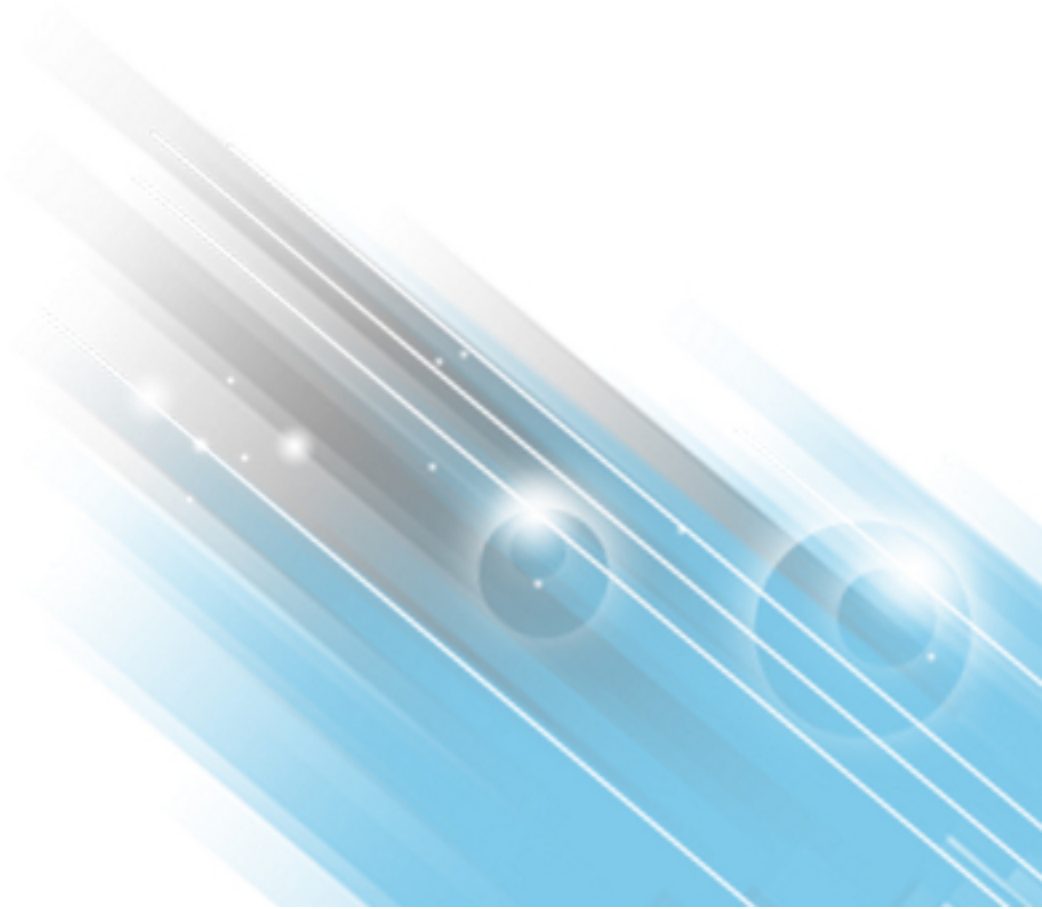
HP Business Service Management

Software Version: 10.01

BSM Connector User Guide



Document Release Date: 13 October 2015
Software Release Date: October 2015

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 2012-2015 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

AMD and the AMD Arrow symbol are trademarks of Advanced Micro Devices, Inc.

Citrix® and XenDesktop® are registered trademarks of Citrix Systems, Inc. and/or one more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries.

Google™ and Google Maps™ are trademarks of Google Inc.

Intel®, Itanium®, Pentium®, Intel® Xeon®, and Lync® are trademarks of Intel Corporation in the U.S. and other countries.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft®, Windows®, Windows NT®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

Red Hat® is a registered trademark of Red Hat, Inc. in the United States and other countries.

UNIX® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hp.com/group/softwaresupport/search-result?keyword=.

This site requires an HP Passport account. If you do not have one, click the **Create an account** button on the HP Passport Sign in page.

Support

Visit the HP Software Support web site at: https://softwaresupport.hp.com

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software Support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to https://softwaresupport.hp.com and click **Register**.

To find more information about access levels, go to:
https://softwaresupport.hp.com/web/softwaresupport/access-levels

HP Software Solutions & Integrations and Best Practices

Visit HP Software Solutions Now at https://h20230.www2.hp.com/sc/solutions/index.jsp to explore how the products in the HP Software catalog work together, exchange information, and solve business needs.

Visit the Cross Portfolio Best Practices Library at https://hpln.hp.com/group/best-practices-hpsw to access a wide variety of best practice documents and materials.

Contents

Part I: Setting Up and Configuring BSM Connector 11
Chapter 1: BSM Connector Overview 12
Key Features and Benefits 14
BSM Connector Deployment Scenarios 16
User Interface 16
User Authentication 17
Licensing 18
Chapter 2: Setup and Configuration 19
Getting Started 19
How to Log on to BSM Connector 21
How to Manage BSM Connector Processes 23
How to Configure BSM Connector to Communicate with OMi 23
How to Configure BSM Connector to Communicate with a Distributed OMi 25
How to Manage BSM Connector with HPOM 26
Configuring LW-SSO Authentication 29
Backing Up and Restoring BSM Connector Configuration and Policies 32
BSM Connector Policies 32
BSM Connector Backsync Script 32
BSM Connector Specific Settings in XPL 33
BSM Connector Web Application LWSSO Configuration 33
BSM Connector Web Application Logging Configuration 33
BSM Connector Web Application Ports Configuration 34
BSM Connector Web Application User Configuration 34
The BSM Connector REST Web Service Receiver Configuration 35
BSM Connector Flexible Management Policy Template 35
Troubleshooting and Limitations 36
Chapter 3: Security 37
Hardening the BSM Connector Platform 37
Permissions and Credentials 38
Setting Up Smart Card Authentication 38
Chapter 4: Event Synchronization 41
How to Configure Policies for Event Synchronization 42
How to Write Perl Scripts for Event Synchronization 43
Chapter 5: UI Drilldown 45
How to Configure Drilldown into Third-Party Systems 46

Part II: Working with BSM Connector 49
Chapter 6: Policy Management 50
How to Edit Policies 52
How to Copy Policies 54

How to Delete Policies 54
How to Activate and Deactivate Policies 54
How to Export Policies 55
How to Import Policies 57
How to Configure Policy Management Options 58
Chapter 7: Collecting and Viewing Metrics Data 60
How to Collect Metrics 60
Example – Create a Metrics Policy 61
Collecting Metrics with HP Operations Agent 70
Verifying the Status of oacore Process 70
Starting or Stopping the oacore Process 71
Verifying Data Logging 71
Configuring Data Purging 72
Chapter 8: Collecting Topology Data 74
Topology Policies and Topology Synchronization 74
Send Topology Data Directly to RTSM 75
Use Topology Synchronization Rules to Map Data into RTSM CIs and Relationships 77
Topology Synchronization Rules 78
Mapping Syntax 82
Common Mapping File Format 83
Mapping File Syntax 84
Rules 84
Rule Conditions 84
Operator Elements 86
Operand Elements 89
Mapping Elements 96
Filtering 97
Type Mapping 98
Attribute Mapping 99
Relation Mapping 101
XPath Navigation 103
Data Structure 104
Example of an XPath-Navigated Data Structure 106
XPath Expressions and Example Values 106
Uploading the Sync Package to the OMi server 107
Topology Discovery Syntax 107
Troubleshooting 112
Chapter 9: Collecting Generic Output Data 116
How to Collect Generic Output Data 116
Chapter 10: Forwarding Data 118
How to Forward Data 118
Data Forwarding User Interface 119
Configuring Data Forwarding Properties 119
Configuring Data Forwarding Targets 120
Configuring Metric Data Forwarding 121
Configuring Structured Input Data Forwarding 122

Part III: Integrating Data With BSM Connector 124
Chapter 11: Database Policies (Events, Metrics, and Generic Output) 125
How to Collect Event Data from Databases 125
How to Collect Metrics Data from Databases 126
How to Collect Generic Output Data from Databases 127
Database Policy User Interface 127
Configuring Database Policy Properties 128
Configuring the Data Source in Database Policies 129
Configuring Mappings in Database Policies (Events and Metrics Only) 133
Configuring Mappings in Database Policies (Generic Output Only) 135
Configuring Event Defaults in Database Policies 137
Configuring Metrics Defaults in Database Policies 139
Configuring Event Rules in Database Policies 142
Configuring Metrics Rules in Database Policies 145
Configuring Options in Database Policies 149
Troubleshooting Database Policies 150
Chapter 12: Open Message Interface Policies (Events only) 151
How to Collect Event Data from the Open Message Interface 151
opcmsg Command-Line Tool 152
opcmsg Java API 154
opcmsg C API 155
Open Message Interface Policy User Interface 156
Configuring Open Message Interface Policy Properties 156
Configuring Event Defaults in Open Message Interface Policies 156
Configuring Rules in Open Message Interface Policies 158
Configuring Options in Open Message Interface Policies 160
Chapter 13: Perl Script Policies 162
How to Collect Event Data by Using Perl Scripts 162
How to Collect Metrics Data by Using Perl Scripts 163
How to Collect Generic Output Data by Using Perl Scripts 163
Perl Policy User Interface 164
Configuring Perl Policy Properties 165
Configuring the Data Source in Perl Policies 165
Configuring Mappings in Perl Policies (Events and Metrics Only) 167
Configuring Mappings in Perl Policies (Generic Output Only) 169
Configuring Event Defaults in Perl Policies 171
Configuring Metric Defaults in Perl Policies 173
Configuring Event Rules in Perl Policies 176
Configuring Metric Rules in Perl Policies 179
Configuring Options in Perl Policies 183
Chapter 14: REST Web Service Listener Policies 185
How to Collect Event Data Through the REST Web Service Listener 185
How to Collect Metrics Data Through the REST Web Service Listener 186
Configuring Topology Through the REST Web Service Listener 187
How to Collect Generic Output Data Through the REST Web Service Listener 187
REST Web Service Listener Policy User Interface 188

Configuring REST Web Service Listener Policy Properties 189
Configuring the Data Source in REST Web Service Listener Policies 189
Configuring Mappings in REST Web Service Listener Policies 193
Configuring Mappings in REST Web Service Listener Policies (Generic Output Only) 195
Configuring Event Defaults in REST Web Service Listener Policies 197
Configuring Metrics Defaults in REST Web Service Listener Policies 199
Configuring Event Rules in REST Web Service Listener Policies 202
Configuring Metrics Rules in REST Web Service Listener Policies 205
Configuring Options in REST Web Service Listener Policies 209
Troubleshooting REST Web Service Listener Policies 210
Chapter 15: Scheduled Task Policies (Events only) 211
How to Schedule Tasks 211
Scheduled Task Policy User Interface 212
Configuring Scheduled Task Policy Properties 212
Configuring Tasks in Scheduled Task Policies 212
Configuring Schedules in Scheduled Task Policies 214
Configuring Events in Scheduled Task Policies 215
Policy Objects for Scripts 216
Chapter 16: SNMP Interceptor Policies (Events only) 227
How to Collect Event Data from SNMP Traps 228
SNMP Trap Policy User Interface 229
Configuring SNMP Trap Policy Properties 229
Configuring Event Defaults in SNMP Trap Policies 229
Configuring Rules in SNMP Trap Policies 231
Configuring Options in SNMP Trap Policies 233
Chapter 17: Structured Log File Policies (Events, Metrics, and Generic Output) 234
How to Collect Event Data from Structured Log Files 234
How to Collect Metrics Data from Structured Log Files 235
How to Collect Generic Output Data from Structured Log Files 236
Structured Log File Policy User Interface 236
Configuring Structured Log File Policy Properties 237
Configuring Data Source in Structured Log File Policies 238
Configuring Mappings in Structured Log File Policies (Event and Metrics Only) 242
Configuring Mappings in Structured Log File Policies (Generic Output Only) 244
Configuring Event Defaults in Structured Log File Policies 246
Configuring Metrics Defaults in Structured Log File Policies 248
Configuring Event Rules in Structured Log File Policies 251
Configuring Metrics Rules in Structured Log File Policies 254
Configuring Options in Structured Log File Policies 258
Troubleshooting Structured Log File Policies 259
Chapter 18: XML File Policies 260
How to Collect Event Data from XML Files 260
How to Collect Metric Data from XML Files 261
How to Collect Topology Data from XML Files 261
How to Collect Generic Output Data from XML Files 262
XML File Policy User Interface 263

Configuring XML File Policy Properties ... 263
Configuring the Data Source in XML File Policies ... 264
Configuring Mappings in XML File Policies ... 266
Configuring Mappings in XML File Policies (Generic Output Only) ... 269
Configuring Event Defaults in XML File Policies ... 271
Configuring Metric Defaults in XML File Policies ... 272
Configuring Event Rules in XML File Policies ... 274
Configuring Metrics Rules in XML File Policies ... 277
Configuring Options in XML File Policies ... 281

Part IV: BSM Connector Reference ... 283
Appendix A: Pattern Matching in Policy Rules ... 284
Pattern-Matching Details ... 284
User-Defined Variables in Patterns ... 288
Pattern Matching for Variables ... 290
Examples of Pattern Matching in Rule Conditions ... 291
Appendix B: Command-Line Tools ... 293
BSM Connector Configuration Tool ... 293
Local User Configuration Tool ... 297
Appendix C: UI Descriptions ... 299
Defaults and Rules Pages (Events) ... 299
Defaults and Rules Pages (Metrics) ... 316
Indicators Tab ... 320
Mappings Page (Events and Metrics) ... 321
Mappings Page (Generic Output) ... 323
Mappings Tab ... 325
Metric Page (Data Forwarding) ... 325
Operators Tab (Metrics Only) ... 327
Options Page (Events only) ... 328
Options Page (Metrics only) ... 330
Pattern Matching Variables Tab (Events and Metrics Only) ... 332
Policy Variables Tab ... 333
Policy Variables Tab for Database and REST Web Service Listener Policies (Events only) ... 333
Policy Variables Tab for XML File and Structured Log File Policies (Events only) ... 334
Policy Variables Tab for Open Message Interface, Scheduled Task, and SNMP Interceptor Policies (Events only) ... 335
Policy Variables Tab for All Policy Types (Metrics only) ... 338
Properties Page ... 338
Rules Page - Policy Rules ... 339
Sample Data Tab ... 341
Schedule Page ... 343
Source Page ... 345
Start, Success, Failure Event Pages (Scheduled Task Policies) ... 360
Structured Input Page (Data Forwarding) ... 360
Target Page (Data Forwarding) ... 362

Task Page (Scheduled Task Policies) 363

Send Documentation Feedback 365

Part I: Setting Up and Configuring BSM Connector

This section includes:

- "BSM Connector Overview" on page 12
- "Setup and Configuration" on page 19
- "Security" on page 37
- "Event Synchronization" on page 41
- "UI Drilldown" on page 45

Chapter 1: BSM Connector Overview

HP BSM Connector (BSM Connector) is a component of Operations Manager i (OMi) that enables you to integrate data from third-party systems (typically enterprise management systems) in OMi. You can integrate events, metrics, topology, and generic output data in OMi. Third-party systems are those not provided by HP. BSM Connector also works with some HP applications.

BSM Connector uses policies to access the data sources. If the data matches the conditions defined in the policies, the data is forwarded in the form of events or metrics to OMi. Policies can also report topology data to OMi in order to create CIs and CI relationships in OMi's Run-time Service Model (RTSM).

Integration Types

Depending on the third-party data source and what type of data you need to collect, you choose one of the following integration types:

- **Remote integration**. In remote integrations, installation of HP software on the third-party system is not required. BSM Connector can be installed on any system in your environment, provided the system meets the installation prerequisites.
- **Local integration**. You install BSM Connector directly on the third-party system.

Integration Data

BSM Connector can access the following data sources:

Data Source	Integration Type	Integration Data	Description
Structured log files	Local only	Events Metrics Generic output	Structured log file policies watch for specific entries added to a log file that is provided by a third-party system. These entries are then matched against the log file structure fields defined in policies by using the OM pattern-matching language . For details, see "Structured Log File Policies (Events, Metrics, and Generic Output)" on page 234.

Data Source	Integration Type	Integration Data	Description
Database tables	Local or remote	Events Metrics Generic output	Database policies enable you to collect event, metric, and generic output data from a database used by third-party systems by performing a query through a JDBC connection. For details, see "Database Policies (Events, Metrics, and Generic Output)" on page 125.
Perl scripts	Local	Events Metrics Generic output	Perl script policies enable you to collect event, metric, and generic output data using Perl scripts. For details, see "Perl Script Policies" on page 162.
Web service requests	Local or remote	Events Metrics Topology Generic output	REST Web service listener policies process the data received by the BSM Connector REST Web service listener. For details, see "REST Web Service Listener Policies" on page 185.
SNMP traps	Local or remote	Events	SNMP trap policies process SNMP traps received by BSM Connector from third-party systems. For details, see "SNMP Interceptor Policies (Events only)" on page 227.
XML files	Local only	Events Metrics Topology Generic output	XML file policies watch for specific entries added to an XML file provided by a third-party system by trying to match against a regular expression. The log file must be a local file. For details, see "XML File Policies" on page 260.
Open message interface messages	Local only	Events	Open message interface policies process messages sent by the BSM Connector opcmsg command-line tool. For details, see "Open Message Interface Policies (Events only)" on page 151.

Data Source	Integration Type	Integration Data	Description
Scheduled tasks	Local only	Events	Scheduled task policies enable you to schedule commands to run on the BSM Connector system. Scheduled task policies send an event to OMi to indicate the start, success, or failure of the command. For details, see "Scheduled Task Policies (Events only)" on page 211.

BSM Connector Data Consumers

The following applications consume the BSM Connector data:

- Operations Manager i (Events, Topology)
- Performance Graphing (Metrics)

  Note: If you have a CI monitoring five minutes of data and there are multiple data points within the duration, the graph shows only the last data received within the duration.

- Run-time Service Model (Topology)

Available Out of the Box Integrations

You can use one of the out-of-the box integrations that are available for BSM Connector. Alternatively, if you do not find the integration that you are looking for, you can develop your own custom integration.

HP is continually updating the list of the integrations with third-party products. For details and for download information, see the HP Live Network site https://hpln.hp.com/group/bsm-integrations.

Key Features and Benefits

Key Features

- **Topology.** Discover and report topology from the product BSM Connector connects to (based on Web services or XML files). The discovered topology populates the RTSM. You can then manage and work with these discovered configuration items (CIs) and relationships in views.
- **Events.** Get events (from structured log files, database tables, Web services, XML files, SNMP traps, Open Message Interface messages, or scheduled tasks).
- **Metrics.** Get metrics (from structured log files, database tables, Web services, or XML files); metrics are stored on the BSM Connector system.
- **Generic output**. Get generic output data (from structured log files, database tables, Web services, XML files, or gathered by Perl scripts).
- **Web-based UI.** Define policies using Web-based UI.
- **Event synchronization.** After OMi receives an event, you can keep it up to date on the event

source by configuring OMi and BSM Connector to synchronize event changes back to the third-party system that generated the original event.

- **OMi integration.** Manage BSM Connector from OMi.
- **Backward compatibility.** BSM Connector 9.2x configuration and most of the policies can be upgraded to BSM Connector 10.01. For more information, see the upgrade section in the BSM Connector *Installation and Upgrade Guide*.
- **Out-of-the-box connectors.** Supporting OMi's open integration strategy, providing out-of-the-box connectors to HP and third-party products. This enables expanding connectivity of OMi and related products.

  Customers can access connectors via HP Live Network (HPLN); HP is continually updating the list out-of-the box integrations with third-party products. For details and for download information, see the HP Live Network site https://hpln.hp.com/group/bsm-integrations.

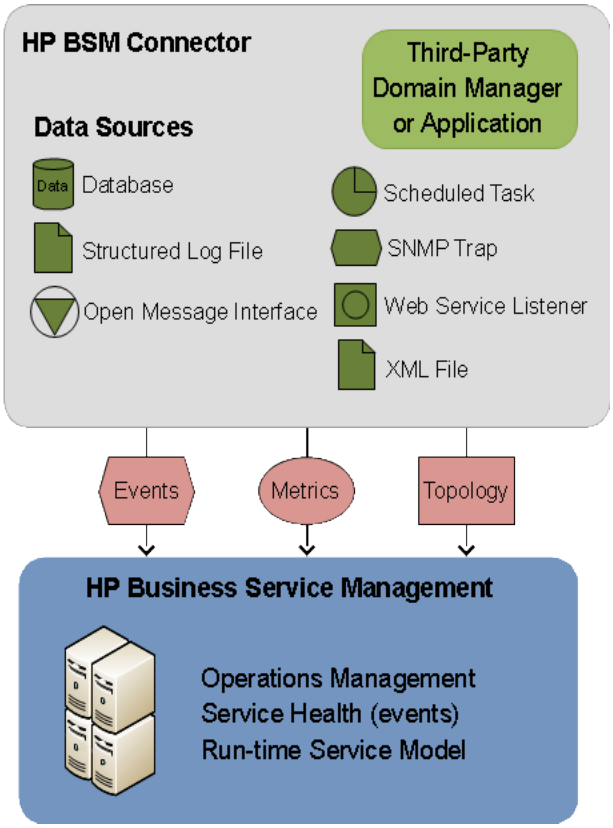
  HP certified partners will also be able to build their own connectors and post them on HPLN. See the HP Live Network site https://hpln.hp.com/group/bsm-integrations.
- **Data forwarding.** Forward collected data to consumer applications.

Benefits

- **Run-time Service Model (RTSM).** BSM Connector can discover and report topology from the product BSM Connector connects to. You can then manage and work with these discovered configuration items (CIs) in views.
- **Performance Perspective.** You can view the BSM Connector metrics in the OMi Performance Perspective.
- **360 Degrees View.** You can view topology and events related data in the OMi360 Degrees View.
- **Topology-based Event Correlation** You can use topology-based event correlation to help you better understand, monitor, and manage the problems that can have an effect on the objects in your IT environment. The Correlation Rules manager enables you to define and deploy rules that use indicators to correlate the events occurring in the different domains throughout the managed IT environment.

BSM Connector Deployment Scenarios

The following figure shows a BSM Connector deployment.



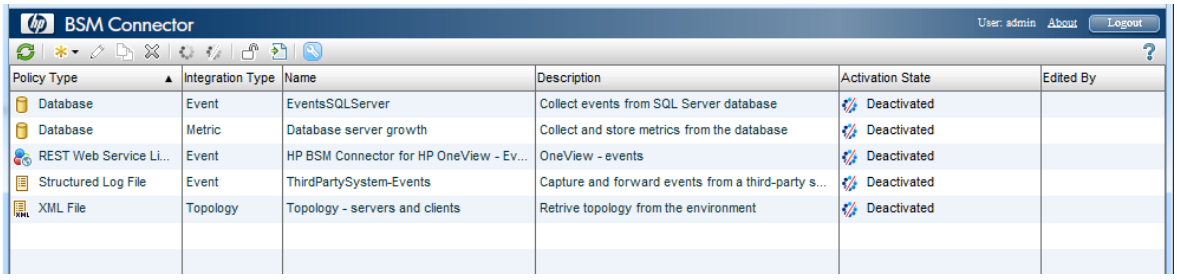
You typically install BSM Connector on the third-party system that acts as data source. For example, to integrate NNMi data into OMi, you install BSM Connector on the NNMi management server. However, BSM Connector can also access data sources on remote systems without the need to install data collectors on these systems.

Where you install BSM Connector depends on the type of data you want to integrate in OMi. BSM Connector must be installed and run locally on the data host when integrating data originating from structured log files, XML files, the open message interface, or scheduled tasks. You can install BSM Connector on any system that meets the requirements if you want to integrate data originating from databases, or receive data through SNMP traps or the REST Web service listener. For details, see "Integration Types" on page 12 and "Integration Data" on page 12.

User Interface

The BSM Connector user interface is Web based; you can therefore access it from anywhere using a supported Web browser.

When you connect to BSM Connector, the policy management list opens as shown below.



For more information about the toolbar buttons and columns, as well as policy management in general, see "Policy Management" on page 50.

User Authentication

BSM Connector supports the following user authentication strategies:

- Single Sign-On (SSO) for OMi (recommended)

  The default single sign-on authentication strategy for OMi is Lightweight Single Sign-On (LW-SSO). LW-SSO is embedded in OMi and does not require an external computer for authentication.

  LW-SSO enables a user to log into OMi once and gain access to all applications without being prompted to log in again. The applications inside OMi trust the authentication, and you do not need further authentication when moving from one application to another. For example, if you configure BSM Connector to use LW-SSO, OMi users can launch the BSM Connector user interface without having to provide additional credentials.

  > **Tip:** OMi enables you to group users to make managing user permissions more efficient. For example, you could add all BSM Connector users to a dedicated group (`BSMC_ADMINS`, for example) and assign permissions to access BSM Connector to the group.

  For more information about LW-SSO, see the *OMi Administration* guide.

- Smart card authentication

  BSM Connector supports user authentication using smart cards. BSM Connector can be configured to use the certificates stored on the smart card in place of the standard model of each user manually entering a user name and password.

- Local user authentication

  Each BSM Connector instance maintains a local user store. These users can access the local BSM Connector instance only and cannot gain access to other applications. When OMi users launch the BSM Connector user interface, they have to provide the credentials of a local BSM Connector user.

For more information about accessing BSM Connector, see "How to Log on to BSM Connector" on page 21.

Licensing

BSM Connector does not require a separate license from the Operations Manager i license. For OMi licensing information, please refer to the license management chapter in the *the OMi Administration Guide*.

Chapter 2: Setup and Configuration

This chapter contains information about setting up and configuring BSM Connector.

Getting Started

1. Plan your integration strategy.

   Consider the type of information you want to view in OMi from your third-party system: events, metrics, or topology.

   Consider the data source: structured log files, databases, Web services data, SNMP traps, XML files, open message interface messages, or scheduled tasks.

2. Install BSM Connector.

   You install BSM Connector locally on the third-party system that provides the integration data. Alternatively, install BSM Connector on any system that meets the installation requirements if you want to access remote data sources such as databases or receive SNMP traps or Web service data. For details on installing BSM Connector, see the interactive HP BSM Connector *Installation and Upgrade Guide*.

   Alternatively, upgrade from a previous version of BSM Connector as described in the BSM Connector *Installation and Upgrade Guide*.

3. Run the `bsmc-conf.[bat|sh]` configuration script that is located in `%ovdatadir%\installation\HPOprBSMC` on Windows, and `/var/opt/OV/installation/HPOprBSMC` on Linux.

4. Grant the BSMC certificate request on OMi side.

5. Configure the BSM Connector instance as a BSM Connector connected server in OMi:

   Administration > Setup and Maintenance > Connected Servers

   For details on configuring a BSM Connector integration server in OMi, see "How to Configure BSM Connector to Communicate with OMi" on page 23.

6. Log on to BSM Connector.

   To access BSM Connector from a Web browser or from within OMi, see "How to Log on to BSM Connector" on page 21.

7. Create or import policies that collect the integration data from the third-party system.

   For details on obtaining out-of-the box integrations, see "Available Out of the Box Integrations" on page 14.

   For more information about importing policies developed on other servers (for example, HP Operations Manager, HP Network Node Manager *i*, or other BSM Connector servers), see "How to Import Policies" on page 57

You can also develop your own policies to collect data from different sources. Table: Policy Development Tasks provides links to the tasks you need to complete for the different data and policy types.

Table: Policy Development Tasks

Data Source	To Integrate Events	To Integrate Metrics	To Integrate Topology	To Collect Generic Output Data
Structured log files	"How to Collect Event Data from Structured Log Files" on page 234	"How to Collect Metrics Data from Structured Log Files" on page 235	Not available.	"How to Collect Generic Output Data from Structured Log Files" on page 236
Databases	"How to Collect Event Data from Databases" on page 125	"How to Collect Metrics Data from Databases" on page 126	Not available.	"How to Collect Generic Output Data from Databases" on page 127
Perl scripts	"How to Collect Event Data by Using Perl Scripts" on page 162	"How to Collect Metrics Data by Using Perl Scripts" on page 163	Not available.	"How to Collect Generic Output Data by Using Perl Scripts" on page 163
Web service requests	"How to Collect Event Data Through the REST Web Service Listener" on page 185	"How to Collect Metrics Data Through the REST Web Service Listener" on page 186	"Configuring Topology Through the REST Web Service Listener" on page 187	"How to Collect Generic Output Data Through the REST Web Service Listener" on page 187
SNMP traps	"How to Collect Event Data from SNMP Traps" on page 228	Not available.	Not available.	Not available.
XML files	"How to Collect Event Data from XML Files" on page 260	"How to Collect Metric Data from XML Files" on page 261	"How to Collect Topology Data from XML Files" on page 261	"How to Collect Generic Output Data from XML Files" on page 262
Open message interface messages	"How to Collect Event Data from the Open Message Interface" on page 151	Not available.	Not available.	Not available.

Data Source	To Integrate Events	To Integrate Metrics	To Integrate Topology	To Collect Generic Output Data
Scheduled tasks	"How to Schedule Tasks" on page 211	Not available.	Not available.	Not available.

8. Activate the policies.

   When you create a new policy or import a policy, the policy exists in the BSM Connector policy repository but does not function yet. You must first activate the policy for it to start accessing the corresponding data source.

   For details on activating policies, see "How to Activate and Deactivate Policies" on page 54.

9. View the data in OMi.

   In OMi, open the applications that consume the integrated data and make sure OMi receives the expected data. For details on the applications that consume BSM Connector integration data, see "BSM Connector Data Consumers" on page 14.

   For example, to view events in OMi, access OMi and open the Event Browser.

How to Log on to BSM Connector

You access BSM Connector using a supported Web browser, from any computer with a network connection to a BSM Connector server.

The BSM Connector user interface opens without prompting for user authentication if single sign-on (SSO) is configured and the user is already logged on to OMi. In this case an LW-SSO session cookie stored in the Web browser contains the OMi authentication information. OMi users can access BSM Connector only if they have the necessary permissions (for example, if they are members of the `BSMC_ADMINS` group).

If BSM Connector is configured for smart card authentication, you are prompted to insert your smart card and enter the PIN.

This section includes:

- "Log On to BSM Connector" below
- "Log Out of BSM Connector" on page 23
- "Troubleshooting" on page 23

Log On to BSM Connector

You can start the BSM Connector user interface in the following ways:

- Start the BSM Connector user interface from a Web browser
  a. To start the BSM Connector user interface, open a Web browser at the following URL:

     `https://<BSM Connector system>:30000/bsmconnector/`

     <BSM Connector system> is localhost or the hostname of the BSM Connector server. If OVTomcat is configured to use a different port, use this port instead.

b. Depending on the security settings in your environment, you may have to log on or provide your smart card PIN. If asked for log-on credentials, type the user name and password of a local user account. The user name and password of the default local user account is:

   User name: `admin`
   Password: `admin`

   Note: BSM Connector locks the user account for 60 seconds after five failed logon attempts. Each subsequent failed logon attempt restarts the lockout period.

- Start the BSM Connector user interface in OMi

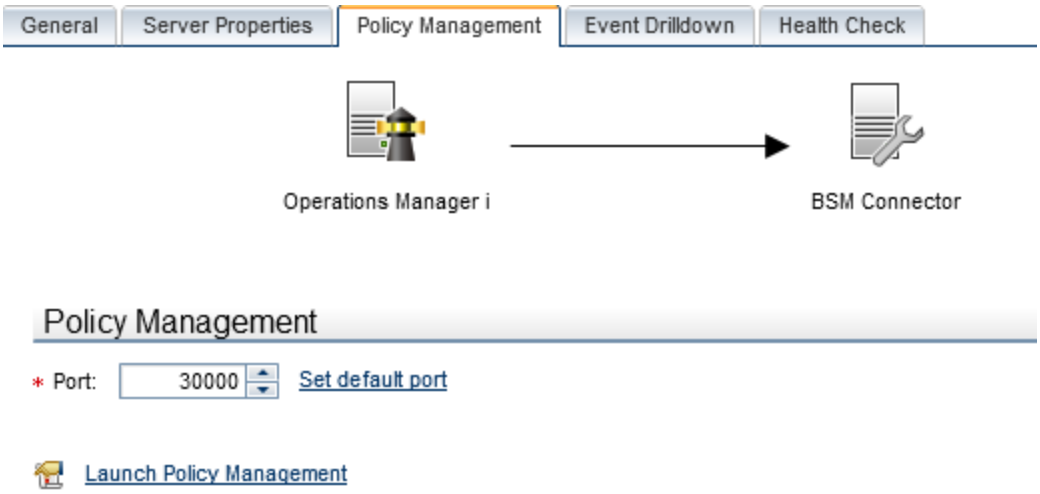
  OMi enables you to start the user interface of BSM Connector or of the third-party system directly from within OMi. The following integrations are available:

  ### BSM Connector UI integrations

  - **Connected Servers manager.** Navigate to **Administration > Setup and Maintenance > Connected Servers**. In the Connected Servers pane, locate the BSM Connector you want to launch and open it for editing.

    In the Edit Server Connection window, navigate to the Policy Management tab, and click the **Launch Policy Management** link.

    The following illustration shows the **Launch Policy Management** link in the Connected Servers manager.

  - **Event context.** In the OMi Event Browser, right-click an event that has been forwarded from a BSM Connector server, and then select **Configure > Integration Policies**.

  Depending on the security settings in your environment, you may have to log on or provide a smart card PIN. If you are prompted for login information, type the user name and password of a local user account. BSM Connector does not require you to log on if single sign-on is configured.

Third-party system UI integration

In the OMi Event Browser, right-click an event that has been forwarded from a BSM Connector server, and then select **Show > Event in Source Manager**.

UI integrations with third-party systems are not available out of the box. For information how to configure a UI integration, see "How to Configure Drilldown into Third-Party Systems" on page 46.

Log Out of BSM Connector

To log out of BSM Connector, click **Logout** at the top of the page.

If a user ends a BSM Connector user interface session by closing the Web browser window or tab (for example by clicking the Close button in the window's title bar or using a menu option), BSM Connector automatically logs the user off after 60 seconds.

Troubleshooting

- **Problem:** The login page appears although a valid smart card exists.

  Solution: The user is not set up as a local user in BSM Connector. Look for the value of the User Principal Name (UPN) in the Subject Alternative Name (SAN) field in the certificate and add a user with that name to BSM Connector. For details, see "Setting Up Smart Card Authentication" on page 38.

How to Manage BSM Connector Processes

BSM Connector has multiple processes that are required to be running to allow users to access BSM Connector. The processes run continuously as background processes without affecting other applications.

HP Operations Agent processes

The HP Operations Agent processes start automatically. However you can manually start and stop HP Operations Agent using the ovc command:

Windows: `%ovinstalldir%\bin\<windows_platform>\ovc`

Linux: `/opt/OV/bin/ovc`

For more information about ovc, type `ovc -help`.

How to Configure BSM Connector to Communicate with OMi

To enable the connection between BSM Connector and OMi, the BSM Connector certificate must be approved on the OMi side, and the BSM Connector instance must be configured as a BSM Connector connected server in OMi.

When you add a BSM Connector integration to OMi, the BSM Connector instance must be accessible to OMi.

1. *Prerequisite.* Install and configure BSM Connector.

   For details, see the interactive BSM Connector Installation and Upgrade Guide.

   Note: On the BSM Connector server, close any running instances of the command `ovconfchg -edit`.

2. *Prerequisite.* In a distributed OMi environment with a load balancer or reverse proxy, configure HP Operations Agent manually.

   In a distributed OMi environment with a reverse proxy or a load balancer, the OMi Data Processing server and the BSM Connector server may not be able to communicate with each other in order to request and install a certificate. To provide the agent with a certificate, you must issue the certificate manually on the OMi certificate server and then import it manually on the BSM Connector system. In addition, you must manually set the MANAGER_ID configuration variable. The MANAGER_ID defines who is allowed to access the agent from outside. For details, see "How to Configure BSM Connector to Communicate with a Distributed OMi" on the next page.

3. *Prerequisite.* Make sure the BSM Connector certificate request is granted on the OMi side.

4. In OMi, open the Connected Servers manager from Administration, select:

   Administration > Setup and Maintenance > Connected Servers

   In the Connected Servers pane, click **New** and select **BSM Connector**. The Create New Server Connection - BSM Connector dialog box opens.

   The information you need to enter depends on the type of integration data you want to collect with this BSM Connector:

   - **General**

     Complete the mandatory **Display Name** and **Name** fields. Additionally complete also the **Description** field. To activate BSM Connector server after creation, enable the appropriate box provided at the bottom of the page.

   - **Server Properties**

     Enter the Fully Qualified DNS name of the server. Optionally, set the Advanced Delivery Options.

   - **Policy Management**

     Set the communication port. To restore the port to the default value (30000), click Set default port. You can launch the BSM Connector by clicking the Launch Policy Management icon, in case this is configured on the BSM Connector.

   - **Event Drilldown**

     Event drilldown enables OMi users to launch the user interface of the third-party system in the context of an event collected through BSM Connector.

5. Click **Finish** to save the new BSM Connector integration.

   After the wizard is successfully completed, a new connected server is created.

6. *Results.* You can view the collected data in OMi, depending on the type of integration data that

you collect. For example, you can view data in RTSM or Service Health.

Note: You must first configure and activate policies in BSM Connector to start the data collection. To stop data collection, deactivate the policies.

For full details about how to configure a BSM Connector connected server, see the OMi online help or the OMi Administration Guide.

How to Configure BSM Connector to Communicate with a Distributed OMi

In a distributed OMi environment with a reverse proxy or a load balancer, the OMi data processing server and the BSM Connector server may not be able to communicate with each other in order to request and install a certificate. To provide the agent with a certificate, you must issue the certificate manually on the OMi certificate server and then import it manually on the BSM Connector system. In addition, you must manually set the MANAGER_ID configuration variable. The MANAGER_ID defines who is allowed to access the agent from outside.

1. On the BSM Connector system, use `ovcoreid` to show the core ID of the system:

   ```
   ovcoreid
   ```

2. On the OMi certificate server (usually the data processing server), use `ovcert` to export the trusted certificate, type:

   ```
   ovcert -exporttrusted -file omi.cer
   ```

3. On the OMi certificate server (usually the data processing server), use `ovcm` to generate a certificate, type:

   ```
   ovcm -issue -file cert.cer -name <FQDN of BSM Connector> -coreid <OvCoreId of BSM Connector> -pass <password>
   ```

4. Securely transfer the generated files to the BSM Connector system.

5. Use `ovcert` to import the certificates from the generated files, type:

   ```
   ovcert -importtrusted -file omi.cer
   ```

   ```
   ovcert -importcert -file cert.cer
   ```

   The command prompts you for the password that you specified when you generated the certificates. Type the password and press **Enter**.

6. On any of the gateway server systems, use `ovcoreid` to show the core ID of the system:

   ```
   ovcoreid -ovrg server
   ```

7. On the BSM Connector system, set the manager and certificate server manually, type:

   ```
   ovconfchg -ns sec.cm.client -set CERTIFICATE_SERVER <FQDN of reverse proxy or load balancer>
   ```

   ```
   ovconfchg -ns sec.core.auth -set MANAGER <FQDN of reverse proxy or load balancer>
   ```

   ```
   ovconfchg -ns sec.core.auth -set MANAGER_ID <OvCoreId of any gateway server>
   ```

8. Run `oainstall` to complete the HP Operations Agent setup, type:

Windows 32-bit: `cscript "%OvInstallDir%\bin\OpC\install\oainstall.vbs" -a -c`

Windows 64-bit: `cscript "%OvInstallDir%\bin\win64\OpC\install\oainstall.vbs" -a -c`

Linux: `/opt/OV/bin/OpC/install/oainstall.sh -a -c`

9. Review the agent installation log file:

   Windows: `%OvDataDir%log\oainstall.log`

   Linux: `/var/opt/OV/log/oainstall.log`

   If the log file contains errors relating to the OvControl service failing to start or restart, complete the following steps:

   a. Manually reinstall ovcd as a Windows service, type:

      `ovcd -install`

   b. Rerun `oainstall`.

10. Securely delete any copies of the files that contain the certificates. Depending on how you generate and transfer the files, you may, for example, have copies in the following locations:
    - on the OMi data processing server
    - on the BSM Connector system
    - on a USB flash drive, CD, or other portable media

11. Add the BSM Connector system to the Connected server list in OMi. For details, see "How to Configure BSM Connector to Communicate with OMi" on page 23.

How to Manage BSM Connector with HPOM

To manage a BSM Connector system with HPOM, you need to configure an agent-based flexible management policy. The policy configures the OMi server as the primary manager of BSM Connector, and the HPOM management server as a secondary and action-allowed management server. This enables the HPOM management server to start actions, and deploy policies and packages.

The policy also configures HP Operations Agent to send the collected data as follows:

- The OMi server receives all events generated by BSM Connector policies that have the **Type** attribute set to **BSMC_Message**.

  The HPOM management server receives all events that do not have the **Type** attribute set to **BSMC_Message**.

  BSM Connector sets the event type attribute automatically to `BSMC_Message`. You can delete the value in a policy but BSM Connector inserts it again when you save the policy. The type attribute is available in the Advanced attributes tab of the policy editors.
- The BSM Connector server stores all metrics data.
- The OMi server receives all topology data (both HP Operations Agent and DFM-discovered data).

Note:

- The flexible management policy cannot be edited in the BSM Connector user interface.
- In the procedures below, use the `ovcoreid` command-line tool to find out the required core ID of

the system.

To manage BSM Connector with HPOM for Windows:

1. Exchange certificates between the OMi and HPOM systems. For information about exchanging certificates, see "To configure trusted certificates:" on page 29.
2. On the HPOM management server, use the **Configure Managed Nodes** dialog box to add the BSM Connector system as a managed node. In the node's properties, manually add the core ID and set the certificate state to `Installed`.
3. On the BSM Connector system, create and activate an agent-based flexible management policy:
   a. Make a copy of the flexible management policy template.

      BSM Connector on Windows

      Open a command prompt and type:

      ```
      copy "%OvDataDir%\conf\HPOprBSMC\f6b8aff0-f467-4851-8407-5b60395648b4_data.template" "%OvDataDir%\conf\HPOprBSMC\f6b8aff0-f467-4851-8407-5b60395648b4_data"
      ```

      ```
      copy "%OvDataDir%\conf\HPOprBSMC\f6b8aff0-f467-4851-8407-5b60395648b4_header.xml.template" "%OvDataDir%\conf\HPOprBSMC\f6b8aff0-f467-4851-8407-5b60395648b4_header.xml"
      ```

      BSM Connector on Linux

      Open a shell and type:

      ```
      cp /var/opt/OV/conf/HPOprBSMC/f6b8aff0-f467-4851-8407-5b60395648b4_data.template /var/opt/OV/conf/HPOprBSMC/f6b8aff0-f467-4851-8407-5b60395648b4_data
      ```

      ```
      cp /var/opt/OV/conf/HPOprBSMC/f6b8aff0-f467-4851-8407-5b60395648b4_header.xml.template /var/opt/OV/conf/HPOprBSMC/f6b8aff0-f467-4851-8407-5b60395648b4_header.xml
      ```

   b. Edit the policy data file `f6b8aff0-f467-4851-8407-5b60395648b4_data`.
   c. Locate the string `${OM_MGR_SRV}` and replace all occurrences with the FQDN of the HPOM management server.

      Locate the string `${OM_MGR_SRV_ID}` and replace all occurrences with the core ID of the HPOM management server.
   d. Locate the string `${OMi_MGR_SRV}` and replace all occurrences with the FQDN of the OMi gateway server.

      Locate the string `${OMi_MGR_SRV_ID}` and replace all occurrences with the core ID of the OMi data processing server.
   e. Save the policy data file. Import the policy in BSM Connector and activate it.
4. In the HPOM console, right-click the node that represents the BSM Connector system and select **All Tasks > Synchronize inventory > Packages**.
5. On the HPOM management server, check that you can manage the BSM Connector system,

type:

```
opcragt -status <BSM Connector hostname>
```

The output should indicate that the agent is running.

For more information about HPOM for Windows, see the HPOM for Windows online help.

To manage BSM Connector with HPOM for UNIX or Linux:

1. Exchange certificates between the OMi and HPOM systems. For information about exchanging certificates, see "To configure trusted certificates:" on the next page.
2. On the HPOM management server, add the BSM Connector system as a managed node, type:

   ```
   opcnode -add_node <BSM Connector hostname>
   ```
3. On the HPOM management server, specify the core ID of the BSM Connector server, type:

   ```
   opcnode -chg_id node_name=<BSM Connector hostname> id=<core ID of BSM Connector system>
   ```
4. On the BSM Connector system, create and activate an agent-based flexible management policy:
   a. Make a copy of the flexible management policy template.

      #### BSM Connector on Windows

      Open a command prompt and type:

      ```
      copy "%OvDataDir%\conf\HPOprBSMC\f6b8aff0-f467-4851-8407-5b60395648b4_data.template" "%OvDataDir%\conf\HPOprBSMC\f6b8aff0-f467-4851-8407-5b60395648b4_data"
      ```

      ```
      copy "%OvDataDir%\conf\HPOprBSMC\f6b8aff0-f467-4851-8407-5b60395648b4_header.xml.template" "%OvDataDir%\conf\HPOprBSMC\f6b8aff0-f467-4851-8407-5b60395648b4_header.xml"
      ```

      #### BSM Connector on Linux

      Open a shell and type:

      ```
      cp /var/opt/OV/conf/HPOprBSMC/f6b8aff0-f467-4851-8407-5b60395648b4_data.template /var/opt/OV/conf/HPOprBSMC/f6b8aff0-f467-4851-8407-5b60395648b4_data
      ```

      ```
      cp /var/opt/OV/conf/HPOprBSMC/f6b8aff0-f467-4851-8407-5b60395648b4_header.xml.template /var/opt/OV/conf/HPOprBSMC/f6b8aff0-f467-4851-8407-5b60395648b4_header.xml
      ```

   b. Edit the policy data file `f6b8aff0-f467-4851-8407-5b60395648b4_data`.
   c. Locate the string `${OM_MGR_SRV}` and replace all occurrences with the FQDN of the HPOM management server.

      Locate the string `${OM_MGR_SRV_ID}` and replace all occurrences with the core ID of the HPOM management server.
   d. Locate the string `${OMi_MGR_SRV}` and replace all occurrences with the FQDN of the OMi gateway server.

Locate the string `${OMi_MGR_SRV_ID}` and replace all occurrences with the core ID of the OMi data processing server.

e. Save the policy data file. Import the policy in BSM Connector and activate it.

5. On the HPOM management server, check that you can manage the BSM Connector system, type:

   `opcragt -status <BSM Connector hostname>`

   The output should indicate that the agent is running.

For more information about HPOM for UNIX or Linux, see the documentation that HPOM for UNIX or Linux provides.

To configure trusted certificates:

In an environment with multiple servers, you must configure each server to trust certificates that the other servers issued. This task involves exporting every server's trusted certificate, and then importing this trusted certificate to every other server. You must also update the agent's trusted certificates, so that the agent also trusts the OMi servers.

Configure trusted certificates for *every* server (OMi gateway and data processing servers, HPOM management server):

1. On *every* OMi server, export the trusted certificate to a file using the following command:

   `ovcert -exporttrusted -file <file>`

   The command generates a file with the name that you specify.

2. Copy each file to *every other* server, and then import the trusted certificate using the following commands:

   `ovcert -importtrusted -file <file>`

   `ovcert -importtrusted -ovrg server -file <file>`

3. On the BSM Connector system, update the trusted certificates using the following command:

   `ovcert –updatetrusted`

Configuring LW-SSO Authentication

BSM Connector can use lightweight single sign-on (LW-SSO) for the user authentication strategy, which allows the BSM Connector users to be managed in the same way as the OMi users and groups. LW-SSO is the recommended strategy for OMi. The BSM Connector installation program enables LW-SSO authentication by default.

You can configure LW-SSO authentication either when you configure BSM Connector using the `bsmc-conf` command or later on using the command `lwsso-conf`. For details on how to use `bsmc-conf`, see *BSM Connector Installation and Upgrade Guide*.

This section describes how to configure LW-SSO authentication for an existing BSM Connector installation using `lwsso-conf`:

- "How to configure LW-SSO authentication for an existing BSM Connector installation" on the next page
- "LW-SSO security warnings" on page 31

How to configure LW-SSO authentication for an existing BSM Connector installation

1. *Prerequisite:* Obtain the following information from OMi:
   - **OMi domain name.** You need to know the domain name of the OMi gateway server to which BSM Connector sends data (for example, `example.com`).

     If the OMi gateway servers and the BSM Connector run in different subdomains, for example, `deu.example.com` and `ind.example.com`, specify only the name of the parent domain, which is `example.com` in this example.

     BSM Connector and the OMi gateway server to which it is reporting must run in the same top-level domain.

   - **LW-SSO token key.** Obtain the token key defined in OMi as follows:
     i. In the OMi user interface, navigate to the Users and Permissions manager:

     Administration > Users > Authentication Management

     ii. In the Single Sign-On Configuration group, view the value of the Token Creation Key (initString) setting.

     iii. Record the value so it will be available to you later in this procedure.

     If the setting is not defined, work with an OMi administrator to define it :

     A. Click **Configure** to open the Authentication Wizard.

     B. Click **Single Sign-On** to view the Single Sign-On panel, and select **Lightweight** for the Single Sign-On Authentication mode.

     C. Generate the **Token Creation Key (initString)**. Record the value so it will be available to you later in this procedure.

     D. Define the domain or subdomains that are participating in the LW-SSO configuration:
     - If OMi and the BSM Connector instances are running in different subdomains type the name of the parent domain in the **HP Operations Manager i** field.
     - If OMi and the BSM Connector instances are running in the same domain, select **Parse automatically** and add the domain to the **Trusted Hosts/Domains** list.

     E. Click **Finish** to save your changes and close the wizard.

   - **BSM Connector groups and roles.** Define the groups and roles that are allowed to log into the BSM Connector:
     i. In the OMi user interface, navigate to the Infrastructure Settings manager:

     Administration > Setup and Maintenance > Infrastructure Settings

     ii. Click **Foundations** and select **Single Sign-On** in the drop-down list.

     iii. Set **Add user groups information to LW-SSO token** to true.

     The default group for BSM Connector is BSMC_ADMINS.

     iv. Set **Add user roles information to LW-SSO token** to true.

2. Use the `lwsso-conf` command to configure LW-SSO:

   `lwsso-conf.[bat|sh] -lwsso_key <lwssoKey> [-lwsso_domain <lwssoDomainName>] [-lwsso_groups <group0> [<group1> ...]]`

where:

- `-lwsso_key <lwssoKey>` is the token key (init string) generated in the OMi.

  Note: Single-sign on can only work if the token key that you type here is exactly the same as the token key on the OMi server.

- `-lwsso_domain <lwssoDomainName>` specifies the domain of the associated OMi gateway server.

- `-lwsso_groups <group0> [<group1> ...]` specifies the OMi users and roles that will have access to BSM Connector. Separate individual groups with spaces (for example, `-lwsso_groups BSMC_ADMINS SUPERUSER`).

3. Restart ovc:

   ```
   ovc -restart
   ```

LW-SSO security warnings

This section describes security warnings relevant to LW-SSO configuration. For more information about LW-SSO, see the Platform Administration Guide.

- Confidential initString parameter in LW-SSO security.

  LW-SSO uses symmetric encryption to validate an LW-SSO token. The initString parameter within the configuration is used for initialization of the secret key. An application creates a token, and each application that uses the same initString parameter validates the token.

  Caution:
  - It is not possible to use LW-SSO without setting the initString parameter.
  - The initString parameter is confidential information and should be treated as such in terms of publishing, transporting, and persistency.
  - The initString should be shared only between applications integrating with each other using LW-SSO.
  - The minimum length of the initString is 12 characters.

- LW-SSO should be disabled unless it is specifically required.
- Symmetric encryption implication.

  LW-SSO uses symmetric cryptography for issuing and validating LW-SSO tokens. Therefore, any application using LW-SSO can issue a token to be trusted by all other applications sharing the same initString. This potential risk is relevant when an application sharing the initString either resides or is accessible in an untrusted location.

Backing Up and Restoring BSM Connector Configuration and Policies

When backing up BSM Connector, keep in mind the following recommendations:

- Back up the BSM Connector configuration and user created polices. *Do not* back up runtime data, such as topology repositories or performance store data. You can find a detailed list of items to back up in the following sections.
- Restore the BSM Connector configuration to a clean BSM Connector installation (that is, a new installation without any modifications in the configuration), because some configuration files may be overwritten during the restore procedure.

BSM Connector Policies

The BSM Connector policies are stored in the following directory:

Linux: `/var/opt/ov/datafiles/policymanagement/store`

Windows: `%OvDataDir%datafiles\policymanagement\store`

Each policy consists of two files, a data file (`*_data`) and a header file (`*_header.xml`).

- **To back up** the policies:

  Copy the policies to a backup location.
- **To restore** the policies:
  a. Copy the policies from the backup location back to the folder where BSM Connector policies are stored.
  b. The copied policies are visible in the BSM Connector UI after you sign in. If a UI session is already open, click the refresh button.

BSM Connector Backsync Script

The backsync functionality is implemented as a single Perl script, located in:

Linux: `/var/opt/ov/conf/backsync/OMBackSync.pl`

Windows: `%OvDataDir%conf\backsync\OMBackSync.pl`

Back up this script if you modified any settings in it.

- **To back up** the script:
  a. Stop the `ombacksync` process:

     `ovc -stop ombacksync`
  b. Copy the `OMBackSync.pl` script to a backup location.

c. Start the `ombacksync` process:

`ovc -start ombacksync`

- **To restore** the script:
  a. Stop the `ombacksync` process.
  b. Copy the `OMBackSync.pl` script from the backup location back to the original location, overwriting the original file in the process.
  c. Start the `ombacksync` process.

BSM Connector Specific Settings in XPL

Several BSM Connector settings are set in XPL. For details on specific contexts, see the sections that follow.

- **To back up the complete local XPL configuration**:
  Copy the following file to a backup location:
  Linux: `/var/opt/ov/conf/xpl/config/local_settings.ini`
  Windows: `%OvDataDir%conf\xpl\config\local_settings.ini`
- **To restore** the XPL configuration, restore it on a per context basis, by using the command line interface provided with XPL:
  `ovconfchg -edit`
  Edit the context options as described in the following sections.

BSM Connector Web Application LWSSO Configuration

The LWSSO configuration consists of an XML configuration file and an XPL setting.

- **To back up** the configuration file and settings:
  a. Copy the following configuration file to a backup location:
  Linux: `/var/opt/ov/conf/HPOprBSMC/lwsso-config.xml`
  Windows: `%OvDataDir%conf\HPOprBSMC\lwsso-config.xml`
  b. Save (write down) the following XPL setting:

```
[HPBSMC.LWSSO]
BSMAccessGroups=<user_group_role_name>
```

- **To restore** the configuration file and settings:
  a. Replace the `lwsso-config.xml` file on the BSM Connector system with the one from the backup.
  b. Restore (add) the `BSMAccessGroups` XPL setting back to the XPL configuration.

BSM Connector Web Application Logging Configuration

The logging configuration for the BSMC Web Application is set in the following configuration file:

Linux: `/var/opt/ov/conf/HPOprBSMC/bsmclog4j.properties`

Windows: `%OvDataDir%conf\HPOprBSMC\bsmclog4j.properties`

- **To back up** the configuration:

  Copy the `bsmclog4j.properties` file to a backup location.
- **To restore** the configuration:

  Replace the `bsmclog4j.properties` file on the BSM Connector system with the one from the backup.

BSM Connector Web Application Ports Configuration

BSMC Web Application ports are configured in a configuration file and in XPL.

- **To back up** the configuration:
  a. Copy the following file to a backup location:

     Linux: `/var/opt/ov/conf/HPOprBSMC/ports.properties`

     Windows: `%OvDataDir%conf\HPOprBSMC\ports.properties`
  b. Save (write down) the following XPL configuration settings:

```
[NONOV.TomcatB]
HTTPPort=30001
HTTPSPort=30000
```

- **To restore** the configuration:
  a. Replace the `ports.properties` configuration file on the BSM Connector system with the one from the backup.
  b. Restore the port configuration in OVTomcat:

     Linux:

```
cd /opt/OV/nonOV/tomcat/b/bin
./ovtomcatbctl –sethttpport <HTTPPort_from_backed_up_XPL_config>
./ovtomcatbctl –sethttpsport <HTTPSPort_from_backed_up_XPL_config>
./ovtomcatbctl –configure
cd /opt/OV/bin./ovc –restart ovtomcatB
```

     Windows:

```
cd %OvInstallDir%\nonOV\tomcat\b\bin
cscript OvTomcatBCtl.vbs –sethttpport <HTTPPort_from_backed_up_XPL_config>
cscript OvTomcatBCtl.vbs –sethttpsport <HTTPSPort_from_backed_up_XPL_config>
cd %OvInstallDir%\bin\win64
ovc –restart ovtomcatB
```

BSM Connector Web Application User Configuration

BSM Connector stores its local admin user configuration in the file:

Linux: `/var/opt/ov/conf/HPOprBSMC/users.properties`

Windows: `%OvDataDir%conf\HPOprBSMC\users.properties`

- **To back up** the configuration file:

  Copy the `users.properties` file to a backup location.
- **To restore** the configuration file:

  Replace the `users.properties` file on the BSM Connector system with the one from the backup.

The BSM Connector REST Web Service Receiver Configuration

The BSM Connector REST Web Service Receiver is configured in XPL. Depending on the actual configuration, only some of the XPL configuration parameters listed below may be present.

- **To back up** the configuration:

  Save (write down) the following XPL configuration settings:

  ```
  [eaagt]
  RESTWS_AUTH_PASSWORD=<parameter_value>
  RESTWS_AUTH_USER=<parameter_value>
  RESTWS_PORT=<parameter_value>
  RESTWS_REGISTER_CB=<parameter_value>
  RESTWS_USE_BASIC_AUTH=<parameter_value>
  ```
- **To restore** the configuration:

  Restore the saved settings back to the XPL configuration on a freshly installed BSM Connector system.

BSM Connector Flexible Management Policy Template

BSM Connector contains an out-of-the-box template for creating an Event forwarding policy (the Flexible Management policy). If you made any customizations to this template file, create a backup of this file.

This template is used by the BSM Connector tool `deploy-flexmgmt.[bat|sh]` to create a policy eligible for event forwarding. The policy template consists of the following two files:

`f6b8aff0-f467-4851-8407-5b60395648b4_data.template`

`f6b8aff0-f467-4851-8407-5b60395648b4_header.xml.template`

The files are located in:

Linux: `/var/opt/ov/conf/HPOprBSMC/`

Windows: `%OvDataDir%conf\HPOprBSMC`

- **To back up** the template:

  Copy the listed `*.template` files to a backup location.
- **To restore** the template:

  Replace the listed `*.template` files on the BSM Connector system with the files from the backup.

Troubleshooting and Limitations

This section provides help in troubleshooting problems relating to BSM Connector in general.

This section includes:

- "Tomcat Logging" below
- "Verify That BSM Connector Can Send Events to OMi" below
- "Debug Trace Logging for Events" below

Tomcat Logging

Tomcat log output is written to the *<OvDataDir>*/log/Tomcat/stderr.log file.

Verify That BSM Connector Can Send Events to OMi

You can use the following procedure to verify that BSM Connector events actually arrive in the Event Browser.

1. Create an open message interface policy. For details, see "Open Message Interface Policies (Events only)" on page 151.
2. In the Options page of the policy, under **Unmatched Events**, select **are forwarded to OMi Server**. For details, see "Configuring Options in Open Message Interface Policies" on page 160.
3. Save the policy and activate it. The activation starts the `opcmsgi` process on the BSM Connector server.
4. Use the `opcmsg` command-line tool to submit messages to the open message interface policy. For details, see "opcmsg Command-Line Tool" on page 152.
5. In OMi, navigate to the Event Browser and verify that the events you submitted using `opcmsg` have arrived.

Debug Trace Logging for Events

To enable debug trace logging for an event, add the custom attribute `__TRACE__` and set it to any value. The attribute creates flow trace logs at the `INFO` level for this event.

The attribute can be set in the Custom Attributes tab of the policy that sends the event. Whenever this custom attribute is set on an event, trace output for this event appears in the trace logs:

- OMi Data Processing Server: `log/opr-backend/opr-flowtrace-backend.log`
- OMi Gateway Server: `log/wde/opr-gateway-flowtrace.log`

By default, only events with the custom attribute `__TRACE__` set are logged to the flow trace log files. To enable flow tracing for all events, set the flow trace log level to `DEBUG`.

Chapter 3: Security

This section includes:

- "Hardening the BSM Connector Platform" below
- "Permissions and Credentials" on the next page
- "Setting Up Smart Card Authentication" on the next page

Hardening the BSM Connector Platform

This section describes several configuration and setup options that can be used to harden the BSM Connector platform.

Network and system security has become increasingly important. As a third-party data integration tool, BSM Connector might have access to some system information which could be used to compromise system security if steps are not taken to secure it. You should use the configuration and setup options in this section to protect the BSM Connector platform.

This section includes:

- "BSM Connector Users" below
- "Hardening the BSM Connector Platform" above

BSM Connector Users

Administrator user. The administrator account configured with the `bsmc-conf` tool. This account is required when adding a BSM Connector integration to OMi. The administrator user can access only the BSM Connector user interface and does not have access to OMi.

Local users. You can add additional users to BSM Connector with the command-line tool **user**. The tool creates local user accounts in the BSM Connector local user store. These users can access BSM Connector only; they cannot access OMi or other OMi applications. For more information about the **user** tool, see "Local User Configuration Tool" on page 297.

Single sign-on. BSM Connector also supports Single Sign-On (SSO) authentication. The default single sign-on authentication strategy for OMi is Lightweight Single Sign-On (LW-SSO). LW-SSO is embedded in OMi and does not require an external computer for authentication.

LW-SSO enables a user to log into OMi once and gain access to all OMi applications without being prompted to log in again. The applications inside OMi trust the authentication, and you do not need further authentication when moving from one application to another. For example, if you configure BSM Connector to use LW-SSO, OMi users can launch the BSM Connector user interface without having to provide additional credentials.

For more information about LW-SSO, see the OMi Platform Administration Guide.

Permissions and Credentials

The following table provides you with basic information about the permissions needed to secure access to remote servers to collect data from databases.

Data Source	Protocol / Technology	User Permissions and Credentials	Notes
Database	JDBC	User credentials are needed to authenticate access to the particular database. Each database has a particular method for providing access control to the particular tables that need to be accessed.	The user needs sufficient permission to execute any specified SQL statements.

Setting Up Smart Card Authentication

BSM Connector supports user authentication using smart cards. If smart card authentication is configured, you cannot log in without a valid smart card.

Learn More

Smart Card Authentication

Smart cards are physical devices used to identify users in secure systems. These cards can be used to store certificates both verifying the user's identity and allowing access to secure environments.

BSM Connector can be configured to use these certificates in place of the standard model of each user manually entering a user name and password. When using smart cards with BSM Connector, users can only log in using the smart card.

Tasks

This section includes:

- "Enable Smart Card Authentication in BSM Connector" below
- "Disable Smart Card Authentication in BSM Connector" on page 40

Enable Smart Card Authentication in BSM Connector

To configure smart card authentication in BSM Connector, complete the following tasks:

1. Import the certificate of your certificate authority to a truststore. You can use an existing one, or create a new one.

Go to the folder `C:\Program Files\HP\HP BTO Software\nonOV\jre\b\bin` (Windows) or `/opt/OV/nonOV/jre/b/bin` (Linux) and run the following command:

```
keytool -import -trustcacerts -alias <your alias> -keystore <path to the truststore file> -file <root CA certificate file>
```

For example, on Linux, to add the certificate "hpca2ssG2_ns.cer" to the truststore "newTrustStore.jks", run:

```
./keytool -import -trustcacerts -alias TSTORE1 -keystore /tmp/newTrustStore.jks -file /tmp/hpca2ssG2_ns.cer
```

You need to define a password and agree to add the certificate to the truststore.

2. Configure the BSM Connector OVTomcat server to require a client certificate for mutual authentication and to use the truststore to which you imported the certificate.

   The file `server.xml` used by OV Tomcat is recreated on each ovc start, with XPL values replacing the values from templates. Therefore run the Operations Agent tool `ovconfchg` to change the settings in the XPL configuration:

   Run `ovconfchg -edit` to open the default system editor (Notepad on Windows, vi on Linux) and load thecurrent XPL configuration.

   The Tomcat related XPL settings are in the XPL namespace NONOV.TomcatB. Configure the Tomcat server to request a client certificate by locating the following section Locate the following seciton:

   ```
   <Connector port="30000" maxThreads="150"
              minSpareThreads="25" maxSpareThreads="75"
              enableLookups="true" disableUploadTimeout="true"
              acceptCount="100" debug="0"
              scheme="https" secure="true" clientAuth="false"
              sslProtocol="TLS"
              KeystoreFile="../groups/serverKeystore"
              KeystoreType="JKS"
              KeystorePass="changeit"/>
   ```

   Change `clientAuth` to `"true"`, and add the following attributes:

   ```
              truststoreFile="<path to the truststore file>"
              truststorePass="<password>"
              truststoreType="JKS"
    />
   ```

   Alternatively, change the settings directly, using the `-set` option:

   ```
   ovconfchg -ns <namespace> -set <parameter> <value>
   ```

   For example:

   ```
   ovconfchg -ns NONOV.TomcatB -set clientAuth "true" -set truststoreFile "<path to the truststore file>" -set truststorePass "<password>" -set truststoreType "JKS"
   ```

3. Start OVTomcat:

   ```
   ovc -start ovtomcatB
   ```

4. Add users to BSM Connector using the BSM Connector **user** command line tool:

a. In the Subject Alternative Name (SAN) field of the certificate, look for the value of the User Principal Name (UPN) in Other Name (OID: 1.3.6.1.4.1.311.20.2.3).

b. Use the **user** command-line tool to add a user to BSM Connector:

```
user -add <value of UPN> <password>
```

> **Note:** The user tool requires a password for each user. However, the password is not used when logging into OMi connector using a smart card. Users must enter their smart card PIN instead.

For more information on the **user** command-line tool, see "Local User Configuration Tool" on page 297.

Disable Smart Card Authentication in BSM Connector

1. The file `server.xml` used by OV Tomcat is recreated on each ovc start, with XPL values replacing the values from templates. Therefore run the Operations Agent tool `ovconfchg` to change the settings in the XPL configuration:
   - Run `ovconfchg -edit` to open the default system editor (Notepad on Windows, vi on Linux) and load thecurrent XPL configuration.

     The Tomcat related XPL settings are in the XPL namespace NONOV.TomcatB. Configure the Tomcat server to not request a client certificate by locating the following section:

     ```
     <Connector port="30000" maxThreads="150"
                minSpareThreads="25" maxSpareThreads="75"
                enableLookups="true" disableUploadTimeout="true"
                acceptCount="100" debug="0"
                scheme="https" secure="true" clientAuth="true"
                sslProtocol="TLS"
                KeystoreFile="../groups/serverKeystore"
                KeystoreType="JKS"
                KeystorePass="changeit"
                truststoreFile="../templates.certificates/truststore.jks"
                truststorePass="<password>"
                truststoreType="JKS"
     />
     ```

     Change `clientAuth` to `"false"`.

   - Alternatively, change the settings directly, using the `-set` option:

     ```
     ovconfchg -ns <namespace> -set <parameter> <value>
     ```

     For example:

     ```
     ovconfchg -ns NONOV.TomcatB -set clientAuth "false"
     ```

2. Restart ovc:

   ```
   ovc -restart ovtomcatB
   ```

Chapter 4: Event Synchronization

BSM Connector enables you to access event sources, and, if certain conditions apply, to forward the detected events as Operations Manager i (OMi) events to OMi. After OMi receives an event, you can keep it up to date on the event source by configuring OMi and BSM Connector to synchronize event changes back to the third-party system that generated the original event. For example, if an OMi operator closes an event originating from NNMi, a notification can be automatically sent to NNMi.

Note: Event changes that are synchronized back only include lifecycle changes to the state closed (that is, the state is only updated when it changes to closed).

Synchronized events have additional runtime parameters that can be inserted in URL tools. For more information about URL tools, see the OMi online help.

To configure event synchronization:

1. Configure the BSM Connector server as a connected server in OMi:

   Administration > Setup and Maintenance > Connected Servers

   For full details about how to configure a BSM Connector as a connected server, see "How to Configure BSM Connector to Communicate with OMi" on page 23.
2. On the BSM Connector **Event Drilldown** tab, you can set the default event drilldown in OMi by specifying the port and URL path of the third-party system for which the drilldown is configured, if these settings are not already defined in BSM Connector. For more information on how to configure event drilldown in OMi, see "How to Configure Drilldown into Third-Party Systems" on page 46.

   However, any event drilldown settings defined in BSM Connector for the respective policies will override the default settings made during the BSM Connectorconfiguration in OMi.
3. Configure policies to include the source event ID in the generated event.

   For more information, see "How to Configure Policies for Event Synchronization" on the next page.
4. Write a Perl script that receives the event changes from OMi and forwards them to the third-party system .

   For more information, see "How to Write Perl Scripts for Event Synchronization" on page 43.

How to Configure Policies for Event Synchronization

The event that BSM Connector forwards to OMi must include the original ID of the event in the third-party system. Otherwise OMi and BSM Connector do not know which event to update in the third-party system.

To configure policies for event synchronization:

1. Create or edit an event integration policy.
2. Set the **Source Event ID** attribute in the **Event Attributes** tab.

   The source event ID attribute may not be available in all event attributes tabs. For example, the default event attributes of SNMP trap and open message interface policy editors do not include this attribute.
3. Click **Save and Close** to save the policy and close the editor.
4. *Optional.* Activate the policy for your changes to take effect.

The next step in configuring event synchronization is to create a Perl script that receives the event changes and forwards them to the corresponding third-party system. See "How to Write Perl Scripts for Event Synchronization" on the next page.

How to Write Perl Scripts for Event Synchronization

The `ombacksync` process on the BSM Connector server receives event updates from the OMi data processing server. To forward these changes to the third-party system, you must provide the `OMBackSync.pl` Perl script that closes the event in the third-party system.

The Perl script must call the subroutine `OMBackSync`, which supports the following parameters:

- `Operation` (`Init` or `Close`)
- `ID` (`Close` only)

When the `ombacksync` process starts, it processes the Perl script and calls the subroutine `OMBackSync` with the parameter `Init`. When the process receives an event with the status closed, `ombacksync` calls the subroutine `OMBackSync` with the parameters `Close` and `ID`.

The script must be named `OMBackSync.pl`. Only one instance of the script can exist in BSM Connector. If multiple integrations are hosted on BSM Connector, all integrations must merge their event synchronization code into a single `OMBackSync.pl` file. When uninstalling an integration, the integration must remove its event synchronization code from the file.

BSM Connector provides an example Perl script (`OMBackSync.pl`) that writes the time, date, operation, and ID to the file `OMBSOutput.txt`.

The `OMBackSync.pl` file is located at:

Windows: `"%OvDataDir%\conf\backsync\OMBackSync.pl"`

Linux: `/var/opt/OV/conf/backsync/OMBackSync.pl`

The `OMBSOutput.txt` file is located at:

Windows: `"%OvDataDir%\tmp\OMBSOutput.txt"`

Linux: `/var/opt/OV/tmp/OMBSOutput.txt`

Caution: The `OMBackSync.pl` script does *not* encrypt user credentials. If you need to add user credentials to the script, for example for connecting to an external management tool, use file system permissions to restrict access to the script.

To forward event changes to the third-party system:

1. Write a Perl script that calls the subroutine `OMBackSync` with the parameters `Operation` and `ID`.
2. Name your Perl script `OMBackSync.pl` and place it in the following folder on the BSM Connector server:

   Windows: `"%OvDataDir%\conf\backsync\OMBackSync.pl"`

   Linux: `/var/opt/OV/conf/backsync/OMBackSync.pl`
3. Restart the `ombacksync` process on the BSM Connector server:

   ```
   ovc -restart ombacksync
   ```

Troubleshooting:

If the `ombacksync` process encounters syntax errors in the Perl script, it generates an event describing the problem and stops. Correct the syntax and restart the `ombacksync` process.

Chapter 5: UI Drilldown

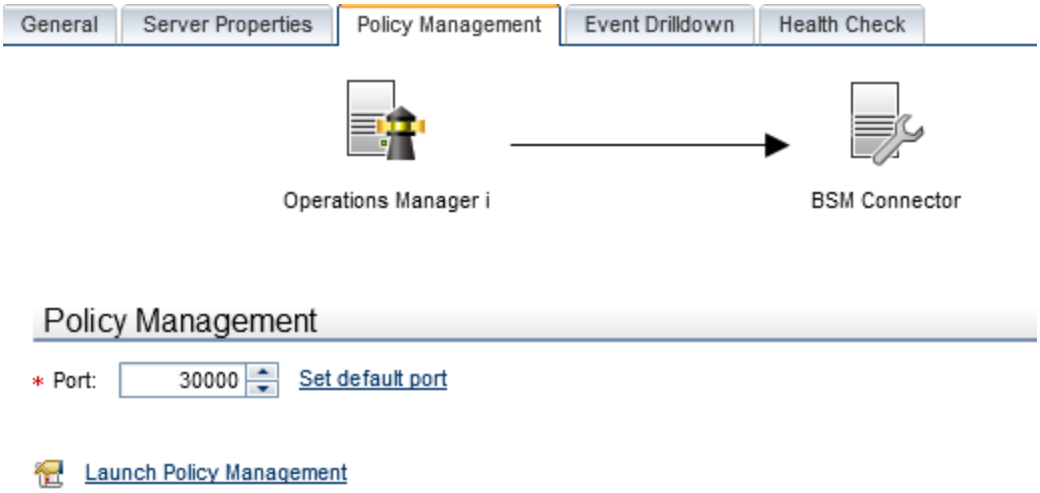
OMi enables you to start the user interface of BSM Connector or of the third-party system directly from within OMi. The following integrations are available:

BSM Connector UI integrations

- **Connected Servers manager.** Navigate to **Administration > Setup and Maintenance > Connected Servers**. In the Connected Servers pane, locate the BSM Connector you want to launch and open it for editing.

  In the Edit Server Connection window, navigate to the Policy Management tab, and click the **Launch Policy Management** link.

  The following illustration shows the **Launch Policy Management** link in the Connected Servers manager.



- **Event context.** In the OMi Event Browser, right-click an event that has been forwarded from a BSM Connector server, and then select **Configure > Integration Policies**.

Depending on the security settings in your environment, you may have to log on or provide a smart card PIN. If you are prompted for login information, type the user name and password of a local user account. BSM Connector does not require you to log on if single sign-on is configured.

Third-party system UI integration

In the OMi Event Browser, right-click an event that has been forwarded from a BSM Connector server, and then select **Show > Event in Source Manager**.

UI integrations with third-party systems are not available out of the box. For information how to configure a UI integration, see "How to Configure Drilldown into Third-Party Systems" on the next page.

How to Configure Drilldown into Third-Party Systems

OMi enables operators to start the user interface of third-party systems in the context of an event in the Event Browser. To launch the third-party system's user interface, right-click the event in the Event Browser and select **Show > Event in Source Manager**.

You can configure drilldown in the following ways:

- Add the third-party system to the BSM Connector integration server configuration in OMi.
- Specify source information in the policies that forward the source events to OMi.

If a policy and a BSM Connector server configuration both contain information about the third-party system, the information in the policy takes precedence.

Tip: Specify the third-party system in the BSM Connector Connected Servers configuration if you want to centrally configure drilldown to the third-party system, as opposed to adding this information to individual policies.

This section includes:

- "To specify the third-party system in the BSM Connector Event Drilldown Settings:" below
- "To configure policies for event drilldown:" on the next page

To specify the third-party system in the BSM Connector Event Drilldown Settings:

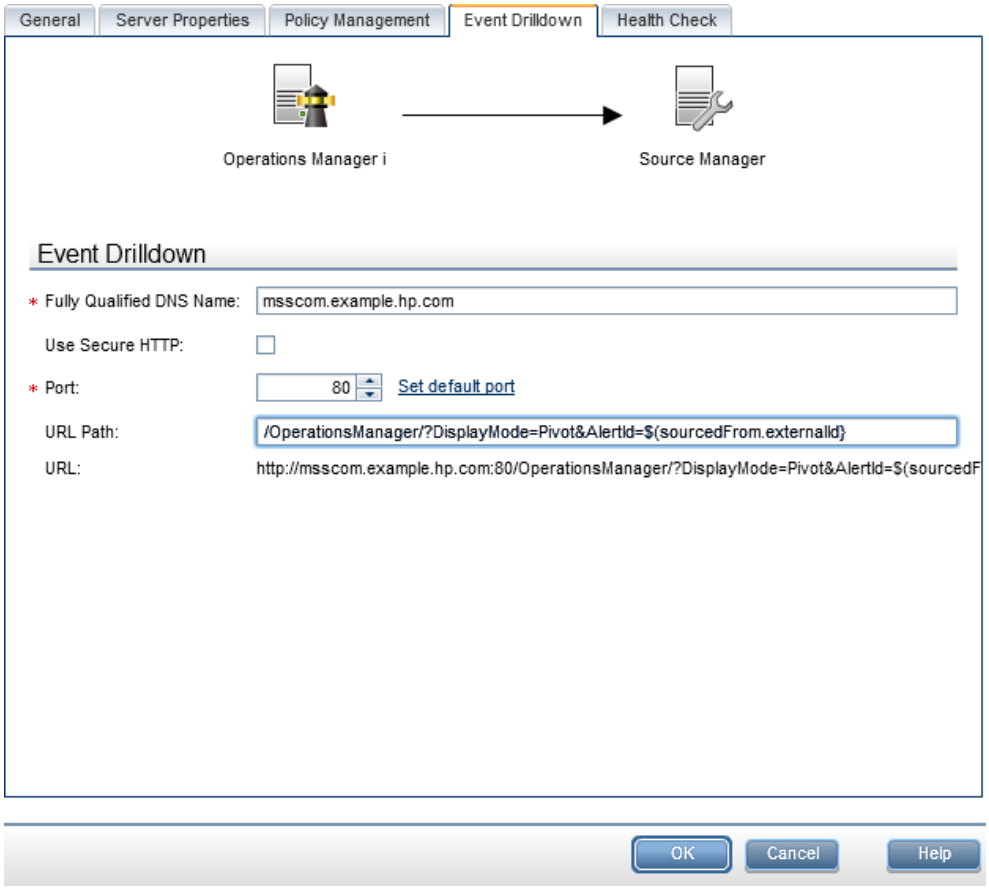
1. Configure the BSM Connector server as a connected server in OMi:

   Administration > Setup and Maintenance > Connected Servers
2. On the **Event Drilldown** tab, complete the following information:

   Fully Qualified DNS Name. Enter the fully qualified DNS name of the computer that hosts the third-party system for which the drilldown is configured (for example, `SCOM.mgmt2.example.com`). For a secure connection, enable the Use Secure HTTP checkbox.

   Port. Specify the communication port on the third-party system.

   URL Path. Specify the root URL of the event in the third-party system. This is the base path of the URL, and does not include FQDN or port.



To drill down to a specific event in the third-party system, append the variable `${sourcedFrom.externalId}` to the URL. OMi replaces the variable at runtime if the event contains the source event ID.

For more information about adding the source event ID to a policy, see "How to Configure Policies for Event Synchronization" on page 42.

For full details about how to configure a BSM Connector as a connected server, see "How to Configure BSM Connector to Communicate with OMi" on page 23.

To configure policies for event drilldown:

1. Create or edit an event integration policy.
2. Find the **Event Drilldown URL** attribute in the **Advanced** tab.

   The event drilldown URL attribute may not be available in all advanced attributes tabs. For example, the default advanced attributes of SNMP trap and open message interface policy editors do not include this attribute.
3. In the **Event Drilldown URL** field, type the URL of the event in the third-party system. This is the complete path of the URL, and includes the FQDN (Fully Qualified Domain Name) of the computer that hosts the third-party system, the communication port, and the root URL path. Example:

   ```
   http://nnmi.example.com:8004/nnm/launch?cmd=showForm
   &objtype=Incident&objuuid=$OPC_CUSTOM[nnm.incident.uuid]
   &menus=true
   ```

This event attribute can also be set by OMi based on connected server configuration. If a policy and a connected server configuration both contain event drilldown information, the information in the policy takes precedence.

4. Click **Save and Close** to save the policy and close the editor.
5. Activate the policy for your changes to take effect.

Part II: Working with BSM Connector

This section includes:

- "Policy Management" on page 50
- "Collecting and Viewing Metrics Data" on page 60
- "Collecting Topology Data" on page 74

Chapter 6: Policy Management

Policies are collections of configuration information used to configure HP Operations Agent on the BSM Connector server to process integration data. When you develop policies, you decide what kinds of data to process, how often to process, what to look for in the data, and what to do if certain conditions apply.

BSM Connector provides policy editors for different policy types (for example, for XML file policies). You can import policies developed and exported on other servers, for example, HP Operations Manager (HPOM) management servers, HP Network Node Manager i (NNMi) management servers, or other BSM Connector servers.

When you create a new policy or import a policy, the policy exists in the BSM Connector policy repository but does not function yet. You must first activate the policy for it to start accessing the corresponding event source or discovering topology data.

Tip: You can sort the information that appears in the columns in the BSM Connector policy list so that data appears in either ascending or descending order, indicated by either an up or down arrow at the top of the column. In addition, you can change the order of columns by dragging columns to other positions.

To access

To start the BSM Connector user interface, open a Web browser at the following URL:

`https://<BSM Connector system>:30000/bsmconnector/`

<BSM Connector system> is localhost or the hostname of the BSM Connector server. If Apache Tomcat is configured to use a different port, use this port instead.

Learn More

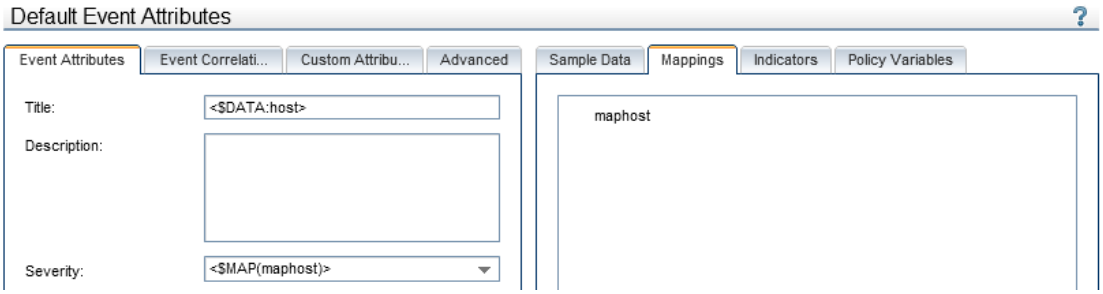
This section includes:

- "Drag and drop" below
- "Policies in cluster environments" on the next page

Drag and drop

Most BSM Connector policy editors offer drag-and-drop to quickly insert sample data, mappings, policy variables, and so on in text boxes.

The following illustration shows the default event attributes page of structured log file policies. You can select data in the Sample Data, Mappings, Indicators, and Policy Variables tabs on the right and drag it to the Event Attributes, Event Correlation, Custom Attributes, and Advanced tabs on the left. BSM Connector inserts the data at the current cursor position.



Policies in cluster environments

Import and activate the same set of policies on all nodes in the cluster.

Make sure that you do not access data stored on the shared disk with the same policies activated on multiple cluster nodes. This may result in duplicate events in OMi after a failover. For example, do not activate the same XML file policies on more than one cluster node if the policies read an XML file on a shared disk.

If the IP address of the resource group (also known as virtual IP address) sends SNMP events that you plan to capture, make sure that the IP address is set up as a CI (configuration item) with the attribute "host is virtual" in OMi Otherwise the events display as unmapped events in OMi.

Tasks

Related tasks

- "How to Edit Policies" on the next page
- "How to Copy Policies" on page 54
- "How to Delete Policies" on page 54
- "How to Activate and Deactivate Policies" on page 54
- "How to Export Policies" on page 55
- "How to Import Policies" on page 57
- "How to Configure Policy Management Options" on page 58

UI Descriptions

Policy Management

UI Element	Description
	Refresh. Reloads the policy list.
	Create. Opens a policy editor to create a new policy.

UI Element	Description
	Edit. Opens the selected policy in a policy editor for editing.
	Copy. Opens the Copy Policy dialog box.
	Delete. Deletes the currently selected policy.
	Activate. Activates the currently selected policy so that it starts functioning.
	Deactivate. Deactivates the currently selected policy. The policy stops on the BSM Connector server.
	Break Edit Lock. Removes the edit lock from the currently selected policy.
	Import. Opens a file selection dialog box for you to select the policy files to import.
	Options. Opens the Options dialog box
Policy Type	The policy type indicates the data source of the policy (for example, structured log file, database, or open message interface).
Integration Type	The integration type indicates the type of data that the policy integrates (for example, event, metrics, or topology).
Policy ID	*Optional:* GUID (globally unique identifier) of the policy.
Name	Name of the policy.
Description	Description of the policy.
Activation State	The activation state indicates if a policy is activated, deactivated, or needs reactivation after modification.
Edited by	The lock icon and the name of the editing user indicate that a policy is being edited and by whom.

How to Edit Policies

BSM Connector enables multiple users to connect to the same server at the same time. However, only one user at a time can edit a policy to prevent changes from other users overwriting each other.

When you edit a policy, BSM Connector locks the policy for you so that only you can save the policy. Note that if you open a policy more than once, only the first editor instance receives the lock and can save the policy.

Locked policies display with a lock icon and the name of the editing user in the list of policies in the BSM Connector user interface. (The tooltip explains who locked the policy and when.)

BSM Connector releases the lock when the policy is closed or when a user forcefully breaks the lock. It is recommended that you only break the edit lock when you have reason to believe that the locked policy editor was abandoned or stopped working. Policy editors that have been forcefully unlocked by another user change to read-only mode and you can no longer save any changes made.

Tip: When you open a locked policy for viewing, reload the policy from time to time to ensure that you are viewing the most recently saved version.

This section includes:

- "To edit unlocked policies:" below
- "To break the edit lock on locked policies:" below

To edit unlocked policies:

1. In the list of policies in the BSM Connector user interface, select the policy that you want to edit.
2. Click in the toolbar. The policy editor opens. The policy is marked with the lock icon and the name of the editing user in the list of policies in the BSM Connector user interface.
3. Modify the policy.
4. Click **Save and Close** to save the policy and close the editor.

Alternatively, right-click the policy that you want to edit and click **Edit** in the context menu. Modify the policy and click **OK** to save your changes.

To break the edit lock on locked policies:

1. In the list of policies in the BSM Connector user interface, select the locked policy that you want to edit.
2. Click in the toolbar. A dialog box opens and lists the name of the editing user, the hostname of the BSM Connector server, and the date and time the policy was locked.

   The date and time displays using the current time zone of the computer on which the BSM Connector user interface runs. The language setting of the Web browser determines the date and time format (for example, `09/14/2010 8:16:38 AM` for English (United States)). If the Web browser and the computer on which the BSM Connector server run have different language settings, the language setting of the Web browser takes precedence. However, English is the default language if the Web browser is configured to use a language that is not available on the server.

3. Click **OK** to break the edit lock. The lock icon and the name of the editing user disappear from the list of policies in the BSM Connector user interface..
4. Open the policy for editing and modify it.
5. Click **Save and Close** to save the policy and close the editor.

Alternatively, right-click the locked policy that you want to edit and click **Unlock** in the context menu. Modify the policy and click **OK** to save your changes.

Tip: You can unlock a policy at any time, even while editing.

How to Copy Policies

Instead of creating a new policy from scratch, you can copy an existing policy and change the copy to meet your needs. Copied policies are by default deactivated.

Note: You can copy only one policy at a time.

How to copy a policy

1. In the list of policies in the BSM Connector user interface, select a policy.
2. Click in the toolbar. The **Copy Policy** dialog box opens.
3. Change the policy name and optionally the description.
4. Click **OK**. The copied policy appears in the list of policies in the BSM Connector user interface and is by default deactivated.

UI description

UI Element	Description
Name	Name of the policy. You can use spaces in the name. The equal sign (=) is not allowed.
Description	Description of what the policy does. You might also add other notes (for example, data sources that are used).

How to Delete Policies

You can only delete deactivated policies and policies that are not being edited by other users. When you delete a policy the policy files are deleted from the file system. To temporarily stop a policy from functioning, deactivate the policy instead.

To delete policies:

1. In the list of policies in the BSM Connector user interface, select the policies that you want to delete.
2. Make sure that the policies are deactivated and not being edited.
3. Click in the toolbar. A message box opens.
4. Confirm that you want to delete the policies.

How to Activate and Deactivate Policies

When you create a new policy or import a policy, the policy exists in the BSM Connector policy repository but does not function yet. You must first activate the policy for it to start accessing the

corresponding data source.

When you edit an existing, active policy, the previous version of the policy remains active on the BSM Connector server and you must reactivate the policy for your changes to take effect.

When you deactivate a policy, the policy remains in the BSM Connector policy repository but does not function until it is activated again.

This section includes:

To activate policies:

1. In the list of policies in the BSM Connector user interface, select the policies that you want to activate. The activation state of at least one of the selected policies must be `deactivated` or `activated (reactivate for new version)`. (If you include an already activated policy in your selection, the policy is ignored and not activated again.)
2. Click in the toolbar. The activation state changes to `activated`.

Alternatively, right-click the policies that you want to activate and click **Activate** in the context menu. The activation state changes to `activated`.

To deactivate policies:

1. In the list of policies in the BSM Connector user interface, select the policies that you want to deactivate. (If you include an already deactivated policy in your selection, the policy is ignored and not deactivated again.)
2. Click in the toolbar. The activation state changes to `deactivated`.

Alternatively, right-click the policies that you want to deactivate and click **Deactivate** in the context menu. The activation state changes to `deactivated`.

How to Export Policies

BSM Connector enables you to import policies developed on other servers, for example, HP Operations Manager (HPOM) management servers, HP Network Node Manager i (NNMi) management servers, or other BSM Connector servers. Most policies can simply be copied from the source BSM Connector server and imported in the target server. However, some policies may contain information that does not apply to the target BSM Connector server. Policies may also contain sensitive information such as user names and passwords that must be secured before the distribution.

Policy content that should be modified or removed before distribution

The following table describes policy content that should be modified or removed before the distribution. In addition to the items listed, you may need to adapt other content to prepare your policies for distribution, for example policy conditions.

Policy Type	Policy Field	Recommended Action
Database policies	Connect string	If necessary, enter placeholders for the database server name and port, for example enter: `jdbc:sqlserver://<hostname>:<dbport>;Database=<name>;`
	Username	Enter a placeholder for the database user name, for example: `<db user name>`.
	Password	Remove the password from the policy.
Structured log file policies	Log File Path / Name	If necessary, change the path or file name.
Scheduled task policies	Password	Remove the password from the policy.
	Username	Remove the user name from the policy.
	Password in commands, Perl, or VBScript	Remove the password from the policy. Alternatively, use the ExecuteCommand method `password` with an encrypted password. For details, see ExecuteCommand: password.
REST Web Service listener policies	Path	Part of the Effective URL for the REST Web Service Topology policy
XML file policies	Log File Path / Name	If necessary, change the path or file name. **Tip:** You can use Windows environment variables or call a command or script that returns the log file name and path. For details, see "Configuring the Data Source in XML File Policies" on page 264
	Topology XML File	Path to the source file for the XML Topology policy.

To export policies:

1. Prepare the policies for distribution:

a. Create copies of the policies. For details, see "How to Copy Policies" on page 54.
b. Edit the copied policies and modify them as suggested in "Policy content that should be modified or removed before distribution" on the previous page.

2. Identify the policy ID of each policy you want to export:
   a. In the toolbar, click to open the Options dialog box.
   b. In the Options dialog box, select **Policy ID** and click **OK**.
      The policy list now shows the ID of each policy in an additional column.
3. On the BSM Connector server, navigate to the policy store:
   - Windows: `"%OvDataDir%\datafiles\policymanagement\store"`
   - Linux: `/var/opt/OV/datafiles/policymanagement/store`
4. Copy the header (`<policyID>_header.xml`) and the data (`<policyID>_data`) files to a temporary location on the target BSM Connector server and import them. For more information about importing policies, see "How to Import Policies" below.

How to Import Policies

BSM Connector provides policy editors for different policy types (for example, for XML file policies). You can import policies developed and exported on other servers, for example, HP Operations Manager (HPOM) management servers, HP Network Node Manager i (NNMi) management servers, or other BSM Connector servers.

Policies Exported from HPOM

When you download policies on an HPOM management server, make sure the resulting policy files support the XML-based policy exchange format:

- HP Operations Manager for Windows: use the `ovpmutil` command-line tool.
- HP Operations Manager for UNIX or Linux: use the `opccfgdwn` command-line tool.

Policies Exported from NNMi

NNMi provides the `nnmopcexport.ovpl` command-line tool, which exports an SNMP trap policy for use with BSM Connector. For more information about running this tool and the NNMi integration in general, see the *NNMi Deployment Reference*.

Policies Exported from Other BSM Connector Servers

You can also import policies developed on other BSM Connector systems, for example, to ensure that the same set of policies is available on multiple BSM Connector systems. This is necessary, for example, when running BSM Connector in a cluster environment.

BSM Connector stores policies in the following folders:

- Windows: `%OvDataDir%\datafiles\policymanagement\store`
- Linux: `/var/opt/OV/datafiles/policymanagement/store`

To import policies by using the BSMC Policy Management UI:

1. In the BSM Connector user interface, click in the toolbar. A file selection dialog box opens.
2. Navigate to the policy files and, for each policy, select both the header (`*_header.xml`) and the data (`*_data`) files.
3. Click **Open** (on Windows) or **OK** (on Linux) to start the import process.

   If the policies with the same policy ID already exist in BSM Connector, you are asked whether you would like to replace them with the newly imported policies.

   The imported policies appear in the list of policies in the BSM Connector user interface. They are by default deactivated.
4. If necessary, edit the imported policies and adapt their contents to the new BSM Connector server.
5. *Optional:* Activate the policies.

To import policies by using the command-line interface:

You can import policies by using the `bsmc_policy.[bat|sh]` script. The script is located at:

Windows:

```
%ovdatadir%installation\HPOprBSMC\bsmc-policy.bat
```

Linux:

```
/var/opt/OV/installation/HPOprBSMC/bsmc-policy.sh
```

Import a policy by using the following command:

`bsmc-policy.[bat|sh] -import -header <full path to policy header file> -data <full path to policy data file>`

How to Configure Policy Management Options

You can choose to show any combination of columns in the BSM Connector policy list

> **Tip:** Click **Default** to restore the default selections.

How to change the column display

1. Click in the toolbar. The **Options** dialog box opens.
2. Select or clear the check box beside the column you want to show or hide.
3. Click **OK** to save your changes.

UI description

UI Element	Description
Policy Type	The policy type indicates the data source of the policy (for example, structured log file, database, or open message interface).
Integration Type	The integration type indicates the type of data that the policy integrates (for example, event, metrics, or topology).
Policy ID	GUID (globally unique identifier) of the policy.
Name	Name of the policy.
Description	Description of the policy.
Activation State	The activation state indicates if a policy is activated, deactivated, or needs reactivation after modification.
Edited by	The lock icon and the name of the editing user indicate that a policy is being edited and by whom.
Default	Click to restore the default selection.

Chapter 7: Collecting and Viewing Metrics Data

You enable capturing metrics data from third-party systems by configuring metrics policies.

Metrics policies depend on the default attributes and rules you define within the policies. The rules define the processing of incoming data and in combination with the defaults define the metrics forwarded to OMi.

How to Collect Metrics

This section contains the overview of the tasks required to perform when collecting metrics.

These tasks are the following:

- "Check the existing topology in OMi" below
- "Create a policy" below
- "View Integration Results" below

1. Check the existing topology in OMi

   When collecting metrics, you need to consider the following:

   - **CIs available in OMi**

     You probably already have CIs in the RTSM for which you want to report data by using BSM Connector policies, but you can create new CIs in BSM Connector integrations.

   - **Data that you want to be reported for these CIs**

     Consider what data you have for these CIs and how it can be attached to the CIs. For more information, see "Example – Create a Metrics Policy" on the next page.

2. Create a policy

   When creating a policy, you need to associate performance data or metrics with CIs that exist in RTSM. Make sure that the values in the BSM Connector data store (`Related CI`) correspond to the values in the RTSM model (`External ID`). If this is not the case, either create mappings or define the rules in BSM Connector so that they match the values in the RTSM model. For more information on how to discover which values are used in the RTMS model, see "Example – Create a Metrics Policy" on the next page.

   Note: The `External ID` values can consist from more than one property, for example, `hostname1;diskC;vol1`. In such a case, make sure that the corresponding policy sets the attribute Related CI for this particular metric to `hostname1;diskC;vol1`.

3. View Integration Results

   After activating the policy in BSM Connector, you can view the results in OMi, by using the **Performance Perspective** view. For more details, see "View the integration results" on page 69.

Example – Create a Metrics Policy

This example describes how to create a metrics integration policy to capture and forward metrics from a third-party system to OMi.

For an overview of the tasks that are required when collecting metrics, see "How to Collect Metrics" on the previous page.

This example includes the following steps:

- "Example – Create a Metrics Policy" above
- "Create a structured log file policy for metrics integration" on page 63
- "View the integration results" on page 69

1. Design stage

   You have an application named `My Oracle Monitoring`. This application writes various metrics from Oracle databases running on different computers to a log file.

   Note: Each table row represents a line in a log file, where the entries are separated with the "|" sign.

Log File Line Number	Data
1	1418724468|ORAPROD03;tungsten.elementary.com|tablespace_free_space [%]|39.3631591796875|tablespace_skew [%]|1.3631591796875|buffer_pool_hit_ratio [%]|98.3631591796875|in_memory_sort_ratio [%]|93.3631591796875|parse_to_execute_ratio [%]|999.363159179688|latch_hit_ratio [%]|89.3631591796875
2	1418724468|ORAPROD02;silver.elementary.com|tablespace_free_space [%]|39.4726257324219|tablespace_skew [%]|1.47262573242188|buffer_pool_hit_ratio [%]|98.4726257324219|in_memory_sort_ratio [%]|93.4726257324219|parse_to_execute_ratio [%]|999.472625732422|latch_hit_ratio [%]|89.4726257324219
3	1418724468|ORADEV;carbon.elementary.com|tablespace_free_space [%]|40.2490844726563|tablespace_skew [%]|2.24908447265625|buffer_pool_hit_ratio [%]|99.2490844726563|in_memory_sort_ratio [%]|94.2490844726563|parse_to_execute_ratio [%]|1000.24908447266|latch_hit_ratio [%]|90.2490844726563

Log File Line Number	Data
4	1418724468|ORAPROD04;platinum.elementary.com|tablespace_free_space [%]|40.1240844726563|tablespace_skew [%]|2.12408447265625|buffer_pool_hit_ratio [%]|99.1240844726563|in_memory_sort_ratio [%]|94.1240844726563|parse_to_execute_ratio [%]|1000.12408447266|latch_hit_ratio [%]|90.1240844726563
5	1418724468|ORAPROD01;palladium.elementary.com|tablespace_free_space [%]|40.2179565429688|tablespace_skew [%]|2.21795654296875|buffer_pool_hit_ratio [%]|99.2179565429688|in_memory_sort_ratio [%]|94.2179565429688|parse_to_execute_ratio [%]|1000.21795654297|latch_hit_ratio [%]|90.2179565429688

Entries in the log file have the following logical structure:

(`timestamp,entity_id,(performance_counter,counter_value){6}`).

Each line consists of two unique fields: `timestamp` and `entity_id`, followed by six repeating field pairs: `performance_counter` and `counter_value`.

2. Check the RTSM model

To check the RTSM model for values that should be matched by the associated CIs, follow these steps:

a. In OMi, select **Administration > RTSM Administration > Modeling > IT Universe Manager**. In the IT Universe Manager, select a view from the drop-down list provided on the Browse Views tab, and choose the one that you want to do the mapping for.

b. On Properties tab, Select `External ID` and check its value because it is a link between the RTSM model and metrics stored in the BSMC data store.

Note: Because `External Id` is not visible in the list of attributes by default, set it to be visible as follows:

i. In OMi, select **Administration > Setup and Maintenance > Infrastructure Settings**.

ii. In the Infrastructure Settings, select the Foundations context and choose `RTSM` from the provided drop-down list.

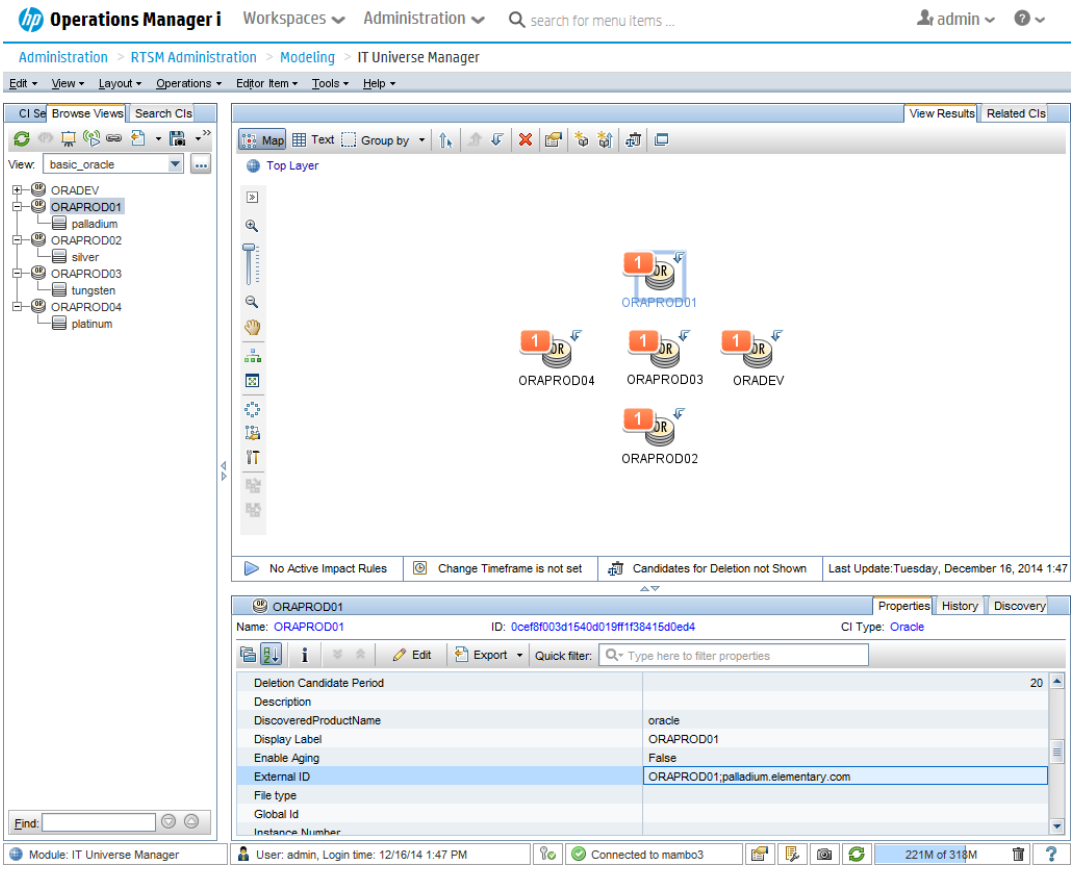
iii. In the **RTSM - General Settings** list, scroll down to the `Object Root` item.

Click the button on the right side to open the editor and set the value of the object root to `data`.

iv. Select **Administration > RTSM Administration > Modeling > CI Type Manager**.

v. Select the CI Type data. On the Attributes tab, double click `External ID` and enable the `Visible` checkbox from the Advanced tab. Click **OK**.

The value of BSMC metric attribute `Related CI` in the OA datastore must match the value of `External ID` in RTSM. To achieve this, either create mappings or define the rules in the BSM Connector appropriately, so that they match the values in the RTSM model. In this example, topology in RTMS is also created by the `My Oracle Monitoring` application, therefore `Related CI` matches `External ID`.

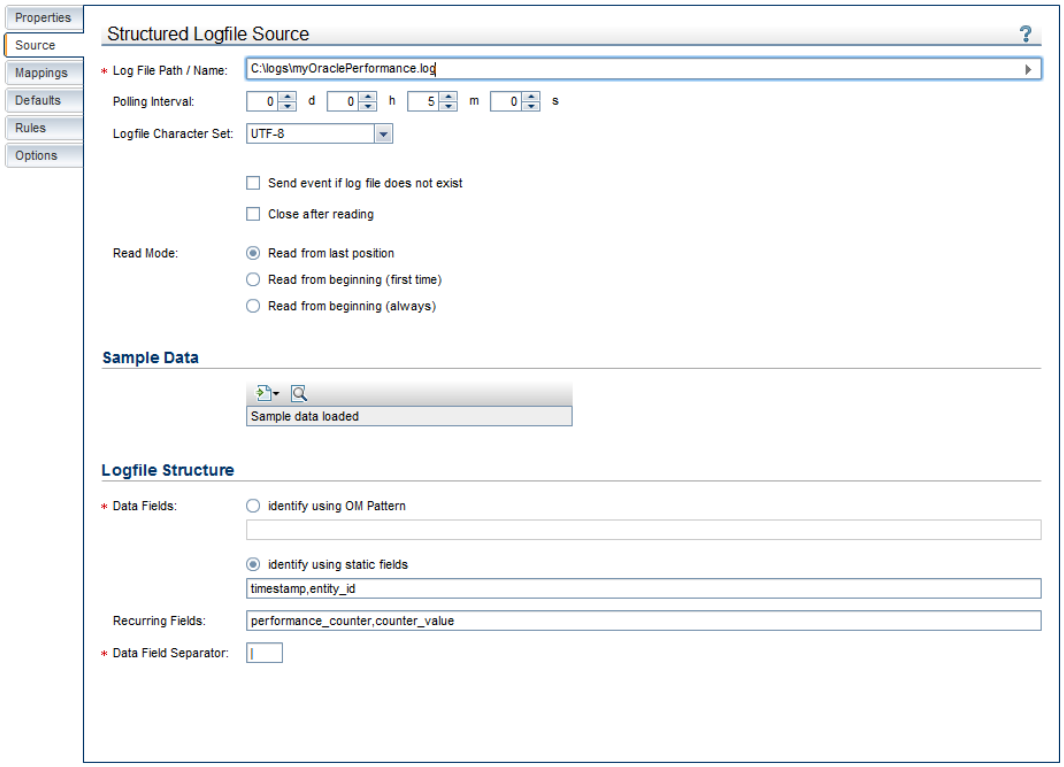


3. Create a structured log file policy for metrics integration

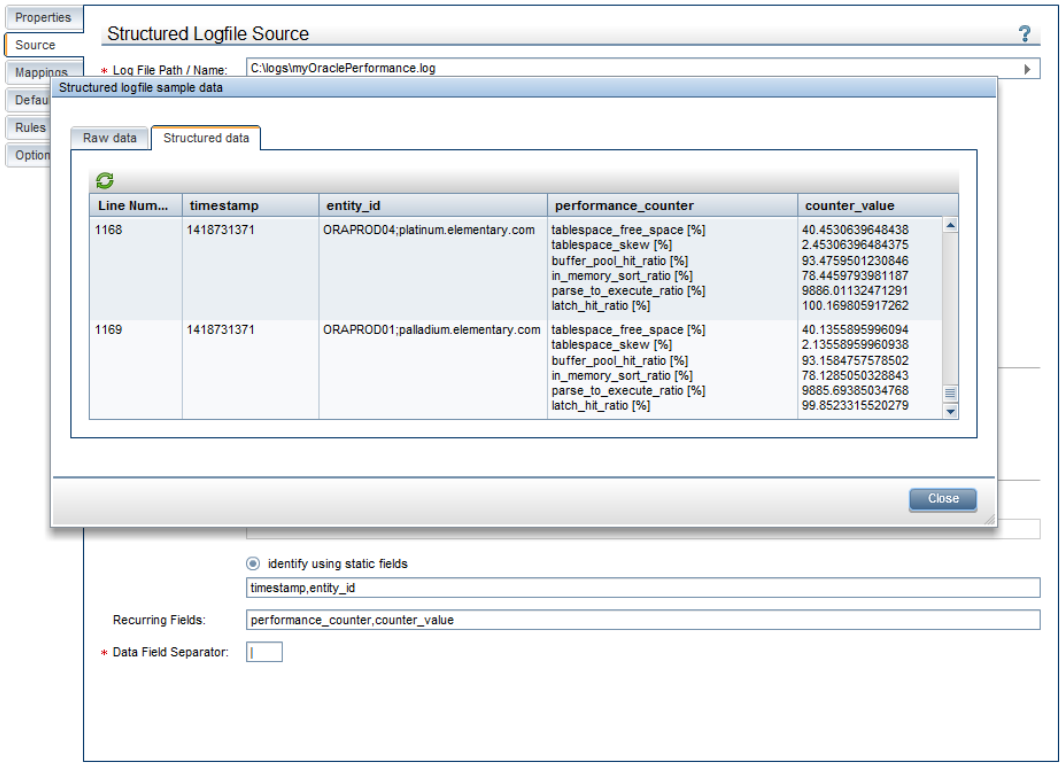
To create a structured log file policy for metrics integration, follow these steps:

Note: For details about structured log file data policies, see "How to Collect Metrics Data from Structured Log Files" on page 235.

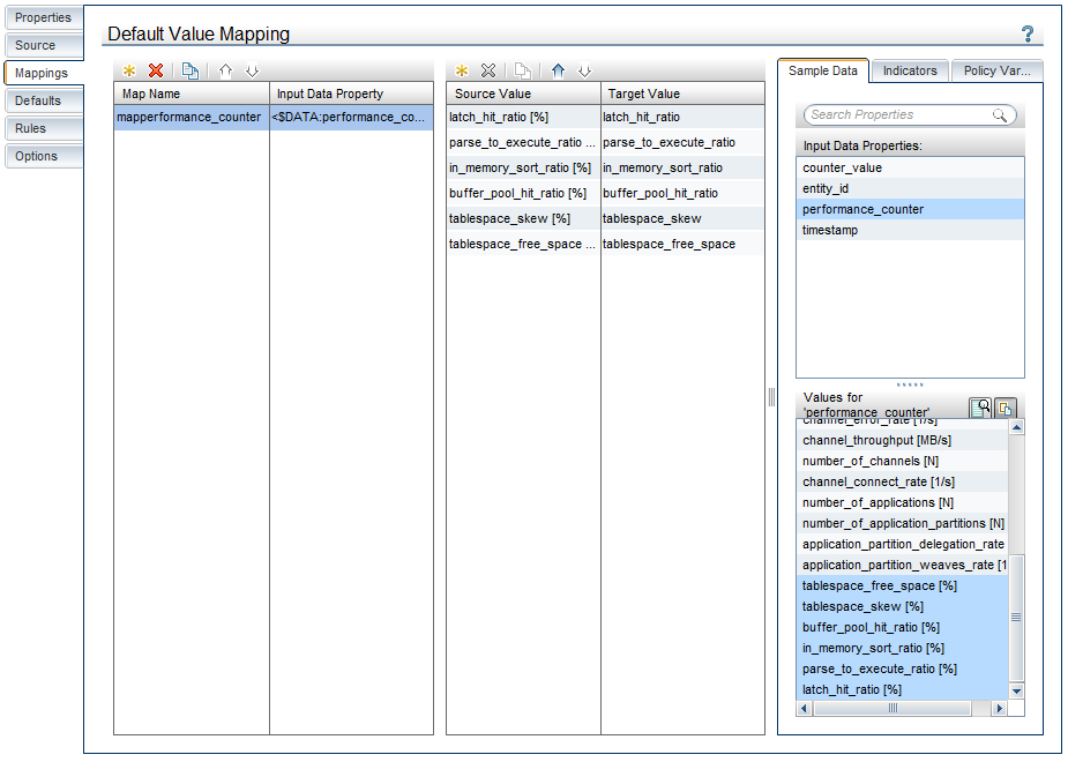
a. In the **Source** page, specify the full path to the log file on the BSM Connector system or a command that returns a path, and also define static and recurring fields according to the logical structure.



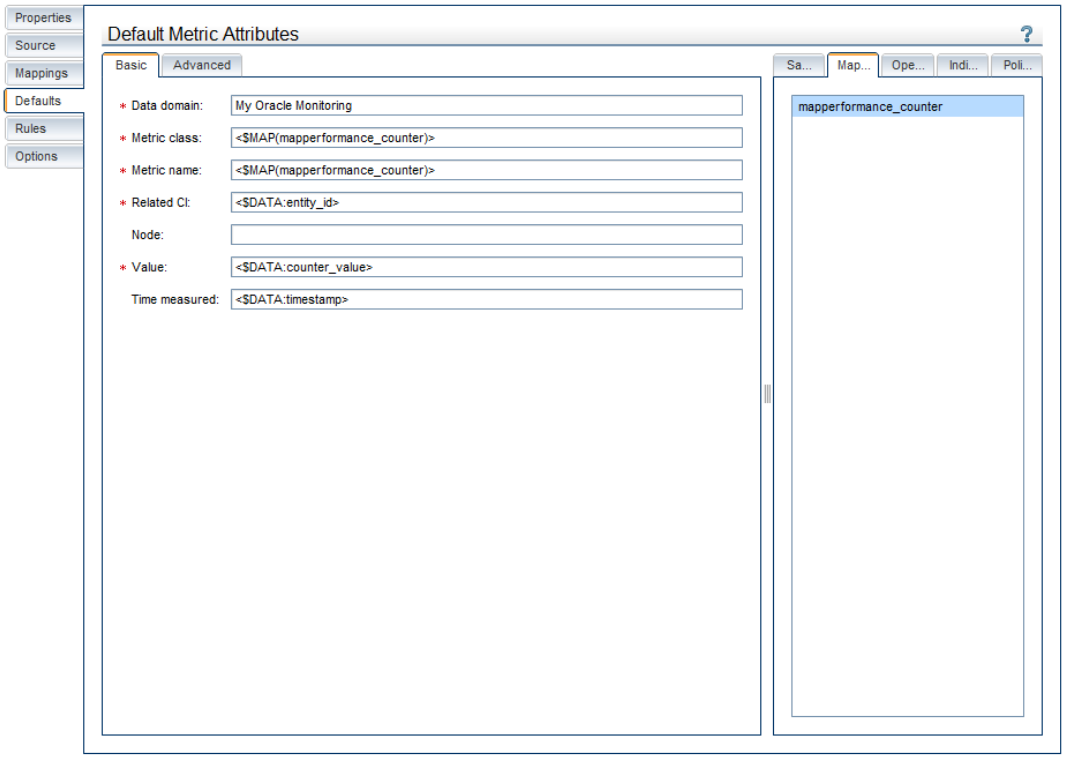
Check results with the view sample data feature ( ) and apply the structure pattern definition by pressing the refresh button ( ). For more information on the structured log file source page, see "Configuring Data Source in Structured Log File Policies" on page 238.



b. Define a mapping for the values of the field `performance_counter` to strip the values of unnecessary information. You can drag the field from the Input Data Properties and drop it on the Default Value Mapping list. For more information on creating the mappings in the structured log file policies, see "Configuring Mappings in Structured Log File Policies (Event and Metrics Only)" on page 242.



c. On the **Defaults** page, define defaults for the BSMC metrics that are created from the policy configuration.
   - Data Domain attribute is set to `My Oracle Monitoring`.
   - Metric class and metric name are both set to the result of the value mapping (*<$MAP (mapperformance_counter)>*) that is created when defining the mapping. Whenever incoming values that are structured by the field definitions are processed by the policy, the actual value is replaced according to the mapping definition.
   - Related CI is set to values of the field `entity_id` directly (*<$DATA:entity_id>*).
   - Node is empty at this stage, it will be set when defining a rule.
   - Value is set directly by using the defined input data field `counter_value` (*<$DATA:counter_value>*).
   - The `timestamp`field values from the input data are in the epoch time format, therefore you can set directly the Time measured field (*<$DATA:timestamp>*).



d. Create a rule to set the Node attribute, as follows:

Note: Because the values of the incoming data field `entity_id` contain both Oracle SID and hostname, it is useful to create a policy rule for extracting the hostname. The rule condition uses OM Pattern Matching to perform the extraction to named OM Pattern Matching variables.

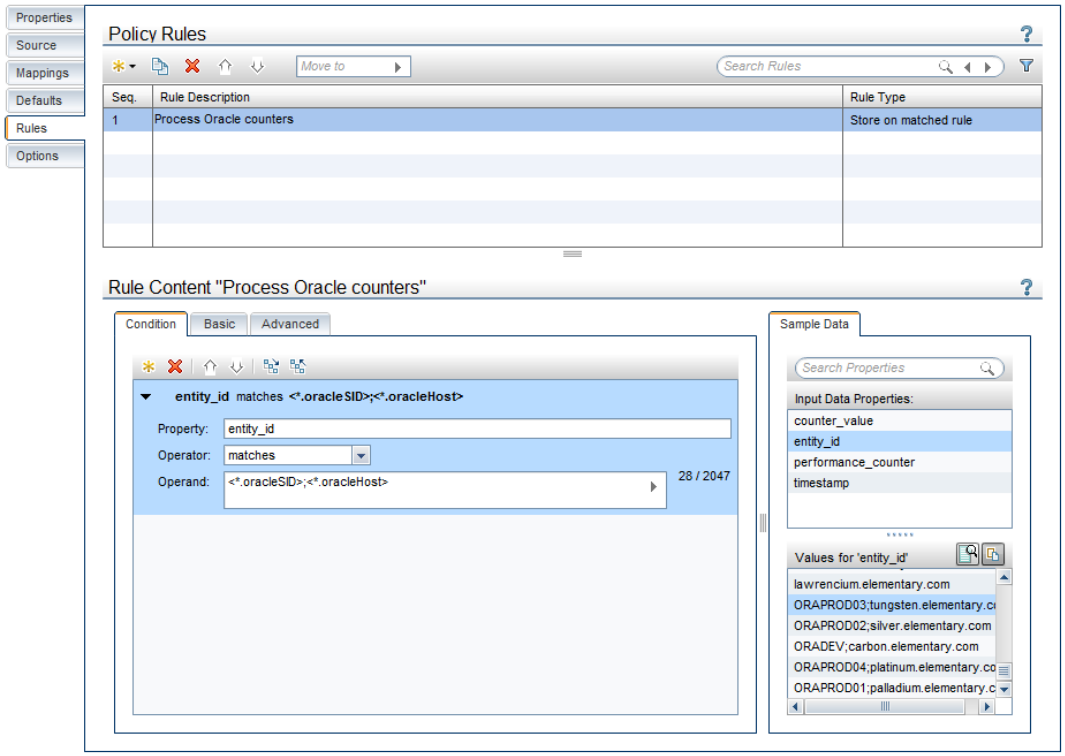
i. In the **Policy Rules** section, click and select the "Store on matched rule" type.

ii. In the **Rule Content** section, click to create a new condition. Click ▸ to expand the new condition. Set the following:

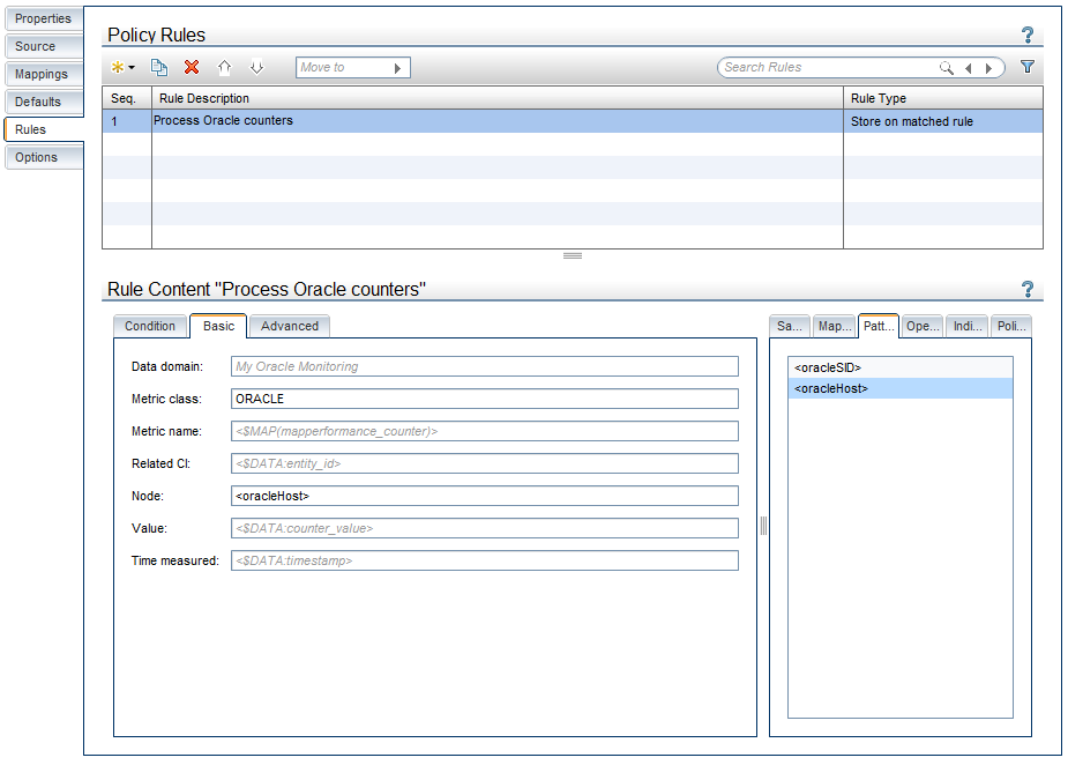
A. In the **Property** field, specify `entity_id`, or drag and drop this input data reference from the Input Data Properties list on the Sample Data tab to the Property field.

B. Select the `matches` operator.

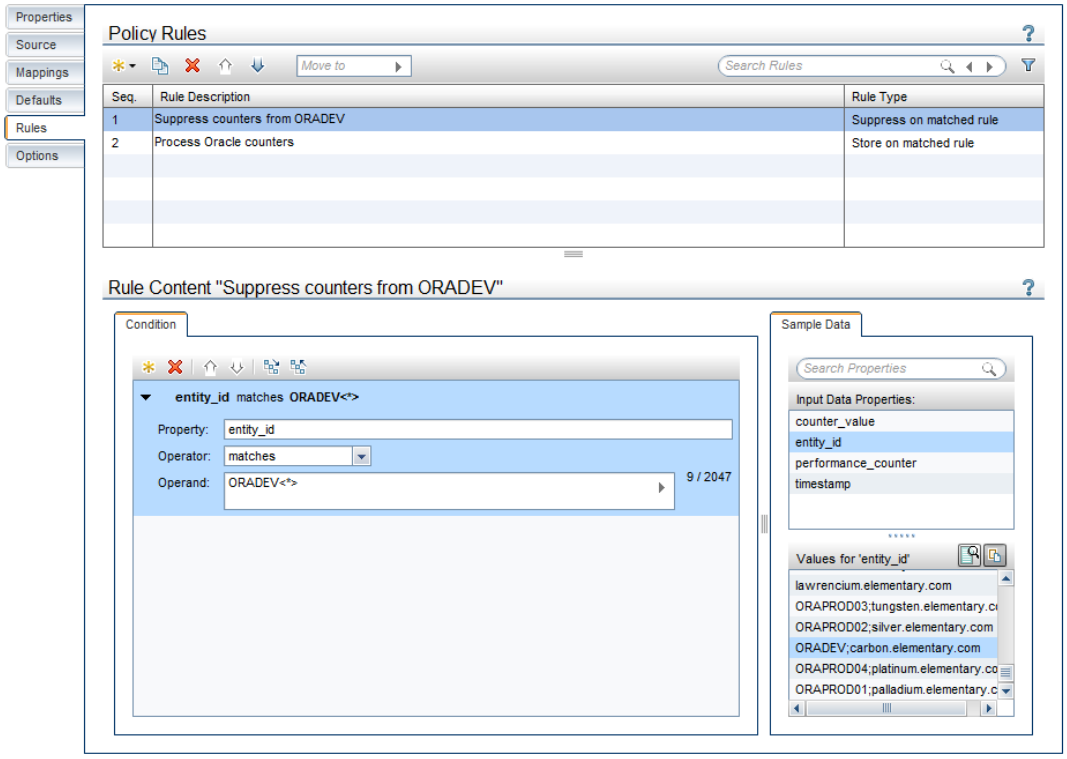
C. In the **Operand** field, type *<*.oracleSID>;<*.oracleHost>*.



To set the Node attribute in this rule, override the default value. It might also be appropriate to set a common Metric class value specific for Oracle related counters in `My Oracle Monitoring`.



e. (*Optionally*) The log file from `My Oracle Monitoring` contains entries for an Oracle development system which should not be integrated at all. This Oracle system is denoted by its SID `ORADEV`. Create a rule of the "Suppress on matched rule" type and place it at the top of the rule list so that the policy skips entries in the log file for that specific instance.



For more information on creating the metric rules in the structured log file policies, see "Configuring Metrics Rules in Structured Log File Policies" on page 254

f. Save and activate the policy.

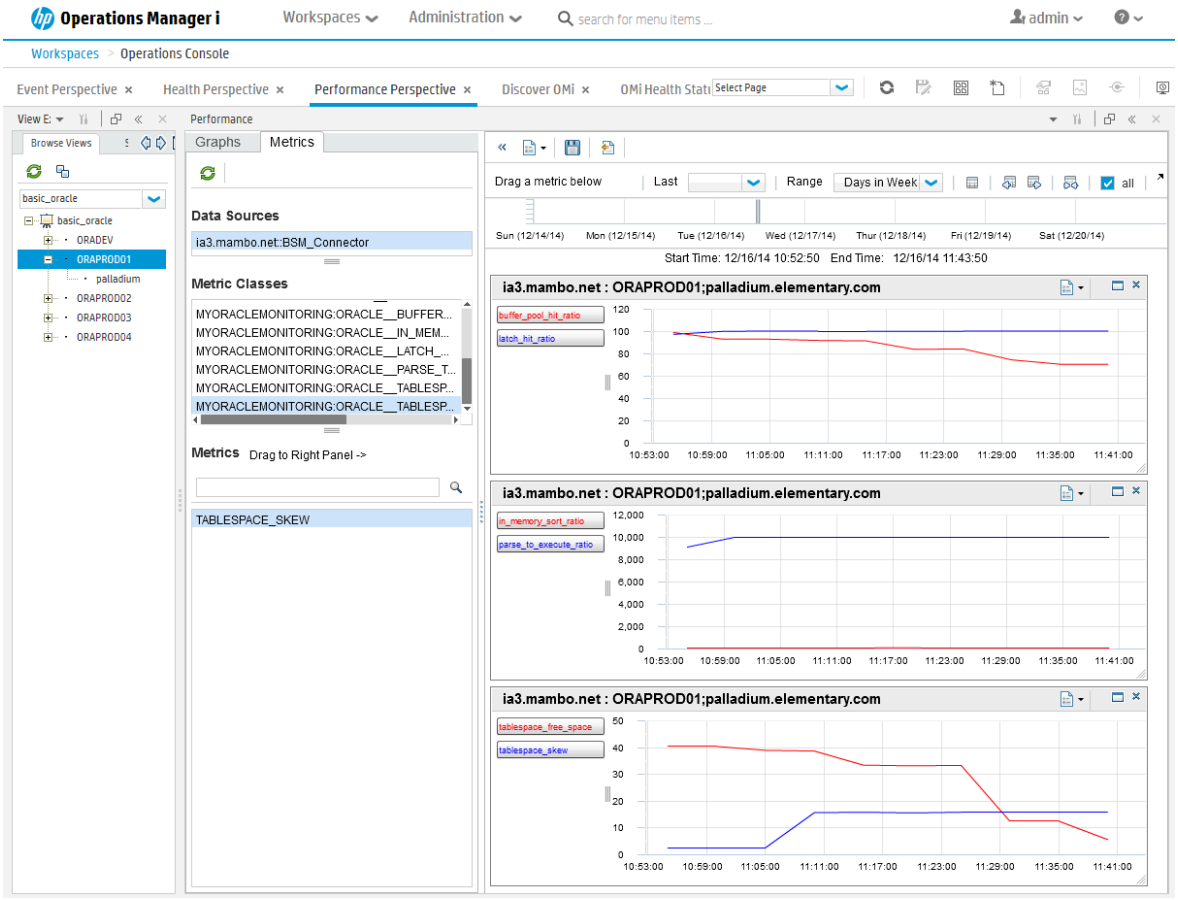
View the integration results

To view the contents of the performance data store in OMi, perform the following:

1. In OMi, select **Workspaces > Operations Console > Performance Perspective**.
2. In the **View Explorer** pane, select the desired view (that includes CI with metrics that you want to analyze) from the Browse Views tab, and then click the Metrics tab in the **Performance** pane.

   Note: If there is no CI with the External Id field that is identical to the Related CI field of the BSM Connector metric policy attribute, Metric Class is mapped to the BSM Connector host.

3. Select the metric for which you want to check the performance and drag it to the subtab on the right to see the graph that shows the values for this metric.



Collecting Metrics with HP Operations Agent

The HP Operations Agent helps you to monitor a system by collecting metrics that indicate the health, performance, availability, and resource utilization of essential elements of the system.

When you use the HP Operations Agent in conjunction with Business Service Management Connector (BSMC), you can add the capability to monitor business applications running on monitored systems.

With its embedded data collector, the HP Operations Agent collects a rich set of system performance metrics from the monitored system to help you analyze the system health. Using minimal system resources, HP Operations Agent continuously collects and summarizes performance and health data across your system and stores the collected data into the metrics data store. The collected data is stored in a single Relational Database Management System (RDBMS) based data store.

Verifying the Status of oacore Process

The **oacore** process collects metrics data and records it in the metric data store. After installing the HP Operations Agent, follow the steps to check the status of the **oacore** process:

1. Log on to the HP Operations Agent node.
2. Run the command:

On Windows x64 system:

`%ovinstalldir%bin\win64\ovc`

On Windows x86 system:

`%ovinstalldir%bin\win32\ovc`

On AIX:

`/usr/lpp/OV/bin/ovc`

On UNIX/Linux/Solaris/HP-UX:

`/opt/OV/bin/ovc`

If the `oacore` process is running, the following message appears:

```
oacore      Operations Agent Core              AGENT,OA     (25357)  Running
```

Starting or Stopping the oacore Process

Follow the steps:

1. Log on to the HP Operations Agent node.
2. Go to the following directory:

   On Windows x64 system:

   `%ovinstalldir%bin\win64\`

   On Windows x86 system:

   `%ovinstalldir%bin\win32\`

   On AIX:

   `/usr/lpp/OV/bin`

   On UNIX/Linux/Solaris/HP-UX:

   `/opt/OV/bin`
3. Run the command:
   - To start the `oacore` process: `ovc -start oacore`
   - To stop the `oacore` process: `ovc -stop oacore`

Verifying Data Logging

Metrics data collected by the `oacore` process is logged in the data store.

Follow these steps:

1. Log on to the HP Operations Agent node.
2. Run the command:

   On Windows x64 system:

   `%ovinstalldir%bin\win64\ovcodautil –obj`

   On Windows x86 system:

```
%ovinstalldir%bin\win32\ovcodautil –obj
```

On AIX:

```
/usr/lpp/OV/bin/ovcodautil –obj
```

On UNIX/Linux/Solaris/HP-UX:

```
/opt/OV/bin/ovcodautil –obj
```

A list of logged metrics appear.

Configuring Data Purging

Data collected by the HP Operations Agent is stored for 172800 seconds (two days) by default. To configure data purging from the command line, follow the steps:

1. Log on to the HP Operations Agent node.
2. Run the command to set the time interval to purge data:

   Note: `AutoPurgeIntervalSecs` defines the schedule at which the purge task would get triggered. By default, the purge task is started at 6 AM.

   On Windows x64 system:

   ```
   %ovinstalldir%bin\win64\ovconfchg -ns oacore -set AutoPurgeIntervalSecs <interval>
   ```

   In this instance, `<interval>` specifies the duration after which data is removed from the metrics data store. `<interval>` is always specified in seconds.

   On Windows x86 system:

   ```
   %ovinstalldir%bin\win32\ovconfchg -ns oacore -set AutoPurgeIntervalSecs <interval>
   ```

   On AIX:

   ```
   /usr/lpp/OV/bin/ovconfchg -ns oacore -set AutoPurgeIntervalSecs <interval>
   ```

   On UNIX/Linux/Solaris/HP-UX:

   ```
   /opt/OV/bin/ovconfchg -ns oacore -set AutoPurgeIntervalSecs <interval>
   ```

3. Run the following command to set the data retention period:

   On Windows x64 system:

   ```
   %ovinstalldir%bin\win64\ovconfchg -ns oacore -set KeepDataForSecs <retention_period>
   ```

   In this instance, *<retention_period>* specifies the duration to store data in the metrics data store. *<retention_period>* is specified in seconds.

   On Windows x86 system:

   ```
   %ovinstalldir%bin\win32\ovconfchg -ns oacore -set KeepDataForSecs <retention_period>
   ```

   On AIX:

   ```
   /usr/lpp/OV/bin/ovconfchg -ns oacore -set KeepDataForSecs <retention_period>
   ```

   On UNIX/Linux/Solaris/HP-UX:

```
/opt/OV/bin/ovconfchg -ns oacore -set KeepDataForSecs <retention_period>
```

4. Run the following command to start agent processes:

   On Windows x64 system:

   ```
   %ovinstalldir%bin\win64\ovc -start oacore
   ```

   On Windows x86 system:

   ```
   %ovinstalldir%bin\win32\ovc -start oacore
   ```

   On AIX:

   ```
   /usr/lpp/OV/bin/ovc -start oacore
   ```

   On UNIX/Linux/Solaris/HP-UX:

   ```
   /opt/OV/bin/ovc -start oacore
   ```

   Note: To see the set data retention period and data purging interval, run the following command:

   ```
   ovconfget oacore
   ```

Chapter 8: Collecting Topology Data

Topology discovery is the process of populating the Run-time Service Model database (RTSM) with CI and CI relationship data. Accurate CI topology information provided by discovery of the system infrastructure is essential for Operations Manager i components, for example, for Topology-based Event Correlation (TBEC) or context-specific tools.

Topology discovery and synchronization enables you to write your own discovery scripts and mapping rules. The discovery scripts discover topology data, the mapping rules map the discovered data to the CI and CI relation model used in the RTSM. The processed and transformed topology data is then stored in the RTSM.

The advantage of using topology discovery and synchronization is that you can separate the discovery technology from the RTSM data model. If the RTMS data model changes, you only need to update the mapping files, not the discovery scripts.

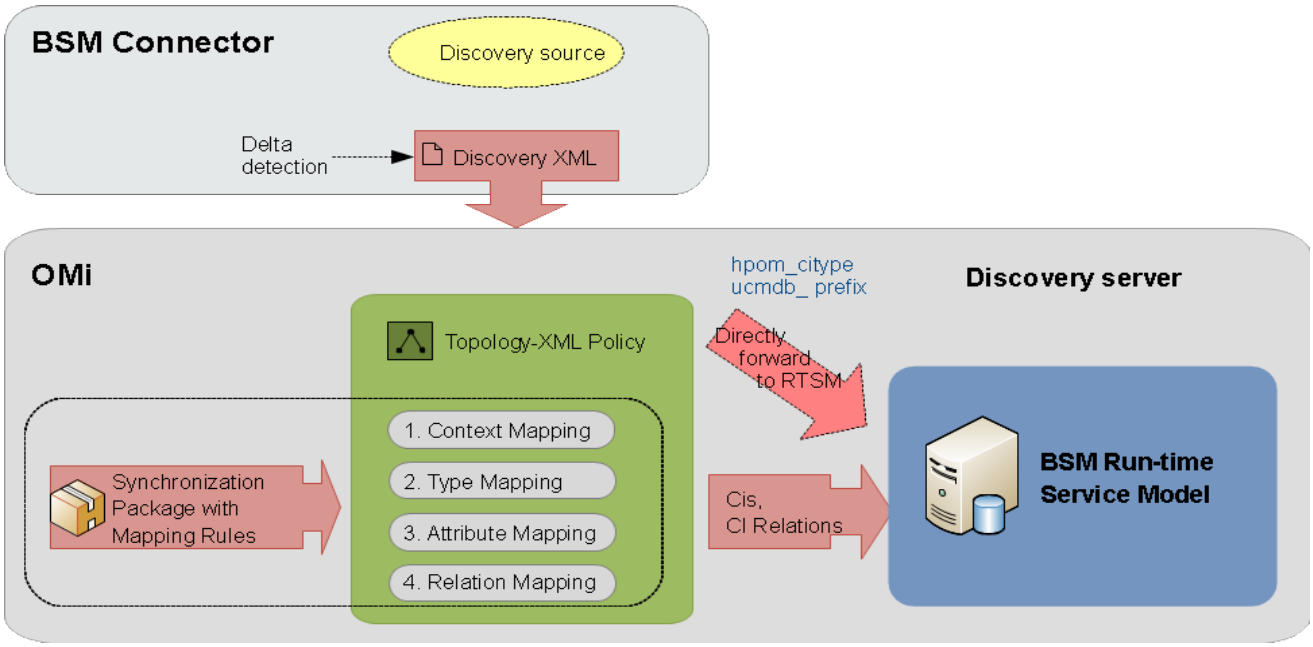
Topology Policies and Topology Synchronization

Topology synchronization enables you to filter the discovered data on the BSM Connector system before sending it to OMi for further processing and mapping on the discovery server.

The discovered data must be available in an XML format that conforms to the BSM Connector topology discovery syntax. To create the XML, write a discovery script or program (using your preferred scripting or programming language) that generates the XML file.

The XML file or REST Web Service topology policy then reads the XML data. Data from the discovery XML can be either *sent to RTSM directly* (if it complies to the RTSM data model and is marked correctly) or *transformed with mapping rules* to CIs and CI relations that comply with the RTSM data model.

To configure the transformation, create a topology synchronization package and include the necessary mapping rules. You then configure the XML or REST Web Service topology policy to apply the synchronization package to the discovery XML file.



Delta detection

A delta file is the difference between the input file and the data in the repository and includes only the minimum of information about the changes: new and changed instances and relations and instances and relations that need to be deleted. Instances are deleted only if they reach their aging limit.

If delta discovery is not enabled, no delta detection is done and the incoming topology file is directly sent to the topology server.

Aging of topology items

If an instance is not reported on several runs in a row,it will age, that is it will be marked with a counter. The counter goes up each time when the algorithm is running (this depends on the interval and the last time the file was written or sent via web service).

Once the *aging limit* (number of runs when it was not found) is reached, the instance will be deleted. You can configure the aging limit in the policy. See "How to configure the XML source (topology)" on page 266.

Send Topology Data Directly to RTSM

For simple topologies, you can directly map data from the discovery XML to RTSM, without the need to provide a sync package.

1. Create a script (or program) that runs on the BSM Connector server and discovers configuration items (CIs) and CI relations. This discovery script must write details of each discovered CI in XML to a file on the BSM Connector system or send it through the REST Web Service. For details on the XML syntax, see "Topology Discovery Syntax" on page 107.

Setting Instance Attributes

To make sure that data is mapped directly to RTSM, you must set the following attributes for the instance you want to map:

- `hpom_citype`

  Sets the CI type as stored in the RTSM.

- `hpom_discoTarget`

  Set this attribute to the value "BSM".

- `ucmdb_<CI_attribute_name>`

  Attribute names with the prefix `ucmdb_` map directly to CI attributes in RTSM. For example, `ucmdb_primary_dns_name` maps to the CI attribute primary_dns_name.

Syntax:

```
<Attributes>
    <attribute name="hpom_citype" value="<CIType>"
    <attribute name="hpom_discoTarget" value="BSM">
    <attribute name="ucmdb_<CI_attribut_name>" value="<CIValue>">
</Attributes>
```

Examples:

In the following example, the CI type set to "host_node" and the attributes "name" and "primary_ dns_name" are mapped to the corresponding CI attributes in RTSM.

```
<NewInstance ref="HOST:server3.company.com">
    <Std>DiscoveredElement</Std>
    <Virtual/>
    <Key>HOST:server3.company.com</Key>
    <Attributes>
        <attribute name="hpom_citype" value="host_node">
        <attribute name="hpom_discoTarget" name="BSM">
        <attribute name="ucmdb_name" value="server3">
        <attribute name="ucmdb_primary_dns_name" value="server3.company.com">
        <attribute name="ucmdb_data_externalid" value="server3.company.com">
    </Attributes>
</NewInstance>
```

In the following example, the CI type set to "sqlserver" and the attributes "name", "database_ dbsid", and "discovered_product_name" are mapped to the corresponding CI attributes in RTSM.

```
<NewInstance ref="SQLSERVER:MSSQLSERVER:server3.company.com">
    <Std>DiscoveredElement</Std>
    <Virtual />
    <Key>SQLSERVER:MSSQLSERVER:server3.company.com</Key>
    <Attributes>
        <Attribute name="hpom_citype" value="sqlserver" />
        <Attribute name="hpom_discoTarget" value="BSM" />
        <Attribute name="ucmdb_name" value="MSSQLSERVER" />
```

```
        <Attribute name="ucmdb_database_dbsid" value="MSSQLSERVER" />
        <Attribute name="ucmdb_discovered_product_name" value="mssqlserver" />
        <Attribute name="ucmdb_data_externalid"
value="server3.company.com/MSSQLSERVER" />
    </Attributes>
</NewInstance>
```

Setting the Relations Type

Make sure the relations type does *not* have the prefix `omrel_`.

For example, the relation type "composition" should be set like:

```
<NewRelationship>
    <Parent>
        <Instance ref="HOST:server3.company.com"/>
    </Parent>
    <GenericRelations>
        <Relations type="composition">
            <Instance ref="SQLSERVER:MSSQLSERVER:server3.company.com"/>
        </Relations>
    </GenericRelations>
</NewRelationship>
```

2. Create a topology XML or Rest Web Service policy. For details on topology policies, see "Configuring Topology Through the REST Web Service Listener" on page 187 and "How to Collect Topology Data from XML Files" on page 261.

Use Topology Synchronization Rules to Map Data into RTSM CIs and Relationships

For complex topologies, you need to create a topology synchronization package that contains mapping rules that map data in to RTSM CIs and relationships.

1. Create a script (or program) that runs on the BSM Connector server and discovers configuration items (CIs) and CI relations. This discovery script must write details of each discovered CI in XML to a file on the BSM Connector system or send it through the REST Web Service. For details on the XML syntax, see "Topology Discovery Syntax" on page 107.
2. Create the sync package:
   a. In a temporary location, create a directory with a name that matches the synchronization package name (for example, `/temp/mysyncpkg`).
   b. Create the package descriptor file and save it in the synchronization package directory (for example, `/temp/mysyncpkg/package.xml`).

      For details on the package descriptor file, see "Package Descriptor File: package.xml" on page 79.
   c. Configure the context mapping file to tag which elements included in discovered data you want to include in the topology synchronization for mapping. The mapping rules contained in

the XML mapping files are applied to the tagged elements.

Save the context mapping file in the synchronization package directory (for example, `/temp/mysyncpkg/contextmapping.xml`).

For details on the context mapping file, see "Context Mapping (Filtering): contextmapping.xml" on the next page.

d. Configure the type mapping file to define the mapping between the discovered CIs to the type of a CI in the RTSM.

Save the type mapping file in the synchronization package directory (for example, `/temp/mysyncpkg/typemapping.xml`).

For details on the type mapping file, see "Type Mapping File: typemapping.xml" on page 80.

e. Configure the attribute mapping file to map the attributes of a discovered CI to the attributes of a CI in the RTSM.

Save the attribute mapping file in the synchronization package directory (for example, `/temp/mysyncpkg/attributemapping.xml`).

For details on the attribute mapping file, see "Attribute Mapping File: attributemapping.xml" on page 81.

f. Configure the relation mapping file to define the CI relationships created in the RTSM between specified discovered CIs.

Save the relation mapping file in the synchronization package directory (for example, `/temp/mysyncpkg/relationmapping.xml`).

For details on the relation mapping file, see "Relation Mapping File: relationmapping.xml" on page 81.

g. *Optional:* Use the supplied XML schema definitions to validate the correctness of the XML mapping files. The definitions are located theOMi server:

`%topaz_home%/conf/opr/topology-sync/schemas`

3. Upload the sync package to the OMi server. See "Uploading the Sync Package to the OMi server" on page 107.
4. Create a topology XML or Rest Web Service policy. For details on topology policies, see "Configuring Topology Through the REST Web Service Listener" on page 187 and "How to Collect Topology Data from XML Files" on page 261.

Topology Synchronization Rules

The filter and mapping rules that process the discovered data are gathered in a synchronization package. You can apply multiple synchronization packages to the topology data discovered by a topology policy.

Topology Synchronization Packages

A synchronization package consists of the following files:

- **package.xml.** The package descriptor file defines the name of the synchronization package and contains a description and the priority level. The priority level determines the order in which synchronization packages are applied.

- **contextmapping.xml.** The context mapping file defines the filter that is applied to the discovered data. Filtering involves assigning a context to those CIs you want to process and send to OMi. Configuring the context enables you to apply mapping rules selectively to CIs of the same context.
- **typemapping.xml.** The type mapping file defines the mapping from the type of a discovered CI to the type of a CI in OMi.
- **attributemapping.xml.** The attribute mapping file defines the mapping between the attributes of a discovered CI and the attributes of a CI in OMi.

  Attribute mapping enables you to change CI attributes and add new attributes to better describe a CI and create a more detailed view of the environment.
- **relationmapping.xml.** Using the relation mapping file, you can define the CI relationships created in OMi between specified discovered CIs.

  Make sure that the specified discovered CIs are created as CIs in the RTSM. Otherwise it is not possible for topology synchronization to create a relationship in the RTSM.

Synchronization packages must be uploaded to the OMi system. See "Uploading the Sync Package to the OMi server" on page 107.

Package Descriptor File: package.xml

A topology synchronization package must include the package descriptor file (`package.xml`). The `package.xml` file defines a topology synchronization package and includes:

- `<Name>` The name of the package must be the same as the name of the subdirectory in which you place the synchronization package:

  `%topaz_home%\conf\opr\topology-sync\sync-packages\`
- `<Description>` Description of the package.
- `<Priority>` Priority level of the package.

  The highest priority is represented by `1`. The default synchronization package is assigned the lowest priority of `10`. A higher priority rule result overwrites a result from a lower priority rule.

  Note: There may be more than one synchronization package with the same priority. The order of execution of the rules between synchronization packages with the same priority is not specified.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Package xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="Schemas/Package.xsd">
   <Name>mysyncpkg</Name>
   <Description>Integrate My Topology in HP OMi</Description>
   <Priority>5</Priority>
</Package>
```

Context Mapping (Filtering): contextmapping.xml

The context mapping file assigns a context to CIs that match specified conditions. BSM Connector synchronizes all the CIs that you map to any context, and ignores all other CIs.

In the following example, the context `myTopology` is assigned to all discovered CIs that have the attribute `myAttribute` with a value of `myAttributeValue`.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="Schemas/Mapping.xsd">
  <Rules>
    <Rule name="Filter my discovered CIs">
      <Condition>
        <Equals>
          <OMAttribute>myAttribute</OMAttribute>
          <Value>myAttributeValue</Value>
        </Equals>
      </Condition>
      <MapTo>
        <Context>myTopology</Context>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

Type Mapping File: typemapping.xml

The type mapping file maps the type of a discovered CI to the type of a CI in OMi.

In the following example, the CIs with the context `myTopology` that have the attribute `myType` with a value of `myTypeValue` are mapped to the CI type `myCIType` in OMi.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="Schemas/Mapping.xsd">
  <Rules Context="myTopology">
    <Rule name="Map myTypeValue to myCIType">
      <Condition>
        <Equals>
          <OMAttribute>myType</OMAttribute>
          <Value>myTypeValue</Value>
        </Equals>
      </Condition>
      <MapTo>
        <CMDBType>
          <Value>myCIType</Value>
        </CMDBType>
      </MapTo>
    </Rule>
```

```
  </Rules>
</Mapping>
```

Attribute Mapping File: attributemapping.xml

The attribute mapping file maps the attributes of a discovered CI to the attributes of a CI in OMi.

In the following example, for the CIs with the context `myTopology` that have the attribute `myAttribute` with a value of `myAttributeValue`, the attribute `title` is mapped to the CI attribute `name` in OMi.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="Schemas/Mapping.xsd">
  <Rules Context="myTopology">
    <Rule name="Map my attributes to RTSM CI attributes">
      <Condition>
        <Equals>
          <OMAttribute>myAttribute</OMAttribute>
          <Value>myAttributeValue</Value>
        </Equals>
      </Condition>
      <MapTo>
        <CMDBAttribute>
          <Name>name</Name>
          <SetValue>
            <OMAttribute>title</OMAttribute>
          </SetValue>
        </CMDBAttribute>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

Relation Mapping File: relationmapping.xml

Using the relation mapping file, you can create CI relationships in OMi between specified discovered CIs.

In the following example, for CIs with the context `myTopology`, the CI `myAttributeValue1` creates a contains relationship to CIs that have the attribute `myAttribute1` with a value of `myAttributeValuew` or `myAttributeValuey`.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="Schemas/Mapping.xsd">
  <Rules Context="myTopology">
```

```
    <Rule name="Create relation from myAttributeValuew to myAttributeValue
(root)">
      <Condition>
        <Equals>
          <OMAttribute>myAttribute1</OMAttribute>
          <Value>myAttributeValuew</Value>
        </Equals>
      </Condition>
      <MapTo>
        <RootContainer>
          <DependentCI relationType="contains">
            <Equals>
              <OMAttribute>myAttribute</OMAttribute>
              <Value>myAttributeValue</Value>
            </Equals>
          </DependentCI>
        </RootContainer>
      </MapTo>
    </Rule>
    <Rule name="Create relation from myAttributeValuey to myAttributeValue
(root)">
      <Condition>
        <Equals>
          <OMAttribute>myAttribute1</OMAttribute>
          <Value>myAttributeValuey</Value>
        </Equals>
      </Condition>
      <MapTo>
        <RootContainer>
          <DependentCI relationType="contains">
            <Equals>
              <OMAttribute>myAttribute1</OMAttribute>
              <Value>myAttributeValue1</Value>
            </Equals>
          </DependentCI>
        </RootContainer>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

Mapping Syntax

Mapping is the mechanism used to map CIs, CI attributes, and CI relationships within the topology model of a third-party system to CIs and CI relationships in the RTSM. The file format, mapping syntax, and XPath query language used to navigate through the CI data structure is described in the following sections:

- "Common Mapping File Format" below
- "Mapping File Syntax" on the next page
- "XPath Navigation" on page 103

Common Mapping File Format

Note: The rule name must be unique for all rules in the current file.

This example illustrates the common parts of the mapping file:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping>
    <Rules Context="web server SPI">
        <Rule name="Apache Server">
            <Condition>
                <!-- ... Boolean operators ... -->
            </Condition>
            <MapTo>
                <!-- ... Target Mappings ... -->
            </MapTo>
        </Rule>
        <!-- ... More Rules ... -->
    </Rules>
    <!-- ... More Rule sets with different contexts ... -->
</Mapping>
```

The components of the mapping files are described in "Mapping File Syntax" on the next page.

Mapping File Syntax

The following sections describe the valid syntax used in topology synchronization mapping files.

- "Rules" below
- "Rule Conditions" below
- "Operator Elements" on page 86
- "Operand Elements" on page 89
- "Mapping Elements" on page 96
- "Filtering" on page 97
- "Type Mapping" on page 98
- "Attribute Mapping" on page 99
- "Relation Mapping" on page 101

Rules

The `<Rules>` tag contains a set of rules. By using the optional `Context` attribute you can restrict these rules to a certain context. See "Filtering" on page 97 for more information.

Rule Conditions

The `<Condition>` element of a rule contains a Boolean operator that specifies how the individual conditions relate to each other, and, for example, is similar to the `<condition>` task in Ant.

Each operator can implement an operation against operands. For example, attribute `hosted_on` has a value ending with `.europe.example.com` (attribute `hosted_on` and `.europe.example.com` are operands) or an operation against one or a set of other nested operators like `<And>`, `<Or>` or `<Not>`.

Condition Examples

Check if the type of the current discovered object is `testtype`.

Example:

```
<Condition>
    <Equals>
        <OMType/>
        <Value>testtype</Value>
    </Equals>
</Condition>
```

Check if the CI is related to a node that is located in the `europe.example.com` domain.

Example:

```
<Condition>
    <EndsWith>
        <XPathResult>//.[node='true']/attributes/
            host_dnsName<XPathResult>
        <Value>.europe.example.com</Value>
    </EndsWith>
</Condition>
```

Operator Elements

True

```
<True/>
```

This operator always returns true when all nested operators return true. It is useful for declaring default (fall-back) rules. In a mapping engine that is using the early-out mode, make sure that this operator is only used at the end of the synchronization package with the lowest priority.

False

```
<False/>
```

Always returns false. You can use the `False` element to temporarily disable rules.

And

```
<And>
    <!-- Operator -->
    <!-- Operator -->
    [... more operators ...]
</And>
```

Returns true when all nested operators return true.

The `<And>` operator is exclusive. This means that if the result of the first operator is false, the next operator is not evaluated. Use this operator to implement rules with higher performance by placing the simplest condition first and the most complex condition at the end.

Or

```
<Or>
    <!-- Operator -->
    <!-- Operator -->
    [... more operators ...]
</Or>
```

Returns true if at least one of the operators returns true.

Not

```
<Not>
    <!-- Operator -->
</Not>
```

Returns true if the operator does not return true.

The `<Not>` operator is exclusive. This means that evaluation stops as soon as a child operator returns true.

Exists

```
<Exists>
    <!-- Operand -->
<Exists>
```

The value of the operand must not be null.

Is Node

```
<IsNode/>
```

True if the CI is imported as a node, which is the case if the CI type is listed in the `nodetypes.xml` file.

True if the element is a managed node in HPOM.

Is Root CI

```
<IsRootCI/>
```

True if the CI is a root CI (a root CI has no parent).

Equals

```
<Equals>
    <!-- Operand -->
    <!-- Operand -->
    <!-- ... -->
</Equals>
```

```
<Equals ignoreCase="[true|false]">
    <!-- Operand -->
    <!-- Operand -->
    <!-- ... -->
</Equals>
```

The values of the operands must be equal. If there are more than two operands, all operands must be equal to each other. Using the optional attribute `ignoreCase`, you can also compare the string values of the operands independent of capitalization. By default the equals operator does not ignore case.

Starts With

```
<StartsWith>
    <!-- Operand -->
    <!-- Operand -->
</StartsWith>
```

The string value of the first operand must start with the value of the second operand.

Ends With

```
<EndsWith>
```

```
    <!-- Operand -->
    <!-- Operand -->
</EndsWith>
```

The string value of the first operand must end with the value of the second operand.

Matches

```
<Matches>
    <!-- Operand -->
    <!-- Operand -->
</Matches>
```

The string value of the first operand must match the regular expression of the second operand.

Example:

```
<Matches>
    <Attribute>host_dnsname</Attribute>
    <Value>.*\.example\.com</Value>
</Matches>
```

For more information on applicable regular expressions, see:

http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

Contains

```
<Contains>
    <!-- Operand -->
    <!-- Operand -->
<Contains>
```

The value returned by the first operand must contain the value of the second operand. If the operand's return type is a list, the list must contain at least one element that is equal to the second operand. If the operand's return type is a string, the value of the second operand must be a substring of the first operand.

Is Deletion CI

```
<IsDeletionCI/>
```

True if the CI is used to delete CIs.

Operand Elements

HPOM Service ID

```
<OMId/>
```

Return type: String

Returns the HPOM ID string of the CI as stored in OMi. The HPOM ID returns different values as follows:

Services: HPOM ID is the service ID

Nodes: HPOM ID is the unique ID

Node Groups: HPOM ID is the node group ID

HPOM Type

```
<OMType/>
```

Return type: String

Returns the HPOM Type stored in OMi. For HPOM services, the HPOM Type is the service type definition. For nodes, the HPOM Type is set to the constant value “node”.

CMDB Type

```
<CMDBType/>
```

Return type: String

Returns the CMDB CI Type ID string of the CI as it is stored in the RTSM. Initially this is returned as null because the CMDB type must initially be set in the Type Mapping. After this is set, the CMDB CI Type ID string should be available.

Caption

```
<Caption/>
```

Return type: String

Returns the caption string of the CI in the RTSM or OMi.

HPOM Attribute

```
<OMAttribute>[Name]</OMAttribute>
```

Return type: String

Returns the value of the HPOM attribute with the given name.

CMDB Attribute

```
<CMDBAttribute>[Name]</CMDBAttribute>
```

Return type: String

Returns the value of the CMDB attribute with the given name as it will be written to the RTSM. There are no attributes available until the attribute mapping has been performed.

Replace

```
<Replace [regExp="true|false"]>
    <In>
        <!-- 1st. Operand -->
    </In>
    <For>
        <!-- 2nd. Operand -->
    </For>
    <By>
        <!-- 3rd. Operand -->
    </By>
</Replace>
```

Return type: String

Replaces the strings in the return value of the first operand for all occurrences of the return value of the second operator by the return value of the third operand. For example, to replace all occurrences of a backslash in the CI caption by an underscore, you must declare the following:

```
<Replace>
    <In>
        <CiCaption/>
    </In>
    <For>
        <Value>\</Value>
    </For>
    <By>
        <Value>_</Value>
    </By>
</Replace>
```

Optionally, you can use regular expressions for the second operand. You can also use back references in the third operand.

For more information on applicable regular expressions, see:

http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

This example uses regular expressions to extract part of a domain name:

```
<Replace regExp="true">
    <In>
        <Attribute>host_dnsname</Attribute>
    </In>
    <For>
        <Value>^[^.]*\.([^.]*).*</Value>
```

```
    </For>
    <By>
        <Value>$1</Value>
    </By>
</Replace>
```

If the attribute `host_dnsname` contains the value `server.rio.example.com`, the result of the `Replace` operand is `rio`.

XPath Result

```
<XPathResult>[XPath]</XPathResult>
```

Return type: String

Returns the value of the XPath expression, which enables you to access data of any CI that is contained in the same hierarchy. The XPath expression must select a string value, if there are multiple matches an arbitrary element is returned.

For more information on XPath, see "XPath Navigation" on page 103.

XPath Result List

```
<XPathResultList>[XPath]</XPathResultList>
```

Return type: List

Returns a list of all matched values.

For more information on XPath, see "XPath Navigation" on page 103.

Value

```
<Value>[String]</Value>
```

Return type: String

Return the constant value.

List

```
<List>
    <!--Operand-->
    <!--Operand-->
    <!--...-->
</List>
```

Return type: List

The list operand is designed for use with operators that accept lists as input parameters, such as the `contains` operator. The list operand contains a list of other operands, the values of which are to be added to the returned list.

Parent CI

```
<ParentCI/>
```

Return type: CI

Returns the parent CI of the current CI. If the current CI is the root CI, null is returned.

Tip: To check for the root CI, use the `IsRoot` operator.

Child CI

```
<ChildCI>
    [Operator]
</ChildCI>
```

```
<ChildCI relationType="[relationType]">
    [Operator]
</ChildCI>
```

Return type: CI

Description: Returns the first child CI of the current CI that matches the enclosed operator.

Optional elements:

`relationType`: Only follow relations with the specified relation type.

Child CI List

```
<ChildCIList>
     [Operator (Optional)]
</ChildCIList>
```

```
<ChildCIList relationType="[relationType]">
     [Operator (Optional)]
</ChildCIList>
```

Return type: List of CIs

Returns all CI children of the current CI.

Optional elements:

`Operator`: Only CIs that match the operator will be returned.

`relationType`: Only follow relations with the specified relation type.

Ancestor CI

```
<AncestorCI>
    [Operator]
</AncestorCI>
```

```
<AncestorCI relationType="[relationType]">
    [Operator]
</AncestorCI>
```

Return type: CI

Returns the first ancestor CI of the current CI that matches the enclosed operator. An ancestor CI is the parent or parent of the parent (and so on) of the current CI.

Optional elements:

`relationType`: The dependency must have the specified relation type.

Descendant CI

```
<DescendantCI>
    [Operator]
</DescendantCI>
```

```
<DescendantCI relationType="[relationType]">
    [Operator]
</DescendantCI>
```

Return type: CI

Returns the first descendant CI of the current CI that matches the enclosed operator. A descendant CI is the child or child of the child (and so on) of the current CI.

Optional elements:

`relationType`: Only follow relations with the specified relation type.

Descendant CI List

```
<DescendantCIList>
    [Operator (Optional)]
</DescendantCIList>
```

```
<DescendantCIList relationType="[relationType]">
    [Operator (Optional)]
</DescendantCIList>
```

Return type: List of CIs

Returns the all descendant CIs of the current CI. A descendant CI is the child or child of the child (and so on) of the current CI.

Optional elements:

`Operator`: Only CIs that match the operator will be returned.

`relationType`: Only follow relations with the specified relation type.

Dependency CI

```
<DependencyCI>
    [Operator]
</DependencyCI>
```

```
<DependencyCI relationType="[relationType]">
    [Operator]
</DependencyCI>
```

Return type: CI

Returns the first dependency CI that matched the included operator.

Optional elements:

`relationType`: Only follow relations with the specified relation type.

Dependency CI List

```
<DependencyCIList>
    [Operator (Optional)]
</DependencyCIList>
```

```
<DependencyCIList relationType="[relationType]">
    [Operator (Optional)]
</DependencyCIList>
```

Return type: CI

Returns the list of dependencies.

Optional elements:

`Operator`: Only CIs that match the operator will be returned.

`relationType`: The dependency must have the specified relation type.

Dependent CI

```
<DependentCI>
    [Operator]
</DependentCI>
```

```
<DependentCI relationType="[relationType]">
    [Operator]
</DependentCI>
```

Return type: CI

Returns the first dependent CI that matched the included operator.

Example for a dependent CI:

ServiceA > hosted_on > HostB

In this case ServiceA is a dependent CI of HostB. That means if you have HostB and want to have all services that depend on this host, you have to use the `<DependentCI>` operand. If you have ServiceA and want to have HostB, you have to use the `<DependencyCI>` operand instead.

Optional elements:

`relationType`: Only follow relations with the specified relation type.

Dependent CI List

```
<DependentCIList>
    [Operator (Optional)]
</DependentCIList>
```

```
<DependentCIList relationType="[relationType]">
    [Operator (Optional)]
</DependentCIList>
```

Return type: CI

Returns the list of dependent CI.

Example for a dependent CI:

ServiceA > hosted_on > HostB

In this case ServiceA is a dependent CI of HostB. That means if you have HostB and want to have all services that depend on this host, you have to use the `<DependentCI>` operand. If you have ServiceA and want to have HostB, you have to use the `<DependencyCI>` operand instead.

Optional elements:

Operator: Only CIs that match the operator will be returned.

relationType: The dependency must have the specified relation type.

From CI Get

```
<From>
    <CI>
        [CI Operand]
    </CI>
    <Get>
        [Operand]
    </Get>
</From>
```

Return type: Return type of the second operand.

Using this operand you can get values from another CI. The first operand [CI Operand] must return a CI instance. The second operand operates on that CI instance and the value of this second operand will be returned by this From operand.

Example:

```
<From>
    <CI>
        <ParentCI>
    </CI>
    <Get>
        <Caption/>
    </Get>
</From>
```

Returns the caption from the parent CI of the current CI.

Origin Server

```
<OriginServer/>
```

Return type: String

This operand returns the hostname of the server that originally received the discovery data before forwarding it to other servers.

Mapping Elements

`<MapTo>` defines the mappings. Each concrete implementation of an engine adds its own XML elements for its individual mappings here.

Mapping Examples

Map the attribute `Caption` to the CI attribute `display_label` in the RTSM.

Example:

```
<MapTo>
   <CMDBAttribute>
      <Name>display_label</Name>
      <SetValue>
         <Caption />
      </SetValue>
   </CMDBAttribute>
</MapTo>
```

Map the `OMID` to the CI attribute `data_name` in the RTSM.

Example:

```
<MapTo>
   <CMDBAttribute>
      <Name>data_name</Name>
      <SetValue>
         <OMId />
      </SetValue>
   </CMDBAttribute>
</MapTo>
```

Filtering

Filtering allows to select interesting parts of topology data by assigning a context to these CIs. This context allows to selectively apply mapping rules to CIs of the same context. All CIs that have no context attached will not be synchronized.

Note: The `<Rules>` tag for filter mapping rules *must not* contain the Context attribute.

Context Mapping

```
<Context>[Context Name]</Context>
```

In the following example, all CIs that are assigned to the type `exch_spi_std_server_role` and that are below a CI with a type `exch_spi_std_server` are assigned to the `exchange` context.

Example:

```
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../mapping.xsd">
  <Rules>
    <Rule name="Exchange Server Role Filter">
      <Condition>
       <And>
        <Exists>
         <XPathResult>ancestor::ci[omType='exch_spi_std_server']
         </XPathResult>
        </Exists>
         <Equals>
           <OMType/>
           <Value>exch_spi_std_server_role</Value>
         </Equals>
       </And>
      </Condition>
      <MapTo>
       <Context>exchange</Context>
      </MapTo>
```

```
    </Rule>
  </Rules>
</Mapping>
```

Type Mapping

The type mapping maps the third-party CI type definitions to their CMDB types in the RTSM.

Mapping

Maps the CI to the specified CMDB type that is the concatenated string of the results of the operators. There must not be more than one `<CMDBType>` element in the `<MapTo>` section.

```
<CMDBType>
    [Operand]
    ...
</CMDBType>
```

In the following example, all CIs that have an OM Type `Exch2k7_ByServer` and that have the context `exchange` assigned are mapped to the CMDB Type `exchangeserver`.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping>

  <Rules context="exchange">
    </Rule>
      <Rule name="Map Exchange Server">
      <Condition>
        <Equals>
          <OMType/>
          <Value>Exch2k7_ByServer</Value>
        </Equals>
      </Condition>
      <MapTo>
        <CMDBType>
          <Value>exchangeserver</Value>
        </CMDBType>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

In the following example, all nodes that have an attribute `OSType` that starts with the string `Windows` are mapped to the CMDB type `nt`.

Example:

```
<Mapping>
   <Rules>
    <Rule name="Map Windows Host Type">
      <Condition>
        <And>
          <IsNode/>
          <StartsWith>
            <OMAttribute>OSType</OMAttribute>
            <Value>Windows</Value>
          </StartsWith>
        </And>
      </Condition>
      <MapTo>
        <CMDBType>
          <Value>nt</Value>
        </CMDBType>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

Attribute Mapping

The attribute mapping file `attributemapping.xml` defines the mapping between the attributes of a discovered object and the attributes of a CI in the RTSM.

Set the value of the attribute of the given name to the returned value of the given operand. If more than one operand is given, the values will be concatenated.

Mapping to String Values

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetValue>
        [Operands]
    </SetValue>
</CMDBCMDBOMAttribute>
```

Mapping to String Values of a Maximum Length

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetValue Length="[IntegerValue]">
        [Operands]
    </SetValue>
</CMDBOMAttribute>
```

Mapping to Integer Values

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetIntValue>
        [Operands]
    </SetIntValue>
</CMDBOMAttribute>
```

Mapping to Boolean Values

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetBoolValue>
        [Operands]
    </SetBoolValue>
</CMDBOMAttribute>
```

Mapping to Long Values

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetLongValue>
        [Operands]
    </SetLongValue>
</CMDBOMAttribute>
```

Mapping to Date Values

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetdateValue>
        [Operands]
    </SetdateValue>
</CMDBOMAttribute>
```

Mapping to Float Values

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetFloatValue>
        [Operands]
    </SetFloatValue>
</CMDBOMAttribute>
```

Mapping to Byte Values

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetByteValue>
        [Operands]
    </SetByteValue>
</CMDBOMAttribute>
```

Mapping to Double Values

```
<CMDBOMAttribute>
    <Name>[Attribute Name]</Name>
    <SetDoubleValue>
        [Operands]
    </SetDoubleValue>
</CMDBOMAttribute>
```

Mapping to StringList Values

```
<CMDBAttribute>
    <Name>[Attribute Name]</Name>
    <SetStringListValue>
        [Operands] (comma-separated)
    </SetStringListValue>
</CMDBAttribute>
```

Mapping to IntList Values

```
<CMDBAttribute>
    <Name>[Attribute Name]</Name>
    <SetIntListValue>
        [Operands] (comma-separated)
    </SetIntListValue>
</CMDBAttribute>
```

Relation Mapping

Using the relation mapping you can create relations between CIs.

```
<RelationTo>
  <To>
     [Operand]
  </To>
  <Type>[RelationType]</Type>
</RelationTo>
```

Define a relation from the current CI to the CI that is returned by the operand. The operand may either return a string, an instance of a CI, a list of CIs or a list of strings. String values must match the OM ID of the CI to which the relation is created. In the case of a list, a relation is created for each item (that is in turn either a string or a CI) contained in the list.

The relation has the type specified by [`RelationType`]. This type is the name of the relation, not the label.

```
<RelationFrom>
  <From>
     [Operand]
  </From>
  <Type>[RelationType]</Type>
</RelationFrom>
```

Works just as the previous mapping, but in the opposite direction.

Root Container Mapping

The CMDB model defines certain root container CIs that have to be created before the actual CI can be created. Topology synchronization must know such relations to be able to create the CIs in the correct order.

```
<RootContainer>
  [Operand]
</RootContainer>
```

The root container of the current CI is set to the CI specified by the return value of the operand. The return value may either be a String or a CI.

Message Alias Mapping for CI Resolution

```
<RedirectMessagesOf>
  [Operand]
</RedirectMessagesOf>
```

The aliases of the current CI is set to the OMId(s) specified by the return value of the operand. The return value may either be a String or a CI or a list of CIs or Strings.

In the following example, the CI with the type `Exch2k7_ByServer` gets a relation of the type `is_impacted_from` to the node on which it is hosted and a relation of the type deployed to all descendant CIs with a type that starts with `Exch2k7_Role_`.

The same node is also the root container CI.

Example:

```
<Mapping>
  <Rules Context="exchange">
    <Rule name="Create relation server to node">
      <Condition>
        <Equals>
          <OMType/>
```

```
          <Value>Exch2k7_ByServer</Value>
        </Equals>
      </Condition>
      <MapTo>
        <RelationTo>
          <To>
            <DependencyCI relationType="hosted_on">
               <True/>
            </DependencyCI>
          </To>
          <Type>is_impacted_from</Type>
        </RelationTo>
        <RelationTo>
          <To>
            <DescendantCIList>
               <StartsWith>
                  <OMType>
                  <Value>Exch2k7_Role_</Value>
               </StartsWith>
            </DependencyCI>
          </To>
          <Type>deployed</Type>
        </RelationTo>
        <RootContainer>
          <DependencyCI relationType="hosted_on">
             <True/>
          </DependencyCI>
        </RootContainer>
        <RedirectMessagesOf>
           <ChildCIList/>
        </RedirectMessagesOf>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

XPath Navigation

XPath is a syntax for defining parts of an XML document. XPath uses path expressions to navigate in XML documents.

XPath is used in the mapping engines to navigate through the CI data structure.

If you are not familiar with the XPath query language, an XPath tutorial can be found at the following Web site:

http://www.w3schools.com/xpath/

Data Structure

The data structure that is exposed to the XPath expression matching used in mapping rules is shown in "Example of an XPath-Navigated Data Structure" on page 106.

CI Data Structure

- OMAttributes

  Contains a map of all original RTSM CI attributes. The key of this map is the name of the RTSM CI attribute that references the RTSM value of the RTSM CI attribute.

- Caption

  Represents the name of the CI to be displayed in Service Navigator. Caption has the same value as the RTSM CI attribute `display_label`.

- Children

  References a list of relations to CIs that have a containment relationship from the current CI to other CIs. Using this field, you can create complex XPath queries to retrieve values of children as well as parents using the "`..`" XPath selector.

- Dependencies

  References a list of relations to dependent CIs. Similar to `Children`. However, referenced objects are contained in a different hierarchy.

- OMid

  Unique ID of a CI.

- Node

  Boolean value that indicates whether this is a node in HPOM.

- Type

  Contains the service type of an HPOM service.

  This is not the display label of the CI Type.

- Service

  Boolean value that indicates whether this element is a service in HPOM.

  Node groups have Node and Service set to FALSE.

Relationship Data Structure

- CI

  Contains the reference to the CI to which the current CI is related.

- RelationType

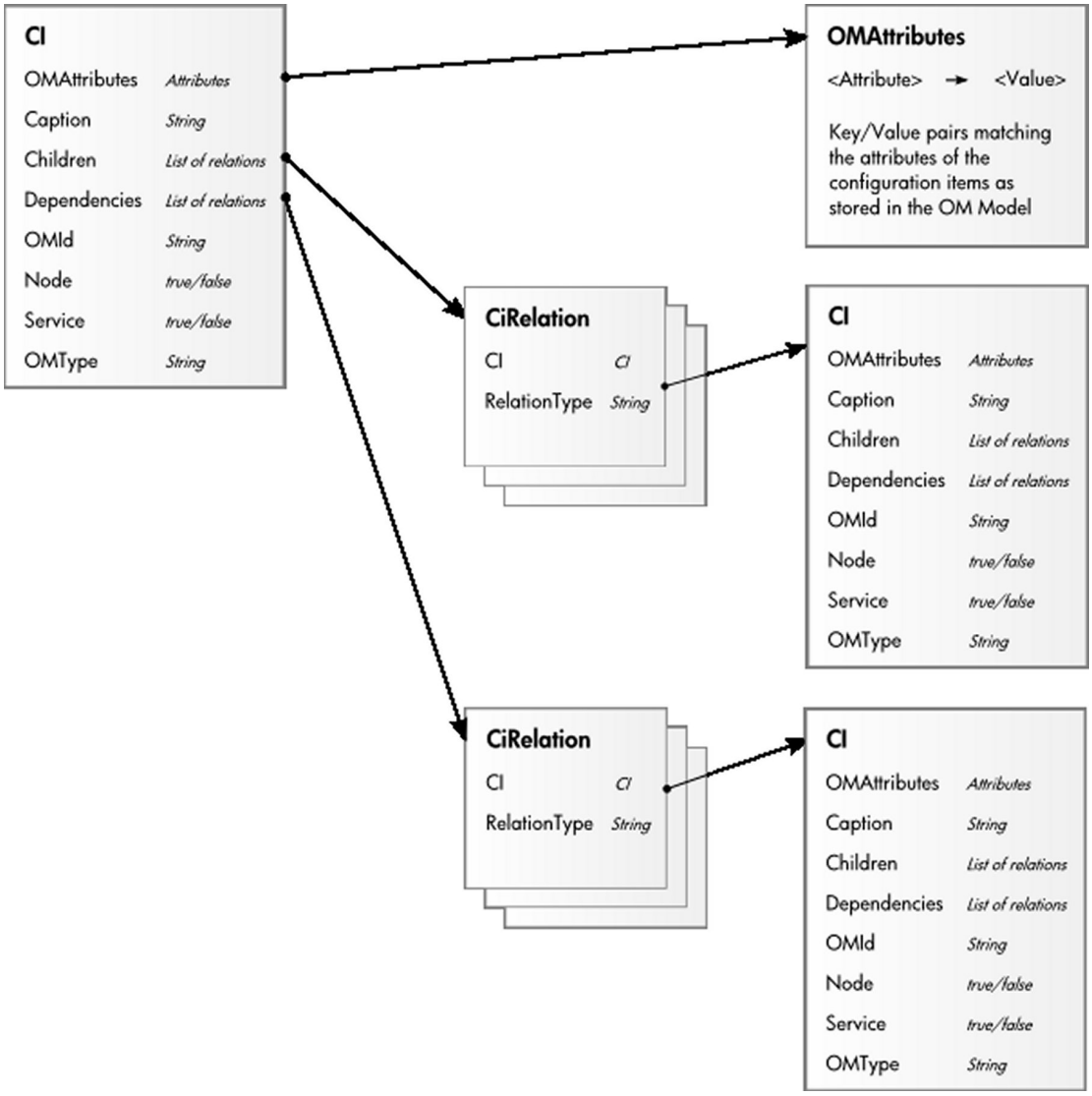
  Relationship type as stored in the RTSM.

  This is not the display label of the CI Type.

  For services:
  Contains `container_f` if it is a containment relation in HPOM.
  Contains `dependency` if it is a dependency relation in HPOM.
  Contains `hosted_on` if it is a hosted on relation in HPOM.

  For nodes:
  Contains `container_node` or `dependency_node`.

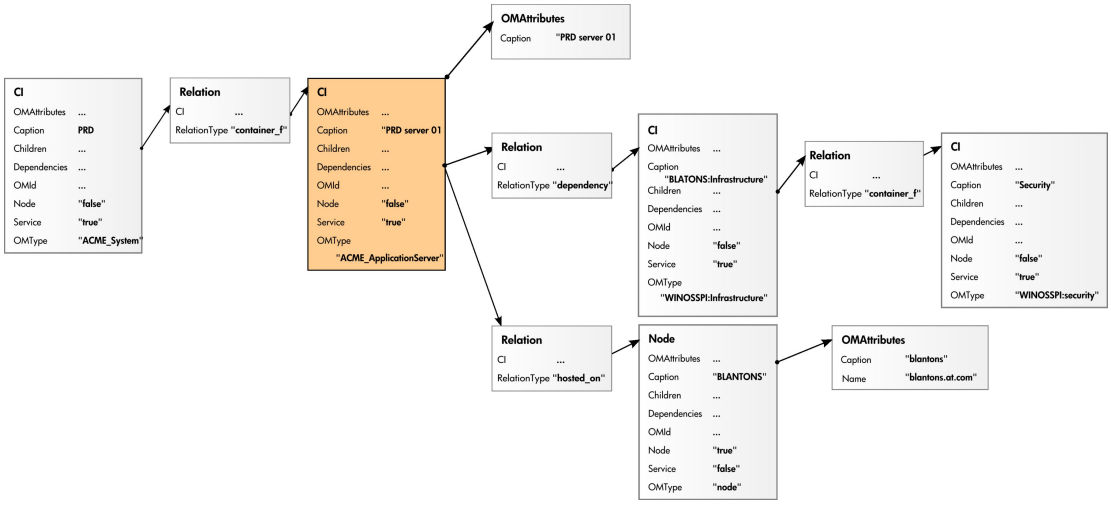
The following figure illustrates the data structure exposed to the navigation.



Example of an XPath-Navigated Data Structure

An example of an XPath-navigated data structure is shown in the figure below. The host is a UNIX system that has an Oracle application running on the HP-UX operating system. The starting point or context for the navigation is the CI that represents the Oracle application (orange background).

The following figure illustrates some XPath examples.



XPath Expressions and Example Values

The following table lists typical XPath expressions and provides an example for each expression.

XPath Expression	Meaning	Example
`Caption`	Caption from CI	`PRD server 01`
`./Caption`	Caption from CI	`PRD server 01`
`/Caption`	Caption of the root (database) CIs	`PRD`
`../../Caption`	Selects the caption from the parent of the parent CI	`PRD`
`../RelationType`	Selects the parent relation type	`container_f`
`../../OMType`	Selects the parent of the parent type	`ACME_System`
`/OMType`	Selects type of the root CI	`ACME_System`
`//.[type='WINOSSPI: Infrastructure']/Caption`	Selects the caption of all CIs of type `WINOSSPI: Infrastructure`	`BLANTONS: Infrastructure`
`//Dependencies[type='hosted_on']/CI/Caption`	Selects the caption of all CIs with a `hosted_on` dependency	`BLANTONS`

`//Dependencies/CI/Caption`	Selects the caption of all CIs that have dependencies	`BLANTONS`

Note: If the Xpath expression selects a node below the starting database node, the "`..`" reads back one step. The following expression reads down to the node `db` and then links back to the starting database node.

```
//dependencies[type='hosted_on']/CI/../..
```

However, if the node `db` is the starting node, the expression `../..` follows the containment links of the node `db`, which is not the dependency relation that is shown in this example. The result depends on the parent container of the node, which is a different hierarchy.

Uploading the Sync Package to the OMi server

Topology data processing is performed in OMi, therefore you need to upload the sync package to the server:

1. On the OMi server, copy the folder that contains the mapping rules to `%topaz_home%\conf\opr\topology-sync\sync-packages\`. For example, if you created the package in the folder `C:\temp\mysyncpkg`, copy the folder `mysyncpkg`.
2. Execute:

   `%topaz_home%\bin\opr-sdtool.[sh|bat] –uploadpackage <path to the sync folder>`
3. Open the OMi UI and add the name of the topology sync folder to the list of packages:
   a. Select **Administration** > **Setup and Maintenance** > **Infrastructure Settings**.
   b. In the Applications drop-down list, select **Operations Management**.
   c. Under Operations Management - HPOM Topology Synchronization Settings search for **Packages for Topology Sync** and click the edit button.
   d. At the end of the string, add a semicolon (`;`) and the folder name and save. For example, to add the folder mysyncpkg, append:
   <existing list>`;mysyncpkg`

Topology Discovery Syntax

The XML file that the discovery script creates and to which the mapping rules are applied must use the BSM Connector topology discovery syntax described in this section.

This section includes:

- "Configuration Item XML Schema Definition (XSD)" on the next page
- "Configuration Item XML Element Description" on page 110
- "Example Discovery XML" on page 111

Configuration Item XML Schema Definition (XSD)

Your discovery script must output XML that conforms to the following schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="Service">
      <xs:complexType>
         <xs:choice maxOccurs="unbounded">
            <xs:element ref="NewInstance" />
            <xs:element ref="NewRelationship" />
         </xs:choice>
      </xs:complexType>
      <xs:key name="InstanceKey">
         <xs:selector xpath="NewInstance">
         </xs:selector>
         <xs:field xpath="Key"></xs:field>
      </xs:key>
      <xs:keyref refer="InstanceKey" name="InstanceKeyRef">
         <xs:selector xpath="NewInstance">
         </xs:selector>
         <xs:field xpath="@ref"></xs:field>
      </xs:keyref>
      <xs:keyref refer="InstanceKey" name="InstanceRef">
         <xs:selector xpath="NewRelationship/*/Instance">
         </xs:selector>
         <xs:field xpath="@ref"></xs:field>
      </xs:keyref>
   </xs:element>
   <xs:element name="NewInstance" type="InstanceType" />
   <xs:complexType name="InstanceType">
      <xs:sequence>
         <xs:element ref="Std" />
         <xs:element ref="Virtual" minOccurs="0" />
         <xs:element ref="Key" />
         <xs:element ref="Attributes" />
      </xs:sequence>
      <xs:attribute name="ref" type="xs:string" use="required" />
   </xs:complexType>
   <xs:element name="NewRelationship" type="RelationType" />
   <xs:complexType name="RelationType">
      <xs:sequence>
         <xs:element ref="Parent" />
         <xs:element ref="GenericRelations" minOccurs="0" />
      </xs:sequence>
   </xs:complexType>
   <xs:element name="Std">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="DiscoveredElement" />
```

```
            </xs:restriction>
         </xs:simpleType>
      </xs:element>
      <xs:element name="Virtual">
         <xs:complexType />
      </xs:element>
      <xs:element name="Key" type="xs:string" />
      <xs:element name="Attributes">
         <xs:complexType>
            <xs:sequence>
               <xs:element ref="Attribute" maxOccurs="unbounded" />
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="Attribute">
         <xs:complexType>
            <xs:attribute name="value" type="xs:string" use="required" />
            <xs:attribute name="name" type="xs:string" use="required" />
         </xs:complexType>
      </xs:element>
      <xs:element name="Parent">
         <xs:complexType>
            <xs:sequence>
               <xs:element ref="Instance" />
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="GenericRelations" type="RelationsList" />
      <xs:complexType name="RelationsList">
         <xs:sequence>
            <xs:element name="Relations" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attribute name="type" type="xs:string" use="required" />
                  <xs:sequence>
                     <xs:element ref="Instance" maxOccurs="unbounded" />
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
      <xs:element name="Instance">
         <xs:complexType>
            <xs:attribute name="ref" type="xs:string" use="required" />
         </xs:complexType>
      </xs:element>
   </xs:schema>
```

Configuration Item XML Element Description

The following table describes the elements that the XML document can contain.

Element	Description
NewInstance	Represents a discovered CI. You must add a `ref` attribute, which must match the unique CI ID that you specify in the *Key* element. You can then use this reference in *Instance* elements in the current XML document if you want to create or delete relationships.
DeleteInstance	Represents a CI that you want to delete immediately. The agent automatically deletes previously discovered CIs from the agent repository if your discovery script runs five times (by default) without including the CI as a *NewInstance* in the XML document. **Note:** You can control how often the discovery script must run before a missing CI is automatically deleted by changing the agent parameter `INSTANCE_DELETION_THRESHOLD` in the `agtrep` namespace. However, if you specify this element, the agent deletes the CI immediately and publishes the change to the RTSM (Run-time Service Model).
NewRelationship	Defines a new relationship between CIs. This element must contain exactly one *Parent* element and can contain one or more *GenericRelations* elements.
DeleteRelationship	Defines relationships that you want to delete. This element must contain exactly one *Parent* element and can contain one or more *GenericRelations* elements.
Std	Must contain the string `DiscoveredElement`.
Virtual	Include this element if the CI is virtual. A virtual CI is abstract and does not exist on any node CI. Omit this element if the CI is hosted on a node CI.
Key	Contains the full CI ID for this CI, which must be unique. You must include this element in all *NewInstance* elements. You must not specify a *NewInstance* with the same key in the same XML document.
Attributes	Contains *Attribute* elements.
Attribute	Each *Attribute* element has the attributes `name` and `value`. You can also specify a datatype. Attributes with the following names have a special meaning: • `hpom_citype` specifies the CI type as stored in the RTSM (for example, `nt`). • Attribute names with the prefix `ucmdb_` map directly to CI attributes (for example, `ucmdb_primary_dns_name` maps to the CI attribute `Primary DNS Name`).

Element	Description
Parent	Contains an *Instance* element, which defines the CI that is the parent of this relationship. The parent instance that you specify must exist in the RTSM and in the agent repository on the BSM Connector server. Therefore, you may need to include a *NewInstance* element to add the parent to the agent repository, even if the parent already exists in the RTSM.
Instance	Has a `ref` attribute that refers to a *NewInstance* element in the current XML document.
GenericRelations	Contains one or more *Relations* elements.
Relations	Has a `type` attribute that refers to the type of relation as stored in the RTSM (for example, `usage`). Contains one or more *Instance* elements, which refer to the CIs that are related to the specified *Parent* element. For complex topologies which require a sync package, add the prefix `omrel_`. For example, if the type is `membership` when no sync package is used, it becomes `omrel_membership` with sync packages.

Example Discovery XML

The following example XML creates three CI instances with the IDs `myUniqueID1`, `myUniqueID2`, and `myUniqueID3`. `myUniqueID1` is the parent CI, and has a contains relationshipt to `myUniqueID2` and `myUniqueID3`.

Example:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Service>
  <NewInstance ref="myUniqueID1">
    <Key>myUniqueID1</Key>
    <Std>DiscoveredElement</Std>
    <Attributes>
      <Attribute name="myAttribute" value="myAttributeValue" />
      <Attribute name="myAttribute1" value="myAttributeValue1" />
      <Attribute name="myType" value="myTypeValue" />
      <Attribute name="title" value="myTitle1" />
    </Attributes>
  </NewInstance>
  <NewInstance ref="myUniqueID2">
    <Key>myUniqueID2</Key>
    <Std>DiscoveredElement</Std>
    <Attributes>
      <Attribute name="myAttribute" value="myAttributeValue" />
      <Attribute name="myAttribute1" value="myAttributeValuew" />
      <Attribute name="myType" value="myTypeValue1" />
      <Attribute name="title" value="myTitle2" />
```

```
    </Attributes>
  </NewInstance>
  <NewInstance ref="myUniqueID3">
    <Key>myUniqueID3</Key>
    <Std>DiscoveredElement</Std>
    <Attributes>
      <Attribute name="myAttribute" value="myAttributeValue" />
      <Attribute name="myAttribute1" value="myAttributeValuey" />
      <Attribute name="myType" value="myTypeValue2" />
      <Attribute name="title" value="myTitle3" />
    </Attributes>
  </NewInstance>
  <NewRelationship>
    <Parent>
      <Instance ref="myUniqueID1"/>
    </Parent>
    <GenericRelations>
      <Relations type="omrel_contains">
        <Instance ref="myUniqueID2"/>
      </Relations>
      <Relations type="omrel_contains">
        <Instance ref="myUniqueID3"/>
      </Relations>
   </GenericRelations>
  </NewRelationship>
</Service>
```

Troubleshooting

This section describes how to troubleshoot BSM Connector local topology synchronization.

This section includes:

- "Troubleshooting local topology synchronization" below
- "Validating XML Mapping Files" on the next page
- "Mapping Log Files" on the next page
- "Writing Rules" on the next page
- "Using the topology CLI tool" on page 114

Troubleshooting local topology synchronization

Make sure the input XML file is written and updated correctly. Each time the topology policy runs, the mapping rules are applied to the whole discovery XML file, not only to appended entries.

For troubleshooting the mapping engine, see "Mapping Log Files" on the next page.

Validating XML Mapping Files

You can use the supplied XML schema definitions to validate the correctness of XML configuration files. You can also use the supplied XML schema definition files to make writing new configuration files easier when using a suitable XML editor (for example Eclipse).

XSD XML Schema Definition is a standard from World Wide Web Consortium (W3C) for describing and validating the contents of XML files. XSD files are provided for all XML configuration files. For more information, see the XML Schema documentation by W3C available from the following Web site: http://www.w3.org/XML/Schema.

Each mapping file is automatically validated against the associated XSD file whenever it is read. If a file cannot be validated, an error message is written to the error log that describes the location of the error in the validated file.

The schema files are stored in the following directory:

`%topaz_home%\conf\opr\topology-sync\schemas`

The files are:

`package.xsd`

Validates the `package.xml` file in each synchronization package.

`mapping.xsd`

Validates the following mapping files contained in the synchronization packages:

- Context mapping - `contextmapping.xml`
- Type mapping - `typemapping.xml`
- Attribute mapping - `attributemapping.xml`
- Relation mapping - `relationmapping.xml`

Mapping Log Files

The mapping engine generates the following log file:

`%topaz_home%\logs\opr-ts.log`

Writing Rules

Use following guidelines when writing rules:

- **Simplify Rule Development.** You can ease the writing of rules by selecting an XML editor that can validate and suggest elements according to an XML schema.
- **Avoid Complex XPath Queries.** Avoid complex XPath queries, especially in general conditions, where such queries must be applied to every CI. If you cannot avoid a complex XPath query, try to narrow the condition using operators such as `OMType`, combined using the `And` operator. Make sure that the simpler, non-XPath conditions are checked first (hint: `And` is an exclusive operator).
- **Match Against Existing Attributes Only.** Accessing attributes that do not exist for all CIs is very performance intensive in combination with a relative expression depending on the complexity of the

CI hierarchy.

- **Avoid Broad XPath Expressions.** Certain complex XPath expressions can result in excessive processing loads. For example, XPath expressions that include the following characteristics:
  - Apply to multiple nodes, such as expressions that contain `//` or `descendants:*/`
  - Do not match nodes or match only on nodes that are very distant from the current node

  The same applies to the `XPathResultList` operator that returns all matched values. The time required for such operations grows approximately quadratically with the size of a hierarchy. Avoid such expressions where possible.

  When using the descendants operator, do not use the star symbol (`*`) as node test, but specify the name of the node of interest. For example, do not use `descendants:*/caption` but use `descendants:ci/caption`.

  If you cannot avoid such an XPath expression within a condition, try to limit its execution by using the exclusive `And` operator and perform simple tests before the `XPathResult` operand is being used. For example, you could first check for the CI type.

Using the topology CLI tool

Use the `topology` command line tool to list all topology repositories, delete repositories, and send the last known delta or repository to the ServiceDiscoveryServer (SvcDscSvr).

The tool is located in

`%OvInstallDir%/bin/win64` (Windows systems)

`/opt/OV/bin` (Linux systems).

Usage

```
topology -h
-list
-publishrepo <SELECTION>
-publishdelta <SELECTION>
-clean <SELECTION>


<SELECTION>:
   [-all | -polname <name>[/<src>|/*] | -polid <uuid>[/<src>|/*]]
```

Option descriptions

Option	Description
-h, --help	Show this help message.
-list	Lists all topology repositories.
-publishrepo <*SELECTION*>	Sends the repository through BBC to the SvcDscSvr.

-publishdelta <*SELECTION*>	Sends the last delta through BBC to the SvcDscSvr.
-clean <*SELECTION*>	Deletes the repository and the last delta file.

Chapter 9: Collecting Generic Output Data

By configuring generic output policies, you can collect generic output data from third-party systems. Data collected by using these policies is often forwarded to consumer applications which do not require data models. Generic output policies require minimal configuration, without the additional overhead (such as defaults, rules, and similar) compared to creating event or metric policies.

Generic output data policies consist of a source definition and, optionally, key field mappings which map the source fields to the required result fields. Additionally, you can completely redefine the output data by renaming the meta model of the incoming data set, dropping unneeded fields, and adding custom fields.

How to Collect Generic Output Data

This section contains an overview of the tasks you must perform when collecting generic output data.

These tasks are the following:

1. **Plan the data collection and mappings**

   Define the data you want to collect and forward to the consumer application and which mappings are required to make the data usable for the application.

2. **Configure the policy sources**

   Collecting generic data from sources is similar as with event and metric policies. You can gather data through the REST web service, use SQL queries to gather data from databases, patterns to gather data from log and XML files, or gather data by using Perl scripts.

   For details on how to configure the sources for each integration type, see the sections titled "How to Collect Generic Output Data" for the respective integrations:

   - "How to Collect Generic Output Data from Databases" on page 127
   - "How to Collect Generic Output Data by Using Perl Scripts" on page 163
   - "How to Collect Generic Output Data from Structured Log Files" on page 236
   - "How to Collect Generic Output Data from XML Files" on page 262
   - "How to Collect Generic Output Data Through the REST Web Service Listener" on page 187

3. **Map the key fields and add additional fields**

   a. Unlike event and metric policies, generic output data policies have no defaults and rules. The mapping is done solely by mapping input filed qualifiers to their replacement fields.

      - With *structured data* such as XML and structured log files, only the eligible fields (the leaf node) are mapped. For example, if the data you gather has the following structure:

        ```
        /data/evt/timestamp
        /data/evt/event_type
        /data/evt/counter
        ```

only the fields `timestamp`, `event_type`, and `counter` are eligible for mapping.

If you want to rename (remap) part of the path too, you need to create additional rules for that part of the path. In our example, the path is:

```
/data/evt
```

Here, the eligible part is now `evt`, so if we map it to `event`, the mapped structure is now:

```
/data/event/timestamp
/data/event/event_type
/data/event/counter
```

Note that all paths that contain the eligible field are now changed.

- With *non-structured data*, such as data gathered with Perl scripts or from databases, fields are directly mapped to replacements. This means that the entire filed is eligible.

b. Decide whether to keep unmapped fields or not. By default, all fields are kept, including the unmapped ones, but you can also choose to discard the unmapped fields.

   With structured data, if unmapped fields are kept, BSM Connector keeps all parents of a mapped key are kept, keeping that the structure is maintained.

c. Optionally, you can add *additional fields* to the data set that is sent to Data Forwarding targets, for example, you can add descriptions, notes, and similar.

   Additional fields are simple key-value pairs and you need to manually add both the keys and values. If an added key field name clashes with the one in the mapped data set, it is discarded. The actual field value can consist of static text and data references from the input data (`<$DATA:…>`). If the input data is structured, additional fields are added as immediate children of the highest level element.

For details on how to configure the mapping for each integration type, see the appropriate section for each integration:

- "Configuring Mappings in Database Policies (Generic Output Only)" on page 135
- "Configuring Mappings in Perl Policies (Generic Output Only) " on page 169
- "Configuring Mappings in Structured Log File Policies (Generic Output Only) " on page 244
- "Configuring Mappings in XML File Policies (Generic Output Only)" on page 269
- "Configuring Mappings in REST Web Service Listener Policies (Generic Output Only)" on page 195

Chapter 10: Forwarding Data

In HP BSM Connector, performance data collected from various sources is stored locally in a store provided by the HP Operations Agent. With Data Forwarding policies, you can forward the collected data directly to consumer applications through HTTP(S) POST requests to REST web service endpoints (targets).

By using data forwarding polices, you can:

- configure several targets that receive data
- define rules on what data is sent to which target
- configure rules to discard specific data for some targets
- send data that was collected by an HP BSM Connector policy before or after the policy rules are processed
- run multiple policies at the same time.

You do not need to change existing HP BSM Connector policies to enable data forwarding.

How to Forward Data

This task describes how to forward data by using BSM Connector data forwarding policies:

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. *Optional prerequisite:* In some data forwarding scenarios you may need to configure a proxy. You can perform this in the underlying OM Agent communication layer. In a most basic case, you can use the OM Agent tool `ovconfchg`, where the basic setting takes the form:

   ```
   [bbc.http]
   PROXY=web-proxy.company.com:8088
   ```

   For details on how to configure the OM Agent communication layer, see the OM Agent documentation.
3. In the BSM Connector user interface, click in the toolbar. Then click **Forwarding >** **Data Forwarding**.

   Alternatively, double-click an existing policy to edit it.
4. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Data Forwarding Properties" on the next page.
5. In the **Targets** page, define the targets to which data is sent to.

   For details, see "Configuring Data Forwarding Targets" on page 120.

6. Configure data forwarding rules:
   - To forward metric data, select the **Metric** page and configure metric data forwarding rules.
     For details, see "Configuring Metric Data Forwarding" on page 121.
   - To forward structured input data, select the **Structured Input** page and configure structured input data forwarding rules.
     For details, see "Configuring Structured Input Data Forwarding" on page 122.
7. Click **Save and Close** to save the policy and close the editor.
8. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

Data Forwarding User Interface

This section includes:

- "Configuring Data Forwarding Properties" below
- "Configuring Data Forwarding Targets" on the next page
- "Configuring Metric Data Forwarding" on page 121
- "Configuring Structured Input Data Forwarding" on page 122

Configuring Data Forwarding Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the policy name and the description of what the policy does.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Forwarding >** **Data Forwarding**.

Alternatively, double-click an existing policy to edit it.

Tasks

How to configure policy properties

In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

For more information about the other fields, see the "Configuring Data Forwarding Properties" above.

Related tasks

- "How to Collect Generic Output Data" on page 116

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring Data Forwarding Targets

The Targets page enables you to define a list of targets which receive the data.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Forwarding >** **Data Forwarding**.

Alternatively, double-click an existing policy to edit it.

Click **Targets** to open the policy Targets page.

Learn More

A receiver target is a REST web service endpoint set up by a consumer application and is defined by its URL. The target can receive data in one of the two supported formats: JSON or XML. The consumer application that maintains the endpoint also defines whether data should be sent with a mechanism that guarantees delivery. You can define several different targets in a single policy.

Note that you must specify the port as part of the URL, even for the standard HTTP port. For example, if your target is `http://computer.company.com/bsmc_data`, using the HTTP port, you need to specify `http://computer.company.com:80/bsmc_data`.

Tasks

How to configure Data Forwarder Targets

1. In the target list, click above the Name column to add a new target.
2. Enter the name of the target and its description.
3. Provide the URL of the web service end point.
4. Select the wire format. The following formats are supported: XML and JSON.
5. To enable guaranteed delivery, select the option **Use Guaranteed Delivery**.
6. Click **Save** to add the target to the list.

UI Descriptions

For a description of the Targets page, see the following section:

- "Configuring Data Forwarding Targets" above

Configuring Metric Data Forwarding

To forward metric data after policy rules are applied, configure the metric forwarding rules.

Metric forwarding rules define what a Data Forwarding policy should do in response to specific metric data. Each rule consists of a condition and of settings for the data generated by the policy. The settings enable you to configure what data BSM Connector forwards to which target.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Forwarding >** **Data Forwarding**.

Alternatively, double-click an existing policy to edit it.

Click **Metric** to open the policy Metric page.

Learn More

This section includes:

- "Rule types" below

Rule types

The rule types are:

- **Forward on matched**

  If matched, BSM Connector forwards the metric data to the specified targets.
- **Discard on matched**

  If matched, BSM Connector discards the metric data.
- **Discard on unmatched**

  If not matched, BSM Connector discards the metric data.

Contrary to rules for other policies where processing stops when a rule is matched, all rules are processed for every data type. BSM Connector then compares the lists of targets to which the data should be sent to with the list of targets where data should be discarded. The final result is a list of targets where the data record should be sent to. This means that discard rules have a higher precedence.

Tasks

How to configure data forwarding:

1. In the Data Forwarding Rules list, click above the Description column to add a new rule. Select the rule type.
2. In the **Properties** tab, enter a description for the rule.

3. In the **Condition** tab, click to add new conditions.
   a. In the **Property** field, enter the name of the property or drag it from the Meta Data tab.
   b. Select the pattern operator.
      For a list of available operators, see "Operator" on page 313.
   c. In the **Operand** field, type the value or pattern that you want the policy to compare with the property.
4. In the **Targets** tab, select one or more targets from the list of available targets. For the rule to affect all targets, you select the option **Affect all targets**.
   Optionally, you can override the target configuration for individual or all targets by selecting the option **Override Target Configuration**. If you select this option, you can change the **Wire Format** and change the setting of the **Use Guaranteed Delivery** option.

UI Descriptions

For a description of the Metric page, see the following sections:

- "Metric Page (Data Forwarding)" on page 325

Configuring Structured Input Data Forwarding

To forward structured input data, that is, data gathered by a policy before the policy rules are applied, configure the structured input forwarding rules.

Structured input data rules define what a Data Forwarding policy should do in response to specific structured input data. Each rule consists of a condition and of settings for the data generated by the policy. The settings enable you to configure what data BSM Connector forwards to which target.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Forwarding >** **Data Forwarding**.

Alternatively, double-click an existing policy to edit it.

Click **Structured Input** to open the policy Structured Input page.

Learn More

This section includes:

- "Rule types" below

Rule types

The rule types are:

- **Forward on matched**

  If matched, BSM Connector forwards the structured input data to the specified targets.
- **Discard on matched**

  If matched, BSM Connector discards the structured input data.
- **Discard on unmatched**

  If not matched, BSM Connector discards the structured input data.

Unlike with rules for other policies where processing stops when a rule is matched, all rules are processed for every data type. BSM Connector then compares the lists of targets to which the data should be sent to with the list of targets where data should be discarded. The final result is a unified list of targets where the data record should be sent to. As a result, discarding rules have a higher precedence than forwarding rules.

Tasks

How to configure data forwarding

1. In the Data Forwarding Rules list, click above the Description column to add a new rule. Select the rule type.
2. In the **Properties** tab, enter a description for the rule.
3. In the **Condition** tab, click to add new conditions.
   a. In the **Property** field, enter the name of the property or drag it from the Meta Data tab.
   b. Select the pattern operator.

      For a list of available operators, see "Operator" on page 313.
   c. In the **Operand** field, type the value or pattern that you want the policy to compare with the property.
4. In the **Targets** tab, select one or more targets from the list of available targets. If the rule should affect all targets, you can select the option **Affect all targets**.

   Optionally, you can override the target configuration for individual or all targets by selecting the option **Override Target Configuration**. If you select this option, you can change the **Wire Format** and change the setting of **Use Guaranteed Delivery**.

UI Descriptions

For a description of the Structured Input page, see the following sections:

- "Structured Input Page (Data Forwarding)" on page 360

Part III: Integrating Data With BSM Connector

This section includes:

- "Database Policies (Events, Metrics, and Generic Output)" on page 125
- "Open Message Interface Policies (Events only)" on page 151
- "Perl Script Policies" on page 162
- "Scheduled Task Policies (Events only)" on page 211
- "SNMP Interceptor Policies (Events only)" on page 227
- "Structured Log File Policies (Events, Metrics, and Generic Output)" on page 234
- "REST Web Service Listener Policies" on page 185
- "XML File Policies" on page 260

Chapter 11: Database Policies (Events, Metrics, and Generic Output)

Database policies enable you to collect data from database tables used by third-party systems by performing a query through a JDBC connection.

The following are examples of data that can be integrated into OMi using database policies:

- Events from monitoring applications event tables or views.
- Data from monitoring applications metrics tables.
- Generic output data from monitoring applications tables.

What Data Is Forwarded

Database policies use a user defined SQL query, an initial value statement, and a static initial value (session variable). The query you provide is used to define a search criterion on the database and with the initial value statement and a set up session variable, you can provide initial values that you can use in filtering. Starting with OMi 10.00, you are no longer limited to a fixed set of SQL clauses but can write more complex queries as well.

For example, if you fetch a field that contains a time stamp, you can write an SQL query to process only data records that happened after the policy was activated. You can do so by setting an appropriate initial value statement that is executed when the policy is activated and then compare the values retrieved in the main SQL clause against this initial value. For details on how the queries are used to fetch and filter records, see "Understanding How Data From an SQL Query is Processed" on page 129.

You use defaults and rules to control the data that is sent from BSM Connector to OMi.

How to Collect Event Data from Databases

This task describes how to collect event data from databases.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Database**.
   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).
   For details, see "Configuring Database Policy Properties" on page 128.
4. In the **Source** page, define the connection to the database and the columns that the policy

accesses.

For details, see "Configuring the Data Source in Database Policies" on page 129.

5. In the **Mappings** page, configure the default mappings of table columns to custom variables.

   For details, see "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133.

6. *Optional.* In the **Defaults** page, configure the default settings for all events generated by the policy (for example, default event correlation settings).

   For details, see "Configuring Event Defaults in Database Policies" on page 137.

7. In the **Rules** page, define what the policy should do in response to a specific type of event.

   For details, see "Configuring Event Rules in Database Policies" on page 142.

8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in Database Policies" on page 149.

9. Click **Save and Close** to save the policy and close the editor.
10. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Metrics Data from Databases

This task describes how to collect metrics from databases.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Database**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Database Policy Properties" on page 128.
4. In the **Source** page, define the connection to the database and the columns that the policy accesses.

   For details, see "Configuring the Data Source in Database Policies" on page 129.
5. In the **Mappings** page, map columns of database tables to custom variables.

   For details, see "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133.
6. *Optional.* In the **Defaults** page, assign default values to the metric attributes.

   For details, see "Configuring Metrics Defaults in Database Policies" on page 139.

7. In the **Rules** page, define what the policy should do in response to a specific metric.
   For details, see "Configuring Metrics Rules in Database Policies" on page 145.
8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).
   For details, see "Configuring Options in Database Policies" on page 149.
9. Click **Save and Close** to save the policy and close the editor.
10. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Generic Output Data from Databases

This task describes how to collect data from databases.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Generic output > Database**.
   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).
   For details, see "Configuring Database Policy Properties" on the next page.
4. In the **Source** page, define the connection to the database and the columns that the policy accesses.
   For details, see "Configuring the Data Source in Database Policies" on page 129.
5. In the **Mappings** page, configure the default mappings of input data properties to custom fields.
   For details, see "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133.
6. Click **Save and Close** to save the policy and close the editor.
7. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

Database Policy User Interface

Database policies display different pages depending on the type of data they are integrating and on the policy origin.

Choose the type of data you want to integrate and then complete the following tasks:

Event data

- "Configuring Database Policy Properties" on the next page

- "Configuring the Data Source in Database Policies" on the next page
- "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133
- "Configuring Event Defaults in Database Policies" on page 137
- "Configuring Event Rules in Database Policies" on page 142
- "Configuring Options in Database Policies" on page 149

Metrics data

- "Configuring Database Policy Properties" below
- "Configuring the Data Source in Database Policies" on the next page
- "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133
- "Configuring Metrics Defaults in Database Policies" on page 139
- "Configuring Metrics Rules in Database Policies" on page 145
- "Configuring Options in Database Policies" on page 149

Generic output data

- "Configuring Database Policy Properties" below
- "Configuring the Data Source in Database Policies" on the next page
- "Configuring Mappings in Database Policies (Generic Output Only)" on page 135

Configuring Database Policy Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the policy name and a description of what the policy does.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Database**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Database**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Database**.

Alternatively, double-click an existing policy to edit it.

Tasks

How to configure policy properties

In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

Related tasks

- "How to Collect Event Data from Databases" on page 125
- "How to Collect Metrics Data from Databases" on page 126

- "How to Collect Generic Output Data from Databases" on page 127

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring the Data Source in Database Policies

The source page of the database policy editor enables you to set up the connection to the database and to specify which database tables the policy accesses.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Database**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Database**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Source** to open the policy Source page.

Learn More

This section includes:

- "Database drivers packages supplied with BSM Connector" below
- "Understanding How Data From an SQL Query is Processed" below

Database drivers packages supplied with BSM Connector

BSM Connector 10.01 does not provide any production driver packages. You need to install a compatible JDBC database driver provided by your database vendor or a third party driver.

In previous releases, BSM Connector provided driver packages that could be used in a *test* environment. Starting with BSM Connector 10.00, the testing drivers are no longer supplied.

Understanding How Data From an SQL Query is Processed

When querying the database - for example with a combination of `select` and `from` SQL clauses - all the rows from the table defined in the clause are retrieved, unless you define a further clause that filters the data. For example, you may want to retrieve only entries that were added after a certain point in time and not all entries in the table. To do so, you can compare the results returned by the SQL clause against a *data map* that you can populate with an initial SQL statement or by setting up a session variable for each key.

When the *initial SQL statement* is executed, the data map is initialized with values. The values returned as last record from the execution of the statement are copied to the data map and returned. You can

then compare the values in your SQL query against the values in the data map by referring to the keys (<$DATA:<*key*>>). This will be replaced with the current value of the <*key*> in the data map.

To make sure that the data map is not empty if the initial SQL statement does not return any values, you should set a session variable for each key. This value is then used to replace <$DATA:<*key*>> in the executable SQL statement.

For an example, see "How to configure the database source" on the next page.

Tasks

This section includes:

- "How to configure the prerequisites" below
- "How to configure the database source" on the next page

How to configure the prerequisites

- There are several key database driver requirements for using this policy.
  - You must install or copy a compatible JDBC database driver or database access API locally on the BSM Connector systems. Many database driver packages are available as compressed (zipped) archive files or .jar files. If the file is in zip format, unzip the contents. A recommended location for the downloaded driver is the directory the `C:/Program Files/HP/HP BTO Software/java`.
  - You must know the syntax for accessing the database driver. See the driver documentation for details.

    Examples of common database driver strings are:
    - **com.microsoft.sqlserver.jdbc.SQLServerDriver** The Microsoft SQL Server JDBC driver.
    - **oracle.jdbc.OracleDriver** The Oracle JDBC driver.
- You must know the syntax for the Database Connection string. The Database Connection string normally includes the class of driver you are using, some key name relating to the supplier of the driver software, followed by a combination of server, host, and port identifiers. For details, see the driver documentation.

  ***Example*:** Database Connection URLs for Microsoft SQL Server for this policy:
  - **jdbc:sqlserver://<*hostname*>:1433;Database=<*name*>;**

    where <*hostname*> is the name of the host where the database is running and <*name*> is the name of the database.
- The database you want to query must be running, have a database name defined, and have at least one named table created in the database. In some cases, the database management software needs to be configured to enable connections by using the middleware or database driver.
- You need a valid user name and password to access and perform a query on the database. In some cases, the machine and user account that BSM Connector is running on must be given permissions to access the database.

- You must know a valid SQL query string for the database instance and database tables in the database you want to query. Consult your database administrator to work out required queries to use.
- Use a database client to connect to the relevant software database. Identify which tables contain the required data (the software schema documentation may help you with this).

How to configure the database source

This task describes how configure the database source and how the policy queries the database.

1. Configure the connection to the database:
   a. In the **Classpath** field, type the location of the `.jar` file that is loaded.

      Example: `C:/Program Files/HP/HP BTO Software/java/sqljdbc4.jar`
      (Microsoft SQL Server installed in the default folder)

   b. In the **JDBC Driver Class** field, type the name of the driver used to connect to the database. Use the Fully Qualified Class Name of the JDBC driver you are using.

      Example: `com.microsoft.sqlserver.jdbc.SQLServerDriver`
      (Microsoft SQL Server Driver class)

   c. In the **Connect string** field, type the URL to the database connection (referred to as an Authentication string).

      Example: `jdbc:sqlserver://system2.company.com:1433;Database=BSMEVENTS`
      (Microsoft SQL Server Driver connection string)

   d. Enter the username and password.

   e. Under **Polling interval**, specify how often the policy queries the database. Use the spinboxes to specify increments of seconds, minutes, hours, and days.

      Note: Make sure that you set this value to a minimum of `15` seconds to be able to save the policy.

   f. Optionally, enter any additional connection properties needed by the database to which you are connecting. See the database documentation for details.

2. Select the **Collection** tab and specify the SQL query:
   a. An **SQL statement** to query the data.

      Write a query that fits the database you are using and the type of data you are querying.

      For example, a basic query might look like this:

      `"select {* | <column>[,<column>...] [,<column> ...]} from <table> [,<table>...] [where <SQL clause to define select criteria>"]`

      `select` clause: enter ***** for all fields or a comma separated list of column names to be retrieved from the database.

      `from` clause: enter a table name or a comma separated list of tables from which the selected columns should be extracted.

      `where` clause: enables you to define the selection criteria. If you not define it, all the rows from the table defined in the `from` clause are retrieved.

The queries are not limited to this example set and you can write more complex ones that suit your needs.

Click to retrieve the specified table columns from the database. The results are displayed in the Sample Data tab of the policy.

See the example at the end of this step to see how you can connect the SQL statement, the initial value statement, and the memento.

b. Define initial values for keys by setting up the **Session Variables**.

Click to add a new field. Type the name of the key and its initial value. Alternatively, you can drag and drop entries from the Initial value sample data tab.

> **Note:** It is recommended that you set up a session variable *for each key*. If no records are found with the initial value statement, the value set up in the session variable is used to replace the key in the executable SQL statement. Without an initial value the SQL statement for the execute method is not valid.

See the example at the end of this step to see how you can connect the SQL statement, the initial value statement, and the session variable.

c. Enter the **Initial value** statement.

An SQL statement that is executed in the init method. It can be used to initialize the data map with values. The values returned as last record from the execution of the statement are copied to the data map and returned.

Click to retrieve the specified table columns from the database. The results are displayed in the Initial Value Sample Data tab of the policy.

Example:

The following example shows how you can set up a combination of an SQL statement, an initial value statement, and a session variable to query the table "dbo.ALL_EVENTS" that contains the columns "ID", "TITLE", and "TIME_RECEIVED", and to collect all new records after the activation of the policy.

SQL statement:

```
"select ID,TITLE,TIME_RECEIVED from dbo.ALL_EVENTS where TIME_RECEIVED >
'<$DATA:TIME_RECEIVED>' order by TIME_RECEIVED;"
```

`<$DATA:TIME_RECEIVED>` will be replaced with the current value of `TIME_RECEIVED` in the data map. Therefore this SQL statement will return all new records that are added to the table after the SQL init statement was called and set the value in the data map.

Initial value statement: The following initial statement is used to fill the data map when the policy is activated:

```
 "select top 1 ID,TIME_RECEIVED,TITLE from dbo.ALL_EVENTS order by TIME_
RECEIVED DESC;"
```

This statement returns the newest record from the table. The values of the three columns are stored in the data map and can later be accessed in the SQL statement.

Session variables: `TIME_RECEIVED "2014-09-09 00:00:00.000"`

This is the initial value for the key `TIME_RECEIVED`. If no records are found with the initial SQL statement, this value is used to replace `<$DATA:TIME_RECEIVED>` in the executable SQL statement.

3. Optionally, select the **Internals** tab and modify the default settings for the:
   a. **Fetch Size** - the maximum number of entries the policy sends at once.
   b. **Result size** - the maximum number of entries the policy retrieves from the database in each run.

   By adjusting the fetch and result sizes you can balance the loads on the database and the BSM Connector system.

Related tasks

- "How to Collect Event Data from Databases" on page 125
- "How to Collect Metrics Data from Databases" on page 126
- "How to Collect Generic Output Data from Databases" on page 127

UI Descriptions

For a description of the Source page, see the following sections:

- "Source Page - Database Policies" on page 345
- "UI Descriptions" on page 299

Configuring Mappings in Database Policies (Events and Metrics Only)

The Mappings page enables you to map columns of database tables to custom variables.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Database**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

A custom variable consists of a map name, an optional database table column, and one or more source and target value pairs. For example, you can assign the table column `SEVERITY` to the map name `mapSeverity`, and add a source value of `Serious`. You can then assign the target value `critical` to

the variable so that BSM Connector inserts the value `critical` into the event in all places where the variable is used and the source value is `Serious` in the database record.

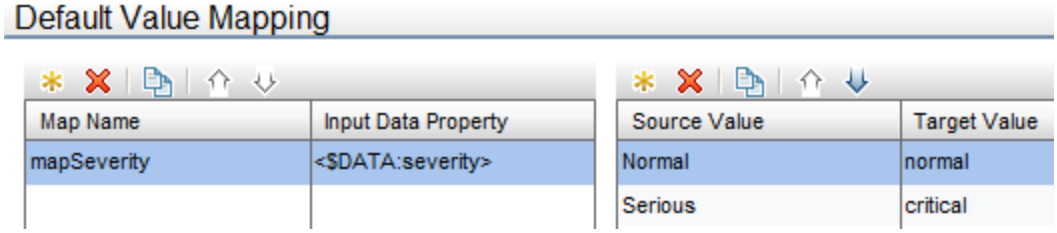


Table columns use the following syntax: `<$DATA:<table_column>>`

where `<table_column>` is the name of the table column in the third-party database, for example `<$DATA:severity>`.

For example, the custom variable `mapSEVERITY` has the table column `SEVERITY` assigned.

Assigning a table column to a map name is optional. If you do not assign a table column to a variable, you must add the source value directly to the variable when you insert the variable in an event attribute.

You can rearrange the information that appears in the lists by clicking the up or down arrows at the top of the list.

The Sample Data tab is empty if no sample data has been loaded into the policy or if the database query fails.

The Table Column tab shows the following information, if sample data is available:

- Input Data Properties

  If sample data is available, then the Input Data Properties section shows the table columns specified in the query.
- Values for `<table column>`

  The Values for <table column> section displays the values of a column selected in the Input Data Properties section. If a value appears more than once, click to show or hide duplicate values. To find values that belong to more than one row, select the value and click . The Database Sample Data window opens and shows all rows that have the selected value.

When you drag a column from the table columns list and drop it on the Default Value Mapping list, BSM Connector automatically adds the default prefix `map` to the map name and inserts the table column. You can then drag one or more column source values from the values list and drop them on the Source Value list. You then finally only have to type the target values.

Tasks

How to configure mappings for table columns

This task describes how to map table columns to custom variables.

1. Create one or more custom variables.

   If you are working with sample data, drag the table column from the table column list to the Map Name column. BSM Connector automatically adds the default prefix `map` to the map name and inserts the column name.

   Alternatively, click above the Map Name column and type the variable name in the map name field. Columns are optional. If you do not assign a column to a variable, you must add the source value directly to the variable when you insert the variable in an event attribute.
2. Add source and target value pairs to each custom variable.
   - If sample data is loaded in BSM Connector, drag a value from the Values for '...' list to the Source Value column, and then type the target value in the corresponding field.

     Alternatively, cClick above the Source Value column and type the source and target values in the corresponding fields.

   - *Optional.* In the Indicators tab, add indicators to the source or target value fields. After loading the indicators from the OMi server, the Indicators tab shows a hierarchy of configuration item types.

     To insert an indicator in a source or target value field, drag the indicator state (for example, `HTTPServer:Normal`) from the Indicators tab and drop it on the corresponding field.

   - *Optional.* In the Policy Variables tab, add policy variables to event or metric attributes. BSM Connector replaces the variables with the appropriate values in the generated event or metric.

     HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from Databases" on page 125
- "How to Collect Metrics Data from Databases" on page 126

UI Descriptions

For a description of the Mappings page, see the following sections:

- "UI Descriptions" on page 299
- "Sample Data Tab - Database Policies" on page 341
- "Indicators Tab" on page 320

Configuring Mappings in Database Policies (Generic Output Only)

The Mappings page enables you to map input field qualifiers to new field names.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Database**.

Alternatively, double-click an existing policy to edit it.

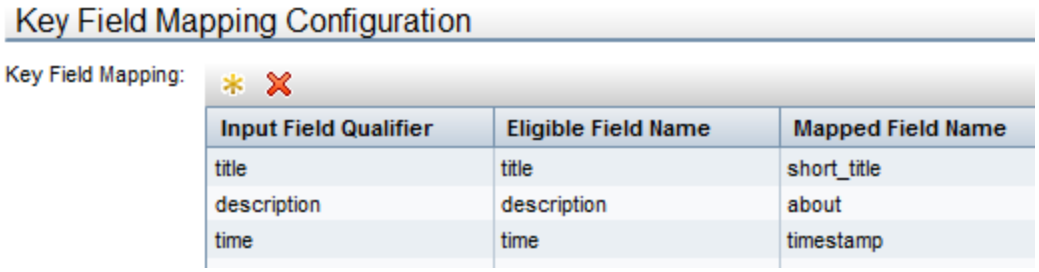
Click **Mappings** to open the policy Mappings page.

Learn More

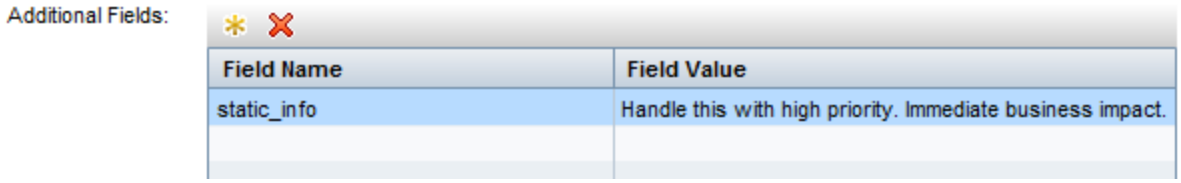
Mappings overview

The key field mapping consists of an input field qualifier, an eligible field name, and a mapped field name. To map the key field, map the eligible field name to the mapped field name. The eligible field name is automatically extracted from the input field qualifier.

Database policies integrate unstructured data. Therefore, the eligible field name is identical to the input data qualifier.



You can add **additional fields** to data that is sent to Data Forwarding targets. Additional fields are simple key-value pairs and you must manually add both the keys and values. If a defined key field name equals a field name in the mapped data set, it is discarded.



The actual field value can consist of user defined strings and data references to the input data (`<$DATA:…>`).

Example:

Additional Field Name	Additional Field Value
combined_text	At `<$DATA:time>`, `<$DATA:event_type>` detected an `<$DATA:event_impact>` impact on system performance. This has happened `<$DATA:counter>` times

The data references in the additional field `combined_text` are replaced when the policy is run. In the above example, after the variables are replaced, the resulting value in the field `combined_text` is:

```
At 12/05/2015 14:01:39, Monitoring: Threshold violation detected an substantial
impact on system performance. This has happened 8264 times.
```

Note that data references must refer to the *input* format, not the mapped format.

Tasks

How to configure mappings for key fields

This task describes how to map key fields.

1. Create one or more key field mappings.

   If you are working with meta data, drag the table column from the Meta Data list to the Input Field Qualifier column. BSM Connector automatically extracts the eligible field name.

   Alternatively, click above the Input Field Qualifier column and type the qualifier in the input field qualifier.
2. Optionally, change the **Keep all input fields** setting. By default, the option is selected and all fields are kept, regardless whether they are mapped or not. To keep only the mapped fields, deselect the option.
3. Add any additional fields as required.

Related tasks

- "How to Collect Generic Output Data from Databases" on page 127

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Generic Output)" on page 323

Configuring Event Defaults in Database Policies

The Defaults page enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Event Attributes page.

Tasks

How to configure event defaults for database policies

This task describes how to configure default settings for all events generated by the policy.

1. Click **Event Attributes** to define default event attributes, such as severity and category.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

2. Click **Event Correlation** to set the type of duplicate event suppression and define the method used to suppress duplicate events.
3. Click **Custom Attributes**to add additional information to all events generated by this policy. For example, you might add a company name, contact information, or a city location to an event.
4. Click **Advanced** to define default advanced attributes such as legacy HPOM attributes and agent MSI (Message Stream Interface) settings.
5. *Optional.* Use the Sample Data tab to drag table columns and values to the policy fields. Alternatively, you can type the path to the table column directly into the attribute box.

   Table columns use the following syntax: <$DATA:<*table_column*>>

   <*table_column*> is the name of the table column in the third-party database.

   BSM Connector replaces the table column at runtime with the value of the specified column. If you insert a column value, the value will be used.

   Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the database query did not succeed. See also "Configuring the Data Source in Database Policies" on page 129.

6. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings page. For more information, see "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133.

   Custom variable into the attribute box using the following syntax:

   <$MAP(<*custom_variable*>)> where <*custom_variable*> is the map name of the variable (for example, `<$MAP(mapSEVERITY)>`).

   If the custom variable does not have a table column Input Data Property, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - Name of the table column, for example `<$MAP(mapSEVERITY)>,<$DATA:SEVERITY>`
   - The source value itself, for example `<$MAP(SEVERITY)>,Critical)>`

7. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators

(ETIs).

8. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   HP recommends to surround variables with quotation marks, for example "<$MSG_NODE>" or "<$MSG_GEN_NODE>", at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from Databases" on page 125

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Sample Data Tab - Database Policies" on page 341
- "Mappings Tab" on page 325
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metrics Defaults in Database Policies

The Default Metric Attributes page enables you to assign default values to the metric attributes. The values can be used when defining the policy rules on the Rules tab, and can also be overridden there.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Metric Attributes page.

Learn More

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

  - Metric name

    Defines the metric name under which the metric appears in the Operations Agent store and consumers.

  - Related CI

    Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

  - Node

    Used to identify a node-like CI to which the performance records are associated to.

  - Value

    The actual performance value which is converted to 64-bit floating point number.

  - Time measured

    The time stamp when the value was determined in the third-party system.

- Advanced attributes:
  - Original metric name

    The name of the metric as used on the third-party system.

  - Unit

    The unit of the metric.

  - Integration id

    An id, used to identify the source of the integration.

Tasks

How to configure metrics defaults for database policies

This task describes how to configure metric attribute defaults for all metrics collected by this policy.

1. Define the metric attributes common to all metrics collected by this policy, such as metric class and name. All metrics in the **Basic** tab marked with a * are required. **Advanced** attributes are optional.
2. *Optional.* Use the Sample Data tab to drag table columns and values to the policy fields. Alternatively, you can type the table column directly into the attribute boxes.

   Table columns use the following syntax: <$DATA:<*table_column*>>

   <*table_column*> is the name of the table column in the third-party database.

   BSM Connector replaces the table column at runtime with the value of the specified column. If you insert a column value, the value will be used.

   Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the database query did not succeed. See also "Configuring the Data Source in Database Policies" on page 129.

3. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings page. For more information, see "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133.

   Custom variable into the attribute box using the following syntax:

   <$MAP(<*custom_variable*>)> where <*custom_variable*> is the map name of the variable (for example, `<$MAP(mapSEVERITY)>`).

   If the custom variable does not have a table column Input Data Property, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - Name of the table column, for example `<$MAP(mapSEVERITY)>,<$DATA:SEVERITY>`
   - The source value itself, for example `<$MAP(SEVERITY)>,Critical)>`

4. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string

   - the pattern definition

   - the output string if pattern matches on the input string

   - the output string if the pattern does not match (optional)

   Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

   `$MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)`

   - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

     For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

5. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.
6. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Metrics Data from Databases" on page 126

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Sample Data Tab - XML File Policies" on page 343
- "Mappings Tab" on page 325
- "Indicators Tab" on page 320
- "Operators Tab (Metrics Only)" on page 327
- "Policy Variables Tab " on page 333

Configuring Event Rules in Database Policies

Rules define the action a policy should take in response to a specific type of incoming event. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual event data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

Rule types

The rule types are:

- **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an event to OMi.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi.

Tasks

How to configure rules in database policies

This task describes how configure policy rules.

1. In the policy **Rules** section, click and select the type of rule to define what the policy should do in response to a specific value in a table column. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the table column and values that the policy searches for in the database that the policy accesses. If the policy finds a match, it may or may not generate an event, depending on the rule type.
   a. Click to create a new condition. New conditions by default use the equals operator.
   b. Click ▸ to expand the new condition.
   c. In the **Property** field, specify the name of the table column that the policy searches (for example, SEVERITY).

      If you are working with sample data, you can drag and drop the table column from the Table Column list to the Properties field.
   d. Select the pattern operator.

      If you select the matches operator, you can type a pattern in the Operand field. For a list of available operators, see "Operator" on page 313.
   e. In the **Operand** field, type the value or pattern that you want the policy to compare with the table column. If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.
3. Use the **Event Attributes** tab to define event attributes (for example, event title and description) for all events generated by this rule.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

4. Use the **Event Correlation** tab to set the type of duplicate event suppression and define the method used to suppress duplicate events.
5. Use the **Custom Attributes** tab to add additional information to all events generated by this rule. For example, you might add a company name, contact information, or a city location to an event.
6. Use the **Advanced** tab to define an event drill-down URL, legacy HPOM attributes, and agent MSI (Message Stream Interface) settings.
7. *Optional.* Use the Sample Data tab to drag table columns and values to the policy fields. Alternatively, you can type the path to the table column directly into the attribute box.

   Table columns use the following syntax: <$DATA:<*table_column*>>

   <*table_column*> is the name of the table column in the third-party database.

   BSM Connector replaces the table column at runtime with the value of the specified column. If you insert a column value, the value will be used.

   Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the database query did not succeed. See also "Configuring the Data Source in Database Policies" on page 129.

8. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings page. For more information, see "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133.

   Custom variable into the attribute box using the following syntax:

   <$MAP(<*custom_variable*>)> where <*custom_variable*> is the map name of the variable (for example, `<$MAP(mapSEVERITY)>`).

   If the custom variable does not have a table column Input Data Property, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - Name of the table column, for example `<$MAP(mapSEVERITY)>,<$DATA:SEVERITY>`
   - The source value itself, for example `<$MAP(SEVERITY)>,Critical)>`

9. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the event attribute.
10. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
11. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

    HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from Databases" on page 125
- "How to Collect Metrics Data from Databases" on page 126

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "Condition Tab - Rules Only" on page 312
- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Sample Data Tab - Database Policies" on page 341
- "Mappings Tab" on page 325
- "Pattern Matching Variables Tab (Events and Metrics Only)" on page 332
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metrics Rules in Database Policies

Rules define the action a policy should take in response to a specific type of incoming metric. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual metric data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

This section includes:

- "Rule types" below
- "Metric attributes" below

Rule types

The rule types are:

- **Store on matched rule.** If matched, BSM Connector stores metrics in a buffer until a maximum number of records or a maximum amount of time is reached. These metrics are sent to OMi when they are requested, and they use the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing.

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

  - Metric name

    Defines the metric name under which the metric appears in the Operations Agent store and consumers.

  - Related CI

    Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

  - Node

    Used to identify a node-like CI to which the performance records are associated to.

  - Value

The actual performance value which is converted to 64-bit floating point number.

- Time measured

  The time stamp when the value was determined in the third-party system.

- Advanced attributes:
  - Original metric name

    The name of the metric as used on the third-party system.

  - Unit

    The unit of the metric.

  - Integration id

    An id, used to identify the source of the integration.

Tasks

How to configure rules for metrics in database policies

1. In the policy **Rules** section, click and select the type of rule to define what the policy should do in response to a specific string in the table column.
2. In the **Rule Content** section, use the **Condition** tab to define the match condition.

   Click to create a new condition. Enter the property and select the operator from the drop-down list. Add the operand. Use the arrows and collapse/expand buttons to navigate the conditions.
3. *Optional.* If you are creating a rule of the type 'store on matched rule', set the attributes (**Basic** or/and **Advanced**) for metrics that you want the rule to override. If default attributes are specified in the Defaults tab, you use the defaults or you can override them as described below.

   The attributes are in two groups, basic and advanced. The metrics in the **Basic** tab are required. **Advanced** attributes are optional.
4. *Optional.* Use the Sample Data tab to drag table columns and values to the policy fields. Alternatively, you can type the table column directly into the attribute boxes.

   Table columns use the following syntax: <$DATA:<*table_column*>>

   <*table_column*> is the name of the table column in the third-party database.

   BSM Connector replaces the table column at runtime with the value of the specified column. If you insert a column value, the value will be used.

   Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the database query did not succeed. See also "Configuring the Data Source in Database Policies" on page 129.
5. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings page. For more information, see "Configuring Mappings in Database Policies (Events and Metrics Only)" on page 133.

Custom variable into the attribute box using the following syntax:

<$MAP(<*custom_variable*>)> where <*custom_variable*> is the map name of the variable (for example, `<$MAP(mapSEVERITY)>`).

If the custom variable does not have a table column Input Data Property, use the following syntax:

<$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

- Name of the table column, for example `<$MAP(mapSEVERITY)>,<$DATA:SEVERITY>`
- The source value itself, for example `<$MAP(SEVERITY)>,Critical)>`

6. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the metric attribute.

7. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string

   - the pattern definition

   - the output string if pattern matches on the input string

   - the output string if the pattern does not match (optional)

   Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

   `$MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)`

   - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

     For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

   Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

8. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

9. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Metrics Data from Databases" on page 126

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "UI Descriptions" on page 299
- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Sample Data Tab - Database Policies" on page 341
- "Operators Tab (Metrics Only)" on page 327
- "Policy Variables Tab " on page 333

Configuring Options in Database Policies

The options tab enables you to configure several policy behaviors, for example which events are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Database**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Options** to open the policy Options page.

Tasks

How to configure options for database file policies

In the Options page, configure which events or metrics are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

Related tasks

- "How to Collect Event Data from Databases" on page 125
- "How to Collect Metrics Data from Databases" on page 126

UI Descriptions

For a description of the Options page, see "Options Page (Events only)" on page 328 or "Options Page (Metrics only)" on page 330.

Troubleshooting Database Policies

This section describes troubleshooting and limitations when working with Database policies.

General troubleshooting guidelines

- To start troubleshooting, check the HP Operations Agent log files in the `%OvDataDir%log` directory (Windows systems) or the `/var/opt/OV/log` directory (Linux systems).
- To investigate issues related to policy execution, first examine the `%OvDataDir%log\System.txt` file (Windows systems) or the `/var/opt/OV/log/System.txt` file (Linux systems).

Note: The log files whose names start with the `opr-` prefix are generated by the OMi web console.

Chapter 12: Open Message Interface Policies (Events only)

BSM Connector can integrate data generated by its own open message interface command, opcmsg. Open message interface policies filter messages by defining rules for these messages.

For example, you might have a suppress on matched rule in the open message interface policy, which for example suppresses all submitted messages with `application=Test`. If you have such a condition and submit the call `opcmsg application=Test object=Object msg_text="Test message"`, then no event will be sent to OMi (the message has been suppressed).

BSM Connector also provides Java classes that you can use to implement a Java program that submits messages to open message interface policies.

How to Collect Event Data from the Open Message Interface

This task describes how to collect event data submitted by the opcmsg command-line tool to the open message interface.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar, then click **Event >** **Open Message Interface**. The open message interface policy editor opens.

   Alternatively, double-click an existing open message interface policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Open Message Interface Policy Properties" on page 156.
4. In the **Default Event Attributes** page, configure the default settings for all events generated by the policy (for example, default event correlation settings).

   For details, see "Configuring Event Defaults in Open Message Interface Policies" on page 156.
5. In the **Policy Rules** page, define what the policy should do in response to a specific type of message submitted to the open message interface.

   For details, see "Configuring Rules in Open Message Interface Policies" on page 158.
6. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in Open Message Interface Policies" on page 160.
7. Click **Save and Close** to save the policy and close the editor.

8. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.
9. Activate the open message interface policy to start the opcmsgi process on the BSM Connector server.
10. Use the `opcmsg` command-line tool to submit messages to the open message interface policy. For details, see "opcmsg Command-Line Tool" below.

    Alternatively, use the available Java classes or C API to implement a program that generates messages. For details, see "opcmsg Java API" on page 154 and "opcmsg C API" on page 155.

opcmsg Command-Line Tool

The command opcmsg generates a message that BSM Connector open message interface policies evaluate.

The opcmsg command-line tool is located in:

- Windows 32-bit: `%OvInstallDir%/bin/opcmsg`
- Windows 64-bit: `%OvInstallDir%/bin/win64/opcmsg`
- Linux: `/opt/OV/bin/opcmsg`

Note: To enable the opcmsg command-line tool, at least one open message interface policy must be activated on the BSM Connector server. Otherwise the command displays the following message:

```
The OVO Message Command is not configured on this system. Contact your OVO
Administrator to configure it. (OpC30-913)
```

Command Synopsis and Options

```
opcmsg [-help]
       [-id]
       [severity=normal|warning|minor|major|critical]
       application=<application>
       object=<object>
       msg_text=<text>
       [msg_grp=<message group>]
       [node=<node>]
       [service_id=<svcid>]
       [-option <var>=<value>]
```

Tip: You can specify any unique prefix for the available options. Note that the prefix for the option severity is **s** while the prefix for the option service_id is **ser**.

Option	Description
[-help]	*Optional.* Prints the usage message of opcmsg. All other options are ignored and no message is submitted.
[-id]	*Optional.* Returns the message ID of the submitted message to stdout. This option also sets the OPCDATA_REMARK_FOR_ACK flag of the message, so that the manager information of the message is held by the message agent.
[severity=normal|warning|minor|major|critical]	*Optional.* Sets the severity of the message. The following severities are supported: normal, warning, minor, major, critical. By default, the severity normal is applied.
application=<application>	Sets the application of the message. **Tip:** Specify the name of the application (or script or program) that is affected by or has detected the event or problem.
object=<object>	Sets the object of the message. **Tip:** Specify the name of the object (or process or sub-program) that is affected by or has detected the event or problem.
msg_text=<text>	Sets the message text of the message. **Tip:** Specify a descriptive text that explains the event or problem in more detail.
[msg_grp=<message group>]	*Optional.* Sets the message group to which the message belongs. By default, no message group is assigned.
[node=<node>]	*Optional.* Sets the node of the message. By default the system name of the BSM Connector system is applied. **Tip:** Specify the fully qualified domain name, the system name, or the IP address of the system on which the event or problem is detected.
[service_id=<svcid>]	*Optional.* Sets the service ID of the message. This is the ID of the service associated with the event. A service ID is a unique identifier for a service and can be used in OMi to identify the node and CI associated with the event.
[-option <var>=<value>]	*Optional.* Sets the variable `<$OPTION(variable)>` to `value`. You can use this variable within the policy conditions to access the value passed the opcmsg call. Special characters must be escaped.

Exit Values

Value	Description
0	Message is successfully generated.
1	Internal error.
2	Syntax or usage error. An error message displays.

Example

To submit a normal message issued when a user logs onto the system, you could set up the following scheduled task

```
opcmsg appl=ScheduledTask obj=login severity=normal msg_t="%USERNAME% logged onto
system %COMPUTERNAME%" -option CIHint=myApplication@@%COMPUTERNAME%
```

opcmsg Java API

HP Operations Agent provides a set of Java classes to generate messages for open message interface policies. The classes can be used to generate messages from within a Java program.

Note: The Java classes can also be used to acknowledge previously generated messages, and to send monitor values to the HP Operations Agent. BSM Connector does not support these uses.

JAR files

The JAR files `jopcagtbase.jar` and `jopcagtmsg.jar` that are necessary to use the APIs are installed as part of HP Operations Agent on the BSM Connector server at:

Windows: `%OvInstallDir%\java`

Linux: `/opt/OV/java`

Prerequisites

To use the Java classes:

- The -classpath parameter used for the javac and java commands must include the `jopcagtbase.jar` and `jopcagtmsg.jar` files.
- The PATH system variable must include the directory where the shared library files reside. The agent installation does this automatically.

Documentation

Javadoc style class documentation is available at the following location:

Windows: `%OvInstallDir%\www\htdocs\jdoc_agent\index.html`

Linux: `/opt/OV/www/htdocs/jdoc_agent/index.html`

For more information, see the section "Java API" in the *HP Operations Agent Reference Guide* PDF.

Examples

Examples of how the API classes can be used from Java are available in the following directory on the BSM Connector server:

Windows: `%OvInstallDir%\examples\jopcagtapi`

Linux: `/opt/OV/OpC/examples/jopcagtapi`

To compile and run the example code on Windows

1. Change to the folder `%OvInstallDir%/examples/jopcagtapi` on the BSM Connector server.
2. Compile the example code, type:

   ```
   javac -classpath
   "%OvInstallDir%/java/jopcagtbase.jar:%OvInstallDir%/java/jopcagtmsg.jar"
   JOpcAgtMsgTest.java
   ```
3. Run the example code, type:

   ```
   java -classpath
   ".:%OvInstallDir%/java/jopcagtbase.jar:%OvInstallDir%/java/jopcagtmsg.jar"
   JOpcAgtMsgTest
   ```

To compile and run the example code on Linux

1. Change to the directory `/opt/OV/OpC/examples/jopcagtapi` on the BSM Connector server.
2. Compile the example code, type:

   ```
   javac -classpath "/opt/OV/java/jopcagtbase.jar:/opt/OV/java/jopcagtmsg.jar"
   JOpcAgtMsgTest.java
   ```
3. Run the example code, type:

   ```
   java -classpath ".:/opt/OV/java/jopcagtbase.jar:/opt/OV/java/jopcagtmsg.jar"
   JOpcAgtMsgTest
   ```

opcmsg C API

HP Operations Agent provides a C-based API (application programming interface) to generate messages for open message interface policies. The API can be used to generate messages from within a C program.

Note: The API can also be used to acknowledge previously generated messages, and to send monitor values to the HP Operations Agent. BSM Connector does not support these uses.

For more information about the C-based message API, see the section "Agent Application Programming Interface" in the *HP Operations Agent Reference Guide* PDF.

Open Message Interface Policy User Interface

This section includes:

- "Configuring Open Message Interface Policy Properties" below
- "Configuring Event Defaults in Open Message Interface Policies" below
- "Configuring Rules in Open Message Interface Policies" on page 158
- "Configuring Options in Open Message Interface Policies" on page 160

Configuring Open Message Interface Policy Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the policy name and a description of what the policy does.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **Open Message Interface**. The open message interface policy editor opens.

Alternatively, double-click an existing open message interface policy to edit it.

Tasks

How to configure policy properties

In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

Related tasks

- "How to Collect Event Data from the Open Message Interface" on page 151

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring Event Defaults in Open Message Interface Policies

The Default Event Attributes page enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **Open Message Interface**. The open message interface policy editor opens.

Alternatively, double-click an existing open message interface policy to edit it.

Click **Defaults** to open the Default Event Attributes page.

Tasks

How to configure events for open message interface policies

This task describes how to configure default settings for all events generated by the policy.

1. Click **Event Attributes** to define default event attributes, such as category.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

2. Click **Event Correlation** to set the type of duplicate event suppression and define the method used to suppress duplicate events.
3. Click **Advanced** to define default advanced attributes such as legacy HPOM attributes and agent MSI (Message Stream Interface) settings.
4. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
5. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from the Open Message Interface" on page 151

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Advanced Tab" on page 310
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Rules in Open Message Interface Policies

Rules define the action a policy should take in response to a specific type of incoming event. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual event data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **Open Message Interface**. The open message interface policy editor opens.

Alternatively, double-click an existing open message interface policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

Rule types

The rule types are:

- **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an event to OMi.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi.

Tasks

How to configure rules in open message interface policies

This task describes how configure policy rules.

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific string in a message. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the attributes and values that the policy searches for in the message that the policy receives from opcmsg. If the policy finds a match, it may or may not generate an event, depending on the rule type.

a. In the **Node** field, type the fully qualified domain name, the node name, or the IP address if you only want to match messages whose node attribute is set to a specific node. Give multiple entries with the OR (|) operator (for example: `node1.example.com|node2.example.com`), or leave blank for all nodes.

   This field corresponds to the `node` option of the opcmsg command.

b. In the **Message Group** field, type the message group if you only want to match messages whose message group attribute is set to a specific message group. Give multiple entries with the OR (|) operator (for example: `msggrp1|msggrp2`), or leave blank for all message groups.

   This field corresponds to the `msg_grp` option of the opcmsg command.

c. In the **Application** field, type the name of the application if you only want to match messages whose application attribute is set to a specific application. Give multiple entries with the OR (|) operator (for example: `appl1|appl2`), or leave blank for all applications.

   This field corresponds to the `application` option of the opcmsg command.

d. In the **Object** field, type the name of the object if you only want to match messages whose object attribute is set to a specific object. Give multiple entries with the OR (|) operator (for example: `object1|object2`), or leave blank for all objects.

   This field corresponds to the `object` option of the opcmsg command.

   Note: Although the term *application* generally refers to a general program name and *object* generally refers to a process or sub-program, you should use these values to assist your own organizational scheme.

e. Clear the **Severity** checkboxes if you only want to match messages whose severity attribute is set to a specific severity. You can select multiple severities but must select at least one.

   This field corresponds to the `severity` option of the opcmsg command.

f. In the **Message Text** field, type the pattern that you want the policy to compare with the message text in the source message that it is evaluating.

   This field corresponds to the `msg_text` option of the opcmsg command.

   Tip: For matching patterns, you can use standard pattern-matching rules of HP Operations Agent. Select the matches operator and click the ▸ icon in the Operand field to open the pattern matching toolbox window. The toolbox includes the following sections:

   - **Pattern Matching Expressions.** Click an expression to insert it into the Operand text box.
   - **Variable Bindings Options.** Variable binding options include the setting of case sensitivity check and the field separators used in the rule. If you do not specify the pattern matching options for the rule, either the defaults (enabled case sensitivity check; the space and the tab character as the separators) or the default options set for the policy are used.

3. Use the **Event Attributes** tab to define event attributes (for example, event title and description) for all events generated by this rule.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

4. Use the **Event Correlation** tab to set the type of duplicate event suppression and define the method used to suppress duplicate events.
5. Use the **Custom Attributes** tab to add additional information to all events generated by this rule. For example, you might add a company name, contact information, or a city location to an event.
6. Use the **Advanced** tab to define an event drill-down URL, legacy HPOM attributes, and agent MSI (Message Stream Interface) settings.

Related tasks

- "How to Collect Event Data from the Open Message Interface" on page 151

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "Condition Tab - Open Message Interface Policy Rules Only" on page 313
- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310

Configuring Options in Open Message Interface Policies

The options tab enables you to configure several policy behaviors, for example which events are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **Open Message Interface**. The open message interface policy editor opens.

Alternatively, double-click an existing open message interface policy to edit it.

Click **Options** to open the policy Options page.

Tasks

How to configure options for open message policies

In the Options page, configure which events are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

For more information about the other fields, see "Configuring Options in Open Message Interface Policies" above.

Related tasks

- "How to Collect Event Data from the Open Message Interface" on page 151

UI Descriptions

For a description of the Options page, see "Options Page (Events only)" on page 328.

Chapter 13: Perl Script Policies

Perl script policies enable you to collect data from systems by periodically running Perl scripts which perform the actual data collection.

The Perl functionality that is available for scripting is limited only by the ootb Perl module in the Operations Agent Perl runtime, providing you with additional flexibility compared to policies that just read different file types (logs, XML) or receive data through an API set or web service.

How Data Is Collected

The policies process the data by first running a Perl script to gather data, applying policy defaults and rules, and then sending the processed data to OMi. The Perl script must provide the data in an array of hashes, which are represented as input properties in BSM Connector.

For details on how to write the Perl scripts and how to pass the data to BSM Connector, see "Configuring the Data Source in Perl Policies " on page 165.

How to Collect Event Data by Using Perl Scripts

This task describes how to collect event data by using Per scripts.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Perl Script**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Perl Policy Properties" on page 165.
4. In the **Source** page, define the path to the Perl script or upload the embedded script, set the polling interval, the input parameters, and define the result array and result data key names.

   For details, see "Configuring the Data Source in Perl Policies " on page 165.
5. In the **Mappings** page, configure the default mappings of the data properties to custom variables.

   For details, see "Configuring Mappings in Perl Policies (Events and Metrics Only) " on page 167.
6. *Optional.* In the **Defaults** page, configure the default settings for all events generated by the policy (for example, default event correlation settings).

   For details, see "Configuring Event Defaults in Perl Policies" on page 171.
7. In the **Rules** page, define what the policy should do in response to specific data generated by Perl scripts.

   For details, see "Configuring Event Rules in Perl Policies" on page 176.

8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).
   For details, see "Configuring Options in Perl Policies" on page 183.
9. Click **Save and Close** to save the policy and close the editor.
10. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Metrics Data by Using Perl Scripts

This task describes how to collect metrics data by using Perl scripts.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Perl Script**.
   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).
   For details, see "Configuring Perl Policy Properties" on page 165.
4. In the **Source** page, define the path to the Perl script or upload the embedded script, set the polling interval, the input parameters, and define the result array and result data key names.
   For details, see "Configuring the Data Source in Perl Policies " on page 165.
5. *Optional.* In the **Defaults** page, assign default values to the metric attributes.
   For details, see "Configuring Metric Defaults in Perl Policies" on page 173.
6. In the **Rules** page, define what the policy should do in response to a specific metric.
   For details, see "Configuring Metric Rules in Perl Policies" on page 179.
7. Click **Save and Close** to save the policy and close the editor.
8. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Generic Output Data by Using Perl Scripts

This task describes how to collect generic output data by using Perl scripts.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Perl Script**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Perl Policy Properties" on the next page.
4. In the **Source** page, define the path to the Perl script or upload the embedded script, set the polling interval, the input parameters, and define the result array and result data key names.

   For details, see "Configuring the Data Source in Perl Policies " on the next page.
5. In the **Mappings** page, configure the default mappings of the input data properties to custom fields.

   For details, see "Configuring Mappings in Perl Policies (Generic Output Only) " on page 169.
6. Click **Save and Close** to save the policy and close the editor.
7. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

Perl Policy User Interface

Perl policies display different pages depending on the type of data they are integrating and on the policy origin.

Choose the type of data you want to integrate and then complete the following tasks:

Event data

- "Configuring Perl Policy Properties" on the next page
- "Configuring the Data Source in Perl Policies " on the next page
- "Configuring Mappings in Perl Policies (Events and Metrics Only) " on page 167
- "Configuring Event Defaults in Perl Policies" on page 171
- "Configuring Event Rules in Perl Policies" on page 176
- "Configuring Options in Perl Policies" on page 183

Metrics data

- "Configuring Perl Policy Properties" on the next page
- "Configuring the Data Source in Perl Policies " on the next page
- "Configuring Mappings in Perl Policies (Events and Metrics Only) " on page 167
- "Configuring Metric Defaults in Perl Policies" on page 173
- "Configuring Metric Rules in Perl Policies" on page 179

- "Configuring Options in Perl Policies" on page 183

Generic output data

- "Configuring Perl Policy Properties" below
- "Configuring the Data Source in Perl Policies " below
- "Configuring Mappings in Perl Policies (Generic Output Only) " on page 169

Configuring Perl Policy Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the name and the description of the policy.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Perl Script**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Perl Script**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Perl Script**.

Alternatively, double-click an existing policy to edit it.

Tasks

How to configure policy properties

In the Properties page, type a name of the policy. You can use spaces in the name. The equal sign (=) is not allowed.

For more information about the other fields, see "Configuring Perl Policy Properties" above.

Related tasks

- "How to Collect Event Data by Using Perl Scripts" on page 162
- "How to Collect Metrics Data by Using Perl Scripts" on page 163
- "How to Collect Generic Output Data by Using Perl Scripts" on page 163

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring the Data Source in Perl Policies

The source page of the Perl Script policy editor enables you to set up the path to the Perl script or upload an embedded Perl script, define the polling interval, specify the subroutine that transfers the data and send any options to it, and specify the result data array and keys.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Perl Script**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Perl Script**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Perl Script**.

Alternatively, double-click an existing policy to edit it.

Click **Source** to open the policy Source page.

Learn More

Understanding How Data is Collected and Processed using Perl Scripts

Collecting and processing data

The Perl script that is run periodically can be either loaded from an external location or embedded in the policy. You must also provide the name of the subroutine (`sub`) that is executed and any parameters that are passed to it.

During the execution of the script, BSM Connector initializes an embedded Perl interpreter (the Operations Agent Perl runtime), and passes all provided parameters to the Perl subroutine. The subroutine then collects and processes the data.

Parallel processing of the Perl script is not supported.

Passing data to polices

Before the subroutine exits, all data to be passed to BSM Connector must be present in the transfer data structure. This data structure must be an *array of hash references* and available as a global variable in the Perl script. You can define the name of this array, but it must match the name that you specify in the policy configuration.

Each hash inside the array indicates a single record of information. The hash is formed by pairs of an attribute name (the hash key) and a value, both represented as strings. In BSM Connector terminology, the attribute name is an input property and can be accessed as `<$DATA:<attribute_name>>` in set operations and as `'<attribute_name>'` in conditions.

Assuming that the hash `%BASIC_METRIC_DATA`, which you use to pass data to BSM Connector, contains the following attribute names:

```
my %BASIC_METRIC_DATA = (
    "counter_name"  => "<value>",
    "counter_value" => "<value>",
    "time_measured" => "<value>"
);
```

In this case, the BSM Connector input properties are:

```
<$DATA:counter_name>
<$DATA:counter_value>
<$DATA:time_measured>
```

If the execution is successful, the subroutine should return `0`. Any other value indicates an error in the script execution and is logged to `system.txt`.

Tasks

This section includes:

- "How to configure the Perl script source" below
- "Configuring the Data Source in Perl Policies " on page 165

How to configure the Perl script source

This task describes how configure the Perl script source and how the policy reads the data.

1. Provide the path to the external Perl script or click to load an embedded script.
2. Define the script polling interval.
3. Provide the name of the subroutine that is executed. The name of the subroutine must match the name of the subroutine in the Perl script.
4. Optionally, provide a list of parameters that are passed to the subroutine. Click to add a new parameter and to remove it.

   The parameters can be saved to the policy encrypted and are decrypted only before the subroutine is called. To encode the parameter values, right-click the parameter and select the option **Encode as password**.
5. Provide the result data array name. The name must match the name of the result data array name in the subroutine.
6. Provide a list of result data key names. Click to add a new key name and to remove it.

Related tasks

- "How to Collect Event Data by Using Perl Scripts" on page 162
- "How to Collect Metrics Data by Using Perl Scripts" on page 163
- "How to Collect Generic Output Data by Using Perl Scripts" on page 163

UI Descriptions

For a description of the Source page, see "Source Page - Perl Source Policies" on page 355.

Configuring Mappings in Perl Policies (Events and Metrics Only)

The Mappings page enables you to map attribute keys gathered with the Perl script to custom variables.

To access

- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Event >** **Perl Script**.
- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Metric >** **Perl Script**.

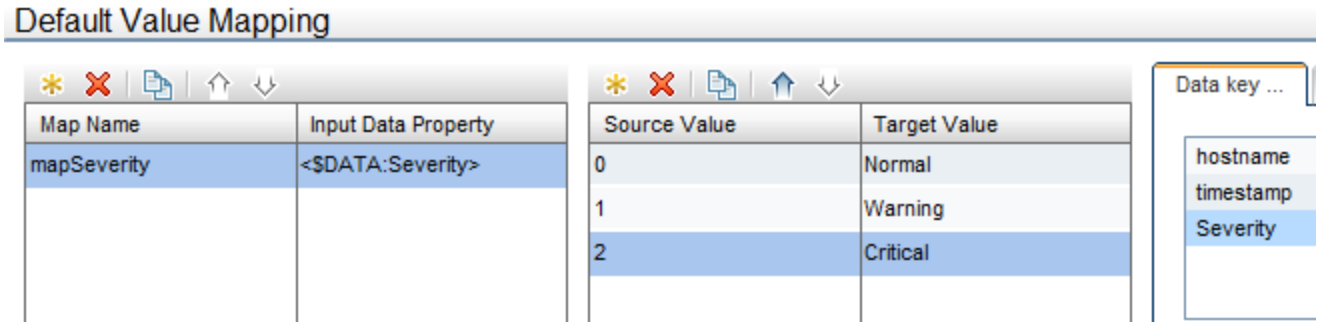
Alternatively, double-click an existing policy to edit it.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

A custom variable consists of a map name, a key attribute name, and one or more source and target value pairs. For example, you can assign the data key name `Severity` to the map name `mapSeverity`, and add a source value of `2`. You can then assign the target value `Critical` to the variable so that BSM Connector inserts the value `Critical` into the event in all places where the variable is used.



Perl attribute key names use the following syntax:

<$DATA:<*attribute_name*>>

where <*attribute_name*> is the data key name in a Perl hash array.

Tasks

How to configure Perl mappings (events and metrics)

This task describes how to map result data key names to custom variables.

1. Create one or more custom variables.

   Drag the data keys from the Data Key names list to the Map Name column. BSM Connector automatically adds the default prefix `map` to the map name and inserts the correct Input Data Property.

   Alternatively, click ✱ above the Map Name column and type the Data key name in the map name field.
2. Add source and target value pairs to each custom variable.

- Click above the Source Value column and type the source and target values in the corresponding fields.
- *Optional.* In the Indicators tab, add indicators to the source or target value fields. After loading the indicators from the OMi server, the Indicators tab shows a hierarchy of configuration item types.

  To insert an indicator in a source or target value field, drag the indicator state (for example, `HTTPServer:Normal`) from the Indicators tab and drop it on the corresponding field.
- *Optional.* In the Policy Variables tab, add policy variables to event or metric attributes. BSM Connector replaces the variables with the appropriate values in the generated event or metric.

  HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data by Using Perl Scripts" on page 162
- "How to Collect Metrics Data by Using Perl Scripts" on page 163

UI Descriptions

For a description of the Mappings page, see the following sections:

- "UI Descriptions" on page 299
- "Sample Data Tab - REST Web Service Listener Policies" on page 342
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Mappings in Perl Policies (Generic Output Only)

The Mappings page enables you to map input field qualifiers gathered with Perl scripts to new fields.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Perl Script**.

Alternatively, double-click an existing policy to edit it.

Click **Mappings** to open the policy Mappings page.

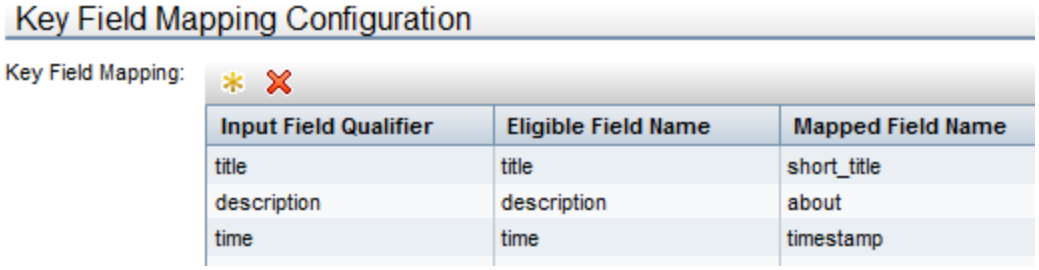
Learn More

Mappings overview

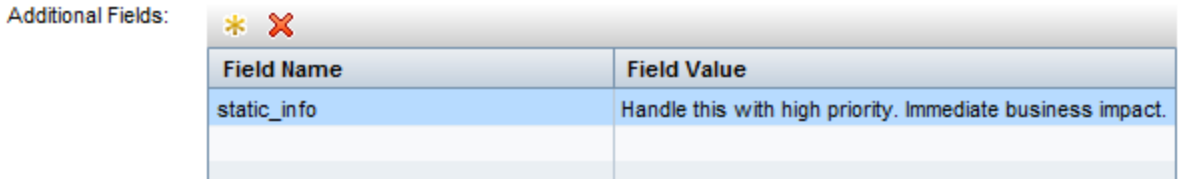
The key field mapping consists of an input field qualifier, an eligible field name, and a mapped field name. To map the key field, map the eligible field name to the mapped field name. The eligible field

name is automatically extracted from the input field qualifier.

Perl script policies integrate unstructured data. Therefore, the eligible field name is identical to the input data qualifier.



You can add **additional fields** to data that is sent to Data Forwarding targets. Additional fields are simple key-value pairs and you must manually add both the keys and values. If a defined key field name equals a field name in the mapped data set, it is discarded.



The actual field value can consist of user defined strings and data references to the input data (`<$DATA:…>`).

Example:

Additional Field Name	Additional Field Value
combined_text	At `<$DATA:time>`, `<$DATA:event_type>` detected an `<$DATA:event_impact>` impact on system performance. This has happened `<$DATA:counter>` times

The data references in the additional field `combined_text` are replaced when the policy is run. In the above example, after the variables are replaced, the resulting value in the field `combined_text` is:

```
At 12/05/2015 14:01:39, Monitoring: Threshold violation detected an substantial
impact on system performance. This has happened 8264 times.
```

Note that data references must refer to the *input* format, not the mapped format.

Tasks

How to configure mappings for key fields

This task describes how to map key fields.

1. Create one or more key field mappings.

   If you are working with meta data, drag the table column from the Meta Data list to the Input Field Qualifier column. BSM Connector automatically extracts the eligible field name.

   Alternatively, click above the Input Field Qualifier column and type the qualifier in the input field qualifier.
2. Optionally, change the **Keep all input fields** setting. By default, the option is selected and all fields are kept, regardless whether they are mapped or not. To keep only the mapped fields, deselect the option.
3. Add any additional fields as required.

Related tasks

- "How to Collect Generic Output Data by Using Perl Scripts" on page 163

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Generic Output)" on page 323

Configuring Event Defaults in Perl Policies

The Event Defaults page enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent uses the defaults for the new event.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Perl Script**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Event Attributes page.

Tasks

How to configure event defaults for Perl policies

This task describes how to configure default settings for all events generated by the policy.

1. Click **Event Attributes** to define default event attributes, such as severity and category.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.
2. Click **Event Correlation** to set the type of duplicate event suppression and define the method

used to suppress duplicate events.

3. Click **Custom Attributes** to add additional information to all events generated by this policy. For example, you might add a company name, contact information, or a city location to an event.
4. Click **Advanced** to define default advanced attributes such as legacy HPOM attributes and agent MSI (Message Stream Interface) settings.
5. Optionally, use the **Data key names** tab to drag data keys to the attribute boxes. Alternatively, you can type the name of the data key directly into the attribute box.

   Perl attribute key names use the following syntax:

   <$DATA:<*attribute_name*>>

   where <*attribute_name*> is the data key name in a Perl hash array.

   BSM Connector replaces the data key at runtime with the value of the specified data key.

   Note: The Data key names tab is empty if no result data key names have been specified in the Source page of the Perl Script policy. See also "Configuring the Data Source in Perl Policies " on page 165.

6. Optionally, use the **Mappings** tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings tab (see also "Configuring Event Defaults in Perl Policies" on the previous page). The default name of the mapping variable is `map<attribute_name>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box by using the following syntax:

   - Map Name list contains the map name of the variable: `map<attribute_name>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:<attribute_name>>`

     For example, the custom variable `mapSeverity` has the `<$DATA:Severity>` input data property, where `Severity` is the name of the hash key value.

7. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
8. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data by Using Perl Scripts" on page 162

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Mappings Tab" on page 325
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metric Defaults in Perl Policies

The Default Metric Attributes page enables you to assign default values to the metric attributes. The values can be used when defining the policy rules on the Rules tab, and can also be overridden there.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Perl Script**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Metric Attributes page.

Learn More

This section includes:

- "Metric attributes" below
- "Configuring Metric Defaults in Perl Policies" above

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

- Metric name

  Defines the metric name under which the metric appears in the Operations Agent store and consumers.

- Related CI

  Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

- Node

  Used to identify a node-like CI to which the performance records are associated to.

- Value

  The actual performance value which is converted to 64-bit floating point number.

- Time measured

  The time stamp when the value was determined in the third-party system.

- Advanced attributes:
  - Original metric name

    The name of the metric as used on the third-party system.

  - Unit

    The unit of the metric.

  - Integration id

    An id, used to identify the source of the integration.

Tasks

How to configure metric defaults for Perl script policies

This task describes how to configure metric attribute defaults for all metrics collected by this policy.

1. Define the metric attributes common to all metrics collected by this policy, such as metric class and name. All metrics in the **Basic** tab marked with a ***** are required. **Advanced** attributes are optional.
2. Optionally, use the **Data key names** tab to drag data keys to the attribute boxes. Alternatively, you can type the name of the data key directly into the attribute box.

   Perl attribute key names use the following syntax:

   <$DATA:<*attribute_name*>>

   where <*attribute_name*> is the data key name in a Perl hash array.

   BSM Connector replaces the data key at runtime with the value of the specified data key.

Note: The Data key names tab is empty if no result data key names have been specified in the Source page of the Perl Script policy. See also "Configuring the Data Source in Perl Policies " on page 165.

3. Optionally, use the **Mappings** tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings tab (see also "Configuring Metric Defaults in Perl Policies" on page 173). The default name of the mapping variable is `map<attribute_name>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box by using the following syntax:

   - Map Name list contains the map name of the variable: `map<attribute_name>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:<attribute_name>>`

     For example, the custom variable `mapSeverity` has the `<$DATA:Severity>` input data property, where `Severity` is the name of the hash key value.

4. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string

   - the pattern definition

   - the output string if pattern matches on the input string

   - the output string if the pattern does not match (optional)

   Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

   ```
   $MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)
   ```

   - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

     For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

   Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

5. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.
6. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Metrics Data by Using Perl Scripts" on page 163

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "UI Descriptions" on page 299
- "Mappings Tab" on page 325
- "Operators Tab (Metrics Only)" on page 327
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Event Rules in Perl Policies

Rules define the action a policy should take in response to a specific type of incoming event. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual event data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Perl Script**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

Rule types

The rule types are:

- **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an

event to OMi.

- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi.

Tasks

How to configure event rules in Perl script policies

This task describes how to configure policy rules.

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific string in the Perl script data. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the values that the policy searches for in the data that the policy reads. If the policy finds a match, it may or may not generate an event, depending on the rule type.
   a. Click to create a new condition. New conditions by default use the equals operator.
   b. Click ▸ to expand the new condition.
   c. In the **Property** field, specify the attribute that the policy searches for.

      You can drag and drop the data key from the Data key names list to the Property field.
   d. Select the pattern operator from the Operator drop-down list.
   e. In the **Operand** field, type the value or pattern that you want the policy to compare with the data key value. Click on the right arrow to open a list of pattern matching expressions.

      For a complete list of available operators, see "Operator" on page 313.
3. Use the **Event Attributes** tab to define event attributes (for example, event title and description) for all events generated by this rule.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.
4. Use the **Event Correlation** tab to set the type of duplicate event suppression and define the method used to suppress duplicate events.
5. Use the **Custom Attributes** tab to add additional information to all events generated by this rule. For example, you might add a company name, contact information, or a city location to an event.
6. Use the **Advanced** tab to define an event drill-down URL, legacy OM attributes, and agent MSI (Message Stream Interface) settings.
7. Optionally, use the **Data key names** tab to drag data keys to the attribute boxes. Alternatively, you can type the name of the data key directly into the attribute box.

   Perl attribute key names use the following syntax:

   <$DATA:<*attribute_name*>>

   where <*attribute_name*> is the data key name in a Perl hash array.

   BSM Connector replaces the data key at runtime with the value of the specified data key.

Note: The Data key names tab is empty if no result data key names have been specified in the Source page of the Perl Script policy. See also "Configuring the Data Source in Perl Policies " on page 165.

8. Optionally, use the **Mappings** tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings tab (see also "Configuring Event Rules in Perl Policies" on page 176). The default name of the mapping variable is `map<attribute_name>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box by using the following syntax:

   - Map Name list contains the map name of the variable: `map<attribute_name>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:<attribute_name>>`

     For example, the custom variable `mapSeverity` has the `<$DATA:Severity>` input data property, where `Severity` is the name of the hash key value.

9. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the event attribute.

10. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

    `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

    - the input string

    - the pattern definition

    - the output string if pattern matches on the input string

    - the output string if the pattern does not match (optional)

    Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

    ```
    $MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)
    ```

    - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

      For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

    Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

11. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of

configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).

12. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

    HP recommends to surround variables with quotation marks, for example "<$MSG_NODE>" or "<$MSG_GEN_NODE>", at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data by Using Perl Scripts" on page 162
- "How to Collect Metrics Data by Using Perl Scripts" on page 163

UI Descriptions

For the description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "Condition Tab - Rules Only" on page 312
- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Mappings Tab" on page 325
- "Pattern Matching Variables Tab (Events and Metrics Only)" on page 332
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metric Rules in Perl Policies

Rules define the action a policy should take in response to a specific type of incoming metric. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual metric data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Perl Script**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

This section includes:

- "Rule types" below
- "Metric attributes" below
- "Configuring Metric Rules in Perl Policies" on the previous page

Rule types

The rule types are:

- **Store on matched rule.** If matched, BSM Connector stores metrics in a buffer until a maximum number of records or a maximum amount of time is reached. These metrics are sent to OMi when they are requested, and they use the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing.

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

  - Metric name

    Defines the metric name under which the metric appears in the Operations Agent store and consumers.

  - Related CI

Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

- Node

  Used to identify a node-like CI to which the performance records are associated to.

- Value

  The actual performance value which is converted to 64-bit floating point number.

- Time measured

  The time stamp when the value was determined in the third-party system.

- Advanced attributes:
  - Original metric name

    The name of the metric as used on the third-party system.

  - Unit

    The unit of the metric.

  - Integration id

    An id, used to identify the source of the integration.

Tasks

How to configure rules for metrics in Perl script policies

This task describes how to configure policy rules.

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific string in the data gathered by the Perl script. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the values that the policy searches for in the data the policy reads. If the policy finds a match, it may or may not generate a metric record, depending on the rule type.
   a. Click to create a new condition. New conditions by default use the `equals` operator.
   b. Click ▸ to expand the new condition.
   c. In the **Property** field, specify the attribute that the policy searches for.

      You can drag and drop the data key from the Data Keys name list to the Properties field.
   d. Select the pattern operator.

      If you select the matches operator, you can type a pattern in the Operand field. For a list of available operators, see "Operator" on page 313.

e. In the **Operand** field, type the value or pattern that you want the policy to compare with the attributes.

3. *Optional.* If you are creating a rule of the type 'Store on matched rule', set the attributes (**Basic** and/or **Advanced**) for metrics that you want the rule to override. All metrics on the **Basic** tab are required. **Advanced** attributes are optional. If default attributes are specified in the Defaults tab, you can use the defaults or you can override them as described below.
4. Optionally, use the **Data key names** tab to drag data keys to the attribute boxes. Alternatively, you can type the name of the data key directly into the attribute box.

   Perl attribute key names use the following syntax:

   <$DATA:<*attribute_name*>>

   where <*attribute_name*> is the data key name in a Perl hash array.

   BSM Connector replaces the data key at runtime with the value of the specified data key.

   Note: The Data key names tab is empty if no result data key names have been specified in the Source page of the Perl Script policy. See also "Configuring the Data Source in Perl Policies " on page 165.

5. Optionally, use the **Mappings** tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings tab (see also "Configuring Metric Rules in Perl Policies" on page 179). The default name of the mapping variable is `map<attribute_name>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box by using the following syntax:

   - Map Name list contains the map name of the variable: `map<attribute_name>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:<attribute_name>>`

     For example, the custom variable `mapSeverity` has the `<$DATA:Severity>` input data property, where `Severity` is the name of the hash key value.

6. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the metric attribute.
7. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string

   - the pattern definition

   - the output string if pattern matches on the input string

   - the output string if the pattern does not match (optional)

Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

```
$MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)
```

- `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

  For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

8. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.
9. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Metrics Data by Using Perl Scripts" on page 163

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "UI Descriptions" on page 299
- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Mappings Tab" on page 325
- "Pattern Matching Variables Tab (Events and Metrics Only)" on page 332
- "Operators Tab (Metrics Only)" on page 327
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Options in Perl Policies

The Options tab enables you to configure several policy behaviors, for example to define which events are logged locally, how the policy manages unmatched events, and to set defaults for pattern matching in policy rules.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Perl Script**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **Perl Script**.

Alternatively, double-click an existing policy to edit it.

Click **Options** to open the policy Options page.

Tasks

How to configure options for the Perl policies

In the Options page, configure which events or metrics are logged locally, how the policy handles unmatched events and metrics, and defaults for pattern matching in policy rules.

For more information, see "Options Page (Events only)" on page 328.

Related tasks

- "How to Collect Event Data by Using Perl Scripts" on page 162
- "How to Collect Metrics Data by Using Perl Scripts" on page 163

UI Descriptions

For a description of the Options page, see "Options Page (Events only)" on page 328.

Chapter 14: REST Web Service Listener Policies

REST Web service policies receive and process data from third party systems through the BSM Connector REST Web service listener since collection is done outside.

The chunks of the XML data are sent to the BSM Connector REST Web service listener by using the HTTP POST method.

What Data Is Collected

REST Web service process XML data that is received by the BSM Connector REST Web service listener. The policies process the data by applying policy defaults and rules, and then send the data to OMi.

How to Collect Event Data Through the REST Web Service Listener

This task describes how to collect event data through the BSM Connector REST Web service listener.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Event >** **REST Web Service Listener**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring REST Web Service Listener Policy Properties" on page 189.
4. In the **Source** page, define the path parameter that is a part of the actual URL where the REST Web Service Listener listens for the XML data from the Web service client.

   For details, see "Configuring the Data Source in REST Web Service Listener Policies " on page 189.
5. In the **Mappings** page, configure the default mappings of the XML properties to custom variables.

   For details, see "Configuring Mappings in REST Web Service Listener Policies " on page 193.
6. *Optional.* In the **Defaults** page, configure the default settings for all events generated by the policy (for example, default event correlation settings).

   For details, see "Configuring Event Defaults in REST Web Service Listener Policies" on page 197.
7. In the **Rules** page, define what the policy should do in response to a specific Web service data.

For details, see "Configuring Event Rules in REST Web Service Listener Policies" on page 202.

8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in REST Web Service Listener Policies" on page 209.
9. Click **Save and Close** to save the policy and close the editor.
10. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Metrics Data Through the REST Web Service Listener

This task describes how to collect metrics data through the BSM Connector REST Web service listener.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **REST Web Service Listener**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring REST Web Service Listener Policy Properties" on page 189.
4. In the **Source** page,define the path parameter that is a part of the actual URL where the REST Web Service Listener listens for the XML data from the Web service client.

   For details, see "Configuring the Data Source in REST Web Service Listener Policies " on page 189.
5. *Optional.* In the **Defaults** page,assign default values to the metric attributes.

   For details, see "Configuring Metrics Defaults in REST Web Service Listener Policies" on page 199.
6. In the **Rules** page, define what the policy should do in response to a specific metric.

   For details, see "Configuring Metrics Rules in REST Web Service Listener Policies" on page 205.
7. Click **Save and Close** to save the policy and close the editor.
8. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

Configuring Topology Through the REST Web Service Listener

You can collect the topology data by using the separate REST Web Service Topology policy, which is configured as follows:

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Topology >** **REST Web Service Listener**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring REST Web Service Listener Policy Properties" on page 189.
4. In the **Source** page,define age to deletion and specify whether delta detection should be done.

   For details, see "How to configure the REST Web service listener source (topology)" on page 192.
5. Click **Save and Close** to save the policy and close the editor.
6. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Generic Output Data Through the REST Web Service Listener

This task describes how to collect event data through the BSM Connector REST Web service listener.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **REST Web Service Listener**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring REST Web Service Listener Policy Properties" on page 189.
4. In the **Source** page, define the path parameter that is a part of the actual URL where the REST

Web Service Listener listens for the XML data from the Web service client.

For details, see "Configuring the Data Source in REST Web Service Listener Policies " on the next page.

5. In the **Mappings** page, configure the default mappings of the XML properties to custom variables.

   For details, see "Configuring Mappings in REST Web Service Listener Policies " on page 193.
6. Click **Save and Close** to save the policy and close the editor.
7. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

REST Web Service Listener Policy User Interface

REST Web service listener policies display different pages depending on the type of data they are integrating and on the policy origin.

Choose the type of data you want to integrate and then complete the following tasks:

Event data

- "Configuring REST Web Service Listener Policy Properties" on the next page
- "Configuring the Data Source in REST Web Service Listener Policies " on the next page
- "Configuring Mappings in REST Web Service Listener Policies " on page 193
- "Configuring Event Defaults in REST Web Service Listener Policies" on page 197
- "Configuring Event Rules in REST Web Service Listener Policies" on page 202
- "Configuring Options in REST Web Service Listener Policies" on page 209

Metrics data

- "Configuring REST Web Service Listener Policy Properties" on the next page
- "Configuring the Data Source in REST Web Service Listener Policies " on the next page
- "Configuring Mappings in REST Web Service Listener Policies " on page 193
- "Configuring Metrics Defaults in REST Web Service Listener Policies" on page 199
- "Configuring Metrics Rules in REST Web Service Listener Policies" on page 205
- "Configuring Options in REST Web Service Listener Policies" on page 209

Topology data

The topology data is integrated by using the separate REST Web Service Topology policy, which is configured as described in the "Configuring Topology Through the REST Web Service Listener" on the previous page

Generic Output data

- "Configuring REST Web Service Listener Policy Properties" on the next page
- "Configuring the Data Source in REST Web Service Listener Policies " on the next page
- "Configuring Mappings in REST Web Service Listener Policies (Generic Output Only)" on page 195

Configuring REST Web Service Listener Policy Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the policy name and a description of what the policy does.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **REST Web Service Listener**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **REST Web Service Listener**.
- In the BSM Connector user interface, click in the toolbar. Then click **Topology >** **REST Web Service Listener**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **REST Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Tasks

How to configure policy properties

In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

For more information about the other fields, see "Configuring REST Web Service Listener Policy Properties" above.

Related tasks

- "How to Collect Event Data Through the REST Web Service Listener" on page 185
- "How to Collect Metrics Data Through the REST Web Service Listener" on page 186
- "Configuring Topology Through the REST Web Service Listener" on page 187
- "How to Collect Generic Output Data Through the REST Web Service Listener" on page 187

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring the Data Source in REST Web Service Listener Policies

The source page of the REST Web service listener policy editor enables you to set up the path as part of URL where the REST Web Service Listener will listen for the XML data. In event and metric integrations, you can additionally upload the sample data to BSM Connector, and to specify one or

more XML tags that create the shortcuts to the XML elements that you want to process. When integrating the topology data, you can specify the age to deletion and to decide whether delta detection should be done.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event > REST Web Service Listener**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metrics > REST Web Service Listener**.
- In the BSM Connector user interface, click in the toolbar. Then click **Topology > REST Web Service Listener**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output > REST Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Source** to open the policy Source page.

Learn More

Customizing REST Web Service Listener

REST Web Service Listener listens for the XML data at the predefined URL. For example, for the event integration this URL should look as follows:

```
http(s)://<bsmc_FQDN>:<configured_port>/bsmc/rest/events/<PATH>
```

In this instance, *<PATH>* is the string, entered in the policy. The REST Web Service Listener for the metric and topology integrations should look as follows:

- `/bsmc/rest/metrics/`*`<PATH>`*
- `/bsmc/rest/topology/`*`<PATH>`*

You can customize the BSMC REST Web Service Listener by using the `bsmc-wsconf.[bat|sh]` utility, located at:

```
$OvDataDir/installation/HPOprBSMC/
```

For example, to enable basic authentication, set the listener port to 12345 and set the credentials any client needs to provide to post data, type the following:

```
bsmc-wsconf.bat -enable_auth -port 12345 -username integrator -password bsmcRESTd8a
```

For the `bsmc-wsconf.[bat|sh]` utility usage details, type:

```
bsmc-wsconf.[bat|sh] -help
```

The following table lists the available XPL configuration parameters that apply to each activated web service policy:

Attribute Name	Default	Description
RESTWS_PORT	30005	BSM Connector will spawn an HTTP(s) server on the specified port.
RESTWS_USE_BASIC_AUTH	false	If set to TRUE, the REST Web Service Listener requires credentials. The basic HTTP authentication mechanism is used.
RESTWS_AUTH_USER		Encrypted user name, used for the basic HTTP authentication. Encryption is done by using the OvSecCore crypt mechanism.
RESTWS_AUTH_PASSWORD		Encrypted password, used for the basic HTTP authentication. Encryption is done by using the OvSecCore crypt mechanism.
RESTWS_REGISTER_CB	false	If set to TRUE, the REST Web Service Listener is registered at the BBC communication broker. This enables accessing the Web Service Listener through the port 383.

You can view the XPL configuration parameters by using the ovconfget command.

Creating the XML data

REST Web Service accepts the regular XML data without an XML header structure ("<?xml version=....") and all-embracing XML root tag. Additionally, you can collect the data for multiple events and send them consolidated to the REST Web Service.

Note: A successful Web Service call should result in HTTP status code 200. Otherwise status codes like 501 or 401 are returned.

The following is an example of the XML event:

```
<BSMCEvent>
     <title>Detected neglectable impact on system hydrogen.elementary.com caused by
monitored aspect workers_restart_rate [1/s]: 0.0000 at Tue Aug  5 13:41:32
2014</title>
     <timeStamp>12/13/14 08:59:57 AM</timeStamp>
     <impact>neglectable</impact>
     <category>Monitor Alert</category>
     <relatedCounter>workers_restart_rate [1/s]</relatedCounter>
     <relatedEntity>hydrogen.elementary.com</relatedEntity>
</BSMCEvent>
```

Tasks

This section includes:

- "How to configure the REST Web service listener source (events and metrics)" on the next page
- "Events and Metrics only: How to load sample data into the policy" on the next page

How to configure the REST Web service listener source (events and metrics)

This task describes how configure the REST Web service listener source and how the policy reads it.

1. Enter the Path parameter string, which is a part of the actual URL where the REST Web Service Listener listens for the XML data from the Web service client.
2. Click to load a sample XML file. You can load a sample file from the BSM Connector system or from the system where the Web browser runs.

   When you load sample data, BSM Connector replaces already loaded data with the new data. This does not affect any mappings that are defined based on previously available sample data.
3. Click to create one or more XML event or metric tags.You can create a tag manually by typing the XML element. If you are working with sample data, you can create a tag by double-clicking the XML element in the list.

   The XML tag creates a shortcut to the XML element that you want the policy to process. An event tag typically identifies an event record in an XML log file, while a metric tag identifies a metric record in an XML log file. You can define more than one XML tag. For example, an XML file may contain two types of events: `<PerformanceAlert>` and `<AvailabilityAlert>`. To process both types, define both elements as event tags.

How to configure the REST Web service listener source (topology)

This task describes how configure the XML source file and how the policy reads it.

1. Enter the Path parameter string, which is a part of the actual URL where the REST Web Service Listener listens for the XML data from the Web service client.
2. Enter the age to deletion - the number of times for the policy to be run for a host that is not part of the topology data. After this number of policy executions, this host is deleted from the server.
3. Optionally, enable delta detection, so that only the difference between the received XML file and the repository is sent to the topology server.

Events and Metrics only: How to load sample data into the policy

1. Create a file that contains the XML data, and then send it to the REST Web Service listener.
2. Click to upload the XML file to the BSM Connector user interface policy editor. The results are displayed in the Sample Data tabs of the policy.

Related tasks

- "How to Collect Event Data Through the REST Web Service Listener" on page 185
- "How to Collect Metrics Data Through the REST Web Service Listener" on page 186
- "Configuring Topology Through the REST Web Service Listener" on page 187
- "How to Collect Generic Output Data Through the REST Web Service Listener" on page 187

UI Descriptions

For a description of the Source page, see "Source Page - REST Web Service Listener Policies" on page 353.

Configuring Mappings in REST Web Service Listener Policies

The Mappings page enables you to map key and value strings contained in Web service requests to custom variables.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **REST Web Service Listener**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **REST Web Service Listener**.

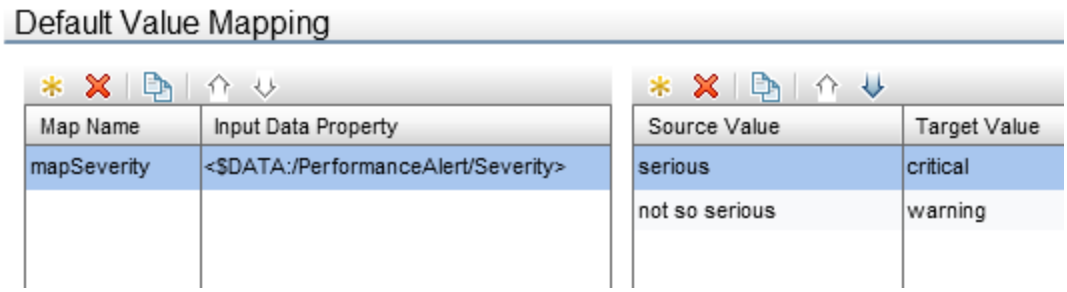
Alternatively, double-click an existing policy to edit it.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

A custom mapping definition consists of a map name (variable), an optional input data property (XML element or XML attribute), and one or more source and target value pairs. For example, you can assign the XML element `Severity` to the map name `mapSeverity`, and add a source value of `serious`. You can then assign the target value `critical` to the variable so that BSM Connector inserts the value `critical` into the event in all places where the variable is used.



XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

<XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

For example, the custom definition with the `mapSeverity` map name has the input data property of `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

XML properties are optional. If you do not assign an input data property to a map name (variable), you must add the source value directly to the variable when you insert the variable in an event attribute.

Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match any specified XML tags.

The Sample Data tab shows the following information if sample data is available:

- XML Properties

  If sample data is available, the XML Properties section of the Sample Data tab shows all XML elements and attributes that match an XML tag.

  The XML Properties section by default shows the short path to the XML property or value. To view the full path, click . The full path begins with the XML tag specified in the Source tab.

  The items in the XML Properties section are by default sorted alphabetically in ascending order.

  To search for an XML property or value, type the search string in the Search Properties box. The list changes as you type; only matching items appear.

- Values for *<XMLProperty>*

  This section displays the values of an XML property selected in the XML Properties section. If a value appears more than once, click to show or hide duplicate values. To find values that belong to more than one XML property, select the value and click . The XML Sample Data window opens and shows all XML properties that have the selected value.

Tasks

How to configure REST Web Service mappings (events and metrics)

This task describes how to map XML elements and attributes to custom variables.

1. Create one or more custom variables.

   If you are working with sample data, drag the XML elements or attributes from the XML Properties list to the Map Name column. BSM Connector automatically adds the default prefix `map` to the map name and inserts the correct path to the XML property.

   Alternatively, click above the Map Name column and type the variable name in the map name field. XML properties are optional. If you do not assign an XML property to a variable, you must add the source value directly to the variable when you insert the variable in an event or a metric attribute.

2. Add source and target value pairs to each custom variable.
   - If sample data is loaded in BSM Connector and shortcuts to XML elements are defined, drag a value from the Values for '...' list to the Source Value column, and then type the target value in the corresponding field.

     Alternatively, cClick above the Source Value column and type the source and target values in the corresponding fields.

   - *Optional.* In the Indicators tab, add indicators to the source or target value fields. After loading the indicators from the OMi server, the Indicators tab shows a hierarchy of configuration item types.

To insert an indicator in a source or target value field, drag the indicator state (for example, `HTTPServer:Normal`) from the Indicators tab and drop it on the corresponding field.

- *Optional.* In the Policy Variables tab, add policy variables to event or metric attributes. BSM Connector replaces the variables with the appropriate values in the generated event or metric.

  HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data Through the REST Web Service Listener" on page 185
- "How to Collect Metrics Data Through the REST Web Service Listener" on page 186

UI Descriptions

For a description of the Mappings page, see the following sections:

- "UI Descriptions" on page 299
- "Sample Data Tab - REST Web Service Listener Policies" on page 342
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Mappings in REST Web Service Listener Policies (Generic Output Only)

The Mappings page enables you to map key and value strings contained in Web service requests to custom variables.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **REST Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

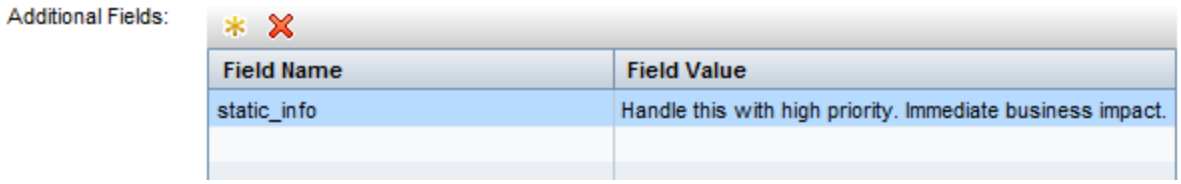
The key field mapping consists of an input field qualifier, an eligible field name, and a mapped field name. To map the key field, map the eligible field name to the mapped field name. The eligible field name is automatically extracted from the input field qualifier.

REST WS policies integrate hierarchical data. Therefore, any field to be mapped must be a leaf node in the internal logical tree-like structure. As a result, the eligible field is the last entry in the tree.

For example, if the input data has the structure `evt/event_type`, the eligible field is `event_type`. This field is then mapped, for example to the field `eventType`.

You can add **additional fields** to data that is sent to Data Forwarding targets. Additional fields are simple key-value pairs and you must manually add both the keys and values. If a defined key field name equals a field name in the mapped data set, it is discarded.

Additional fields are added as immediate children of the highest level element.



The actual field value can consist of user defined strings and data references to the input data (`<$DATA:…>`).

Example:

Additional Field Name	Additional Field Value
combined_text	At `<$DATA:/evt/time_occured>`, `<$DATA:/evt/event_type>` detected an `<$DATA:/evt/event_impact>` impact on system performance. This has happened `<$DATA:/evt/event_counter>` times.

The data references in the additional field `combined_text` are replaced when the policy is run. In the above example, after the variables are replaced, the resulting value in the field `combined_text` is:

```
At 12/05/2015 14:01:39, Monitoring: Threshold violation detected an substantial
impact on system performance. This has happened 8264 times.
```

Note that data references must refer to the *input* format, not the mapped format.

Tasks

How to configure mappings for key fields

This task describes how to map key fields.

1. Create one or more key field mappings.

   If you are working with meta data, drag the table column from the Meta Data list to the Input Field Qualifier column. BSM Connector automatically extracts the eligible field name.

   Alternatively, click above the Input Field Qualifier column and type the qualifier in the input field qualifier.
2. Optionally, change the **Keep all input fields** setting. By default, the option is selected and all fields are kept, regardless whether they are mapped or not. To keep only the mapped fields, deselect the option.
3. Add any additional fields as required.

Related tasks

- "How to Collect Event Data Through the REST Web Service Listener" on page 185
- "How to Collect Metrics Data Through the REST Web Service Listener" on page 186

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Generic Output)" on page 323

Configuring Event Defaults in REST Web Service Listener Policies

The Event Defaults page enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **REST Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Event Attributes page.

Tasks

How to configure event defaults for REST Web service listener policies

This task describes how to configure default settings for all events generated by the policy.

1. Click **Event Attributes** to define default event attributes, such as severity and category.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

2. Click **Event Correlation** to set the type of duplicate event suppression and define the method used to suppress duplicate events.
3. Click **Custom Attributes** to add additional information to all events generated by this policy. For example, you might add a company name, contact information, or a city location to an event.
4. Click **Advanced** to define default advanced attributes such as legacy HPOM attributes and agent MSI (Message Stream Interface) settings.
5. *Optional.* Use the Sample Data tab to drag XML properties (XML elements and attributes) and values to the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

<XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

Note: The Sample Data tab is empty if no sample data is loaded into BSM Connector or no XML data has been received for the policy.

6. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the mappings tab (see also "Configuring Event Defaults in REST Web Service Listener Policies" on the previous page). The default name of the mapping variable is `map<XMLProperty>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box using the following syntax:

   - Map Name list contains the map name of the variable: `map<XMLProperty>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:/<XMLTag>/<XMLProperty>>`

     For example, the custom variable `mapSeverity` has the following input data property: `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

7. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
8. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data Through the REST Web Service Listener" on page 185

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Sample Data Tab - REST Web Service Listener Policies" on page 342
- "Mappings Tab" on page 325

- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metrics Defaults in REST Web Service Listener Policies

The Default Metric Attributes page enables you to assign default values to the metric attributes. The values can be used when defining the policy rules on the Rules tab, and can also be overridden there.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **REST Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Metric Attributes page.

Learn More

This section includes:

- "Metric attributes" below
- "Configuring Metrics Defaults in REST Web Service Listener Policies" above

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

  - Metric name

    Defines the metric name under which the metric appears in the Operations Agent store and consumers.

- Related CI

  Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

- Node

  Used to identify a node-like CI to which the performance records are associated to.

- Value

  The actual performance value which is converted to 64-bit floating point number.

- Time measured

  The time stamp when the value was determined in the third-party system.

- Advanced attributes:
  - Original metric name

    The name of the metric as used on the third-party system.

  - Unit

    The unit of the metric.

  - Integration id

    An id, used to identify the source of the integration.

Tasks

How to configure metrics defaults for REST Web service listener policies

This task describes how to configure metric attribute defaults for all metrics collected by this policy.

1. Define the metric attributes common to all metrics collected by this policy, such as metric class and name. All metrics in the **Basic** tab marked with a ***** are required. **Advanced** attributes are optional.
2. *Optional.* Use the Sample Data tab to drag XML properties (XML elements and attributes) and values to the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

   XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

   <XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

   BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

   Note: The Sample Data tab is empty if no sample data is loaded into BSM Connector or no XML data has been received for the policy.

3. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the mappings tab (see also "Configuring Metrics Defaults in REST Web Service Listener Policies" on page 199). The default name of the mapping variable is `map<XMLProperty>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box using the following syntax:

   - Map Name list contains the map name of the variable: `map<XMLProperty>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:/<XMLTag>/<XMLProperty>>`

     For example, the custom variable `mapSeverity` has the following input data property: `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

4. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string

   - the pattern definition

   - the output string if pattern matches on the input string

   - the output string if the pattern does not match (optional)

   Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

   ```
   $MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)
   ```

   - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

     For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

   Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

5. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.
6. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Metrics Data Through the REST Web Service Listener" on page 186

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Sample Data Tab - REST Web Service Listener Policies" on page 342
- "Mappings Tab" on page 325
- "Operators Tab (Metrics Only)" on page 327
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Event Rules in REST Web Service Listener Policies

Rules define the action a policy should take in response to a specific type of incoming event. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual event data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **REST Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

Rule types

The rule types are:

- **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an event to OMi.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi.

Tasks

How to configure event rules in REST Web service listener policies

This task describes how to configure policy rules.

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific string in the XML data. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the XML properties and values that the policy searches for in the XML data that the policy reads. If the policy finds a match, it may or may not generate an event, depending on the rule type.
   a. Click to create a new condition. New conditions by default use the equals operator.
   b. Click ▸ to expand the new condition.
   c. In the **Property** field, specify the XML element or attribute that the policy searches for. You must specify the XML path from the XML event tag to the property, separated by slash marks (/) (for example, `/PerformanceAlert/Severity`).

      If you are working with sample data, you can drag and drop the XML element or attribute from the XML Properties list to the Properties field.
   d. Select the pattern operator.

      If you select the matches operator, you can type a pattern in the Operand field. For a list of available operators, see "Operator" on page 313.
   e. In the **Operand** field, type the value or pattern that you want the policy to compare with the XML property. If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.
3. Use the **Event Attributes** tab to define event attributes (for example, event title and description) for all events generated by this rule.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.
4. Use the **Event Correlation** tab to set the type of duplicate event suppression and define the method used to suppress duplicate events.
5. Use the **Custom Attributes** tab to add additional information to all events generated by this rule. For example, you might add a company name, contact information, or a city location to an event.
6. Use the **Advanced** tab to define an event drill-down URL, legacy HPOM attributes, and agent MSI (Message Stream Interface) settings.
7. *Optional.* Use the Sample Data tab to drag XML properties (XML elements and attributes) and values to the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

   XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

   <XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

Note: The Sample Data tab is empty if no sample data is loaded into BSM Connector or no XML data has been received for the policy. See also "Configuring the Data Source in REST Web Service Listener Policies " on page 189.

8. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the mappings tab (see also "Configuring Event Rules in REST Web Service Listener Policies" on page 202). The default name of the mapping variable is `map<XMLProperty>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box using the following syntax:

   - Map Name list contains the map name of the variable: `map<XMLProperty>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:/<XMLTag>/<XMLProperty>>`

     For example, the custom variable `mapSeverity` has the following input data property: `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

9. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the event attribute.

10. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).

11. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

    HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data Through the REST Web Service Listener" on page 185
- "How to Collect Metrics Data Through the REST Web Service Listener" on page 186

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "Condition Tab - Rules Only" on page 312
- "Event Attributes Tab" on page 300

- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Sample Data Tab - REST Web Service Listener Policies" on page 342
- "Mappings Tab" on page 325
- "Pattern Matching Variables Tab (Events and Metrics Only)" on page 332
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metrics Rules in REST Web Service Listener Policies

Rules define the action a policy should take in response to a specific type of incoming metric. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual metric data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **REST Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

This section includes:

- "Rule types" below
- "Metric attributes" on the next page
- "Configuring Metrics Rules in REST Web Service Listener Policies" above

Rule types

The rule types are:

- **Store on matched rule.** If matched, BSM Connector stores metrics in a buffer until a maximum number of records or a maximum amount of time is reached. These metrics are sent to OMi when they are requested, and they use the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing.

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

  - Metric name

    Defines the metric name under which the metric appears in the Operations Agent store and consumers.

  - Related CI

    Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

  - Node

    Used to identify a node-like CI to which the performance records are associated to.

  - Value

    The actual performance value which is converted to 64-bit floating point number.

  - Time measured

    The time stamp when the value was determined in the third-party system.

- Advanced attributes:
  - Original metric name

    The name of the metric as used on the third-party system.

  - Unit

The unit of the metric.

- Integration id

  An id, used to identify the source of the integration.

Tasks

How to configure rules for metrics in REST Web service listener policies

This task describes how to configure policy rules.

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific string in the XML file. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the XML properties and values that the policy searches for in the XML file that the policy reads. If the policy finds a match, it may or may not generate a metric record, depending on the rule type.
   a. Click to create a new condition. New conditions by default use the equals operator.
   b. Click ▸ to expand the new condition.
   c. In the **Property** field, specify the XML element or attribute that the policy searches for. You must specify the XML path from the XML metric tag to the property, separated by slash marks (/) (for example, `/PerformanceAlert/Severity`).

      If you are working with sample data, you can drag and drop the XML element or attribute from the XML Properties list to the Properties field.
   d. Select the pattern operator.

      If you select the matches operator, you can type a pattern in the Operand field. For a list of available operators, see "Operator" on page 313.
   e. In the **Operand** field, type the value or pattern that you want the policy to compare with the XML property. If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.
3. *Optional.* If you are creating a rule of the type 'store on matched rule', set the attributes (**Basic** or/and **Advanced**) for metrics that you want the rule to override. All metrics on the **Basic** tab are required. **Advanced** attributes are optional. If default attributes are specified in the Defaults tab, you use the defaults or you can override them as described below.
4. *Optional.* Use the Sample Data tab to drag XML properties (XML elements and attributes) and values to the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

   XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

   <XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

   BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

   Note: The Sample Data tab is empty if no sample data is loaded into BSM Connector or no XML data has been received for the policy. See also "Configuring the Data Source in REST

Web Service Listener Policies " on page 189.

5. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the mappings tab (see also "Configuring Metrics Rules in REST Web Service Listener Policies" on page 205). The default name of the mapping variable is `map<XMLProperty>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box using the following syntax:

   - Map Name list contains the map name of the variable: `map<XMLProperty>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:/<XMLTag>/<XMLProperty>>`

     For example, the custom variable `mapSeverity` has the following input data property: `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

6. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the metric attribute.

7. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string
   - the pattern definition
   - the output string if pattern matches on the input string
   - the output string if the pattern does not match (optional)

   Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

   ```
   $MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)
   ```

   - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

     For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

   Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

8. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

9. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example "<$MSG_NODE>" or "<$MSG_GEN_NODE>", at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Metrics Data from Structured Log Files" on page 235

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "UI Descriptions" on page 299
- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Sample Data Tab - REST Web Service Listener Policies" on page 342
- "Mappings Tab" on page 325
- "Pattern Matching Variables Tab (Events and Metrics Only)" on page 332
- "Operators Tab (Metrics Only)" on page 327
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Options in REST Web Service Listener Policies

The options tab enables you to configure several policy behaviors, for example which events are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **REST Web Service Listener**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **REST Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Options** to open the policy Options page.

Tasks

How to configure options for the REST Web Service listener policies

In the Options page, configure which events or metrics are logged locally, how the policy handles

unmatched events and metrics, and defaults for pattern matching in policy rules.

For more information about the other fields, see "Options Page (Events only)" on page 328.

Related tasks

- "How to Collect Event Data Through the REST Web Service Listener" on page 185
- "How to Collect Metrics Data Through the REST Web Service Listener" on page 186

UI Descriptions

For a description of the Options page, see "Options Page (Events only)" on page 328.

Troubleshooting REST Web Service Listener Policies

This section describes troubleshooting and limitations when working with REST Web Service Listener policies.

General troubleshooting guidelines

- To start troubleshooting, check the HP Operations Agent log files in the `%OvDataDir%log` directory (Windows systems) or the `/var/opt/OV/log` directory (Linux systems).
- To investigate issues related to policy execution, first examine the `%OvDataDir%log\System.txt` file (Windows systems) or the `/var/opt/OV/log/System.txt` file (Linux systems).

  Note: The log files whose names start with the `opr-` prefix are generated by the OMi web console.

Chapter 15: Scheduled Task Policies (Events only)

Scheduled task policies enable you to schedule commands or scripts to run on the BSM Connector system, and will send an event to OMi to indicate the success or failure of the command or script. Use this policy if you want to run commands or scripts on the BSM Connector system once or according to a specific schedule.

How to Schedule Tasks

This section describes how to schedule tasks.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar, then click **Event >** **Scheduled Task**. The scheduled task policy editor opens.

   Alternatively, double-click an existing scheduled task policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Scheduled Task Policy Properties" on the next page.
4. In the **Task** page, specify the command or script that you want to run.

   For details, see "Configuring Tasks in Scheduled Task Policies" on the next page.
5. In the **Schedule** page, configure the schedule according to which the command or script should run.

   Note: If you do not configure a schedule, the command or script runs every minute.

   For details, see "Configuring Schedules in Scheduled Task Policies" on page 214.
6. In the **Start Event**, **Success Event**, and **Failure Event** pages, design the start, success, or failure events that you want to receive.

   For details, see "Configuring Events in Scheduled Task Policies" on page 215.
7. Click **Save and Close** to save the policy and close the editor.
8. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

Scheduled Task Policy User Interface

This section includes:

- "Configuring Scheduled Task Policy Properties" below
- "Configuring Tasks in Scheduled Task Policies" below
- "Configuring Schedules in Scheduled Task Policies" on page 214
- "Configuring Events in Scheduled Task Policies" on page 215

Configuring Scheduled Task Policy Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the policy name and a description of what the policy does.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **Scheduled Task**. The scheduled task policy editor opens.

Alternatively, double-click an existing scheduled task policy to edit it.

Tasks

How to configure policy properties

In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

Related tasks

- "How to Schedule Tasks" on the previous page

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring Tasks in Scheduled Task Policies

In the **Task** page, you specify a command or script that you want to run on the BSM Connector system.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **Scheduled Task**. The scheduled task policy editor opens.

Alternatively, double-click an existing scheduled task policy to edit it.

Click **Task**.

Tasks

How to configure tasks

In **Task type**, select one of the following options:

- Command

  By default, the command runs under the same account as the agent is running, which is Local System or root by default.

  - **Command**: Type the complete path and extension of the command that you want to run on the BSM Connector system (for example, `%OvDataDir%\bin\instrumentation\cleanup.exe`). The file that you specify should exist on the system.
  - **Username**: Type the user name under which the command should be run. The user must exist and have permission to run the command on the system. If you specify a non-existent user, the command fails to run.
  - **Password**: Specify a password for the user. If the password changes, the policy must be updated and reactivated.

- VB Script

  Type the VB script in the window. Alternatively, click **Load VB script file from Client** to load an existing script.

  Tip: Use the policy method `Rule.Status` to specify whether the task is successful. For example, to specify that the task has failed (and trigger a failure message), use `Rule.Status=False`. (See "Rule Object" on page 218 .)

  Note: HP Operations Agent uses a generic Microsoft scripting engine to run VBScript scripts.You can therefore use standard VBScript objects (for example, the FileSystemObject object) in your scripts. Objects that are specific to wscript or cscript (for example, the WScript object) are not supported.

- Perl Script

  Type the Perl script in the window. Alternatively, click **Load VB script file from client** to load an existing script.

  Tip: Use the policy method `$Rule->Status` to specify whether the task is successful. For example, to specify that the task has failed (and trigger a failure message), use `$Rule.Status (False)`. (See "Rule Object" on page 218 .)

  Note: The agent runs as a service that has no standard input, standard output, or standard error. Therefore, the predefined file handles STDIN, STDOUT, and STDERR are not available for Perl

scripts in scheduled task policies. It is also not possible to open file handles that use command pipes or capture the standard output from commands within backticks (`).

Related tasks

- "How to Schedule Tasks" on page 211

UI Descriptions

For a description of the Task page, see "Task Page (Scheduled Task Policies)" on page 363.

Configuring Schedules in Scheduled Task Policies

Scheduled task policies can start commands either once or according to a schedule.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **Scheduled Task**. The scheduled task policy editor opens.

Alternatively, double-click an existing scheduled task policy to edit it.

Click **Schedule**.

Tasks

How to configure schedules

In the Schedule page, select one of the following schedule options:

- **Once.** When **Once** is selected, the command will be run on one specific day at the time you indicate.

  Note: If the selected date or time occurs in the past, the command is not executed, and the Schedule tab shows a warning.

- **Once per interval.** When **Once per interval** is selected, the command will be run once each time the interval that you indicate passes.

  Note: Make sure the minimum value that you set is 15 seconds. If you set less than 15 seconds, the policy can be saved, but you will be notified about the wrong interval value by the scheduled policy editor.

- **Advanced.** When Advanced is selected, you can indicate specific days and times when the command should be run. You select specific days of the week, specific days of the month, and specific months. This allows you to specify odd schedules such as, "On Monday when it falls on the 2nd of the month." You can also indicate that the command should only be run during a specific year.

Note: If you select Advanced but then do not specify a schedule, the command by default runs every minute.

Tip: To select multiple times, click the time, press **Ctrl** or **Shift**, and select additional times.

To select the entire time range, click . To delete a selected time, click .

Related tasks

- "How to Schedule Tasks" on page 211

UI Descriptions

For a description of the Schedule page, see "Schedule Page - Scheduled Task Policies" on page 343.

Configuring Events in Scheduled Task Policies

Scheduled task policies can send a notification event to OMi when a command or script starts, finishes, or fails.You can design the start, success, or failure events that you want to receive by completing the information in the Start Event, Success Event, and Failure Event tabs.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **Scheduled Task**. The scheduled task policy editor opens.

Alternatively, double-click an existing scheduled task policy to edit it.

Click **Start Event**, **Success Event**, or **Failure Event**.

Tasks

How to configure events for scheduled task policies

This section describes how configure events generated by scheduled task policies.

1. Select **Send Start Event**, **Send Success Event**, or **Send Failure Event**.
2. Click **Event Attributes** to define default event attributes, such as severity and category.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.
3. Click **Event Correlation** to set the type of duplicate event suppression and define the method used to suppress duplicate events.
4. Click **Custom Attributes**to add additional information to all events generated by this policy. For example, you might add a company name, contact information, or a city location to an event.

5. Click **Advanced** to define default advanced attributes such as legacy HPOM attributes and agent MSI (Message Stream Interface) settings.
6. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
7. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   HP recommends to surround variables with quotation marks, for example "<$MSG_NODE>" or "<$MSG_GEN_NODE>", at least for those variables whose values can contain space characters.

Related tasks

- "How to Schedule Tasks" on page 211

UI Descriptions

For a description of the Start, Success, and Failure Event pages, see the following sections:

- "Start, Success, Failure Event Pages (Scheduled Task Policies)" on page 360
- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Policy Objects for Scripts

The objects listed here are available for scheduled task policies and can be manipulated with Visual Basic Scripting Edition or with Perl. These policy objects can only be used in scripts that run within a policy. They cannot be used in standalone scripts that are executed at a command prompt.

Caution: Policy scripts provide administrators with a powerful tool to evaluate and manipulate data. If, however, a script is incorrectly written, it could cause the agent to fail. Hewlett-Packard Company is not responsible for agent failures resulting from incorrectly written scripts.

This section includes:

- "Policy Object" on the next page
- "Rule Object" on page 218
- "ConsoleMessage Object" on page 218
- "ExecuteCommand Object" on page 223

Policy Object

This object is used to access the attributes of a policy.

Policy Method:	`CreateObject`
Parameter:	*progID* (string of format: `[Vendor.]Component[.Version]`)
Return Type:	VB Script: IDispatch Perl: not applicable
VB Script Syntax:	`Policy.CreateObject("progID")`
Perl Syntax:	not applicable
Description:	Creates a component instance of a COM object. Note that this method is valid only on Windows nodes, and cannot be used in a Perl script.

Policy Method:	`Execute`
Parameter:	*command* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`Policy.Execute("command")`
Perl Syntax:	`$Policy->Execute("command");`
Description:	Run the specified command asynchronously. The command is executed in the context of agent security, so could be run as Local System or any other user-selected user to run the agent. The method will return immediately. See the ExecuteCommand method Command for more information about how to indicate commands.

Policy Method:	`Output`
Parameter:	*string*
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`Policy.Output("string")`
Perl Syntax:	`$Policy->Output("string");`
Description:	Appends the string to the annotation field of the event sent to BSM in response to the success or failure of a scheduled task.

Policy Method:	`ExecuteEx`
Parameter:	*command* (string)

Policy Method:	`ExecuteEx`
Return Type:	VB Script: BSTR Perl: string
VB Script Syntax:	`Policy.ExecuteEx("command")`
Perl Syntax:	`$Policy->ExecuteEx("command");`

Description:	Run the specified command synchronously and wait for it to complete before returning the output of the command.
	• **Security.** The command is executed in the context of agent security, so could be run as Local System or any other user-selected user to run the agent.
	• **Return values.** If the command is successful, STDOUT is returned. If the command is not successful (return value non-zero), the string "ERROR:\n" followed by STDERR will be returned.
	To handle non-zero return values, run ExecuteEx in an eval function and then check the result, for example for the string ERROR.
	Perl script example:
	`eval '$ReturnText = $ExecuteCommand->ExecuteEx()';`
	`$ReturnText =$@ if $@;`
	• **Paths.** You must use complete paths or ensure that any needed path is included in the PATH variable.
	Example: `dir_con = Policy.ExecuteEx ("cmd /c dir c:\")`

Rule Object

In scheduled task policies, the Rule object is used to indicate whether the command has succeeded or failed. TRUE = command succeeded, FALSE = command failed.

Rule Method:	`Status`
Parameter:	void
Return Type:	VB Script: Boolean Perl: integer
VB Script Syntax:	for put: `Rule.Status = boolvalue` for get: boolvalue = `Rule.Status`
Perl Syntax:	for put: `$Rule.Status(boolvalue);` for get: `boolvalue = $Rule.Status();`
Description:	For scheduled task policies, FALSE indicates that the scheduled task failed.

ConsoleMessage Object

The ConsoleMessage object provides a method for sending events directly to BSM. Events sent in this way are not intercepted by an open message interface policy, but instead are sent directly to the server.

Multiple uses of the Send method are supported. The same script can then send multiple events to OMi depending on which problem it detects.

Note: You cannot use action variables with the ConsoleMessage object.

ConsoleMessage Method:	`Application`
Parameter:	*application* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.Application = "application"`
Perl Syntax:	`$ConsoleMessage->Application("application");`
Description:	This optional method sets the content of **Application** in the event properties.

ConsoleMessage Method:	`Object`
Parameter:	*object* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.Object = "object"`
Perl Syntax:	`$ConsoleMessage->Object("object");`
Description:	This optional method sets the content of **Object** in the event properties.

ConsoleMessage Method:	`MsgText`
Parameter:	*msgtext* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.MsgText = "msgtext"`
Perl Syntax:	`$ConsoleMessage->MsgText("msgtext");`
Description:	This method sets the message text for the event.

ConsoleMessage Method:	`Severity`
Parameter:	*severity* (valid strings are: Unknown|Normal|Warning|Minor|Major|Critical)
Return Type:	VB Script: void Perl: void

ConsoleMessage Method:	`Severity`
VB Script Syntax:	`ConsoleMessage.Severity = "severity"`
Perl Syntax:	`$ConsoleMessage->Severity("severity");`
Description:	Sets the severity of the event that is sent. If not specifically set with this method, the default is Normal. If an invalid string is supplied, severity Unknown will be used.

ConsoleMessage Method:	`MsgGrp`
Parameter:	*messagegroup* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.MsgGrp = "messagegroup"`
Perl Syntax:	`$ConsoleMessage->MsgGrp("messagegroup");`
Description:	Sets the value for the **Message Group** in event properties. If this method does not supply a value, Misc is used.

ConsoleMessage Method:	`Node`
Parameter:	*nodename* (IP address or fully qualified hostname)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.Node = "nodename"`
Perl Syntax:	`$ConsoleMessage->Node("nodename");`
Description:	Sets the value for **Primary Node Name** that will be displayed in the event properties. IP addresses and fully qualified hostnames are valid. If this method does not supply a value, the hostname of the system is used by default.

ConsoleMessage Method:	`ServiceId`
Parameter:	*serviceid* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.ServiceId = "serviceid"`
Perl Syntax:	`$ConsoleMessage->ServiceId("serviceid");`
Description:	This optional method sets the Service ID for the event.

ConsoleMessage Method:	`MessageType`
Parameter:	*messagetype* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.MessageType = "messagetype"`
Perl Syntax:	`$ConsoleMessage->MessageType("messagetype");`
Description:	This optional method sets the value for the **message type** field of the event properties.

ConsoleMessage Method:	`MessageKey`
Parameter:	*messagekey* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.MessageKey = "messagekey"`
Perl Syntax:	`$ConsoleMessage->MessageKey("messagekey");`
Description:	This optional methods sets a key for event correlation.

ConsoleMessage Method:	`AcknowledgeMessageKey`
Parameter:	*messagekey* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.AcknowledgeMessageKey = "messagekey"`
Perl Syntax:	`$ConsoleMessage->AcknowledgeMessageKey("messagekey");`
Description:	This optional method sets the message key to indicate which events are automatically closed.

ConsoleMessage Method:	`TroubleTicket`
Parameter:	*Booleanvalue*
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.TroubleTicket = Booleanvalue`
Perl Syntax:	`$ConsoleMessage->TroubleTicket(Booleanvalue);`
Description:	This optional method specifies if the event is to be sent to a trouble ticket interface. Default is FALSE.

ConsoleMessage Method:	**`Notification`**
Parameter:	*Booleanvalue*
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.Notification = Booleanvalue`
Perl Syntax:	`$ConsoleMessage->Notification(Booleanvalue);`
Description:	This optional method specifies if the event is sent to the notification mechanism. Default is FALSE.

ConsoleMessage Method:	**`AgentMSI`**
Parameter:	*type* (valid strings are: copy|divert|none)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.AgentMSI = "type"`
Perl Syntax:	`$ConsoleMessage->AgentMSI("type");`
Description:	This optional method specifies if the event is to be sent through the message stream interface on the agent. Default (or if string misspelled) is none.

ConsoleMessage Method:	**`ServerMSI`**
Parameter:	*type* (valid strings are: copy|divert|none)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.ServerMSI = "type"`
Perl Syntax:	`$ConsoleMessage->ServerMSI("type");`
Description:	This optional method specifies if event is sent through the event stream interface on the server. Default (or if string misspelled) is none.

ConsoleMessage Method:	**`Send`**
Parameter:	void
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.Send()`
Perl Syntax:	`$ConsoleMessage->Send();`

ConsoleMessage Method:	**Send**
Description:	This method sends the event to the OMi server. The MsgText method must set the message text before using this method. Multiple uses of the Send method are supported. Policy variables will not be expanded.

ExecuteCommand Object

Object used for requesting a command to be run. It starts a command to be run by the HP Operations Agent.

ExecuteCommand Method:	**Command**
Parameter:	*command* (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ExecuteCommand.Command = "command"`
Perl Syntax:	`$ExecuteCommand->Command("command");`
Description:	This mandatory method is the name of the command to run with all necessary parameters. **Note:** For scripts that will run on Windows systems, internal commands such as Copy, Rename, and DIR use a command interpreter that must be started before the command can be run. For commands of this type, the command must be preceded with `cmd /k`, followed by any other parameters required.

ExecuteCommand Method:	**KillonTimeout**
Parameter:	*seconds* (integer)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ExecuteCommand.KillonTimeout = seconds;`
Perl Syntax:	`$ExecuteCommand->KillonTimeout(seconds);`
Description:	This method sets the maximum time, in seconds, that the command will run. The default is unlimited. Valid only with the StartEx method.

ExecuteCommand Method:	**UserName**
Parameter:	username (string)

ExecuteCommand Method:	`UserName`
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ExecuteCommand.UserName = "username"`
Perl Syntax:	`$ExecuteCommand->UserName("username");`
Description:	User name under which the command should be run. Optional, default is $AGENT_USER.

ExecuteCommand Method:	`Password`
Parameter:	password (string)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ExecuteCommand.Password = "password"`
Perl Syntax:	`$ExecuteCommand->Password("password");`

ExecuteCommand Method:	`Password`
Description:	Password for accessing the specified user account. To prevent the password from being visible in the script, use the following instructions: 1. Open a command prompt. 2. Change directory to the agent install directory: `<install_dir>/bin/<arch>/OpC/install` 3. Encrypt your password with the command: `opcpwcrpt <yourpassword>` 4. Use the output string as the password in your script. In some cases it is better not to supply a password. Should I provide the password or not? Executing the command without the password is the easier of the two methods, but it has some restrictions that make it unsuitable in some situations. The lists below show the restrictions and advantages of both methods. **Without a password:** • For Windows systems, resources accessed through the network are not available. • For all systems, changed passwords do not invalidate the policy. **With a password:** • For all systems, resources accessed through the network are available. • For all systems, the encrypted password is sent over the network. • For all systems, if the password changes, the policy must be updated and redeployed.

ExecuteCommand Method:	`Start`
Parameter:	void
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ExecuteCommand.Start()`
Perl Syntax:	`$ExecuteCommand->Start();`
Description:	Run the command specified by ExecuteCommand.Command and return immediately the control to the script so the next lines can be processed right away.

ExecuteCommand Method:	`StartEx`
Parameter:	void
Return Type:	VB Script: BSTR Perl: String
VB Script Syntax:	`ExecuteCommand.StartEx`
Perl Syntax:	`$ExecuteCommand->StartEx();`

Description:	Run ExecuteCommand.Command and wait until it finishes. Commands can be run synchronously or asynchronously, as needed. Multiple uses of the Start method are supported. This way, the same script can trigger multiple external commands.
	
	If the command is successful, STDOUT is returned. If the command is not successful (return value non-zero), the string "ERROR:\n" followed by STDERR will be returned.
	
	To handle non-zero return values, run StartEx in an eval function and then check the result, for example for the string ERROR.
	
	Perl script example:
	
	`eval '$ReturnText = $ExecuteCommand->StartEx()';`
	`$ReturnText =$@ if $@;`

Chapter 16: SNMP Interceptor Policies (Events only)

SNMP trap policies configure HP Operations Agent to watch for SNMP traps received by BSM Connector from third-party systems. The third-party system must be configured to send traps to the BSM Connector server. The SNMP trap policy reads SNMP traps, and responds when a character pattern that you choose is found in an SNMP trap.

SNMP trap policies collect data from any SNMP trap (version 1 and 2) received by BSM Connector, and send events to OMi containing preferred values from the original SNMP trap.

Before setting up an SNMP trap policy, you should be clear about the purpose and usage of the data in OMi.

Event data that is forwarded to OMi is controlled by policy rules. Policy rules consist of a condition and of settings for the event generated by the policy. The condition uses regular expressions to match the source data. The settings enable you to configure the event that BSM Connector sends to OMi.

Note: You can use this policy type to forward information from HP Network Node Manager i (NNMi) to OMi. When you install BSM Connector on an NNMi management server, you must enable the HP Operations Agent integration with NNMi so that the HP Operations Agent can receive SNMP traps from the NNMi northbound interface. To enable the NNMi integration, follow the procedures in the *NNMi Deployment Reference* instead of the one below.

Configuring the Trap Interceptor (opctrapi) Process

By default, the trap interceptor process (opctrapi) of HP Operations Agent uses the Net-SNMP APIs to receive SNMPv1 and SNMPv2 traps at port 162. If port 162 is already bound by another process (for example the Microsoft SNMP Trap service or the Linux snmptrapd process), the trap interceptor process fails to start.

You can reconfigure the trap interceptor process to listen at another port by setting the SNMP_TRAP_PORT variable. Alternatively, for Windows systems, you can configure the trap interceptor process to subscribe to the Microsoft SNMP Trap service. This configuration provides the trap interceptor process with SNMPv1 traps only.

To configure HP Operations Agent for the Microsoft SNMP Trap service, complete the following steps:

1. Open a command prompt, and then type:

   ```
   ovconfchg -ns eaagt -set SNMP_SESSION_MODE WIN_SNMP
   ```

2. Restart the trap interceptor process:

   ```
   ovc -restart opctrapi
   ```

For more information about the `SNMP_SESSION_MODE` and `SNMP_TRAP_PORT` variables, see the HP Operations Agent Reference Guide.

Troubleshooting

- If on Windows, the trap interceptor process does not receive SNMPv2 traps, make sure the Microsoft SNMP Trap service is disabled, and the SNMP_SESSION_MODE variable is set to NETSNMP.
- To check that BSM Connector receives SNMP traps, complete the following steps:
  a. Edit the SNMP trap policy and open the Options page.
  b. In the Options page, select one or more of the **Log Local Events** options.
  c. Reactivate the SNMP trap policy.
  d. Open the following log file and check that SNMP trap events are logged:

     Windows: `%OvDataDir%\log\OpC\opcmsglg`

     Linux: `/var/opt/OV/log/OpC/opcmsglg`

  For details on logging local events, see "Configuring Options in SNMP Trap Policies" on page 233.

How to Collect Event Data from SNMP Traps

This task describes how to collect event data from SNMP traps.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar, then click **Event >** **SNMP Interceptor**. The SNMP Trap Policy editor opens.

   Alternatively, double-click an existing SNMP trap policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring SNMP Trap Policy Properties" on the next page.
4. *Optional.* In the **Default Event Attributes** page, configure the default settings for all events generated by the policy (for example, default event correlation settings).

   For details, see "Configuring Event Defaults in SNMP Trap Policies" on the next page.
5. In the **Rules** page, define what the policy should do in response to a specific type of event.

   For details, see "Configuring Rules in SNMP Trap Policies" on page 231.
6. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in SNMP Trap Policies" on page 233.
7. Click **Save and Close** to save the policy and close the editor.
8. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.
9. Activate the SNMP trap policy to start the opctrapi process on the BSM Connector server.

SNMP Trap Policy User Interface

This section includes:

- "Configuring SNMP Trap Policy Properties" below
- "Configuring Event Defaults in SNMP Trap Policies" below
- "Configuring Rules in SNMP Trap Policies" on page 231
- "Configuring Options in SNMP Trap Policies" on page 233

Configuring SNMP Trap Policy Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the policy name and a description of what the policy does.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **SNMP Interceptor**. The SNMP Trap Policy editor opens.

Alternatively, double-click an existing SNMP trap policy to edit it.

Tasks

How to configure policy properties

In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

Related tasks

- "How to Collect Event Data from SNMP Traps" on the previous page

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring Event Defaults in SNMP Trap Policies

The Default Event Attributes page enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **SNMP Interceptor**. The SNMP Trap Policy editor opens.

Alternatively, double-click an existing SNMP trap policy to edit it.

Click **Defaults** to open the Default Event Attributes page.

Tasks

Configure events for SNMP trap policies

This task describes how to configure default settings for all events generated by the policy.

1. Click **Event Attributes** to define default event attributes, such as severity and category.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

2. Click **Event Correlation** to specify the Event Key and Close Events with Key. Additionally you can suppress deduplication on server by selecting the available check box.
3. Click **Advanced** to define default advanced attributes such as legacy HPOM attributes and agent MSI (Message Stream Interface) settings.
4. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
5. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from SNMP Traps" on page 228

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Advanced Tab" on page 310
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Rules in SNMP Trap Policies

Rules define the action a policy should take in response to a specific type of incoming event. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual event data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **SNMP Interceptor**. The SNMP Trap Policy editor opens.

Alternatively, double-click an existing SNMP trap policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

Rule types

The rule types are:

- **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an event to OMi.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi.

Tasks

How to configure rules in SNMP trap policies

This task describes how to configure policy rules.

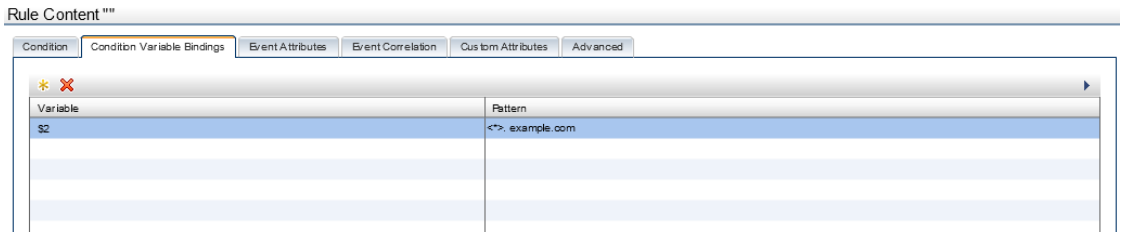
1. In the **Policy Rules** section, click in the toolbar and select the rule type. Then type a description for the rule. After a rule has been added, you can change the rule type by clicking the current rule type in the list of rules and selecting another rule type from the drop-down list.

   Alternatively, select an existing rule and click to copy the rule. You can then rewrite the description of the copied rule and edit the rule.
2. In the **Rule Content** section, use the **Condition** tab to define the match condition for the SNMP

traps. You can match SNMP traps generated from a specific node, or SNMP traps with specific IDs.

3. In the **Condition Variable Bindings** tab, select the variable bindings you want the policy to read, and write one or more match patterns for each binding. You can use pattern-matching rules when matching variable bindings.

   For example, $2 contains in many SNMP events the hostname of the sender of the SNMP event. To only match events from systems in the domain example.com, use the pattern `<*>.example.com`:



4. Use the **Event Attributes** tab to define event attributes (for example, event title and description) for all events generated by this rule.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

5. Use the **Event Correlation** tab to set the type of duplicate event suppression and define the method used to suppress duplicate events.
6. Use the **Custom Attributes** tab to add additional information to all events generated by this rule. For example, you might add a company name, contact information, or a city location to an event.
7. Use the **Advanced** tab to define an event drill-down URL, legacy HPOM attributes, and agent MSI (Message Stream Interface) settings.

Related tasks

- "How to Collect Event Data from SNMP Traps" on page 228

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "Condition Definition Tab - SNMP Interceptor Policy Rules Only" on page 314
- "Condition Variable Bindings Tab - SNMP Interceptor Policy Rules Only" on page 316
- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310

Configuring Options in SNMP Trap Policies

The options tab enables you to configure several policy behaviors, for example which events are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

To access

In the BSM Connector user interface, click in the toolbar, then click **Event >** **SNMP Interceptor**. The SNMP Trap Policy editor opens.

Alternatively, double-click an existing SNMP trap policy to edit it.

Click **Options** to open the policy Options page.

Tasks

How to configure options for SNMP trap policies

In the Options page, configure which events are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

For more information about the other fields, see "Configuring Options in SNMP Trap Policies" above.

Related tasks

- "How to Collect Event Data from SNMP Traps" on page 228

UI Descriptions

For a description of the Options page, see "Options Page (Events only)" on page 328.

Chapter 17: Structured Log File Policies (Events, Metrics, and Generic Output)

This policy type reads entries in a log file and matches the desired text against the log file structure fields defined by using the OM pattern-matching language. When the policy finds a match and certain conditions apply, the policy responds by sending event, metrics, or generic output data to OMi. You can define the details of the data sent to OMi based on information in the log file.

Note: You must have the format and syntax of the log file that the policy reads. You must define a log file structure to match on the entries in the log file that contain the data you want to read and forward to OMi. For examples of how the log file structure is defined, see "Defining a log file structure by using OM pattern-matching language" on page 238.

What Data Is Collected

The structured log file policy sends to BSM Connector data that is extracted from any log file row that matched against the log file structure fields defined by OM pattern-matching language.

Before setting up the structured log file policy, you should be clear about the purpose and usage of the data in OMi.

The specific data that is forwarded to OMi (for events), stored on the BSM Connector system (for metrics), or forwarded to a consumer application (generic output) is controlled by the type of policy (event, metrics, or generic output policy) and by the defaults and rules that the policy applies to the data (events and metrics only).

How to Collect Event Data from Structured Log Files

This task describes how to collect event data from structured log files.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Structured Log File**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Structured Log File Policy Properties" on page 237.
4. In the **Source** page, define the structure pattern of the log file by using the OM pattern-matching language.

For details, see "Configuring Data Source in Structured Log File Policies" on page 238.

5. In the **Mappings** page, map the structured log file's input data properties to custom variables.

   For details, see "Configuring Mappings in Structured Log File Policies (Event and Metrics Only) " on page 242.

6. *Optional.* In the **Defaults** page, configure the default settings for all events generated by the policy (for example, default event correlation settings).

   For details, see "Configuring Event Defaults in Structured Log File Policies" on page 246.

7. In the **Rules** page, define what the policy should do in response to a specific type of event.

   For details, see "Configuring Event Rules in Structured Log File Policies" on page 251.

8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in Structured Log File Policies" on page 258.

9. Click **Save and Close** to save the policy and close the editor.
10. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Metrics Data from Structured Log Files

This task describes how to collect metrics from structured log files.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **Structured Log File**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Structured Log File Policy Properties" on page 237.
4. In the **Source** page, define the structure pattern of the log file by using either the OM pattern-matching language or static fields. Additionally define the possible recurring fields.

   For details, see "Configuring Data Source in Structured Log File Policies" on page 238.
5. In the **Mappings** page, map the structured log file's input data properties to custom variables.

   For details, see "Configuring Mappings in Structured Log File Policies (Event and Metrics Only) " on page 242.
6. *Optional.* In the **Defaults** page, assign default values to the metric attributes.

   For details, see "Configuring Metrics Defaults in Structured Log File Policies" on page 248.

7. In the **Rules** page, define what the policy should do in response to a specific metric.

   For details, see "Configuring Metrics Rules in Structured Log File Policies" on page 254.
8. Click **Save and Close** to save the policy and close the editor.
9. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Generic Output Data from Structured Log Files

This task describes how to collect generic output data from structured log files.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Structured Log File**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring Structured Log File Policy Properties" on the next page.
4. In the **Source** page, define the structure pattern of the log file by using the OM pattern-matching language.

   For details, see "Configuring Data Source in Structured Log File Policies" on page 238.
5. The Mappings page enables you to map the log file input data properties, which are the structural fields extracted from log files, to custom variables. For details, see "Configuring Mappings in Structured Log File Policies (Generic Output Only) " on page 244
6. Click **Save and Close** to save the policy and close the editor.
7. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

Structured Log File Policy User Interface

Structured log file policies display different pages depending on the type of data they are integrating and on the policy origin.

Choose the type of data you want to integrate and then complete the following tasks:

Event data

- "Configuring Structured Log File Policy Properties" on the next page

- "Configuring Data Source in Structured Log File Policies" on the next page
- "Configuring Mappings in Structured Log File Policies (Event and Metrics Only) " on page 242
- "Configuring Event Defaults in Structured Log File Policies" on page 246
- "Configuring Event Rules in Structured Log File Policies" on page 251
- "Configuring Options in Structured Log File Policies" on page 258

Metrics data

- "Configuring Structured Log File Policy Properties" below
- "Configuring Data Source in Structured Log File Policies" on the next page
- "Configuring Mappings in Structured Log File Policies (Event and Metrics Only) " on page 242
- "Configuring Metrics Defaults in Structured Log File Policies" on page 248
- "Configuring Metrics Rules in Structured Log File Policies" on page 254
- "Configuring Options in Structured Log File Policies" on page 258

Generic output data

- "Configuring Structured Log File Policy Properties" below
- "Configuring Data Source in Structured Log File Policies" on the next page
- "Configuring Mappings in Structured Log File Policies (Generic Output Only) " on page 244

Configuring Structured Log File Policy Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the policy name and a description of what the policy does.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Structured Log File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **Structured Log File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

Tasks

How to configure policy properties

In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

For more information about the other fields, see "Configuring Structured Log File Policy Properties" above.

Related tasks

- "How to Collect Event Data from Structured Log Files" on page 234
- "How to Collect Metrics Data from Structured Log Files" on page 235
- "How to Collect Generic Output Data from Structured Log Files" on page 236

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring Data Source in Structured Log File Policies

The source page of the structured log file policy editor enables you to specify which log file the policy reads. You can also configure the policy to extract data, to which the log file structure is applied, from the log file. The policy retains that structured data for later reuse in other pages of the policy editor.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Structured Log File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **Structured Log File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Source** to open the policy Source page.

Learn More

Defining a log file structure by using OM pattern-matching language

A log file structure is defined by using the OM pattern-matching language, so that the dynamic parts of the text-based events can be extracted from any log file row, assigned to variables, and then used as parameters to build the event description or to set other attributes. For more information on the OM pattern matching in policy rules, see "Pattern Matching in Policy Rules" on page 284.

You can use an asterisk (<*>) wildcard in the **Log File Path / Name** field to match multiple file names. For example, to match the source file names `events.1.log` and `events.2.log`, use the pattern `<path>/events<*>.xml` in the Log File Path / Name field. Note that the <*> wildcard is the only supported OM pattern in log file paths. For more information on pattern matching, see "Pattern-Matching Details" on page 284.

Example 1: Use OM pattern-matching language to extract the log file structure from the following log file line:

```
Mon, 28 Jul 2014 23:19:29 GMT;SEVERE;frogproc;123456;ERR-123;failed connect to
db ‘pond’
```

This is done by defining the fields from which the log file line is logically constituted, and allocating the correspondent variables by which these fields can be organized within a structure. The log file line in this example is constituted logically from the following fields:

```
 timestamp;severity;processname;pid;errorcode;errortext
```

Allocate the appropriate variable extractions to all fields by using the OM pattern-matching language, as follows:

```
<*.timestamp>;<*.severity>;<*.processname>;<*.pid>;<*.errorcode>;<*.errortext>
```

Now each field from the log file line can be identified by the variable name, that can be also used in all subsequent policy operations, such as mappings, default attributes and rules.

For example, when setting the `Title` field of the event attribute with the value of the `errortext` field, you should enter `<$DATA:errortext>` in the `Title` field of the Event Attributes tab of the editor, or you could just drag the `errortext` property from the Sample Data tab to the `Title` field .

In the Rules tab the field is simply referred to as `errortext` in the `Property` field.

Defining a log file structure by using static fields (Metric only)

In addition to defining a log file structure by using the OM pattern-matching language, you can identify a log file structure by using the static fields.

Example 2: Static fields are actually word lists of nonrecurring data from the log file separated by comma. In case only one metric per line is present, all fields can be addressed. For example:

Use static fields to extract the log file structure from the following log file line:

```
1380004749|tcpc113.RIESLING.INTERN|LogicalDisk|C:|% Free
Space|66.379264831543|Microsoft.Windows.Server.2008.LogicalDisk
```

This is done by defining the fields from which the log file line is logically constituted, and by using these fields as static fields with the defined field separator char. The log file line in this example is constituted logically from the following fields:

```
timestamp|hostname|entitytype|entityid|countername|countervalue|scomtype
```

The corresponding static fields should be entered as follows:

```
timestamp,hostname,entitytype,entityid,countername,countervalue,scomtype
```

Field separator char is a pipe symbol (|).

Note that the static fields require comma instead of the pipe symbol as a delimiter.

Note: This is actually the recommended method, due to performance reasons.

Using recurring fields in defining a log file structure (Metric only)

The "Recurring fields" configuration parameter is useful in case when more than one performance value is present within a single log file line. This is actually a word list that contains the recurring part from the

log line. Each recurrence creates a record in the store.

Example 3: Extract the log file structure from the following log file lines by using also the reccuring fields:

```
1380004749|tcpc113.RIESLING.INTERN|LogicalDisk|C:|% Free
Space|66.379264831543|Current Disk Queue Length|0|Avg. Disk
sec/Transfer|0.000484383897855878
```

```
1380004748|tcpc113.RIESLING.INTERN|Network Interface|10|Bytes
Total/sec|55230.0703125|Current Bandwidth|1000000000
```

This is done by defining the fields from which the log file line is logically constituted, and then identifying which of them can be addressed as static fields, and which can be described as a variable part that consists of an arbitrary number of countername-countervalue pairs. These are the recurring fields. The log file lines in this example are constituted logically from the following fields:

```
timestamp|hostname|entitytype|entityid|countername_1|countervalue_1|countername_
2|countervalue_2|countername_3|countervalue_3
```

```
timestamp|hostname|entitytype|entityid|countername_1|countervalue_1|countername_
2|countervalue_2
```

The corresponding static fields should be entered as follows:

```
timestamp,hostname,entitytype,entityid
```

In addition, enter the following recurring fields:

```
countername,countervalue
```

Field separator char is a pipe symbol (`|`).

Static fields can also be specified by using the OM pattern-matching language. However, this is not the recommended method, because of the performance reasons. The syntax is as follows:

```
 <*.timestamp>\|<*.hostname>\|<*.entitytype>\|<*.entityid>
```

Setting the Line Start Indicator (Event only)

Line start indicator enables you to differentiate the structured log file entries based on their logical relationship, regardless of their span in the log file. When the log entry that represents a single logical unit has a span of more than one line in the log file, you can easily differentiate it from other log entries by identifying a line start indicator from a log file, and then specifying the matched line start pattern by using the OM pattern matching language.

For example, the following `tomcat.log` file excerpt contains four logically separated log entries that are expanded over multiple log lines, however all of them start with timestamp (May 19, 2015 2:39:01 PM):

```
May 19, 2015 2:39:01 PM org.apache.catalina.core.StandardService initInternal
SEVERE: Failed to initialize connector [Connector[HTTP/1.1-30000]]
org.apache.catalina.LifecycleException: Failed to initialize component [Connector
[HTTP/1.1-30000]]
       at org.apache.catalina.util.LifecycleBase.init(LifecycleBase.java:106)
       at org.apache.catalina.core.StandardService.initInternal
```

```
(StandardService.java:559)
       at org.apache.catalina.util.LifecycleBase.init(LifecycleBase.java:102)
       at org.apache.catalina.core.StandardServer.initInternal
(StandardServer.java:821)
       at org.apache.catalina.util.LifecycleBase.init(LifecycleBase.java:102)
       at org.apache.catalina.util.LifecycleBase.init(LifecycleBase.java:102)
       ... 12 more

May 19, 2015 2:39:01 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 3622 ms
May 19, 2015 2:39:01 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Catalina
May 19, 2015 2:39:01 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: HP OpenView TomcatB
```

Therefore, in this case, the line start indicator must match the timestamp pattern, as follows:

```
<*> <#>, <4#> <#>:<#>:<#> <2*>
```

In this instance,the following applies:

Log text	May	19	2015	2	39	01	PM
OM Pattern	<*>	<#>	<4#>	<#>	<#>	<#>	<2*>
Description	Matches string	Matches digit(s)	Matches 4 digit(s)	Matches digit(s)	Matches digit(s)	Matches digit(s)	Matches string with length 2

Note: The punctuation marks and spaces in the line start pattern represent the static strings derived from the log file text.

Tasks

How to configure the structured log file source

This task describes how to configure the structured log file source file and how the policy reads it.

1. Type the full path to the log file on the BSM Connector system.
2. Click to load a sample log file. You can load a sample file from the BSM Connector system or from the system where the Web browser runs.

   When you load sample data, BSM Connector replaces already loaded data with the new data. This does not affect any mappings that are defined based on previously available sample data.
3. *For the Event integration only:*
   - In the Logfile Structure field, enter an OM pattern by which the log file will be structured. You can see the newly structured data on the Structured data tab of the Structured logfile sample data window.

- In the Line Start Indicator field, enter a line start pattern that matches the line start indicator from the log file. For more information on how this pattern is defined, see "Setting the Line Start Indicator (Event only)" on page 240

For the Metric integration only: In the Logfile Structure field:

a. Choose the data field by which the log file structure will be identified. You can choose either to identify by using OM pattern, or by using static fields.

b. Define the recurring fields in the log file structure.

c. Enter the field separator.

Related tasks

- "How to Collect Event Data from Structured Log Files" on page 234
- "How to Collect Metrics Data from Structured Log Files" on page 235
- "How to Collect Generic Output Data from Structured Log Files" on page 236

UI Descriptions

For a description of the Source page, see "Source Page - Structured Log File Policies (Events and Metrics Only)" on page 347.

Configuring Mappings in Structured Log File Policies (Event and Metrics Only)

The Mappings page enables you to map the log file input data properties, which are the structural fields extracted from log files, to custom variables.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Structured Log File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

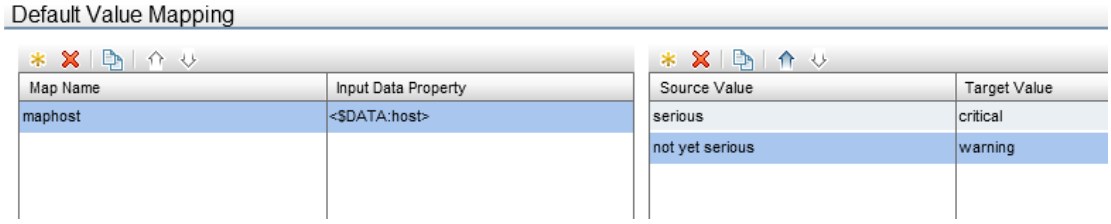
Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

A custom variable consists of a map name, a field extracted from the log file by using the OM pattern matching language, and one or more source and target value pairs. For example, you can assign the pattern matching field `host` to the map name `maphost`, and add a source value of `critical`. You can then assign the target value `serious` to the variable so that BSM Connector inserts the value `critical`

into the event in all places where the variable is used and the source value is `serious` in the log file.



Input data references use the following syntax: <$DATA:<*InputReferenceField*>>

Field values originate from the structured log file pattern settings defined for the data source.

For example, the custom variable `maphost` has the pattern matching field `host` assigned.

Assigning a pattern matching field to a map name is optional. If you do not assign a pattern matching field to a variable, you must add the source value directly to the variable when you insert the variable in an event attribute.

The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match the structured log file pattern.

The Sample Data tab shows the following information, if sample data is available:

- Input Data Properties

  If sample data is available, the Input Data Properties section shows all fields that match the structured log file pattern.

  The items in the Input Data Properties section are by default sorted alphabetically in ascending order.
- Values for 'host'

  This section displays the values of a field selected in the Input Data Properties section. If a value appears more than once, click to show or hide duplicate values. To find values that belong to more than one group, select the value and click . The Sample Data window opens and shows all fields that have the selected value.

When you drag a field from the Input Data Properties list and drop it on the Default Value Mapping List, BSM Connector automatically adds the default prefix `map` to the map name and inserts the pattern matching field. You can then drag one or more structured log file source values from the values list and drop them on the Source Value list. You then finally only have to type the target values.

Tasks

How to configure structured log file mappings

This task describes how to map pattern matching fields to custom variables.

1. Create one or more custom variables.

   If you are working with sample data, drag the field from the Input Data Properties list to the Map Name column. BSM Connector automatically adds the default prefix `map` to the map name and inserts the group name.

Alternatively, click above the Map Name column and type the variable name in the map name field. Fields are optional. If you do not assign a field to a variable, you must add the source value directly to the variable when you insert the variable in an event attribute.

2. Add source and target value pairs to each custom variable.
   - If sample data is loaded in BSM Connector, drag a value from the Values for '...' list to the Source Value column, and then type the target value in the corresponding field.

     Alternatively, cClick above the Source Value column and type the source and target values in the corresponding fields.

   - *Optional.* In the Indicators tab, add indicators to the source or target value fields. After loading the indicators from the OMi server, the Indicators tab shows a hierarchy of configuration item types.

     To insert an indicator in a source or target value field, drag the indicator state (for example, `HTTPServer:Normal`) from the Indicators tab and drop it on the corresponding field.

   - *Optional.* In the Policy Variables tab, add policy variables to event or metric attributes. BSM Connector replaces the variables with the appropriate values in the generated event or metric.

     HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from Structured Log Files" on page 234

UI Descriptions

For a description of the Mappings page, see the following sections:

- "UI Descriptions" on page 299
- "Sample Data Tab - Structured Log File Policies" on page 341
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Mappings in Structured Log File Policies (Generic Output Only)

The Mappings page enables you to map the log file input data properties, which are the structural fields extracted from log files, to custom variables.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

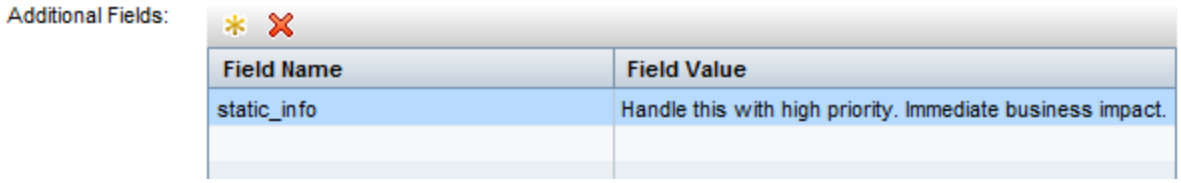
The key field mapping consists of an input field qualifier, an eligible field name, and a mapped field name. To map the key field, map the eligible field name to the mapped field name. The eligible field name is automatically extracted from the input field qualifier.

Log policies integrate hierarchical data. Therefore, any field to be mapped must be a leaf node in the internal logical tree-like structure. As a result, the eligible field is the last entry in the tree.

For example, if the input data has the structure `evt/event_type`, the eligible field is `event_type`. This field is then mapped, for example to the field `eventType`.

You can add **additional fields** to data that is sent to Data Forwarding targets. Additional fields are simple key-value pairs and you must manually add both the keys and values. If a defined key field name equals a field name in the mapped data set, it is discarded.

Additional fields are added as immediate children of the highest level element.



The actual field value can consist of user defined strings and data references to the input data (`<$DATA:…>`).

Example:

Additional Field Name	Additional Field Value
combined_text	At `<$DATA:/evt/time_occured>`, `<$DATA:/evt/event_type>` detected an `<$DATA:/evt/event_impact>` impact on system performance. This has happened `<$DATA:/evt/event_counter>` times.

The data references in the additional field `combined_text` are replaced when the policy is run. In the above example, after the variables are replaced, the resulting value in the field `combined_text` is:

```
At 12/05/2015 14:01:39, Monitoring: Threshold violation detected an substantial
impact on system performance. This has happened 8264 times.
```

Note that data references must refer to the *input* format, not the mapped format.

Tasks

How to configure mappings for key fields

This task describes how to map key fields.

1. Create one or more key field mappings.

   If you are working with meta data, drag the table column from the Meta Data list to the Input Field Qualifier column. BSM Connector automatically extracts the eligible field name.

   Alternatively, click above the Input Field Qualifier column and type the qualifier in the input field qualifier.
2. Optionally, change the **Keep all input fields** setting. By default, the option is selected and all fields are kept, regardless whether they are mapped or not. To keep only the mapped fields, deselect the option.
3. Add any additional fields as required.

Related tasks

- "How to Collect Generic Output Data from Structured Log Files" on page 236

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Generic Output)" on page 323

Configuring Event Defaults in Structured Log File Policies

The Defaults page enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Event Attributes page.

Tasks

How to configure events for structured log file policies

This task describes how to configure default settings for all events generated by the policy.

1. Click **Event Attributes** to define default event attributes, such as severity and category.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

2. Click **Event Correlation** to set the type of duplicate event suppression and define the method used to suppress duplicate events.
3. Click **Custom Attributes** to add additional information to all events generated by this policy. For example, you might add a company name, contact information, or a city location to an event.
4. Click **Advanced** to define default advanced attributes such as legacy HPOM attributes and agent MSI (Message Stream Interface) settings.
5. *Optional.* Use the Sample Data tab to drag the input data references to the attribute boxes. Alternatively, you can type the reference directly into the attribute box.

   Input data references use the following syntax: <$DATA:<*InputReferenceField*>>.

   Field names originate from the structured log file pattern settings defined for the data source.

   BSM Connector replaces the pattern matching field at runtime with the value of the specified field. If you insert a field value, the value will be used.

   Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match the structured log file pattern specified in the source page. See also "Configuring Data Source in Structured Log File Policies" on page 238.

6. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings page (see also "Configuring Mappings in Structured Log File Policies (Event and Metrics Only) " on page 242).

   Alternatively, type the custom variable into the attribute box using the following syntax:

   <$MAP(<*CustomVariable*>)> where <*CustomVariable*> is the map name of the variable (for example, `<$MAP(maphost)>`).

   If the custom variable does not have a input data reference assigned, use the following syntax:

   <$MAP(<*CustomVariable*>,<*SourceValue*>) where <*SourceValue*> can be one of the following:

   - Name of the input data reference, for example `<$MAP(maphost)>,<$DATA:host>`
   - The source value itself, for example `<$MAP(maphost,Critical)>`

7. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
8. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   HP recommends to surround variables with quotation marks, for example "`<$MSG_NODE>`" or "`<$MSG_GEN_NODE>`", at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from Structured Log Files" on page 234

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Sample Data Tab - Structured Log File Policies" on page 341
- "Mappings Tab" on page 325
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metrics Defaults in Structured Log File Policies

The Default Metric Attributes page enables you to assign default values to the metric attributes. The values can be used when defining the policy rules on the Rules tab, and can also be overridden there.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Metric Attributes page.

Learn More

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

  - Metric name

    Defines the metric name under which the metric appears in the Operations Agent store and consumers.

  - Related CI

    Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

  - Node

    Used to identify a node-like CI to which the performance records are associated to.

  - Value

    The actual performance value which is converted to 64-bit floating point number.

  - Time measured

    The time stamp when the value was determined in the third-party system.

- Advanced attributes:
  - Original metric name

    The name of the metric as used on the third-party system.

  - Unit

    The unit of the metric.

  - Integration id

    An id, used to identify the source of the integration.

Tasks

How to configure metrics defaults for structured log file policies

This task describes how to configure metric attribute defaults for all metrics collected by this policy.

1. Define the metric attributes common to all metrics collected by this policy, such as metric class and name. All metrics in the **Basic** tab marked with a * are required. **Advanced** attributes are

optional.

2. *Optional.* Use the Sample Data tab to drag the input data references to the attribute boxes. Alternatively, you can type the reference directly into the attribute box.

   Input data references use the following syntax: <$DATA:<*InputReferenceField*>>.

   Field names originate from the structured log file pattern settings defined for the data source.

   BSM Connector replaces the pattern matching field at runtime with the value of the specified field. If you insert a field value, the value will be used.

   Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match the structured log file pattern specified in the source page. See also "Configuring Data Source in Structured Log File Policies" on page 238.

3. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings page (see also "Configuring Mappings in Structured Log File Policies (Event and Metrics Only) " on page 242).

   Alternatively, type the custom variable into the attribute box using the following syntax:

   <$MAP(<*CustomVariable*>)> where <*CustomVariable*> is the map name of the variable (for example, `<$MAP(maphost)>`).

   If the custom variable does not have a input data reference assigned, use the following syntax:

   <$MAP(<*CustomVariable*>,<*SourceValue*>) where <*SourceValue*> can be one of the following:

   - Name of the input data reference, for example `<$MAP(maphost)>,<$DATA:host>`
   - The source value itself, for example `<$MAP(maphost,Critical)>`

4. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string

   - the pattern definition

   - the output string if pattern matches on the input string

   - the output string if the pattern does not match (optional)

   Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

   `$MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)`

   - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

     For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

   Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

5. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.
6. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example "<$MSG_NODE>" or "<$MSG_GEN_NODE>", at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Metrics Data from Structured Log Files" on page 235

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Sample Data Tab - Structured Log File Policies" on page 341
- "Mappings Tab" on page 325
- "Operators Tab (Metrics Only)" on page 327
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Event Rules in Structured Log File Policies

Rules define the action a policy should take in response to a specific type of incoming event. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual event data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

Rule types

The rule types are:

- **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an event to OMi.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi.

Tasks

How to configure rules in log file policies

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific pattern matching field. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the pattern matching fields and values that the policy searches for in the structured log file that the policy reads. If the policy finds a match, it may or may not generate an event, depending on the rule type.
   a. Click to create a new condition. New conditions by default use the equals operator.
   b. Click ▸ to expand the new condition.
   c. In the **Property** field, specify the pattern matching field that the policy searches for (for example, `host`).

      If you are working with sample data, you can drag and drop the pattern matching field from the Input Data Properties list to the Properties field.
   d. Select the pattern operator.

      If you select the matches operator, you can type a pattern in the Operand field. For a list of available operators, see "Operator" on page 313.
   e. In the **Operand** field, type the value or pattern that you want the policy to compare with the input data reference field. If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.
3. Use the **Event Attributes** tab to define event attributes (for example, event title and description) for all events generated by this rule.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.
4. Use the **Event Correlation** tab to set the type of duplicate event suppression and define the method used to suppress duplicate events.

5. Use the **Custom Attributes** tab to add additional information to all events generated by this rule. For example, you might add a company name, contact information, or a city location to an event.
6. Use the **Advanced** tab to define an event drill-down URL, legacy HPOM attributes, and agent MSI (Message Stream Interface) settings.
7. *Optional.* Use the Sample Data tab to drag the input data references to the attribute boxes. Alternatively, you can type the reference directly into the attribute box.

   Input data references use the following syntax: <$DATA:<*InputReferenceField*>>.

   Field names originate from the structured log file pattern settings defined for the data source.

   BSM Connector replaces the pattern matching field at runtime with the value of the specified field. If you insert a field value, the value will be used.

   Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match the structured log file pattern specified in the source page. See also "Configuring Data Source in Structured Log File Policies" on page 238.

8. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings page (see also "Configuring Mappings in Structured Log File Policies (Event and Metrics Only) " on page 242).

   Alternatively, type the custom variable into the attribute box using the following syntax:

   <$MAP(<*CustomVariable*>)> where <*CustomVariable*> is the map name of the variable (for example, `<$MAP(maphost)>`).

   If the custom variable does not have a input data reference assigned, use the following syntax:

   <$MAP(<*CustomVariable*>,<*SourceValue*>) where <*SourceValue*> can be one of the following:

   - Name of the input data reference, for example `<$MAP(maphost)>,<$DATA:host>`
   - The source value itself, for example `<$MAP(maphost,Critical)>`

9. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the event attribute.
10. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
11. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

    HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from Structured Log Files" on page 234

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "Condition Tab - Rules Only" on page 312
- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Sample Data Tab - Structured Log File Policies" on page 341
- "Policy Variables Tab " on page 333

Configuring Metrics Rules in Structured Log File Policies

Rules define the action a policy should take in response to a specific type of incoming metric. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual metric data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

Note: For examples and end-to-end workflow information on collecting metrics data, see "Collecting and Viewing Metrics Data" on page 60.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

This section includes:

- "Rule types" on the next page
- "Metric attributes" on the next page

Rule types

The rule types are:

- **Store on matched rule.** If matched, BSM Connector stores metrics in a buffer until a maximum number of records or a maximum amount of time is reached. These metrics are sent to OMi when they are requested, and they use the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing.

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

  - Metric name

    Defines the metric name under which the metric appears in the Operations Agent store and consumers.

  - Related CI

    Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

  - Node

    Used to identify a node-like CI to which the performance records are associated to.

  - Value

    The actual performance value which is converted to 64-bit floating point number.

  - Time measured

    The time stamp when the value was determined in the third-party system.

- Advanced attributes:

- Original metric name

  The name of the metric as used on the third-party system.

- Unit

  The unit of the metric.

- Integration id

  An id, used to identify the source of the integration.

Tasks

How to configure rules for metrics in log file policies

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific pattern matching field. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the pattern matching fields and values that the policy searches for in the structured log file that the policy reads. If the policy finds a match, it may or may not generate an event, depending on the rule type.
   a. Click to create a new condition. New conditions by default use the equals operator.
   b. Click ▸ to expand the new condition.
   c. In the **Property** field, specify the pattern matching field that the policy searches for (for example, `host`).

      If you are working with sample data, you can drag and drop the pattern matching field from the Input Data Properties list to the Properties field.
   d. Select the pattern operator.

      If you select the matches operator, you can type a pattern in the Operand field. For a list of available operators, see "Operator" on page 313.
   e. In the **Operand** field, type the value or pattern that you want the policy to compare with the pattern matching field. If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.
3. *Optional.* If you are creating a rule of the type 'store on matched rule', set the attributes (**Basic** or/and **Advanced**) for metrics that you want the rule to override. If default attributes are specified in the Defaults tab, you use the defaults or you can override them as described below.
4. *Optional.* Use the Sample Data tab to drag the input data references to the attribute boxes. Alternatively, you can type the reference directly into the attribute box.

   Input data references use the following syntax: <$DATA:<*InputReferenceField*>>.

   Field names originate from the structured log file pattern settings defined for the data source.

   BSM Connector replaces the pattern matching field at runtime with the value of the specified field. If you insert a field value, the value will be used.

   Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match the structured log file pattern specified in the source page.

See also "Configuring Data Source in Structured Log File Policies" on page 238.

5. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the Mappings page (see also "Configuring Mappings in Structured Log File Policies (Event and Metrics Only) " on page 242).

   Alternatively, type the custom variable into the attribute box using the following syntax:

   <$MAP(<*CustomVariable*>)> where <*CustomVariable*> is the map name of the variable (for example, `<$MAP(maphost)>`).

   If the custom variable does not have a input data reference assigned, use the following syntax:

   <$MAP(<*CustomVariable*>,<*SourceValue*>) where <*SourceValue*> can be one of the following:

   - Name of the input data reference, for example `<$MAP(maphost)>,<$DATA:host>`
   - The source value itself, for example `<$MAP(maphost,Critical)>`

6. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the metric attribute.

7. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string

   - the pattern definition

   - the output string if pattern matches on the input string

   - the output string if the pattern does not match (optional)

   Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

   ```
   $MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)
   ```

   - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

     For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

   Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

8. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

9. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example "<$MSG_NODE>" or "<$MSG_GEN_NODE>", at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Metrics Data from Structured Log Files" on page 235

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "UI Descriptions" on page 299
- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Sample Data Tab - Structured Log File Policies" on page 341
- "Operators Tab (Metrics Only)" on page 327
- "Policy Variables Tab " on page 333

Configuring Options in Structured Log File Policies

The options tab enables you to configure several policy behaviors, for example which events are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **Structured Log File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metrics >** **Structured Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Options** to open the policy Options page.

Tasks

How to configure options for structured log file policies

In the Options page, configure which events and metrics are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

For more information about the other fields, see "Options Page (Events only)" on page 328.

Related tasks

- "How to Collect Event Data from Structured Log Files" on page 234

UI Descriptions

For a description of the Options page, see "Options Page (Events only)" on page 328.

Troubleshooting Structured Log File Policies

This section describes troubleshooting and limitations when working with Log File policies.

General troubleshooting guidelines

- To start troubleshooting, check the HP Operations Agent log files in the `%OvDataDir%log` directory (Windows systems) or the `/var/opt/OV/log` directory (Linux systems).
- To investigate issues related to policy execution, first examine the `%OvDataDir%log\System.txt` file (Windows systems) or the `/var/opt/OV/log/System.txt` file (Linux systems).

> **Note:** The log files whose names start with the `opr-` prefix are generated by the OMi web console.

Chapter 18: XML File Policies

XML file policies read values in XML files and respond when the value that you choose appears in the file. XML file policies are especially suited for integrating data from third-party systems that can store their information in XML format.

XML file policies process XML files and send data to OMi when certain conditions apply. You can define the attributes of the OMi event or metric based on information in the XML file. This enables you to process data generated by other applications and to convert it to OMi events or metrics.

XML file policies process exactly the XML elements and attributes that you define. The XML syntax is not important to the policy, as long as the information is embedded in XML elements and attributes.

Tip: If the application does not store its data in XML files, you may write a program or script that extracts the events, metrics, or topology data from wherever they are stored, formats the data using XML syntax, and generates an XML file with the data. If you have control over the XML elements that are used in the XML file, choose XML elements and attributes that map to event or metric attributes and values. This will simplify the policy.

How to Collect Event Data from XML Files

This task describes how to collect event data from XML files.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Event >** **XML File**.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring XML File Policy Properties" on page 263.
4. In the **Source** page, define the XML file that the policy reads (for example, the path and name of the XML file).

   For details, see "Configuring the Data Source in XML File Policies" on page 264.
5. In the **Mappings** page, configure the default mappings of XML elements and attributes to custom variables.

   For details, see "Configuring Mappings in XML File Policies" on page 266.
6. *Optional.* In the **Defaults** page, configure the default settings for all events generated by the policy (for example, default event correlation settings).

   For details, see "Configuring Event Defaults in XML File Policies" on page 271.
7. In the **Rules** page, define what the policy should do in response to a specific type of event.

For details, see "Configuring Event Rules in XML File Policies" on page 274.

8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in XML File Policies" on page 281.
9. Click **Save and Close** to save the policy and close the editor.
10. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Metric Data from XML Files

This task describes how to collect metric data from XML files.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Metric > XML File**.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring XML File Policy Properties" on page 263.
4. In the **Source** page, define the XML file that the policy reads (for example, the path and name of the XML file).

   For details, see "Configuring the Data Source in XML File Policies" on page 264.
5. In the **Mappings** page, configure the default mappings of XML elements and attributes to custom variables.

   For details, see "Configuring Mappings in XML File Policies" on page 266.
6. *Optional.* In the **Defaults** page, assign default values to the metric attributes.

   For details, see "Configuring Metric Defaults in XML File Policies" on page 272.
7. In the **Rules** page, define what the policy should do in response to a specific metric.

   For details, see "Configuring Metrics Rules in XML File Policies" on page 277.
8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in XML File Policies" on page 281.
9. Click **Save and Close** to save the policy and close the editor.
10. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Topology Data from XML Files

This task describes how to collect topology data from XML files that are produced by topology discovery scripts.

Note: For detailed information on topology discovery scripts, see "Topology Policies and Topology Synchronization" on page 74.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Topology >** **XML File**.

   Alternatively, double-click an existing policy to edit it.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring XML File Policy Properties" on the next page.
4. In the **Source** page,define the path of the XML file and set the age to deletion.

   For details, see "How to configure the XML source (topology)" on page 266.
5. Click **Save and Close** to save the policy and close the editor.
6. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

How to Collect Generic Output Data from XML Files

This task describes how to collect generic output data from XML files.

1. *Prerequisite:* Make sure the following applies:
   a. Your BSM Connector is configured to work with OMi by using the `bsmc-conf` tool.
   b. The certificate request is approved on the OMi side.
   c. Your BSM Connector is set up as a connected server in OMi.
2. In the BSM Connector user interface, click in the toolbar. Then click **Event >** **XML File**.
3. In the **Properties** page, define information that is related to the policy itself (for example, the name and description of the policy).

   For details, see "Configuring XML File Policy Properties" on the next page.
4. In the **Source** page, define the XML file that the policy reads (for example, the path and name of the XML file).

   For details, see "Configuring the Data Source in XML File Policies" on page 264.
5. In the **Mappings** page, configure the default mappings of XML elements and attributes to custom variables.

   For details, see "Configuring Mappings in XML File Policies" on page 266.
6. Click **Save and Close** to save the policy and close the editor.
7. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click in the toolbar.

XML File Policy User Interface

This section includes:

- "Configuring XML File Policy Properties" below
- "Configuring the Data Source in XML File Policies" on the next page
- "Configuring Mappings in XML File Policies" on page 266
- "Configuring Event Defaults in XML File Policies" on page 271
- "Configuring Event Rules in XML File Policies" on page 274
- "Configuring Options in XML File Policies" on page 281

Configuring XML File Policy Properties

Every policy has a set of properties that identify and describe the policy. Properties include, for example, the policy name and a description of what the policy does.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **XML File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **XML File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Topology >** **XML File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **XML File**.

Tasks

How to configure policy properties

In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

For more information about the other fields, see "Configuring XML File Policy Properties" above.

Related tasks

- "How to Collect Event Data from XML Files" on page 260
- "How to Collect Metric Data from XML Files" on page 261
- "How to Collect Topology Data from XML Files" on page 261
- "How to Collect Generic Output Data from XML Files" on the previous page

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 338.

Configuring the Data Source in XML File Policies

The source page of the XML file policy editor enables you to specify which XML file the policy reads. You can also set options that configure how the policy reads the file.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **XML File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **XML File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Topology >** **XML File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **XML File**.

Alternatively, double-click an existing policy to edit it.

Click **Source** to open the policy Source page.

Learn More

Requirements for XML source files (events)

XML files must meet the following criteria so that they can be processed correctly by XML file policies:

- XML file requirements:
  - Use file rolling to create two or more XML input files. BSM Connector first reads the oldest file before starting to process the oldest of the newer files. Each file should not be larger than 2 GB.

    You can use the asterisk (<*>) wildcard string in the **Log File Path / Name** text box on the Source page to match multiple file names. For example, to match the XML source file names `events.1.xml` and `events.2.xml`, use the pattern `<path>/events<*>.xml` in the Log File Path / Name text box. Note that the <*> wildcard string is the only supported OMi pattern in log file paths. For more information on pattern matching, see "Pattern-Matching Details" on page 284.

    When using file rolling, make sure the **Close after reading** option on the Source page is selected.

  - Make sure the polling interval is shorter than the frequency in which data is written to the file. The polling internval must not be shorter than `15 s`, otherwise the policy cannot be saved.
- XML format requirements:
  - The root element is optional.
  - If a root element exists, it must not be closed by its end tag.
  - All other XML elements must be complete.

  The following example XML document begins with the root tag `<AllAlerts>` and contains two types of events: performance alerts and availability alerts. If you define the XML elements

<PerformanceAlert> and <AvailabilityAlert> as XML event tags in the Source page of XML file policies, only those events are processed by XML file policies.

Example:

```
<AllAlerts>
  <AvailabilityAlert>
     <Title>Host Unreachable</Title>
     <Severity>Critical</Severity>
     <TimeOccured>02/11/10 03:52:18AM</TimeOccured>
     <Object>Host:fish.example.com</Object>
  </AvailabilityAlert>
  <PerformanceAlert>
     <Title>Disk IO rate high</Title>
     <Severity>Warning</Severity>
     <TimeOccured>02/11/10 04:08:31AM</TimeOccured>
     <Object>Disk:disk0:dog.example.com</Object>
  </PerformanceAlert>
  <AvailabilityAlert>
     <Title>Web Application unresponsive</Title>
     <Severity>Critical</Severity>
     <TimeOccured>02/11/10 05:01:26AM</TimeOccured>
     <Object>WebApp:http://employeeportal.intra.example.com</Object>
  </AvailabilityAlert>
  <PerformanceAlert>
     <Title>Phyiscal Read Rate high for Bufferpool BP1</Title>
     <Severity>Warning</Severity>
     <TimeOccured>02/11/10 08:37:09AM</TimeOccured>
     <Object>DB:USRDB:cat.example.com</Object>
  </PerformanceAlert>
  <PerformanceAlert>
     <Title>Phyiscal Read Rate high for Bufferpool BP1</Title>
     <Severity>Warning</Severity>
     <TimeOccured>02/11/10 08:37:09AM</TimeOccured>
     <Object>DB:USRDB:cat.example.com</Object>
  </PerformanceAlert>
</AllAlerts>
```

Requirements for XML files (topology)

XML files must follow the topology discovery syntax. See "Topology Discovery Syntax" on page 107.

Tasks

How to configure the XML source (events and metrics)

This task describes how configure the XML source file and how the policy reads it.

1. Type the full path to the XML file on the BSM Connector system.
2. Click to load a sample XML file. You can load a sample file from the BSM Connector system or from the system where the Web browser runs.

   When you load sample data, BSM Connector replaces already loaded data with the new data. This does not affect any mappings that are defined based on previously available sample data.
3. Click to create one or more XML tags. You can create a tag manually by typing the XML element. If you are working with sample data, you can create a tag by double-clicking the XML element in the list.

   The XML tag creates a shortcut to the XML element that you want the policy to process. An event tag typically identifies an event record in an XML log file, while a metric tag identifies a metric record in an XML log file. You can define more than one XML tag. For example, an XML file may contain two types of events: `<PerformanceAlert>` and `<AvailabilityAlert>`. To process both types, define both elements as event tags.

How to configure the XML source (topology)

This task describes how configure the XML source file and how the policy reads it.

1. Type the full path to the XML file on the BSM Connector system.
2. Enter the age to deletion - the number of times for the policy to be run for a host that is not part of the topology data. After this number of policy executions, this host is deleted from the server.
3. Optionally, enable delta detection, so that only the difference between the received XML file and the repository is sent to the topology server.

Related tasks

- "How to Collect Event Data from XML Files" on page 260
- "How to Collect Metric Data from XML Files" on page 261
- "How to Collect Topology Data from XML Files" on page 261
- "How to Collect Generic Output Data from XML Files" on page 262

UI Descriptions

For a description of the Source page, see "Source Page - XML File Policies" on page 355.

Configuring Mappings in XML File Policies

The Mappings page enables you to map XML elements and attributes to custom variables.

To access

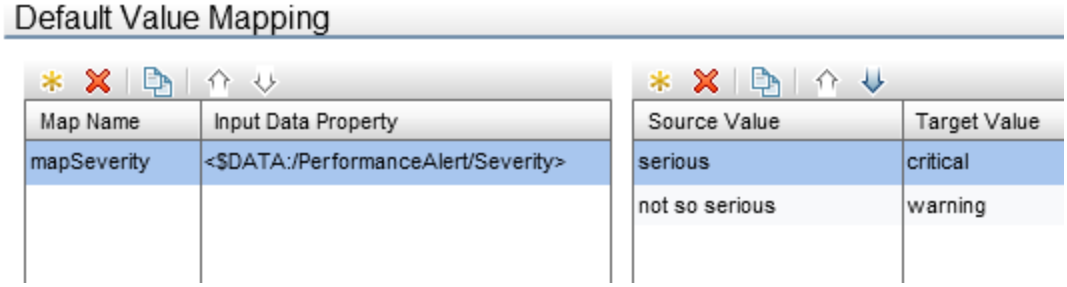
- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **XML File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **XML File**.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

A custom mapping definition consists of a map name (variable), an optional input data property (XML element or XML attribute), and one or more source and target value pairs. For example, you can assign the XML element `Severity` to the map name `mapSeverity`, and add a source value of `serious`. You can then assign the target value `critical` to the variable so that BSM Connector inserts the value `critical` into the event in all places where the variable is used.



XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

<XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

For example, the custom definition with the `mapSeverity` map name has the input data property of `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

XML properties are optional. If you do not assign an input data property to a map name (variable), you must add the source value directly to the variable when you insert the variable in an event attribute.

Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match any specified XML tags.

The Sample Data tab shows the following information if sample data is available:

- XML Properties

  If sample data is available, the XML Properties section of the Sample Data tab shows all XML elements and attributes that match an XML tag.

  The XML Properties section by default shows the short path to the XML property or value. To view the full path, click . The full path begins with the XML tag specified in the Source tab.

  The items in the XML Properties section are by default sorted alphabetically in ascending order.

  To search for an XML property or value, type the search string in the Search Properties box. The list changes as you type; only matching items appear.

- Values for *<XMLProperty>*

  This section displays the values of an XML property selected in the XML Properties section. If a value appears more than once, click to show or hide duplicate values. To find values that belong

to more than one XML property, select the value and click . The XML Sample Data window opens and shows all XML properties that have the selected value.

When you drag an XML element or an XML attribute from the XML properties list and drop it on the Default Value Mapping list, BSM Connector automatically adds the default prefix `map` to the map name and inserts the correct path to the XML property. You can then drag one or more XML source values from the XML values list and drop them on the Source Value list. You then finally only have to type the target values.

Tasks

How to configure XML mappings

This task describes how to map XML elements and attributes to custom variables.

1. Create one or more custom variables.

   If you are working with sample data, drag the XML elements or attributes from the XML Properties list to the Map Name column. BSM Connector automatically adds the default prefix `map` to the map name and inserts the correct path to the XML property.

   Alternatively, click above the Map Name column and type the variable name in the map name field. XML properties are optional. If you do not assign an XML property to a variable, you must add the source value directly to the variable when you insert the variable in an event attribute.
2. Add source and target value pairs to each custom variable.
   - If sample data is loaded in BSM Connector, drag a value from the Values for '...' list to the Source Value column, and then type the target value in the corresponding field.

     Alternatively, cClick above the Source Value column and type the source and target values in the corresponding fields.

   - *Optional.* In the Indicators tab, add indicators to the source or target value fields. After loading the indicators from the OMi server, the Indicators tab shows a hierarchy of configuration item types.

     To insert an indicator in a source or target value field, drag the indicator state (for example, `HTTPServer:Normal`) from the Indicators tab and drop it on the corresponding field.

   - *Optional.* In the Policy Variables tab, add policy variables to event or metric attributes. BSM Connector replaces the variables with the appropriate values in the generated event or metric.

     HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from XML Files" on page 260

UI Descriptions

For a description of the Mappings page, see the following sections:

- "UI Descriptions" on page 299
- "Sample Data Tab - XML File Policies" on page 343
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Mappings in XML File Policies (Generic Output Only)

The Mappings page enables you to map XML elements and attributes to custom variables.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Generic output >** **XML File**.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

The key field mapping consists of an input field qualifier, an eligible field name, and a mapped field name. To map the key field, map the eligible field name to the mapped field name. The eligible field name is automatically extracted from the input field qualifier.

XML policies integrate hierarchical data. Therefore, any field to be mapped must be a leaf node in the internal logical tree-like structure. As a result, the eligible field is the last entry in the tree.

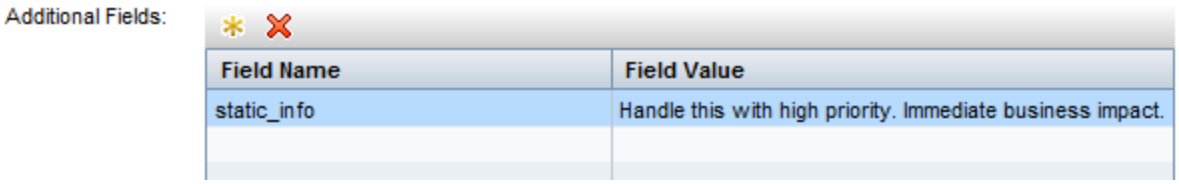
For example, if the input data has the structure `evt/event_type`, the eligible field is `event_type`. This field is then mapped, for example to the field `eventType`.

Key Field Mapping:

Input Field Qualifier	Eligible Field Name	Mapped Field Name
evt	evt	event
evt/event_type	event_type	eventType
evt/event_counter	event_counter	eventCounter
evt/event_criticality	event_criticality	eventSeverity

You can add **additional fields** to data that is sent to Data Forwarding targets. Additional fields are simple key-value pairs and you must manually add both the keys and values. If a defined key field name equals a field name in the mapped data set, it is discarded.

Additional fields are added as immediate children of the highest level element.



The actual field value can consist of user defined strings and data references to the input data (<$DATA:…>).

Example:

Additional Field Name	Additional Field Value
combined_text	At <$DATA:/evt/time_occured>, <$DATA:/evt/event_type> detected an <$DATA:/evt/event_impact> impact on system performance. This has happened <$DATA:/evt/event_counter> times.

The data references in the additional field `combined_text` are replaced when the policy is run. In the above example, after the variables are replaced, the resulting value in the field `combined_text` is:

```
At 12/05/2015 14:01:39, Monitoring: Threshold violation detected an substantial
impact on system performance. This has happened 8264 times.
```

Note that data references must refer to the *input* format, not the mapped format.

Tasks

How to configure mappings for key fields

This task describes how to map key fields.

1. Create one or more key field mappings.

   If you are working with meta data, drag the table column from the Meta Data list to the Input Field Qualifier column. BSM Connector automatically extracts the eligible field name.

   Alternatively, click above the Input Field Qualifier column and type the qualifier in the input field qualifier.
2. Optionally, change the **Keep all input fields** setting. By default, the option is selected and all fields are kept, regardless whether they are mapped or not. To keep only the mapped fields, deselect the option.
3. Add any additional fields as required.

Related tasks

- "How to Collect Event Data from XML Files" on page 260

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Generic Output)" on page 323

Configuring Event Defaults in XML File Policies

The Default Event Attributes page enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **XML File**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the Default Event Attributes page.

Tasks

How to configure events for XML file policies

This task describes how to configure default settings for all events generated by the policy.

1. Click **Event Attributes** to define default event attributes, such as severity and category.

   Tip: After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

   To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

2. Click **Event Correlation** to set the type of duplicate event suppression and define the method used to suppress duplicate events.
3. Click **Custom Attributes** to add additional information to all events generated by this policy. For example, you might add a company name, contact information, or a city location to an event.
4. Click **Advanced** to define default advanced attributes such as legacy HPOM attributes and agent MSI (Message Stream Interface) settings.
5. *Optional.* Use the Sample Data tab to drag XML properties (XML elements and attributes) and values to the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

   XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

   <XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

   BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

   Note: The Sample Data tab is empty if no sample data is loaded into BSM Connector or no XML data has been received for the policy.

6. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the mappings tab (see also "Configuring Event Defaults in XML File Policies" on the previous page). The default name of the mapping variable is `map<XMLProperty>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box using the following syntax:

   - Map Name list contains the map name of the variable: `map<XMLProperty>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:/<XMLTag>/<XMLProperty>>`

     For example, the custom variable `mapSeverity` has the following input data property: `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

7. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
8. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from XML Files" on page 260

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Sample Data Tab - XML File Policies" on page 343
- "Mappings Tab" on page 325
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metric Defaults in XML File Policies

The Metric page enables you to indicate default settings for all metrics generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **XML File**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the Default Metric Attributes page.

Tasks

How to configure metrics defaults for XML file policies

This task describes how to configure metric attribute defaults for all metrics collected by this policy.

1. Define the metric attributes common to all metrics collected by this policy, such as metric class and name. All metrics in the **Basic** tab marked with a * are required. **Advanced** attributes are optional.
2. *Optional.* Use the Sample Data tab to drag XML properties (XML elements and attributes) and values to the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

   XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

   <XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

   BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

   Note: The Sample Data tab is empty if no sample data is loaded into BSM Connector or no XML data has been received for the policy.

3. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the mappings tab (see also "Configuring Metric Defaults in XML File Policies" on the previous page). The default name of the mapping variable is `map<XMLProperty>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box using the following syntax:

   - Map Name list contains the map name of the variable: `map<XMLProperty>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:/<XMLTag>/<XMLProperty>>`

     For example, the custom variable `mapSeverity` has the following input data property: `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

4. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

- the input string
- the pattern definition
- the output string if pattern matches on the input string
- the output string if the pattern does not match (optional)

Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

```
$MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)
```

- `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

  For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

5. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.
6. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from XML Files" on page 260

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Sample Data Tab - XML File Policies" on page 343
- "Mappings Tab" on page 325
- "Operators Tab (Metrics Only)" on page 327
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Event Rules in XML File Policies

Rules define the action a policy should take in response to a specific type of incoming event. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual event data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Event >** **XML File**.

Click **Rules** to open the policy Rules page.

Learn More

Rule types

The rule types are:

- **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an event to OMi.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi.

Tasks

How to configure event rules in XML policies

This task describes how configure policy rules.

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific string in the XML file. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the XML properties and values that the policy searches for in the XML file that the policy reads. If the policy finds a match, it may or may not generate an event, depending on the rule type.
   a. Click to create a new condition. New conditions by default use the equals operator.
   b. Click ▸ to expand the new condition.
   c. In the **Property** field, specify the XML element or attribute that the policy searches for. You must specify the XML path from the XML event tag to the property, separated by slash marks (/) (for example, `/PerformanceAlert/Severity`).

      If you are working with sample data, you can drag and drop the XML element or attribute from the XML Properties list to the Properties field.
   d. Select the pattern operator.

If you select the matches operator, you can type a pattern in the Operand field. For a list of available operators, see "Operator" on page 313.

e. In the **Operand** field, type the value or pattern that you want the policy to compare with the XML property. If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.

3. Use the **Event Attributes** tab to define event attributes (for example, event title and description) for all events generated by this rule.

> **Tip:** After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.
>
> To insert an indicator, drag the indicator with its state from the Indicators tab to the policy.

4. Use the **Event Correlation** tab to set the type of duplicate event suppression and define the method used to suppress duplicate events.
5. Use the **Custom Attributes** tab to add additional information to all events generated by this rule. For example, you might add a company name, contact information, or a city location to an event.
6. Use the **Advanced** tab to define an event drill-down URL, legacy HPOM attributes, and agent MSI (Message Stream Interface) settings.
7. *Optional.* Use the Sample Data tab to drag XML properties (XML elements and attributes) and values to the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

   XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

   <XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

   BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

> **Note:** The Sample Data tab is empty if no sample data is loaded into BSM Connector or no XML data has been received for the policy. See also "Configuring the Data Source in XML File Policies" on page 264.

8. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the mappings tab (see also "Configuring Event Rules in XML File Policies" on page 274). The default name of the mapping variable is `map<XMLProperty>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box using the following syntax:

   - Map Name list contains the map name of the variable: `map<XMLProperty>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:/<XMLTag>/<XMLProperty>>`

     For example, the custom variable `mapSeverity` has the following input data property: `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

9. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the event attribute.

10. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
11. *Optional.* In the Policy Variables tab, add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

    HP recommends to surround variables with quotation marks, for example `"<$MSG_NODE>"` or `"<$MSG_GEN_NODE>"`, at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from XML Files" on page 260

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "Condition Tab - Rules Only" on page 312
- "Event Attributes Tab" on page 300
- "Event Correlation Tab" on page 305
- "Custom Attributes Tab" on page 309
- "Advanced Tab" on page 310
- "Sample Data Tab - XML File Policies" on page 343
- "Mappings Tab" on page 325
- "Pattern Matching Variables Tab (Events and Metrics Only)" on page 332
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Metrics Rules in XML File Policies

Rules define the action a policy should take in response to a specific type of incoming metric. Each rule consists of the following:

- A condition for the incoming data

  The condition is the part of a policy that describes the data source.
- Settings for the outgoing event

  The settings define the actual metric data that BSM Connector sends to OMi.

A policy must contain at least one rule. If the policy contains multiple rules, they are evaluated consecutively. After the condition is matched in one rule, rule evaluation stops.

To access

In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **XML File**.

Alternatively, double-click an existing policy to edit it.

Click **Rules** to open the policy Rules page.

Learn More

Rule types

The rule types are:

- **Store on matched rule.** If matched, BSM Connector stores metrics in a buffer until a maximum number of records or a maximum amount of time is reached. These metrics are sent to OMi when they are requested, and they use the settings defined for the rule. If you do not configure these settings, the default settings are used.
- **Suppress on matched rule.** If matched, BSM Connector stops processing.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing.

Metric attributes

Each time a metric policy runs, it extracts raw data from its defined data source and builds a metric structure.

A metric structure consists of these attributes:

- Basic attributes:
  - Data domain

    The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes.

  - Metric class

    Defines the metric class under which the metric appears in the Operations Agent store and consumers.

  - Metric name

    Defines the metric name under which the metric appears in the Operations Agent store and consumers.

  - Related CI

    Used to identify an instance of a performance record and associates it with a concrete CI instance. For details on how to associate the Related CI with the values in the RTSM model, see "Create a policy" on page 60. For an example, see "Example – Create a Metrics Policy" on page 61.

  - Node

Used to identify a node-like CI to which the performance records are associated to.

- Value

  The actual performance value which is converted to 64-bit floating point number.

- Time measured

  The time stamp when the value was determined in the third-party system.

- Advanced attributes:
  - Original metric name

    The name of the metric as used on the third-party system.

  - Unit

    The unit of the metric.

  - Integration id

    An id, used to identify the source of the integration.

Tasks

How to configure metric rules in XML policies

This task describes how configure policy rules.

1. In the **Policy Rules** section, click and select the type of rule to define what the policy should do in response to a specific string in the XML file. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to define the match condition.

   Click to create a new condition. Enter the property and select the operator from the drop-down list. Add the operand. Use the arrows and collapse/expand buttons to navigate the conditions.
3. *Optional.* If you are creating a rule of the type 'store on matched rule', set the attributes (**Basic** or/and **Advanced**) for metrics that you want the rule to override. If default attributes are specified in the Defaults tab, you use the defaults or you can override them as described below.

   The attributes are in two groups, basic and advanced. The advanced attributes are optional.
4. *Optional.* Use the Sample Data tab to drag XML properties (XML elements and attributes) and values to the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

   XML properties use the following syntax: <$DATA:/*<XMLProperty>*>

   <XMLProperty> is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters.

   BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

   Note: The Sample Data tab is empty if no sample data is loaded into BSM Connector or no XML data has been received for the policy. See also "Configuring the Data Source in XML File

Policies" on page 264.

5. *Optional.* Use the Mappings tab to add mapping definitions to the attribute boxes.

   Mappings are custom variables that you define in the mappings tab (see also "Configuring Metrics Rules in XML File Policies" on page 277). The default name of the mapping variable is `map<XMLProperty>`, for example `mapSeverity`.

   Alternatively, type the custom variable into the attribute box using the following syntax:

   - Map Name list contains the map name of the variable: `map<XMLProperty>`, for example `mapSeverity`.
   - Input Data Property list item: `<$DATA:/<XMLTag>/<XMLProperty>>`

     For example, the custom variable `mapSeverity` has the following input data property: `<$DATA:/PerformanceAlert/Severity>` where `Severity` is a child element of `PerformanceAlert`.

6. Optionally, use the **Pattern Matching Variables** tab to add variables created with pattern matching.

   Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the metric attribute.

7. *Optional.* Use the Operators tab to apply operators to the attribute values. Two functions are available:

   `<$MATCH()>`, to test a string or a variable against a pattern. The $MATCH function accepts three or four parameters:

   - the input string
   - the pattern definition
   - the output string if pattern matches on the input string
   - the output string if the pattern does not match (optional)

   Example: The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is as follows:

   `$MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)`

   - `<$DATETIME(FORMAT,VALUE)>`, to convert the format of dates from the common format to the UNIX systems time (Epoch time) format.

     For detailed description of the format, see "Pattern Matching in Policy Rules" on page 284.

   Note: To apply operators to the attribute values, you can drag and drop them to a text field in the left pane of the same policy editor page. The appropriate tooltips are shown while performing this operation, which describe the role of the dragged operator.

8. *Optional.* Use the Indicators tab to add indicators to the source or target value fields. After loading the indicators from the connected OMi server, the Indicators tab shows a hierarchy of configuration item types.

9. *Optional.* In the Policy Variables tab, add policy variables to metric attributes. BSM Connector replaces the variables with the appropriate values in the generated metric.

   HP recommends to surround variables with quotation marks, for example "<$MSG_NODE>" or "<$MSG_GEN_NODE>", at least for those variables whose values can contain space characters.

Related tasks

- "How to Collect Event Data from XML Files" on page 260

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 339
- "UI Descriptions" on page 299
- "Default Metric Attributes — Basic Tab" on page 316
- "Default Metric Attributes — Advanced Tab" on page 319
- "Sample Data Tab - REST Web Service Listener Policies" on page 342
- "Mappings Tab" on page 325
- "Pattern Matching Variables Tab (Events and Metrics Only)" on page 332
- "Indicators Tab" on page 320
- "Policy Variables Tab " on page 333

Configuring Options in XML File Policies

The options tab enables you to configure several policy behaviors, for example which events or metrics are logged locally, how the policy handles unmatched events, and defaults for pattern matching in policy rules.

To access

- In the BSM Connector user interface, click in the toolbar. Then click **Event >** **XML File**.
- In the BSM Connector user interface, click in the toolbar. Then click **Metric >** **XML File**.

Alternatively, double-click an existing policy to edit it.

Tasks

How to configure options for XML policies

In the Options page, configure which events or metric records are logged locally, how the policy handles unmatched events or metric records, and defaults for pattern matching in policy rules.

For more information about the other fields, see "Options Page (Events only)" on page 328.

Related tasks

- "How to Collect Event Data from XML Files" on page 260

UI Descriptions

For a description of the Options page, see "Options Page (Events only)" on page 328.

Part IV: BSM Connector Reference

This section includes:

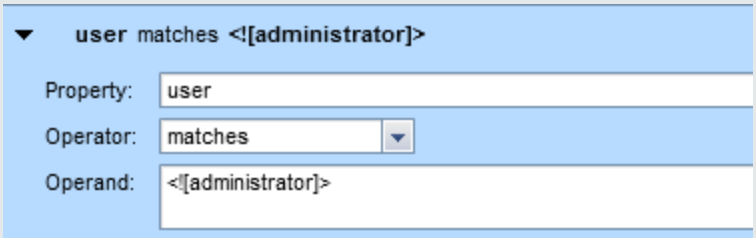
- "Pattern Matching in Policy Rules" on page 284
- "Command-Line Tools" on page 293
- "UI Descriptions" on page 299

Appendix A: Pattern Matching in Policy Rules

BSM Connector uses pattern matching to determine if an event or metric record should be sent to OMi or should be suppressed.

BSM Connector compares the pattern that you specify with the data contained in XML elements (this can be any data source, such as the database or XML file record) or attributes. When the pattern is matched, the policy sends or suppresses an event or metric record, depending on the rule type.

Example: A structured log file policy contains a rule of the type "event on matched rule". A condition compares the pattern `<![administrator]>` with the value of "user".



If the value of "user" is `administrator`, the pattern does not match and the next condition is evaluated. If the value is `user1`, the pattern matches and an event is sent to OMi.

For more information about the pattern matching syntax used to in rule conditions, see "Pattern-Matching Details" below.

Pattern-Matching Details

HP Operations Agent provides a powerful pattern-matching language that reduces the number of conditions you must use. Selected, dynamic parts of text-based events can be extracted, assigned to variables, and used as parameters to build the event description or to set other attributes.

The pattern-matching language enables you to very accurately specify the character string that you want a rule to match.

Note: In text boxes where pattern-matching expressions are allowed you can click ▸ for a shortcut menu with pattern-matching expressions that can be selected and inserted into the text box.

Matching special characters

Ordinary characters are expressions which represent themselves. Any character of the supported character set may be used. However, if any of the following special characters are used they must be prefaced with a backslash (\) that masks their usual function.

```
\ [ ] < > | ^ $
```

If ^ and $ are not used as anchoring characters, that is, not as first or last characters, they are considered ordinary characters and do not need to be masked.

Matching characters at the beginning or end of a line

If the caret (^) is used as the first character of the pattern, only expressions discovered at the beginning of lines are matched. For example, "^ab" matches the string "ab" in the line "abcde", but not in the line "xabcde".

If the dollar sign is used as the last character of a pattern, only expressions at the end of lines are matched. For example, "de$" matches "de" in the line "abcde", but not in the line "abcdex".

Matching multiple characters

Patterns used to match strings consisting of an arbitrary number of characters require one or more of the following expressions:

- <*> matches any string of zero or more arbitrary characters (including separators)
- <*n**> matches a string of *n* arbitrary characters (including separators)
- <#> matches a sequence of one or more digits
- <*n*#> matches a number composed of *n* digits
- <_> matches a sequence of one or more field separators
- <*n*_> matches a string of *n* separators
- <@> matches any string that contains no separator characters, in other words, a sequence of one or more non-separators; this can be used for matching words
- </> matches one or more line breaks
- <*n*/> matches exactly *n* line breaks
- <S> matches one or more white space characters: space, tab and new line characters (" ", \t, \n, \r)
- <*n*S> matches exactly *n* white space characters

> **Note:** On Windows operating systems, a new line consists of two white space characters (\n\r).

Separator characters are configurable for each pattern. By default, separators are the space and the tab characters.

Matching two or more different expressions

Two expressions separated by the special character vertical bar (|) matches a string that is matched by either expression. For example, the pattern:

```
[ab|c]d
```

matches the string "abd" and the string "cd".

Matching text that does not contain an expression

The NOT **operator** (!) must be used with delimiting square brackets, for example:

```
<![WARNING]>
```

The pattern above matches all text which does not contain the string "WARNING".

The **NOT operator** may also be used with complex subpatterns:

```
SU <*> + <@.tty> <![root|[user[1|2]]].from>-<*.ot>
```

The above pattern makes it possible to generate a "switch user" event for anyone who is not user1, user2 or root. Therefore the following would be matched:

```
SU 03/25 08:14 + ttyp2 user11-root
```

However, this line would not be matched, because it contains an entry concerning "user2":

```
SU 03/25 08:14 + ttyp2 user2-root
```

Notice that if the subpattern including the **not operator** does not find a match, the **not operator**behaves like a <*>: it matches zero or more arbitrary characters. For this reason, the pattern-matching expression: <![1|2|3]> matches any character or any number of characters, except 1, 2, or 3.

Mask (\ ) Operator

The backslash (\ ) is used to mask the special meaning of the characters:

```
[ ] < > | ^ $
```

A special character preceded by \ results in an expression that matches the special character itself.

Notice that because ^ and $ only have special meaning when placed at the beginning and end of a pattern respectively, you do not need to mask them when they are used within the pattern (in other words, not at beginning or end).

The only exception to this rule is the tab character, which is specified by entering "\t" into the pattern string.

Bracket ([and]) Expressions

The brackets ([and]) are used as delimiters to group expressions. To increase performance, brackets should be avoided wherever they are unnecessary. In the pattern:

```
ab[cd[ef]gh]
```

all brackets are unnecessary--"abcdefgh" is equivalent.

Bracketed expressions are used frequently with the **OR operator**, the **NOT operator** and when using **subpatterns** to assign strings to variables.

Numeric range operators

HP Operations Agent provides six numeric range operators that can be used in pattern matching. The operators are used in this way:

Operator name	Syntax	Example/Explanation

Less than	<[*pattern This is a match pattern you provide that returns the number to be compared*] -lt *n This is the value against which you want to test the number returned by the match pattern*>	<[<#>] -lt 5> matches every number less than 5
Less than or equal to	**<[*pattern*] -le *n* >**	<[<#>] -le 5> matches 5 and every number less than or equal to 5
Greater than	**<[*pattern*] -gt *n* >**	<[<#>] -gt 5> matches every number greater than 5
Greater than or equal to	**<[*pattern*] -ge *n* >**	<[<#>] -ge 5> matches 5 and every number greater than or equal to 5
Equal to	**<[*pattern*] -eq *n* >**	<[<#>] -eq 5> matches 5 or 5.0
Not equal to	**<[*pattern*] -ne *n* >**	<[<#>] -ne 5> matches every number but 5 and 5.0
The operators can also be combined to produce matches according to ranges of numbers:		
Matches numbers that belong to the interval, excluding the limits	**< *n* -lt [*pattern*] -lt *n* >**	<5 -lt [<#>] -lt 10> matches every number between 5 and 10 (but not 5 or 10)
Matches numbers that belong to the interval, including the limits	**< *n* -le [*pattern*] -le *n* >**	<5 -le [<#>] -le 10> matches every number between 5 and 10 (including 5 and 10)

Matches numbers that do not belong to the interval, excluding the limits	< *n* **-gt** [*pattern*] **-gt** *n* >	<10 -gt [<#>] -gt 5> matches every number between 5 and 10 (but not 5 or 10)
Matches numbers that do not belong to the interval, including the limits	< *n* **-ge** [*pattern*] **-ge** *n* >	<10 -ge [<#>] -ge 5> matches every number between 5 and 10 (including 5 and 10)

User-Defined Variables in Patterns

Any matched string can be assigned to a variable, which can be used to compose events. To define a parameter, add ". `parametername` " before the closing bracket. The pattern:

```
^errno: <#.number> - <*.error_text>
```

matches an event such as:

```
errno: 125 - device does not exist
```

and assigns "125" to **number** and "device does not exist" to **error_text**.

When using these variables, the syntax is *<variable_name>* (for example, <number>).

Rules by which HP Operations Agent assigns strings to variables

In matching the pattern <*.**var1**><*.**var2**> against the string "abcdef", it is not immediately clear which substring of the input string will be assigned to each variable. For example, it is possible to assign an empty string to **var1** and the whole input string to **var2**, as well as assigning "a" to **var1** and "bcdef" to **var2**, and so forth.

The pattern matching algorithm always scans both the input line and the pattern definition (including alternative expressions) from left to right. <*> expressions are assigned as few characters as possible. <#>, <@>, <S> expressions are assigned as many characters as possible. Therefore, **var1** will be assigned an empty string in the example above.

To match an input string such as:

```
this is error 100: big bug
```

use a pattern such as:

```
error<#.errnumber>:<*.errtext>
```

In which:

- "100" is assigned to **errnumber**
- "big bug" is assigned to **errtext**

For performance and pattern readability purposes, you can specify a delimiting substring between two expressions. In the above example, ":" is used to delimit <#> and <*>.

Matching <@.**word**><#.**num**> against "abc123" assigns "abc12" to **word** and "3" to **num**, as digits are permitted for both <#> and <@>, and the left expression takes as many characters as possible.

Patterns without expression anchoring can match any substring within the input line. Therefore, patterns such as:

```
this is number<#.num>
```

are treated in the same way as:

```
<*>this is number<#.num><*>
```

Using subpatterns to assign strings to variables

In addition to being able to use a single operator, such as ***** or **#**, to assign a string to a variable, you can also build up a complex subpattern composed of a number of operators, according to the following pattern: <[*subpattern*].**var**>

For instance: <[<@>file.tmp].**fname**>

In the example above, the period (.) between "file" and "tmp" matches a similar dot character, while the dot between "]" and "**fname**" is necessary syntax. This pattern would match a string such as "Logfile.tmp" and assigns the complete string to **fname**.

Other examples of subpatterns are:

- <[Error|Warning].**sev**>
- <[Error[<#.n><*.msg>]].**complete**>$

In the first example above, any line with either the word "Error" or the word "Warning" is assigned to the variable, **sev**. In the second example, any line containing the word "Error" has the error number assigned to the variable, **n**, and any further text assigned to **msg**. Finally, the word "Error", the error number, and the text are assigned to **complete**.

The second example requires the dollar sign ($) at the end to anchor the expression. As mentioned above, patterns without expression anchoring can match any substring within the input line. Therefore, the pattern:

```
<[Error[<#.n><*.msg>]].complete>
```

would be treated as:

```
<*><[Error[<#.n><*.msg>]].complete><*>
```

Patterns are evaluated from left to right, and `<*>` expressions are assigned as few characters as possible. Therefore, without a dollar sign ($) to anchor the end of the expression, the `<*.msg>`

expression always matches zero characters, and the remainder of the line is matched with the implicit <*> expression at the end.

Pattern Matching for Variables

You can test a string or variable against a pattern, and define an output string that is conditional on the result. You can do this using **$MATCH**, which has the following syntax:

```
$MATCH(string, pattern, true, [false])
```

Specify the parameters as follows:

`string`
Specify a literal string (for example, `TEST STRING`) or a policy variable (for example <$LOGPATH>).

`pattern`
Specify a pattern, using HP Operations Agent pattern matching syntax. You can create user-defined variables in the pattern to use in the parameters `true` and `false`. The pattern is case sensitive.

`true`
Specify a string to return if the string and pattern match. You can specify a literal string, or a user-defined variable, or a policy variable.

`false`
Optional. Specify a string to return if the string and pattern do not match. You can specify a literal string, or a user-defined variable, or a policy variable.

Separate each parameter with a comma (,). To specify a comma within a parameter, you must precede it with two backslashes (\\).

You can use `$MATCH` within your policies in the following event attributes:

- Application
- Automatic command
- Category
- Custom message attribute values
- Message key
- Message type
- Object
- Service ID
- Title

Note: You can use $MATCH only once in each message attribute. You cannot use $MATCH recursively.

Example

A policy can read a number of log files. The name of the path of the log file is available in the policy variable <$LOGPATH>. If part of the log file path corresponds to an application name, you can use $MATCH to set the application event attribute as follows:

```
$MATCH(<$LOGPATH>,<@.application>.log, <application>, Unknown)
```

Examples of Pattern Matching in Rule Conditions

The following examples show some of the many ways in which the pattern-matching language can be used.

- `Error`

  Recognizes any event containing the keyword `Error` at any place in the event. (It is case sensitive by default.)

- `panic`

  Matches all events containing `panic`, `Panic`, `PANIC` anywhere in the text of the event, when case sensitive mode is switched off.

- `logon|logoff`

  Uses the **OR operator** to recognize any event containing the keyword `logon` or `logoff`.

- `^getty:<*.msg> errno<*><#.errnum>$`

  Recognizes any event such as: `getty: cannot open ttyxx errno : 6` or `getty: can't open ttyop3; errno 16`

  In the example `getty: cannot open ttyxx errno : 6`, the string "cannot open ttyxx" is assigned to the variable **msg**. The digit 6 is assigned to the variable **errnum**. Note that the dollar sign ($) is used as an anchoring symbol to specify that the digit 6 will only be matched if it is at the end of the line.

- `^errno[ |=]<#.errnum> <*.errtext>`

  Matches events such as: `errno 6 - no such device or address` or `errno=12 not enough core`.

  Note the space before the **OR operator**. The expression in square brackets matches either this blank space, or the "equals" sign. The space between <#.**errnum**> and <*.**errtext**> is used as a delimiter. Although not strictly required for assignments to the variables shown here, this space serves to increase performance.

- `^hugo:<*>:<*.uid>:`

  Matches any **/etc/passwd** entry for user `hugo` and returns the user ID to variable **uid**. Notice that ":" in the middle of the pattern is used to delimit the string passed to **uid** from the preceding string. The colon ":" at the end of the pattern is used to delimit the string passed to **uid** from the succeeding group ID in the input pattern. Here, the colon is necessary not only as a speed enhancement, but also as a means of logical separation between strings.

- `^Warning:<*.text>on node<@.node>$`

  Matches any event such as: `Warning: too many users on node hpbbx` and assigns `too many users` to **text**, and `hpbbx` to **node**.

- `^<*.line1><1/><*.line2><1/><*.line3><1/><*.line4>$`

  Matches four lines of text, for example:

  ```
  Security ID:     S-1-5-21-3358208617-1210941181-189752109-500
  Account Name:    Administrator
  ```

```
Account Domain:  EXAMPLE
Logon ID:        0x228a2
```

There is one line break between each line. The pattern assigns each line of text to a variable.

- `<<#> -le 45>`

  This pattern matches all strings containing a number which is less than or equal to 45. For example, the event: *ATTENTION: Error 40 has occurred* would be matched.

  Note that the number 45 in the pattern is a true numeric value and not a string. Numbers higher than 45, for instance, "4545" will not be matched even if they contain the combination, "45".

- `<15 -lt <2#> -le 87>`

  This pattern matches any event in which the first two digits of a number are within the range 16-87. For instance, the event: *Error Message 3299* would be matched. The string: *Error Message 9932* would not be matched.

- `^ERROR_<[<#.err>] -le 57>`

  This pattern matches any text starting with the string "ERROR_" immediately followed by a number less than, or equal to, 57.

  For example, the event: *ERROR_34: processing stopped* would be matched and the string 34 would be assigned to the variable, *err*.

- `<120 -gt [<#>1] -gt 20>`

  Matches all numbers between 21 and 119 which have 1 as their last digit. For instance, events containing the following numbers would be matched: 21, 31, 41... 101... 111 and so on.

- `Temperature <*> <@.plant>: <<#> -gt 100> F$`

  This pattern matches strings such as: "Actual Temperature in Building A: 128 F". The letter "A" would be assigned to the variable, *plant*.

- `Error <<#> -eq 1004>`

  This pattern matches any event containing the string "Error" followed by a space and the sequence of digits, "1004".

  For example, *Warning: Error 1004 has occurred* would be matched by this pattern. However, *Error 10041* would not be matched by this pattern.

- `WARNING <<#> -ne 107>`

  This pattern matches any event containing the string "WARNING" followed by a space and any sequence of one or more digits, except "107". For example, the event: *Application Enterprise (94/12/45 14:03): WARNING 3877* would be matched.

Appendix B: Command-Line Tools

You can use the following command-line tools to configure BSM Connector:

- "BSM Connector Configuration Tool" below
- "Local User Configuration Tool" on page 297

Tool location

The BSM Connector command-line tools are located on the BSM Connector server in:

Windows: `%OvDataDir%installation\HPOprBSMC\`

Linux: `/var/opt/OV/installation/HPOprBSMC/`

Log files

All BSM Connector command-line tools write information to the following log file on the BSM Connector server:

Windows: `%OvDataDir%log\opr-pm-cli.log`

Linux: `/var/opt/OV/log/opr-pm-cli.log`

The tools append log information to the file when you run them again.

BSM Connector Configuration Tool

You can reconfigure BSM Connector by running the configuration program **bsmc-conf**.

The **bsmc-conf** tool is located on the BSM Connector server at the following location:

Windows: `%OvDataDir%\installation\HPOprBSMC\bsmc-conf.bat`

Linux: `/var/opt/OV/installation/HPOprBSMC/bsmc-conf.sh`

Command Syntax

```
bsmc-conf.[bat|sh] -srv <hostname> [-cs <hostname>] [-f] [-flexmgmt]
   -admin_user <username> <password>
   [-https_port <port>] [-http_port <port>]
   [-lwsso_domain <domain_name>] [-lwsso_groups <group0> [<group1> ...]]
   [-lwsso_key <initString>]
```

This section includes:

- "BSM Connector Configuration Tool" above
- "To reconnect BSM Connector to another OMi server" on page 296
- "To install and configure BSM Connector on an HPOM managed node" on page 296

- "Log files" on page 297
- "Using ovconfchg" on page 297

Option Descriptions

Option	Value
`-srv <hostname>`	Specifies the OMi Gateway Server to which the BSM Connector sends the collected data. `<hostname>` is the full qualified domain name (FQDN) of the OMi Gateway Server. Default: not set
`-cs <hostname>`	Specifies the certificate server for your OMi environment. `<hostname>` is the FQDN or IP address of the certificate server. Use this option if you want to issue certificates from a certificate server other than the OMi server specified using the `-s` option (for example from an HP Operations Manager server or a OMi server from another OMi deployment). If you omit this option, `bsmc-conf` uses the server specified with the `-srv` option. Note: When accessing OMi through an IIS secure reverse proxy (SRP), enter the FQDN of the OMi gateway server. For more information on configuring an IIS reverse proxy for OMi, see the *OMi Administration guide*.
`-flexmgmt`	Creates the flexible management policy **BSM Connector FlexMgmt Policy**.The policy configures the agent to send the following data to the OMi server: • Events generated by BSM Connector policies. These events automatically have the Type attribute set to BSMC_Message.
`[-f]`	Force mode (used to reconnect your HP Operations Agent).
`-admin_user <username> <password>`	Specify the credentials for the BSM Connector administrator. This user can access the local BSM Connector instance only by logging on with these credentials, and cannot gain access to OMi. Default: not set
`-https_port <port>`	Specify the secure (HTTPS) port at which the BSM Connector user interface runs. If not specified, port 30000 is used.

`-http_port <port>`	Specifies the HTTP port at which the BSM Connector user interface runs. By default, the connection is redirected and HTTPS is used instead.
`-lwsso_domain <domain_name>`	Specifies the domain of the associated OMi gateway server. For more information about obtaining the domain name, see the *Installation and Upgrade Guide*. LW-SSO does not require an external computer for authentication. LW-SSO enables a user to log into OMi once and gain access to all BSM applications in the same domain without being prompted to log in again. If you configure BSM Connector to use LW-SSO, OMi users can launch the BSM Connector user interface without having to provide additional credentials. For more information about LW-SSO, see the OMi Admin Guide. Default: <lwsso_disabled>
`-lwsso_groups [<group0> [<group1>...]`	Specifies the OMi groups that will have access to BSM Connector. Separate individual groups with spaces (for example, `-lwsso_groups BSMC_ADMINS BSMC_OPERS`). For information about users and roles, see the *Installation and Upgrade Guide*. **Note:** You cannot add the `admin` user to a group. If you want a super admin to be able to also have an LW-SSO connection to the BSM Connector, you need to create a separate user which is a super-admin (the check box **Super-Admin** is selected) and put this user in a group. By default, LW-SSO is disabled.
`-lwsso_key <initString>`	Init string from SSO config. The token key (init string) generated in the OMi Users and Permissions manager. **Note:** Single-sign on can only work if the token key that you type here is exactly the same as the token key on the OMi server. For more information about obtaining the token key, see the *Installation and Upgrade Guide*. Default: <lwsso_disabled>

`-omc_srv <hostname>`	FQDN of the HPOM management server. If you omit this option, **bsmc-conf** takes the value of `MANAGER` from the internal configuration settings file and inserts it as target server in the flexible management policy **BSM Connector FlexMgmt Policy**. All events that do not have the **Type** attribute set to **BSMC_Message** are sent to this server. **Tip:** Use the following command to find the current value of `MANAGER`: `ovconfget sec.core.auth MANAGER`

Example

To configure HP BSM Connector to connect to the server "server1.company.com", with "server2.company.com" as the certificate server, using the user "admin" with the password "long3Not4short1password!" and using the default HTTPS port, execute:

```
bsmc-conf.bat -srv server1.company.com -cs server2.company.com -admin_user admin
long3Not4short1password!
```

To reconnect BSM Connector to another OMi server

1. On the BSM Connector server, type the following command to reconnect the system to another OMi server:

   `bsmc-conf -srv <new OMi server> -force`
2. Add the BSM Connector as a connected server to the OMi Connected Servers manager (**Administration > Setup and Maintenance > Connected Servers**).
3. In the Certificate Requests Manager (**Administration > Setup and Maintenance > Certificate Requests**), grant the certificate generated by **bsmc-conf**.
4. Delete the BSM Connector integration from the old OMi server.
5. **Results.** BSM Connector sends all collected data to the new OMi server.

To install and configure BSM Connector on an HPOM managed node

1. On the BSM Connector server, type the following command to configure the BSM Connector on an HPOM managed node:

   `bsmc-conf -srv <new OMi server> -force -flexmgmt -omc_srv <HPOM management server>`
2. Add the BSM Connector as a connected server to the OMi Connected Servers manager (**Administration > Setup and Maintenance > Connected Servers**).

   Do *not* select the **Configure policy management** checkbox.
3. In the Certificate Requests Manager (**Administration > Setup and Maintenance > Certificate Requests**), grant the certificate generated by **bsmc-conf**.
4. **Results.** The new OMi server becomes the primary manager of the HP Operations Agent. The flexible management policy **BSM Connector FlexMgmt Policy** configures the agent to send the

following data to the OMi server:

- Events generated by BSM Connector policies. These events automatically have the **Type** attribute set to **BSMC_Message**.

  The HPOM management server receives only events that do not have the **Type** attribute set to **BSMC_Message**.

Log files

bsmc-conf writes information to the following log file:

Windows: `%OvDataDir%log/opr-pm-cli.log`

Linux: `/var/opt/OV/log/opr-pm-cli.log`

The tool appends log information to the file when you run the program again.

Using ovconfchg

The command-line tool `ovconfchg` enables you to change one or more configuration parameters in the internal configuration settings file. `ovconfchg` is an expert tool and should be used by experienced administrators only.

To run `ovconfchg`:

Windows: `ovconfchg -edit`

Linux: `/opt/OV/bin/ovconfchg -edit`

The `-edit` option starts a text editor to edit the settings file. After you have saved your changes, you must restart the application.

Local User Configuration Tool

Each BSM Connector instance maintains a local users store. These users can access the local BSM Connector instance only and cannot not gain access to other BSM Connector applications. When BSM Connector users launch the BSM Connector user interface, they have to provide the credentials of a local BSM Connector user, or, if smart card authentication is configured, the smart card PIN.

To add additional users to the user store, run the BSM Connector local user configuration tool **user**.

Note: Users added the local user store cannot be specified as BSM Connector user in the **Main Settings** of the OMi Connected Servers manager (**Administration > Setup and Maintenance > Connected Servers**). You must enter the BSM Connector administration user that you configured in the configuration wizard during installation.

The **user** tool is located on the BSM Connector server in:

Windows: `%OvDataDir%\installation\HPOprBSMC\user.bat`

Linux: `/var/opt/OV/installation/HPOprBSMC/user.sh`

`user` accepts the following options:

```
user -help | -list | -add <username> <password> | -delete <username> | -version
```

Command options

`-h,-help`

Shows tool usage and description.

`-l,-list`

Lists the users stored in the local user store.

`-a,-add <username> <password>`

Adds a user to the local users store. If the user already exists, the password is overwritten.

The username and password must contain ASCII characters only. The user name must contain at least three characters. Valid characters in user names are alphanumeric characters (`a-z`, `A-Z`, and `0-9`), hyphens (`-`), underscores (_), periods (`.`), and the at sign (`@`).

When configuring BSM Connector for smart card authentication, use the value of the User Principal Name (UPN) in the certificate as user name.

`-d,-delete <username>`

Deletes the specified user from the local user store.

`-v,-version`

Displays the BSM Connector version number.

Local user store

The local user store is located in the following file on the BSM Connector system:

Windows: `%OvDataDir%\conf\HPOprBSMC\users.properties`

Linux: `/var/opt/OV/conf/HPOprBSMC/users.properties`

To add local users

1. *Windows.* Run the local user configuration tool:
   a. Open a command prompt and type:
      `cd %OvDataDir%\installation\HPOprBSMC`
   b. Run the BSM Connector local user configuration tool, type:
      `user.bat -add <username> <password>`

   Linux. Start the BSM Connector local user configuration tool, type:
   `/var/opt/OV/installation/HPOprBSMC/user.sh -add <username> <password>`
2. *Optional.* Review the log file at:
   Windows: `%OvDataDir%log\opr-pm-cli.log`
   Linux: `/var/opt/OV/log/opr-pm-cli.log`
   The program appends log information to the file when you run the program again.

Appendix C: UI Descriptions

- Defaults and Rules Pages (Events) 299
- Defaults and Rules Pages (Metrics) 316
- Indicators Tab 320
- Mappings Page (Events and Metrics) 321
- Mappings Page (Generic Output) 323
- Mappings Tab 325
- Metric Page (Data Forwarding) 325
- Operators Tab (Metrics Only) 327
- Options Page (Events only) 328
- Options Page (Metrics only) 330
- Pattern Matching Variables Tab (Events and Metrics Only) 332
- Policy Variables Tab 333
- Properties Page 338
- Rules Page - Policy Rules 339
- Sample Data Tab 341
- Schedule Page 343
- Source Page 345
- Start, Success, Failure Event Pages (Scheduled Task Policies) 360
- Structured Input Page (Data Forwarding) 360
- Target Page (Data Forwarding) 362
- Task Page (Scheduled Task Policies) 363

Defaults and Rules Pages (Events)

Note: In the default event attributes of open message interface policies, you can set only the Category attribute.

In the default event attributes of SNMP trap policies, you can set only the Severity and Category attributes.

You can set the other event attributes within individual rules.

Event Attributes Tab

UI Element	Description
Title	Brief description of the nature of the event.
Description	Detailed description of the event.
Severity	Severity assigned to the event.
Time Created	*Database, Structured Log File, REST Web Service Listener, and XML File policies only:* Date and time when the event was created. Use the following conventions when specifying the date and time attribute: • **Integers.** BSM Connector interprets integers in the policy source as seconds since 00:00:00 UTC on 1 January 1970 (Unix time). For example, 1276600333 is 15 June 2010, at 11:12:13. • **Default time formats.** BSM Connector by default interprets the following time formats: `yyyy-mm-ddTHH:MM:SS` (for example, 2010-06-15T11:12:13) `mm/dd/yyyy HH:MM:SS` (for example, 06/15/2010 11:12:13) Additional time zone formats are supported: `yyyy-mm-ddTHH:MM:SS tz` (for example, 2010-06-15T11:12:13 +3) `mm/dd/yyyy HH:MM:SS tz`(for example, 06/15/2010 11:12:13 -2) where `tz` is a number for the offsett to the UTC time zone. You can also use half or quarter hours (.25, .5, or .75, for example, 2010-06-15T11:12:13 -2.75). If you want to create your own pattern, you can store the time zone information in <@.tz> to match all above mentioned time zones. • **Pattern matching.** You can use the function `<$DATETIME(FORMAT,VALUE)>` to specify a pattern (`FORMAT`) that matches the time string in the policy source (`VALUE`). You can use standard pattern-matching rules when matching values. By default, pattern matching for the time format is case sensitive. The default field separators are the space and the tab characters. `FORMAT` must be enclosed in quotation marks (`"FORMAT"`) and accepts the following variables: `H` (hours), `M` (minutes), `S` (seconds). If H, M, or S is not set, the hour, minute, or second displays as zero. `d` (day), `m` (month), `y` (year). If d or m is not set, the day or month display as one. If y is not set, the current year is assumed. If y is less than 100, the current millennium is assumed; for example, if y matches 10, the year displays as 2010. It is not possible to match a year earlier than 1970. `p` (P.M.) If p is set, BSM Connector adds 12 hours to the hours that precede the variable.

UI Element	Description
	VALUE is one of the following: ◦ **Database policies.** VALUE is the table column or value to match. Table columns use the following syntax: <$DATA:<*table_column*>> where <*table_column*> is the name of the table column in the third-party database, for example <$DATA:severity>. Examples: To match the time format 06/15/2010 11:12:13 in the table column <$DATA:/BSMConnectorEvent/TIMERECEIVED>, type <$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M>:<#.S>$",<$DATA:/BSMConnectorEvent/TIMERECEIVED>)> To match the time format 11:12 15.06.2010 in the table column <$DATA:/BSMConnectorEvent/TIMERECEIVED>, type <$DATETIME("^<#.H>:<#.M> <#.d>.<#.m>.<#.y>$",<$DATA:/BSMConnectorEvent/TIMERECEIVED>)> To match the time format 06/15/2010 1:35 PM in the table column <$DATA:/BSMConnectorEvent/TIMERECEIVED>, type <$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M> <2*.p>$",<$DATA:/BSMConnectorEvent/TIMERECEIVED>)> ◦ **Structured Log File policies.** VALUE is the input data reference value to match. Input data references use the following syntax: <$DATA:<*InputReferenceField*>> Field values originate from the structured log file pattern settings defined for the data source. Examples: To match the time format 06/15/2010 11:12:13 in the input data reference <$DATA:timestamp>, type <$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M>:<#.S>$",<$DATA:timestamp>)> To match the time format 11:12 15.06.2010 in the input data reference <$DATA:timestamp>, type <$DATETIME("^<#.H>:<#.M> <#.d>.<#.m>.<#.y>$",<$DATA:timestamp>)> To match the time format 06/15/2010 1:35 PM in the input data reference <$DATA:timestamp>, type <$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M> <2*.p>$",<$DATA:timestamp>)> ◦ **REST Web Service Listener policies.** VALUE is the XML property or

UI Element	Description
	value to match. XML properties use the following syntax: <$DATA:/*<XMLProperty>*> *<XMLProperty>* is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters. Examples: To match the time format `06/15/2010 11:12:13` in the XML property `<$DATA:/BSMCEvent/timeStamp>`, type `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M>:<#.S>$",<$DATA:/BSMCEvent/timeStamp>)>` To match the time format `11:12 15.06.2010` in the XML property `<$DATA:/BSMCEvent/timeStamp>`, type `<$DATETIME("^<#.H>:<#.M> <#.d>.<#.m>.<#.y>$",<$DATA:/BSMCEvent/timeStamp>)>` To match the time format `06/15/2010 1:35 PM` in the XML property `<$DATA:/BSMCEvent/timeStamp>`, type `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M> <2*.p>$",<$DATA:/BSMCEvent/timeStamp>)>` ◦ **XML File policies.** `VALUE` is the XML property or value to match. XML properties use the following syntax: <$DATA:/*<XMLProperty>*> *<XMLProperty>* is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters. Examples: To match the time format `06/15/2010 11:12:13` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M>:<#.S>$",<$DATA:/SCOM_Alert/TimeRaised>)>` To match the time format `11:12 15.06.2010` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type `<$DATETIME("^<#.H>:<#.M> <#.d>.<#.m>.<#.y>$",<$DATA:/SCOM_Alert/TimeRaised>)>` To match the time format `06/15/2010 1:35 PM` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M> <2*.p>$",<$DATA:/SCOM_Alert/TimeRaised>)>` If you leave the attribute empty or if none of the time formats above can be matched, then the date and time when the agent created the event appears in OMi. This time always appears using the time zone of the agent at creation time (for example, 11:30 (CET/winter). This means that this time always appears in this fixed time zone.

UI Element	Description
Category	Name of the logical group to which the event belongs (for example, Database, Security, or Network). The event category is similar in concept to the HP Operations Manager message group.
Subcategory	Name of the logical subgroup (category) to which the event belongs (for example, Oracle (database), Accounts (security), or Routers (network)).
ETI	Contains the event type indicator (ETI) resolution hint, which OMi uses to associate the event with an ETI and for event correlation. Use the format `<ETI name>:<ETI state>:<metric value>`. Specify the name of the indicator (for example, CPULoad), the indicator state (for example, High), and, optionally, the metric value (for example, 80). When OMi receives an event with an ETI resolution hint of CPULoad:High, and the ETI and state exist, the Event Type Indicator attribute is set to CPULoad:High in the event. The metric value is optional and serves informational purposes only.
Node	Name of the system where the event occurred (for example, node.example.com).

Related CI	Contains the CI that is related to the metric (for example, oraclesid01@@node.example.com or C:@@server.example.com). Use the format `<CI 1>:<CI 2>:...:<CI n>@@<hostname>`.
	
	Best practices for related CIs
	
	It is necessary to differentiate between CIs that have a Composition relationship to a node, and those that do not have such a relationship:
	
	• For "hosted on" CIs
	
	`<key attribute 1>:<key attribute 2>:<key attribute n>@@<hostname>`
	
	Typically, a "hosted on" CI is a sub-type of "Running Software". For example, a CI of type `websphereas` has a Composition relationship to a node.
	
	• For virtual CIs
	
	`<key attribute 1>:<key attribute 2>:<key attribute n>`
	
	A virtual CI does not have a strong containment relationship (Composition relationship) to node.
	
	An example of a typical virtual CI type is cluster. This CI type does not have a strong containment relationship to a node.
	
	Tip: If you have problems resolving non-hosted CIs, provide the RTSM ID of the desired CI by using the format `UCMDB:<ci_uuid>`.
	
	For more information about CI resolution in OMi, see the *OMi User guide* or online help.

UI Element	Description
Sub Component	Information used to identify a subcomponent of a CI. This CI subcomponent is used to calculate an aggregated status within OMi's Service Health for selected CIs. If an HI is populated by events from multiple components, you can specify a component name in this field in order to ensure the correct calculation of the HI state. For example, if you have a Computer CI with two CPUs, `cpu #1` and `cpu #2`, events from both CPUs will be sent to the same `CPU Load` HI. By default, the events will override each other and create an incorrect HI state. To prevent this, you can populate Sub Component with values `"cpu #1"` and `"cpu #2"` which will cause the HI state to be calculated as an aggregated state between the two events.
Source CI	Contains the source related CI. For example, type the name and instance of the third-party system that provides events (for example, NNMi@@mgmt1.example.com or SCOM@@mgmt2.example.com). If you enter a source related CI, OMi tries to find the corresponding CI in the RTSM.
Source Event ID	ID of the event in the third-party system. This ID is required for synchronization of event changes with the source event. It also enables drilldown into the third-party system in the Event Browser in OMi. **Tip:** The file that the policy reads usually contains the source event ID. If you are working with sample data, you can drag the source event ID from the Sample Data tab and add it to the source event ID field.
Send with closed status (For the Open message interface, SNMP interceptor, and scheduled task policies)	Sets the event's lifecycle status to Closed before sending it to OMi.
Send with closed status (For the REST Web Service Listener, Database, Structured Log File, and XML File policies)	Sets the event's lifecycle status to Closed before sending it to OMi. Possible values: **yes** = Sets lifecycle status to Closed. **no** = Does not set the lifecycle status to Closed. **Default value:** empty (= no) **Tip:** Click ▸ and select **yes** or **no** from the menu.

Event Correlation Tab

Note: You cannot set the following attributes in the event defaults of open message interface and SNMP trap policies:

- Close Events with Key
- Suppress Deduplication on Server

You can set these event attributes within individual rules.

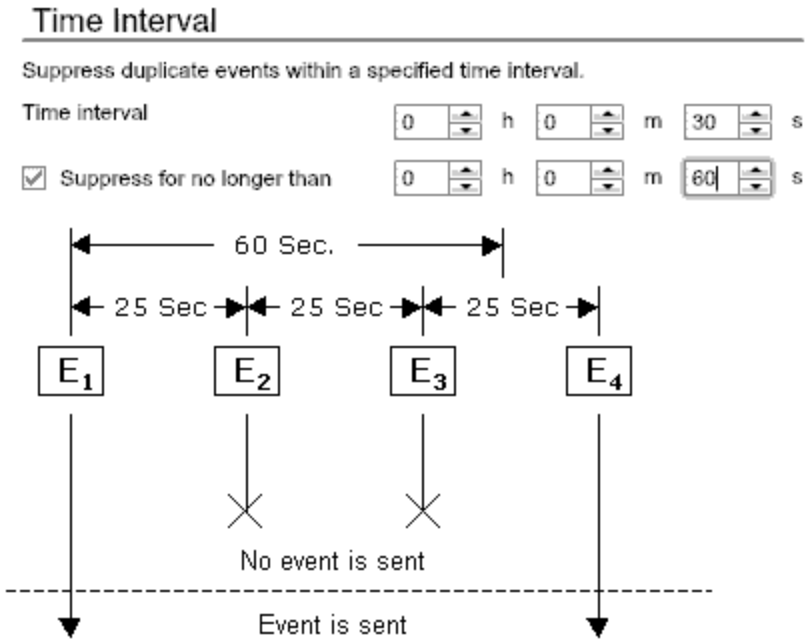
In scheduled task policies, you can set only the following event correlation attributes:

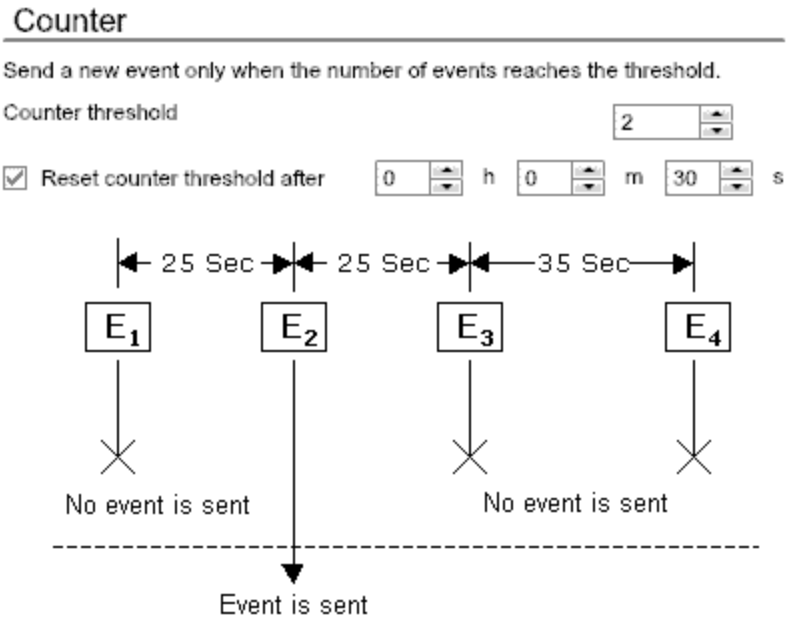
- Event Key
- Suppress Deduplication on Server

UI Element	Description
Event Key	An identifier used to identify duplicates and for Close Events with Key.

Close Events with Key	If events with the event key that you type here exist in the OMi event database when this event is received, these events are automatically closed. You can use pattern matching and variables to match multiple event keys. For example, consider the following pattern:
	
	`<$MSG_SEV>:<$MSG_NODE_NAME>`
	
	This pattern is evaluated by first replacing the variables with the values that they resolve to, for example:
	
	`critical:cabbage.example.com`
	
	This pattern is then compared using pattern matching rule against the event keys for all events in the OMi event database. Any key that you provide in the policy is treated as a simplified OM pattern in OMi. Therefore a plain string is treated as a substring and not as a complete match. The key in our example will match:
	
	`critical:cabbage.example.com:TEST`
	`critical:cabbage.example.com:TEST1`
	`critical:cabbage.example.com:TEST2A`
	
	and so on.
	
	To ensure that that the key matches only exact values, enclose the attribute value in an OM Pattern Expression, starting with ^ (start of line) and ending with $ (end of line), for example:
	
	`^critical:cabbage.example.com:TEST$`
	
	Note: An HP Operations Manager i Event Management Foundation License is required to enable the Close Events with Key feature in OMi.

UI Element	Description
Suppress Deduplication on Server	Stops automatic discarding of new events that are duplicates of existing events.
Event Suppression	
Suppress events which are	• **Generated by the same input event.** Select this option to suppress events that were sent in response to two separate input events that are identical except for the date and time that the event was generated (for example, identical entries in a log file). • **Generated by the same rule.** Select this option to suppress events that match the pattern specified for the selected rule. This is a more general setting for the suppression of duplicate events. For example, a policy might contain a rule with this match pattern: `Error Message<#>` The log file lines `Error Message10` and `Error Message20` are not identical, but would both match this rule. • **Identical relative to their attributes.** Select this option to suppress either events that have the same event key or (if no event key is present) events that have identical event attributes (except for the date and time that the event was generated).
Suppression Method	For event correlation, you can define one of three correlation methods: • **Time Interval.** This correlation method lets you define an interval during which duplicate events will be ignored. For more information, read this detailed example. **Time interval correlation example** In the illustration below, the interval is set to 30 seconds, but the suppression is limited to 60 seconds.

UI Element	Description
	 The E_x represents events that are identical. a. The first input event (E1) matches a rule in the policy. The policy sends an event and starts timing. b. A second matching event (E2) occurs 25 seconds later. This event occurred *less than 30* seconds after the first event, and is therefore suppressed. c. A third matching event (E3) occurs *less than 30 seconds after the second event*, and so is also suppressed. d. The next matching event (E4) occurs less than thirty seconds after the third event, but is also *more than 60 seconds after the first event*, and therefore the policy sends an event. • **Counter.** This correlation method counts the number of matching input events and sends an event only after the number of matching input events equals the counter threshold. The counter can also be reset to zero after a time period that you specify. For more information, read this detailed example. **Counter correlation example**

UI Element	Description
	 The E_x represent events that are identical. a. The first input event (E1) matches a rule in the policy, and the counter increments to one. No event is sent. b. A second matching event (E2) occurs, the counter increments to two, an event is sent, and the counter resets. c. A third matching event (E3), and the counter increments to one. No event is sent. d. The next matching event (E4) occurs *more than thirty seconds* after the third event. Since at thirty seconds the counter was reset to zero, the counter now increments to one. No event is sent. • **Time Interval/Counter.** If you use the Time interval and Counter together, events are evaluated first by the timer. If an event passes the timer, it is then evaluated by the counter, which either suppresses it or sends an event to OMi. **Note:** If you specify just time interval correlation or just counter-based correlation in an individual event, any event defaults for the other correlation method also apply. For example, if you specify time interval correlation for an event, and the event defaults specify counter-based correlation, the combined time interval and counter-based correlation applies to both new rules and existing rules. You can change this default behavior, so that only the correlation method that you specify in the individual event applies. To change the default behavior, set the parameter `OPC_IGNORE_DEFAULT_MSG_`

UI Element	Description
	`CORRELATION=TRUE` in the `eaagt` namespace on the node. You can configure this parameter using `ovconfchg` at a command prompt.
Time Interval	Time interval during which duplicate events are ignored.
Suppress for no longer than	Time interval after which duplicate events are no longer ignored.
Counter threshold	Value that triggers an event if met or crossed.
Reset counter threshold after	Time interval after which the counter is reset to 0.

Custom Attributes Tab

Note: You cannot set custom attributes in the event defaults of the following policy types:

- Open message interface policies
- SNMP trap policies

You can set custom attributes within individual rules for these policy types.

UI Element	Description
	Create New Custom Attribute: Creates a new custom attribute.
	Delete Custom Attribute: Deletes an existing custom attribute.

UI Element	Description

Name	The name of the custom attribute. The name is case-insensitive.
	Custom attributes are additional attributes that contain any information that is meaningful to you. For example, you might add a company name, contact information, or a city location to an event. You can have more than one custom attribute attached to a single event.
	The following custom attribute names cannot be used because they are reserved for internal use:
	`Description`
	`EtiHint`
	`HP_OPR_SAAS_CUSTOMER_ID`
	`NoDuplicateSuppression`
	`RelatedCiHint`
	`SourceCiHint`
	`SourcedFromExternalId`
	`SourcedFromExternalUrl`
	`SubCategory`
	`SubCiHint`

Value	Value of the custom attribute.

Advanced Tab

Note: You cannot set the following attributes in the event defaults of open message interface and SNMP trap policies:

- Event Drilldown URL
- Type

You can set these event attributes within individual rules.

UI Element	Description
Event Drilldown	

UI Element	Description
Event Drilldown URL	*Rules only for Open Message Interface and SNMP Interceptor Policies:* URL of the event in thethird-party system. This is the complete path of the URL, and includes the FQDN (fully qualified domain name) of the computer that hosts the third-party system, the communication port, and the root URL path (for example, `http://nnmi.example.com:8004/nnm/launch?cmd=showForm&objtype=Incident&objuuid=$OPC_CUSTOM[nnm.incident.uuid]&menus=true`). Event drilldown information enables OMi users to launch the user interface of the third-party system in the context of an event. **Tip:** To drill down to a specific event in the third-party system, add the source event ID to the URL. If you are working with sample data, you can drag the source event ID from the Sample Data tab and add it to the Event Drilldown URL field. **Note:** This event attribute can also be set by OMi based on a BSM Connector integration server configuration. If a policy and an integration server configuration both set this attribute, the information in the policy takes precedence.
OM Attributes	
Application	Application that caused the event to occur. Unlike the Related CI attribute, which is a direct relationship to a CI in the RTSM, the application attribute is a simple string-type attribute (for example, Oracle and OS).
Object	Device such as a computer, printer, or modem. Unlike the Related CI attribute, which is a direct relationship to a CI in the RTSM, the object attribute is a simple string-type attribute (for example, `C:`, and `/dev/spool`).
Type	*Rules only for Open Message Interface and SNMP Interceptor policies:* String used to organize different types of events within an event category or subcategory (for example, users or applications, accounts and security). The attribute is automatically set to `BSMC_Message`. You can delete the value but it will be inserted when you save the policy.
HPOM Service ID	ID of the service associated with the event. A service ID is a unique identifier for a service and can be used in OMi to identify the node and CI associated with the event.
Agent MSI	*Rules only for Open Message Interface and SNMP Interceptor policies:* Indicates the agent MSI settings configured in the event defaults: **Agent MSI not enabled.** Output to the agent MSI is not configured in the event defaults. **Divert events.** The Divert events setting is configured in the event defaults. **Copy events.** The Copy events setting is configured in the event defaults.

UI Element	Description
Agent MSI	*Rules only for the Open Message Interface and SNMP Interceptor policies:* The message stream interface (MSI) allows external applications to interact with the internal event flow of HP Operations Agent. The external application can be a read-write application, for example, an event processing program that can read events, modify attributes, and generate new events for retransmission to the server. The application could also read events, or send its own events.
Divert events	If Agent MSI is enabled, diverts an event to the MSI instead of to the server when an event is requested by an external application.
Copy events	If Agent MSI is enabled, sends the event to the server, and a copy of the event to the MSI.

Condition Tab - Rules Only

The following policy types use this condition tab:

- Database policies
- Structured log file policies
- REST Web service listener policies
- XML file policies

UI Element	Description
	New Item. Creates a new condition with the default operator equals.
	Delete Item. Deletes the selected condition.
	Move Up. Moves the selected condition higher in the condition order.
	Move Down. Moves the selected condition lower in the condition order.
	Expand. Expands the list of conditions to display all details.
	Collapse. Collapses the list of conditions to display only the names and hide the details.
	Click to expand the details of a condition.
	Click to hide the details of a condition.

UI Element	Description
Property	*Database policies:* Input data property that the policy searches for (for example, `/BSMConnectorEvent/Severity`). *Structured Log File policies:* Input data property that the policy searches for (for example, `host`). *REST Web Service Listener policies:* XML Property that the policy searches for. You must prefix the XML property with `<XML Event/Metric Tag>` (`/<XML Event/Metric Tag>/<full path to XML Property>`). *XML File policies:* XML Property that the policy searches for. You must prefix the XML property with `<XML Event/Metric Tag>` (`/<XML Event/Metric Tag>/<full path to XML Property>`).
Operator	The following operators are available: • equals • not equals • less than • greater than • less or equal • greater or equal • matches (Enables you to enter a pattern in the Operand field.)
Operand	Value or pattern that you want the policy to compare with the table column (database policies), or the input data reference (structured log file policies), or the value of the XML property (REST Web service listener policies and XML policies). If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.

Condition Tab - Open Message Interface Policy Rules Only

UI Element	Description
Node	Fully qualified domain name, node name, or IP address that the policy compares with the node in the source message. Separate multiple entries with the OR operator (|) or leave blank to match all nodes. This field corresponds to the `node` option of the opcmsg command.

UI Element	Description
Message Group	Message group that the policy compares with the message group in the source message. Separate multiple entries with the OR operator (|) or leave blank to match all message groups. This field corresponds to the `msg_grp` option of the opcmsg command.
Application	Application that the policy compares with the application in the source message. Separate multiple entries with the OR operator (|) or leave blank to match all applications. This field corresponds to the `application` option of the opcmsg command.
Object	Object that the policy compares with the object in the source message. Separate multiple entries with the OR operator (|) or leave blank to match all objects. This field corresponds to the `object` option of the opcmsg command. **Note:** Although the term *application* generally refers to a general program name and *object* generally refers to a process or sub-program, you should use these values to assist your own organizational scheme.
Severity	Severity that the policy compares with the severity in the source message. At least one severity must be selected. This field corresponds to the `severity` option of the opcmsg command.
Message Text	Message text or pattern that the policy compares with the message text in the source message.

Condition Definition Tab - SNMP Interceptor Policy Rules Only

UI Element	Description
Node	FQDN (Fully Qualified Domain Name), the primary node name, or the IP address of the configuration item for which you want to forward events. If you only want to match SNMP events from a specific configuration item, type the FQDN (Fully Qualified Domain Name), the primary node name, or the IP address. Give multiple entries with the **OR** operator (for example, `celery.example.com|broccoli.example.com`), or leave blank for all configuration items.

Event Object ID	Complete Event Object Identifier for the SNMP trap that you want to match. For example: `.1.3.6.1.4.1.11.2.17.1.0.40000001`
SNMPv1 notation	If selected, you can specify only part of the identifier rather than the complete event object ID. For example, by specifying only the Enterprise ID, you can match all events with a specific Enterprise ID.
Enterprise ID	Enterprise ID for incoming SNMP traps to be compared with this condition. The enterprise ID is a vendor-specific identifier for the trap. Standard BSM Connector pattern-matching syntax may not be used in this field; however, it is possible to match a range of objects by entering only a prefix. For instance, the pattern: .1.3.6.1.4.1.11.2.17 would match: .1.3.6.1.4.1.11.2.17.1 .1.3.6.1.4.1.11.2.17.2 and so on.
Generic ID	Generic Trap ID. Possible values are: • (0) **ColdStart** • (1) **WarmStart** • (2) **LinkDown** • (3) **LinkUp** • (4) **Authentification** • (5) **EgpNeighborLoss** • (6) **EnterpriseSpecific** • (7) **don't care** If you select **(6) EnterpriseSpecific**, you can type in the specific trap ID. Select **don't care** to intercept any kind of trap.
Specific ID	Type in the specific trap ID if you have selected **(6) EnterpriseSpecific** in Generic ID. Enterprise-specific SNMP traps can be implemented by vendors on their specific network devices. The specific trap ID is used to identify the source of the trap.

Note: The SNMP syntax used by the editor requires that the trap string begins with a point.

Condition Variable Bindings Tab - SNMP Interceptor Policy Rules Only

UI Element	Description
	Creates a new variable binding.
	Deletes the selected variable binding.
	Opens the Variable Bindings Options page.
Variable	Variable binding you want the policy to read. 1 represents the first variable binding in the event, 2 the second variable, and so on. You do not need to prefix the variable with a dollar sign ($); BSM Connector does this automatically.
Pattern	Match pattern for the binding. You can click the ▸ button to open the pattern matching expression toolbox. **Tip:** For matching patterns, you can use standard pattern-matching rules of HP Operations Agent. Select the matches operator and click the ▸ icon in the Operand field to open the pattern matching toolbox window. The toolbox includes the following sections: • **Pattern Matching Expressions.** Click an expression to insert it into the Operand text box. • **Variable Bindings Options.** Variable binding options include the setting of case sensitivity check and the field separators used in the rule. If you do not specify the pattern matching options for the rule, either the defaults (enabled case sensitivity check; the space and the tab character as the separators) or the default options set for the policy are used.

Defaults and Rules Pages (Metrics)

Default Metric Attributes — Basic Tab

UI Element	Description
Data domain	The namespace of the integrated performance records, used in the Operations Agent store to avoid clashes. **Example:** "BSMConnector"

UI Element	Description
Metric class	Defines the metric class. Metric class and metric name are concatenated as the metric name that appears in the Operations Agent store and consumers. **Example:** "Windows CPU Monitor"
Metric name	Defines the metric name. Metric class and metric name are concatenated as the metric name that appears in the Operations Agent store and consumers. **Example:** "CPU Utilization"

Related CI	Contains the CI that is related to the metric (for example, oraclesid01@@node.example.com or C:@@server.example.com). Use the format `<CI 1>:<CI 2>:...:<CI n>@@<hostname>`.
	Best practices for related CIs
	It is necessary to differentiate between CIs that have a Composition relationship to a node, and those that do not have such a relationship:
	• For "hosted on" CIs
	`<key attribute 1>:<key attribute 2>:<key attribute n>@@<hostname>`
	Typically, a "hosted on" CI is a sub-type of "Running Software". For example, a CI of type `websphereas` has a Composition relationship to a node.
	• For virtual CIs
	`<key attribute 1>:<key attribute 2>:<key attribute n>`
	A virtual CI does not have a strong containment relationship (Composition relationship) to node.
	An example of a typical virtual CI type is cluster. This CI type does not have a strong containment relationship to a node.
	Tip: If you have problems resolving non-hosted CIs, provide the RTSM ID of the desired CI by using the format `UCMDB:<ci_uuid>`.
	For more information about CI resolution in OMi, see the *OMi User guide* or online help.

UI Element	Description
Node	Used to identify a node-like CI to which the performance records are associated to. **Example:** "dbsys1.company.com"
Value	The actual performance value. It is automatically converted to a double-precision number. **Example:** 80
Time measured	The time stamp when the value was determined in the third-party system, expressed in the following formats: • **Integers.** BSM Connector interprets integers in the policy source as seconds since 00:00:00 UTC on 1 January 1970 (Unix time). For example, 1276600333 is 15 June 2010, at 11:12:13. Use the $DATETIME function to convert it. For details about the function, see "Time Created" on page 300. • **Default time formats.** BSM Connector by default interprets the following time formats: `yyyy-mm-ddTHH:MM:SS` (for example, 2010-06-15T11:12:13) `mm/dd/yyyy HH:MM:SS` (for example, 06/15/2010 11:12:13) Additional time zone formats are supported: `yyyy-mm-ddTHH:MM:SS tz` (for example, 2010-06-15T11:12:13 +3) `mm/dd/yyyy HH:MM:SS tz`(for example, 06/15/2010 11:12:13 -2) where `tz` is a number for the offsett to the UTC time zone. You can also use half or quarter hours (.25, .5, or .75, for example, 2010-06-15T11:12:13 -2.75). If you want to create your own pattern, you can store the time zone information in <@.tz> to match all above mentioned time zones.

Default Metric Attributes — Advanced Tab

UI Element	Description
Original metric name	The name of the metric as used on the third-party system. **Example:**
Unit	The unit of the metric values. **Example:** "MB"
Integration id	An ID, used to identify the source of the integration. **Example**: "DB-ORA"

Condition Tab - Rules Only

This condition tab is used by all metric policies.

UI Element	Description
	New Item. Creates a new condition with the default operator equals.
	Delete Item. Deletes the selected condition.
	Move Up. Moves the selected condition higher in the condition order.
	Move Down. Moves the selected condition lower in the condition order.
	Expand. Expands the list of conditions to display all details.
	Collapse. Collapses the list of conditions to display only the names and hide the details.
	Click to expand the details of a condition.
	Click to hide the details of a condition.

UI Element	Description
Property	*Database policies:* Input data property that the policy searches for (for example, `/BSMConnectorEvent/Severity`). *Structured Log File policies:* Input data property that the policy searches for (for example, `host`). *REST Web Service Listener policies:* XML Property that the policy searches for. You must prefix the XML property with `<XML Event/Metric Tag>` (`/<XML Event/Metric Tag>/<full path to XML Property>`). *XML File policies:* XML Property that the policy searches for. You must prefix the XML property with `<XML Event/Metric Tag>` (`/<XML Event/Metric Tag>/<full path to XML Property>`).
Operator	The following operators are available: • equals • not equals • less than • greater than • less or equal • greater or equal • matches (Enables you to enter a pattern in the Operand field.)
Operand	Value or pattern that you want the policy to compare with the table column (database policies), or the input data reference (structured log file policies), or the value of the XML property (REST Web service listener policies and XML policies). If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.

Indicators Tab

UI Element	Description
	Refresh. Loads the configured indicators from the connected OMi server. **Note:** • Loading indicators from the OMi server may take a few seconds. • The BSM Connector server must be configured as a BSM Connector integration server in OMi for the indicators to load successfully.

UI Element	Description
<Search …>	Entered search string is used to search the indicators and highlight only the indicators containing the specified string. To search for indicators with specific text strings in the name, type the string in the <Search …> field and click the button. The first matching indicator is selected in the list of rules. Click the and buttons to move to the previous and next matching indicator.
<Indicators>	Hierarchy of configuration item types with associated health indicators (HIs), which are applicable for the event integration only, and event type indicators (ETIs). To insert an indicator with a state in a policy, drag and drop the indicator from the Indicators tab to the relevant field in the policy.

Mappings Page (Events and Metrics)

UI Element	Description
	Create new mapping definition. Adds a new mapping definition to the list of mappings.
	Delete mapping definition. Deletes the selected mapping definition.
	Copy Mapping Definition. Creates a copy of the selected mapping definition.
	Move Up. Moves the selected mapping definition up to a higher position.
	Move Down. Moves the selected mapping definition down to a lower position.
Map Name	Name of the custom variable. BSM Connector automatically adds the default prefix `map` to the map name if the variable has been created from sample data.

Input Data Property	*Database policies:* Table column assigned to the custom variable. BSM Connector replaces the table column at runtime with the value of the specified column. If you insert a value, the value will be used. *Structured Log File policies:* Input data property assigned to the custom variable. BSM Connector replaces the pattern matching field at runtime with the value of the specified field. If you insert a value, the value will be used. *REST Web Service Listener policies:* XML property assigned to the custom variable. BSM Connector replaces the XML property at runtime with the value of the specified XML property. If you insert a value, the value will be used. *XML File policies:* XML element or attribute assigned to the custom variable. XML properties use the following syntax: <$DATA:/*<XMLProperty>*> *<XMLProperty>* is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters. BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used. *Perl Script policies:* Data key assigned to the custom variable. Perl attribute key names use the following syntax: <$DATA:<*attribute_name*>> where <*attribute_name*> is the data key name in a Perl hash array. BSM Connector replaces the attribute name at runtime with the value of the specified key.
	Create new mapping. Adds a new pair of source and target values to the mapping definition.
	Delete mapping. Deletes the selected source and target value pair.
	Copy Value Mapping. Creates a copy of the selected value mapping.
	Move Up. Moves the selected value mapping up to a higher position.
	Move Down. Moves the selected value mapping down to a lower position.

Source Value	*Database policies:*Original value of the table column. *Structured Log file policies:*Original value of the input data reference. *REST Web Service Listener policies:*Original value associated with the XML property from the XML file. *XML File policies:*Original value of the XML element or attribute. *Perl Script policies:* Original value of the input data reference.
Target Value	*Database policies:*New value of the table column. *Structured Log File policies:*New value of the input data reference. *REST Web Service Listener policies:*New value associated with XML property. *XML File policies:*New value of the XML element or attribute. *Perl Script policies:* New value of the input data reference.

Mappings Page (Generic Output)

UI Element	Description
	Create new mapping definition. Adds a new mapping definition to the list of mappings.
	Delete mapping definition. Deletes the selected mapping definition.

Input Field Qualifier	*Database policies:* Table column assigned to the custom variable. BSM Connector replaces the table column at runtime with the value of the specified column. If you insert a value, the value will be used. *Structured Log file policies:* Input data property assigned to the custom variable. BSM Connector replaces the pattern matching field at runtime with the value of the specified field. If you insert a value, the value will be used. *REST Web Service listener policies:* XML property assigned to the custom variable. BSM Connector replaces the XML property at runtime with the value of the specified XML property. If you insert a value, the value will be used. *XML File policies:* XML element or attribute assigned to the custom variable. XML properties use the following syntax: <$DATA:/*<XMLProperty>*> *<XMLProperty>* is the path from the root XML element of XML data to a specific XML element or attribute within that data. XML path uses slashes (/) as the path delimiters. Only the leaf node is eligible for mapping. BSM Connector replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used. *Perl Script policies:* Data key assigned to the custom variable. Perl attribute key names use the following syntax: <$DATA:<*attribute_name*>> where <*attribute_name*> is the data key name in a Perl hash array. BSM Connector replaces the attribute name at runtime with the value of the specified key. If you are working with meta data, you can drag and drop a key from the **Meta Data** tab.
Eligible Field Name	The field that is eligible for mapping. The field is extracted automatically from the Input Field Qualifier. Some sources integrate hierarchical data (XML file, REST WS, and log file). In case of such data, any field to be mapped must be a leaf node in the internal logical tree-like structure.
Mapped Field Name	The replacement for the input field.

<table>
<tr><td>Keep All Input Fields</td><td>If the option is selected, all input is kept regardless if the mapping does occur or not. If the option is not selected, the fields that are not mapped are dropped.

In case of hierarchical data, such as XML, all ancestry of a mapped key field is kept as well so that structure is maintained. For example, if you define the mapping shown in the table below unchecking the option to keep all input fields:

<table><tr><th>Input field qualifier</th><th>Replacement</th></tr><tr><td>/test_event</td><td>Event</td></tr><tr><td>/test_event/event_info</td><td>eventDetails</td></tr></table>
The outgoing key fields are:

/Event

/Event/eventDetails</td></tr>
<tr><td> </td><td>New item Adds a new additional, user defined field.</td></tr>
<tr><td> </td><td>Delete item. Deletes the additional field.</td></tr>
<tr><td>Field Name</td><td>The name of the additional user defined field.</td></tr>
<tr><td>Field Value</td><td>The value of the additional user defined field. The value can contain static text defined by the user and data references into the input data (<$DATA:…>).</td></tr>
<tr><td colspan="2">Meta Data tab</td></tr>
<tr><td>Input Data Properties</td><td>A list of keys derived from input data. You can drag and drop the keys.</td></tr>
</table>

Mappings Tab

UI Element	Description
<Mappings>	Displays the mapping definitions configured for the policy.

Metric Page (Data Forwarding)

Metric Data Forwarding Rules

UI Element	Description

	Adds a new data forwarding rule to the list of rules.
	Deletes the selected data forwarding rule.
Description	A short description of the data forwarding rule.
Rule Type	Displays the type of the rule selected for this target.
Data Forwarding Target	Displays the URL of the target.

Properties Tab

UI Element	Description
Description	Describes the purpose of the rule.
Rule type	Defines the rule type. The following types are available: • Forward on matched • Discard on matched • Discard on unmatched

Condition Tab

UI Element	Description
Property	Input data property. You can drag and drop a value from the **Meta Data** tab.
Operator	The following operators are available: • equals • not equals • less than • greater than • less or equal • greater or equal • matches
Operand	A numeric value between 0 and 65536.

Meta Data tab

UI Element	Description

<list of available keys>	Displays the available keys either from the BSM Connector Metric Meta Model (when forwarding metric data) or keys derived directly from the input data (when forwarding structured input data). You can drag and drop the keys in to the **Property** field.

Targets Tab

UI Element	Description
Affect all targets	If selected, the rule affects all configured targets.
<list of targets>	Lists all targets that are configured on the **Targets** page. Click on a target to enable it.
Override Target Configuration	Overrides the target configuration for selected targets.
Wire format	Selects the wire format - XML or JSON.
Use Guaranteed Delivery	If selected, guaranteed delivery is used.

Operators Tab (Metrics Only)

Expressions and Functions	Description
<$MATCH (*inputStr* , *patternDef* , *outputStr* ,*outputStrNoMatch*)>	Tests a string or variable against a pattern. The $MATCH function accepts three or four parameters: - the input string - the pattern definition - the output string if pattern matches on the input string - the output string if the pattern does not match (optional) **Example:** The data of the input field hostname start always with "TEST" (for example "TESTABC"). The $MATCH function to use the string after "TEST" is: `$MATCH(<$DATA:hostname>,TEST<*.prefix>,<prefix>)`
<$DATETIME(FORMAT,VALUE)>	Converts regular dates to UNIX time (Epoch time).

Options Page (Events only)

UI Element	Description
Log Local Events	Defines which incoming events are logged. These events are logged on the BSM Connector system in the log file: Windows: `%OvDataDir%\log\OpC\opcmsglg` Linux: `/var/opt/OV/log/OpC/opcmsglg`
that match a rule and trigger an event	Logs any events in the event source that match the policy rules.
that match a rule and are ignored	Logs any events in the event source that are suppressed. (Suppressed events are not sent to OMi.)
that do not match any rule	Logs any events that do not match any of the rules in the policy.
Unmatched Events	Send an event to OMi when the input event does not match any rule in the policy because none of the conditions apply or because the policy does not contain any rules. This ensures that unexpected events that might be important do not go unreported. By default, unmatched events are ignored. Each policy that sends unmatched events to OMi creates an event with the default values of the policy. **Tip:** If you want a policy to send events only with the default values, omit all rules from the policy. **Note:** Open message interface and SNMP trap policies: The agent creates an event for an unmatched event only if the input event is unmatched in all policies on the BSM Connector system. The agent sends only one event for each unmatched input event. Database, structured log file, REST Web service listener, and XML file policies: If several event policies forward unmatched events to OMi, you could receive multiple events about a single input event.

UI Element	Description
are forwarded to OMi Server	Sends unmatched events to OMi.
are forwarded to OMi Server with state 'closed'	Sets the unmatched event's lifecycle status to Closed before sending it to OMi.
are ignored	Ignores unmatched events.
Store records that do not match any rule	BSM Connector stores events that do not match any rule and sends them to OMi.
Pattern Matching Options	Defines case sensitivity and field separators for all rules.
Case sensitive check	Defines whether the case (uppercase or lowercase) of a text string is considered when the pattern of a rule is compared with the source data. When switched on, a match only occurs if the use of uppercase and lowercase letters is exactly the same in both the source data and the pattern. This is the default setting.
Field Separators	Defines which characters should be considered to be field separators. Field separators are used in the pattern as separator characters for the rule condition. You can define up to seven separators, including these special characters: • \n New line (NL) • \t Horizontal tab (HT) • \v Vertical tab (VT) • \b Backspace (BS) • \r Carriage return (CR) • \f Form feed (FF) • \a Alert (BEL) • \\ Backslash (\) For example, if you wanted a backslash, an asterisk, and the letter A to define the fields in the event, you would type *A (with no spaces separating the characters). If you leave this box empty, the default separators (a blank and the tab character) are used by default. You can set case sensitivity and separator characters for individual rules in a policy by clicking the ▸ button in rule's match condition.

UI Element	Description
Apply to All	Applies the pattern matching options to all existing rules in a policy. This overwrites any modifications made to the pattern matching options in individual rules. If you change the pattern matching options and do not click Apply to all, they only apply to all new rules in a policy.

Options Page (Metrics only)

UI Element	Description
Log Local Events	Defines which incoming events are logged. These events are logged on the BSM Connector system in the log file: Windows: `%OvDataDir%\log\OpC\opcmsglg` Linux: `/var/opt/OV/log/OpC/opcmsglg`
that match a rule and trigger an event	Logs any events in the event source that match the policy rules.
that match a rule and are ignored	Logs any events in the event source that are suppressed. (Suppressed events are not sent to OMi.)
that do not match any rule	Logs any events that do not match any of the rules in the policy.

UI Element	Description
Unmatched Events	Send an event to OMi when the input event does not match any rule in the policy because none of the conditions apply or because the policy does not contain any rules. This ensures that unexpected events that might be important do not go unreported. By default, unmatched events are ignored. Each policy that sends unmatched events to OMi creates an event with the default values of the policy. **Tip:** If you want a policy to send events only with the default values, omit all rules from the policy. **Note:** Open message interface and SNMP trap policies: The agent creates an event for an unmatched event only if the input event is unmatched in all policies on the BSM Connector system. The agent sends only one event for each unmatched input event. Database, structured log file, REST Web service listener, and XML file policies: If several event policies forward unmatched events to OMi, you could receive multiple events about a single input event.
are forwarded to OMi Server	Sends unmatched events to OMi.
are forwarded to OMi Server with state 'closed'	Sets the unmatched event's lifecycle status to Closed before sending it to OMi.
are ignored	Ignores unmatched events.
Store records that do not match any rule	BSM Connector stores events that do not match any rule and sends them to OMi.
Pattern Matching Options	Defines case sensitivity and field separators for all rules.

UI Element	Description
Case sensitive check	Defines whether the case (uppercase or lowercase) of a text string is considered when the pattern of a rule is compared with the source data. When switched on, a match only occurs if the use of uppercase and lowercase letters is exactly the same in both the source data and the pattern. This is the default setting.
Field Separators	Defines which characters should be considered to be field separators. Field separators are used in the pattern as separator characters for the rule condition. You can define up to seven separators, including these special characters: • \n New line (NL) • \t Horizontal tab (HT) • \v Vertical tab (VT) • \b Backspace (BS) • \r Carriage return (CR) • \f Form feed (FF) • \a Alert (BEL) • \\ Backslash (\) For example, if you wanted a backslash, an asterisk, and the letter A to define the fields in the event, you would type *A (with no spaces separating the characters). If you leave this box empty, the default separators (a blank and the tab character) are used by default. You can set case sensitivity and separator characters for individual rules in a policy by clicking the ▸ button in rule's match condition.
Apply to All	Applies the pattern matching options to all existing rules in a policy. This overwrites any modifications made to the pattern matching options in individual rules. If you change the pattern matching options and do not click Apply to all, they only apply to all new rules in a policy.

Pattern Matching Variables Tab (Events and Metrics Only)

UI Element	Description
<variables>	Displays the user-defined variables configured in the Condition tab. For more information about assigning strings to variables, see "User-Defined Variables in Patterns" on page 288.

Policy Variables Tab

Policy Variables Tab for Database and REST Web Service Listener Policies (Events only)

Variable	Description
<$MSG_APPL>	Returns the name of the application associated with the input event that caused the message. Sample output: `/usr/bin/su(1) Switch User`
<$MSG_GEN_ NODE>	Returns the IP address of the node that sends the event. Sample output: `192.168.1.123`.
<$MSG_GEN_ NODE_NAME>	Returns the host name of the node that sends the event. Sample output: `node123.example.com`.
<$MSG_GRP>	Returns the default category of the event. Sample output: `Security`
<$MSG_ID>	Returns the unique identity number of the event, as generated by the HP Operations Agent. Note that identity numbers are not generated for suppressed messages. Sample output: `6e998f80-a06b-71d0-012e-0f887a7c0000`
<$MSG_NODE>	Returns the IP address of the node on which the original event took place. Sample output: `192.168.1.123`
<$MSG_NODE_ NAME>	Returns the name of the node on which the original event took place. This is the hostname that the agent resolves for the node. This variable is not fixed, however, and can be changed by a policy on a per-event basis. For example, if the policy is receiving SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated. If the policy is reading a log file on a network share where applications on several nodes write messages, you could extract the name of the node from the error message, save it in a user-defined variable, and assign it to MSG_NODE_ NAME.
<$MSG_SERVICE>	Returns the service name associated with the event.
<$MSG_OBJECT>	Delivers the name of the object associated with the event.
<$MSG_SEV>	Returns the default value for the severity of the event. Sample output: `Normal`
<$MSG_TEXT>	Returns the full text of the event. For open message interface policies, this value is the msg_text parameter submitted by the opcmsg command. Sample output: `SU 03/19 16:13 + ttyp7 bill-root`

Variable	Description
<$MSG_TIME_CREATED>	Returns the time the message was created on the managed node in seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time. Sample output: 950008585
<$MSG_TYPE>	Delivers the name set for message type.

Policy Variables Tab for XML File and Structured Log File Policies (Events only)

Variable	Description
<$LOGFILE>	Returns the name of the log file that contains the input event. Sample output: program_log.txt
<$LOGPATH>	Returns the name and path of the log file that contains the input event. Sample output: C:\temp\mylogfile\program_log.txt
<$MSG_APPL>	Returns the name of the application associated with the input event that caused the message. Sample output: /usr/bin/su(1) Switch User
<$MSG_GEN_NODE>	Returns the IP address of the node that sends the event. Sample output: 192.168.1.123.
<$MSG_GEN_NODE_NAME>	Returns the host name of the node that sends the event. Sample output: node123.example.com.
<$MSG_GRP>	Returns the default category of the event. Sample output: Security
<$MSG_ID>	Returns the unique identity number of the event, as generated by the agent. Note that identity numbers are not generated for suppressed messages. Sample output: 6e998f80-a06b-71d0-012e-0f887a7c0000
<$MSG_NODE>	Returns the IP address of the node on which the original event took place. Sample output: 192.168.1.123
<$MSG_NODE_NAME>	Returns the name of the node on which the original event took place. This is the hostname that the agent resolves for the node. This variable is not fixed, however, and can be changed by a policy on a per-event basis. For example, if the policy is receiving SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated. If the policy is reading a log file on a network share where applications on several nodes write messages, you could extract the name of the node from the error message, save it in a user-defined variable, and assign it to MSG_NODE_NAME.
<$MSG_SERVICE>	Returns the service name associated with the event.
<$MSG_OBJECT>	Delivers the name of the object associated with the event.

Variable	Description
<$MSG_SEV>	Returns the default value for the severity of the event. Sample output: Normal
<$MSG_TEXT>	Returns the full text of the event. For open message interface policies, this value is the msg_text parameter submitted by the opcmsg command. Sample output: SU 03/19 16:13 + ttyp7 bill-root
<$MSG_TIME_CREATED>	Returns the time the message was created on the managed node in seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time. Sample output: 950008585
<$MSG_TYPE>	Delivers the name set for message type.

Policy Variables Tab for Open Message Interface, Scheduled Task, and SNMP Interceptor Policies (Events only)

Variable	Description
<$#> (SNMP Only)	Returns the number of variables in an enterprise-specific SNMP event (generic event 6 Enterprise specific ID). Sample output: 2
<$*> (SNMP Only)	Returns all variables assigned to the event up to the possible fifteen. Sample output: [1] .1.1 (OctetString): arg1 [2] .1.2 (OctetString): turnip.example.com
<$@> (SNMP Only)	Returns the time the event was received as the number of seconds since Jan 1, 1970 using the *time_t* representation. Sample output: 859479898
<$1> (SNMP Only)	Returns one or more of the fifteen possible event parameters that are part of an SNMP event. (<$1> returns the first variable, <$2> returns the second variable, and so on.)
<$\>1> (SNMP Only)	Returns all attributes greater than *n* as *value* strings, useful for printing a variable number of arguments. <$\>0> is equivalent to $* without sequence numbers, names, or types. Sample output: bokchoy.example.com
<$\>+1> (SNMP Only)	Returns all attributes greater than *n* as *name:value* string. Sample output: .1.2: asparagus.example.com
<$+2> (SNMP Only)	Returns the nth variable binding as *name:value*. Sample output: .1.2: artichoke.example.com
<$\>-n > (SNMP Only)	Returns all attributes greater than *n* as *[seq] name (type): value* strings. Sample output: [2] .1.2 (OctetString): cauliflower.example.com
<$-2> (SNMP Only)	Returns the nth variable binding as *[seq] name-type:value*. Sample output: [2] .1.2 (OctetString): brusselsprouts.example.com

Variable	Description
<$A> (SNMP Only)	Returns the node that produced the event. Sample output: eggplant.example.com
<$C> (SNMP Only)	Returns the community of the event. Sample output: public
<$E> (SNMP Only)	Returns the enterprise ID of the event. Sample output: .1.3.6.1.4.1.11.2.17.1
<$e> (SNMP Only)	Returns the enterprise object ID. Sample output: .1.3.6.1.4.1.11.2.17.1
<$F> (SNMP Only)	Returns the textual name of the remote postmaster daemon's computer if the event was forwarded. Sample output: cress.example.com
<$G> (SNMP Only)	Returns the generic event ID. Sample output: 6
<$MSG_APPL>	Returns the name of the application associated with the input event that caused the message. Sample output: /usr/bin/su(1) Switch User
<$MSG_GEN_NODE>	Returns the IP address of the node that sends the event. Sample output: 192.168.1.123.
<$MSG_GEN_NODE_NAME>	Returns the host name of the node that sends the event. Sample output: node123.example.com.
<$MSG_GRP>	Returns the default category of the event. Sample output: Security
<$MSG_ID>	Returns the unique identity number of the event, as generated by the HP Operations Agent. Note that identity numbers are not generated for suppressed messages. Sample output: 6e998f80-a06b-71d0-012e-0f887a7c0000
<$MSG_NODE>	Returns the IP address of the node on which the original event took place. Sample output: 192.168.1.123
<$MSG_NODE_NAME>	Returns the name of the node on which the original event took place. This is the hostname that the agent resolves for the node. This variable is not fixed, however, and can be changed by a policy on a per-event basis. For example, if the policy is receiving SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated. If the policy is reading a log file on a network share where applications on several nodes write messages, you could extract the name of the node from the error message, save it in a user-defined variable, and assign it to MSG_NODE_NAME.
<$MSG_SERVICE>	Returns the service name associated with the event.
<$MSG_OBJECT> (Open Message Inteface and Scheduled Task Only)	Delivers the name of the object associated with the event.

Variable	Description
<$MSG_OBJECT> (SNMP Only)	Returns the name of the object associated with the event. This is set in the Event Defaults section of the policy editor.
<$MSG_SEV>	Returns the default value for the severity of the event. Sample output: Normal
<$MSG_TEXT>	Returns the full text of the event. For open message interface policies, this value is the msg_text parameter submitted by the opcmsg command. Sample output: SU 03/19 16:13 + ttyp7 bill-root
<$MSG_TIME_CREATED>	Returns the time the message was created on the managed node in seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time. Sample output: 950008585
<$MSG_TYPE>	Delivers the name set for message type.
<$N> (SNMP Only)	Returns the event name (textual alias) of the event format specification used to format the event, as defined in the Event Configurator. Sample output: OV_Node_Down
<$NAME> (Scheduled Task Only)	Returns the name of the policy that sent the event. Sample output: cpu_ util
<$O> (SNMP Only)	Returns the name (object identifier) of the event. Sample output: .1.3.6.1.4.1.11.2.17.1.0.58916865
<$o> (SNMP Only)	Returns the numeric object identifier of the event. Sample output: .1.3.6.1.4.1.11.2.17.1.0.58916865
<$OPTION(N)> (Open Message Interface Only)	Returns the value of an optional variable that is set by opcmsg (for example, <$OPTION(A)>, < $OPTION(B)>, and so on.).
<$PROG> (Scheduled Task Only)	Returns the name of the program executed by the scheduled task policy Sample output: check_for_upgrade.bat
<$R> (SNMP Only)	Returns the true source of the event. This value is inferred through the transport mechanism which delivered the event. Sample output: carrot.example.com
<$r> (SNMP Only)	Returns the implied source of the event. This may not be the true source of the event if the true source is proxying for another source, such as when an application running locally is reporting information about a remote node. Sample output: rutabaga.example.com
<$S> (SNMP Only)	Returns the specific event ID. Sample output: 5891686
<$s> (SNMP Only)	Returns the event's severity. Sample output: Normal
<$T> (SNMP Only)	Returns the event time stamp. Sample output: 0

Variable	Description
<$USER> (Scheduled Task Only)	Returns the name of the user under which the scheduled task was executed. Sample output: `administrator`
<$V> (SNMP Only)	Returns the event type, based on the transport from which the event was received. Currently supported types are SNMPv1, SNMPv2, CMIP, GENERIC, and SNMPv2INFORM. Sample output: `SNMPv1`
<$X> (SNMP Only)	Returns the time the event was received using the local time representation. Sample output: `17:24:58`
<$x> (SNMP Only)	Returns the date the event was received using the local date representation. Sample output: `03/27/10`

Policy Variables Tab for All Policy Types (Metrics only)

Variable	Description
<$MSG_GEN_NODE>	Returns the IP address of the node that sends the event. Sample output: `192.168.1.123`.
<$MSG_GEN_NODE_NAME>	Returns the host name of the node that sends the event. Sample output: `node123.example.com`.

Properties Page

UI Element	Description
Name	Name of the policy. You can use spaces in the name. The equal sign (=) is not allowed. **Tip:** Include the name of the integrated software in the policy name.
Description	Description of what the policy does. You might also add other notes (for example, data sources that are used).
Category	Name of the logical group to which the policy belongs. Categories may help you to better group your policies. Separate multiple categories with commas.
Policy ID	GUID (globally unique identifier) assigned to the policy when it is first created.

UI Element	Description
Last Modification	The date and time that the policy was saved. The date and time displays using the current time zone of the computer on which the Web browser runs. The language setting of the Web browser determines the date and time format (for example, `09/14/2010 8:16:38 AM` for English (United States)). If the Web browser and the computer on which the server run have different language settings, the language setting of the Web browser takes precedence. However, English is the default language if the Web browser is configured to use a language that is not available on the server.
Last Modified by	The name of the user active when the policy was saved.

Rules Page - Policy Rules

Policy Rules List

UI Element	Description
	Event policies: **Create New Rule:** Provides the following options: • **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used. • **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an event to OMi. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi. *Metrics policies:* **Create New Rule:** Provides the following options: • **Store on matched rule.** If matched, BSM Connector stores metrics in a buffer until a maximum number of records or a maximum amount of time are reached, when it sends them to OMi. The metrics use the settings defined for the rule. If you do not configure these settings, the default settings are used. • **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send metrics to OMi. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to OMi.
	Copy Rule. Copies the selected rule. You can then rewrite the description of the copied rule and edit the rule.
	Delete Rule. Deletes the selected rule.

UI Element	Description
	Move Up. Moves the selected rule higher in the rule order.
	Move Down. Moves the selected rule lower in the rule order.
<Move to>	Entered number is used to select the rule with that sequence number in the list of rules. To select a specific rule in the rule list, type the rule's sequence number in the <Move to> field and click the ▸ button.
<Search rules>	Entered search string is used to search the rule descriptions and highlight only the rules containing the specified string. To search for rules with specific text strings in the rule description, type the string in the <Search rules> field and click the 🔍 button. The first matching rule is selected in the list of rules. Click the ◂ and ▸ buttons to move the previous and next matching rule.
	Activate/Deactivate Rule Filter. Activates and deactivates the rule filter.
Seq.	Sequence number of the rules. Rules are evaluated in a specific order. When one condition is matched, no additional rules are evaluated.
Rule Description	Description of the rule. It is good practice to use a description that helps you remember what the rule does.
Rule Type	The three rule types of event policies are: • **Event on matched rule.** If matched, BSM Connector sends an event to OMi. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used. • **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send an event to OMi. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to OMi. The three rule types of metrics policies are: • **Store on matched rule.** If matched, BSM Connector stores metrics in a buffer until a maximum number of records or a maximum amount of time are reached, when it sends them to OMi. The metrics use the settings defined for the rule. If you do not configure these settings, the default settings are used. • **Suppress on matched rule.** If matched, BSM Connector stops processing and does not send metrics to OMi. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to OMi. You can change the rule type by clicking the current rule type in the list of rules and selecting another rule type from the drop-down list.

Sample Data Tab

Sample Data Tab - Database Policies

UI Element	Description
<Search Properties>	Entered search string is used to find a table column. The list changes as you type; only matching items appear. To clear the search results, click .
	Find Matching Events. To find values that belong to more than one table row, select the value and click . The Database Sample Data window opens and shows all rows that have the selected value.
	Toggle Deduplication. Shows or hides duplicate values.
Input Data Properties	Shows all columns that are returned by the database query. **Note:** The Sample Data tab is empty if no sample data has been loaded into the policy or if the database query did not succeed. See also "Configuring the Data Source in Database Policies" on page 129.
Values for <...>	Displays the values of the column selected in the Input Data Properties section.

Sample Data Tab - Structured Log File Policies

UI Element	Description
<Search Properties>	Entered search string is used to find a pattern matching field . The list changes as you type; only matching items appear. To clear the search results, click .
	Find Matching Events. To find values that belong to more than one pattern matching field , select the value and click . The Structured Log File Sample Data window opens and shows all pattern matching fields that have the selected value.
	Toggle Deduplication. Shows or hides duplicate values.

UI Element	Description
Input Data Properties	Shows all pattern matching fields that are extracted from the log file by using the OM pattern-matching language. **Note:** The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match the structured log file pattern specified in the source page. See also "Configuring Data Source in Structured Log File Policies" on page 238.
Values for <...>	Displays the values of the pattern matching field selected in the Input Data Properties section.

Sample Data Tab - REST Web Service Listener Policies

UI Element	**Description**
<Search Properties>	Entered search string is used to find an XML property. The list changes as you type; only matching items appear. To clear the search results, click .
	Toggle Short/Full Path Notation. Shows or hides the full path to the XML property. The full path starts with *<xml event/metric tag> /<path to xml property>* . The XML properties section by default shows the short path to the XML property.
	Find Matching Events. To find values that belong to more than one XML property, select the value and click . The Web Service Sample Data window opens and shows all XML properties that have the selected value.
	Toggle Deduplication. Shows or hides duplicate values.
XML Properties	Shows all XML properties that have been received by the REST Web service listener for this policy. **Note:** The Sample Data tab is empty if no sample data has been loaded into the policy or if no XML data has been received for the policy. For information about loading sample data into a REST Web service listener policy, see "Events and Metrics only: How to load sample data into the policy" on page 192.
Values for <...>	Displays the values of the XML properties selected in the XML Properties section.

Sample Data Tab - XML File Policies

UI Element	Description
<Search Properties>	Entered search string is used to find an XML property or value. The list changes as you type; only matching items appear. To clear the search results, click .
	Toggle Short/Full Path Notation. Shows or hides the full path to the XML property or value. The full path begins with the XML tag specified in the Source tab. The XML Properties section by default shows the short path to the XML property or value.
	Find Matching Events. To find values that belong to more than one XML property, select the value and click . The XML Sample Data window opens and shows all XML properties that have the selected value.
	Toggle Deduplication. Shows or hides duplicate values.
XML Properties	Shows all XML elements and attributes that match an XML tag. **Note:** The XML properties list is empty if no sample data has been loaded into the policy or if the sample data does not match any specified XML tags.
Values for <...>	Displays the values of the XML property selected in the XML Properties section.

Schedule Page

Schedule Page - Scheduled Task Policies

UI Element	Description
	Clears the selection.
	Select All. Selects all units of time.

UI Element	Description
Scheduling Options	The following options are available: • **Once.** When **Once** is selected, the command runs on one specific day at the time you indicate. **Note:** If the selected date or time occurs in the past, the command is not executed, and the Schedule tab shows a warning. • **Once per interval.** When **Once per interval** is selected, the command runs once each time the interval that you indicate passes. **Note:** Make sure the minimum value that you set is 15 seconds. If you set less than 15 seconds, the policy can be saved, but you will be notified about the wrong interval value by the scheduled policy editor. • **Advanced.** When **Advanced** is selected, you can indicate specific days and times when the command should be run. You select specific days of the week, specific days of the month, and specific months. This allows you to specify odd schedules such as, "On Monday when it falls on the 2nd of the month." You can also indicate that the command should only be run during a specific year. **Note:** If you select Advanced but then do not specify a schedule, the command by default runs every minute.
Once	
Set to current time	Selects the current time in the schedule.
Minute of Hour	0 to 59 minutes.
Hours of Day	1 to 12 AM and 1 to 12 PM.
Date: <>	Date when the command should run. Click the calendar icon to open a calendar view for the current month.
Once per interval	
Interval: <> d <> h <> m <> s	Interval in days, hours, minutes, and seconds. The interval must be bigger than 0 seconds.
Advanced (daily execution)	
Minute of Hour	0 to 59 minutes.

UI Element	Description
Hours of Day	1 to 12 AM and 1 to 12 PM.
Days of Month	1 to 31 days of the month.
Months of Year	Months from January to December.
Days of Week	Days of the week from Sunday to Saturday.
Restrict schedule to the year	Select to schedule the task for the specified year only.

Source Page

Source Page - Database Policies

UI Element	Description
Connection settings tab	
Classpath	The `jar` files that need to be loaded to use the JDBC driver class.
JDBC driver class	The driver used to connect to the database. Use the fully qualified class name (FQCN) of your JDBC driver.
Connect string	The URL to a database connection (referred to as an Connection string). For details about the connection string, see the driver documentation.
Username	User name used to log on to the database.
Password	Password used to log on to the database.
Polling interval	How often the policy queries the database (in days, hours, minutes, and seconds). **Default value:** 5 minutes **Minimum value:** 3 seconds Note that if the value is set to less than `15`, the policy cannot be saved.

UI Element	Description
Additional connection properties	Any additional connection properties needed by a driver. For details see the driver documentation. Click to add a new property. Type the name of the property and its value. To remove a property, select the property and click .
Collection settings tab	
SQL statement	The SQL query used to query the database. For details on how to write queries, see "How to configure the database source" on page 131. To retrieve sample data using the specified query, click . **Note:** BSM Connector can only load a maximum of 50 MB of sample data.
Session variables	A list of keys and initial values for them. These values are used in the data map if the initial value statement fails to load the values to the data map. Click to add a new field. Type the name of the key and its initial value. Alternatively, you can drag and drop entries from the Initial value sample data tab. To remove a key, select the key and click .
Initial value statement	An SQL statement that is executed in the init method during the policy activation. It can be used to initialize the data map with values. The values returned as last record from the execution of the statement are copied to the data map and returned. To retrieve sample data using the specified query, click . **Note:** BSM Connector can only load a maximum of 50 MB of sample data.
Internals tab	
Fetch size	Maximum number of rows the policy retrieves from the database for each cycle. **Default value**: 100 rows If the number of result rows exceeds the set maximum, the policy retrieves the remaining rows (those that exceeded the maximum) on future cycles, until all result rows are retrieved. The value should be sufficient to keep up with database table growth, yet small enough to avoid java.lang.OutOfMemoryException errors. Further, policy run frequency should also be considered. Make sure that the rate at which data is collected by the policy—which is dependent on both policy run frequency and network/system speed—is greater than, or equal to, the rate of data insertion on the third-party system.

UI Element	Description
Result size	Maximum number of rows that are collected on the receiver side before the entries are processed. By modifying this parameter together with the fetch size you can balance the loads on the database and the BSM Connector system. **Default value**: 100 rows
Data source tag	A user defined descriptive name for the collected data. This tag can be used to distinguish data sets generated by different policies. This may be useful in data forwarding scenario, for example for consumer applications that receive data from BSM Connector.

Source Page - Database Policies - Sample Data

UI Element	Description
<Search Properties>	Entered search string is used to find a table column. The list changes as you type; only matching items appear. To clear the search results, click .
	Find Matching Events. Use it to find values that belong to more than one Data input property.
	Toggle deduplication.

Source Page - Database Policies - Initial Value Sample Data

UI Element	Description
<Search Properties>	Entered search string is used to find a table column. The list changes as you type; only matching items appear. To clear the search results, click .
	Find Matching Events. Use it to find values that belong to more than one Data input property.
	Toggle deduplication.

Source Page - Structured Log File Policies (Events and Metrics Only)

UI Element	Description
Structured Logfile source	

Log File Path / Name	Path and name of the structured log file that the policy reads. **Note:** BSM Connector cannot process log files that are larger than 2 GB.
Polling Interval	Determines how often the policy reads the structured log file (in days, hours, minutes, and seconds). This period of time is the polling interval. The larger polling interval is set, the less performance is needed. However, more memory is used (this depends on the amount of the data in the log file). Setting the polling interval below 30 seconds is not recommended, the default setting is usually appropriate. Note that a policy begins to evaluate data *after* the first polling interval passes, unless the `Read from beginning (first time)` or `Read from beginning (always)` radio button is selected, which results in evaluation of the already existing data at the policy activation. A shorter polling interval is better when you are testing a policy. Default value: 5 minutes **Note:** Make sure that you set this value to a minimum of `15` seconds to be able to save the policy.
Logfile Character Set	Name of the character set used by the structured log file that the policy reads. **Note:** It is important to choose the correct character set. If the character set that the policy is expecting does not match the character set in the structured log file, pattern matching may not work, and the event details can have incorrect characters or be truncated in OMi. If you are unsure of which character set is used by the structured log file that the policy reads, consult the documentation of the program that writes the file. Default value: UTF-8
Send event if log file does not exist	BSM Connector sends an event if the specified structured log file does not exist. Default value: not selected

Close after reading	If you select this option, the file handle of the structured log file closes and reopens again after the polling interval. The file is read from the last position. If this file had a rollover in the meantime, it is read from the beginning. If the name of the structured log file changes, and a new file was started in the meantime, the policy continues to read the new structured log file and the original structured log file data is lost. If you do not select this option, the file handle remains and is read entirely each time, unless there is a newer file with the same name (or name pattern). In that case the original structured log file is read to the end, and then the newer file is read. Therefore, no data is lost. Consider the following example: a policy reads the structured log file `system.log`.While there is still some unread data in the `system.log` file, it is renamed to `system_Monday.log`, and a new version of the `system.log` file is written. In case this option is selected, the unread data from the `system_Monday.log` file remains unread and the `system.log` file is read entirely. In case this option is not selected, the unread data from the new `system.log` file is read after the reading of the `system_Monday.log` file is completed. Default value: not selected

<table>
<tr><td>Read Mode</td><td>The read mode of a structured log file policy indicates whether the policy processes the entire file or only new entries.

<table>
<tr><td>Read from last position. The policy reads only new—appended—entries written in the structured log file while the policy is activated. If the file decreases in size between readings, then the entire file is read. Entries that are added to the file when the policy is disabled are not processed by the policy.

Choose this option if you are concerned only with entries that occur when the policy is enabled.</td><td>Advantage: No chance of reading the same entry twice. (Unless the file decreases in size because some entries were deleted.)

Disadvantage: Entries written to file while the policy is disabled or the agent is not running are not processed by the policy.</td></tr>
<tr><td>Read from beginning (first time). The policy reads the complete structured log file each time the policy is activated or the agent restarts. This ensures that all entries in the file are compared with the rules in the policy. Each successive time that the policy reads the file, only new (appended) entries in the file are processed.

Choose this option if you want to ensure that every existing and future entry in the file is processed by the policy while it is activated.</td><td>Advantage: Every existing and future entry in the file will be processed by the policy.

Disadvantage: Duplicate entries can occur if an activated policy is deactivated and reactivated, or if the agent stops and restarts.</td></tr>
<tr><td>Read from beginning (always). The policy reads the complete structured log file every time it detects that the file has changed. The policy scans the file at the specified polling interval. If no change is detected, the file is not processed. Any entries overwritten while the agent is not running or the policy is deactivated will not be evaluated by the policy.

Choose this option if the policy reads a file that is overwritten, rather than appended.</td><td>Advantage: Ensures that files that are overwritten are correctly processed.

Disadvantage: Only valid for files that are overwritten, rather than appended.</td></tr>
</table>

Note: Every policy reads the same structured log files independently from any other policies. This means, for example, that if "Policy 1" with read mode Read from beginning (first time) is activated and "Policy 2" with the same read mode already exists, "Policy 1" still reads the entire file after it has been activated.

Default value: Read from last position</td></tr>
</table>

Sample Data	
	Loads the log file into BSM Connector. The log file can be loaded from Server or from the local file system. **Note:** BSM Connector can only load a maximum of 50 MB of sample data.

	Opens the Structured logfile sample data dialog box. This dialog box contains the following tabs:
	• Raw data
	Raw data tab displays the actual lines of the uploaded log file marked with the numbers. For example:
	`1380004749|tcpc113.RIESLING.INTERN|LogicalDisk|C:|% Free Space|66.379264831543|Microsoft.Windows.Server.2008.LogicalDisk`
	• Structured data
	Structured data tab displays the lines of the uploaded log file after the log file pattern is applied. These lines are now divided by the fields that were identified by the applied pattern. These field can be used throughout the rest of the policy configuration, for example, as Input Data Properties from the Sample Data in the Mappings and the Defaults tabs. For example, for the above stated example log line, these fields would be the following:
	`timestamp|hostname|entitytype|entityid|countername|countervalue|scomtype`

Logfile Structure	
Log File Pattern (for events only)	A pattern by which the log file's structure is extracted, and which will be used in all other policy operations.This pattern should comply with the standard pattern definition used by all HP Operations Managerproducts (OM pattern). For example, this structure could like as follows: `<*.timestamp>|<*.hostname>|<*.entitytype>|<*.entityid>|<*.countername>|<*.countervalue>|<*.scomtype>` For more information about how this structure is extracted from a log file, see "Configuring Data Source in Structured Log File Policies" on page 238. For more information about the OM pattern matching details, see "Pattern Matching in Policy Rules" on page 284.

Data Fields (for metrics only)	**identify using OM Pattern**	Identifying the log file's structure components by using the OM pattern. For example: `<*.timestamp>|<*.hostname>|<*.entitytype>|<*.entityid>|<*.countername>|<*.countervalue>|<*.scomtype>` This field can contain the data that matches the whole line or just the static part of the log line. For more information about the OM pattern matching details, see "Pattern Matching in Policy Rules" on page 284. **Note:** In case a value in the **identify using static fields** field is already specified, the value of this field is ignored.
	identify using static fields	Identifying the log file's structure components by using static fields. For example: `timestamp,hostname,entitytype,entityid,countername,countervalue,scomtype` For more information on static fields, see "Defining a log file structure by using static fields (Metric only)" on page 239. **Tip:** This is a recommended method (applies also for recurring fields) for extracting a structure from the log file due to performance reasons.
Recurring Fields (for metrics only)	A word list that contains the recurring part from the log line. Each recurrence creates a record in the store. For example: `countername,countervalue` For more information about the recurring fields, see "Using recurring fields in defining a log file structure (Metric only)" on page 239.	
Data Field Separator (for metrics only)	The separator that is used as a data separator in the log file.	
Line Start Indicator (for events only)		
Line Start Pattern	This field enables you to differentiate the structured log file entries based on their logical relationship, regardless of their span in the log file. You can do this by identifying a line start indicator from a log file, and then specifying the matched line start pattern by using the OM pattern matching language. For more information, see "Setting the Line Start Indicator (Event only)" on page 240.	

Source Page - REST Web Service Listener Policies

UI Element	Description
Path	A string entered in the policy. This string is a part of the actual URL where the REST Web Service Listener listens for the XML data. For example, for the events this URL should look as follows: `http(s)://<BSMC_FQDN>:<configured_port>/bsmc/rest/events/<PATH>` Only the following characters are allowed in a path: `[a-zA-Z0-9()-=*.?;,+/:&_]` For more information on customizing the REST Web Service Listener, see "Configuring the Data Source in REST Web Service Listener Policies " on page 189.
Character Set	Name of the character set used by the XML data that the policy reads. **Note:** It is important to choose the correct character set. If the character set that the policy is expecting does not match the character set in the XML data, pattern matching may not work, and the event details can have incorrect characters or be truncated in OMi. If you are unsure of which character set is used by the XML data that the policy reads, consult the documentation of the program that writes the file. Default value: UTF-8
Sample Data (Events and Metrics only)	
	Uploads the XML file to the BSM Connector. **Note:** BSM Connector can only load a maximum of 50 MB of sample data.
	Opens the REST Web Service Sample Data dialog box. This dialog box displays the XML elements and values contained in the uploaded XML sample data.
XML Event Tag (Events only) Enables you to specify one or more XML event tags. The XML event tag creates a shortcut to the XML element that you want to process. An event tag typically identifies an event record in an XML data. You can define more than one event tag.	
	Create a new XML event tag manually. Enables you to type an XML element in the provided box. **Create a new XML event tag from XML sample data.** Opens the XML Sample Data Outline dialog box. This dialog box displays the XML elements and attributes contained in the uploaded XML sample data.

UI Element	Description
	Deletes the selected XML event tag. **Caution:** Deleting an event tag that is referenced in a policy corrupts the policy and renders it unusable.
XML Metric Tag (Metrics only)	Enables you to specify one or more XML metric tags. The XML metric tag creates a shortcut to the XML element that you want to process. A metric tag typically identifies a metric record in an XML data. You can define more than one metric tag.
	Create a new XML metric tag manually. Enables you to type an XML element in the provided box. **Create a new XML metric tag from XML sample data.** Opens the XML Sample Data Outline dialog box. This dialog box displays the XML elements and attributes contained in the uploaded XML sample data.
	Deletes the selected XML metric tag. **Caution:** Deleting a metric tag that is referenced in a policy corrupts the policy and renders it unusable.
Detect deltas (Topology only)	A delta is the difference between the received XML file and the repository. The delta is the bare minimum of information about the changes. It discovers all changes without regards to the RTSM model. If an instance or relation is not discovered for a certain amount of time (Age to deletion), it is deleted. The counter goes up each time when the algorithm is running (this depends on the interval and the last time the file was written / sent via web service). If not selected,no delta detection is done and the incoming topology file is directly sent to the topology server. Default value: selected.
Age to deletion (Topology only)	A number of times for the policy to be run for a host that is not part of the topology data. After this number of policy executions, this host is deleted from the server.

Source Page - Perl Source Policies

UI Element	Description
Script	The Perl script to be executed by the opcgeni runtime. The script can either be embedded or loaded from an external file: • **Load Perl script file from filesystem path on BSMC server** The path to the external Perl script that is run. • **Use embedded script** Click to upload the Perl script file to the Perl Script policy. For more information on how to write Perl scripts, see "Understanding How Data is Collected and Processed using Perl Scripts" on page 166.
Polling Interval	Specifies the interval between two script executions in days, hours, minutes, and seconds.
Subroutine name	The name of the Perl function (subroutine) which is called to perform data collection.
Input Parameters	An ordered list of input parameters, passed as arguments to the Perl sub function in the configured order. The parameters can be saved encrypted to the policy and are decrypted before the sub function is called. To encode the parameter values, click the parameter and then click the arrow that is displayed on the right of the parameter name and select the option **Encode as password**.
Result data array name	The name of the Perl array of hash references used to transfer data between the embedded Perl interpreter and the opcgeni pipeline. They must match the ones in the Perl script.
Result data key names	A list of attribute names of interest for each hash element contained in the created Perl array of hashes. These make up the input properties to the policy processing pipeline.
Data source tag	A user defined descriptive name for the collected data. This tag can be used to distinguish data sets generated by different policies. This may be useful in data forwarding scenario, for example for consumer applications that receive data from BSM Connector.

Source Page - XML File Policies

UI Element	Description

Log File Path / Name Events and Metrics only) **Topology XML File** (Topology only)	Path and name of the XML file that the policy reads. Type the drive letter and the full path for the location of this file on the BSM Connector system. You can use the following configurations to make your policy more flexible: • Windows environment variables (for example, `winnt` or `clusterlog`). The syntax for these variables is `<$variablename>`, for example `<$winnt>`. • Script or command that returns the path and name of the log file you want to access. For example, type `<`command`>` where `command` is the name of a script that returns the path and name of the log file you want the policy to read. The command can also return more than one log file path separated by spaces. The HP Operations Agent processes each of the files using the same options and conditions as configured for this policy. This is very useful when you want to dynamically determine the log file path or process multiple instances of a log file. • Pattern matching. The pattern-matching language enables you to very accurately specify the file names that you want the policy to match. For example, you can use the pattern `<path>/events<*>.xml` to match XML source file names such as `events.1.xml` and `events.2.xml`. For more information on pattern matching, see "Pattern-Matching Details" on page 284. **Note:** BSM Connector cannot process XML files that are larger than 2 GB.
Polling Interval	Determines how often the policy reads the XML file (in days, hours, minutes, and seconds). This period of time is the polling interval. The larger polling interval is set, the less performance is needed. However, more memory is used (this depends on the amount of the data in the XML file). Setting the polling interval below 30 seconds is not recommended, the default setting is usually appropriate. Note that a policy begins to evaluate data *after* the first polling interval passes, unless the `Read from beginning (first time)` or `Read from beginning (always)` radio button is selected, which results in evaluation of the first data at the policy activation. A shorter polling interval is better when you are testing a policy. Default value: 5 minutes **Note:** Make sure that you set this value to a minimum of `15` seconds to be able to save the policy.

Logfile Character Set (Event and Metrics only)	Name of the character set used by the XML file that the policy reads. **Note:** It is important to choose the correct character set. If the character set that the policy is expecting does not match the character set in the XML file, pattern matching may not work, and the event details can have incorrect characters or be truncated in OMi. If you are unsure of which character set is used by the XML file that the policy reads, consult the documentation of the program that writes the file. Default value: UTF-8
Send event if log file does not exist (Events and Metrics only) **Send OMi event if topology file does not exist** (Topology only)	BSM Connector sends an event if the specified XML file does not exist. Default value: not selected
Close after reading (Events and Metric only)	If you select this option, the file handle of the XML file closes and reopens again after the polling interval. The XML file is read from the last position. If this file had a rollover in the meantime, it is read from the beginning. If the name of the XML file changes, and a new file was started in the meantime, the policy continues to read the new XML file and the original XML file data is lost. If you do not select this option, the file handle remains and is read entirely each time, unless there is a newer file with the same name (or name pattern). In that case the original XML file is read to the end, and then the newer file is read. Therefore, no data is lost. Consider the following example: a policy reads the XML file `system.xml`.While there is still some unread data in the `system.xml` file, it is renamed to `systemMonday.xml`, and a new version of the `system.xml` file is written. In case this option is selected, the unread data from the `systemMonday.xml` file remains unread and the `system.xml` file is read entirely. In case this option is not selected, the unread data from the new `system.xml` file is read after the reading of the `systemMonday.xml` file is completed. Default value: not selected

<table>
<tr>
<td>Read Mode (Events and Metric only)</td>
<td>The read mode of an XML file policy indicates whether the policy processes the entire file or only new entries.
<table>
<tr>
<td>Read from last position. The policy reads only new—appended—entries written in the XML file while the policy is activated. If the file decreases in size between readings, then the entire file is read. Entries that are added to the file when the policy is disabled are not processed by the policy.

Choose this option if you are concerned only with entries that occur when the policy is enabled.</td>
<td>Advantage: No chance of reading the same entry twice. (Unless the file decreases in size because some entries were deleted.)

Disadvantage: Entries written to file while the policy is disabled or the agent is not running are not processed by the policy.</td>
</tr>
<tr>
<td>Read from beginning (first time). The policy reads the complete XML file each time the policy is activated or the agent restarts. This ensures that all entries in the file are compared with the rules in the policy. Each successive time that the policy reads the file, only new (appended) entries in the file are processed.

Choose this option if you want to ensure that every existing and future entry in the file is processed by the policy while it is activated.</td>
<td>Advantage: Every existing and future entry in the file will be processed by the policy.

Disadvantage: Duplicate entries can occur if an activated policy is deactivated and reactivated, or if the agent stops and restarts.</td>
</tr>
<tr>
<td>Read from beginning (always). The policy reads the complete XML file every time it detects that the file has changed. The policy scans the file at the specified polling interval. If no change is detected, the file is not processed. Any entries overwritten while the agent is not running or the policy is deactivated will not be evaluated by the policy.

Choose this option if the policy reads a file that is overwritten, rather than appended.</td>
<td>Advantage: Ensures that files that are overwritten are correctly processed.

Disadvantage: Only valid for files that are overwritten, rather than appended.</td>
</tr>
</table>
Note: Every policy reads the same XML files independently from any other policies. This means, for example, that if "Policy 1" with read mode Read</td>
</tr>
</table>

	from beginning (first time) is activated and "Policy 2" with the same read mode already exists, "Policy 1" still reads the entire file after it has been activated. Default value: Read from last position
Sample Data (Events and Metrics only)	Enables you to upload an XML sample file. BSM Connector makes the XML elements and values of the sample file available to you in the Event and Rules pages so that you can insert them by dragging and dropping.
	Load sample data from server. Loads an XML sample file from the OMi Connector system. **Load sample data from local file system.** Loads an XML sample file from the system where the Web browser runs. **Note:** BSM Connector can only load a maximum of 50 MB of sample data.
	Opens the XML Sample Data dialog box. This dialog box displays the contents of the uploaded XML sample file.
XML Event Tag (Events only)	Enables you to specify one or more XML event tags. The XML event tag creates a shortcut to the XML element that you want to process. An event tag typically identifies an event record in an XML file. You can define more than one event tag.
	Create new XML event tag manually. Enables you to type an XML element in the provided box. **Create new XML event tag from XML sample data.** Opens the XML Sample Data Outline dialog box. This dialog box displays the XML elements and attributes contained in the uploaded XML sample data.
	Deletes the selected XML event tag. **Caution:** Deleting an event tag that is referenced in a policy corrupts the policy and renders it unusable.
XML Metric Tag (Metrics only)	Enables you to specify one or more XML metric tags. The XML metric tag creates a shortcut to the XML element that you want to process. A metric tag typically identifies a metric record in an XML file. You can define more than one metric tag.
	Create new XML metric tag manually. Enables you to type an XML element in the provided box. **Create new XML metric tag from XML sample data.** Opens the XML Sample Data Outline dialog box. This dialog box displays the XML elements and attributes contained in the uploaded XML sample data.

	Deletes the selected XML metric tag. **Caution:** Deleting a metric tag that is referenced in a policy corrupts the policy and renders it unusable.
Detect deltas (Topology only)	A delta is the difference between the received XML file and the repository. The delta is the bare minimum of information about the changes. It discovers all changes without regards to the RTSM model. If an instance or relation is not discovered for a certain amount of time (Age to deletion), it is deleted. The counter goes up each time when the algorithm is running (this depends on the interval and the last time the file was written / sent via web service). If not selected,no delta detection is done and the incoming topology file is directly sent to the topology server. Default value: selected.
Age to deletion (Topology only)	A number of times for the policy to be run for a host that is not part of the topology data. After this number of policy executions, this host is deleted from the server.

Start, Success, Failure Event Pages (Scheduled Task Policies)

UI Element	Description
Send Start Event	Click to send an event when the command begins to run.
Send Success Event	Click to send an event when the command completes successfully.
Send Failure Event	Click to send an event when the command fails to run or fails to complete successfully.

Structured Input Page (Data Forwarding)

Metric Data Forwarding Rules

UI Element	Description

	Adds a new data forwarding rule to the list of rules.
	Deletes the selected data forwarding rule.
Description	A short description of the data forwarding rule.
Rule Type	Displays the type of the rule selected for this target.
Data Forwarding Target	Displays the URL of the target.

Properties Tab

UI Element	Description
Description	Describes the purpose of the rule.
Rule type	Defines the rule type. The following types are available: • Forward on matched • Discard on matched • Discard on unmatched

Condition Tab

UI Element	Description
Property	Input data property. You can drag and drop a value from the **Meta Data** tab.
Operator	The following operators are available: • equals • not equals • less than • greater than • less or equal • greater or equal • matches
Operand	A numeric value between 0 and 65536.

Meta Data tab

UI Element	Description

<list of available keys>	Displays the available keys either from the BSM Connector Metric Meta Model (when forwarding metric data) or keys derived directly from the input data (when forwarding structured input data). You can drag and drop the keys in to the **Property** field.

Targets Tab

UI Element	Description
Affect all targets	If selected, the rule affects all configured targets.
<list of targets>	Lists all targets that are configured on the **Targets** page. Click on a target to enable it.
Override Target Configuration	Overrides the target configuration for selected targets.
Wire format	Selects the wire format - XML or JSON.
Use Guaranteed Delivery	If selected, guaranteed delivery is used.

Target Page (Data Forwarding)

UI Element	Description
<Data Forwarding Targets>	A table of configured receiver targets. The table lists the target name, description, URL, wire format and whether guaranteed delivery is used.
Name	The user defined name of the receiving target. The filed is mandatory.
Description	A brief description of the target.
URL	The URL of the web service endpoint. The field is mandatory.
Wire format	The wire format the target expects. The supported formats are XML and JSON.
Use Guaranteed Delivery	Select this option to use guaranteed delivery.

Task Page (Scheduled Task Policies)

UI Element	Description
	Load <perl / VB Script> from Client Opens a file selection dialog box for you to select the VB or Perl script to load into the policy.
Task Type	Type of task: • Command • VB Script • Perl Script
Command	Complete path and extension of the command that you want to run (for example, `%OvDataDir%\bin\instrumentation\cleanup.exe`). The file that you specify should exist on the system. By default, the command runs under the same account as the agent is running, which is Local System or root by default.
Username	User name under which the command should be run. The user must exist and have permission to run the command on the system. If you specify a non-existent user, the command fails to run.
Password	Password for the user. If the password changes, the policy must be updated and reactivated.
VB Script	Code that defines the VB script. Instead of typing the script into the field, you can upload an existing script. **Tip:** Use the policy method `Rule.Status` to specify whether the task is successful. For example, to specify that the task has failed (and trigger a failure event), use `Rule.Status=False`. (See "Rule Object" on page 218 .) **Note:** HP Operations Agent uses a generic Microsoft scripting engine to run VBScript scripts.You can therefore use standard VBScript objects (for example, the FileSystemObject object) in your scripts. Objects that are specific to wscript or cscript (for example, the WScript object) are not supported.

UI Element	Description
Perl Script	Code that defines the Perl script. Instead of typing the script into the field, you can upload an existing script. **Tip:** Use the policy method `$Rule->Status` to specify whether the task is successful. For example, to specify that the task has failed (and trigger a failure message), use `$Rule->Status(False)`. (See "Rule Object" on page 218 .) **Note:** The agent runs as a service that has no standard input, standard output, or standard error. Therefore, the predefined file handles STDIN, STDOUT, and STDERR are not available for Perl scripts in scheduled task policies. It is also not possible to open file handles that use command pipes or capture the standard output from commands within backticks (`).

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on BSM Connector User Guide (Business Service Management 10.01)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to ovdoc-asm@hp.com.

We appreciate your feedback!