
`  PARAM "recurringDataFields"  "errorid, text"`
`  GROUP_END`
	See **Structured Log File Source** section

`  GROUP_END` `GROUP_END`	Do not change

`  MAP "map1"`
`  DESCRIPTION ""`
`  INPUT "input1"`
`  FROM "source1" TO "target1"`
`  FROM "source2" TO "target2"`
`  MAP "map2"`
`  DESCRIPTION ""`
`  INPUT "input2"`
`  FROM "source3" TO "target3"`
`  FROM "source4" TO "target4"`
	Optional section
	
	See **Mappings** section

`DEFAULTMSG`	Do not change **Start of Default metric attributes**

`SERVERLOGONLY "false"`
`CUSTOM "dataDomain" "domain"`
`CUSTOM "relatedCi" "related CI"`
`CUSTOM "metricClass" "class"`
`CUSTOM "name" "name"`
`CUSTOM "originalName" "orig name"`
`CUSTOM "sampleValue" "value"`
`CUSTOM "unit" "unit"`
`CUSTOM "node" "node"`
`CUSTOM "timeMeasured" "time"`
`CUSTOM "integrationId" "id"`
	See **Set Metric Attributes** section
	
	- this part sets the default metric values

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types: Event generating rules, starting with `MSGCONDITIONS` Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS` There can be multiple blocks of each type. Rules are processed from top to bottom until the first rule matches. Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks. Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks. See **Condition** section.	
`SUPPRESSCONDITIONS`	Suppress on matched rule section

`DESCRIPTION "suppress on matched rule"`
`CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206"`
`CONDITION`
`"SomeProperty" == "SomeValue"`	

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`	

`MSGCONDITIONS`	**Start of metric generating Rules section**

`DESCRIPTION "first rule"`
`CONDITION_ID "E559EB8A-D785-8A34-1BCC-11B572603A87"`
`CONDITION`
`"SomeProperty" == "SomeValue"`
	1. Rule
	Rule section
	
	ID, description, and condition of the first rule

`SET`	Do not change, **start of Rule Set section**

`CUSTOM "relatedCi" "related CI"`
`CUSTOM "metricClass" "metricclass"`
`CUSTOM "name" "metricname"`
`CUSTOM "originalName" "orig metric name"`
`CUSTOM "sampleValue" "value"`
`CUSTOM "unit" "unit"`
`CUSTOM "node" "node"`
`CUSTOM "timeMeasured" "time"`
`CUSTOM "integrationId" "id"`
	See **Set Metric Attributes** section
	
	- this part sets the non-default attributes of the metric generated by the first rule

`DESCRIPTION …`	2. Rule **Rule section**
`SET …`	See **Set Metric Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

10. Syntax of Metric from XML File policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Metric from XML File policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – can be changed in Policy Properties

`POLTYPE "metricxml"`
`  GROUP "ROOT"`
`  GROUP "content"`
	Do not change

`  GROUP "options"`
`  PARAM "isForwardUnMatchedMsgToSrv"  "1"`
`  PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"`
`  PARAM "fieldSep"  "&#x20;&#x09;"`
`  PARAM "caseSensitive"  "1"`
`  PARAM "isLogMatchedCond"  "1"`
`  PARAM "isLogMatchedSuppressCond"  "1"`
`  PARAM "isLogUnMatchedCond"  "1"`
`  GROUP_END`
	See **Metric Options** section

`GROUP "sources"`
`  PARAM "logpath"  "/path/logfile.xml"`
`  PARAM "interval"  "5m"`
`  PARAM "chSet"  "69"`
`  PARAM "readMode"  "fromLastPos"`
`  PARAM "noLogfileMsg"  "1"`
`  PARAM "closeAfterRead"  "1"`
`  GROUP "roots"`
`  GROUP "rootPair"`
`  PARAM "root"  "someXMLtag"`
`  GROUP_END`
`  GROUP_END`
`  GROUP_END`
	See **XML File Source section** (event, metrics, generic output)

`  GROUP_END` `GROUP_END`	Do not change

`  MAP "map1"`
`  DESCRIPTION ""`
`  INPUT "input1"`
`  FROM "source1" TO "target1"`
`  FROM "source2" TO "target2"`
`  MAP "map2"`
`  DESCRIPTION ""`
`  INPUT "input2"`
`  FROM "source3" TO "target3"`
`  FROM "source4" TO "target4"`
	Optional section
	
	See **Mappings** section

`DEFAULTMSG`	Do not change **Start of Default metric attributes**

`SERVERLOGONLY "false"`
`CUSTOM "dataDomain" "domain"`
`CUSTOM "relatedCi" "related CI"`
`CUSTOM "metricClass" "class"`
`CUSTOM "name" "name"`
`CUSTOM "originalName" "orig name"`
`CUSTOM "sampleValue" "value"`
`CUSTOM "unit" "unit"`
`CUSTOM "node" "node"`
`CUSTOM "timeMeasured" "time"`
`CUSTOM "integrationId" "id"`
	See **Set Metric Attributes** section
	
	- this part sets the default metric values

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types: Event generating rules, starting with `MSGCONDITIONS` Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS` There can be multiple blocks of each type. Rules are processed from top to bottom until the first rule matches. Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks. Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks. See **Condition** section.	
`SUPPRESSCONDITIONS`	Suppress on matched rule section

`DESCRIPTION "suppress on matched rule"`
`CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206"`
`CONDITION`
`"SomeProperty" == "SomeValue"`
	1.Rule

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`
	2.Rule

`MSGCONDITIONS`	**Start of metric generating Rules section**

`DESCRIPTION "third rule"`
`CONDITION_ID "E559EB8A-D785-8A34-1BCC-11B572603A87"`
`CONDITION`
`"SomeProperty" == "SomeValue"`
	3. Rule
	Rule section
	ID, description, and condition of the first rule

`SET`	Do not change, **start of Rule Set section**

`CUSTOM "relatedCi" "related CI"`
`CUSTOM "metricClass" "metricclass"`
`CUSTOM "name" "metricname"`
`CUSTOM "originalName" "orig metric name"`
`CUSTOM "sampleValue" "value"`
`CUSTOM "unit" "unit"`
`CUSTOM "node" "node"`
`CUSTOM "timeMeasured" "time"`
`CUSTOM "integrationId" "id"`
	See **Set Metric Attributes** section
	
	- this part sets the non-default attributes of the metric generated by the first rule

`DESCRIPTION …`	4. Rule **Rule section**
`SET …`	See **Set Metric Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

11. Syntax of Metric from Perl Script policy template

Example policy template

```
SYNTAX_VERSION 13

GENERIC_SOURCE "Example Metric from Perl script policy template"
DESCRIPTION "Example policy that uses many available features to
illustrate policy template syntax."
```

Do not change – can be changed in Policy Properties

```
POLTYPE "metricperl"
   GROUP "ROOT"
      GROUP "content"
```

Do not change

```
       GROUP "options"
            PARAM "isForwardUnMatchedMsgToSrv"  "1"
            PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"
            PARAM "fieldSep"  "&#x20;&#x09;"
            PARAM "caseSensitive"  "1"
            PARAM "isLogMatchedCond"  "1"
            PARAM "isLogMatchedSuppressCond"  "1"
            PARAM "isLogUnMatchedCond"  "1"
       GROUP_END
```

See **Metric Options** section

```
         GROUP "sources"
            PARAM "interval"  "5m"
            PARAM "perlscriptname"  "/"
            PARAM "perlscript"  "IyEvdXNyL2JIyEvdXNyL2Jpbi9wZXJsCnVzZSB"
            ATT "encoding"  "base64"
            PARAM "perlsubname"  "mysub"
            PARAM "perldatarray"  "myarray"
            GROUP "datafields"
               PARAM "dataField"  "mydatafield1"
               PARAM "dataField"  "mydatafield2"
               PARAM "dataField"  "mydatafield3"
            GROUP_END
            PARAM "sourceTag"  "SomeTag"
            GROUP "inputparameters"
               PARAM "parameter"  "myuser"
               ATT "index"  "0"
               PARAM "parameter"  "86A0037691E47881D5CF"
               ATT "encryption"  "ovcrypt"
               ATT "index"  "1"
            GROUP_END
         GROUP_END
```

See **Perl Script Source** section

```
      GROUP_END
   GROUP_END
```

Do not change

```
       MAP "map1"
       DESCRIPTION ""
       INPUT "input1"
         FROM "source1" TO "target1"
         FROM "source2" TO "target2"
       MAP "map2"
       DESCRIPTION ""
       INPUT "input2"
         FROM "source3" TO "target3"
         FROM "source4" TO "target4"
```

Optional section

See Mappings section

`DEFAULTMSG`	Do not change **Start of Default metric attributes**

`SERVERLOGONLY "false"`
`CUSTOM "dataDomain" "domain"`
`CUSTOM "relatedCi" "related CI"`
`CUSTOM "metricClass" "class"`
`CUSTOM "name" "name"`
`CUSTOM "originalName" "orig name"`
`CUSTOM "sampleValue" "value"`
`CUSTOM "unit" "unit"`
`CUSTOM "node" "node"`
`CUSTOM "timeMeasured" "time"`
`CUSTOM "integrationId" "id"`
	See **Set Metric Attributes** section
	
	- this part sets the default metric values

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types: Event generating rules, starting with `MSGCONDITIONS` Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS` There can be multiple blocks of each type. Rules are processed from top to bottom until the first rule matches. Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks. Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks. See **Condition** section.	
`SUPPRESSCONDITIONS`	Suppress on matched rule section

`DESCRIPTION "suppress on matched rule"`
`CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206"`
`CONDITION`
`"SomeProperty" == "SomeValue"`	

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`	

`MSGCONDITIONS`	**Start of metric generating Rules section**

`DESCRIPTION "first rule"`
`CONDITION_ID "E559EB8A-D785-8A34-1BCC-11B572603A87"`
`CONDITION`
`"SomeProperty" == "SomeValue"`
	1. Rule
	Rule section
	
	ID, description, and condition of the first rule

`SET`	Do not change, **start of Rule Set section**

`CUSTOM "relatedCi" "related CI"`
`CUSTOM "metricClass" "metricclass"`
`CUSTOM "name" "metricname"`
`CUSTOM "originalName" "orig metric name"`
`CUSTOM "sampleValue" "value"`
`CUSTOM "unit" "unit"`
`CUSTOM "node" "node"`
`CUSTOM "timeMeasured" "time"`
`CUSTOM "integrationId" "id"`
	See **Set Metric Attributes** section
	
	- this part sets the non-default attributes of the metric generated by the first rule

`DESCRIPTION …`	2. Rule **Rule section**
`SET …`	See **Set Metric Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

12. Syntax of Metric from DB policy template

Example policy template

```
SYNTAX_VERSION 13

GENERIC_SOURCE "Example Metric from Database policy template"
DESCRIPTION "Example policy that uses many available features to illustrate
policy template syntax."
```

Do not change – can be changed in Policy Properties

```
POLTYPE "metricdb"
   GROUP "ROOT"
      GROUP "content"
```

Do not change

```
       GROUP "options"
            PARAM "isForwardUnMatchedMsgToSrv"  "1"
            PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"
            PARAM "fieldSep"  "&#x20;&#x09;"
            PARAM "caseSensitive"  "1"
            PARAM "isLogMatchedCond"  "1"
            PARAM "isLogMatchedSuppressCond"  "1"
            PARAM "isLogUnMatchedCond"  "1"
         GROUP_END
```

See **Metric Options** section

```
       GROUP "sources"
            PARAM "interval"  "5m0s"
            GROUP "configuration"
               PARAM "jdbcclasspath"  "/opt/OV/java/sqljdbc42.jar"
               PARAM "jdbcdriver"
"com.microsoft.sqlserver.jdbc.SQLServerDriver"
               PARAM "connectstring"
"jdbc:sqlserver://dbserver.example.com:1433;Database=Event;"
               PARAM "username"  "sa"
               PARAM "password"  "%%DBUSERPASSWORD%%"
               ATT "encryption"  "ovcrypt"
               PARAM "sqlinit"  "SELECT MAX(time_received) as timestamp
FROM event.dbo.all_events"
               PARAM "sql"  "SELECT (SELECT COUNT(*) FROM
event.dbo.all_events WHERE state =
                     'OPEN') AS openstate,
                        (SELECT count(*) FROM event.dbo.all_events WHERE
                        state = 'IN_PROGRESS')
                        AS in_progress,
                        (SELECT count(*) FROM event.dbo.all_events WHERE
            state = 'RESOLVED') AS
                        resolved,
                        (SELECT count(*) FROM event.dbo.all_events WHERE
            state = 'CLOSED' AND
                        time_received > '<$DATA:timestamp>') AS closed,
                        'OMi Prod' AS OMi"
               PARAM "resultsetsize"  "100"
               PARAM "fetchsize"  "100"
               GROUP "connectproperties"
                  GROUP "field"
                     PARAM "name"  "SomeProperty"
                     PARAM "value"  "someValue"
                  GROUP_END
               GROUP_END
            GROUP_END
```

See

DB Source section

<pre>GROUP "data"
GROUP "field"
PARAM "name" "timestamp"
PARAM "value" "2020-09-01 10:00:00"
GROUP_END
GROUP_END
PARAM "sourceTag" "SomeTag"
GROUP_END</pre>	

<pre>GROUP_END
GROUP_END</pre>
	Do not change

<pre>MAP "map1"
DESCRIPTION ""
INPUT "input1"
FROM "source1" TO "target1"
FROM "source2" TO "target2"
MAP "map2"
DESCRIPTION ""
INPUT "input2"
FROM "source3" TO "target3"
FROM "source4" TO "target4"</pre>
	Optional section
	
	See **Mappings** section

<pre>DEFAULTMSG</pre>	Do not change **Start of Default metric attributes**

<pre>SERVERLOGONLY "false"
CUSTOM "dataDomain" "domain"
CUSTOM "relatedCi" "related CI"
CUSTOM "metricClass" "class"
CUSTOM "name" "name"
CUSTOM "originalName" "orig name"
CUSTOM "sampleValue" "value"
CUSTOM "unit" "unit"
CUSTOM "node" "node"
CUSTOM "timeMeasured" "time"
CUSTOM "integrationId" "id"</pre>
	See **Set Metric Attributes** section
	
	- this part sets the default metric values

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types: Event generating rules, starting with `MSGCONDITIONS` Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS` There can be multiple blocks of each type. Rules are processed from top to bottom until the first rule matches. Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks. Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks. See **Condition** section.	
<pre>SUPPRESSCONDITIONS</pre>	Suppress on

	matched rule section
DESCRIPTION "suppress on matched rule" CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206" CONDITION "SomeProperty" == "SomeValue"	
SUPP_UNM_CONDITIONS	Suppress on unmatched rule section
DESCRIPTION "suppress on unmatched rule" CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC" CONDITION "SomeProperty" ~= "<1 -gt <#> -gt 50>"	
MSGCONDITIONS	**Start of metric generating Rules section**
DESCRIPTION "first rule" CONDITION_ID "E559EB8A-D785-8A34-1BCC-11B572603A87" CONDITION "SomeProperty" == "SomeValue"	1. Rule **Rule section** ID, description, and condition of the first rule
SET	Do not change, **start of Rule Set section**
CUSTOM "relatedCi" "related CI" CUSTOM "metricClass" "metricclass" CUSTOM "name" "metricname" CUSTOM "originalName" "orig metric name" CUSTOM "sampleValue" "value" CUSTOM "unit" "unit" CUSTOM "node" "node" CUSTOM "timeMeasured" "time" CUSTOM "integrationId" "id"	See **Set Metric Attributes** section - this part sets the non-default attributes of the metric generated

	by the first rule
`DESCRIPTION …`	2. Rule **Rule section**
`SET …`	See **Set Metric Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

13. Syntax of Metric from REST Web Service Listener policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Metric from REST Web Service Listener policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – can be changed in Policy Properties

`POLTYPE "metricwsxml"`
`GROUP "ROOT"`
`GROUP "content"`
	Do not change

`GROUP "options"`
`PARAM "isForwardUnMatchedMsgToSrv" "1"`
`PARAM "isLogOnlyUnMatchedMsgToSrv" "0"`
`PARAM "fieldSep" "&#x20;&#x09;"`
`PARAM "caseSensitive" "1"`
`PARAM "isLogMatchedCond" "1"`
`PARAM "isLogMatchedSuppressCond" "1"`
`PARAM "isLogUnMatchedCond" "1"`
`GROUP_END`
	See **Metric Options** section

`GROUP "sources"`
`PARAM "logpath" "/receiver"`
`PARAM "chSet" "69"`
`GROUP "roots"`
`GROUP "rootPair"`
`PARAM "root" "someXMLtag"`
`GROUP_END`
`GROUP_END`
`GROUP_END`
	See **REST Web Service** Listener Source section

`GROUP_END` `GROUP_END`	Do not change

`MAP "map1"`
`DESCRIPTION ""`
`INPUT "input1"`
`FROM "source1" TO "target1"`
`FROM "source2" TO "target2"`
`MAP "map2"`
`DESCRIPTION ""`
`INPUT "input2"`
`FROM "source3" TO "target3"`
`FROM "source4" TO "target4"`
	Optional section
	
	See **Mappings** section

`DEFAULTMSG`	Do not change **Start of Default metric attributes**

`SERVERLOGONLY "false"`
`CUSTOM "dataDomain" "domain"`
`CUSTOM "relatedCi" "related CI"`
`CUSTOM "metricClass" "class"`
`CUSTOM "name" "name"`
`CUSTOM "originalName" "orig name"`
`CUSTOM "sampleValue" "value"`
`CUSTOM "unit" "unit"`
`CUSTOM "node" "node"`
`CUSTOM "timeMeasured" "time"`
`CUSTOM "integrationId" "id"`
	See **Set Metric Attributes** section
	
	- this part sets the default metric values

Following the `DEFAULTMSG` section, there can be a mix of rules of 3 types: Event generating rules, starting with `MSGCONDITIONS` Suppress on matched rules, starting with `SUPPRESSCONDITIONS` and Suppress on unmatched rules, starting with `SUPP_UNM_CONDITIONS` There can be multiple blocks of each type. Rules are processed from top to bottom until the first rule matches. Event generating rules consist of `DESCRIPTION`, `CONDITION` and `SET` blocks. Suppress rules consist of `DESCRIPTION` and `CONDITION` blocks. See **Condition** section.	
`SUPPRESSCONDITIONS`	Suppress on matched rule section

`DESCRIPTION "suppress on matched rule"`
`CONDITION_ID "C19E4CAF-376A-4635-4A19-593A42DC5206"`
`CONDITION`
`"SomeProperty" == "SomeValue"`
	1.Rule

`SUPP_UNM_CONDITIONS`	Suppress on unmatched rule section

`DESCRIPTION "suppress on unmatched rule"`
`CONDITION_ID "52E58B05-C497-0436-CFD6-593B1ACDDFDC"`
`CONDITION`
`"SomeProperty" ~= "<1 -gt <#> -gt 50>"`
	2.Rule

`MSGCONDITIONS`	**Start of metric generating Rules section**

`DESCRIPTION "third rule"`
`CONDITION_ID "E559EB8A-D785-8A34-1BCC-11B572603A87"`
`CONDITION`
`"SomeProperty" == "SomeValue"`
	3. Rule
	Rule section
	
	ID, description, and condition of the first rule

`SET`	Do not change, **start of Rule Set section**

`CUSTOM "relatedCi" "related CI"`
`CUSTOM "metricClass" "metricclass"`
`CUSTOM "name" "metricname"`
`CUSTOM "originalName" "orig metric name"`
`CUSTOM "sampleValue" "value"`
`CUSTOM "unit" "unit"`
`CUSTOM "node" "node"`
`CUSTOM "timeMeasured" "time"`
`CUSTOM "integrationId" "id"`
	See **Set Metric Attributes** section
	
	- this part sets the non-default attributes of the metric generated by the first rule

`DESCRIPTION …`	4. Rule **Rule section**
`SET …`	Rule set and **Set Metric Attributes** section
Add as many rules as required in any sequence, using `MSGCONDITIONS`, `SUPPRESSCONDITIONS` and `SUPP_UNM_CONDITIONS` blocks. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
	End of policy

14. Syntax of Generic Output from XML File policy template

Example policy template

```
SYNTAX_VERSION 13

GENERIC_SOURCE "Example Generic Output from XML File policy template"
DESCRIPTION "Example policy that uses many available features to
illustrate policy template syntax"
```

Do not change – change in Policy Properties

```
POLTYPE "genoutxml"
   GROUP "ROOT"
      GROUP "content"
         GROUP "options"
         GROUP_END
```

Do not change

```
         GROUP "sources"
            GROUP "roots"
               GROUP "rootPair"
                  PARAM "root"  "someXMLtag"
               GROUP_END
            GROUP_END
            PARAM "logpath"  "/path/logile.xml"
            PARAM "interval"  "5m"
            PARAM "chSet"  "69"
            PARAM "readMode"  "fromLastPos"
            PARAM "noLogfileMsg"  "1"
            PARAM "closeAfterRead"  "1"
```

See **XML File Source** section (event, metrics, generic output)

```
            GROUP "KEYMAPRULE"
               PARAM "KEEP_INPUT_FIELDS"  "1"
               GROUP "KEYFIELD_MAPPINGS"
                  PARAM "KF_MAP"  "KEEP"
                  ATT "KEY_IN"  "someFiieldinXML"
                  ATT "KEY_OUT"  "someOutputFileld"
                  PARAM "KF_MAP"  "KEEP"
                  ATT "KEY_IN"  "someOtherFieldinInput"
                  ATT "KEY_OUT"  "someOtherOutputField"
               GROUP_END
               GROUP "ADDITIONAL_FIELDS"
                  PARAM "ADD_FIELD"  "ADD"
                  ATT "NAME"  "anAdditionalField"
                  ATT "VALUE"  "SomeFieldinInput, SomeOtherFieldinInput"
               GROUP_END
            GROUP_END
```

See **Schema section (Generic Output)**

```
         GROUP_END
      GROUP_END
   GROUP_END
```

Do not change

```
      MAP "map1"
      DESCRIPTION ""
      INPUT "input1"
        FROM "source1" TO "target1"
        FROM "source2" TO "target2"
      MAP "map2"
      DESCRIPTION ""
      INPUT "input2"
        FROM "source3" TO "target3"
        FROM "source4" TO "target4"
```

Optional section

See Mappings section

DEFAULTMSG	Do not change End of Policy

15. Syntax of Generic Output from Perl Script policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Generic Output from Perl Script policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – change in Policy Properties

`POLTYPE "genoutperl"`
`GROUP "ROOT"`
`GROUP "content"`
`GROUP "options"`
`GROUP_END`
	Do not change

`GROUP "sources"`
`PARAM "interval" "5m"`
`PARAM "perlscriptname" "/"`
`PARAM "perlscript" "IyEvdXNyL2JIyEvdXNyL2Jpbi9wZXJsCnVzZSB"`
`ATT "encoding" "base64"`
`PARAM "perlsubname" "mysub"`
`PARAM "perldatarray" "myarray"`
`GROUP "datafields"`
`PARAM "dataField" "mydatafield1"`
`PARAM "dataField" "mydatafield2"`
`PARAM "dataField" "mydatafield3"`
`GROUP_END`
`PARAM "sourceTag" "SomeTag"`
`GROUP "inputparameters"`
`PARAM "parameter" "myuser"`
`ATT "index" "0"`
`PARAM "parameter" "86A0037691E47881D5CF"`
`ATT "encryption" "ovcrypt"`
`ATT "index" "1"`
`GROUP_END`
`GROUP_END`
	See **Perl Script Source** section

`GROUP "KEYMAPRULE"`
`PARAM "KEEP_INPUT_FIELDS" "1"`
`GROUP "KEYFIELD_MAPPINGS"`
`PARAM "KF_MAP" "KEEP"`
`ATT "KEY_IN" "someFiieldinXML"`
`ATT "KEY_OUT" "someOutputFileld"`
`PARAM "KF_MAP" "KEEP"`
`ATT "KEY_IN" "someOtherFieldinInput"`
`ATT "KEY_OUT" "someOtherOutputField"`
`GROUP_END`
`GROUP "ADDITIONAL_FIELDS"`
`PARAM "ADD_FIELD" "ADD"`
`ATT "NAME" "anAdditionalField"`
`ATT "VALUE" "SomeFieldinInput, SomeOtherFieldinInput"`
`GROUP_END`
`GROUP_END`
	See **Schema section (Generic Output)**

`GROUP_END` `GROUP_END` `GROUP_END`	Do not change

`MAP "map1"`
`DESCRIPTION ""`
`INPUT "input1"`
`FROM "source1" TO "target1"`
	Optional section

FROM "source2" TO "target2" MAP "map2" DESCRIPTION "" INPUT "input2" FROM "source3" TO "target3" FROM "source4" TO "target4"	See Mappings section
DEFAULTMSG	Do not change End of Policy

16. Syntax of Generic Output from DB policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Generic Output from Database policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – change in Policy Properties

`POLTYPE " genoutdb"`
`GROUP "ROOT"`
`GROUP "content"`
`GROUP "options"`
`GROUP_END`
	Do not change

`GROUP "sources"`	

`GROUP "KEYMAPRULE"`
`PARAM "KEEP_INPUT_FIELDS" "1"`
`GROUP "KEYFIELD_MAPPINGS"`
`PARAM "KF_MAP" "KEEP"`
`ATT "KEY_IN" "someFiieldinXML"`
`ATT "KEY_OUT" "someOutputFileld"`
`PARAM "KF_MAP" "KEEP"`
`ATT "KEY_IN" "someOtherFieldinInput"`
`ATT "KEY_OUT" "someOtherOutputField"`
`GROUP_END`
`GROUP "ADDITIONAL_FIELDS"`
`PARAM "ADD_FIELD" "ADD"`
`ATT "NAME" "anAdditionalField"`
`ATT "VALUE" "SomeFieldinInput, SomeOtherFieldinInput"`
`GROUP_END`
`GROUP_END`
	See **Schema section (Generic Output)**

`PARAM "interval" "5m0s"` `GROUP "configuration"` `PARAM "jdbcclasspath" "/opt/OV/java/sqljdbc42.jar"` `PARAM "jdbcdriver" "com.microsoft.sqlserver.jdbc.SQLServerDriver"` `PARAM "connectstring" "jdbc:sqlserver://dbserver.example.com:1433;Database=Event;"` `PARAM "username" "sa"` `PARAM "password" "%%DBUSERPASSWORD%%"` `ATT "encryption" "ovcrypt"` `PARAM "sqlinit" "SELECT MAX(time_received) as timestamp FROM event.dbo.all_events"` `PARAM "sql" "SELECT (SELECT COUNT(*) FROM event.dbo.all_events WHERE state = 'OPEN') AS openstate, (SELECT count(*) FROM event.dbo.all_events WHERE state = 'IN_PROGRESS') AS in_progress, (SELECT count(*) FROM event.dbo.all_events WHERE state = 'RESOLVED') AS resolved, (SELECT count(*) FROM event.dbo.all_events WHERE state = 'CLOSED' AND time_received > '<$DATA:timestamp>') AS closed, 'OMi Prod' AS OMi"` `PARAM "resultsetsize" "100"` `PARAM "fetchsize" "100"` `GROUP "connectproperties"`	See **DB** Source section

```
                GROUP "field"
                   PARAM "name"  "SomeProperty"
                   PARAM "value"  "someValue"
                GROUP_END
             GROUP_END
          GROUP_END
          GROUP "data"
             GROUP "field"
                PARAM "name"  "timestamp"
                PARAM "value"  "2020-09-01 10:00:00"
             GROUP_END
          GROUP_END
          PARAM "sourceTag"  "SomeTag"
       GROUP_END
```

```
          GROUP "KEYMAPRULE"
             PARAM "KEEP_INPUT_FIELDS"  "1"
             GROUP "KEYFIELD_MAPPINGS"
                PARAM "KF_MAP"  "KEEP"
                ATT "KEY_IN"  "someFiieldinXML"
                ATT "KEY_OUT"  "someOutputFileld"
                PARAM "KF_MAP"  "KEEP"
                ATT "KEY_IN"  "someOtherFieldinInput"
                ATT "KEY_OUT"  "someOtherOutputField"
             GROUP_END
             GROUP "ADDITIONAL_FIELDS"
                PARAM "ADD_FIELD"  "ADD"
                ATT "NAME"  "anAdditionalField"
                ATT "VALUE"  "SomeFieldinInput, SomeOtherFieldinInput"
             GROUP_END
          GROUP_END
```

See Schema section (Generic Output)

```
       GROUP_END
    GROUP_END
GROUP_END
```

Do not change

```
      MAP "map1"
      DESCRIPTION ""
      INPUT "input1"
        FROM "source1" TO "target1"
        FROM "source2" TO "target2"
      MAP "map2"
      DESCRIPTION ""
      INPUT "input2"
        FROM "source3" TO "target3"
        FROM "source4" TO "target4"
```

Optional section

See Mappings section

```
      DEFAULTMSG
```

Do not change End of Policy

17. Syntax of Generic Output from REST Web Service Listener policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Generic Output from XML File policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – change in Policy Properties

`POLTYPE "genoutwsxml"`
`GROUP "ROOT"`
`GROUP "content"`
`GROUP "options"`
`GROUP_END`
	Do not change

`GROUP "sources"`
`PARAM "logpath" "/receiver"`
`PARAM "chSet" "69"`
`GROUP "roots"`
`GROUP "rootPair"`
`PARAM "root" "someXMLtag"`
`GROUP_END`
`GROUP_END`
	See
	
	REST Web Service Listener Source section

`GROUP "KEYMAPRULE"`
`PARAM "KEEP_INPUT_FIELDS" "1"`
`GROUP "KEYFIELD_MAPPINGS"`
`PARAM "KF_MAP" "KEEP"`
`ATT "KEY_IN" "someFiieldinXML"`
`ATT "KEY_OUT" "someOutputFileld"`
`PARAM "KF_MAP" "KEEP"`
`ATT "KEY_IN" "someOtherFieldinInput"`
`ATT "KEY_OUT" "someOtherOutputField"`
`GROUP_END`
`GROUP "ADDITIONAL_FIELDS"`
`PARAM "ADD_FIELD" "ADD"`
`ATT "NAME" "anAdditionalField"`
`ATT "VALUE" "SomeFieldinInput, SomeOtherFieldinInput"`
`GROUP_END`
`GROUP_END`
	See **Schema section (Generic Output)**

`GROUP_END` `GROUP_END` `GROUP_END`	Do not change

`MAP "map1"`
`DESCRIPTION ""`
`INPUT "input1"`
`FROM "source1" TO "target1"`
`FROM "source2" TO "target2"`
`MAP "map2"`
`DESCRIPTION ""`
`INPUT "input2"`
`FROM "source3" TO "target3"`
`FROM "source4" TO "target4"`
	Optional section
	
	See Mappings section

`DEFAULTMSG`	Do not change End of Policy

18. Syntax of Topology from XML File policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Topology from XML File policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – can be changed in Policy Properties

`POLTYPE "xmltopo"`
`  GROUP "ROOT"`
`  GROUP "content"`
`  GROUP "options"`
`  GROUP_END`
	Do not change

`  GROUP "sources"`
`  PARAM "topofile"  "/path/to/file"`
`  PARAM "interval"  "1h"`
`  PARAM "chSet"  "69"`
`  PARAM "noTopofileMsg"  "1"`
`  PARAM "ageToDeletion"  "3"`
`  PARAM "doDeltaDetection"  "1"`
`  GROUP_END`
	See
	
	XML File Source section (topology)

`  GROUP_END`
`  GROUP_END`
`DEFAULTMSG`
	Do not change
	End of Policy

19. Syntax of Topology from REST Web Service Listener policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Topology from REST Web Service Listener policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax."`
	Do not change – can be changed in Policy Properties

`POLTYPE "wsxmltopo"`
`GROUP "ROOT"`
`GROUP "content"`
`GROUP "options"`
`GROUP_END`
	Do not change

`GROUP "sources"`
`PARAM "topopath" "/listener"`
`PARAM "chSet" "69"`
`PARAM "ageToDeletion" "3"`
`PARAM "doDeltaDetection" "1"`
`GROUP_END`
	See **REST Web Service Listener Source section** (topology)

`GROUP_END`
`GROUP_END`
`DEFAULTMSG`
	Do not change
	End of Policy

20. Syntax of Data Forwarding policy template

Example policy template

`SYNTAX_VERSION 13`
 	
`GENERIC_SOURCE "Example Data Forwarding policy template"`
`DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`
	Do not change – can be changed in Policy Properties

`POLTYPE "datasender"`
`GROUP "ROOT"`
`GROUP "content"`
`GROUP "options"`
`GROUP_END`
	Do not change

`GROUP "sources"` `GROUP "TARGETS"`	Start of Targets section

`GROUP "TARGET"`
`PARAM "id" "7215BE36-10F8-A2AC-9CD1-209FCCFE4A00"`
`PARAM "NAME" "Target1XML"`
`PARAM "URL" "http://target1.example.com/receiver"`
`PARAM "WIREFORMAT" "XML"`
`PARAM "SENDMODE" "SNF"`
`PARAM "DESCRIPTION" "Example of a target with XML wire format and guaranteed delivery"`
`GROUP_END`
	1. Target, see **Data Forwarding** Targets section

`GROUP "TARGET"`
`PARAM "id" "F893917D-1B91-7B55-D969-20A0A51F83CF"`
`PARAM "NAME" "Target2JSON"`
`PARAM "URL" "http://target12.example.com/receiver"`
`PARAM "WIREFORMAT" "JSON"`
`PARAM "SENDMODE" "FAF"`
`PARAM "DESCRIPTION" "Example of a target with JSON wire format and no guaranteed delivery"`
`GROUP_END`
	2. Target, see **Data Forwarding** Targets section

Add as many targets as needed. See the policy example that is part of the *Example Policy Templates* Management Pack for additional target examples.	
`GROUP_END`	End of Targets Section
`GROUP "RULES"`	Start of Rules section

`GROUP "RULE"`
`PARAM "RULETYPE" "MATCH"`
`PARAM "DATATYPE" "METRIC"`
`PARAM "id" "497F81B3-B46B-5F98-A6FA-20A2012BD76A"`
`PARAM "DESCRIPTION" "Forward on matched, no overwrite, affect all"`
`GROUP "CONDITION"`
`PARAM "OPER" "SomeProperty"`
`ATT "OP" "=="`
`ATT "OPER" "SomeValue"`
`GROUP_END`
`GROUP "TARGETS"`
	1. Rule, example of a Metric Data Forwarding Rule, see
	
	Data Forwarding Rules section

PARAM "AFFECT_ALL" "1" GROUP_END GROUP_END	Data Forwarding Targets section
GROUP "RULE" PARAM "RULETYPE" "MATCH" PARAM "DATATYPE" "RAW" PARAM "id" "E1DF7AD2-2DB3-5B9E-7FFD-20A76E9193A9" PARAM "DESCRIPTION" "Forward on matched" GROUP "CONDITION" PARAM "OPER" "SomeProperty" ATT "OP" "~=" ATT "OPER" "<*>" GROUP_END GROUP "TARGETS" PARAM "AFFECT_ALL" "1" GROUP_END GROUP_END	2.Rule, example of a Structured Input Data Forwarding rule, see **Data Forwarding** Rules section
Add as many rules as needed. See the policy example that is part of the *Example Policy Templates* Management Pack for additional rule examples.	
GROUP_END	End of Rules section
GROUP_END GROUP_END GROUP_END DEFAULTMSG	Do not change End of Policy

21. Syntax of Operations Connector High Availability policy template

Example policy template

`GENERIC_SOURCE "Example Operations Connector High Availability policy template"` `DESCRIPTION "Example policy that uses many available features to illustrate policy template syntax"`	Do not change – can be changed in Policy Properties

`POLTYPE "bsmcha"`
`GROUP "ROOT"`
`GROUP "content"`
`GROUP "options"`
`GROUP_END`
	Do not change

`PARAM "type" "CMD"`
`PARAM "command" "activationstatecommand"`
`GROUP "policies"`
`GROUP "policy"`
`PARAM "name" "VROPS_Cx_MetricRegistration"`
`PARAM "id" "322614db-a8d7-499e-a845-2ac77c82a39f"`
`PARAM "mode" "MANAGED"`
`GROUP_END`
`GROUP "policy"`
`PARAM "name" "VROPS_Cx_EventIntegration"`
`PARAM "id" "7cd113a2-6aa0-43ef-b7ef-45ea0580d6ad"`
`PARAM "mode" "ALWAYS"`
`GROUP_END`
`GROUP_END`
	See **Activation State and** Policies section

`GROUP "components"`
`GROUP "component"`
`PARAM "type" "OVC"`
`PARAM "name" "component1"`
`PARAM "mode" "ALWAYS"`
`PARAM "file" "component.reg"`
`GROUP_END`
`GROUP "component"`
`PARAM "type" "EXTERNAL"`
`PARAM "name" "component2"`
`PARAM "mode" "MANAGED"`
`PARAM "startCmd" "componentcmd -start"`
`PARAM "stopCmd" "componentcmd -stop"`
`GROUP_END`
`GROUP_END`
	See Components section

`GROUP "syncItems"`
`GROUP "syncItem"`
`PARAM "file" "/path/file"`
`GROUP_END`
`GROUP "syncItem"`
`PARAM "file" "c:\\path\\anotherfile"`
`GROUP_END`
`GROUP_END`
	See **Sync Items** section

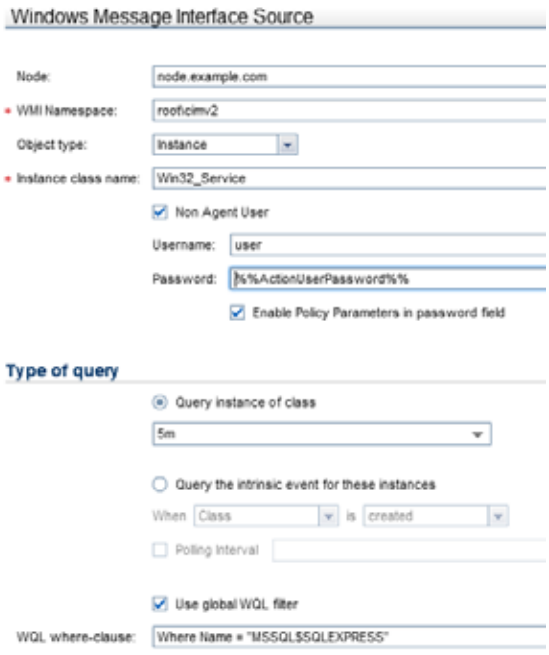
`GROUP_END`
`GROUP_END`
`DEFAULTMSG`
	Do not change End of Policy

Detailed Syntax Descriptions

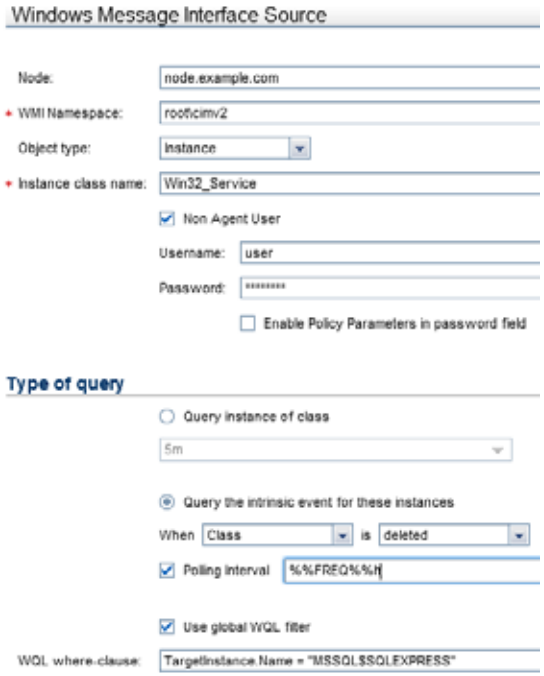
22.Windows Management Interface Source section

UI Reference

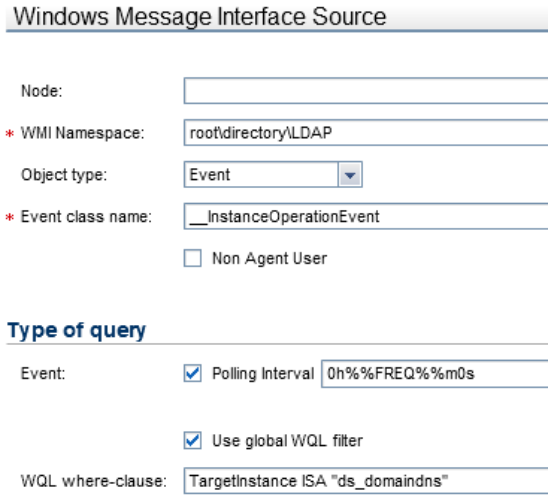
Object Type: Instance, query instance



Object Type: Instance, query intrinsic event



Object Type: Event



For more information about the Windows Management Interface Source section, see the Configure Windows Management Interface Policies topic.

Example of corresponding policy syntax

Object Type: Instance, query instance

```
WMI_USERNAME "user"
WMI_PASSWORD "%%ActionUserPassword%%"
NAMESPACE "\\\\node.example.com\\root\\cimv2"
CLASS "Win32_Service"
WHERE_CLAUSE "Where Name = \"MSSQL$SQLEXPRESS\""
QUERY_LANGUAGE "WQL"
INTERVAL "5m"
```

Object Type: Instance, query intrinsic event

```
NAMESPACE "\\\\node.example.com\\root\\cimv2"
CLASS "Win32_Service"
WITHIN "%%FREQ%%h"
WHERE_CLAUSE "TargetInstance.Name = \"MSSQL$SQLEXPRESS\""
CLASS_DELETION_EVENT
QUERY_LANGUAGE "WQL"
```

Object Type: Event

```
NAMESPACE "\\\\.\\root\\directory\\LDAP"
CLASS "__InstanceOperationEvent"
WHERE_CLAUSE "TargetInstance ISA \"ds_domaindns\""
WITHIN "0h%%FREQ%%m0s"
QUERY_LANGUAGE "WQL"
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Node and WMI Namespace	`NAMESPACE`	Build namespace using `\\<node>\<WMI namespace>` but escape each \ with another \ Example: `node.example.com` and `root\cimv2`, use `NAMESPACE "\\\\node.example.com\\root\\cimv2"`
Object Type	-	-
Event / Instance Class name	`CLASS`	`"<user-defined>"`
Non-Agent user	-	-
Username	`WMI_USER`	`"<user-defined>"`
Password	`WMI_PASSWORD`	`"<user-defined>"` see the Passwords section - use password parameter using `%%<parameter_name>%%` syntax
Use global WQL-filter WQL where-clause	`WHERE_CLAUSE`	`"<user-defined>"` escape each " with \
Query instance of class + interval selection	`INTERVAL`	`"<user-defined>"` following d/h/m/s syntax, e.g. `"3d5m0s"` or `"2h"` Lowest possible value: 15 sec

Query intrinsic event
When Class/Instance/Namespace is created/modified/deleted
	`CLASS_CREATION_EVENT`
	`CLASS_MODIFICATION_EVENT`
	`CLASS_DELETION_EVENT`
	`INSTANCE_CREATION_EVENT`
	`INSTANCE_MODIFICATION_EVENT`
	`INSTANCE_DELETION_EVENT`
	`NAMESPACE_CREATION_EVENT`
	`NAMESPACE_MODIFICATION_EVENT`
	`NAMESPACE_DELETION_EVENT`
		Use one of the corresponding syntax elements

Polling interval	`WITHIN`	`"<user-defined>"` following d/h/m/s syntax, e.g. `"3d5m0s"` or `"2h"` Lowest possible value: 15 sec
Event Polling interval	`WITHIN`	`"<user-defined>"` following d/h/m/s syntax, e.g. `"3d5m0s"` or `"2h"` Lowest possible value: 15 sec

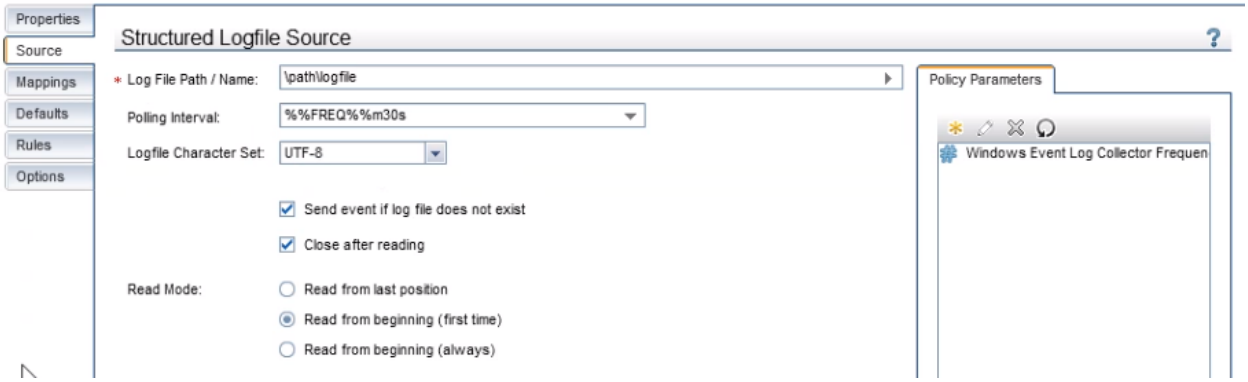
Usage Reference

The Windows Management Interface Source section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Windows Management Interface policy template

23.Structured Log File Source section

UI Reference



For more information about the Structured Log File Source section, see the Configure Structured Log File Policies topic.

Example of corresponding policy syntax

```
GROUP "sources"
    PARAM "logpath"  "/path/logfile"
    PARAM "interval"  "%%FREQ%%m30s"
    PARAM "chSet"  "69"
    PARAM "readMode"  "firstFromBegin"
    PARAM "noLogfileMsg"  "1"
    PARAM "closeAfterRead"  "1"
    GROUP "struliPatterns"
       PARAM "struliPattern"  "^<@.Severity>,<@.SourceCI>,<@.RelatedCI>,<*.Timestamp>"
    GROUP_END
    GROUP "startPatterns"
       PARAM "startPattern"  "<*>,<*>,<#>/<#>/<#> <#>:<#>:<#> <2*>,<*>"
    GROUP_END
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values

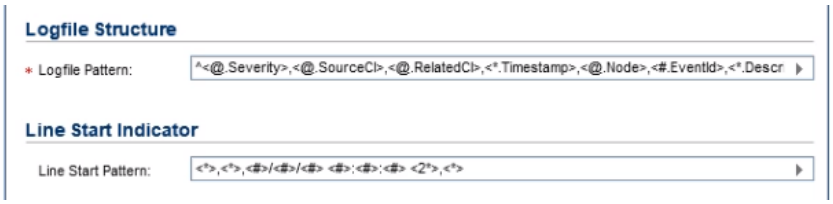
Log File Path / Name	`PARAM "logpath"`	`"<user-defined>"`
		For Windows specify path to logfile using \\ as delimiter, e.g.
		Windows: `C:\path\logfile.log`
		`"c:\\path\\logfile.log"`
		Unix: `/path/logfile.log`

Polling Interval	`PARAM "interval"`	`"<user-defined>"` following d/h/m/s syntax, e.g. `"3d5m0s"` or `"2h"` Lowest possible value: 15 sec
Log File Character Set	`PARAM "chSet"`	See Table below
Send event if file does not exist	`PARAM "noLogfileMsg"`	`"1"` (selected) or `"0"` (unselected)
Close after reading	`PARAM "closeAfterRead"`	`"1"` (selected) or `"0"` (unselected)

Read from last position
Read from beginning (first time)
Read from beginning (always)
	`PARAM "readMode"`	`"fromLastPos"`
		`"firstFromBegin"`
		`"alwaysFromBegin"`

Character Set	Value	Character Set	Value	Character Set	Value
ACP 1250	30	ISO 8859-5	9	OEMCP 861	21
ACP 1251	29	ISO 8859-6	10	OEMCP 862	22
ACP 1252	4	ISO 8859-7	11	OEMCP 863	23
ASCII	0	ISO 8859-8	12	OEMCP 864	24
Big-5 Taiwanese	73	ISO 8859-9	13	OEMCP 865	25
EBCDIC	3	OEMCP 437	7	OEMCP 866	26
EUC Japanese	65	OEMCP 737	15	OEMCP 869	27
EUC Korean	70	OEMCP 775	16	ROMAN 8	2
EUC Taiwanese	71	OEMCP 850		Shift-JIS	65
GB-2312-80 Chinese	72	OEMCP 852	17	TIS 620	14
ISO 8859-1	1	OEMCP 855	18	UCS-2	67
ISO 8859-15	28	OEMCP 857	19	UTF-8	69
ISO 8859-2	8	OEMCP 860	20		

UI Reference – Logfile Structure (events)



Example of corresponding policy syntax

```
GROUP "sources"
            PARAM "logpath"  "/path/logfile"
            PARAM "interval"  "%%FREQ%%m30s"
            PARAM "chSet"  "69"
            PARAM "readMode"  "firstFromBegin"
            PARAM "noLogfileMsg"  "1"
            PARAM "closeAfterRead"  "1"
            GROUP "struliPatterns"
               PARAM "struliPattern"
"^<@.Severity>,<@.SourceCI>,<@.RelatedCI>,<*.Timestamp>,<@.Node>,<#.EventId>,<*.Description>"
            GROUP_END
            GROUP "startPatterns"
               PARAM "startPattern"  "<*>,<*>,<#>/<#>/<#> <#>:<#>:<#> <2*>,<*>"
            GROUP_END
         GROUP_END
```

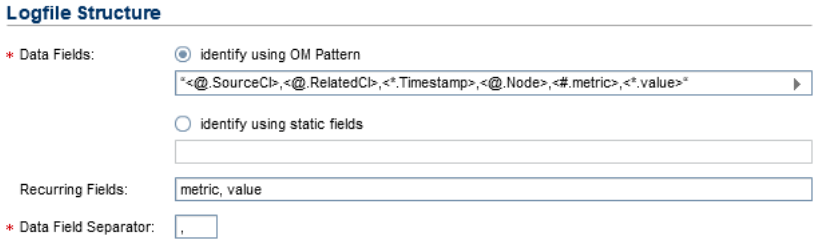
Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values

Logfile Pattern	`GROUP "struliPatterns"`
	`PARAM "struliPattern"`
	`GROUP_END`
		`"<user-defined pattern>"`

Line Start Pattern	`GROUP "startPatterns"`
	`PARAM "startPattern"`
	`GROUP_END`
		`"<user-defined pattern>"`

UI Reference – Logfile Structure (metrics)



For more information about the Structured Log File Source section, see the Configure Structured Log File Policies topic.

Example of corresponding policy syntax

```
        GROUP "sources"
            PARAM "logpath"  "/path/to/logfile"
            PARAM "interval"  "5m"
            PARAM "chSet"  "69"
            PARAM "readMode"  "fromLastPos"
            PARAM "noLogfileMsg"  "1"
            PARAM "closeAfterRead"  "1"
            PARAM "dataFieldSeparator"  ","
            PARAM "recurringDataFields"  "metric, value"
            PARAM "pattern"
""<@.SourceCI>,<@.RelatedCI>,<*.Timestamp>,<@.Node>,<#.metric>,<*.value>""
        GROUP_END
OR
        GROUP "sources"
            PARAM "logpath"  "/path/to/logfile"
            PARAM "interval"  "5m"
            PARAM "chSet"  "69"
            PARAM "readMode"  "fromLastPos"
            PARAM "noLogfileMsg"  "1"
            PARAM "closeAfterRead"  "1"
            PARAM "dataFieldSeparator"  ","
            PARAM "staticDataFields"  "time"
            PARAM "recurringDataFields"  "metric, value"
        GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Data Fields Identify using OM Pattern:	PARAM "pattern"	"<user-defined pattern>"
Data Fields Identify using static fields	PARAM "staticDataFields"	"<user-defined>"
Recurring Fields	PARAM "recurringDataFields"	"<Comma-separate list of recurring fields>"
Data Field Separator	PARAM "dataFieldSeparator"	"<user-defined>"

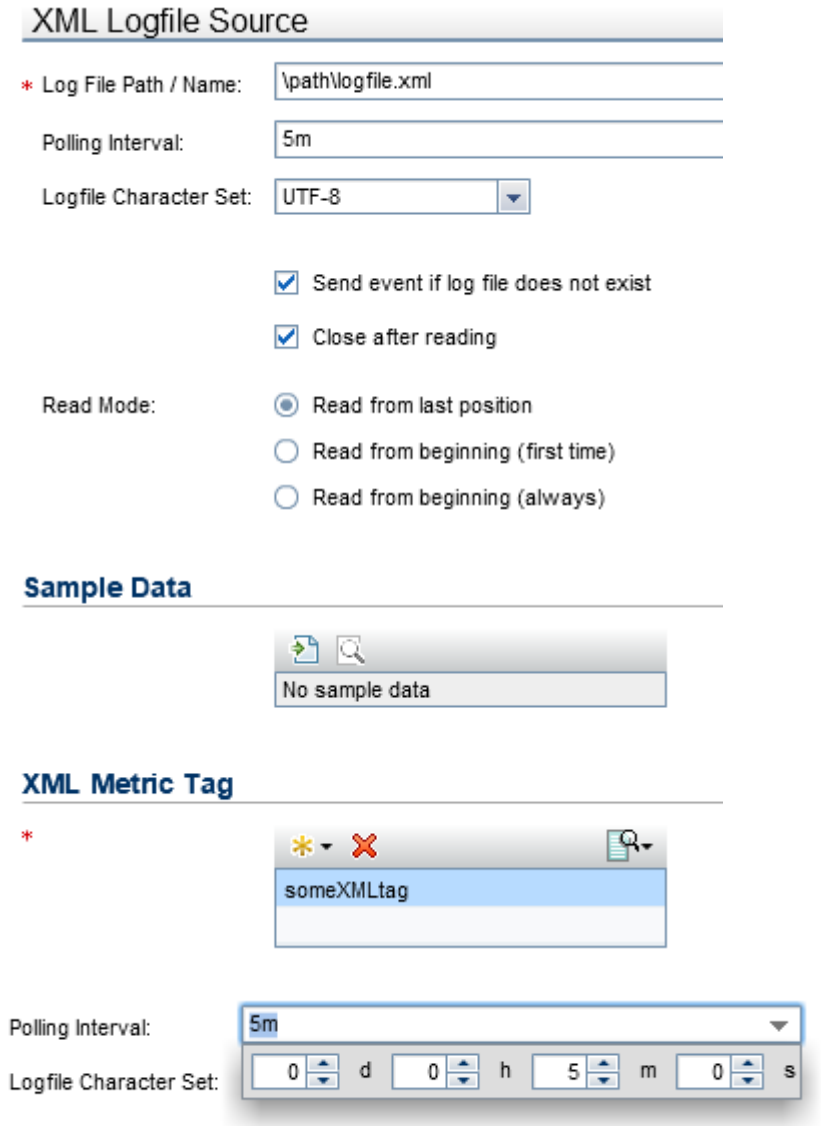
Usage Reference

The Structured Log File Source section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Event from Structured Log File policy template
- Syntax of Metric from Structured Log File policy template

24. XML File Source section (event, metrics, generic output)

UI Reference



For details, see the Source Page topic.

Example of corresponding policy syntax

```
GROUP "sources"
        PARAM "logpath"   "/path/logfile.xml"
        PARAM "interval"   "5m"
        PARAM "chSet"   "69"
        PARAM "readMode"   "fromLastPos"
        PARAM "noLogfileMsg"   "1"
        PARAM "closeAfterRead"   "1"
        GROUP "roots"
           GROUP "rootPair"
              PARAM "root"   "someXMLtag"
           GROUP_END
        GROUP_END
     GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values

Log File Path / Name	`PARAM "logpath"`	`"<user-defined>"`
		For Windows specify path to logfile using \\ as delimiter, e.g.
		Windows: `C:\path\logfile.log`
		`"c:\\path\\logfile.log"`
		Unix: `/path/logfile.log`

Poling Interval	`PARAM "interval"`	`"<user-defined>"` following d/h/m/s syntax, e.g. `"3d5m0s"` or `"2h"` Lowest possible value: 15 sec
Log File Character Set	`PARAM "chSet"`	See the table below
Send event if file does not exist	`PARAM "noLogfileMsg"`	`"1"` (selected) or `"0"` (unselected)
Close after reading	`PARAM "closeAfterRead"`	`"1"` (selected) or `"0"` (unselected)

Read from last position
Read from beginning (first time)
Read from beginning (always)
	`PARAM "readMode"`	`"fromLastPos"`
		`"firstFromBegin"`
		`"alwaysFromBegin"`

Sample Data	-	-
XML Metric Tag	`PARAM "root"`	`"<user-defined>"`

Character Set	Value	Character Set	Value	Character Set	Value
ACP 1250	30	ISO 8859-5	9	OEMCP 861	21
ACP 1251	29	ISO 8859-6	10	OEMCP 862	22
ACP 1252	4	ISO 8859-7	11	OEMCP 863	23
ASCII	0	ISO 8859-8	12	OEMCP 864	24
Big-5 Taiwanese	73	ISO 8859-9	13	OEMCP 865	25
EBCDIC	3	OEMCP 437	7	OEMCP 866	26
EUC Japanese	65	OEMCP 737	15	OEMCP 869	27
EUC Korean	70	OEMCP 775	16	ROMAN 8	2
EUC Taiwanese	71	OEMCP 850		Shift-JIS	65
GB-2312-80 Chinese	72	OEMCP 852	17	TIS 620	14
ISO 8859-1	1	OEMCP 855	18	UCS-2	67

ISO 8859-15	28	OEMCP 857	19	UTF-8	69
ISO 8859-2	8	OEMCP 860	20		

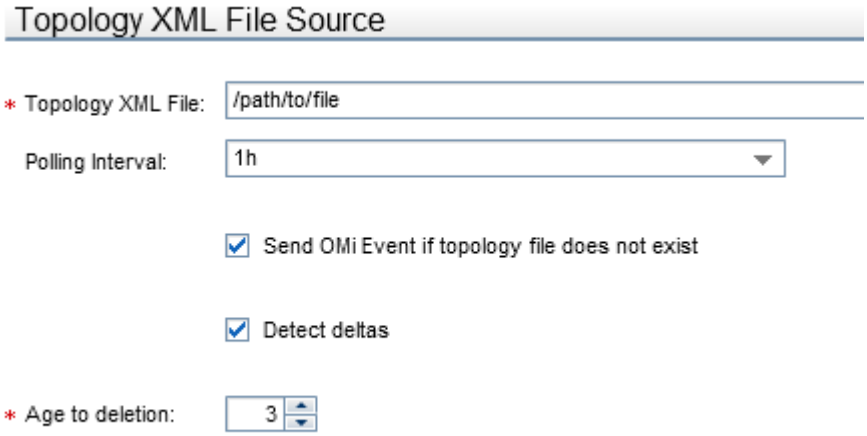
Usage Reference

The Mappings section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Event from XML File policy template
- Syntax of Metric from XML File policy template
- Syntax of Generic Output from XML File policy template

25.XML File Source section (topology)

UI Reference



For more information, see the Configure XML File Policies topic.

Example of corresponding policy syntax

```
GROUP "sources"
   PARAM "topofile"   "/path/to/file"
   PARAM "interval"   "1h"
   PARAM "chSet"   "69"
   PARAM "noTopofileMsg"   "1"
   PARAM "ageToDeletion"   "3"
   PARAM "doDeltaDetection"   "1"
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Topology XML File	`PARAM "topofile"`	`"<user-defined>"`
Poling Interval	`PARAM "interval"`	`"<user-defined>"` following hms syntax Lowest possible 15 sec
-	`PARAM "chSet"`	`"69"` (UTF-8)
Send event if file does not exist	`PARAM "noTopofileMsg"`	`"1"` (selected) or `"0"` (unselected)
Detect deltas	`PARAM "doDeltaDetection"`	`"1"` (selected) or `"0"` (unselected)
Age to deletion	`PARAM "ageToDeletion"`	`"<user-defined>"` After this number of policy executions, a CI that is no longer part of the data is deleted from the server.

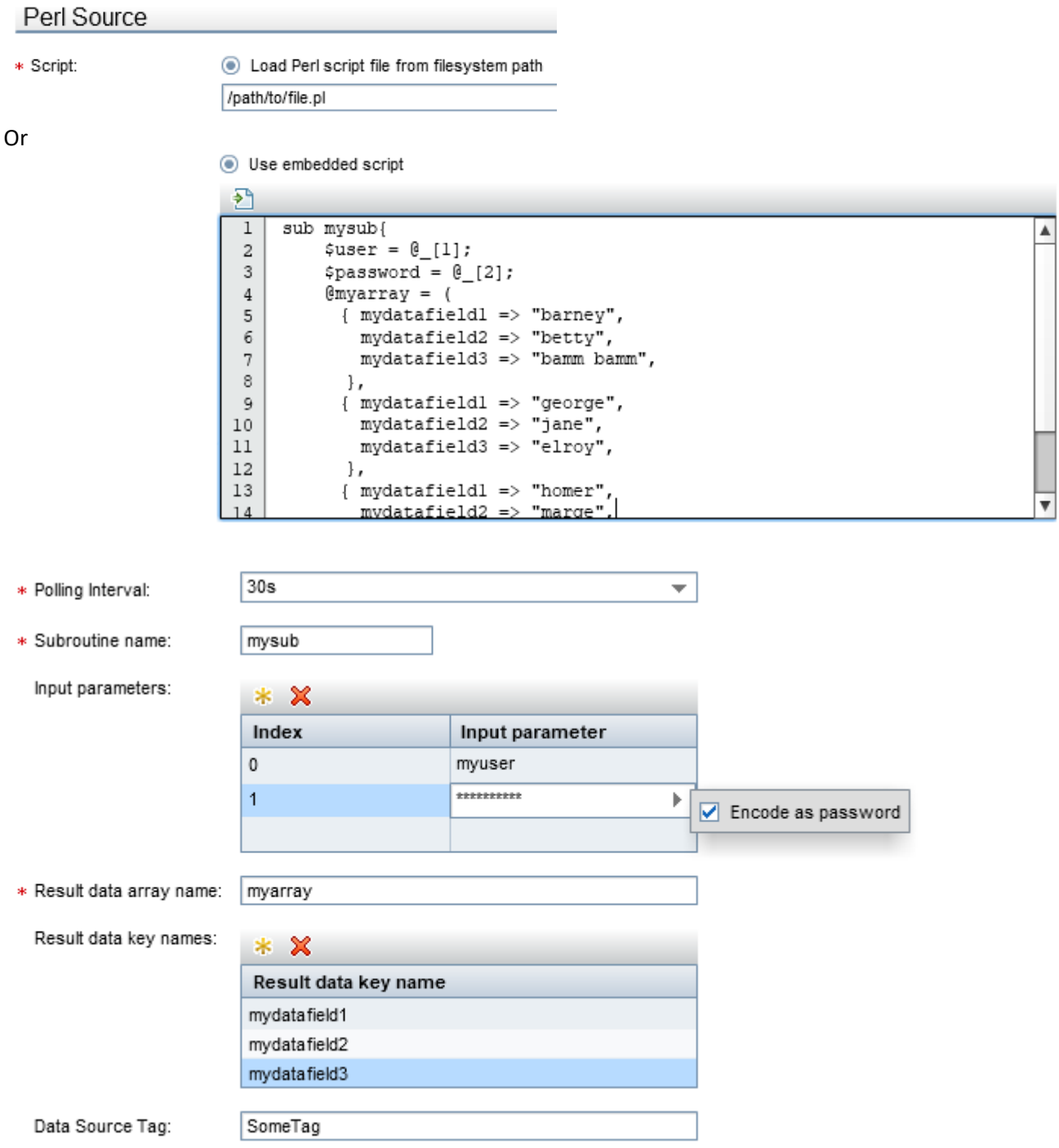
Usage Reference

The XML File Source (topology) section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Topology from XML File policy template

26. Perl Script Source section

UI Reference



For more information, see the Configure Perl Script Policies topic.

Example of corresponding policy syntax

```
          GROUP "sources"
             PARAM "interval"  "5m"
             PARAM "perlscriptname"  "/path/to/file.pl"
             PARAM "perlsubname"  "subroutine"
             PARAM "perldatarray"  "myarray"
             GROUP "datafields"
                PARAM "dataField"  "mydatafield1"
                PARAM "dataField"  "mydatafield2"
                PARAM "dataField"  "mydatafield3"
             GROUP_END
             PARAM "sourceTag"  "SomeTag"
             GROUP "inputparameters"
                PARAM "parameter"  "myuser"
                ATT "index"  "0"
                PARAM "parameter"  "86A0037691E47881D5CF"
                ATT "encryption"  "ovcrypt"
                ATT "index"  "1"
             GROUP_END
          GROUP_END
OR
          GROUP "sources"
             PARAM "interval"  "5m"
             PARAM "perlscriptname"  "/"
             PARAM "perlscript"  "IyEvdXNyL2JIyEvdXNyL2Jpbi9wZXJsCnVzZSB"
             ATT "encoding"  "base64"
             PARAM "perlsubname"  "mysub"
             PARAM "perldatarray"  "myarray"
             GROUP "datafields"
                PARAM "dataField"  "mydatafield1"
                PARAM "dataField"  "mydatafield2"
                PARAM "dataField"  "mydatafield3"
             GROUP_END
             PARAM "sourceTag"  "SomeTag"
             GROUP "inputparameters"
                PARAM "parameter"  "myuser"
                ATT "index"  "0"
                PARAM "parameter"  "86A0037691E47881D5CF"
                ATT "encryption"  "ovcrypt"
                ATT "index"  "1"
             GROUP_END
          GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Load Perl script from filesystem path	`PARAM "perlscriptname"`	`"<user-defined>"` if perl script is read from file system, otherwise must be set to "/"
Use embedded script	`PARAM "perlscript" "IyEvdXNyL2JIyEvdXNyL2Jpbi9wZXJsCnVzZSB" ATT "encoding"  "base64"`	User-defined Perl script code encoded in base64
Poling Interval	`PARAM "interval"`	`"<user-defined>"` following hms syntax Lowest possible 15 sec

Subroutine name	`PARAM "perlsubname"`	`"<user-defined>"` Must match the subroutine used in the Perl script.

Input parameters	`GROUP "inputparameters"`
	`  PARAM "parameter"  "myuser"`
	`  ATT "index"  "0"`
	`  PARAM "parameter" "86A0037691E47881D5CF"`
	`  ATT "encryption"  "ovcrypt"`
	`  ATT "index"  "1"`
	`GROUP_END`
		User-defined input parameters to subroutine
		Parameters that will contain passwords need to be encrypted using opr-pwcrypt.

Result data array name	`PARAM "perldatarray"`	`"<user-defined>"` Must match the array of hashes used in the Perl script.

Result data key names	`GROUP "datafields"`
	`  PARAM "dataField"  "mydatafield1"`
	`  PARAM "dataField"  "mydatafield2"`
	`  PARAM "dataField"  "mydatafield3"`
	`GROUP_END`
		User defined data field names
		Must match the attribute names of interest for each hash element contained in array of hashes

Data Source Tag	`PARAM "sourceTag"`	`"<user-defined>"`

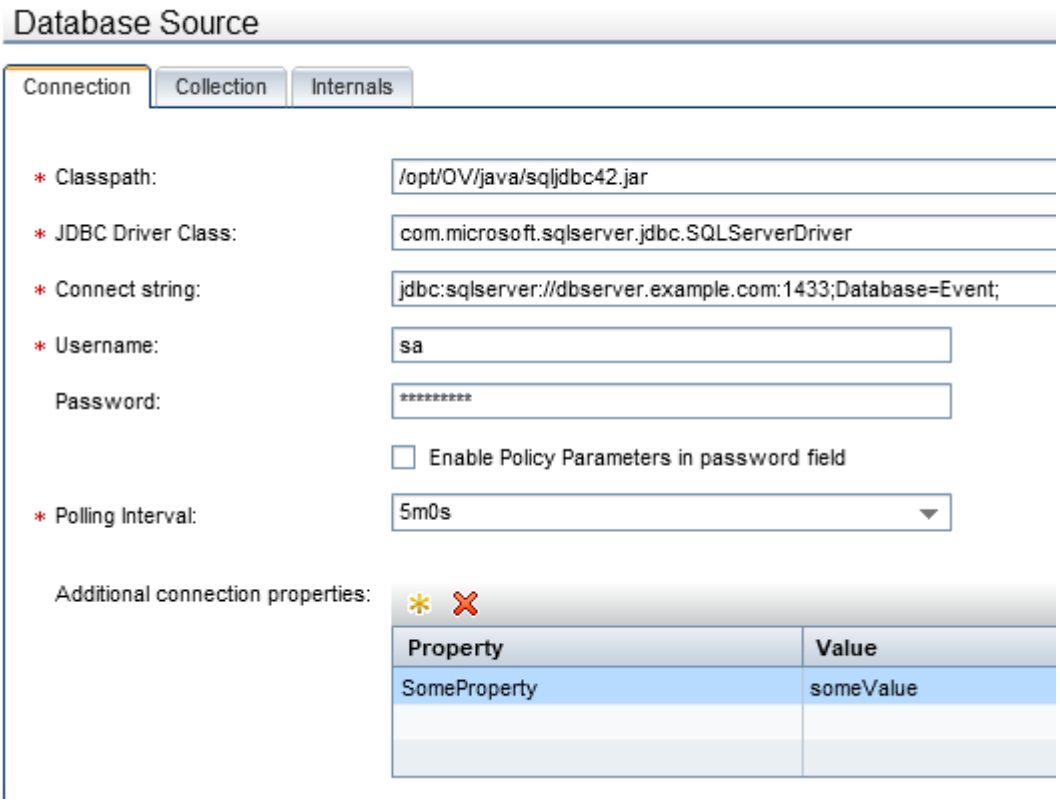
Usage Reference

The XML File Source (topology) section is used in the syntax of the following policy types (click on link to navigate back):

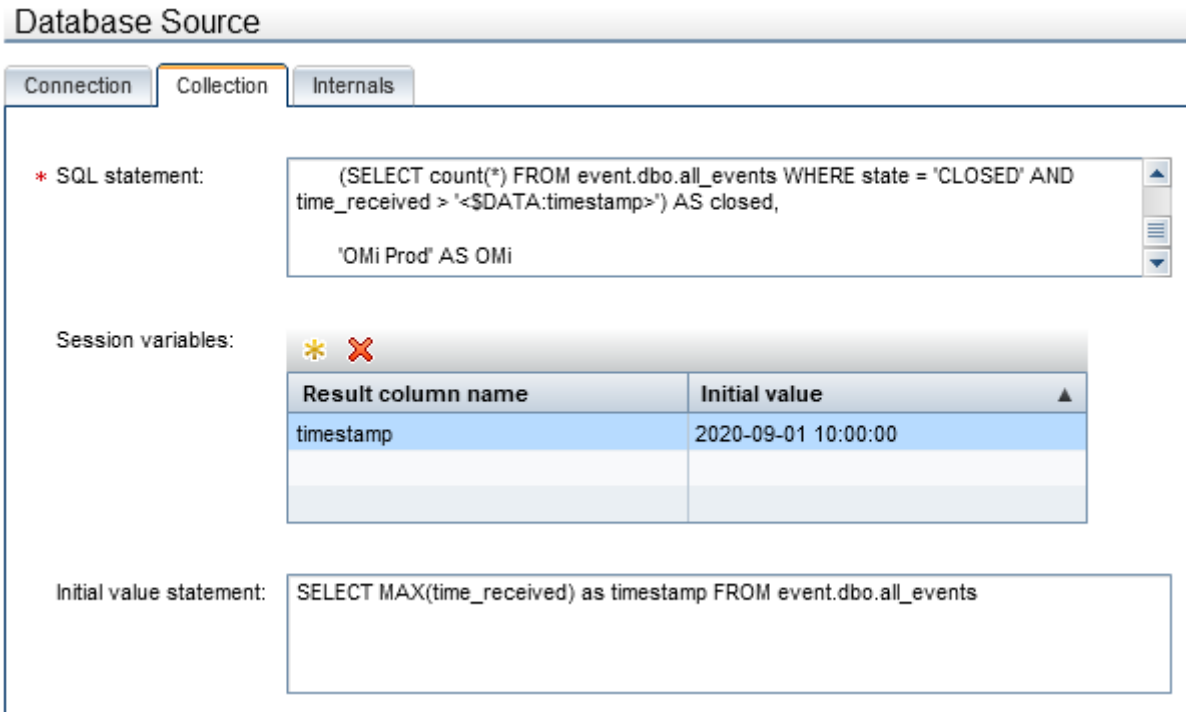
- Syntax of Event from Perl Script policy template
- Syntax of Metric from Perl Script policy template
- Syntax of Generic Output from Perl Script policy template

27.DB Source section

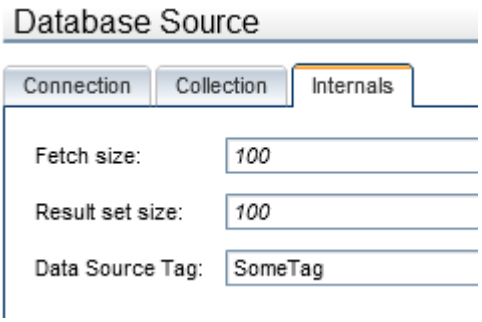
UI Reference



For details, see the Connection Tab topic.



For details, see the Collection Tab topic.



For details, see the Internals Tab topic.

Example of corresponding policy syntax

```
GROUP "sources"
   PARAM "interval"  "5m0s"
   GROUP "configuration"
      PARAM "jdbcclasspath"  "/opt/OV/java/sqljdbc42.jar"
      PARAM "jdbcdriver"  "com.microsoft.sqlserver.jdbc.SQLServerDriver"
      PARAM "connectstring"  "jdbc:sqlserver://dbserver.example.com:1433;Database=Event;"
      PARAM "username"  "sa"
      PARAM "password"  "%%DBUSERPASSWORD%%"
      ATT "encryption"  "ovcrypt"
      PARAM "sqlinit"  "SELECT MAX(time_received) as timestamp FROM event.dbo.all_events"
      PARAM "sql"  "SELECT (SELECT COUNT(*) FROM event.dbo.all_events WHERE state =
              'OPEN') AS openstate,
                 (SELECT count(*) FROM event.dbo.all_events WHERE state = 'IN_PROGRESS')
                 AS in_progress,
                 (SELECT count(*) FROM event.dbo.all_events WHERE state = 'RESOLVED') AS
                 resolved,
                 (SELECT count(*) FROM event.dbo.all_events WHERE state = 'CLOSED' AND
                 time_received > '<$DATA:timestamp>') AS closed,
                 'OMi Prod' AS OMi"
      PARAM "resultsetsize"  "100"
      PARAM "fetchsize"  "100"
      GROUP "connectproperties"
         GROUP "field"
            PARAM "name"  "SomeProperty"
            PARAM "value"  "someValue"
         GROUP_END
      GROUP_END
   GROUP_END
   GROUP "data"
      GROUP "field"
         PARAM "name"  "timestamp"
         PARAM "value"  "2020-09-01 10:00:00"
      GROUP_END
   GROUP_END
   PARAM "sourceTag"  "SomeTag"
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Classpath	`PARAM "jdbcclasspath"`	`"<user-defined>"`
JDBC Driver Class	`PARAM "jdbcdriver"`	`"<fully qualified class name (FQCN) of the JDBC driver>"`

Connect string	`PARAM "connectstring"`	`"<The URL to a database connection>"`
Username	`PARAM "username"`	`"<user-defined>"`
Password	`PARAM "password"`	`"<user-defined>"`
Enable Policy Parameters in password field	`n/a`	
Polling interval	`PARAM "interval"`	`"<user-defined>"` following d/h/m/s syntax, e.g. `"3d5m0s"` or `"2h"` Lowest possible value: 3 sec, default 5 min
Property	`PARAM "name"` within `GROUP "connectproperties"`	`"<user-defined>"`
Value	`PARAM "value"`	`"<user-defined>"`
SQL statement	`PARAM "sql"`	`"<user-defined>"`
Result column name	`PARAM "name"` within `GROUP "field"`	`"<user-defined>"`
Initial value	`PARAM "value"`	`"<user-defined>"`
Initial value statement	`PARAM "sqlinit"`	`"<user-defined>"`
Fetch size	`PARAM "fetchsize"`	`"<user-defined>"`, default 100 rows if not specified
Result set size	`PARAM "resultsetsize"`	`"<user-defined>"`, default 100 rows if not specified
Data Source Tag	`PARAM "sourceTag"`	`"<user-defined>"`

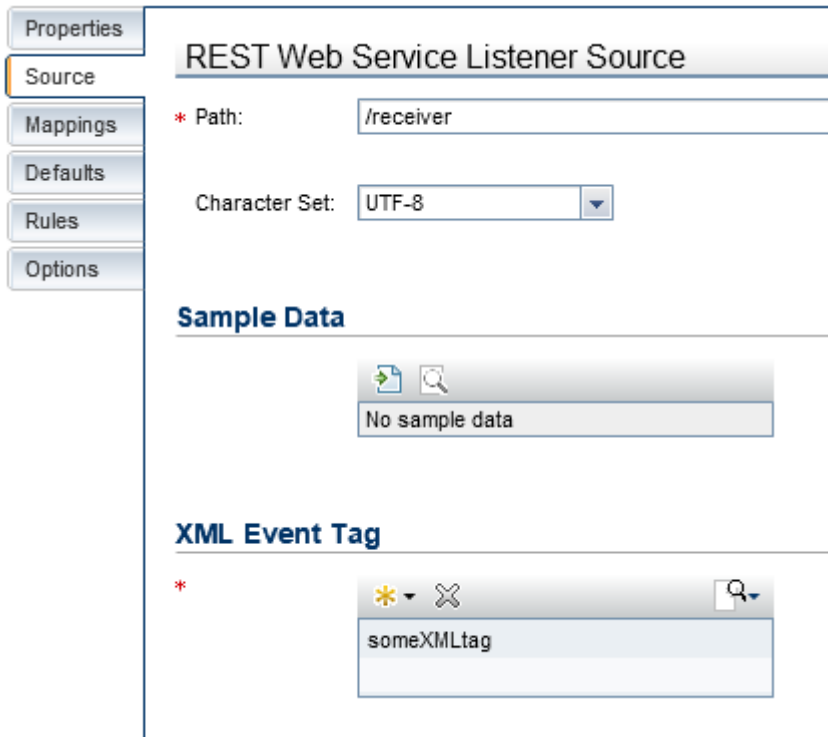
Usage Reference

The Database source section is used in the syntax of the following policy types (click the link to navigate back):

- Syntax of Event from DB policy template
- Syntax of Metric from DB policy template
- Syntax of Generic Output from DB policy template

28. REST Web Service Listener Source section

UI Reference



For details, see the REST Web Service Listener Source topic.

Example of corresponding policy syntax

```
GROUP "sources"
   PARAM "logpath"  "/receiver"
   PARAM "chSet"  "69"
   GROUP "roots"
      GROUP "rootPair"
         PARAM "root"  "someXMLtag"
      GROUP_END
   GROUP_END
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Path	`PARAM "logpath"`	`"<user-defined>"` Path that is appended to OA/Connector listener URL Only the following characters are allowed in a path: [a-zA-Z0-9()-=*.?;,+/:&_]
Character Set	`PARAM "chSet"`	See the table below
XML Event Tag	`PARAM "root"`	`"<user-defined>"` must match XML element name of element you want to process

Character Set	Value	Character Set	Value	Character Set	Value
ACP 1250	30	ISO 8859-5	9	OEMCP 861	21
ACP 1251	29	ISO 8859-6	10	OEMCP 862	22
ACP 1252	4	ISO 8859-7	11	OEMCP 863	23
ASCII	0	ISO 8859-8	12	OEMCP 864	24
Big-5 Taiwanese	73	ISO 8859-9	13	OEMCP 865	25
EBCDIC	3	OEMCP 437	7	OEMCP 866	26
EUC Japanese	65	OEMCP 737	15	OEMCP 869	27
EUC Korean	70	OEMCP 775	16	ROMAN 8	2
EUC Taiwanese	71	OEMCP 850		Shift-JIS	65
GB-2312-80 Chinese	72	OEMCP 852	17	TIS 620	14
ISO 8859-1	1	OEMCP 855	18	UCS-2	67
ISO 8859-15	28	OEMCP 857	19	UTF-8	69
ISO 8859-2	8	OEMCP 860	20		

Usage Reference

The REST Web Service Listener source section is used in the syntax of the following policy types (click the link to navigate back):

- Syntax of Event from REST Web Service Listener policy template
- Syntax of Metric from REST Web Service Listener policy template
- Syntax of Generic Output from REST Web Service Listener policy template

29. REST Web Service Listener Source section (topology)

UI Reference

For more information, see the Configure REST Web Service Listener Policies topic.

Example of corresponding policy syntax

```
GROUP "sources"
    PARAM "topopath"  "/listener"
    PARAM "chSet"  "69"
    PARAM "ageToDeletion"  "3"
    PARAM "doDeltaDetection"  "1"
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Path	`PARAM "topopath"`	`"<user-defined>"` Path that is appended to OA/Connector listener URL. Only the following characters are allowed in a path: [a-zA-Z0-9()-=*.?;,+/:&_]
-	`PARAM "chSet"`	`"69"` (UTF-8)
Age to Deletion	`PARAM "ageToDeletion"`	`"<user-defined>"` After this number of policy executions, a CI that is no longer part of the data is deleted from the server.
Detect Deltas	`PARAM "doDeltaDetection"`	`"1"` (selected) or `"0"` (unselected)

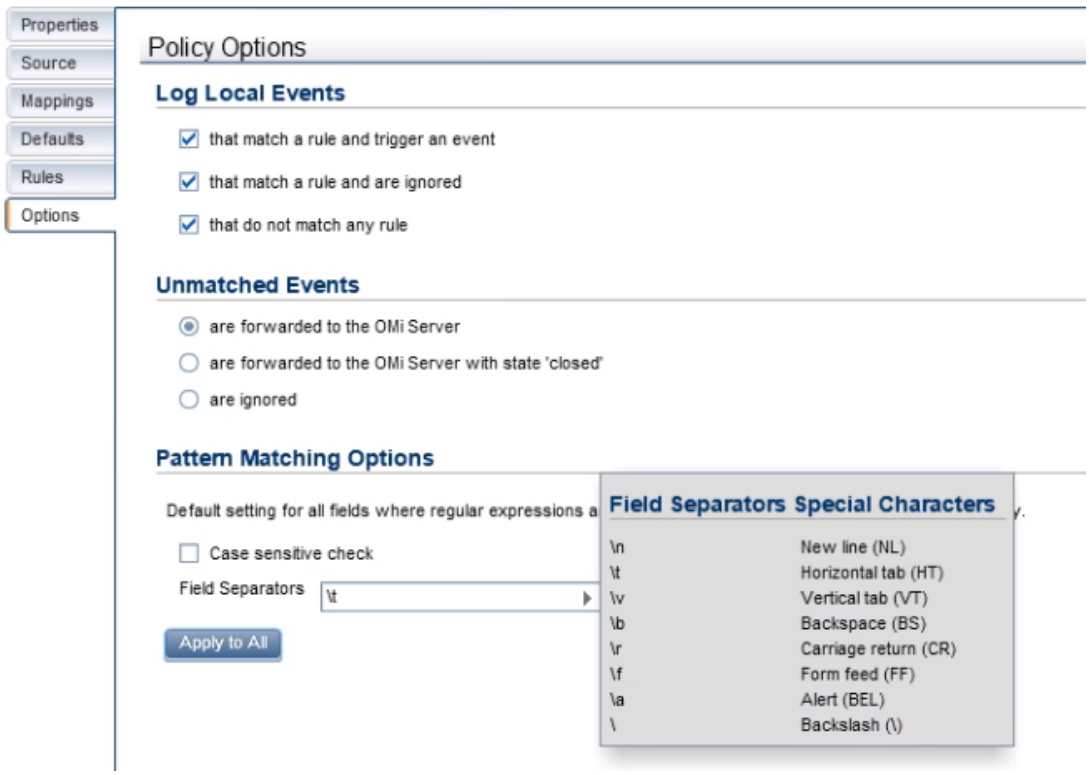
Usage Reference

The REST Web Service Listener source (topology) section is used in the syntax of the following policy types (click the link to navigate back):

- Syntax of Topology from REST Web Service Listener policy template

30. Event Options section (Windows Management Interface)

UI Reference



For details, see the Options section in the Configure Windows Management Interface Policies topic.

Example of corresponding policy syntax

```
ICASE
LOGMATCHEDMSGCOND
LOGMATCHEDSUPPRESS
LOGUNMATCHED
FORWARDUNMATCHED
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
that match a rule and trigger an event	LOGMATCHEDMSGCOND	-
that match a rule and are ignored	LOGMATCHEDSUPPRESS	-
do not match any rule	LOGUNMATCHED	-
are forwarded to the OMi Server or are forwarded to the OMi Server with state "closed" or are ignored	FORWARDUNMATCHED or UNMATCHEDLOGONLY or Do not specify anything	-

Case sensitive check	ICASE	Use ICASE for case insensitive check
Field Separators	SEPARATORS	See the table below

Field Separator in UI	Value in RPE
\n (New line (NL))
\t (Horizontal tab (HT))	
\v (Vertical tab (VT))	&#x0b;
\b (Backspace (BS))	&#x08;
\r (Carriage return (CR))
\f (From feed (FF))	
\a (Alert (BEL))	&#x07;
\ (Backslash (\))	\\;
Space	

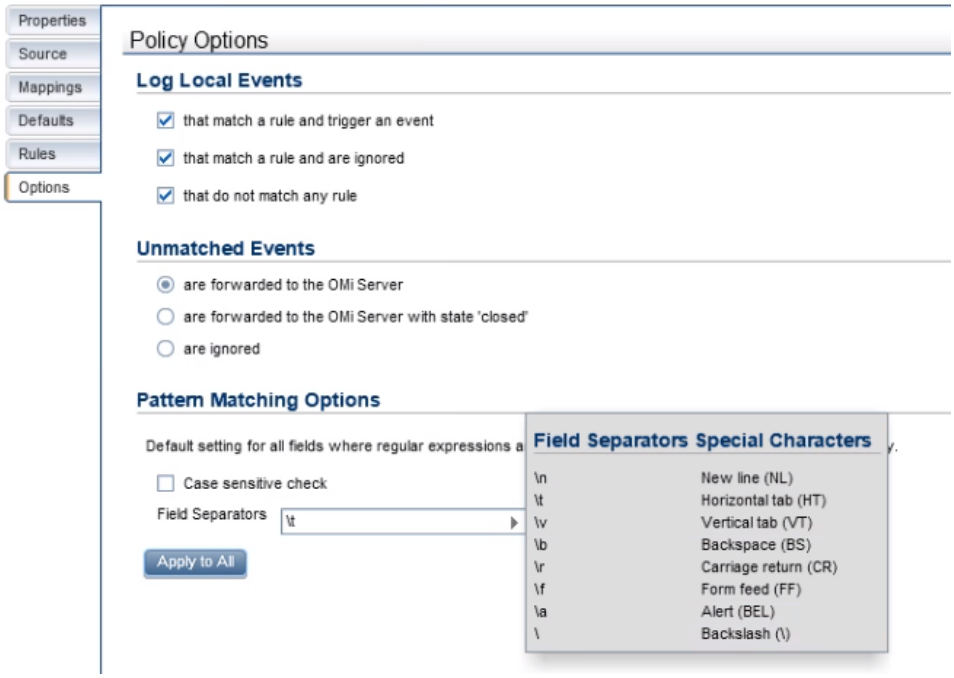
Usage Reference

The Event Options section (Windows Management Interface) is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Windows Management Interface policy template

31. Event Options section

UI Reference



For details, see the Options section in the Configure Structured Log File Policies topic.

Example of corresponding policy syntax

```
GROUP "options"
     PARAM "isForwardUnMatchedMsgToSrv"  "1"
     PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"
     PARAM "fieldSep"  "&#x20;&#x09;"
     PARAM "caseSensitive"  "1"
     PARAM "isLogMatchedCond"  "1"
     PARAM "isLogMatchedSuppressCond"  "1"
     PARAM "isLogUnMatchedCond"  "1"
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
that match a rule and trigger an event	`PARAM "isLogMatchedCond"`	`"1"` (selected) or `"0"` (unselected)
that match a rule and are ignored	`PARAM "isLogMatchedSuppressCond"`	`"1"` or `"0"`
do not match any rule	`PARAM "isLogUnMatchedCond"`	`"1"` or `"0"`

are forwarded to the OMi Server
 	
or
are forwarded to the OMi Server with state "closed"
	`PARAM "isForwardUnMatchedMsgToSrv" "1"`
	`PARAM "isLogOnlyUnMatchedMsgToSrv" "0"`
	
	or
	`PARAM "isForwardUnMatchedMsgToSrv" "0"`
	`PARAM "isLogOnlyUnMatchedMsgToSrv" "1"`	

or are ignored	or Do not specify anything	
Case sensitive check	`PARAM "caseSensitive"`	`"1" or "0"`
Field Separators	`PARAM "fieldSep"`	See the table below

Field Separator in UI	Value in RPE
\n (New line (NL))
\t (Horizontal tab (HT))	
\v (Vertical tab (VT))	&#x0b;
\b (Backspace (BS))	&#x08;
\r (Carriage return (CR))
\f (From feed (FF))	
\a (Alert (BEL))	&#x07;
\ (Backslash (\))	\\;
Space	

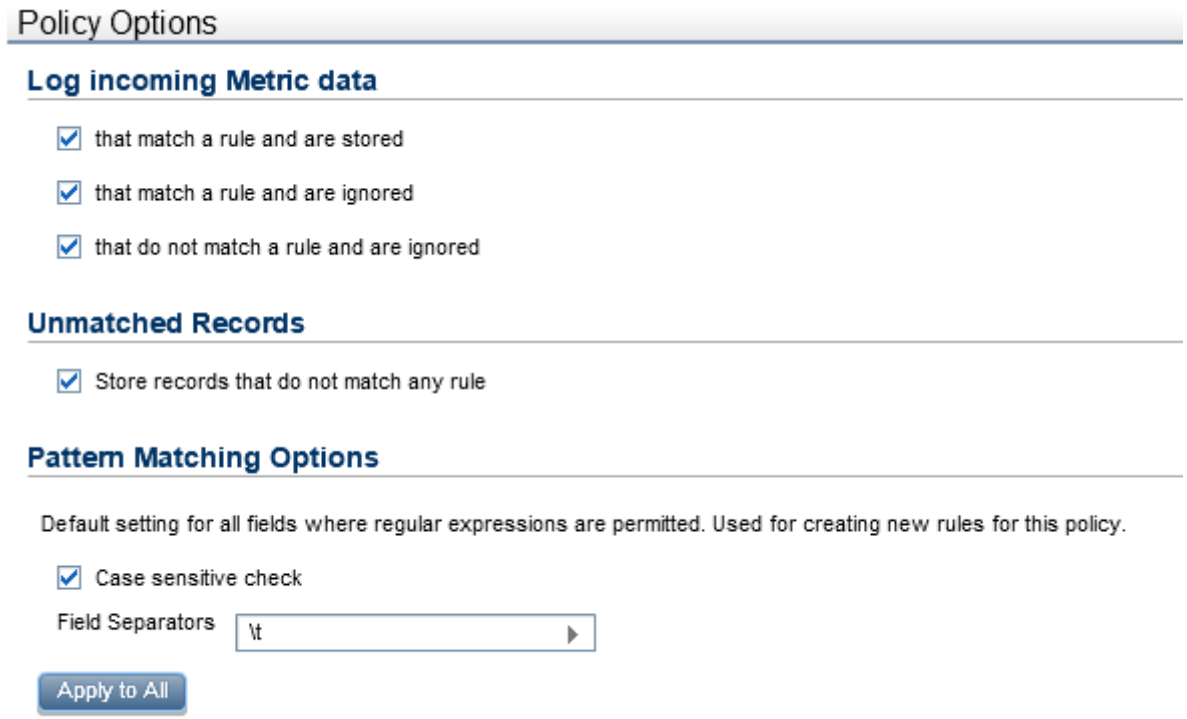
Usage Reference

The Event Options section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Event from Structured Log File policy template
- Syntax of Event from XML File policy template
- Syntax of Event from Perl Script policy template
- Syntax of Event from DB policy template
- Syntax of Event from REST Web Service Listener policy template

32.Metric Options section

UI Reference



For more information, see the Metric Options section in the Configure Structured Log File Policies topic.

Example of corresponding policy syntax

```
GROUP "options"
     PARAM "isForwardUnMatchedMsgToSrv"  "1"
     PARAM "isLogOnlyUnMatchedMsgToSrv"  "0"
     PARAM "fieldSep"  "&#x20;&#x09;"
     PARAM "caseSensitive"  "1"
     PARAM "isLogMatchedCond"  "1"
     PARAM "isLogMatchedSuppressCond"  "1"
     PARAM "isLogUnMatchedCond"  "1"
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
that match a rule and trigger an event	`PARAM "isLogMatchedCond"`	`"1"` (selected) or `"0"` (unselected)
that match a rule and are ignored	`PARAM "isLogMatchedSuppressCond"`	`"1" or "0"`
do not match any rule	`PARAM "isLogUnMatchedCond"`	`"1" or "0"`
are forwarded to the OMi Server	`PARAM "isForwardUnMatchedMsgToSrv"`	`"1" or "0"`
are forwarded to the OMi Server with state "closed"	`PARAM "isLogOnlyUnMatchedMsgToSrv"`	`"1" or "0"`

Case sensitive check	PARAM "caseSensitive"	"1" or "0"
Field Separators	PARAM "fieldSep"	See the table below

Field Separator in UI	Value in RPE
\n (New line (NL))
\t (Horizontal tab (HT))	
\v (Vertical tab (VT))	&#x0b;
\b (Backspace (BS))	&#x08;
\r (Carriage return (CR))
\f (From feed (FF))	
\a (Alert (BEL))	&#x07;
\ (Backslash (\))	\\;
Space	

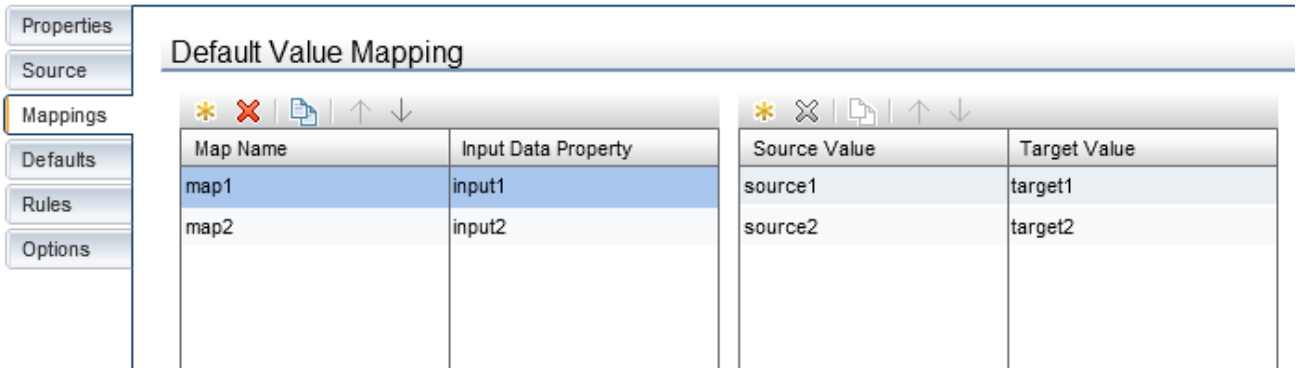
Usage Reference

The Options section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Metric from Structured Log File policy template
- Syntax of Metric from XML File policy template
- Syntax of Metric from Perl Script policy template
- Syntax of Metric from REST Web Service Listener policy template
- Syntax of Metric from DB policy template

33.Mappings section

UI Reference



For details, see the Mappings (Events and Metrics) section in the Configure Structured Log File Policies topic.

Example of corresponding policy syntax

```
MAP "map1"
 DESCRIPTION ""
 INPUT "input1"
   FROM "source1" TO "target1"
   FROM "source2" TO "target2"
 MAP "map2"
 DESCRIPTION ""
 INPUT "input2"
   FROM "source3" TO "target3"
   FROM "source4" TO "target4"
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Map Name	`MAP`	`"<user defined>"`
Input Data Property	`INPUT`	`"<user defined>"` must match some property of input data
Source Value / Target Value	`FROM "<source>" TO "<target>"`	`"<user defined source>"` must match values of input data and `"<user defined target>"`

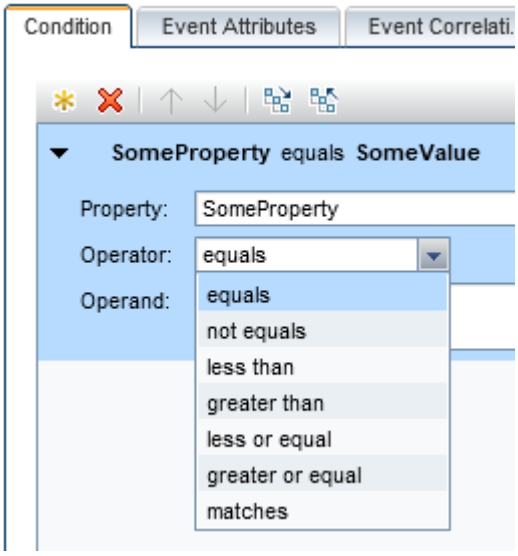
Usage Reference

The Mappings section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Event from Structured Log File policy template
- Syntax of Event from XML File policy template
- Syntax of Event from Perl Script policy template
- Syntax of Event from DB policy template
- Syntax of Event from REST Web Service Listener policy template
- Syntax of Metric from Structured Log File policy template
- Syntax of Metric from XML File policy template
- Syntax of Metric from Perl Script policy template
- Syntax of Metric from DB policy template
- Syntax of Metric from REST Web Service Listener policy template

34. Condition section

UI Reference



For details, see the Condition section in the Configure Structured Log File Policies topic.

Example of corresponding policy syntax

```
CONDITION
     "SomeProperty" == "SomeValue"
```

Important Note:
In a condition, do not use the <$DATA:<someProperty>> syntax to refer to a property, but specify the property directly as shown in this table:

Policy template type	Reference	Example source-specific extraction of properties	Example reference as part of condition
Structured Log File	variable	PARAM "struliPattern" "<#.EventId>,<*.Description>"	"EventId" == "Sec514" "Description" == "SomeValue"
Database	columnname	PARAM "sql" "SELECT count(*) FROM event.dbo.all_events WHERE state = 'IN_PROGRESS') AS in_progress"	"in_progress" >= 0
XML File and REST Web Service Listener	path to XML property from Root	Example XML content: <?xml version="1.0" encoding="UTF-8"?> <Event> <EventName>EndRequest</EventName> <Category>Administrative</Category> </Event>	"/Event/EventName" == "EndRequest" "/Event/Category" == "Administrative"
Perl	AttributeName	Example perl transfer data structure: my %BASIC_METRIC_DATA = ("counter_name" => "<Value>", "counter_value" => "<Value>");	"counter_name" == "someName" "counter_value" <= 10

WMI	`instance property`	WMI either returns a WMI instance or a WMI event with a TargetInstance. Properties of both can be accessed in the same way	`"__CLASS" == "SomeClassName"`

Mapping of UI elements to Syntax elements

UI element	Syntax element
equals	`==`
not equals	`!=`
less than	`<`
greater than	`>`
less or equal	`<=`
greater or equal	`>=`
matches	`~=`

Usage Reference

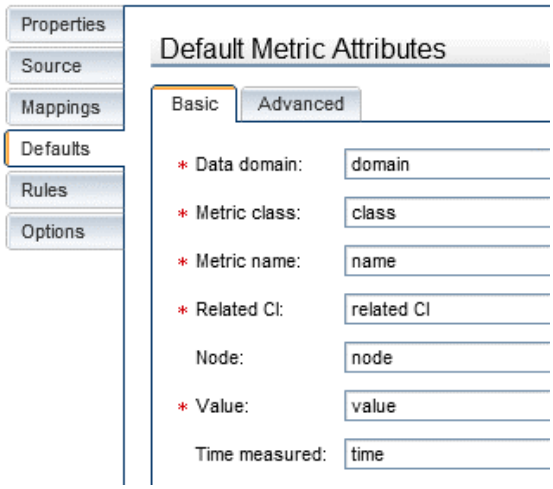
The Condition section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Windows Management Interface policy template
- Syntax of Event from Structured Log File policy template
- Syntax of Event from XML File policy template
- Syntax of Event from Perl Script policy template
- Syntax of Event from DB policy template
- Syntax of Event from REST Web Service Listener policy template
- Syntax of Metric from Structured Log File policy template
- Syntax of Metric from XML File policy template
- Syntax of Metric from Perl Script policy template
- Syntax of Metric from DB policy template
- Syntax of Metric from REST Web Service Listener policy template

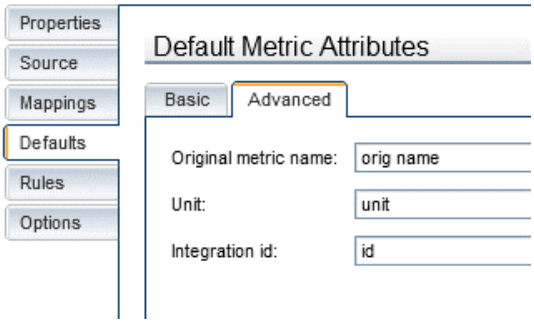
35.Set Metric Attributes section

UI Reference

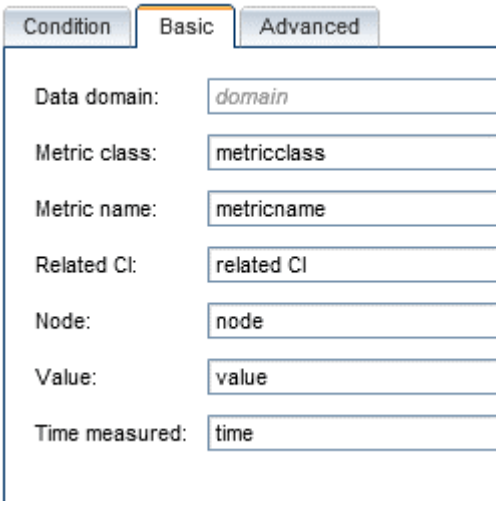
Default - Basic



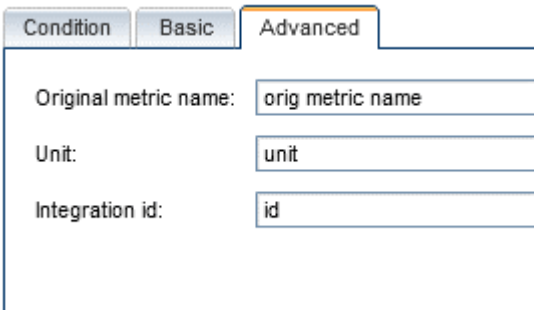
Default - Advanced



UI As part of Rule - Basic



UI as part of Rule - Advanced



For details on the Advanced and Basic tabs, see UI Reference topic.

Example of corresponding policy syntax

```
DEFAULTMSG
          SERVERLOGONLY "false"
          CUSTOM  "dataDomain"  "domain"
          CUSTOM  "relatedCi"  "related CI"
          CUSTOM  "metricClass"  "class"
          CUSTOM  "name"  "name"
          CUSTOM  "originalName"  "orig name"
          CUSTOM  "sampleValue"  "value"
          CUSTOM  "unit"  "unit"
          CUSTOM  "node"  "node"
          CUSTOM  "timeMeasured"  "time"
          CUSTOM  "integrationId"  "id"
```

Or as part of a rule:

```
SET
        CUSTOM  "relatedCi"  "related CI"
        CUSTOM  "metricClass"  "metricclass"
        CUSTOM  "name"  "metricname"
        CUSTOM  "originalName"  "orig metric name"
        CUSTOM  "sampleValue"  "value"
        CUSTOM  "unit"  "unit"
        CUSTOM  "node"  "node"
        CUSTOM  "timeMeasured"  "time"
        CUSTOM  "integrationId"  "id"
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Data Domain	`CUSTOM "dataDomain"`	`"<user-defined>"`
Metric class	`CUSTOM "metricClass"`	`"<user-defined>"`
Metric name	`CUSTOM "name"`	`"<user-defined>"`
Related CI	`CUSTOM "relatedCi"`	`"<user-defined>"` see Best_practices_for_related_CIs
Node	`CUSTOM "node"`	`"<user-defined>"` FQDN of node
Value	`CUSTOM "sampleValue"`	`"<user-defined>"`
Time measured	`CUSTOM "timeMeasured"`	`"<user-defined>"` for time formats see Basic Tab
Original metric name	`CUSTOM "originalName"`	`"<user-defined>"`

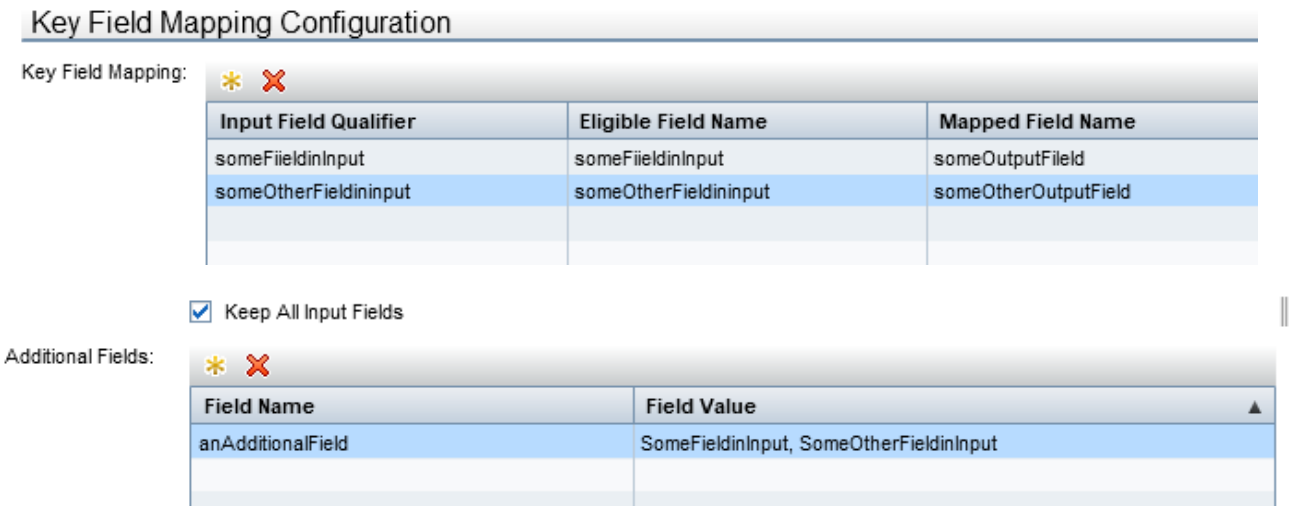
Usage Reference

The Set Metrics section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Metric from Structured Log File policy template
- Syntax of Metric from XML File policy template
- Syntax of Metric from Perl Script policy template
- Syntax of Metric from DB policy template
- Syntax of Metric from REST Web Service Listener policy template

36. Schema section (Generic Output)

UI Reference



For details, see the Schema section topic.

Example of corresponding policy syntax

```
GROUP "KEYMAPRULE"
   PARAM "KEEP_INPUT_FIELDS"  "1"
   GROUP "KEYFIELD_MAPPINGS"
      PARAM "KF_MAP"  "KEEP"
      ATT "KEY_IN"  "someFieldinInput"
      ATT "KEY_OUT"  "someOutputFileld"
      PARAM "KF_MAP"  "KEEP"
      ATT "KEY_IN"  "someOtherFieldinInput"
      ATT "KEY_OUT"  "someOtherOutputField"
   GROUP_END
   GROUP "ADDITIONAL_FIELDS"
      PARAM "ADD_FIELD"  "ADD"
      ATT "NAME"  "anAdditionalField"
      ATT "VALUE"  "SomeFieldinInput, SomeOtherFieldinInput"
   GROUP_END
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Input Field Qualifier	`ATT "KEY_IN"`	`"<user defined source>"` must match a field in input data
Mapped Field Name	`ATT "KEY_OUT"`	`"<user defined source>"` defines field name in output data
Keep All Input Fields	`PARAM "KEEP_INPUT_FIELDS"`	`"1"` (selected) or `"0"` (unselected)
Additional Fields	-	-
Field Name	`ATT "NAME"`	`"<user defined>"` defines additional field name in output data
Field Value	`ATT "VALUE"`	`"<user defined>"` The field value can include a combination of user-defined strings, policy parameters, field names, and operators.

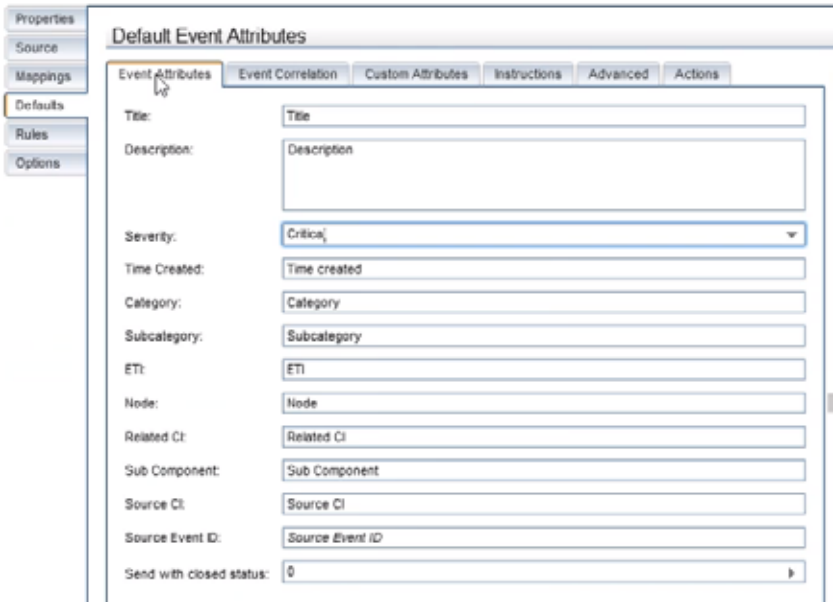
Usage Reference

The Schema section is used in the syntax of the following policy types (click on link to navigate back):

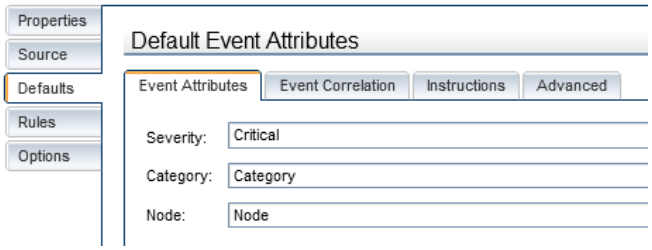
- Syntax of Generic Output from XML File policy template
- Syntax of Generic Output from Perl Script policy template
- Syntax of Generic Output from DB policy template
- Syntax of Generic Output from REST Web Service Listener policy template

37.Set Event Attributes section

UI Reference – Default Event Attributes

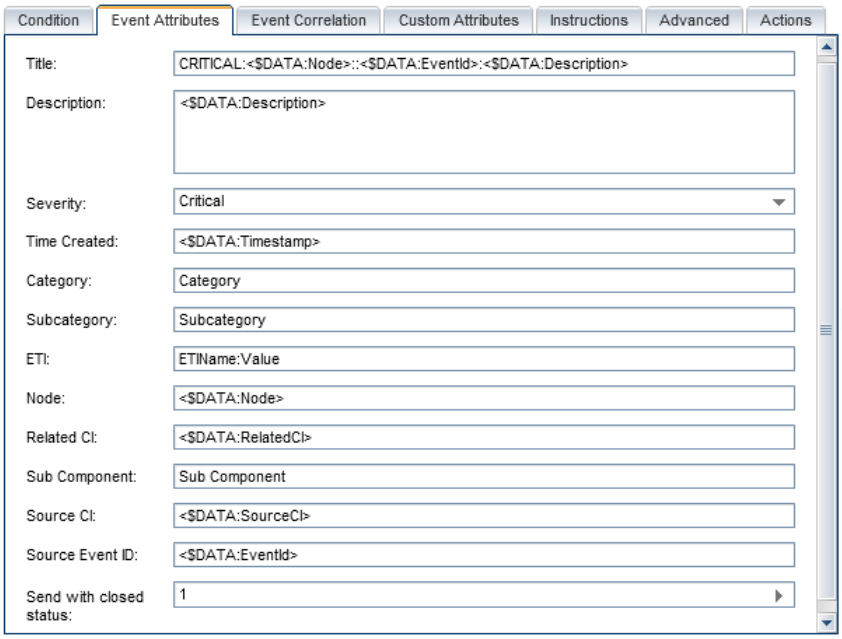


UI Reference – Default Event Attributes – Windows Management Interface



This policy type does not allow to set any other attributes in the Defaults section.

UI Reference – Event Attributes as part of a rule



Example of corresponding policy syntax (from a rule)

Please see High-level usage guidelines for details about <$DATA:> references.

```
SERVERLOGONLY "true"
SEVERITY "Critical"
TIMECREATED "<$DATA:Timestamp>"
NODE IP 0.0.0.0  "<$DATA:Node>"
APPLICATION "<DATA:application>"
MSGGRP "Category"
OBJECT "<$DATA:object>"
MSGTYPE "<$DATA:type>"
SERVICE_NAME "<DATA:HPOM service ID>"
MSGKEY "<$DATA:Node>:<$DATA:EventId>"
MSGKEYRELATION ACK  "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS "
CUSTOM  "Description"  "<$DATA:Description>"
CUSTOM  "SubCategory"  "Subcategory"
CUSTOM  "RelatedCiHint"  "<$DATA:RelatedCI>"
CUSTOM  "NoDuplicateSuppression"  "false"
CUSTOM  "EtiHint"  "ETIName:Value"
CUSTOM  "SourceCiHint"  "<$DATA:SourceCI>"
CUSTOM  "SubCiHint"  "Sub Component"
CUSTOM  "SourcedFromExternalId" "<$DATA:EventId>"
CUSTOM  "CA_1"  "2"
CUSTOM  "SourcedFromExternalUrl"  "https://default.example.com/event/<$DATA:EventId>"
TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"
```

Note: The lines can appear in any sequence. The shown sequence corresponds with the sequence used in the example policy.

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Title	`TEXT`	`"<user-defined>"`
Description	`CUSTOM  "Description"`	`"<user-defined>"`

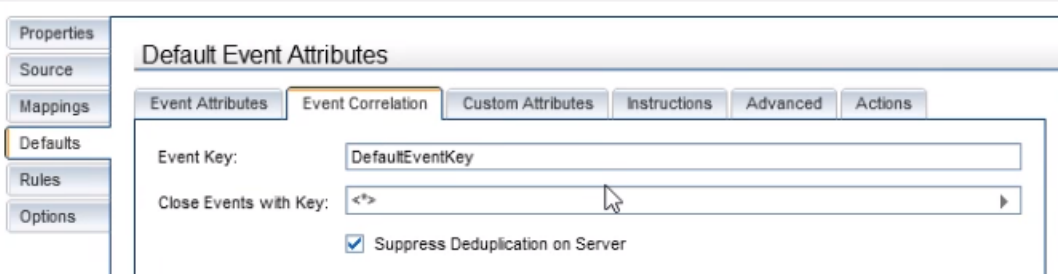
Severity	`SEVERITY`	`Critical`
		`Warning`
		`Major`
		`Minor`
		`Normal`

Time created	`TIMECREATED`	`"<user-defined>"` for allowed time formats, see Time created.
Category	`MSGGRP`	`"<user-defined>"`
Subcategory	`CUSTOM  "SubCategory"`	`"<user-defined>"`
ETI	`CUSTOM  "EtiHint"`	`"<user-defined>"`

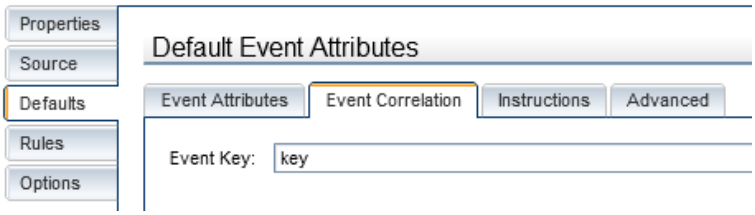
Node	`NODE IP 0.0.0.0 "<fqdn>"`
	`Or`
	`NODE IP <ip address>`
		`"<user-defined fqdn>" or`
		`"<user-defined ip address>"`
		`Examples:`
		`NODE IP 0.0.0.0 "node.example.com"`
		`Or`
		`NODE IP 10.10.10.10`

Related CI	CUSTOM "RelatedCiHint"	"<user-defined>" Use the format <CI Hint 1>:<CI Hint 2>:...:<CI Hint n>@@<FQDN>
Sub component	CUSTOM "SubCiHint"	"<user-defined>" search for Sub Component in online-help
Source CI	CUSTOM "SourceCiHint"	"<user-defined>" Use the format <CI Hint 1>:<CI Hint 2>:...:<CI Hint n>@@<FQDN>
Source Event ID	CUSTOM "SourcedFromExternalId"	"<user-defined>"
Send with closed status	SERVERLOGONLY	"true" or "false"

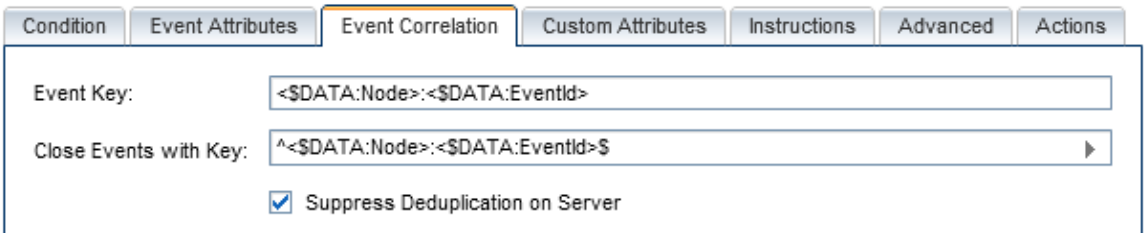
UI Reference – Default Event Correlation



UI Reference – Default Event Correlation – Windows Management Interface



UI Reference – Event Correlation as part of a rule



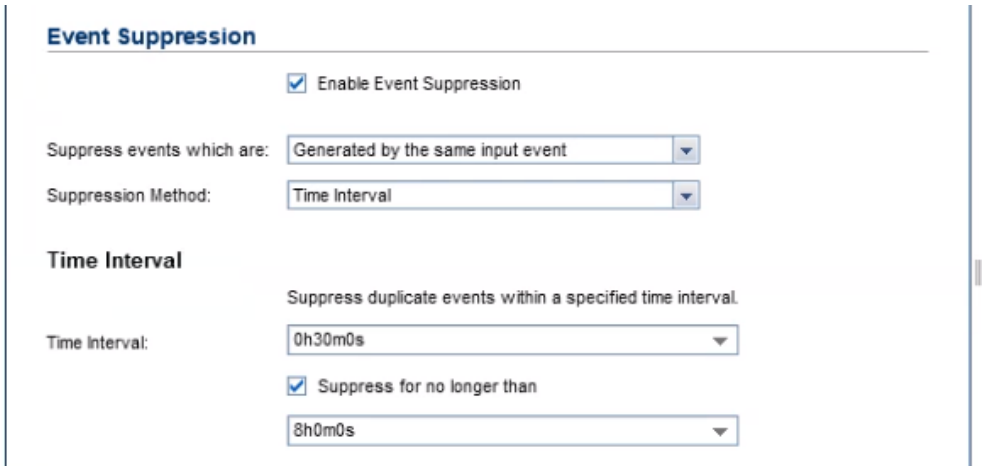
Examples of corresponding policy syntax (from a rule)

```
SERVERLOGONLY "true"
SEVERITY "Critical"
TIMECREATED "<$DATA:Timestamp>"
NODE IP 0.0.0.0  "<$DATA:Node>"
APPLICATION "<DATA:application>"
MSGGRP "Category"
OBJECT "<DATA:object>"
MSGTYPE "<DATA:type>"
SERVICE_NAME "<DATA:HPOM service ID>"
MSGKEY "<$DATA:Node>:<$DATA:EventId>"
MSGKEYRELATION ACK  "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS "  "
CUSTOM  "Description"  "<$DATA:Description>"
CUSTOM  "SubCategory"  "Subcategory"
CUSTOM  "RelatedCiHint"  "<$DATA:RelatedCI>"
CUSTOM  "NoDuplicateSuppression"  "true"
CUSTOM  "EtiHint"  "ETIName:Value"
CUSTOM  "SourceCiHint"  "<$DATA:SourceCI>"
CUSTOM  "SubCiHint"  "Sub Component"
CUSTOM  "SourcedFromExternalId" "<$DATA:EventId>"
CUSTOM  "CA_1"  "2"
CUSTOM  "SourcedFromExternalUrl"  "https://default.example.com/event/<$DATA:EventId>"
TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"
```

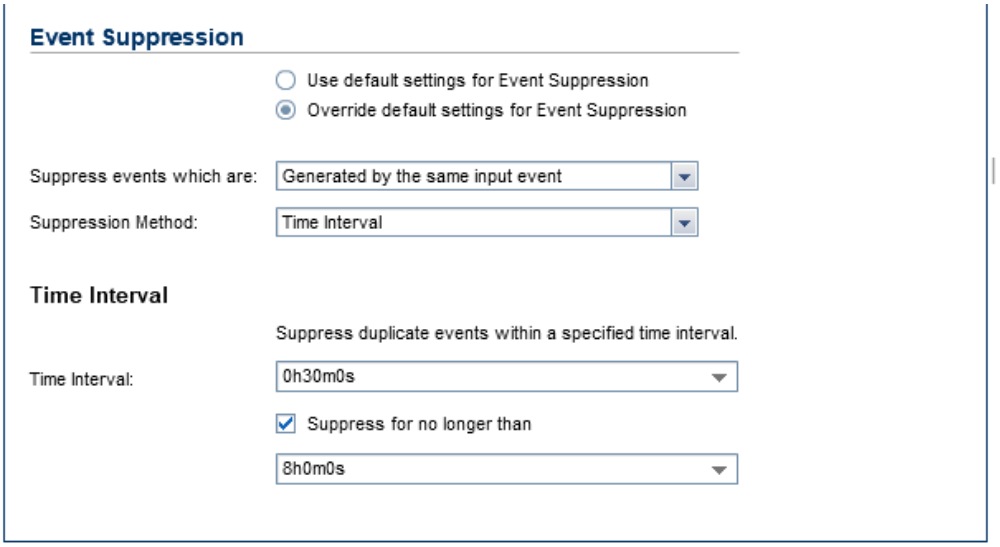
Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Event Key	`MSGKEY`	`"<user-defined>"` see online-help
Close Events with Key	`MSGKEYRELATION ACK ... SEPARATORS " "`	`"<user-defined>"` see online-help
Suppress Deduplication on Server	`CUSTOM "NoDuplicateSuppression"`	`"true"` or `"false"`

UI Reference – Default Event Suppression



UI Reference – Event Suppression as part of a rule



Examples of corresponding policy syntax (from a rule)

```
SUPP_DUPL_IDENT
       "0h30m0s" RESEND "8h0m0s"
```

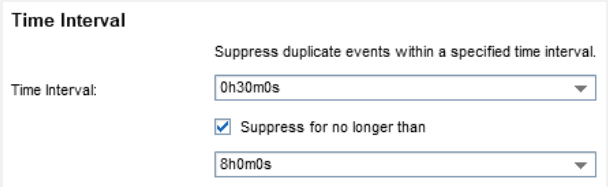
OR

```
SUPP_DUPL_COND
        "0h30m0s" RESEND "8h0m0s"
        COUNTER_THRESHOLD 3
        RESET_COUNTER_INTERVAL "24h0m0s"
```

OR

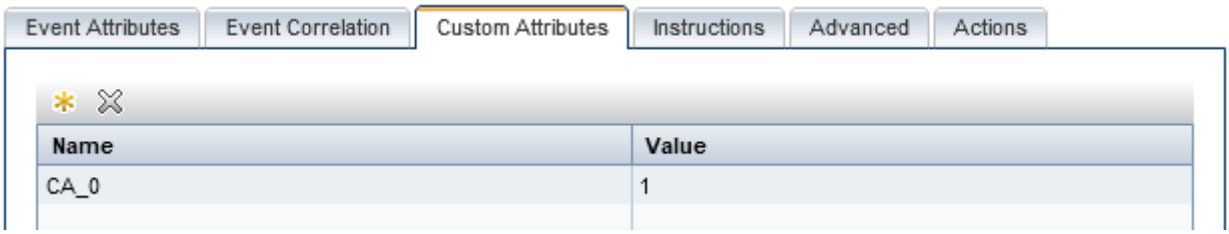
```
SUPP_DUPL_IDENT_OUTPUT_MSG
         COUNTER_THRESHOLD 3
         RESET_COUNTER_INTERVAL "24h0m0s"
```

Mapping of UI elements to Syntax elements

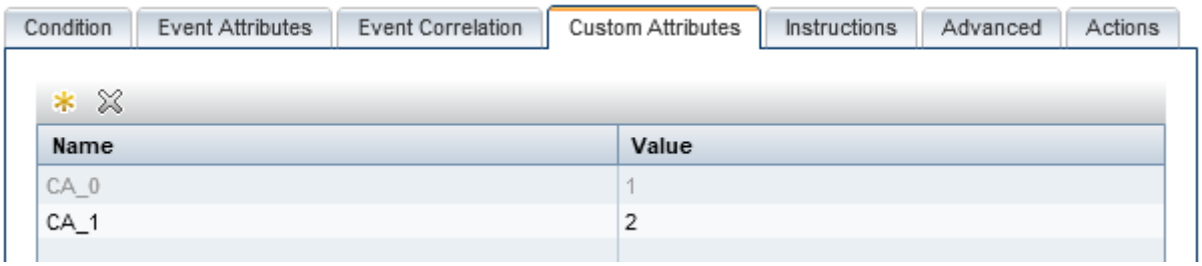
UI element	Syntax element	Possible values
Suppress event which are: Generated by same input event	`SUPP_DUPL_IDENT`	-
Suppress event which are: Generated by same rule	`SUPP_DUPL_COND`	-
Suppress event which are: Identical relative to their attributes	`SUPP_DUPL_IDENT_OUTPUT_MSG`	-
Suppression method: Time interval 	`"0h30m0s" RESEND "8h0m0s"` Omit `Resend ""` if *Suppress for no longer than* is not selected	Specify interval using d/h/m/s syntax, e.g. `"3d5m0s"` or `"2h"`
Suppression method: Time interval / Counter 	`"0h30m0s" RESEND "8h0m0s" COUNTER_THRESHOLD 3 RESET_COUNTER_INTERVAL "24h0m0s"` Omit `RESET_COUNTER_INTERVAL "24h0m0s"` if *Reset Counter threshold after* is not selected	Specify interval using d/h/m/s syntax, e.g. `"3d5m0s"` or `"2h"` Specify counter threshold as integer
Suppression method: Counter	`COUNTER_THRESHOLD 3 RESET_COUNTER_INTERVAL "24h0m0s"`	Specify counter threshold as integer

	Omit `RESET_COUNTER_INTERVAL "24h0m0s"` if *Reset Counter threshold after* is not selected	

UI Reference – Default Custom Attributes



UI Reference – Custom Attributes as part of a rule



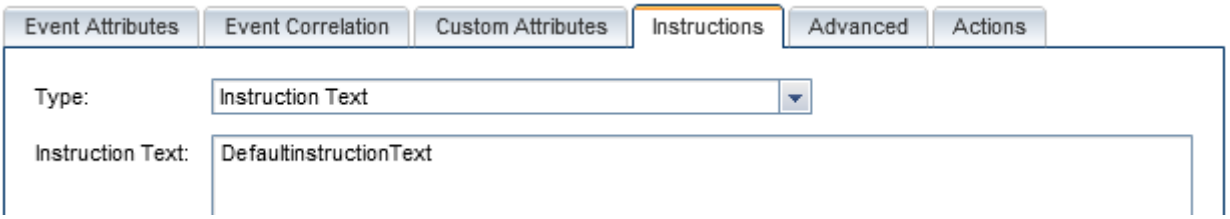
Examples of corresponding policy syntax (from a rule)

```
SERVERLOGONLY "true"
SEVERITY "Critical"
TIMECREATED "<$DATA:Timestamp>"
NODE IP 0.0.0.0  "<$DATA:Node>"
APPLICATION "<DATA:application>"
MSGGRP "Category"
OBJECT "<DATA:object>"
MSGTYPE "<DATA:type>"
SERVICE_NAME "<DATA:HPOM service ID>"
MSGKEY "<$DATA:Node>:<$DATA:EventId>"
MSGKEYRELATION ACK  "^<$DATA:Node>:<$DATA:EventId>$" SEPARATORS "
CUSTOM  "Description"  "<$DATA:Description>"
CUSTOM  "SubCategory"  "Subcategory"
CUSTOM  "RelatedCiHint"  "<$DATA:RelatedCI>"
CUSTOM  "NoDuplicateSuppression"  "true"
CUSTOM  "EtiHint"  "ETIName:Value"
CUSTOM  "SourceCiHint"  "<$DATA:SourceCI>"
CUSTOM  "SubCiHint"  "Sub Component"
CUSTOM  "SourcedFromExternalId" "<$DATA:EventId>"
CUSTOM  "CA_1"  "2"
CUSTOM  "SourcedFromExternalUrl"  "https://default.example.com/event/<$DATA:EventId>"
TEXT "CRITICAL:<$DATA:Node>::<$DATA:EventId>:<$DATA:Description>"
```

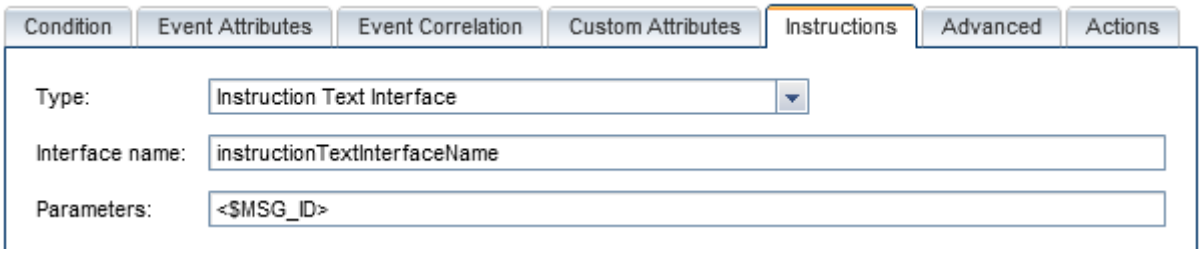
Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Name and Value	`CUSTOM  "user defined CA name" "user defined CA value"`	`<Any user-defined value>`

UI Reference – Default Instructions



UI Reference – Instructions as part of a rule



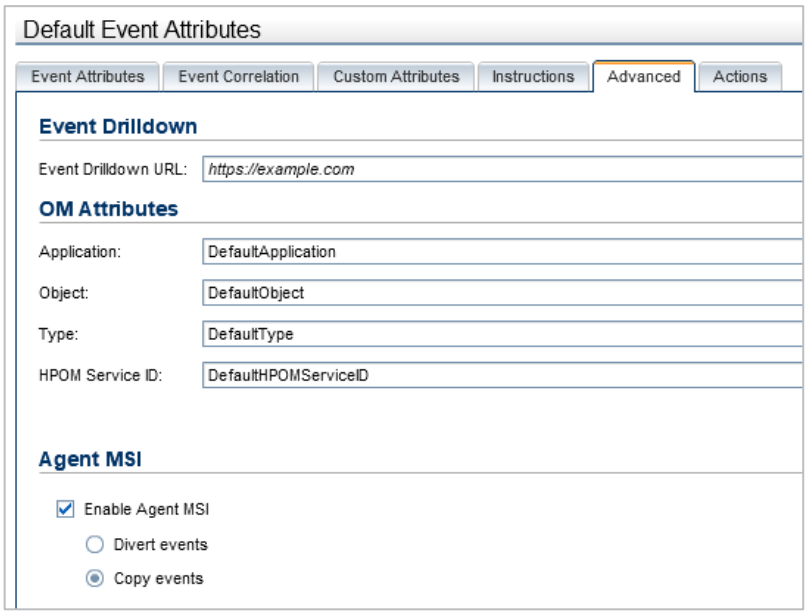
Examples of corresponding policy syntax (from a rule)

```
            INSTRUCTION_TEXT_INTERFACE "instructionTextInterfaceName"
            INSTRUCTION_PARAMETERS "<$MSG_ID>"
OR
            HELPTEXT "hardcoded instruction text"
            HELP "20DFC18A-9B3B-5052-DFC8-11F8D4A8FF05"
```

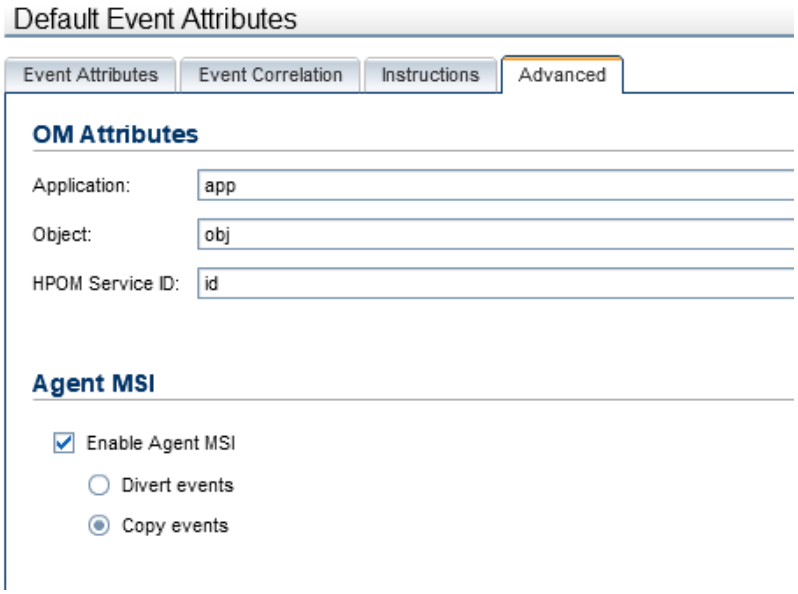
Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Type: Instruction Text		
Instruction Text	HELPTEXT	"<user-defined>"
	HELP	"<user-generated UUID>"
Type: Instruction Text Interface		
	INSTRUCTION_TEXT_INTERFACE	<user-defined interface name> Must map to a defined interface configured in Administration - Operations Console - External Instructions
Parameters	INSTRUCTION_PARAMETERS	"<user-defined>"

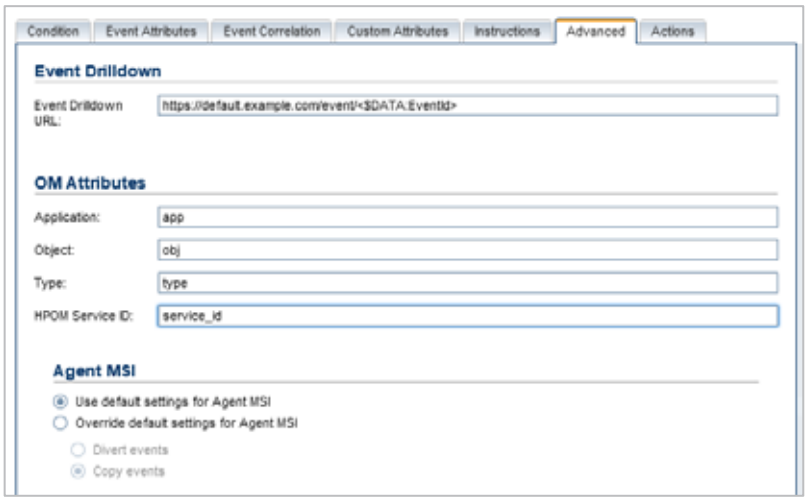
UI Reference – Default Advanced



UI Reference – Default Advanced – Windows Management Interface



UI Reference – Advanced as part of a rule



For details, see the Advanced (Default Event Attributes) section in the Configure Structured Log File Policies topic.

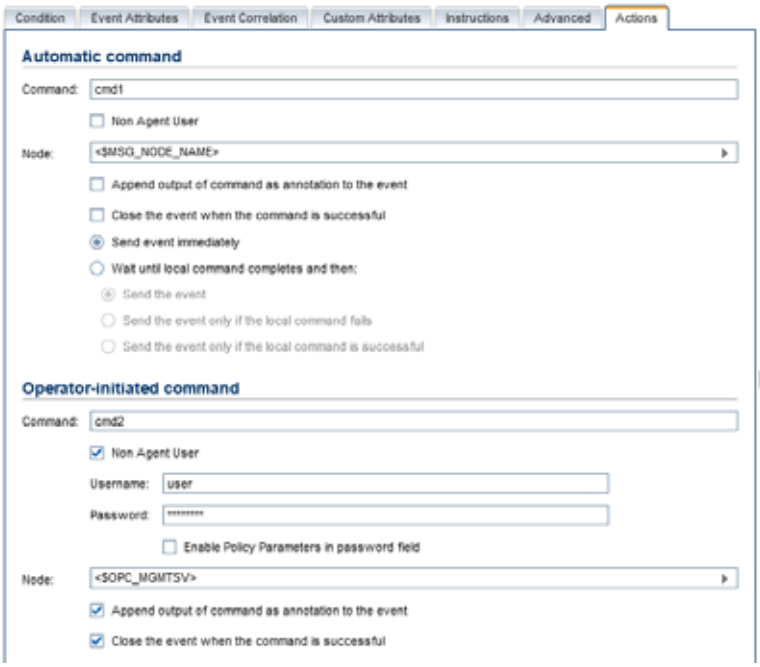
Examples of corresponding policy syntax (from a rule)

```
                CUSTOM  "SourcedFromExternalUrl"  "https://default.example.com/event/<$DATA:EventId>"
                APPLICATION "app"
                OBJECT "obj"
                MSGTYPE "type"
                SERVICE_NAME "service_id"
AND
                MPI_AGT_DIVERT_MSG
OR
                MPI_AGT_COPY_MSG
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Event Drilldown URL	`CUSTOM  "SourcedFromExternalUrl"`	`"<user-defined URL>"`, e.g. `https://default.example.com/event/<$DATA:EventId`
Application	`APPLICATION`	`"<user-defined>"`
Object	`OBJECT`	`"<user-defined>"`
Type	`MSGTYPE`	`"<user-defined>"`
HPOM Service ID	`SERVICE_NAME`	`"<user-defined>"`
Enable Agent MSI	-	-
Use default settings for Agent MSI	-	-
Override default settings for Agent MSI	-	-
Divert events	`MPI_AGT_DIVERT_MSG`	-
Copy events	`MPI_AGT_COPY_MSG`	-

UI Reference – Actions



For details, see the Actions section in the Configure Structured Log File Policies topic.

Examples of corresponding policy syntax (from a rule)

```
              AUTOACTION "cmd1" ACTIONNODE IP 0.0.0.0  "<$MSG_NODE_NAME>"
                         SIGNATURE ""
              OPACTION "cmd2" ACTIONNODE IP 0.0.0.0  "<$OPC_MGMTSV>" ANNOTATE ACK
                         USER "user" PWD "%%ActionUserPassword%%"
                         SIGNATURE ""
OR
              AUTOACTION "cmd1" ACTIONNODE IP 0.0.0.0  "<$MSG_NODE_NAME>"
                           SEND_MSG_AFTER_LOC_AA
                         SIGNATURE ""
OR
              AUTOACTION "cmd1" ACTIONNODE IP 0.0.0.0  "<$MSG_NODE_NAME>"
                           SEND_MSG_AFTER_LOC_AA SEND_FAILED_MSG
                         SIGNATURE ""
OR
              AUTOACTION "cmd1" ACTIONNODE IP 0.0.0.0  "<$MSG_NODE_NAME>"
                           SEND_MSG_AFTER_LOC_AA SEND_OK_MSG
                         SIGNATURE ""
OR
              OPACTION "cmd2" ACTIONNODE IP 0.0.0.0  "<$OPC_MGMTSV>" ACK
                         USER "user" PWD "%%ActionUserPassword%%"
                         SIGNATURE ""
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Command	`AUTOACTION "cmd1" <...> SIGNATURE ""`	`"<user-defined>"`
Non-agent user Username Password		
	`USER`	`"<user-defined>"`
	`PWD`	`"<user-defined>"`, see the **Passwords** section - use password parameter using `%%<parameter_name>%%` syntax

Node	`ACTIONNODE IP 0.0.0.0 "<fqdn>" SIGNATURE ""`
	Or
	`ACTIONNODE IP <ip address> SIGNATURE ""`
		`"<user-defined fqdn>" or "<user-defined ip address>"`
		Examples:
		`ACTIONNODE IP 0.0.0.0 "node.example.com"`
		Or
		`ACTIONNODE IP 10.10.10.10`

Append output of command as annotation to the event	`ANNOTATE`	
Close the event when the command is successful	`ACK`	`"<user-defined>"`
Send event immediately	-	Default
Wait until local command completes and then		
Send the event	`SEND_MSG_AFTER_LOC_AA`	
Send the event only if the local command fails	`SEND_MSG_AFTER_LOC_AA SEND_FAILED_MSG`	
Send the event only if the local command is successful	`SEND_MSG_AFTER_LOC_AA SEND_OK_MSG`	

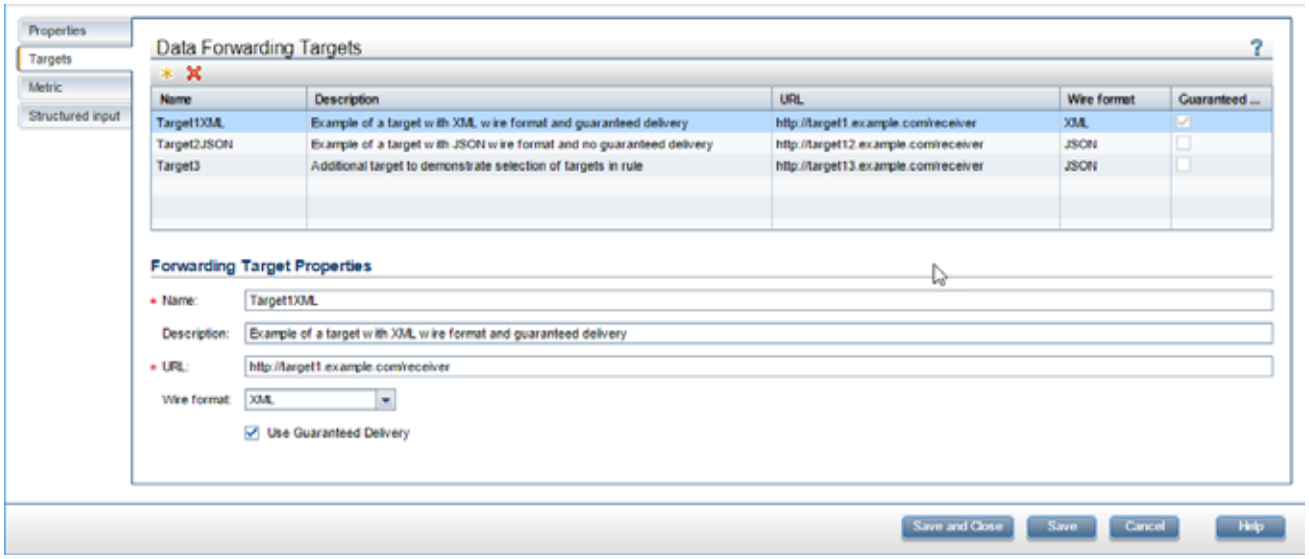
Usage Reference

The Set Event Attributes section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Windows Management Interface policy template
- Syntax of Event from Structured Log File policy template
- Syntax of Event from XML File policy template
- Syntax of Event from Perl Script policy template
- Syntax of Event from DB policy template
- Syntax of Event from REST Web Service Listener policy template

38. Data Forwarding Targets section

UI Reference



For more information, see the Configure Data Forwarding Policies topic.

Example of corresponding policy syntax

```
GROUP "TARGETS"
   GROUP "TARGET"
      PARAM "id"  "7215BE36-10F8-A2AC-9CD1-209FCCFE4A00"
      PARAM "NAME"  "Target1XML"
      PARAM "URL"  "http://target1.example.com/receiver"
      PARAM "WIREFORMAT"  "XML"
      PARAM "SENDMODE"  "SNF"
      PARAM "DESCRIPTION"  "Example of a target with XML wire format …"
   GROUP_END
OR
  GROUP "TARGET"
      PARAM "id"  "F893917D-1B91-7B55-D969-20A0A51F83CF"
      PARAM "NAME"  "Target2JSON"
      PARAM "URL"  "http://target12.example.com/receiver"
      PARAM "WIREFORMAT"  "JSON"
      PARAM "SENDMODE"  "FAF"
      PARAM "DESCRIPTION"  "Example of a target with JSON wire format …"
   GROUP_END
```

UI element	Syntax element	Possible values
Name	`PARAM "NAME"`	`"<user-defined>"` The user defined name of the receiving target. The field is mandatory.
Description	`PARAM "DESCRIPTION"`	`"<user-defined>"` A brief description of the target.
URL	`PARAM "URL"`	`"<user-defined>"` The URL of the REST web service endpoint. The field is mandatory.
Wire format	`PARAM "WIREFORMAT"`	`"XML" or "JSON"` The wire format the target expects.

Use Guaranteed Delivery	PARAM "SENDMODE"	"SNF" (for guaranteed delivery) "FAF" (for non- guaranteed delivery)

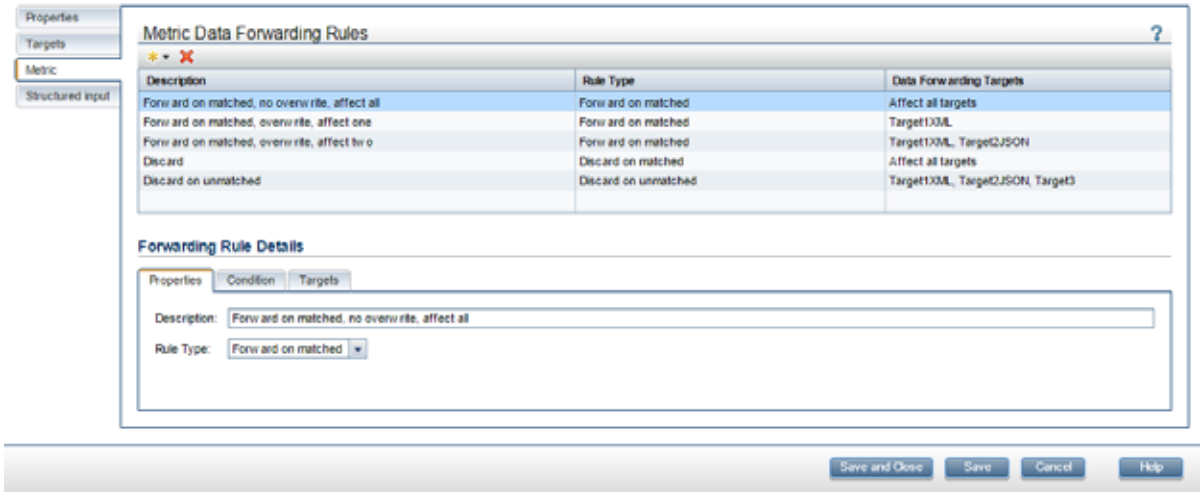
Usage Reference

The Data Forwarding Targets section is used in the syntax of the following policy types (click on link to navigate back):

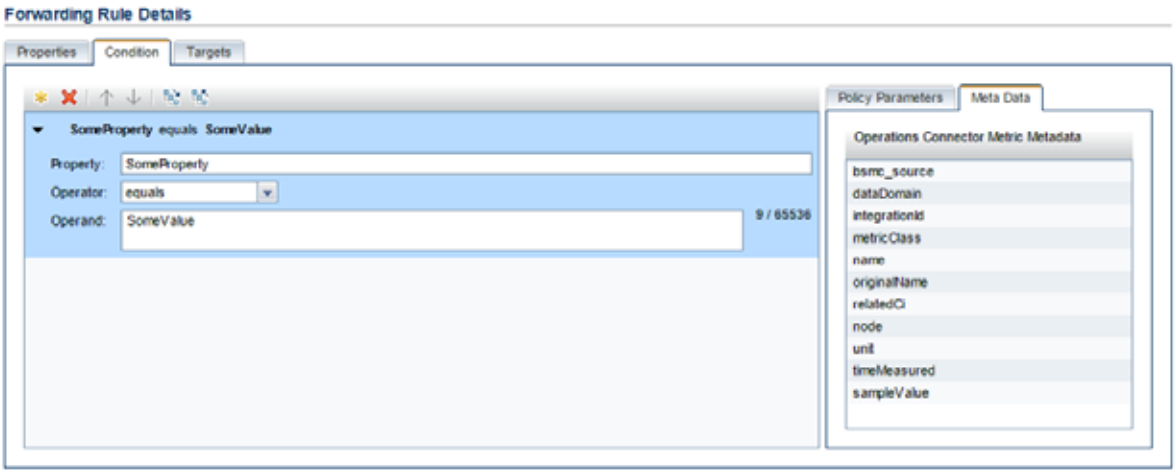
- Syntax of Data Forwarding policy template

39. Data Forwarding Rules section

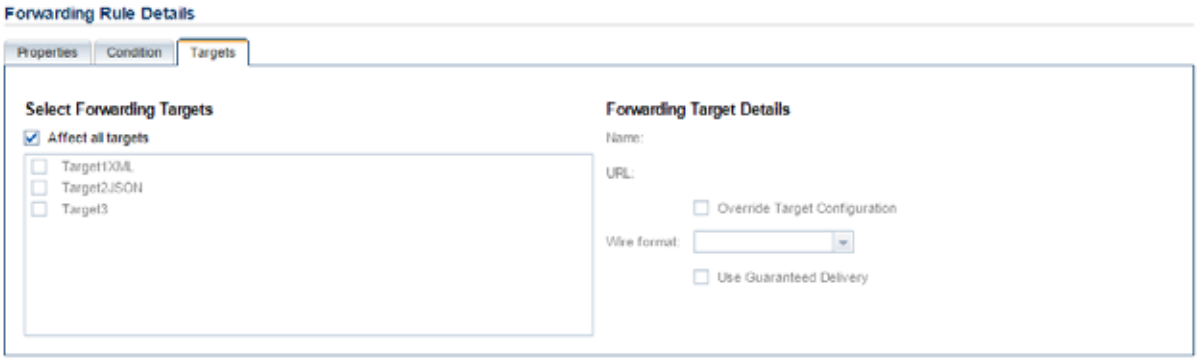
UI Reference - Tab Properties



UI Reference - Tab Condition



UI Reference - Tab Targets



For more information, see the Configure Data Forwarding Policies topic.

Example of corresponding policy syntax

```
GROUP "RULE"
   PARAM "RULETYPE"  "MATCH"
   PARAM "DATATYPE"  "METRIC"
   PARAM "id"  "3E36F397-99D2-53E3-6A38-20A628F9DEEB"
   PARAM "DESCRIPTION"  "Forward on matched, overwrite, affect two"
   GROUP "CONDITION"
      PARAM "OPER"  "SomeProperty"
      ATT "OP"  "<"
      ATT "OPER"  "someValue"
   GROUP_END
   GROUP "TARGETS"
      PARAM "AFFECT_ALL"  "0"
      PARAM "TARGET"  "Target1XML"
      ATT "ID"  "7215BE36-10F8-A2AC-9CD1-209FCCFE4A00"
      ATT "OVERRIDECONFIG"  ""
      ATT "SENDMODE"  ""
      ATT "WIREFORMAT"  ""
      PARAM "TARGET"  "Target2JSON"
      ATT "ID"  "F893917D-1B91-7B55-D969-20A0A51F83CF"
      ATT "OVERRIDECONFIG"  ""
      ATT "SENDMODE"  ""
      ATT "WIREFORMAT"  ""
   GROUP_END
GROUP_END
```

OR

```
GROUP "RULE"
   PARAM "RULETYPE"  "MATCH"
   PARAM "DATATYPE"  "RAW"
   PARAM "id"  "E1DF7AD2-2DB3-5B9E-7FFD-20A76E9193A9"
   PARAM "DESCRIPTION"  "Forward on matched"
   GROUP "CONDITION"
      PARAM "OPER"  "SomeProperty"
      ATT "OP"  "~="
      ATT "OPER"  "<*>"
   GROUP_END
   GROUP "TARGETS"
      PARAM "AFFECT_ALL"  "1"
   GROUP_END
 GROUP_END
```

The elements in the pages ‘Metric’ and ‘Structured input’ are the same.
To distinguish the context ‘Metric’ and ‘Structured input’, different values need to be used for the syntax element PARAM “DATATYPE”.

UI element	Syntax element	Possible values
Page: Metric	`PARAM "DATATYPE"`	`"METRIC"`
Page: Structured input	`PARAM "DATATYPE"`	`"RAW"`

Elements in the pages ‘Metric’ and ‘Structured input’:

UI element	Syntax element	Possible values
Description	`PARAM "DESCRIPTION"`	`"<user-defined>"` A brief description of the rule.

Rule Type:
Forward on matched
or
Discard on matched
or
Discard on unmatched
	`PARAM "RULETYPE" "MATCH"`
	or
	`PARAM "RULETYPE" "SUPP_MATCH"`
	or
	`PARAM "RULETYPE" "SUPP_UNMATCH"`	

Condition	`GROUP "CONDITION"`	
Property	`PARAM "OPER"`	<user-defined> must match input data property or can be one of the following Meta Data properties: Operations Connector Metric Metadata bsmc_source dataDomain integrationId metricClass name originalName relatedCI node unit timeMeasured sampleValue Also see section ‘UI Reference - Tab Condition (Tab Meta Data selected)’ above.
Operator	`ATT "OP"`	Please see Condition section
Operand	`ATT "OPER"`	<user-defined>
Targets	`GROUP "TARGETS"`	See **Data Forwarding** Targets section
Affect all targets	`PARAM "AFFECT_ALL"`	`"1"` (selected) or `"0"` (unselected)
If `PARAM "AFFECT_ALL" "0"` use the following elements to define values for specific targets:		
Selected target	`PARAM "TARGET"`	<user-defined> must match name of target defined in **Data Forwarding** Targets section
Override Target Configuration	`ATT "OVERRIDECONFIG"`	`""` (no overwrite) or `"TRUE"` (overwrite)
Wire format	`ATT "WIREFORMAT"`	`"XML"` or `"JSON"`
Use Guaranteed Delivery	`ATT "SENDMODE"`	`"SNF"` (for guaranteed delivery) or `"FAF"` (for non- guaranteed delivery)

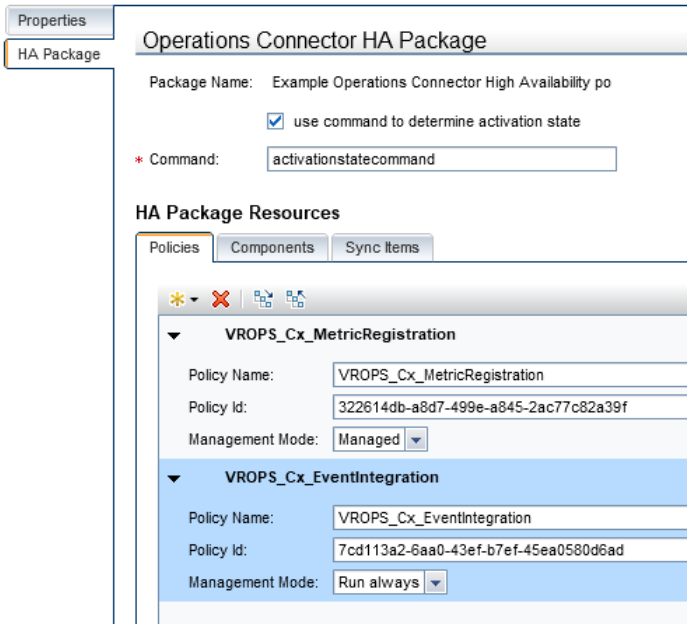
Usage Reference

The Data Forwarding Rules section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Data Forwarding policy template

40. Activation State and Policies section

UI Reference



For details, see Configure Operations Connector High Availability Policies topic.

Examples of corresponding policy syntax

```
PARAM "type"  "CMD"
PARAM "command"  "activationstatecommand"
GROUP "policies"
   GROUP "policy"
      PARAM "name"  "VROPS_Cx_MetricRegistration"
      PARAM "id"  "322614db-a8d7-499e-a845-2ac77c82a39f"
      PARAM "mode"  "MANAGED"
   GROUP_END
   GROUP "policy"
      PARAM "name"  "VROPS_Cx_EventIntegration"
      PARAM "id"  "7cd113a2-6aa0-43ef-b7ef-45ea0580d6ad"
      PARAM "mode"  "ALWAYS"
   GROUP_END
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Use command to determine activation state and Command	`PARAM "type"  "CMD"` `PARAM "command"`	`"<user-defined>"`
Policy name	`PARAM "name"`	`"<user-defined>"` ID and name must refer to same policy
Policy Id	`PARAM "id"`	`"<user-defined>"` ID and name must refer to same policy

Management Mode Managed Run Always	 PARAM "mode" "MANAGED" PARAM "mode" "ALWAYS"	-

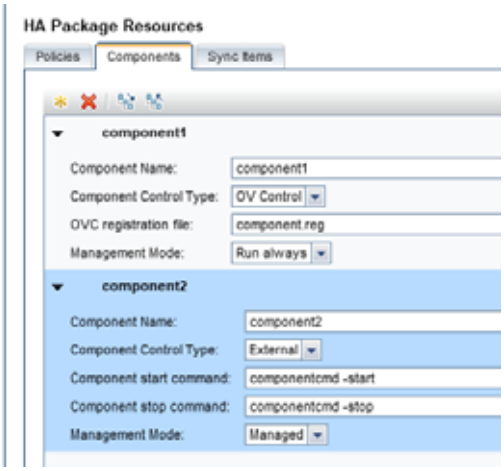
Usage Reference

The Policies section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Operations Connector High Availability policy template

41. Components section

UI Reference



For details, see Configure Operations Connector High Availability Policies topic.

Examples of corresponding policy syntax

```
GROUP "components"
   GROUP "component"
      PARAM "type"   "OVC"
      PARAM "name"   "component1"
      PARAM "mode"   "ALWAYS"
      PARAM "file"   "component.reg"
   GROUP_END
   GROUP "component"
      PARAM "type"   "EXTERNAL"
      PARAM "name"   "component2"
      PARAM "mode"   "MANAGED"
      PARAM "startCmd"   "componentcmd -start"
      PARAM "stopCmd"   "componentcmd -stop"
   GROUP_END
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values
Component Name	`PARAM "name"`	`"<user-defined>"`

Component Control Type
Managed
Run Always
	
	`PARAM "mode"  "MANAGED"`
	`PARAM "mode"  "ALWAYS"`
		-

OVC Registration File	`PARAM "file"`	`"<user-defined>"`
Component start command	`PARAM "startCmd"`	`"<user-defined>"`
Component stop command	`PARAM "stopCmd"`	`"<user-defined>"`

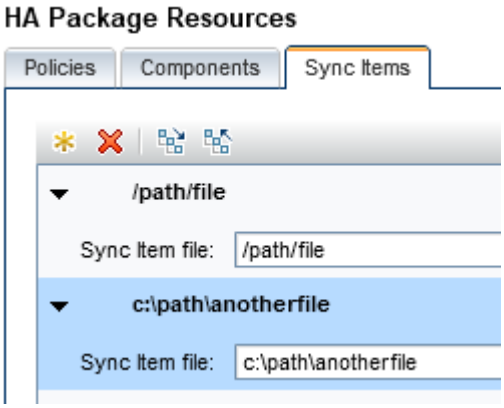
Usage Reference

The Components section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Operations Connector High Availability policy template

42. Sync Items section

UI Reference



For details, see Configure Operations Connector High Availability Policies topic.

Examples of corresponding policy syntax

```
GROUP "syncItems"
   GROUP "syncItem"
      PARAM "file"  "/path/file"
   GROUP_END
   GROUP "syncItem"
      PARAM "file"  "c:\\path\\anotherfile"
   GROUP_END
GROUP_END
```

Mapping of UI elements to Syntax elements

UI element	Syntax element	Possible values

Sync Item file	`PARAM "file"`	`"<user-defined>"`
		For Windows specify path to file using \\ as delimiter, e.g.
		Windows: `C:\path\file`
		`"c:\\path\\file"`
		Unix: `/path/file`

Usage Reference

The Policies section is used in the syntax of the following policy types (click on link to navigate back):

- Syntax of Operations Connector High Availability policy template

43. Micro Focus Trademark Information

MICRO FOCUS and the Micro Focus logo, among others, are trademarks or registered trademarks of Micro Focus (IP) Limited or its subsidiaries in the United Kingdom, United States and other countries. All other marks are the property of their respective owners.

44. Company Details

Company name: Micro Focus International plc
Place of registration: England and Wales
Registered number: 5134647
Registered address: The Lawn, 22-30 Old Bath Road, Berkshire, RG14 1Q