HP Business Service Management

For the Linux and Windows® operating systems

Software Version: 9.21

Using BSM Connector

Document Release Date: February 2013

Software Release Date: February 2013

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

Copyright Notice

Trademark Notices

Acknowledgements

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

This product includes software developed by the JDOM Project (http://www.jdom.org/).

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:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Note: This document was last updated: Friday, January 25, 2013

PDF Version of Online Help

This document is a PDF version of the online help. This PDF file is provided so you can easily print multiple topics from the help information or read the online help in PDF format.

Support

Visit the HP Software Support Online web site at:

http://www.hp.com/go/hpsoftwaresupport

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

HP Software online 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:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Chapter 1

Introducing BSM Connector

HP BSM Connector (BSM Connector) is a component of HP Business Service Management (BSM) that enables you to integrate data from third-party systems (typically enterprise management systems) in BSM. You can integrate events, metrics, and topology data in BSM. 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 BSM. Policies can also report topology data to BSM in order to create CIs and CI relationships in BSM'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
Log files	Local or remote	Events Metrics Topology	Log file policies watch for specific entries added to a log file provided by a third-party system by trying to match against a regular expression.
Database tables	Local or remote	Events Metrics Topology	Database policies enable you to collect event and time series of data from database tables used by third-party systems by performing a query through a JDBC connection.
Web service requests	Local or remote	Events Metrics Topology	Web service listener policies process messages received by the BSM Connector Web service listener.

Data Source	Integration Type	Integration Data	Description
SNMP traps	Local or remote	Events	SNMP trap policies process SNMP traps received by BSM Connector from third-party systems.
XML files	Local or remote	Events	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 can be a local file or a remote file that is available through a network share. For example, you can install the BSM Connector for SCOM and BSM Connector on a separate computer, or you can install the software on the same computer that runs the SCOM server.
Open message interface messages	Local only	Events	Open message interface policies process messages sent by the BSM Connector opcmsg command line tool.
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 BSM to indicate the start, success, or failure of the command.
Custom topology	Local only	Topology	Custom topology policies (based on DFM technology) enable you to run discovery scripts on the BSM Connector system. The discovery scripts collect topology data and send the data to BSM to create CIs and CI relationships in BSM's RTSM.

Data Source	Integration Type	Integration Data	Description
Discovery (deprecated)	Local only	Topology	Discovery policies (based on HP Operations Agent discovery technology) execute discovery scripts on the BSM Connector server. The discovery agent stores the data that it discovers in the agent repository (agtrep), which is a local data store of configuration items and their relationships.The agent then sends and synchronizes the local repository with the RTSM. BSM Connector supports agent-based discovery only for discovery policies that have been migrated from BSM Integration Adapter or imported from HP Operations Manager (HPOM). You can edit migrated discovery policies. HPOM service auto-discovery policies cannot be edited because they are incompatible with the BSM Connector legacy discovery policy editor.

BSM Applications That Use BSM Connector Data

The following BSM applications consume BSM Connector data:

- Events
  - Operations Management
  - Service Health
  - Service Level Management
- Metrics
  - Operations Management Performance Graphing

    > **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.

  - Service Health
  - Service Health Analyzer
  - Service Level Management
  - System Availability Management reports
- Topology

- Operations Management
- Run-time Service Model
- Service Health
- Service Health Analyzer
- Service Level Management

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.

BSM Connector integrates with the following HP products:

- HP ArcSight ESM
- HP ArcSight Logger
- HP Network Node Manager i

In addition, HP provides the following out-of-the box integrations with third-party products:

- HP BSM Connector for IBM Tivoli
- HP BSM Connector for Microsoft System Center Operations Manager (SCOM)
- HP BSM Connector for Nagios

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.

Key Features and Benefits

Key Features

- **Topology.** Discover and report topology from the product BSM Connector connects to (based on log files, database tables, Web services, or custom data sources using topology scripts). The discovered topology populates the RTSM. You can then manage and work with these discovered configuration items (CIs) in views.
- **Events.** Get events (from log files, database tables, Web services, XML files, SNMP traps, Open Message Interface messages, or scheduled tasks).
- **Metrics.** Get metrics (from log files, database tables, or Web services); metrics are stored in the BSM Profile Database.
- **Web-based UI.** Define policies using Web-based UI.
- **Event synchronization.** After BSM receives an event, you can keep it up to date on the event source by configuring BSM and BSM Connector to synchronize event changes back to the third-party system that generated the original event.
- **BSM integration.** Manage BSM Connector from BSM.

- **Backward compatibility.** Backward compatibility with previous integrations based on BSM Integration Adapter. (SiteScope Technology Integration Monitors remain available for existing integrations in BSM 9.20. For new integrations, BSM Connector should be used.
- **Out-of-the-box connectors.** Supports BSM's open integration strategy, providing out-of-the-box connectors to HP and third-party products. This enables expanding connectivity of BSM and related products (Operations Manager *i*, Service Health Analyzer).

  Customers can access connectors via HP Live Network (HPLN); NNMi Connector, BSM Connector for Nagios, BSM Connector for Microsoft SCOM, and BSM Connector for IBM Tivoli are already available.

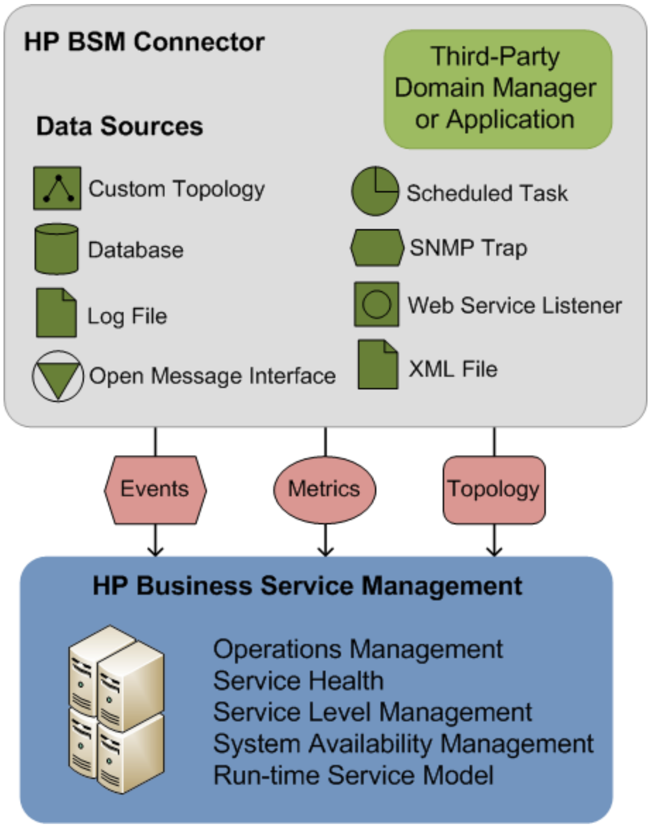
  HP certified partners will also be able to build their own connectors and post them on HPLN. See the HP Live Network site https://hpln.hp.com/group/bsm-integrations.

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.
- **Service Health and Service Level Management.** Health indicators assigned to BSM Connector event and metrics data provide a more detailed view of the health of a configuration item (CI).
- **Reports.** Ability to create and view reports of BSM Connector metrics in System Availability Management and Operations Management Performance Graphing.
- **Service Health Analyzer (SHA):** SHA analyzes metrics, calculates the dynamic baseline to detect if each metric behaves normally or not, and automatically correlates related metrics to a single meaningful event.

BSM Connector Deployment Scenarios

The following figure shows a BSM Connector deployment.



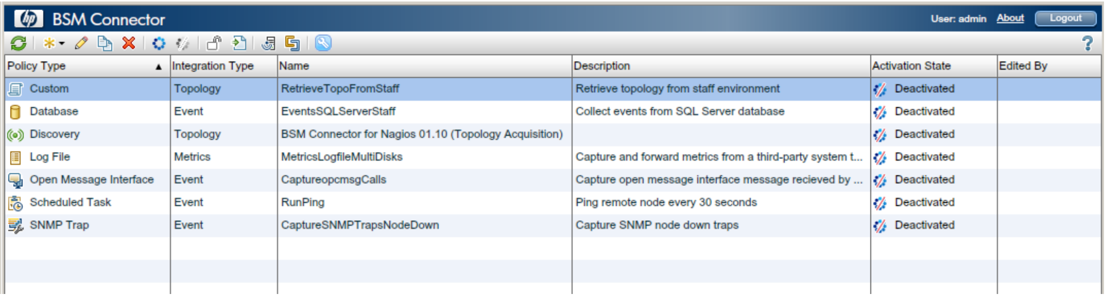
You typically install BSM Connector on the third-party system that acts as data source. For example, to integrate NNMi data into BSM, you install BSM Connector on the NNMi management server. However, BSM Connector can also access data sources on remote systems without the need to install data collectors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

Where you install BSM Connector depends on the type of data you want to integrate in BSM. BSM Connector must be installed and run locally on the data host when integrating data originating from 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, log, or XML files, or receive data through SNMP traps or the Web service listener. For details, see "Integration Types" on page 14 and "Integration Data" on page 14.

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.



Policy Type ▲	Integration Type	Name	Description	Activation State	Edited By
Custom	Topology	RetrieveTopoFromStaff	Retrieve topology from staff environment	Deactivated	
Database	Event	EventsSQLServerStaff	Collect events from SQL Server database	Deactivated	
Discovery	Topology	BSM Connector for Nagios 01.10 (Topology Acquisition)		Deactivated	
Log File	Metrics	MetricsLogfileMultiDisks	Capture and forward metrics from a third-party system t...	Deactivated	
Open Message Interface	Event	CaptureopcmsgCalls	Capture open message interface message recieved by ...	Deactivated	
Scheduled Task	Event	RunPing	Ping remote node every 30 seconds	Deactivated	
SNMP Trap	Event	CaptureSNMPTrapsNodeDown	Capture SNMP node down traps	Deactivated	

For more information about the toolbar buttons and columns, as well as policy management in general, see "Managing Policies" on page 103.

User Authentication

BSM Connector supports the following user authentication strategies:

- Single Sign-On (SSO) for BSM applications (recommended)

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

  LW-SSO enables a user to log into BSM once and gain access to all BSM applications without being prompted to log in again. The applications inside BSM 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, BSM users can launch the BSM Connector user interface without having to provide additional credentials.

  > **Tip:** BSM 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 (`IA_ADMINS`, for example) and assign permissions to access BSM Connector to the group.

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

- 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 BSM applications. When BSM 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 24.

Chapter 2

Setting Up and Configuring BSM Connector

Licensing

BSM Connector does not require a license. However, BSM and the BSM applications that consume BSM Connector data do require licenses. To use these applications, you must obtain licenses from HP. For more information visit HP Software Support Online (http://www.hp.com/go/hpsoftwaresupport).

Getting Started

1. Plan your integration strategy.

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

   Consider the data source: log files, databases, web services messages, 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 log files, or receive SNMP traps or Web service messages. For details on installing BSM Connector, see the interactive HP BSM Connector Installation and Migration Guide.

   Alternatively, upgrade from BSM Integration Adapter to BSM Connector as described in the BSM Connector *Installation and Migration Guide*.

3. Configure the BSM Connector instance as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see "How to Configure BSM Connector to Communicate with BSM" on page 27.

4. Log on to BSM Connector.

   To access BSM Connector from a Web browser or from within BSM, see "How to Log on to BSM Connector" on page 24.

5. 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 17.

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 110

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 Only
Log files	"How to Collect Event Data from Log Files" on page 147	"How to Collect Metrics Data from Log Files" on page 148	"How to Collect Topology Data from Log Files" on page 149
Database tables	"How to Collect Event Data from Databases" on page 115	"How to Collect Metrics Data from Databases" on page 116	"How to Collect Topology Data Only from Databases" on page 117
Web service requests	"How to Collect Event Data Through the Web Service Listener" on page 271	"How to Collect Metrics Data Through the Web Service Listener" on page 272	"How to Collect Topology Data Through the Web Service Listener" on page 273
SNMP traps	"How to Collect Event Data from SNMP Traps" on page 264	Not available.	"How to Report Topology Data" on page 334
XML files	"How to Collect Event Data from XML Files" on page 311	Not available.	"How to Report Topology Data" on page 334
Open message interface messages	"How to Collect Event Data from the Open Message Interface" on page 235	Not available.	"How to Report Topology Data" on page 334
Scheduled tasks	"How to Schedule Tasks" on page 247	Not available.	"How to Report Topology Data" on page 334

6. 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 108.

7. View the data in BSM.

   In BSM, open the applications that consume the integrated data and make sure BSM receives the expected data. For details on the applications that consume BSM Conncector integration data, see "BSM Applications That Use BSM Connector Data" on page 16.

For example, to view events in BSM, access Operations Management and open the Event Browser.

Configuration Items in BSM

BSM Connector policies can collect and forward topology data to BSM to create CIs and CI relationships in the RTSM. Accurate CI topology information provided by discovery of the system infrastructure is essential for the following BSM applications:

- Operations Management (for example, for Topology-based Event Correlation (TBEC), service health monitoring (Health Perspective), context-specific tools)
- Service Health
- Service Level Management

BSM Connector forwards the topology data to create a CI when the CI is discovered by BSM Connector for the first time as a result of the policy retrieving and matching data (regardless of whether the CI exists in the RTSM). In addition, every time BSM Connector discovers changes to the CI's properties, it forwards them to the RTSM. This prevents overloading the RTSM with CI updates coming from the policy.

This section includes:

- "Aging of CIs in RTSM" below
- "Touching of CIs in BSM Connector" below

Aging of CIs in RTSM

BSM removes CIs from the RTSM if the CIs have not had any activity over a period of time; that is, if their last access time has not changed. The CIs created from BSM Connector data are also subject to this aging policy. To prevent the aging policy from acting on CIs that BSM Connector has sent to BSM, BSM Connector sends a list of active CIs to BSM in addition to the topology data. This process is known as topology synchronization.

For details on setting the time interval for topology synchronization, see "BSM Connector Integration Administration" in the BSM Application Administration Guide. For details on the aging mechanism, see "CI Lifecycle and the Aging Mechanism" in the RTSM Administration Guide.

Note: *BSM Connector topology synchronization* is not to be confused with *local topology synchronization* in BSM Connector. During local topology synchronization, BSM Connector applies mapping rules to discovered topology to transform the discovered objects to CIs and CI relations that can then be stored in the RTSM.

Touching of CIs in BSM Connector

BSM Connector stores information about the CIs that it synchronizes with the RTSM in a local cache. The cache is updated each time topology for a new CI or updates to an existing CI are reported.

The local cache does not contain all information about the CI, only the CI ID and a flag indicating whether the CI has been touched or not since the last topology synchronization; that is, whether topology data has been received for the CI or not. With the next synchronization, BSM Connector sends only those CI IDs to the RTSM that have been touched. RTSM then updates the last access time of the CIs, which prevents them from becoming deletion candidates in the RTSM.

BSM Connector by default distinguishes between CIs for which topology data has been received and CIs that have had no such activity. Inactive CIs are not touched and are not sent to the RTSM with the next synchronization. Their last access time is not updated so they become subject to the BSM aging policy.

You can configure BSM Connector to always touch CIs regardless of whether topology data has been received for them. This guarantees that CIs reported by BSM Connector are never deleted automatically from the RTSM because their last access time is updated every time BSM Connector synchronizes topology data with BSM.

The always touch mode is recommended for BSM Connector topology-only policies that collect data from incremental data sources such as log files or databases. The first time the policies run, they collect all CIs from the data source. The next time the policies run, they collect new and changed CIs only and then forward only the additions and changes to BSM. The always touch mode ensures that the already existing, unchanged CIs are also reported and are thus not deleted.

To enable the always touch mode, set the parameter **_alwaysPerformTouch** to true in the **<BSM Connector root directory>/groups/master.config** file.

> **Note:** The always touch mode is a global setting that affects *all* policies of a BSM Connector instance that forward topology data to BSM. You cannot have multiple policies requiring different aging modes in the same instance of BSM Connector.

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. For more information about supported web browsers, see the *BSM Connector Release Notes*.

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 BSM. In this case an LW-SSO session cookie stored in the web browser contains the BSM authentication information. BSM users can access BSM Connector only if they have the necessary permissions (for example, if they have the `SUPERUSER` role assigned or are members of the `BSMC_ADMINS` group).

This section includes:

- "Log On to BSM Connector" below
- "Log Out of BSM Connector" on page 26
- "Troubleshooting" on page 26

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 Apache Tomcat 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. 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 BSM

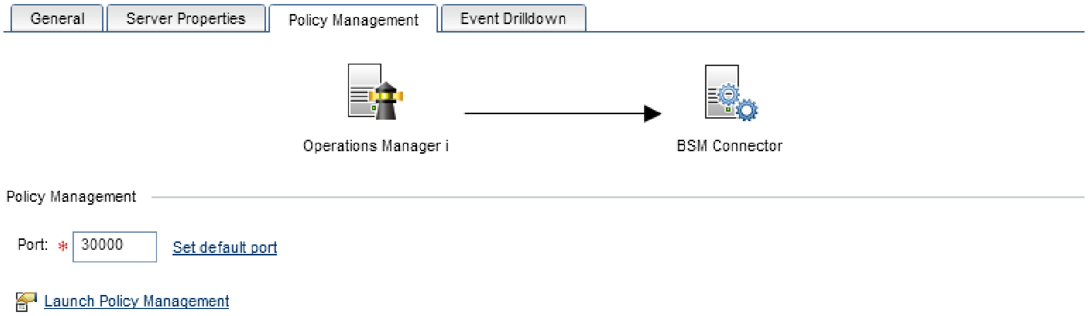
BSM enables you to start the user interface of BSM Connector or of the third-party system directly from within BSM. The following integrations are available:

BSM Connector UI integrations

- **BSM Connector integrations page.** Navigate to **Admin > Integrations > BSM Connector Integrations**. In the Summary view, locate the BSM Connector you want to launch. Click the name in the Name column.
- **Connected Servers manager.** Navigate to **Admin > Operations Management > Setup > Connected Servers**. In the Connected Servers details pane, locate the BSM Connector you want to launch and click the Launch Policy Management icon.

  Alternatively, edit the BSM Connector you want to launch, navigate to the Policy Management tab, and click the Launch Policy Management icon.

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



Port: * 30000

- **Event context.** In the Operations Management 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. 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.

For details, see "How to Configure Launch of the BSM Connector User Interface" on page 93.

Third-party system UI integration

In the Operations Management 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 94.

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: Accessing the BSM Connector user interface produces a "web page not found" error in the browser, and the Tomcat log file `<BSM Connector root directory>/logs/tomcat.log` contains the following lines:

```
LifecycleException:  service.getName(): "Catalina";
Protocol handler start failed: java.io.IOException:
Keystore was tampered with, or password was incorrect
```

Solution: The self-signed certificate that is provided out-of-the-box with BSM Connector must be replaced with a server certificate from a CA in PKCS12 format. Alternatively, manually generate a server certificate using a Java Keystore (JKS) and have it signed by your CA. For details, see "How to Prepare BSM Connector for Using SSL" on page 44.

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.

BSM Connector on Windows service

Use Windows Services to manage the BSM Connector service:

1. Start the Windows Task Manager.
2. Click the **Services** tab.

   The default BSM Connector service name is **HP BSM Connector**.

BSM Connector on Linux processes

- To start the main BSM Connector process, use one of the following commands:

  ```
  /opt/HP/BSMConnector/start
  ```

  ```
  service bsmc start
  ```

- To stop the main process, use one of the following commands:

  ```
  /opt/HP/BSMConnector/stop
  ```

```
service bsmc stop
```

- To restart the main BSM Connector process, use the following command:

  ```
  service bsmc restart
  ```

- To print the status of the main BSM Connector process, use the following command:

  ```
  service bsmc status
  ```

> **Note:** The bsmc service is not by default registered on Linux. To register the service, complete the following steps:
>
> 1. Copy the init script `bsmc` to `/etc/init.d`:
>
>    ```
>    cp /opt/HP/BSMConnector/bin/bsmc /etc/init.d/bsmc.tmp
>    ```
>
> 2. Convert the script from DOS to UNIX format, for example:
>
>    ```
>    dos2unix /etc/init.d/bsmc.tmp /etc/init.d/bsmc
>    ```
>
> 3. Add execution rights to the script:
>
>    ```
>    chmod +x /etc/init.d/bsmc
>    ```
>
> 4. Register the script:
>
>    ```
>    chkconfig --add bsmc
>    ```

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\ovc`

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

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

How to Configure BSM Connector to Communicate with BSM

To enable the connection between BSM Connector and BSM, the BSM Connector instance must be configured as a BSM Connector integration server in BSM. This involves adding the BSM Connector server to the BSM Integrations page in BSM (**Admin > Integrations > BSM Connector Integrations**).

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. In BSM, check that a profile database exists, access:

Admin > Platform > Setup and Maintenance > Manage Profile Databases

The profile database stores metrics data and is required, even if no metrics data will be sent from the BSM Connector.

If a profile database does not yet exist, create one as described in "Database Administration" in the BSM Platform Administration Guide.

3. In BSM, open the New BSM Connector page, select:

   Admin > Integrations > BSM Connector Integrations

   In the left pane, click * **New BSM Connector**. The New BSM Connector page displays.

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

   - **Main Settings.** Complete the mandatory **Display name** and **Host name** fields. Accept or change the default HTTP and HTTPS port numbers.

     > **Note:** If you are working with a BSM Connector that is not accessible to BSM (for example in HP Software-as-a-Service), the procedure for the connection includes creating an empty profile in BSM and creating an Integration Preference for BSM in BSM Connector. For task details, see "Configuring Integration Preferences for Inaccessible Profiles" on page 33.

   - **Event Drilldown Settings.** These settings apply to event data only and are optional.

     Event drilldown enables BSM users to launch the user interface of the third-party system in the context of an event collected through BSM Connector. Event synchronization sends event lifecycle changes back to the event source. For example, if a BSM operator closes an event originating from NNMi, a notification can be automatically sent to NNMi.

   - **BSM Gateway Server Settings.** These settings are mandatory and are completed automatically. Accept or change the defaults as required.

     The required information includes the name of the BSM Gateway Server to which the BSM Connector sends the collected data. The name of the user to log into the BSM Gateway Server from the BSM Connector is also required because BSM Connector subsequently uses this user name to authenticate internal calls to BSM.

   - **Policy Management Settings.** These settings are optional. Accept or change the defaults as required.

     The **Configure policy management** settings establish secure communication between BSM Connector and the BSM Gateway Server based on certificates. The Configure policy management check box sends a certificate request to BSM, which must then be granted in BSM. Secure communication is required for policy activation in BSM Connector and for event forwarding from BSM Connector to BSM.

   - **Profile Settings.** These settings are mandatory and are completed automatically. Accept or change the defaults as required.

     The settings define a profile database for a BSM Connector in BSM. The profile database stores metrics data and is required, even if no metrics data will be sent from the BSM

Connector. If a profile does not yet exist in BSM, create one as described in "Database Administration" in the BSM Platform Administration Guide.

- **Topology Settings.** These settings are mandatory for topology data only. Accept or change the defaults as required.

  The settings define how often the topology data is synchronized between BSM and BSM Connector, from which routing domain the topology data is collected, and to which port on the BSM Gateway Server BSM Connector sends the collected data.

4. Click **Submit** to save the new BSM Connector integration.

   Setting up a BSM Connector integration on this page automatically creates a BSM Connector connected server in Operations Management.

5. *Results.* You can view the collected data in different BSM applications, depending on the type of integration data that you collect. For example, you can view data in Operations Management, RTSM, Service Health, or BSM Reports.

   > **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 integration server, see the BSM online help or the BSM Application Administration Guide.

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 BSM 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 BSM 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 server receives all metrics data.

- The BSM 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 BSM and HPOM systems. For information about exchanging certificates, see "To configure trusted certificates:" on page 32.
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\HPOprIA\3F9A8F04-B5E3-43C3-999B-7A9492C35014_data.template" "%OvDataDir%\conf\HPOprIA\3F9A8F04-B5E3-43C3-999B-7A9492C35014_data"
   ```

   ```
   copy "%OvDataDir%\conf\HPOprIA\3F9A8F04-B5E3-43C3-999B-7A9492C35014_header.xml.template" "%OvDataDir%\conf\HPOprIA\3F9A8F04-B5E3-43C3-999B-7A9492C35014_header.xml"
   ```

   #### BSM Connector on Linux

   Open a shell and type:

   ```
   cp /var/opt/OV/conf/HPOprIA/3F9A8F04-B5E3-43C3-999B-7A9492C35014_data.template /var/opt/OV/conf/HPOprIA/3F9A8F04-B5E3-43C3-999B-7A9492C35014_data
   ```

   ```
   cp /var/opt/OV/conf/HPOprIA/3F9A8F04-B5E3-43C3-999B-7A9492C35014_header.xml.template /var/opt/OV/conf/HPOprIA/3F9A8F04-B5E3-43C3-999B-7A9492C35014_header.xml
   ```

   b. Edit the policy data file `3F9A8F04-B5E3-43C3-999B-7A9492C35014_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 BSM gateway server.

Locate the string `${OMi_MGR_SRV_ID}` and replace all occurrences with the core ID of the BSM 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 BSM 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\HPOprIA\3F9A8F04-B5E3-43C3-999B-
   7A9492C35014_data.template" "%OvDataDir%\conf\HPOprIA\3F9A8F04-
   B5E3-43C3-999B-7A9492C35014_data"
   ```

   ```
   copy "%OvDataDir%\conf\HPOprIA\3F9A8F04-B5E3-43C3-999B-
   7A9492C35014_header.xml.template"
   "%OvDataDir%\conf\HPOprIA\3F9A8F04-B5E3-43C3-999B-7A9492C35014_
   header.xml"
   ```

   #### BSM Connector on Linux

   Open a shell and type:

   ```
   cp /var/opt/OV/conf/HPOprIA/3F9A8F04-B5E3-43C3-999B-
   7A9492C35014_data.template /var/opt/OV/conf/HPOprIA/3F9A8F04-
   B5E3-43C3-999B-7A9492C35014_data
   ```

   ```
   cp /var/opt/OV/conf/HPOprIA/3F9A8F04-B5E3-43C3-999B-
   7A9492C35014_header.xml.template
   ```

```
/var/opt/OV/conf/HPOprIA/3F9A8F04-B5E3-43C3-999B-7A9492C35014_
header.xml
```

b. Edit the policy data file `3F9A8F04-B5E3-43C3-999B-7A9492C35014_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 BSM gateway server.

   Locate the string `${OMi_MGR_SRV_ID}` and replace all occurrences with the core ID of the BSM 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 BSM servers.

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

1. On *every* BSM 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 BSM Connector to Send Compressed Data to BSM

How to Configure BSM Connector to Send Bulk Topology Data to the Run-time Service Model

BSM Connector topology data can be sent to BSM's Run-time Service Model (RTSM) either zipped or unzipped. The request includes a parameter that indicates to the RTSM whether the results being sent are in zipped or unzipped format.

1. Open the following file: **<BSM Connector root directory>\discovery\discovery_agent.properties**.

2. Locate the line beginning **appilog.agent.probe.send.results.zipped**. If the line does not exist, add it to the file.

3. Change the value to **=true**.

4. Restart BSM Connector:

   Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

   BSM Connector data is zipped before being sent to RTSM.

How to Configure BSM Connector to Send Compressed Metrics Data to BSM

By default, when metrics data is sent from the BSM Connector server to BSM it is sent uncompressed. You can configure BSM Connector to compress metric (ss_t) samples in gzip before they are sent to BSM (where they are decompressed).

To enable data compression of BSM Connector metrics data, complete the following steps:

1. Open the following file:

   ```
   <BSM Connector root directory>/groups/master.config
   ```

2. Set the value of `_compressDataInGzipFormat` to `true`.

3. Restart the BSM Connector server:

   Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

Configuring Integration Preferences for Inaccessible Profiles

When you add a BSM Connector integration to BSM, the BSM Connector instance must be accessible to BSM. If this is not the case, for example, when working in HP Software-as-a-Service, you may add the BSM Connector with an inaccessible profile to BSM, and then configure the connection and integration in the BSM Integration Preferences dialog box in BSM Connector.

To access

In the BSM Connector user interface, click **BSM Integration Preferences**. The BSM Integration Preferences dialog box opens.

Tasks

This section includes:

- "How to configure integration preferences for inaccessible profiles" below
- "How to delete a BSM integration preference" on the next page

How to configure integration preferences for inaccessible profiles

This task describes the steps involved in integrating BSM Connector in BSM when the BSM Connector is inaccessible to BSM, for example when working in HP Software-as-a-Service.

1. *Prerequisite.* Install BSM Connector.

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

2. *Prerequisite.* Create an empty profile for BSM Connector in BSM:

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

   b. In BSM, open the New BSM Connector page, click:

      Admin > Integrations > BSM Connector Integrations

      In the left pane, click ✱ **New BSM Connector**. The New BSM Connector page displays.

   c. In the New BSM Connector page, complete the following information:

      - **Display name.** Name for the BSM Connector integration.
      - **Host name.** Hostname of the BSM Connector server. You may enter a dummy hostname for the purpose of creating an empty profile.
      - **BSM Connector is inaccessible from BSM.** Creates an empty profile for the BSM Connector.
      - **Profile name.** In the Profile Settings pane, you may optionally enter the name for this profile. If you do not enter a profile name, the display name is used.

   d. Click **Submit** to save the new BSM Connector integration.

3. In the BSM Integration Preferences dialog box, complete the following mandatory information:

   - **Business Service Management FQDN.** Enter the fully qualified domain name or IP address of the BSM gateway server (or load balancer).
   - **Business Service Management user name.** Enter the user name of a BSM user with administrative privileges.
   - **Business Service Management user password.** Enter the password of the BSM administrator user.

- **BSM Connector profile to use.** Click **Load available profiles from Business Service Management** and select the previously created profile.

4. Click **Save** to save the integration preferences.

   If **Configure policy management** is selected in the Event Reporting Settings pane, BSM Connector sends a certificate request to BSM, which you must grant before the BSM Connector can be used:

   a. Navigate to **Admin > Operations Management > Setup > Certificate Requests** and locate the request.

      It may take a few minutes for the certificate request to appear.

   b. Grant the corresponding certificate request.

How to delete a BSM integration preference

When you delete a BSM integration preference you must also delete the BSM Connector integration with the inaccessible profile from the BSM deployment:

1. On the BSM Connector server, delete the BSM integration preference:

   a. In the BSM Connector user interface, click **BSM Integration Preferences**. The BSM Integration Preferences dialog box opens.

   b. Click ✖ **Delete**.

   c. In the warning dialog box, click **OK** to confirm that you want to delete the BSM Connector profile from BSM.

   d. Results.

      Deleting a BSM integration preference deletes all the BSM-related settings from the BSM Connector server. This also sends a message to the BSM server to release the BSM Connector agent from the corresponding profile.

      Resetting the configuration permanently removes the connection to BSM. BSM Connector will stop transferring data to BSM and cannot be reconnected to the current profile. Thus, new data collected from this BSM Connector will not appear in BSM.

2. Delete the BSM Connector integration with the inaccessible profile from your BSM deployment:

   a. Open the BSM Connector Integrations page, select:

      Admin > Integrations > BSM Connector Integrations

   b. Select the BSM Connector that you want to delete and click ✖.

   c. In the warning dialog box, click **OK** to confirm that you want to delete the BSM Connector profile from BSM.

   d. Results.

      Deleting a BSM Connector integration deletes the BSM Connector registration from BSM as well as the corresponding connected server in Operations Management.

UI Descriptions

BSM Integration Preferences

UI Element	Description
💾	**Save Integration Preferences.** Saves the integration settings.
✏	**Edit Integration Preferences.** Enables you to edit the integration preferences.
✖	**Delete All Integration Related Preferences.** Deletes all the BSM-related settings from the BSM Connector server. This also sends a message to the BSM server to release the BSM Connector agent from the corresponding profile. To complete the deletion, you must also delete the BSM Connector integration with the inaccessible profile from BSM. For details, see "How to delete a BSM integration preference" on the previous page. Resetting the configuration permanently removes the connection to BSM. BSM Connector will stop transferring data to BSM and cannot be reconnected to the current profile. Thus, new data collected from this BSM Connector will not appear in BSM. **Note:** If you delete the current integration preferences, you have to create or use a different profile to reconnect BSM Connector with BSM. BSM does not enable you to select a previously used profile.
🔄	**Force BSM Connector to Synchronize Configuration with Business Service Management.** Forces BSM Connector to resend all its third-party metrics definitions and metadata to BSM. **Hard synchronization.** Deletes the existing metrics definitions and metadata from BSM for this BSM Connector profile, and then forces BSM Connector to resend all its third-party metrics definitions and metadata to BSM.
BSM Integration Main Settings	
Business Service Management FQDN	Fully qualified domain name or IP address of the BSM gateway server or load balancer (depending on your BSM deployment) to which you want to connect this BSM Connector.
BSM Certificate Server FQDN	Fully qualified domain name or IP address of the BSM certificate server.
Business Service Management user name	User name of a BSM administrator-level user.

UI Element	Description
Business Service Management user password	Password for the specified user.
BSM Connector profile to use	BSM Connector profile in which BSM stores the data collected by BSM Connector. Click **Load available profiles from Business Service Management** to display a list of available empty profiles. **Note:** The profile must previously have been configured in BSM's BSM Connector Integrations page.
Web Server Security Settings	
Authentication user name	User name for the Web server authentication. Basic authentication and IDM-SSO. **Note:** This is the BSM Web server, not the BSM Connector Web server.
Authentication password	Password for the authentication of the Web server (basic authentication).
Use SSL (HTTPS)	Specifies that the Web server is configured for SSL. **Default value:** not selected
Proxy Server Settings	
Fully qualified domain name	If BSM Connector connects to the BSM Gateway Server through a proxy, specify the proxy name or IP address.
User name	User name for the proxy.
User password	Password associated with the user name above.
Topology Reporting Settings	
Topology resynchronization time interval (days)	Number of days for BSM Connector to synchronize topology data with BSM. **Default value:** 7 days **Minimum value:** 1 day **Note:** All topologies created by BSM Connector and stored in the RTSM are subjected to the aging process. For more information on aging, see "Aging of CIs in RTSM" on page 23.

UI Element	Description
Default routing domain	Routing domain of the topology probe. If you specify a new routing domain value, restart BSM Connector for the change to take effect. For more information on routing domains, see the Data Flow Management Guide. **Default value:** DefaultDomain
Topology receiver port	The port at which the BSM Gateway Server listens for topology data. **Default value:** 80
Topology receiver SSL port	The SSL port at which the BSM Gateway Server listens for topology data. **Default value:** 443
Topology anti-aging offset (minutes)	Offset from midnight, in minutes, for running the topology synchronization process. For more information on aging, see "Aging of CIs in RTSM" on page 23. **Default value:** 0 **Example:** To run topology synchronization at 1:30 am, enter an offset of 90.
Event Reporting Settings	
Installed agent version	Displays the version of the HP Operations Agent that is installed on the BSM Connector system.

UI Element	Description
Configure policy management	Enables the activation of policies and the collection of events through this BSM Connector. Policy activation and event collection require secure communication between BSM Connector and the BSM Gateway Server to which it is sending the event data. BSM Gateway Servers and other HP BTO Software applications use certificates to identify themselves and communicate securely with each other. If this checkbox is set when defining a new BSM Connector integration, a certificate request is automatically generated when you save the integration preferences. You must grant the certificate request before the BSM Connector can be used. To grant the certificate: 1. On the BSM Connector page, click 💾**Save** to save the integration preferences. 2. Navigate to **Admin > Operations Management > Setup > Certificate Requests** and locate the request. It may take a few minutes for the certificate request to appear. 3. Grant the corresponding certificate request. If this checkbox is cleared when defining a new BSM Connector integration, no certificate request is generated. The resulting BSM Connector integration cannot activate policies and therefore cannot collect any data. **Default value:** selected

Running BSM Connector Securely

This section includes:

- "Hardening the BSM Connector Platform" below
- "Permissions and Credentials" on the next page
- "Configuring BSM Connector to Use SSL" on page 42

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
- "Password Encryption" below
- "Using Secure Socket Layer (SSL) to Access BSM Connector" below

BSM Connector Users

Administrator user. By default, BSM Connector is installed with one administrator user account. This account is required when adding a BSM Connector integration to BSM. The administrator user name and password cannot be changed after the BSM Connector installation. The administrator user can access the BSM Connector user interface only and does not have access to BSM or other BSM applications.

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 BSM or other BSM applications. For more information about the **user** tool, see "Local User Configuration Tool" on page 100.

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

LW-SSO enables a user to log into BSM once and gain access to all BSM applications without being prompted to log in again. The applications inside BSM 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, BSM users can launch the BSM Connector user interface without having to provide additional credentials.

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

Password Encryption

BSM Connector passwords for servers accessed remotely are encrypted using a method called Triple Data Encryption Standard, or TDES. TDES applies the Data Encryption Algorithm on each 64-bit block of text three successive times, using either two or three different keys. As a result, unauthorized users cannot reproduce the original password in a reasonable amount of time.

Using Secure Socket Layer (SSL) to Access BSM Connector

BSM Connector uses SSL to encrypt communication between the server and the user interface. This requires installing a certificate from a Certificate Authority (CA). For more information, see "How to Prepare BSM Connector for Using SSL" on page 44

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 or log files.

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.
Log File (Windows)	NetBIOS	Windows permissions for read-only access to log file.	
Log File UNIX/ Linux)	Telnet, SSH, rlogin	Need shell access to the remote server. It is also necessary for the logged-in user to have permissions to run specific executable programs. Read-only file permissions to the target file system.	It is possible to restrict logged-in users' access by using UNIX group permissions for the command that BSM Connector would run. A list of the relevant commands for a particular operating system can be found in the templates.os files.

Configuring BSM Connector to Use SSL

BSM Connector uses SSL to encrypt communication with the user interface. Because the self-signed certificate that the BSM Connector uses by default is not from a trusted certificate authority, you normally see certificate errors in the Web browser when you connect to BSM Connector.

To secure the connection and prevent the certificate errors, you can obtain a server certificate from a trusted certificate authority and then import the certificate.

BSM Connector can also be configured to connect to a BSM server that requires SSL with or without a client certificate. If the BSM server requires SSL, BSM Connector sends metrics and topology data in secure mode. Event data is always sent securely.

The diagram "Secure Communication Overview " on the next page illustrates the secure communication channels between the browser-based BSM Connector user interface, the BSM Connector server, and BSM. The numbers in the diagram correspond to the sections that explain the configuration required to communicate securely:

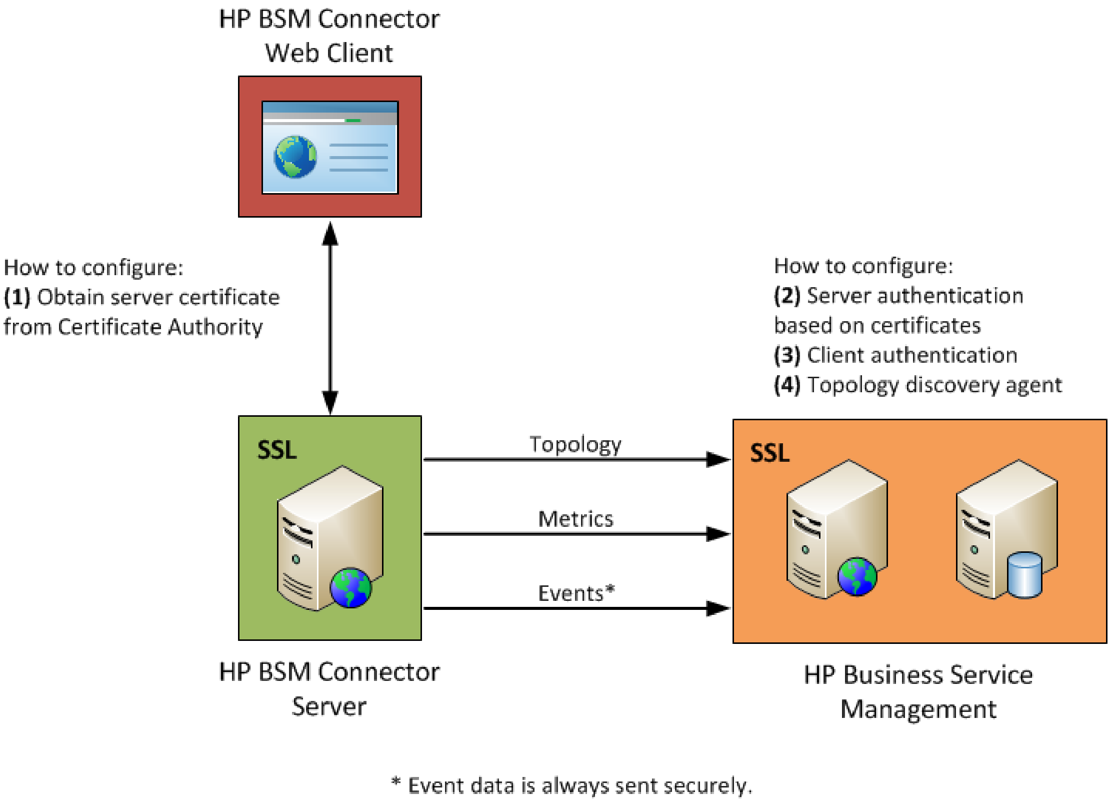
(1) "How to Prepare BSM Connector for Using SSL" on page 44

(2) "How to Configure BSM Connector to Send Metrics and Topology to a BSM Server Over a Secure Channel" on page 45

(3) "How to Configure BSM Connector to Connect to a BSM Server That Requires a Client Certificate" on page 47

(4) "How to Configure the Topology Discovery Agent in BSM Connector When the BSM Server Requires a Client Certificate" on page 51

Secure Communication Overview


HP BSM Connector Web Client
How to configure:
(1) Obtain server certificate from Certificate Authority
How to configure:
(2) Server authentication based on certificates
(3) Client authentication
(4) Topology discovery agent
SSL
Topology
Metrics
Events*
SSL
HP BSM Connector Server
HP Business Service Management
* Event data is always sent securely.

Keytool

BSM Connector is shipped with **Keytool**. Keytool is a key and certificate management utility. It enables users to administer their own public/private key pairs and associated certificates for authentication using digital signatures. It also enables users to cache the public keys of other persons and organizations they communicate with. This is installed in **<BSM Connector root directory>\java\bin**.

You can find out more about keytool at the Oracle Web site:

http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html

How to Prepare BSM Connector for Using SSL

BSM Connector is shipped with a self-signed certificate. You must replace the certificate with a certificate issued by your Certificate Authority (CA) as described below.

1. Obtain a server certificate from your CA issued to the BSM Connector server. Typically this certificate is issued in PKCS12 format with a password-protected private key.

2. Change the Tomcat configuration to use the PKCS12 certificate instead of the default self-signed Java certificate. Locate the following lines in the **<BSM Connector root directory>\Tomcat\conf\server.xml** configuration file:

   ```
   keystore="..\groups\serverKeystore" keystoreType="JKS"
   keystorePass="changeit"/>
   ```

   Change them to:

   ```
   keystore="path to server certificate in PKCS12 format"
   keystoreType="PKCS12"
   keystorePass="password for the private key"
   ```

   Restart Tomcat.

If you cannot obtain a server certificate from a CA in PKCS12 format, manually generate a server certificate using a Java Keystore (JKS) and have it signed by your CA. See the section "Creating a Keystore" in the BSM Hardening Guide.

How to Configure BSM Connector to Send Metrics and Topology to a BSM Server Over a Secure Channel

This task describes the steps involved in enabling secure communication between BSM Connector and BSM when the BSM server requires SSL. If the BSM server requires SSL, BSM Connector sends metrics and topology data in secure mode. Event data is always sent securely.

1. On the BSM Connector server, establish trust by importing the Certificate Authority (CA) root certificate of the authority that issued the BSM server certificates for the external access points (load balancer, reverse proxy, BSM Web server). Run the following command from the **<BSM Connector root directory>\java\bin** directory:

   ```
   keytool -import -trustcacerts -alias <your alias> -keystore
   ..\lib\security\cacerts -file <root CA certificate file>
   ```

2. In BSM, select **Admin > Integrations > BSM Connector Integrations**, and click the **New BSM Connector** button to add the BSM Connector instance. In the New BSM Connector page, make sure the following settings are configured:

   - **Main Settings:** In **HTTP port number** specify the *HTTPS* port number of the BSM Connector. **HTTP port number** and **HTTPS port number** must both contain the same value here. By default, BSM Connector uses the HTTPS port number 30000, but it can be configured to use a different port number instead.

   - **BSM Gateway Server Settings:** Check that the **Gateway server name/IP address** contains the correct server name and port (if not the default port 443). Enter the name of the external access point, if the BSM environment includes a load balancer, reverse proxy, or BSM Web server.

     > **Note:** When you configure BSM Connector in the New BSM Connector page in **Admin > Integrations > BSM Connector Integrations**, add the fully qualified domain name of the BSM external access point in the **Gateway server name/IP address** field. You must use the same name as in the BSM server certificate.

   - **Profile Settings:** Select the **Web Server Use SSL** checkbox. You must also select the **Use SSL** check box if BSM Connector is configured to use SSL.

3. *If the gateway servers are configured to use the Apache HTTP Server:* In BSM, select **Admin > Platform > Setup and Maintenance > Infrastructure Settings**, click **Foundations** and then select **Platform Administration**.

   Make sure all alphabetic characters in the following settings are lower-case:

   - Default Virtual Gateway Server for Application Users URL

   - Default Virtual Gateway Server for Data Collectors URL

   > **Note:** The BSM server certificate must also have been created using a lower-case host name.

4. On the computer that runs the BSM Connector, activate the HP Operations Agent manually using one of the following commands:

- On Windows:

  ```
  cscript %ovinstalldir%\bin\win64\OpC\install\opcactivate.vbs -srv
  <BSM gateway server>
  ```

- On Linux:

  ```
  /opt/OV/bin/OpC/install/opcactivate -srv <BSM gateway server>
  ```

How to Configure BSM Connector to Connect to a BSM Server That Requires a Client Certificate

This task describes the steps involved in enabling secure communication between BSM Connector and BSM when the BSM server requires a client certificate.

1. If you obtained the client certificate in Java keystore (JKS) format, copy it to the **<BSM Connector root directory>\templates.certificates** folder, and then continue with "Check the keystore contents" on page 49.

   > **Note:**
   >
   > - Make sure that the private key password is at least six characters long, and that the private key and keystore passwords are the same.
   > - In addition, make sure that the above keystore contains the CA certificate that issued it.

   Otherwise, perform the following steps (if you did not obtain the client certificate in JKS format). See also the section "Creating a Keystore" in the BSM Hardening Guide.

   a. Create a keystore under **<BSM Connector root directory>\templates.certificates** by running the following command from the **<BSM Connector root directory>\java\bin** directory:

      ```
      keytool -genkeypair -keyalg RSA -alias bsmc -keystore <BSM
      Connector root directory>\templates.certificates\.ks -storepass
      <your_keystore_password>
      ```

      For example:

      ```
      keytool -genkeypair -keyalg RSA -alias bsmc -keystore
      C:\BSMConnector\templates.certificates\.ks -storepass changeit
      What is your first and last name?
      [Unknown]: domain.name
      What is the name of your organizational unit?
      [Unknown]: dept
      What is the name of your organization?
      [Unknown]: XYZ Ltd
      What is the name of your City or Locality?
      [Unknown]:  New York
      What is the name of your State or Province?
      [Unknown]:  USA
      What is the two-letter country code for this unit?
      [Unknown]:  US
      Is CN=domain.name, OU=dept, O=XYZ Ltd, L=New York, ST=USA, C=US
      correct?
      [no]:  yes
      Enter key password for <BSM Connector>
      Press Enter to use the same password as the keystore password.
      ```

b. Create a certificate request for this keystore by running the following command from the **<BSM Connector root directory>\java\bin** directory:

```
keytool -certreq -alias bsmc -file c:\bsmc.csr -keystore <BSM
Connector root directory>\templates.certificates\.ks -storepass
<your_keystore_password>
```

For example:

```
keytool -certreq -alias bsmc -file c:\bsmc.csr -keystore
C:\BSMConnector\templates.certificates\.ks -storepass changeit
```

c. Have your certificate authority sign the certificate request. Copy/paste the contents of the .csr file into your Certificate Authority Web form.

d. Download the signed client certificate in BASE-64 format to **<BSM Connector root directory>\templates.certificates\clientcert.cer**.

e. Download the certificate authority certificate in BASE-64 format to **c:\**.

f. Import the certificate authority certificate into the JKS keystore by running the following command:

```
keytool -import -alias ca -file c:\ca.cer -keystore <BSM
Connector root directory>\templates.certificates\.ks -storepass
<your_keystore_password>
```

For example:

```
keytool -import -alias ca -file c:\ca.cer -keystore
C:\BSMConnector\templates.certificates\.ks -storepass changeit
Owner: CN=dept-CA, DC=domain.name
Issuer: CN=dept-CA, DC=domain.name
Serial number: 2c2721eb293d60b4424fe82e37794d2c
Valid from: Tue Jun 17 11:49:31 IDT 2008 until: Mon Jun 17
11:57:06 IDT 2013
Certificate fingerprints:
MD5:  14:59:8F:47:00:E8:10:93:23:1C:C6:22:6F:A6:6C:5B
SHA1:
17:2F:4E:76:83:5F:03:BB:A4:B9:96:D4:80:E3:08:94:8C:D5:4A:D5
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

g. Import the client certificate into the keystore by running the following command:

```
keytool -import -alias bsmc -file <BSM Connector root
directory>\templates.certificates\certnew.cer -keystore <BSM
Connector root directory>\templates.certificates\.ks -storepass
<your_keystore_password>
```

For example:

```
keytool -import -alias bsmc -file
c:\BSMConnector\templates.certificates\certnew.cer -keystore
C:\BSMConnector\templates.certificates\.ks -storepass changeit
```

The certificate reply is installed in the keystore **<BSM Connector root directory>\java\bin** directory.

2. Check the keystore contents

   Run the following command from the **<BSM Connector root directory>\java\bin** directory, and enter the keystore password:

   ```
   keytool -list -keystore <BSM Connector root
   directory>\templates.certificates\.ks
   ```

   For example:

   ```
   keytool -list -keystore C:\BSMConnector\templates.certificates\.ks
   Enter keystore password:  changeit

   Keystore type: jks
   Keystore provider: SUN

   Your keystore contains 2 entries

   ca, Mar 8, 2012, trustedCertEntry,
   Certificate fingerprint (MD5):
   14:59:8F:47:00:E8:10:93:23:1C:C6:22:6F:A6:6C:5B
   bsmc, Mar 8, 2012, keyEntry,
   Certificate fingerprint (MD5):
   C7:70:8B:3C:2D:A9:48:EB:24:8A:46:77:B0:A3:42:E1

   C:\BSMConnector\java\bin>
   ```

3. To use this keystore for client certificate, add the following lines to the **<BSM Connector root directory>\groups\master.config** file:

   ```
   _urlClientCert=<keystoreName>
   _urlClientCertPassword=<keystorePassword>
   ```

   For example:

   ```
   _urlClientCert=.ks
   _urlClientCertPassword=changeit
   ```

4. Save the changes to the file.

5. Restart the BSM Connector server:

   Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

6. In BSM, select **Admin > Integrations > BSM Connector Integrations**, and click the **New BSM Connector** button to add the BSM Connector instance.

Troubleshooting

If the connection between BSM Connector and BSM fails, check the following log files for errors:

- BSM Connector machine:

  <BSM Connector root directory>\log\bac_integration.log

- BSM Gateway Server:

  <HPBSM root directory>\log/topaz_all.ejb.log

  Check for the following error:

  ```
  javax.net.ssl.SSLHandshakeException:
  sun.security.validator.ValidatorException: PKIX path building
  failed: sun.security.provider.certpath.SunCertPathBuilderException:
  unable to find valid certification path to requested target
  ```

  Possible solutions: The error indicates one of the following:

  - The CA root certificate was not imported into the BSM JVM.
  - The BSM server was not restarted after the import. (A server restart is always required after a certificate import. This is a known limitation.)

How to Configure the Topology Discovery Agent in BSM Connector When the BSM Server Requires a Client Certificate

After configuring BSM Connector to connect to the BSM Gateway Server using a client certificate (see "How to Configure BSM Connector to Connect to a BSM Server That Requires a Client Certificate" on page 47), you need to perform the following steps for discovery to report topology to the BSM server.

1. Import the Certificate Authority (CA) certificate (or BSM server certificate) into the discovery trust store (**<BSM Connector root directory>\WEBINF\classes\security\MAMTrustStoreExp.jks**) with password (the default password for the discovery trust store is **logomania**, which encrypted, is: **[22,-8,116,-119,-107,64,49,93,-69,57,-13,-123,-32,-114,-88,-61]**):

   ```
   keytool -import -alias <your_CA> -keystore <BSM Connector root
   directory>\WEB-INF\classes\security\MAMTrustStoreExp.jks -storepass
   <your_keystore_password>
   ```

   For example:

   ```
   keytool -import -alias AMQA_CA -file c:\ca.cer -keystore
   C:\BSMConnector\WEB-INF\classes\security\MAMTrustStoreExp.jks -
   storepass logomania
   ```

   > **Note:** The private key password must be at least six characters, and the password for the private key and password for the keystore must be the same.

2. Check the contents of trustStore using the following command:

   ```
   keytool -list -keystore <BSM Connector root directory>\WEB-
   INF\classes\
   security\MAMTrustStoreExp.jks -storepass <your_keystore_password>
   Keystore type: <Keystore_type>
   Keystore provider: <Keystore_provider>
   Your keystore contains 2 entries
   mam, Nov 4, 2004, trustedCertEntry,Certificate fingerprint (MD5):
   <Certificate_fingerprint>
   amqa_ca, Dec 30, 2010, trustedCertEntry,Certificate fingerprint
   (MD5):
   <Certificate_fingerprint>
   ```

   For example:

   ```
   keytool -list -keystore C:\BSMConnector\WEB-INF\classes\security\
   MAMTrustStoreExp.jks -storepass logomania
   Keystore type: JKS
   Keystore provider: SUN
   Your keystore contains 2 entries
   mam, Nov 4, 2004, trustedCertEntry,
   Certificate fingerprint (MD5):
   ```

```
C6:78:0F:58:32:04:DF:87:5C:8C:60:BC:58:75:6E:F7
amqa_ca, Dec 30, 2010, trustedCertEntry,
Certificate fingerprint (MD5):
5D:47:4B:52:14:66:9A:6A:0A:90:8F:6D:7A:94:76:AB
```

3. Copy the BSM Connector client keyStore (.ks) from **<BSM Connector root directory>\templates.certificates** to **<BSM Connector root directory>\WEB-INF\classes\security\**.

4. In the **ssl.properties** file, update the **javax.net.ssl.keyStore** property to the keyStore name. For example, `javax.net.ssl.keyStore=.ks`.

5. Change the BSM Connector client keyStore password to match the Discovery password for keystore (default is logomania).

   ```
   keytool -storepasswd -new <Discovery_keystore_password>
   -keystore <BSM Connector root directory>\WEB-
   INF\classes\security\.ks
   -storepass <your_keystore_password>
   ```

   For example:

   ```
   keytool -storepasswd -new logomania -keystore C:\BSMConnector\WEB-
   INF\classes\security\.ks -storepass changeit
   ```

6. Change private key password to match Discovery password for keyStore:

   ```
   keytool -keypasswd -alias bsmc -keypass <your_keystore_password>
   -new <Discovery_keystore_password> -keystore <BSM Connector root
   directory>\WEB-INF\classes\security\.ks -storepass <your_keystore_
   password>
   ```

   For example:

   ```
   keytool -keypasswd -alias bsmc -keypass changeit -new logomania -
   keystore
   C:\BSMConnector\WEB-INF\classes\security\.ks -storepass logomania
   ```

7. Verify keystore using new password:

   ```
   keytool -list -v -keystore <BSM Connector root directory>\WEB-
   INF\classes\security\.ks -storepass <your_keystore_password>
   ```

   For example:

   ```
   keytool -list -v -keystore C:\BSMConnector\WEB-
   INF\classes\security\.ks
   -storepass logomania
   ```

8. Restart the BSM Connector server:

   Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

9. In BSM, select **Admin > Integrations > BSM Connector Integrations**, and click the **New BSM Connector** button to add the BSM Connector instance. In the **Profile Settings** pane,

make sure to select the **Web Server Use SSL** checkbox.

10. Check the topology appears in **BSM > Admin > RTSM Administration > IT Universe Manager > System Monitors** view.

Troubleshooting

- Check the bac-integration.log located in **<BSM Connector root directory>\logs\bac_integration\** for the following errors:

```
2010-12-30 11:03:06,399 [TopologyReporterSender]
(TopologyReporterSender.java:364)
ERROR - failed to run main topology agent.
topologyCommand=TopologyCommand{commandType
=RUN_SCRIPT, …
java.lang.IllegalArgumentException: cannot find script with
name=create_monitor.py at com.mercury.sitescope.integrations
.bac.topology.dependencies.DependenciesCrawler
.findDependencies(DependenciesCrawler.java:60)
at com.mercury.sitescope.integrations.bac.topology.
dependencies.ScriptDependenciesFinder
.find(ScriptDependenciesFinder.java:80)
at com.mercury.sitescope.integrations.bac.topology.
TopologyReporterSender.getDependencies
(TopologyReporterSender.java:552)
at com.mercury.sitescope.integrations.bac.topology.
TopologyReporterSender.send(TopologyReporterSender.java:347)
at com.mercury.sitescope.integrations.bac.topology.
TopologyReporterSender.run(TopologyReporterSender.java:304)
at java.lang.Thread.run(Thread.java:619)
```

  Verify that the certificate and keyStore passwords are identical.

Collecting and Viewing Metrics Data

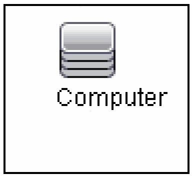
You enable capturing metrics data from third-party systems by configuring metrics policies.

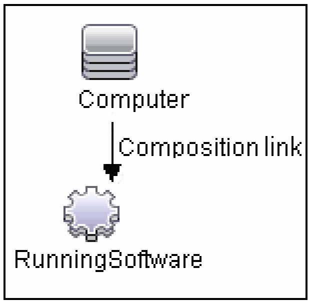
Metrics policies depend on the default monitor attributes and rules you define within the policies. The rules define the processing of incoming data and in combination with the defaults define the metrics forwarded to BSM.

Metrics can also be mapped to a topology to forward data to the correct CI hierarchy in BSM. You can configure topology settings for the policy by selecting a predefined scripts, or configuring your own topology script during policy creation.

Selecting a Topology

The following topology scripts are available for integrating events, metrics, and topology:

Topology Script	Events Integration	Metrics Integration	Topology Integration
No Topology Select if you do not want to send any topology (although metrics and event data is still sent).	yes	yes	not available
Computer - Monitor (legacy) Creates a topology with a Computer CI connected to a SiteScope CI with a Monitored By link. **Note:** The Computer - Monitor topology script has been deprecated. For new metrics integrations, use the Computer, Computer - Running Software, or a custom topology script. The Computer - Monitor topology integration requires that the names or IP addresses of the nodes that it adds to the RTSM are accessible through DNS resolution. To successfully add a Node CI specified in a policy's `Target` field to the RTSM, BSM Connector must be able to resolve the node's fully qualified domain name and IP address through a DNS service.	not available	yes	not available
Computer Creates a topology with a Computer CI. 	yes	yes	yes

Topology Script	Events Integration	Metrics Integration	Topology Integration
Computer - Running Software Creates a topology with a Computer CI and a Running Software CI connected to it with a **Composition** relationship. The following illustrates the topology created for the Computer - Running Software integration type which retrieves data from a third-party system: 	yes	yes	yes
Custom You create your own topology if you want the retrieved data to be forwarded to specific CIs and not one of the predefined topology scripts. **Note:** You should only select **Custom** if you are familiar with the Jython language, since you must create the topology script in Jython yourself. Depending on the data type you want to collect, we recommend that you select and edit one of the predefined scripts.	yes	yes	yes

Related Tasks

- "How to Collect Metrics with Computer - Monitor Topology" below
- "How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60
- "How to Collect Metrics Without Reporting Topology" on page 72
- "How to View Metrics in BSM User-Defined Reports" on page 83

How to Collect Metrics with Computer - Monitor Topology

The following task describes how to collect metrics data with Computer - Monitor topology.

This task includes the following steps:

- "Configure BSM integration" below
- "Plan the topology flow" below
- "Create a policy for metrics integration " below
- "Assign group permissions if using System Availability Management reports" below
- "View integration results" on the next page

1. **Configure BSM integration**

   Integrate BSM Connector and BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see "BSM Connector Integration Administration" in the BSM online help or the BSM Application Administration Guide.

2. **Plan the topology flow**

   Before you start, plan the following:

   - The metrics you will have.
   - The health indicators (HIs) you want to be created on the Computer CI you will report in topology.
   - For most of the default HIs, there are already HI and Key Performance Indicator (KPI) assignments, and there is no need to create new ones.
   - The metrics you want to map to the HIs.

3. **Create a policy for metrics integration**

   Make sure to add an indicator in the Defaults or Rules pages, for example by dragging it from the Indicators tab. This maps your metrics to HIs. Unmapped metrics are automatically mapped to the **Legacy System** HI.

   Select **Computer - Monitor** from the topology script list in the Topology page. You do not need to fill any topology script.

   > **Note:** The Computer - Monitor topology integration requires that the names or IP addresses of the nodes that it adds to the RTSM are accessible through DNS resolution. To successfully add a Node CI specified in a policy's `Target` field to the RTSM, BSM Connector must be able to resolve the node's fully qualified domain name and IP address through a DNS service.

4. **Assign group permissions if using System Availability Management reports**

   > **Note:** You only need to perform to assign group permissions if you have a System Availability Management license and plan to use System Availability Management reports, or if you plan to use BSM custom reports.

If you configure a generic integration monitor with a Metrics field mapping, you must assign for each defined user, permissions to view SiteScope groups and their subgroups in System Availability Management reports and custom reports. For more information, see the section on Permissions in the BSM Platform Administration Guide in the BSM Help.

5. **View integration results**

   After activating the policy in BSM Connector, you can view the results in the following applications:

   In Service Health:

   a. In BSM, select **Applications > Service Health > Top View**.

   b. In the drop down list, select:

      - **System Hardware Monitoring** to view the status of the Computer CI.
      - **System Monitors** view to view the monitor and its status.

   In System Availability Management Reports:

   If you have a System Availability Management license, you can also view the data of your integration in System Availability Management reports. In the different reports, specify a filter for the data that you want to be displayed in the graphs.

   Configure the filter to include the following values that you defined in the policy:

   - **Target**: Select a value that was defined in the **Target** field in the policy's Defaults or Rules pages.
   - **Monitor type:** Select a value that was defined in the **Monitor Type** field in the policy's Defaults or Rules pages.
   - **Monitor title/name**: Select a value that was defined in the **Monitor Name** field in the policy's Defaults or Rules pages.
   - **Measurement**: Select a value that was defined in the **Metric(n)** field (where n is an integer identifying the metric) in the policy's Defaults or Rules pages.

Example – Collecting Metrics With Computer - Monitor Topology

This example describes how to create a metrics integration policy to capture and forward metrics from a third-party system that monitors different disks to BSM using the Computer - Monitor topology script.

For a task related to this example, see "How to Collect Metrics with Computer - Monitor Topology" on page 55.

This example includes the following steps:

- "Design stage" on the next page
- "Create a log file policy for metrics integration" on the next page
- "View the integration results" on page 59

1. **Design stage**

   You have a third-party application that writes to a log file. It writes to the log the disk usage of different computers.

   Entries in the log file:

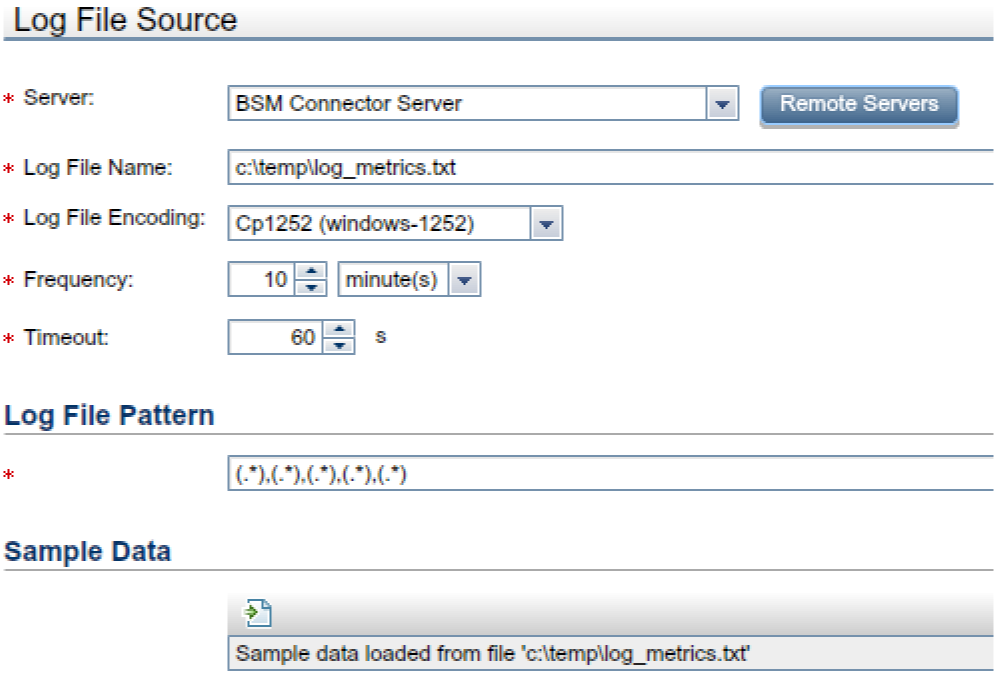
   ```
   labamrnd42,disk,d,65,warning
   labamrnd42,disk,d,70,warning
   labamrnd42,disk,d,70,warning
   ```

2. **Create a log file policy for metrics integration**

   Since the application writes to log files, create a log file policy for metrics integration. Use the **Computer - Monitor** topology script, and select the **HostDiskUtilization** indicator. There is no need to create assignments for this health indicator (HI) or Key Performance Indicator (KPI) since there are existing assignments for them.

   For details about log file data, "Integrating Log File Data into BSM" on page 147.

   a. In the **Source** page, specify the name of the log file and the log file pattern.



Log File Source

* Server: BSM Connector Server

* Log File Name: c:\temp\log_metrics.txt

* Log File Encoding: Cp1252 (windows-1252)

* Frequency: 10 minute(s)

* Timeout: 60 s

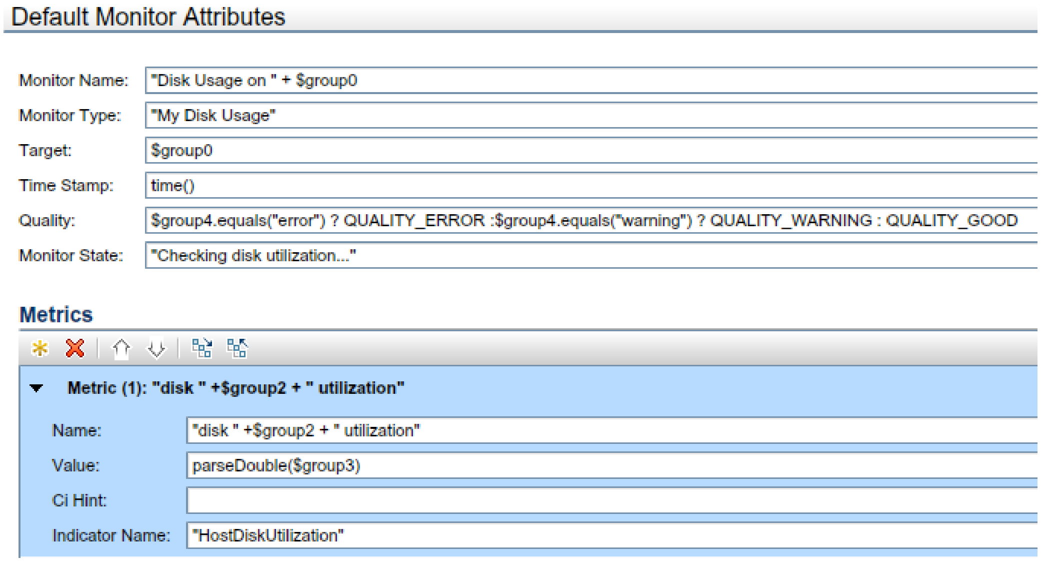
Log File Pattern

* (.*),(.*),(.*),(.*),(.*)

Sample Data

Sample data loaded from file 'c:\temp\log_metrics.txt'

b. In the **Defaults** page, assign default values to the metric attributes, for example:



Default Monitor Attributes

Monitor Name:	"Disk Usage on " + $group0
Monitor Type:	"My Disk Usage"
Target:	$group0
Time Stamp:	time()
Quality:	$group4.equals("error") ? QUALITY_ERROR :$group4.equals("warning") ? QUALITY_WARNING : QUALITY_GOOD
Monitor State:	"Checking disk utilization..."

Metrics

Metric (1): "disk " +$group2 + " utilization"

Name:	"disk " +$group2 + " utilization"
Value:	parseDouble($group3)
Ci Hint:	
Indicator Name:	"HostDiskUtilization"

Drag the **HostDiskUtilization** indicator from the Indicators tab and drop it in the **Indicator Name** field.

You can leave **CI Hint** field empty because BSM Connector sets it automatically.

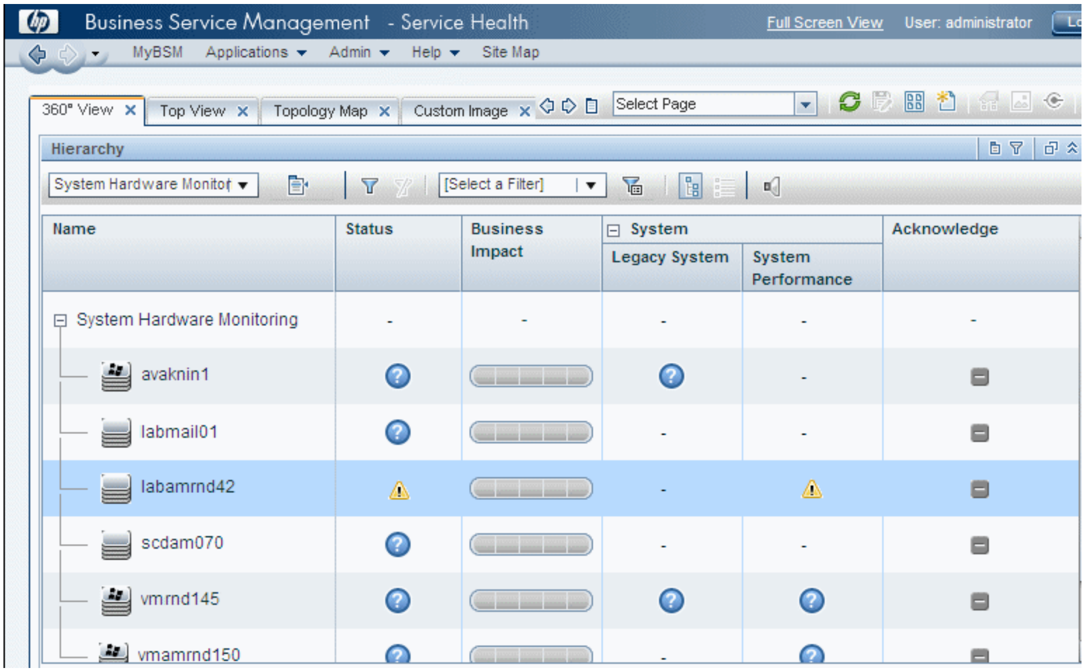
c. In the **Topology** page, select the **Computer - Monitor** topology script.

> **Note:** The Computer - Monitor topology integration requires that the names or IP addresses of the nodes that it adds to the RTSM are accessible through DNS resolution. To successfully add a Node CI specified in a policy's `Target` field to the RTSM, BSM Connector must be able to resolve the node's fully qualified domain name and IP address through a DNS service.

d. Save and activate the policy.

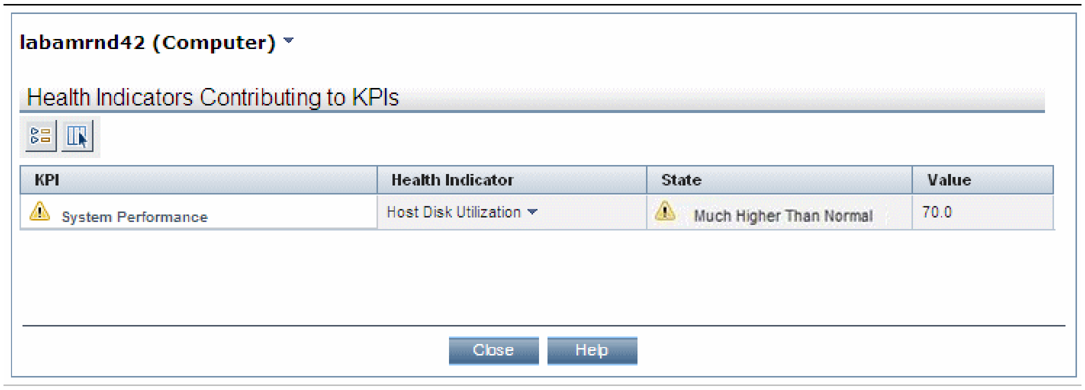
3. View the integration results

In BSM, select **Applications > Service Health** and view the target computer being monitored (`labamrnd42`) in the System Hardware Monitoring view.



Name	Status	Business Impact	System		Acknowledge
			Legacy System	System Performance	
System Hardware Monitoring	-	-	-	-	-
avaknin1	?	▭	?	-	▬
labmail01	?	▭	-	-	▬
labamrnd42	⚠	▭	-	⚠	▬
scdam070	?	▭	-	-	▬
vmrnd145	?	▭	?	?	▬
vmamrnd150	?	▭	-	?	▬

These are the results of the indicator status on the monitored machine:



KPI	Health Indicator	State	Value
⚠ System Performance	Host Disk Utilization	⚠ Much Higher Than Normal	70.0

How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data

The following task describes how to collect metrics with custom, Computer, or Computer - Running Software topology data.

This task includes the following steps:

- "Select an Indicator" on the next page
- "Define an HI Assignment" on the next page
- "Define a KPI Assignment for each CI type - optional" on page 62

- "Configure BSM integration" on the next page
- "Design the topology flow" on page 63
- "Create a policy for metrics integration " on page 63
- "Create or customize the topology script" on page 63
- "Assign group permissions if using System Availability Management reports" on page 63
- "View Integration Results" on page 64

1. ## Select an Indicator

   To view the status of a CI, you need a health indicator (HI) which provides a fine-grained measure of the health of the CI. In most cases, you want to view the HI state in BSM's Service Health. The HI is also used in Service Level Management (SLM). For details on HIs in Service Health and SLM, see the BSM Application Administration Guide.

   > **Note:** For alignment reasons, we recommend using an out-of-the-box HI; only create your own HI if you do not find an existing HI that fits your needs.

   To select an existing HI or create a new HI:

   a. In BSM, select **Admin > Service Health / Service Level Management > Repositories > Indicators**.

   b. From the CI type hierarchy in the left pane, select the CI type which you are going to report from your topology script. The assigned indicators for the CI type are displayed in the Indicators pane. When you select an indicator, its details are displayed in the right pane. For more details on topology scripts, see "Create or customize the topology script" on page 63.

   c. Check if you already have an existing HI that fits your requirements. If you do not, create a new one. For details on how to create HIs, see the BSM Application Administration Guide.

2. ## Define an HI Assignment

   After you select an HI, you need to define an HI assignment that will assign the HI to a CI. The assignment also defines which data samples will be captured by this HI and which business rule will be used to calculate the status of the HI according to the data samples.

   For more information on HI assignments, see the BSM Application Administration Guide.

   To define an HI assignment:

   a. In BSM, select **Admin > Service Health / Service Level Management > Assignments > Health Indicator Assignments**.

   b. From the CI type hierarchy in the left pane, select the CI type which you are going to report from your topology script. The assigned indicators for the CI type are displayed in the Indicators pane. When you select an indicator, its details are displayed in the right pane. For more details on topology scripts, see "Create or customize the topology script" on page 63.

   c. Create a new HI assignment. For details, see the BSM Application Administration Guide.

- In the **Condition** area, enter the policy name of your integration in the **Monitored By** property. This enables you to distinguish between the CIs reported by your integration to other CIs of this type.
- Select the HI you chose in the previous step ("Select an Indicator" on the previous page).
- Choose the business rule to use for the HI calculation. We recommend using the **SiteScope Worst Status Rule**. You can also use the **SiteScope Consecutive Worst Status Log** or **SiteScope Best Status Rule**.
- In the selector, enter the following:
  - `eti_id = (Binary) <<Health Indicator Type ID>>`
  - `ci_id = (Binary) <<CI ID>>`
  - `sampleType = (String) ss_t`

  The policy sends metrics samples (ss_t) that contain the same eti_id as your ETI and same CI ID as the CI's.

  The eti_id in the sample is sent by BSM Connector according to the Indicator Name in the policy.

  The ci_id is found by the CI resolver in BSM. For it to find the CI, it uses the CI hint sent by BSM Connector in the sample, according to the CI Hint in the policy.

3. Define a KPI Assignment for each CI type - optional

Verify whether you have an appropriate Key Performance Indicator (KPI) assignment, or create if one does not already exist.The assignment determines which KPI to put on the CI and for which HIs.

If you use one of the default HIs then there should already be a default KPI assignment for your HI and you do not need to create one.

For more information on KPI assignments, see the BSM Application Administration Guide.

To create a KPI assignment:

a. In BSM, select **Admin > Service Health > Repositories > Indicators**.

b. From the CI type hierarchy in the left pane, select the CI type which you are going to report from your topology script. The assigned indicators for the CI type are displayed in the Indicators pane. When you select an indicator, its details are displayed in the right pane. For more details, see "Create or customize the topology script" on the next page.

c. Create a new KPI assignment.

d. In the KPI assignment, the related HI should be the one you chose in "Select an Indicator" on the previous page.

> **Note:** If you also want to view the integration results in Service Level Management (SLM), you need to define the Service Level Agreement (SLA).

4. Configure BSM integration

Integrate BSM Connector and BSM:

Admin > Integrations > BSM Connector Integrations

For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

5. **Design the topology flow**

   When planning the design strategy for the custom topology flow, you need to consider the following:

   - **CIs to be reported to BSM**

     Consider the entities that you want to report to BSM. You might also want to report metrics and to view the health status of these entities in BSM's Service Health.

     For example, you have a third-party application named `TPA` that logs its metrics to a database. The entries in this database contain performance data on CPU and network usage of different computers. In this case, you will probably want to create a topology that reports Computer CIs to BSM.

   - **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. What is the relevant HI for the data being reported? If there is no such indicator, consider creating a new HI. Which KPI or HI assignments create the desired HIs and KPIs? If no such assignments exist, consider creating your own.

6. **Create a policy for metrics integration**

   Make sure to add the HI you chose previously in "Select an Indicator" on page 61 in the Defaults or Rules pages, for example by dragging it from the Indicators tab.

   Specify a hint for the CI in the CI Hint field in the Defaults or Rules pages. This hint is used by the CI resolver in BSM to recognize the CI to which the metrics should be attached.

   Depending on the topology you want to report, select **Custom**, **Computer**, or **Computer - Running Software** from the topology script list in the Topology page.

7. **Create or customize the topology script**

   For details on custom topology scripts, see "Configuring Scripts in Custom Topology Policies" on page 336.

8. **Assign group permissions if using System Availability Management reports**

   > **Note:** You only need to perform to assign group permissions if you have a System Availability Management license and plan to use System Availability Management reports, or if you plan to use BSM custom reports.

   If you configure a generic integration monitor with a Metrics field mapping, you must assign for each defined user, permissions to view SiteScope groups and their subgroups in System Availability Management reports and custom reports. For more information, see the section on Permissions in the BSM Platform Administration Guide in the BSM Help.

9. **View Integration Results**

   After activating the policy in BSM Connector, you can view the results in the following applications:

   - **Custom topologies.** Create a view in RTSM to view the results of the integration in BSM's Service Health or Service Level Management application. The view should describe the topology you defined in "Create or customize the topology script" on the previous page.

     For details on creating the view, see "Modeling Studio Page" in the Modeling Guide in the BSM Documentation Library.

     Computer topologies. You can view the results in the System Hardware Monitoring view.

     Computer - Running Software topologies. You can view the results in the System Software Monitoring view.

     If you defined the integration for SLM as well, you can view the integration results in SLM reports. For more information on SLM and on the reports, see "SLM Reports - Overview" the BSM User Guide.

   - If you have a System Availability Management license, you can also view the integration data in System Availability Management reports. In the different reports, specify a filter for the data that you want to be displayed in the graphs.

     Configure the filter to include the following values that you defined in the policy"How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60:

     - **Target**: Select a value that was defined in the **Target** field in the policy's Defaults or Rules pages.

     - **Monitor type:** Select a value that was defined in the **Monitor Type** field in the policy's Defaults or Rules pages.

     - **Monitor title/name**: Select a value that was defined in the **Monitor Name** field in the policy's Defaults or Rules pages.

     - **Measurement**: Select a value that was defined in the **Metric(n)** field (where n is an integer identifying the metric) in the policy's Defaults or Rules pages.

Example – Collecting Metrics With Custom Topology

This example describes how to create a metrics integration policy to capture and forward metrics from a third-party system that monitors different Oracle databases to BSM using the custom topology script. This script enables you to create your own topology.

For a task related to this example, see "How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60.

This example includes the following steps:

- "Design stage" on the next page
- "Select an indicator" on the next page

- "Define an HI assignment" below
- "Define a KPI Assignment" on page 67
- "Create a log file policy for metrics integration" on page 69
- "View the integration results" on page 71

1. ## Design stage

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

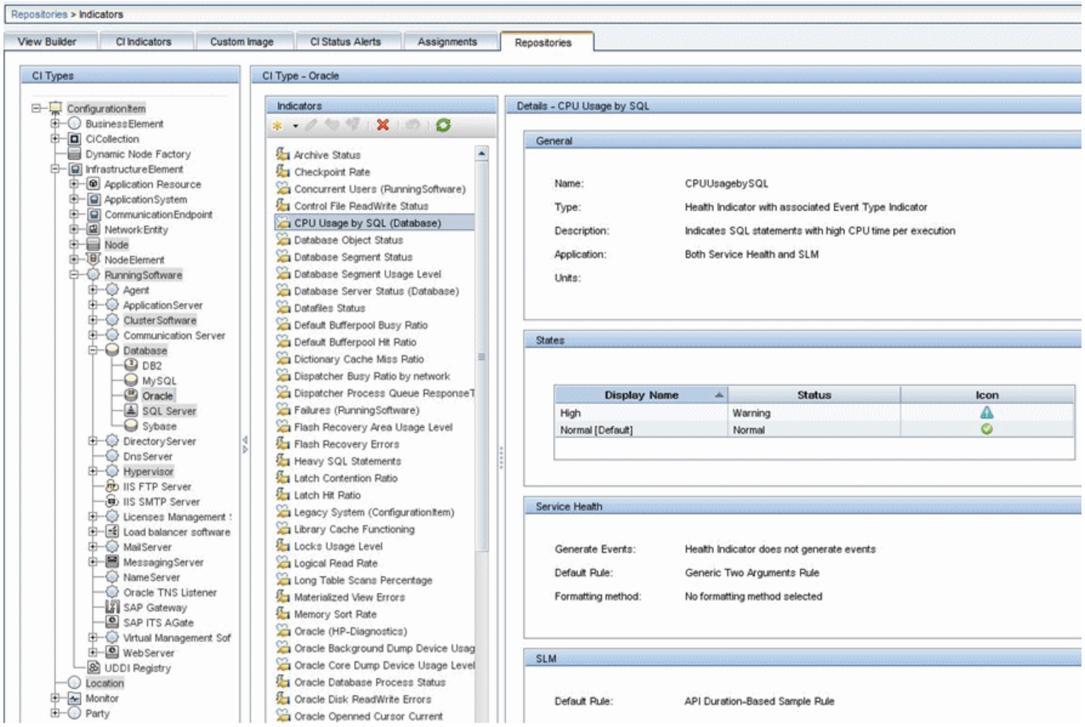
   Entries in the log file:

   ```
   amrnd153,27,1,good
   amrnd153,82,1,warning
   amrnd153,80,1,warning
   ```

2. ## Select an indicator

   In BSM, select **Admin > Service Health > Repositories > Indicators**. For the `My Oracle Monitoring` application, use the **CPU Usage by SQL (Database)** indicator.

   This indicator is defined on the Oracle CI type (the CI that will be reported), and is appropriate for the metric being read from the log. This metric describes the amount of CPU that Oracle uses.



ConfigurationItem
BusinessElement
CiCollection
Dynamic Node Factory
InfrastructureElement
Application Resource
ApplicationSystem
CommunicationEndpoint
NetworkEntity
Node
NodeElement
RunningSoftware
Agent
ApplicationServer
Cluster Software
Communication Server
Database
DB2
MySQL
Oracle
SQL Server
Sybase
DirectoryServer
DnsServer
Hypervisor
IIS FTP Server
IIS SMTP Server
Licenses Management :
Load balancer software
MailServer
MessagingServer
NameServer
Oracle TNS Listener
SAP Gateway
SAP ITS AGate
Virtual Management Sof
WebServer
UDDI Registry
Location
Monitor
Party

Archive Status
Checkpoint Rate
Concurrent Users (RunningSoftware)
Control File ReadWrite Status
CPU Usage by SQL (Database)
Database Object Status
Database Segment Status
Database Segment Usage Level
Database Server Status (Database)
Datafiles Status
Default Bufferpool Busy Ratio
Default Bufferpool Hit Ratio
Dictionary Cache Miss Ratio
Dispatcher Busy Ratio by network
Dispatcher Process Queue ResponseT
Failures (RunningSoftware)
Flash Recovery Area Usage Level
Flash Recovery Errors
Heavy SQL Statements
Latch Contention Ratio
Latch Hit Ratio
Legacy System (ConfigurationItem)
Library Cache Functioning
Locks Usage Level
Logical Read Rate
Long Table Scans Percentage
Materialized View Errors
Memory Sort Rate
Oracle (HP-Diagnostics)
Oracle Background Dump Device Usag
Oracle Core Dump Device Usage Level
Oracle Database Process Status
Oracle Disk ReadWrite Errors
Oracle Openned Cursor Current

Details - CPU Usage by SQL

General

Name: CPUUsagebySQL
Type: Health Indicator with associated Event Type Indicator
Description: Indicates SQL statements with high CPU time per execution
Application: Both Service Health and SLM
Units:

States

Display Name	Status	Icon
High	Warning	⚠
Normal [Default]	Normal	✓

Service Health

Generate Events: Health Indicator does not generate events
Default Rule: Generic Two Arguments Rule
Formatting method: No formatting method selected

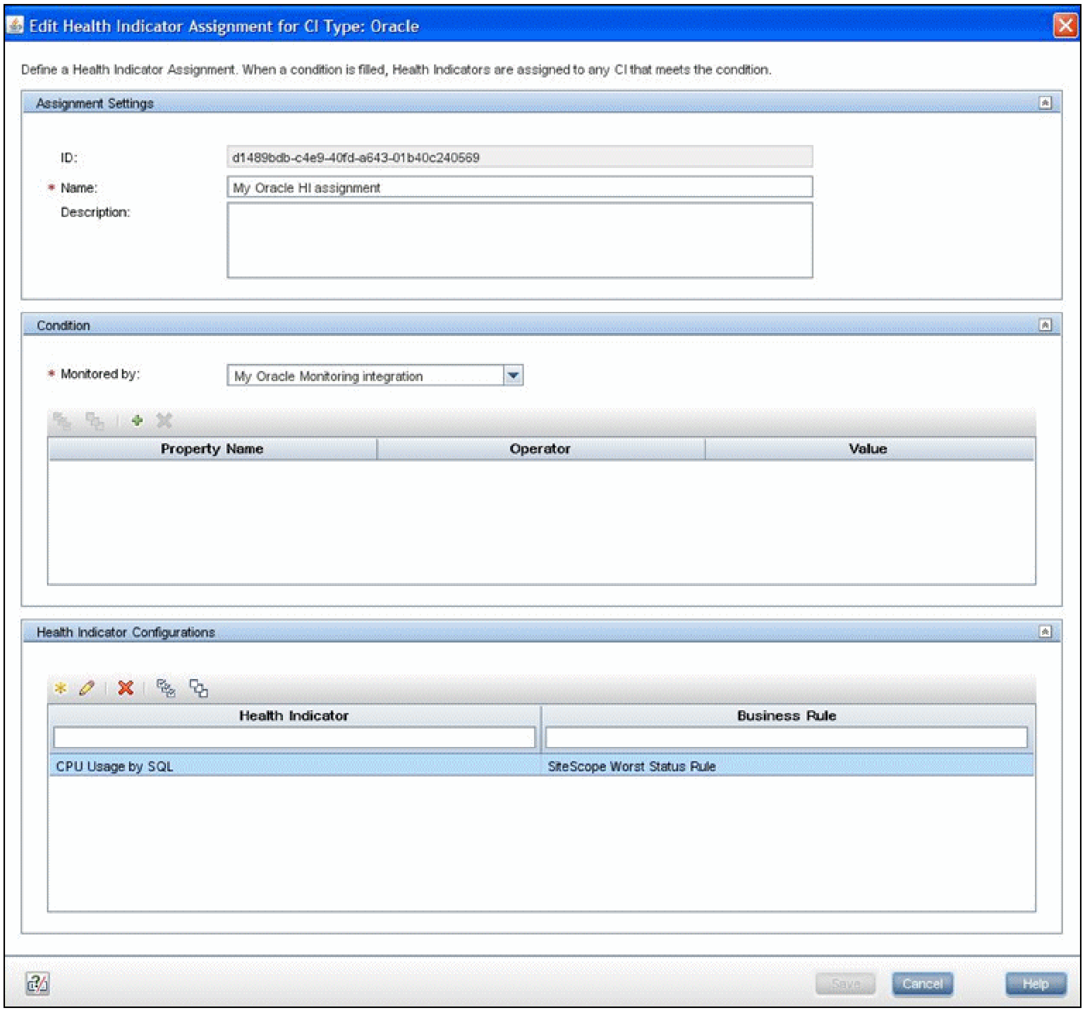
SLM

Default Rule: API Duration-Based Sample Rule

3. ## Define an HI assignment

   In BSM, select **Admin > Service Health > Assignments > Health Indicator Assignments** and create the indicator assignment.

In the **Monitored by** property, enter the name of the BSM Connector policy, `My Oracle Monitoring integration` in this example. This value is used to distinguish Oracle CIs reported by this integration from other Oracle CIs that are being reported. This assigns the `CPU Usage by SQL` indicator to Oracle CIs that are reported by this integration only.



Edit Health Indicator Assignment for CI Type: Oracle

Define a Health Indicator Assignment. When a condition is filled, Health Indicators are assigned to any CI that meets the condition.

Assignment Settings

ID: d1489bdb-c4e9-40fd-a643-01b40c240569

* Name: My Oracle HI assignment

Description:

Condition

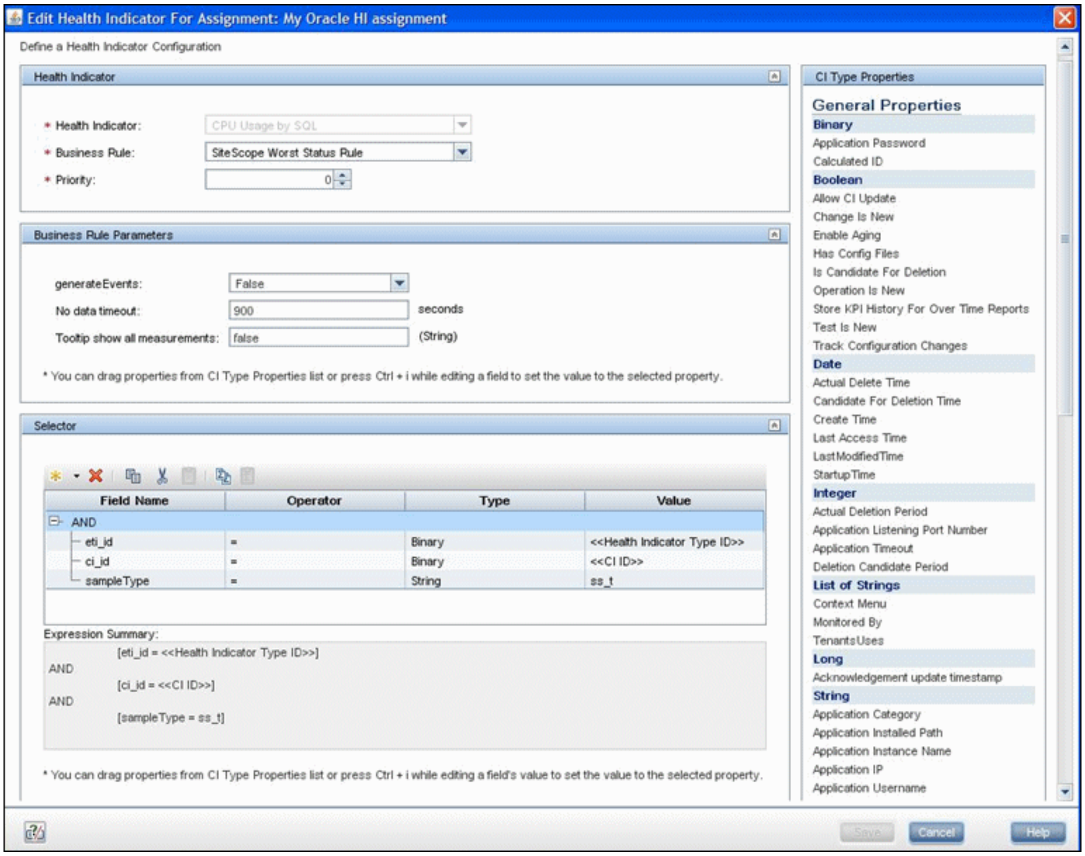
* Monitored by: My Oracle Monitoring integration

Property Name	Operator	Value

Health Indicator Configurations

Health Indicator	Business Rule
CPU Usage by SQL	SiteScope Worst Status Rule

The following page opens when you edit the indicator in this assignment:



Edit Health Indicator For Assignment: My Oracle HI assignment

Define a Health Indicator Configuration

Health Indicator

* Health Indicator:	CPU Usage by SQL
* Business Rule:	SiteScope Worst Status Rule
* Priority:	0

Business Rule Parameters

generateEvents:	False	
No data timeout:	900	seconds
Tooltip show all measurements:	false	(String)

* You can drag properties from CI Type Properties list or press Ctrl + i while editing a field to set the value to the selected property.

Selector

Field Name	Operator	Type	Value
AND			
eti_id	=	Binary	<<Health Indicator Type ID>>
ci_id	=	Binary	<<CI ID>>
sampleType	=	String	ss_t

Expression Summary:

```
          [eti_id = <<Health Indicator Type ID>>]
AND
          [ci_id = <<CI ID>>]
AND
          [sampleType = ss_t]
```

* You can drag properties from CI Type Properties list or press Ctrl + i while editing a field's value to set the value to the selected property.

CI Type Properties

General Properties

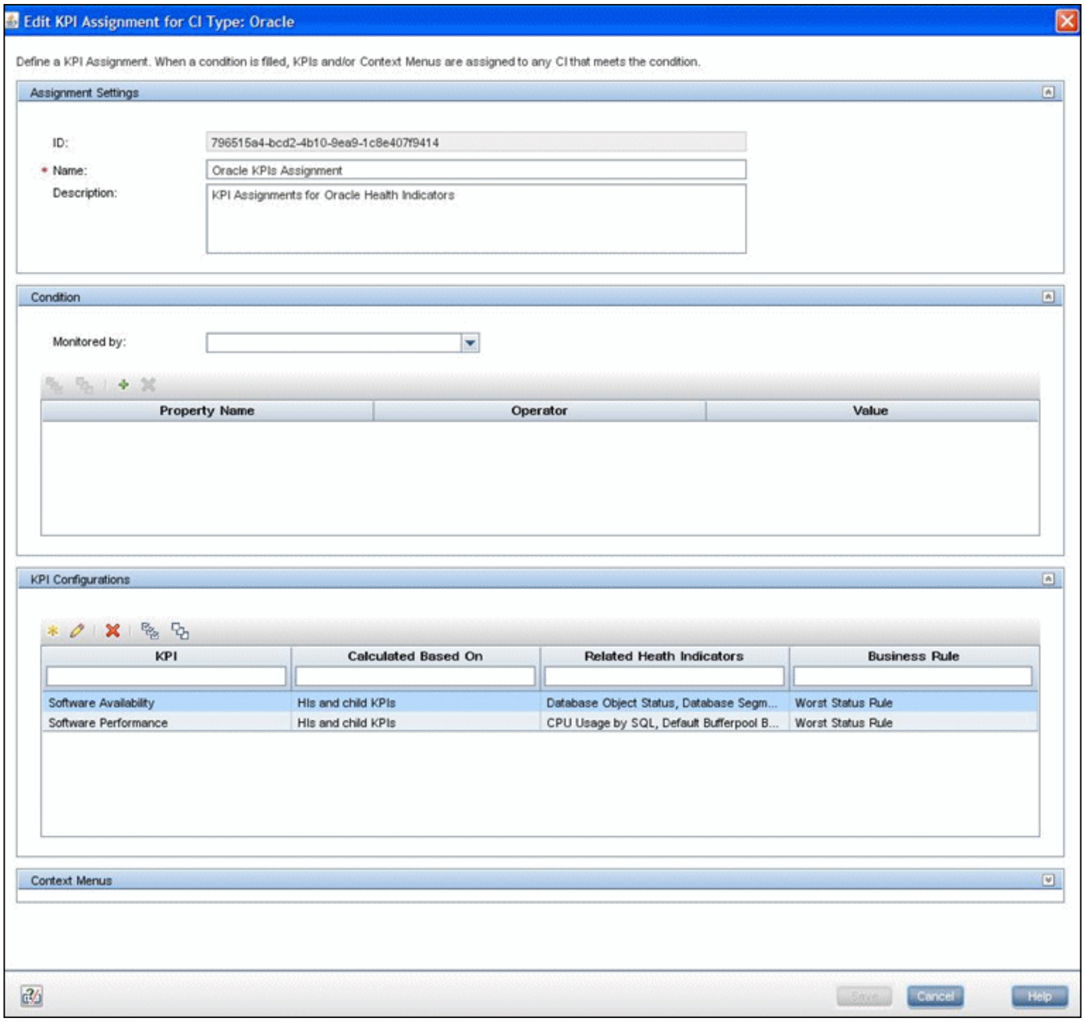
Binary
Application Password
Calculated ID
Boolean
Allow CI Update
Change Is New
Enable Aging
Has Config Files
Is Candidate For Deletion
Operation Is New
Store KPI History For Over Time Reports
Test Is New
Track Configuration Changes
Date
Actual Delete Time
Candidate For Deletion Time
Create Time
Last Access Time
LastModifiedTime
StartupTime
Integer
Actual Deletion Period
Application Listening Port Number
Application Timeout
Deletion Candidate Period
List of Strings
Context Menu
Monitored By
TenantsUses
Long
Acknowledgement update timestamp
String
Application Category
Application Installed Path
Application Instance Name
Application IP
Application Username

The CPU Usage by SQL indicator is calculated using the SiteScope Worst Status Rule. The selector defines that data of type ss_t (the metrics data type) with the same ci_id and eti_id as the current CI and ETI that is reported by the integration.

4. Define a KPI Assignment

Since you used an out-of-the-box indicator, you do not need to create a KPI assignment as there is an existing Service Health KPI assignment on Oracle CIs.

In BSM, select **Admin > Service Health > Assignments > KPI Assignments**, and in the CI Type tree select **Oracle** and choose **Oracle KPI Assignment**.



Edit KPI Assignment for CI Type: Oracle

Define a KPI Assignment. When a condition is filled, KPIs and/or Context Menus are assigned to any CI that meets the condition.

Assignment Settings

ID:	796515a4-bcd2-4b10-9ea9-1c8e407f9414
* Name:	Oracle KPIs Assignment
Description:	KPI Assignments for Oracle Health Indicators

Condition

Monitored by:

Property Name	Operator	Value

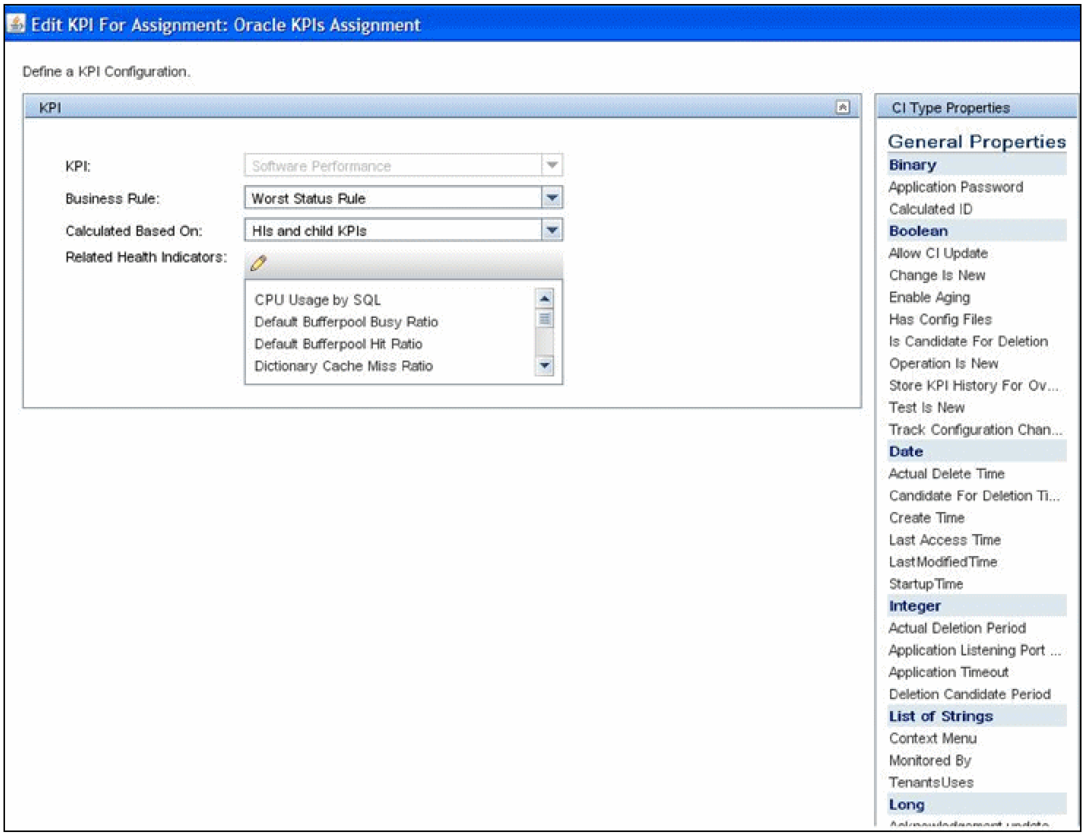
KPI Configurations

KPI	Calculated Based On	Related Heath Indicators	Business Rule
Software Availability	HIs and child KPIs	Database Object Status, Database Segm...	Worst Status Rule
Software Performance	HIs and child KPIs	CPU Usage by SQL, Default Bufferpool B...	Worst Status Rule

Context Menus

Select the **Software Performance** KPI:

You can see that one of the indicators related to this KPI is the `CPU Usage by SQL` indicator which you used.



Edit KPI For Assignment: Oracle KPIs Assignment

Define a KPI Configuration.

KPI

KPI: Software Performance

Business Rule: Worst Status Rule

Calculated Based On: HIs and child KPIs

Related Health Indicators:

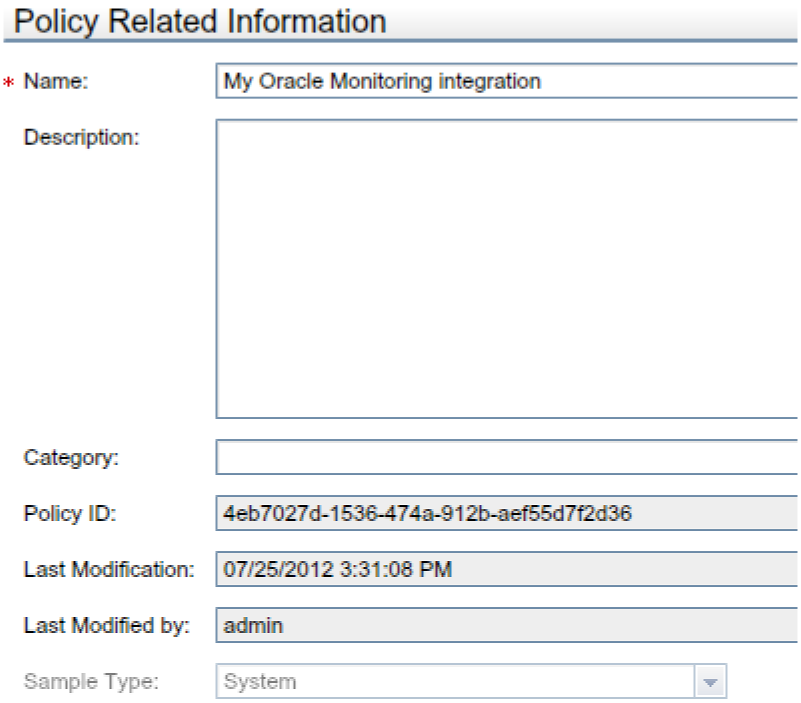
CPU Usage by SQL
Default Bufferpool Busy Ratio
Default Bufferpool Hit Ratio
Dictionary Cache Miss Ratio

5. Create a log file policy for metrics integration

Since the application writes to log files, create a log file policy named `My Oracle Monitoring integration` for metrics integration. The topology that you want to report includes Oracle CIs for which HI status will be set.

For details about log file data, "Integrating Log File Data into BSM" on page 147.

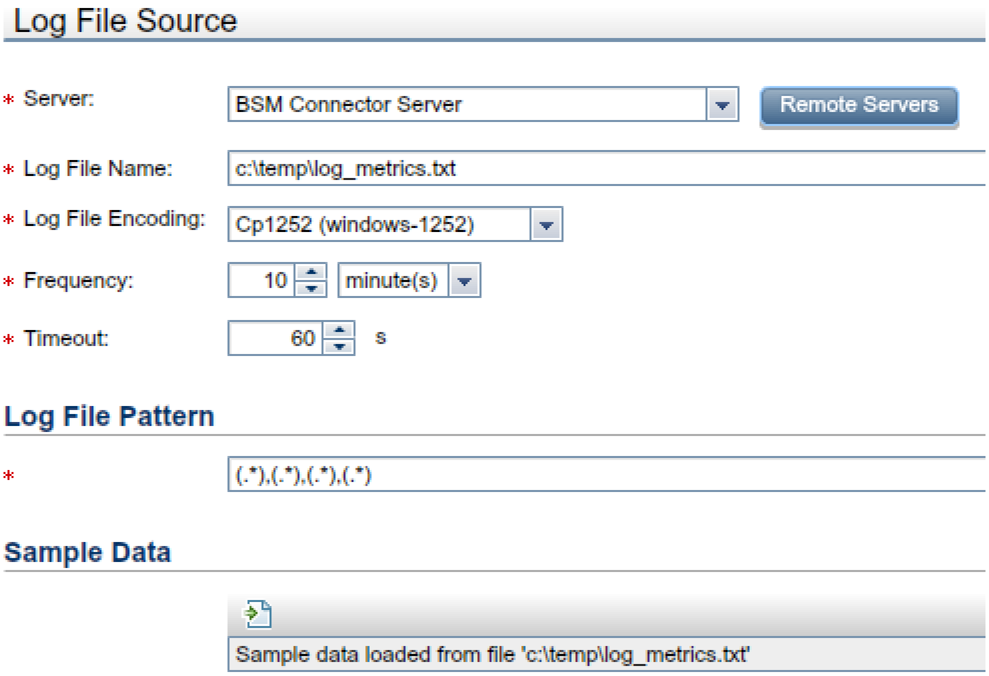
a. In the **Properties** page, type `My Oracle Monitoring integration` in the Name field.



Policy Related Information

* Name:	My Oracle Monitoring integration
Description:	
Category:	
Policy ID:	4eb7027d-1536-474a-912b-aef55d7f2d36
Last Modification:	07/25/2012 3:31:08 PM
Last Modified by:	admin
Sample Type:	System

b. In the **Source** page, specify the name of the log file and the log file pattern.



Log File Source

* Server:	BSM Connector Server
* Log File Name:	c:\temp\log_metrics.txt
* Log File Encoding:	Cp1252 (windows-1252)
* Frequency:	10 minute(s)
* Timeout:	60 s

Log File Pattern

*	(.*),(.*),(.*),(.*)

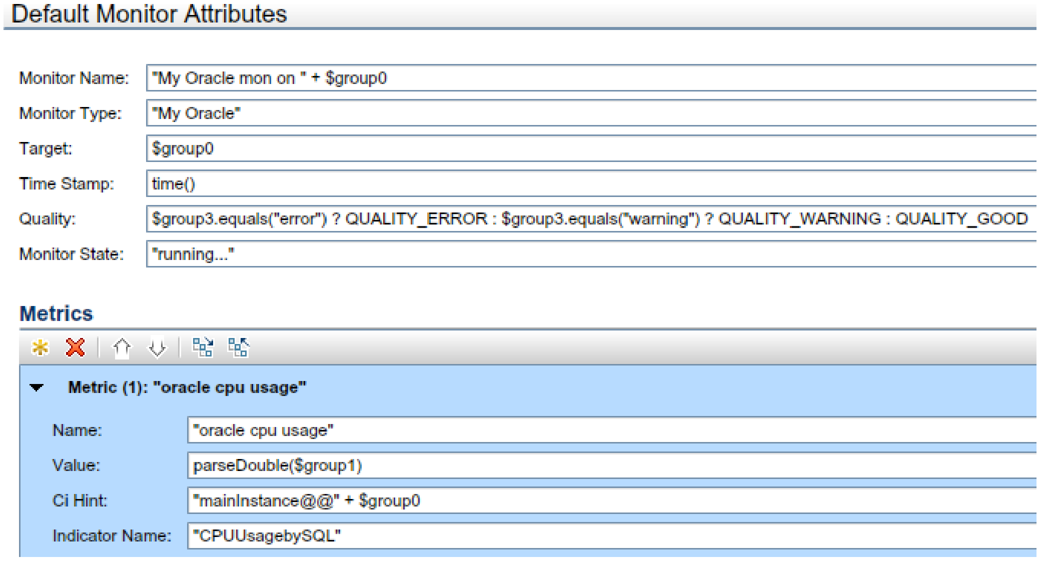
Sample Data

Sample data loaded from file 'c:\temp\log_metrics.txt'

c. In the **Defaults** page, assign default values to the metric attributes:

- The monitor name is `My Oracle mon on $group0` where `$group0` is also the target computer on which the Oracle database is running.

- You define a new monitor type: `My Oracle`.
- The quality that is sent is conditional and depends on what is written in the log file.
- The metric name is `oracle cpu usage` and its value is taken from the log file.
- The CI hint is in the format `<<oracle sid>>@@<<computer name>>`. The CI hint is used by the CI Resolver in BSM to find the CI to which this metric should be attached.
- The ETI to which the metric is mapped is `CPUUsagebySQL`.



Default Monitor Attributes

Monitor Name:	"My Oracle mon on " + $group0
Monitor Type:	"My Oracle"
Target:	$group0
Time Stamp:	time()
Quality:	$group3.equals("error") ? QUALITY_ERROR : $group3.equals("warning") ? QUALITY_WARNING : QUALITY_GOOD
Monitor State:	"running..."

Metrics

Metric (1): "oracle cpu usage"

Name:	"oracle cpu usage"
Value:	parseDouble($group1)
Ci Hint:	"mainInstance@@" + $group0
Indicator Name:	"CPUUsagebySQL"

d. In the **Topology** page, select **Custom** and create the topology script.

> **Note:** The topology script to use is available in a Jython file attached to this PDF. To view the attachment, select **View > Navigation Panels > Attachments**, and select **jython_metrics_custom_topo.py**.

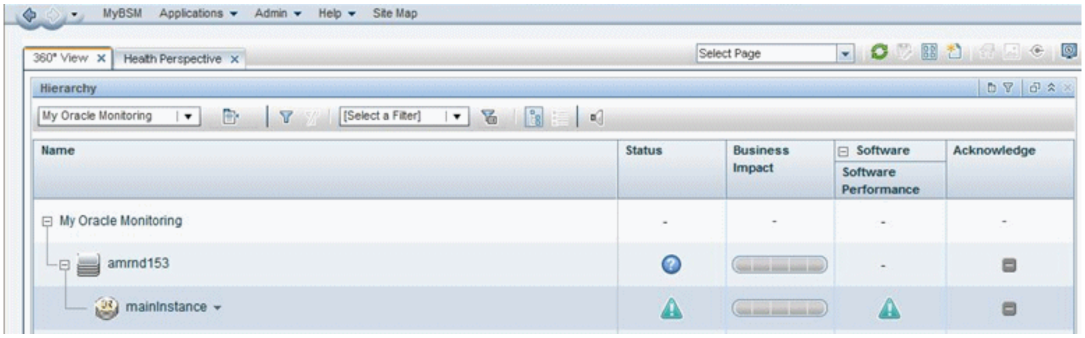
In the custom script, you can see that:

- `system_lib.createNode(Framework)` creates the node on which the database is running.
- `modeling.createDatabaseOSH` creates the Oracle CI.
- `mainInstance` is the SID of the Oracle you used.
- `ems_lib.addMonitoredByForThirdPartySoftware` sets the **Monitored by** attribute of the Oracle CIs to the values `BSM Connector` and the name of the policy (`My Oracle Monitoring integration` in this example), which is the condition you gave in the indicator assignment (see "Define an HI assignment" on page 65).

e. Save and activate the policy.

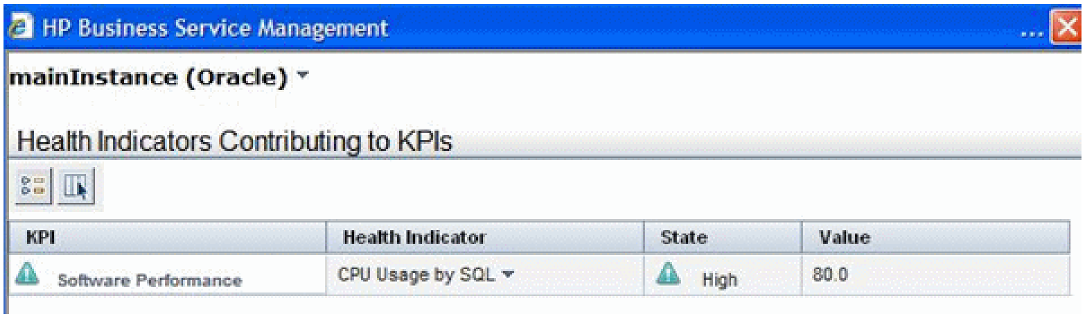
6. View the integration results

In BSM, select **Applications > Service Health** and manually create a view for the integration. For example, here we created a view named `My Oracle Monitoring` (the integration result is also displayed in System Software Monitoring view):



Name	Status	Business Impact	Software	Acknowledge
			Software Performance	
My Oracle Monitoring	-	-	-	-
ammd153	?		-	
mainInstance	⚠		⚠	

These are the results of the indicator status on the monitored machine:



mainInstance (Oracle)

Health Indicators Contributing to KPIs

KPI	Health Indicator	State	Value
⚠ Software Performance	CPU Usage by SQL	⚠ High	80.0

The state and value are the same as you assigned in the policy.

How to Collect Metrics Without Reporting Topology

The following tasks describe how to collect metrics with no topology data.

This task includes the following steps:

- "Select an Indicator" below
- "Define an HI Assignment" on the next page
- "Define a KPI Assignment for each CI type - optional" on page 74
- "Configure BSM integration" on page 74
- "Check the existing topology in BSM" on page 74
- "Create a policy for metrics integration " on page 75
- "Assign group permissions if using System Availability Management reports" on page 75
- "View Integration Results" on page 75

1. Select an Indicator

To view the status of a CI, you need a health indicator (HI) which provides a fine-grained measure of the health of the CI. In most cases, you want to view the HI state in BSM's Service Health. The HI is also used in Service Level Management (SLM). For details on HIs in Service Health and SLM, see the BSM Application Administration Guide.

> **Note:** For alignment reasons, we recommend using an out-of-the-box HI; only create your own HI if you do not find an existing HI that fits your needs.

To select an existing HI or create a new HI:

a. In BSM, select **Admin > Service Health / Service Level Management > Repositories > Indicators**.

b. From the CI type hierarchy in the left pane, select the CI type which you are going to use. The assigned indicators for the CI type are displayed in the Indicators pane. When you select an indicator, its details are displayed in the right pane.

c. Check if you already have an existing HI that fits your requirements. If you do not, create a new one. For details on how to create HIs, see the BSM Application Administration Guide.

2. Define an HI Assignment

After you select an HI, you need to define an HI assignment that will assign the HI to a CI. The assignment also defines which data samples will be captured by this HI and which business rule will be used to calculate the status of the HI according to the data samples.

For more information on HI assignments, see the BSM Application Administration Guide.

To define an HI assignment:

a. In BSM, select **Admin > Service Health / Service Level Management > Assignments > Health Indicator Assignments**.

b. From the CI type hierarchy in the left pane, select the CI type which you are going to use. The assigned indicators for the CI type are displayed in the Indicators pane. When you select an indicator, its details are displayed in the right pane.

c. Create a new HI assignment. For details, see the BSM Application Administration Guide.

   - In the **Condition** area, enter the policy name of your integration in the **Monitored By** property. This enables you to distinguish between the CIs reported by your integration to other CIs of this type.
   - Select the HI you chose in the previous step ("Select an Indicator" on the previous page).
   - Choose the business rule to use for the HI calculation. We recommend using the **SiteScope Worst Status Rule**. You can also use the **SiteScope Consecutive Worst Status Log** or **SiteScope Best Status Rule**.
   - In the selector, enter the following:
     - `eti_id = (Binary) <<Health Indicator Type ID>>`
     - `ci_id = (Binary) <<CI ID>>`
     - `sampleType = (String) ss_t`

For an example, see "Example – Create a Metrics Policy Not Reporting Topology" on page 76.

The policy sends metrics samples (ss_t) that contain the same eti_id as your ETI and same CI ID as the CI's.

The eti_id in the sample is sent by BSM Connector according to the Indicator Name in the policy.

The ci_id is found by the CI resolver in BSM. For it to find the CI, it uses the CI hint sent by BSM Connector in the sample, according to the CI Hint in the policy.

3. Define a KPI Assignment for each CI type - optional

Verify whether you have an appropriate Key Performance Indicator (KPI) assignment, or create if one does not already exist.The assignment determines which KPI to put on the CI and for which HIs.

If you use one of the default HIs then there should already be a default KPI assignment for your HI and you do not need to create one.

For more information on KPI assignments, see the BSM Application Administration Guide.

To create a KPI assignment:

a. In BSM, select **Admin > Service Health > Repositories > Indicators**.

b. From the CI type hierarchy in the left pane, select the CI type which you are going to use. The assigned indicators for the CI type are displayed in the Indicators pane. When you select an indicator, its details are displayed in the right pane.

c. Create a new KPI assignment.

d. In the KPI assignment, the related HI should be the one you chose in "Select an Indicator" on page 72.

> **Note:** If you also want to view the integration results in Service Level Management (SLM), you need to define the Service Level Agreement (SLA).

4. Configure BSM integration

Integrate BSM Connector and BSM:

Admin > Integrations > BSM Connector Integrations

For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

5. Check the existing topology in BSM

When collecting metrics without topology data, you need to consider the following:

- **CIs available in BSM**

  Since you are using the No Topology option, you probably already have CIs in the RTSM to which you want to report data using BSM Connector policies.

- **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. What is the relevant HI for the data being reported? If there is no such HI, consider creating a new HI. Which KPI or HI assignments create the desired HIs and KPIs? If no such assignments exist, consider creating your own.

For example, if you have data on CPU usage and network usage, you can use the `CPU Load` and `Interface Utilization` HIs that are defined for the Computer CI type, and you can use the `System Performance` KPI. Check if there are HI and KPI assignments that meet your needs, and if not, consider creating them.

6. ### Create a policy for metrics integration

   Make sure to add the HI you chose previously in "Select an Indicator" on page 72 in the Defaults or Rules pages, for example by dragging it from the Indicators tab.

   Specify a hint for the CI in the CI Hint field in the Defaults or Rules pages. This hint is used by the CI resolver in BSM to recognize the CI to which the metrics should be attached.

   Select No Topology from the topology script list in the Topology page.

7. ### Assign group permissions if using System Availability Management reports

   > **Note:** You only need to perform to assign group permissions if you have a System Availability Management license and plan to use System Availability Management reports, or if you plan to use BSM custom reports.

   If you configure a generic integration monitor with a Metrics field mapping, you must assign for each defined user, permissions to view SiteScope groups and their subgroups in System Availability Management reports and custom reports. For more information, see the section on Permissions in the BSM Platform Administration Guide in the BSM Help.

8. ### View Integration Results

   After activating the policy in BSM Connector, you can view the results in the following applications:

   - Create a view in RTSM to view the results of the integration in BSM's Service Health or Service Level Management application. The view should describe the topology you defined in "How to Collect Metrics Without Reporting Topology" on page 72.

     For details on creating the view, see "Modeling Studio Page" in the Modeling Guide in the BSM Documentation Library.

     If you defined the integration for SLM as well, you can view the integration results in SLM reports. For more information on SLM and on the reports, see the BSM Application Administration Guide.

   - If you have a System Availability Management license, you can also view the integration data in System Availability Management reports. In the different reports, specify a filter for the data that you want to be displayed in the graphs.

     Configure the filter to include the following values that you defined in the policy "How to Collect Metrics Without Reporting Topology" on page 72:

- **Target**: Select a value that was defined in the **Target** field in the policy's Defaults or Rules pages.
- **Monitor type:** Select a value that was defined in the **Monitor Type** field in the policy's Defaults or Rules pages.
- **Monitor title/name**: Select a value that was defined in the **Monitor Name** field in the policy's Defaults or Rules pages.
- **Measurement**: Select a value that was defined in the **Metric(n)** field (where n is an integer identifying the metric) in the policy's Defaults or Rules pages.

Example – Create a Metrics Policy Not Reporting Topology

This example describes how to create a metrics integration policy to capture and forward metrics from a third-party system that monitors different Oracle databases to BSM using a custom topology script. This script enables you to create your own topology.

This example describes how to create a metrics integration policy to capture and forward metrics from a third-party system that monitors different Oracle databases to BSM using the No Topology option. This option is used to send metrics when a topology already exists in BSM, and there is no need to report the CIs.

For a task related to this example, see "How to Collect Metrics Without Reporting Topology" on page 72.

This example includes the following steps:

- "Design stage" below
- "Select an indicator" below
- "Define an HI assignment" on the next page
- "Define a KPI Assignment" on page 78
- "Create a log file policy for metrics integration" on page 80
- "View the integration results" on page 82

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.

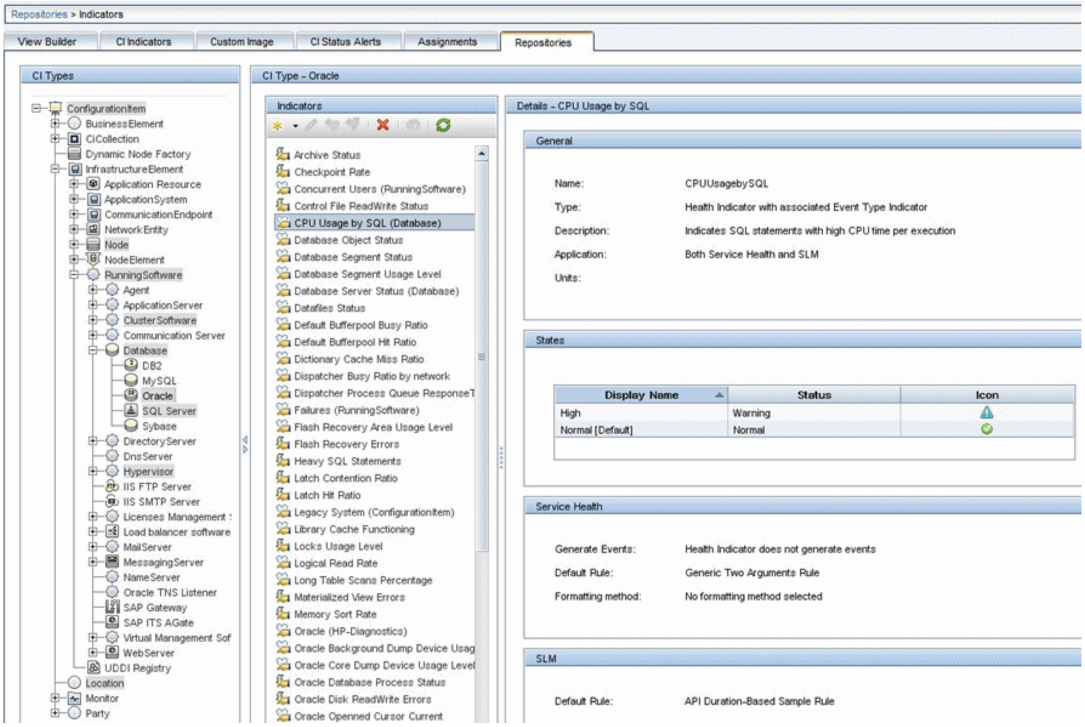
Entries in the log file:

```
amrnd153,27,1,good
amrnd153,82,1,warning
amrnd153,80,1,warning
```

2. Select an indicator

In BSM, select **Admin > Service Health > Repositories > Indicators**. For the `My Oracle Monitoring` application, use the **CPU Usage by SQL (Database)** indicator.

This indicator is defined on the Oracle CI type (the CI that will be reported), and is appropriate for the metric being read from the log. This metric describes the amount of CPU that Oracle uses.



Details - CPU Usage by SQL

General

Name: CPUUsagebySQL
Type: Health Indicator with associated Event Type Indicator
Description: Indicates SQL statements with high CPU time per execution
Application: Both Service Health and SLM
Units:

States

Display Name	Status	Icon
High	Warning	⚠
Normal [Default]	Normal	✓

Service Health

Generate Events: Health Indicator does not generate events
Default Rule: Generic Two Arguments Rule
Formatting method: No formatting method selected

SLM

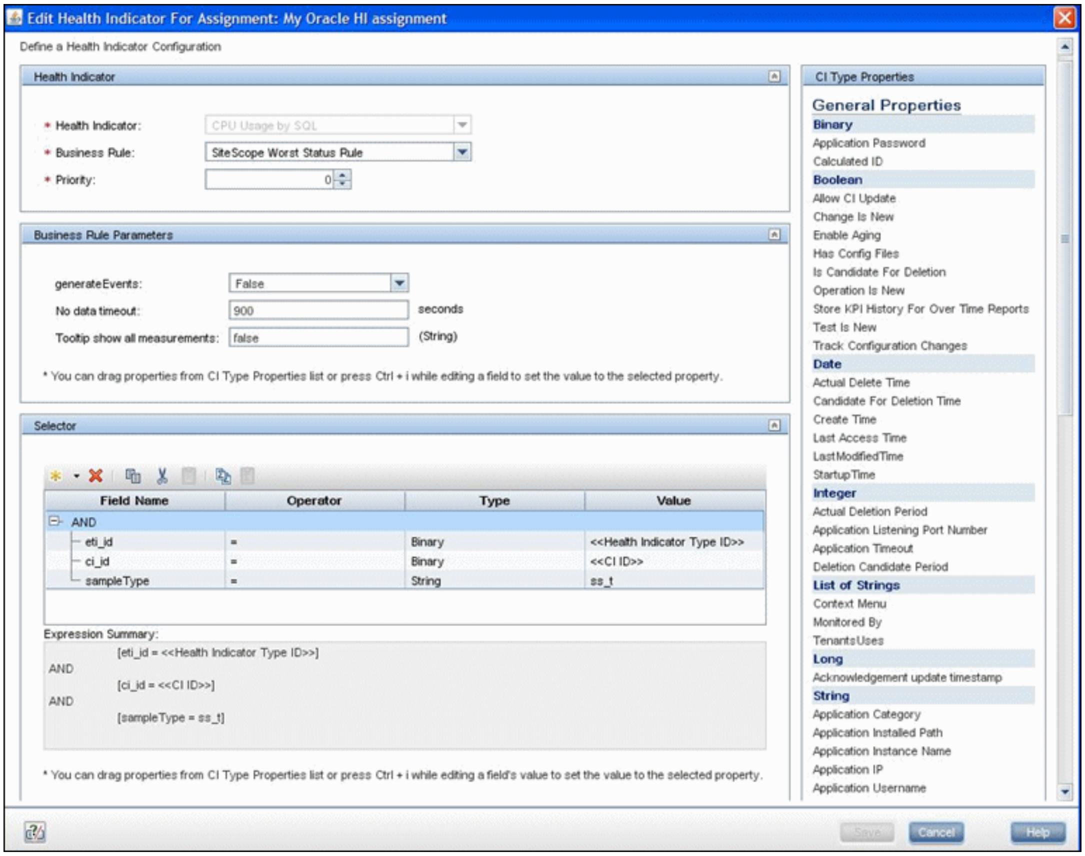
Default Rule: API Duration-Based Sample Rule

3. ## Define an HI assignment

   In BSM, select **Admin > Service Health > Assignments > Health Indicator Assignments** and create the indicator assignment.

   The assignment condition should match the Oracle CIs on which you want to define the indicators (and not other Oracle CIs that do not belong to this integration). In the indicator assignment, select the **CPU Usage by SQL** indicator.

The following page opens when you edit the indicator in this assignment:



Edit Health Indicator For Assignment: My Oracle HI assignment

Define a Health Indicator Configuration

Health Indicator

* Health Indicator: CPU Usage by SQL
* Business Rule: SiteScope Worst Status Rule
* Priority: 0

Business Rule Parameters

generateEvents: False
No data timeout: 900 seconds
Tooltip show all measurements: false (String)

* You can drag properties from CI Type Properties list or press Ctrl + i while editing a field to set the value to the selected property.

Selector

Field Name	Operator	Type	Value
AND			
eti_id	=	Binary	<<Health Indicator Type ID>>
ci_id	=	Binary	<<CI ID>>
sampleType	=	String	ss_t

Expression Summary:

```
          [eti_id = <<Health Indicator Type ID>>]
AND
          [ci_id = <<CI ID>>]
AND
          [sampleType = ss_t]
```

* You can drag properties from CI Type Properties list or press Ctrl + i while editing a field's value to set the value to the selected property.

CI Type Properties

General Properties

Binary
Application Password
Calculated ID
Boolean
Allow CI Update
Change Is New
Enable Aging
Has Config Files
Is Candidate For Deletion
Operation Is New
Store KPI History For Over Time Reports
Test Is New
Track Configuration Changes
Date
Actual Delete Time
Candidate For Deletion Time
Create Time
Last Access Time
LastModifiedTime
StartupTime
Integer
Actual Deletion Period
Application Listening Port Number
Application Timeout
Deletion Candidate Period
List of Strings
Context Menu
Monitored By
TenantsUses
Long
Acknowledgement update timestamp
String
Application Category
Application Installed Path
Application Instance Name
Application IP
Application Username

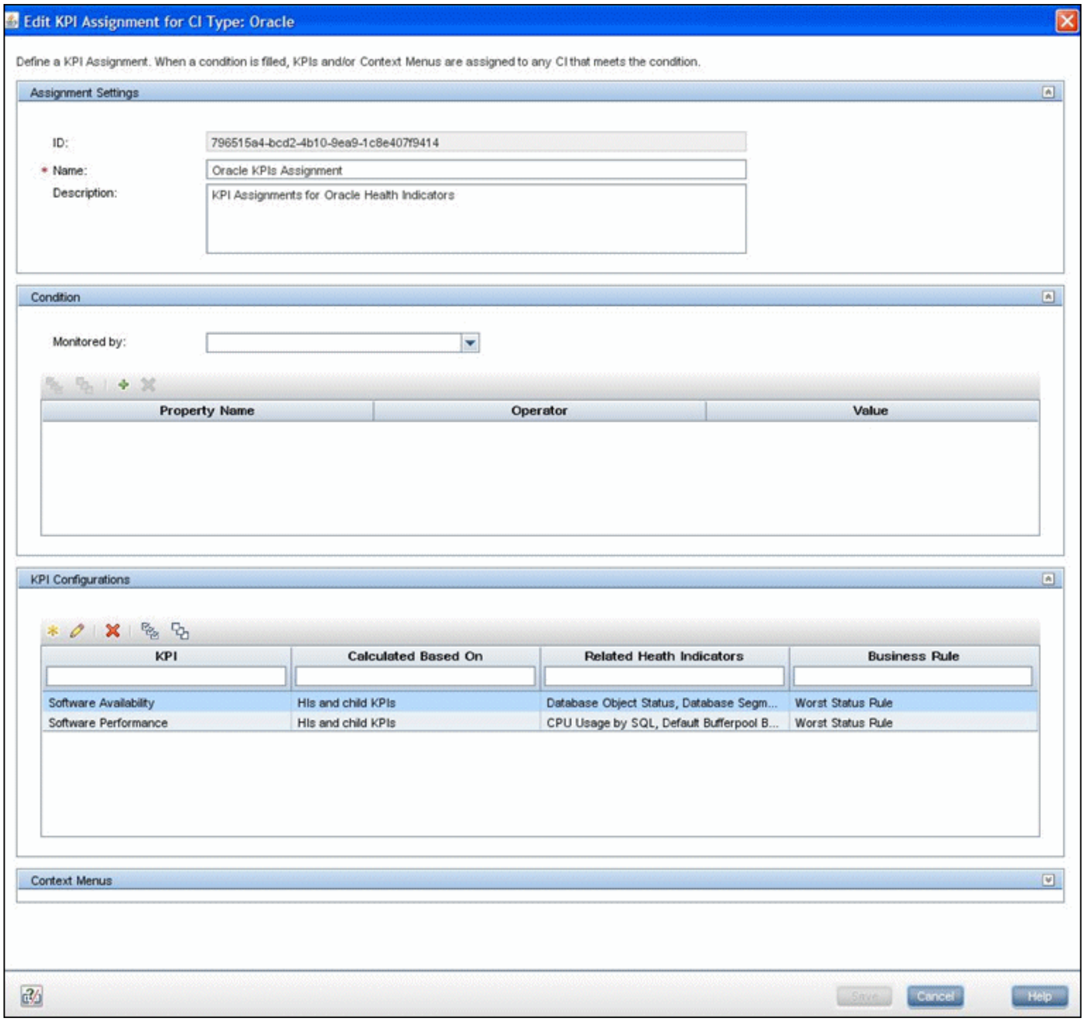
The `CPU Usage by SQL` indicator is calculated using the SiteScope Worst Status Rule. The selector defines that data of type `ss_t` (the metrics data type) with the same `ci_id` and `eti_id` as the current CI and ETI are captured by this indicator on this Oracle CI.

4. Define a KPI Assignment

Since you used an out-of-the-box indicator, you do not need to create a KPI assignment as there is an existing Service Health KPI assignment on Oracle CIs.

In BSM, select **Admin > Service Health > Assignments > KPI Assignments**, and in the CI Type tree select **Oracle** and choose **Oracle KPI Assignment**.

In the **Monitored by** property, enter the name of the BSM Connector policy, `My Oracle Monitoring integration` in this example. This value is used to distinguish Oracle CIs reported by this integration from other Oracle CIs that are being reported. This assigns the `CPU Usage by SQL` indicator to Oracle CIs that are reported by this integration only.



Edit KPI Assignment for CI Type: Oracle

Define a KPI Assignment. When a condition is filled, KPIs and/or Context Menus are assigned to any CI that meets the condition.

Assignment Settings

ID: 796515a4-bcd2-4b10-9ea9-1c8e407f9414

* Name: Oracle KPIs Assignment

Description: KPI Assignments for Oracle Health Indicators

Condition

Monitored by:

Property Name	Operator	Value

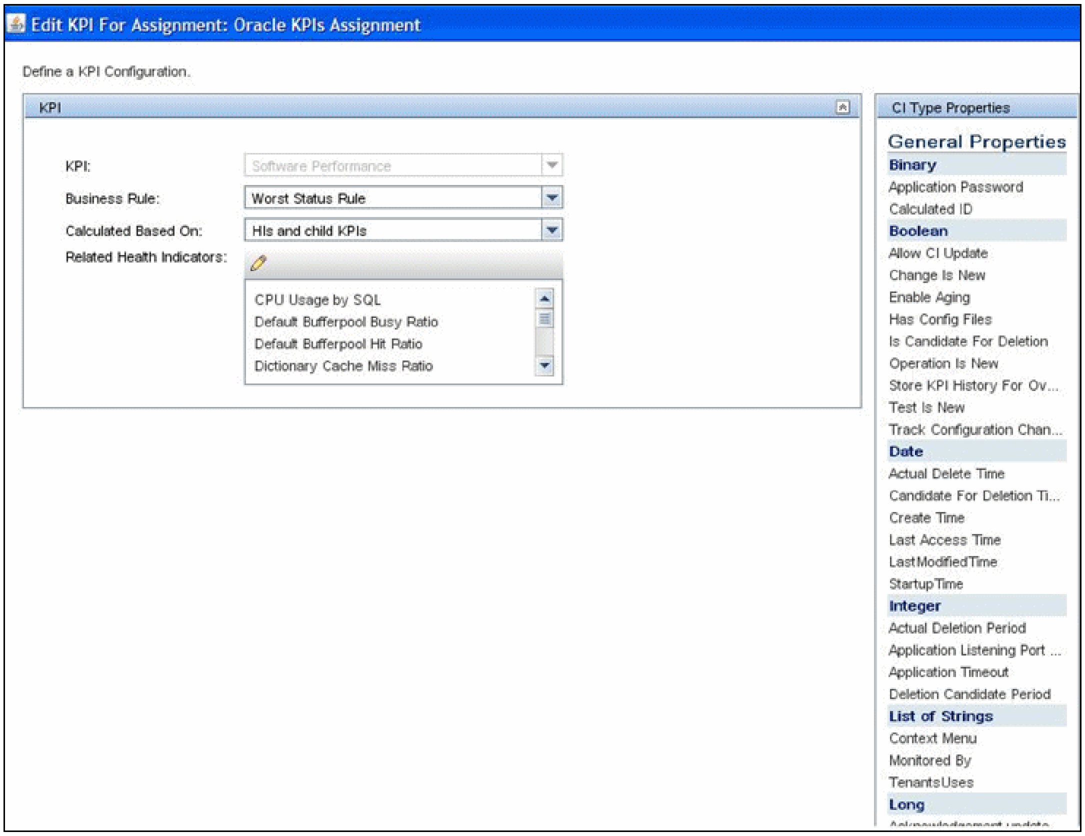
KPI Configurations

KPI	Calculated Based On	Related Heath Indicators	Business Rule
Software Availability	HIs and child KPIs	Database Object Status, Database Segm...	Worst Status Rule
Software Performance	HIs and child KPIs	CPU Usage by SQL, Default Bufferpool B...	Worst Status Rule

Context Menus

Select the **Software Performance** KPI:

You can see that one of the indicators related to this KPI is the `CPU Usage by SQL` indicator which you used.



Edit KPI For Assignment: Oracle KPIs Assignment

Define a KPI Configuration.

KPI

KPI: Software Performance

Business Rule: Worst Status Rule

Calculated Based On: HIs and child KPIs

Related Health Indicators:

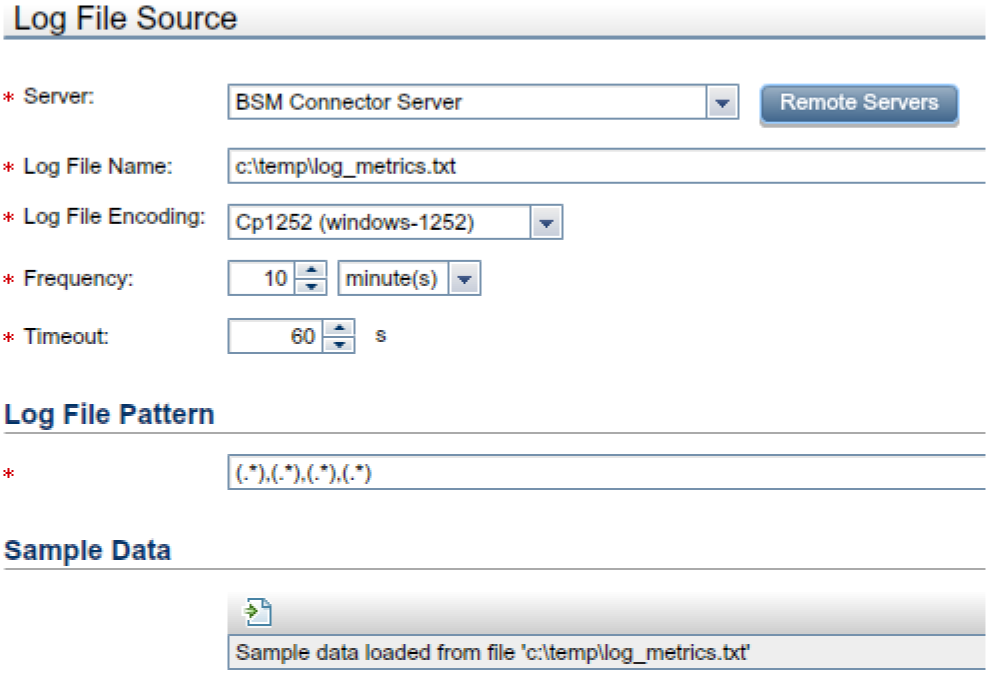
CPU Usage by SQL
Default Bufferpool Busy Ratio
Default Bufferpool Hit Ratio
Dictionary Cache Miss Ratio

5. **Create a log file policy for metrics integration**

   Since the application writes to log files, you need to create a metrics integration. The metrics will be assigned to Oracle CIs that already exist in RTSM; therefore the CIs do not need to be reported. The data is assigned to HIs on the CIs.

   For details about log file data, "Integrating Log File Data into BSM" on page 147.

a. In the **Source** page, specify the name of the log file and the log file pattern.



Log File Source

* Server: BSM Connector Server

* Log File Name: c:\temp\log_metrics.txt

* Log File Encoding: Cp1252 (windows-1252)

* Frequency: 10 minute(s)

* Timeout: 60 s

Log File Pattern

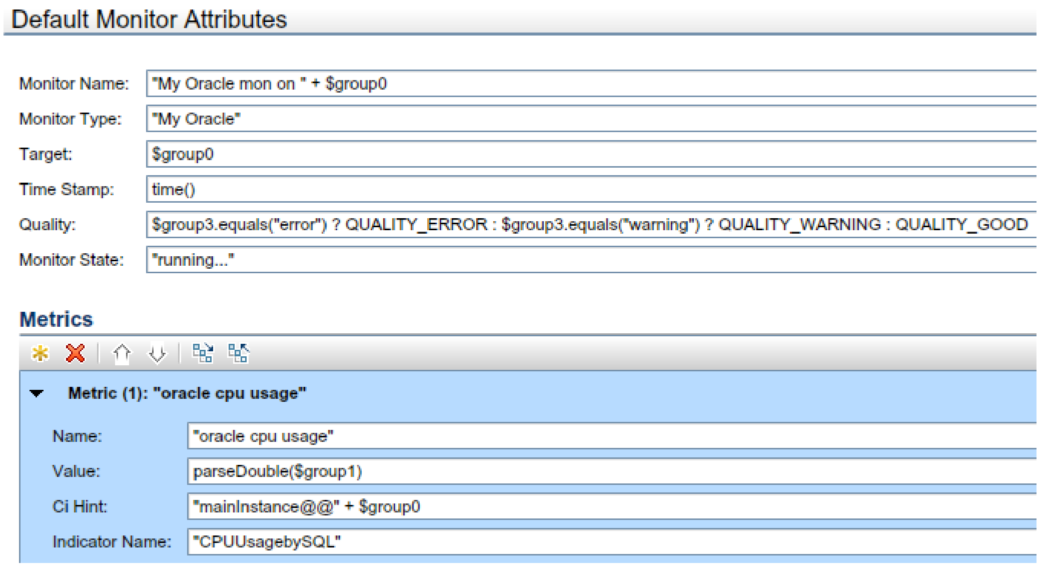
* (.*),(.*),(.*),(.*)

Sample Data

Sample data loaded from file 'c:\temp\log_metrics.txt'

b. In the **Defaults** page, assign default values to the metric attributes:

- The monitor name is `My Oracle mon on $group0` where `$group0` is also the target computer on which the Oracle database is running.
- You define a new monitor type: `My Oracle`.
- The quality that is sent is conditional and depends on what is written in the log file.
- The metric name is `oracle cpu usage` and its value is taken from the log file.
- The CI hint is in the format `<<oracle sid>>@@<<computer name>>`. The CI hint is used by the CI Resolver in BSM to find the CI to which this metric should be attached.
- The ETI to which the metric is mapped is `CPUUsagebySQL`.



Default Monitor Attributes

Monitor Name:	"My Oracle mon on " + $group0
Monitor Type:	"My Oracle"
Target:	$group0
Time Stamp:	time()
Quality:	$group3.equals("error") ? QUALITY_ERROR : $group3.equals("warning") ? QUALITY_WARNING : QUALITY_GOOD
Monitor State:	"running..."

Metrics

Metric (1): "oracle cpu usage"

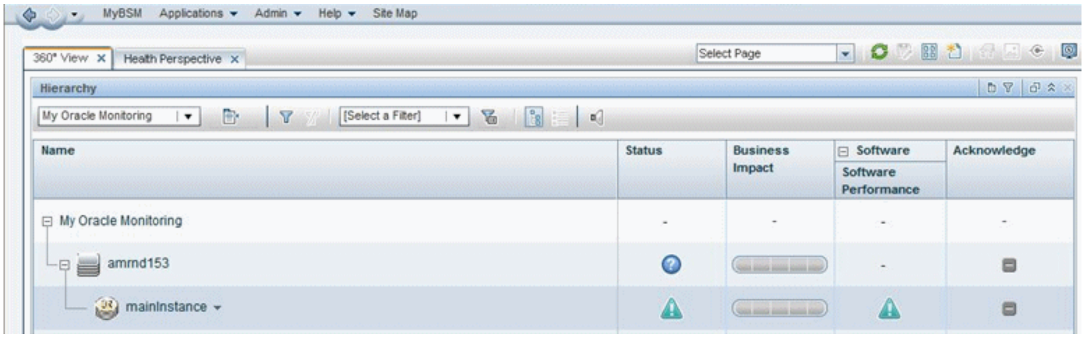
Name:	"oracle cpu usage"
Value:	parseDouble($group1)
Ci Hint:	"mainInstance@@" + $group0
Indicator Name:	"CPUUsagebySQL"

c. In the **Topology** page, select **No Topoloy**.

d. Save and activate the policy.

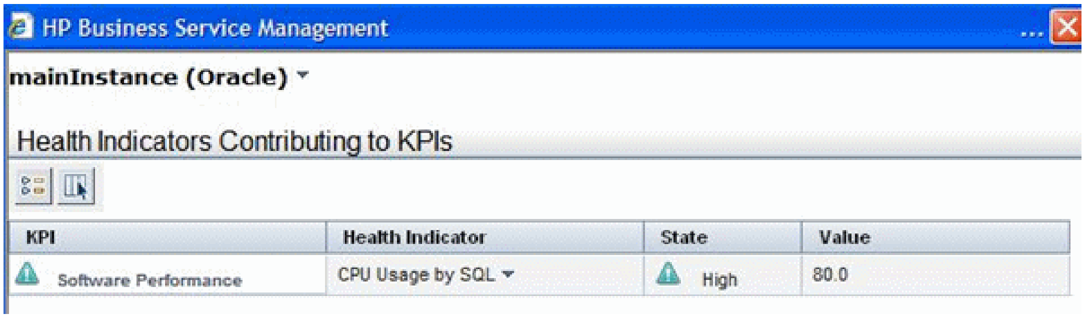
6. View the integration results

In BSM, select **Applications > Service Health** and manually create a view for the integration. For example, here we created a view named `My Oracle Monitoring` (the integration result is also displayed in System Software Monitoring view):



Name	Status	Business Impact	Software Software Performance	Acknowledge
My Oracle Monitoring	-	-	-	-
ammd153	?		-	
mainInstance	⚠		⚠	

These are the results of the indicator status on the monitored machine:



HP Business Service Management

mainInstance (Oracle)

Health Indicators Contributing to KPIs

KPI	Health Indicator	State	Value
⚠ Software Performance	CPU Usage by SQL	⚠ High	80.0

The state and value are the same as you assigned in the policy.

How to View Metrics in BSM User-Defined Reports

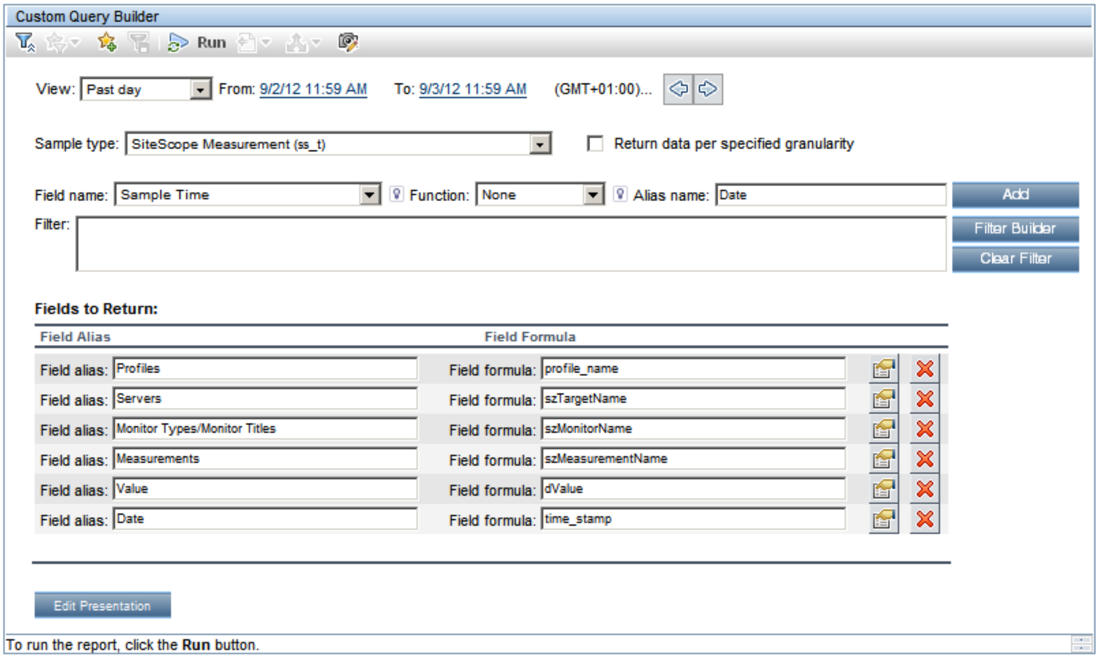
If a System Availability Management license is installed in your BSM environment, you can view metrics from BSM Connectors, filtered by monitored servers, monitor types, and measurements in SiteScope Cross-Performance reports, which can be accessed through **Applications > System Availability Management > SiteScope Over Time Reports > Cross-Performance**.

However, you can also generate reports for BSM Connector metrics in BSM, even if a System Availability Management license is not installed. (BSM Connector does not require a System Availability Management license.)

To configure a user-defined report:

1. In BSM, access **Applications > User Reports > Custom Query Builder**.
2. For **Sample type**, select **SiteScope Measurement (ss_t)**.
3. Add the following fields and optionally change the field aliases to more user-friendly names:

Field name	Optionally change field alias to
Profile Name	Profiles
Target Name	Servers
Monitor Name	Monitor Types/Monitor Titles
Measurement Name	Measurements
Measurement Value	Value
Sample Time	Date



Custom Query Builder

View: Past day From: 9/2/12 11:59 AM To: 9/3/12 11:59 AM (GMT+01:00)...

Sample type: SiteScope Measurement (ss_t) □ Return data per specified granularity

Field name: Sample Time Function: None Alias name: Date

Filter:

Fields to Return:

Field Alias	Field Formula
Field alias: Profiles	Field formula: profile_name
Field alias: Servers	Field formula: szTargetName
Field alias: Monitor Types/Monitor Titles	Field formula: szMonitorName
Field alias: Measurements	Field formula: szMeasurementName
Field alias: Value	Field formula: dValue
Field alias: Date	Field formula: time_stamp

To run the report, click the **Run** button.

4. Click **Edit Presentation** and choose one of the following layouts in the wizard:

   - **Table**

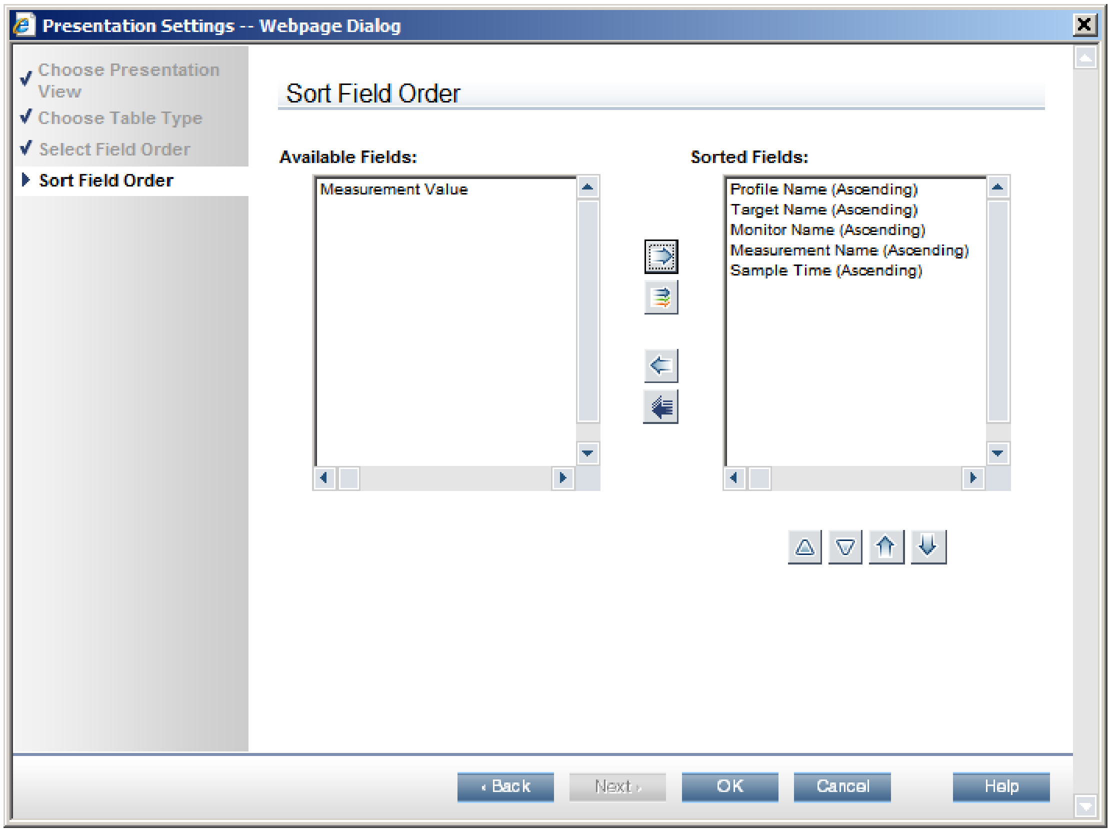
     Make sure **Table** is selected and click **Next**.

     Make sure **Regular** is selected and click **Next**.

     Change the **Field Order** by moving Sample Time up by one in the list, and then click **Next**.

Specify the **Sort Order**. Move all fields except for Measurement Value from the **Available Fields** list to the **Sorted Fields** list with ascending order. The Sorted Fields list should list the following fields in this order:

- Profile Name (Ascending)
- Target Name (Ascending)
- Monitor Name (Ascending)
- Measurement Name (Ascending)
- Sample Time (Ascending)



Sort Field Order

Available Fields:

Measurement Value

Sorted Fields:

Profile Name (Ascending)
Target Name (Ascending)
Monitor Name (Ascending)
Measurement Name (Ascending)
Sample Time (Ascending)

Click **OK** to close the Presentation Settings Wizard.

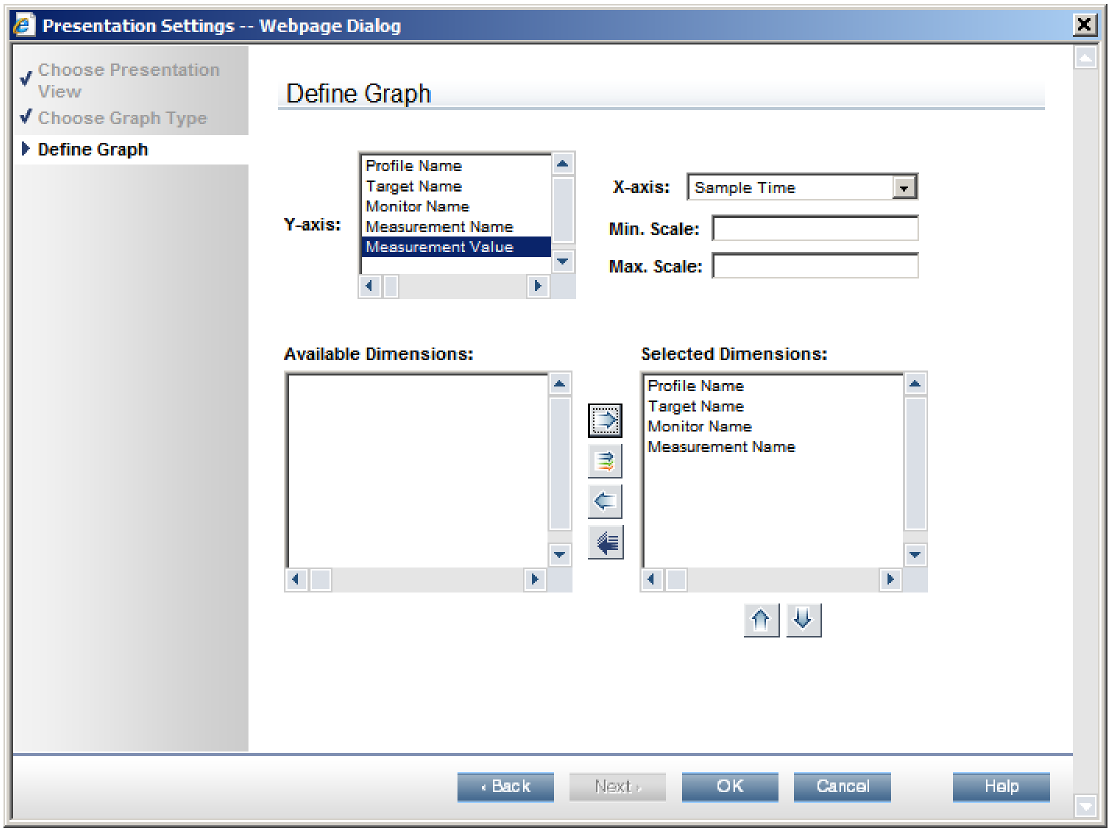
- ### Graph

  Select **Graph** and click **Next**.

  Select **Line** and click **Next**.

  In the Define Graph page, configure the following settings:

  - Y-axis: Measurement Value
  - X-axis: Sample Time
  - Available Dimensions: none
  - Selected Dimensions: Profile Name, Target Name, Monitor Name, Measurement Name


Presentation Settings -- Webpage Dialog
✓ Choose Presentation View
✓ Choose Graph Type
Define Graph
‹ Back
Next ›
OK
Cancel
Help

Define Graph

Y-axis: Profile Name, Target Name, Monitor Name, Measurement Name, Measurement Value

X-axis: Sample Time

Min. Scale:

Max. Scale:

Available Dimensions:

Selected Dimensions: Profile Name, Target Name, Monitor Name, Measurement Name

Click **OK** to close the Presentation Settings Wizard.

5. In the Custom Query Builder, click **Save Favorite** to save your report for later use.
6. Click **Favorites**, select the saved report, and run it.
7. *Optional:* Click **Filter Builder** to create a filter on specific fields for the report. For example, apply a filter on the Target Name field to filter on selected monitored servers.

Tip: You can select metrics from more than one BSM Connector instance (profile) for the same report.

Fore more information on BSM user-defined reports, see the chapter "Build a Custom Query Using Custom Query Builder" in the the BSM User Guide.

Metrics Troubleshooting

This section provides help in troubleshooting problems relating to metrics integration.

This section includes:

- "Metrics integration policy error logs" below
- "Metrics error logs" below
- "Additional troubleshooting information" on the next page

Metrics integration policy error logs

BSM Connector converts metrics integration policies to an internal file format during policy activation. If the policy converts successfully, the policy can be activated and starts collecting metrics data.

Any errors encountered during the conversion are logged in the file:

```
<BSM Connector root directory>/logs/error.log
```

To troubleshoot activation problems that are caused by conversion errors, open the log file and search for lines starting with `ERROR`.

The error message indicates in which line the error was encountered. In the section immediately above the error message, look for the line `[$DEFAULT_PARAMETERS$]` and start the line count from there.

Metrics error logs

You can configure BSM Connector to write the metrics it sends to BSM to the following log file:

```
<BSM Connector root directory>/logs/bac_integration/samples.log
```

1. Open the following file in a text editor:

   ```
   <BSM Connector root directory>/conf/core/Tools/log4j/PlainJava/bac_integration.properties
   ```

2. Add the following lines:

   ```
   #####################################################################-
   ################
   # Sample printer
   #####################################################################-
   ################
   log4j.category.SamplePrinter=DEBUG, samples.appender
   log4j.additivity.SamplePrinter=false
   log4j.-
   appender.samples.appender=org.apache.log4j.RollingFileAppender
   log4j.appender.samples.appender.File=../${log.file.path}
   /samples.log
   log4j.appender.samples.appender.MaxFileSize=1000KB
   log4j.appender.samples.appender.MaxBackupIndex=5
   ```

```
log4j.-
appender.samples.appender.layout=org.apache.log4j.PatternLayout
log4j.appender.samples.appender.layout.ConversionPattern=%d [%t]
(%F:%L) %-5p - %m%n
log4j.appender.samples.appender.encoding=${general.encoding}
```

3. Save the file. It may take a few seconds for the changes to take effect.

   The results are logged to the `samples.log` file

Additional troubleshooting information

- For troubleshooting database policies, see "Troubleshooting Database Policies" on page 144.
- For troubleshooting log file policies, see "Troubleshooting Log File Policies" on page 231.
- For troubleshooting Web service listener policies, see "Troubleshooting Web Service Listener Policies" on page 304.

Synchronizing Events

BSM Connector enables you to access event sources, and, if certain conditions apply, to forward the detected events as HP Business Service Management (BSM) events to BSM. After BSM receives an event, you can keep it up to date on the event source by configuring BSM and BSM Connector to synchronize event changes back to the third-party system that generated the original event. For example, if a BSM Operations Management 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 Operations Management online help.

To configure event synchronization:

1. Configure the BSM Connector server as an integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For full details about how to configure a BSM Connector integration server, see the BSM online help.

   > **Note:** Setting up a BSM Connector integration server automatically creates a BSM Connector connected server in Operations Management.

2. In the BSM Connector **Event Drilldown Settings**, select the **Enable event drilldown and synchronization** checkbox.

   This checkbox also enables event drilldown, if supported by the connected server. See also "Configuring Drilldown" on page 92.

   > **Note:**
   >
   > If you clear the **Enable event drilldown and synchronization** checkbox, BSM deletes information about event changes and is therefore not able to update BSM Connector when the checkbox is selected again. When the checkbox is selected but the BSM Connector integration server is temporarily unavailable, BSM buffers information about event changes, and tries to resend them when the server is available again.

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 BSM and forwards them to the third-party system .

   For more information, see "How to Write Perl Scripts for Event Synchronization" on page 91.

How to Configure Policies for Event Synchronization

The event that BSM Connector forwards to BSM must include the original ID of the event in the third-party system. Otherwise BSM 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 **OK** 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 the event changes from the BSM 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 your 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`.

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

The `OMBSOutput.txt` file is located at:

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

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

The `OMBackSync.pl` file is located at:

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

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

> **Note:** 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.

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
   ```

Configuring Drilldown

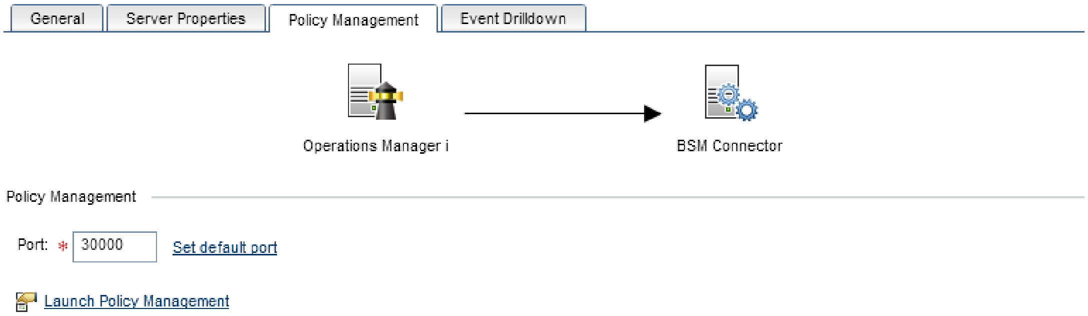
BSM enables you to start the user interface of BSM Connector or of the third-party system directly from within BSM. The following integrations are available:

BSM Connector UI integrations

- **BSM Connector integrations page.** Navigate to **Admin > Integrations > BSM Connector Integrations**. In the Summary view, locate the BSM Connector you want to launch. Click the name in the Name column.
- **Connected Servers manager.** Navigate to **Admin > Operations Management > Setup > Connected Servers**. In the Connected Servers details pane, locate the BSM Connector you want to launch and click the Launch Policy Management icon.

  Alternatively, edit the BSM Connector you want to launch, navigate to the Policy Management tab, and click the Launch Policy Management icon.

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



Port: * 30000

- **Event context.** In the Operations Management 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. 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.

For details, see "How to Configure Launch of the BSM Connector User Interface" on the next page.

Third-party system UI integration

In the Operations Management 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 94.

How to Configure Launch of the BSM Connector User Interface

This task shows you how to configure launch of the BSM Connector user interface from within BSM.

To configure launch of a BSM Connector server from within BSM:

1. Configure the BSM Connector server as an integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

2. In **Event Drilldown Settings**, select the **Enable event drilldown and synchronization** checkbox.

   This checkbox also enables event synchronization, if supported by the connected server. See also "Synchronizing Events" on page 89.

For full details about how to configure a BSM Connector integration server, see the BSM online help.

> **Note:** Setting up a BSM Connector integration server automatically creates a BSM Connector connected server in Operations Management.

How to Configure Drilldown into Third-Party Systems

BSM enables operators to start the user interface of third-party systems in the context of an event in the Operations Management Event Browser. To launch the third-party system's user interface, right-click the event in the Operations Management 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 BSM.
- Specify source information in the policies that forward the source events to BSM.

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 integration server 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 an integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

2. In the **Event Drilldown Settings**, complete the following information:

   External Event Processing Server. Enter the fully qualified DNS name of the computer that hosts the third-party system (for example, `SCOM.mgmt2.example.com`).

   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. BSM 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 90.

   Use SSL. Select to change the communication protocol used by the third-party system.

   Enable event drilldown and synchronization. Select to enable drilldown and event synchronization, if supported by the connected server.

For full details about how to configure a BSM Connector integration server, see the BSM online help.

> **Note:** Setting up a BSM Connector integration server automatically creates a BSM Connector connected server in Operations Management.

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 BSM 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 **OK** to save the policy and close the editor.
5. Activate the policy for your changes to take effect.

Support for Internet Protocol Version 6

BSM Connector database, log file, and Web service listener policies can communicate with remote servers that have IPv6 addresses.

The level of support for IPv6 depends on the operating system on which BSM Connector is installed:

- **Windows Server 2008.** Windows Server 2008 has full-featured support for IPv6, which is installed and enabled by default. As a result, IPv6 is supported by BSM Connector database, log file, and Web service listener policies when BSM Connector is installed on Windows Server 2008 and later.
- **Windows Server 2003.** Support for IPv6 on Windows Server 2003 is limited, as many core services and networking components do not support it.
- **UNIX.** IPv6 is also fully supported when BSM Connector is installed on Linux operating systems that provide full support for IPv6.

Supported Protocols

The following protocols are supported when IPv6 is used in BSM Connector installed on Windows and UNIX platforms:

Target	BSM Connector Installed on Windows Platform	BSM Connector Installed on UNIX Platform
Windows	NetBios	SSH
UNIX	Not supported	SSH

Note:

- BSM Connector installed on Windows platforms can access Windows machines only.
- NetBIOS is supported when BSM Connector is installed on Windows platforms only .
- SSH is supported only when BSM Connector is installed on UNIX machines.

How to Configure BSM Connector to Prefer IPv6 Addresses

By default, BSM Connector connects to remote servers using IPv4 addresses. If you want your environment to resolve host names to IPv6, you can configure BSM Connector to prefer IPv6 addresses:

1. Set the `_preferIPV6Address` property value to `=true` in the `<BSM Connector root directory>\BSMConnector\groups\master.config` file.

   Restart BSM Connector for your changes to this setting to take effect:

   Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

2. Make sure the host name resolves to both an IPv4 and an IPv6 address. If the host name resolves only to an IPv4 address, then the IPv4 address is used.

3. Specify a host name for the remote server. If an IP address is specified, the prefer IPv6 setting has no effect on the host since the IP address determines the IP version that is used.

   For details on specifying a hostname for remote servers for log file policies, see "Configuring Remote Servers " on page 189.

Working in a Mixed IPv4 and IPv6 Environment

When working in a mixed environment where both IPv4 and IPv6 are used, the DNS server might return both an IPv4 and an IPv6 address for a host name. To instruct BSM Connector which IP address to use for each resolved host name, you can:

- Configure BSM Connector to prefer IPv6 addresses, and perform one of the following (for the hosts that you want to use the IPv4 protocol):
  - Enter the IP address instead of the host name for the specified remote server.
  - Configure the DNS server so that the host name resolves to the IP address that you want to use for the remote server. You can do this by removing the IPv6 address from the DNS server for the specified host.

- Configure BSM Connector to not prefer IPv6 addresses (this is the default), and perform the following (for the hosts that you want to use the IPv6 protocol):
  - Enter the IP address instead of the host name for the specified remote servers.
  - Configure the DNS server so that the host name resolves to the IP address that you want to use for the specified remote servers. You can do this by removing the IPv4 address from the DNS server for the specified hosts.

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 100

Tool location

The BSM Connector command line tools are located on the BSM Connector server in:

Windows: `%OvDataDir%\installation\HPOprIA\`

Linux: `/var/opt/OV/installation/HPOprIA/`

Log files

All BSM Connector command line tools write information to the following log file on the BSM Connector server:

```
<BSM Connector root dir>/logs/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\HPOprIA\bsmc-conf.bat`

Linux: `/var/opt/OV/installation/HPOprIA/bsmc-conf.sh`

bsmc-conf accepts the following options:

```
bsmc-conf.[bat|sh] -srv <hostname> [ -cert_srv <hostname> ] [ -f ] [ -
flexmgmt [ -omc_srv <hostname> ] ] | -help | -version
```

This section includes:

- "Command options" on the next page
- "To reconnect BSM Connector to another BSM server" on page 99
- "To install and configure BSM Connector on an HPOM managed node" on page 99
- "Log files" on page 100
- "Using ovconfchg" on page 100

Command options

`-s`,`-srv` *`<hostname>`*

Specifies the BSM server.

`<hostname>` is the fully qualified domain name (FQDN) of the BSM server. In a distributed BSM deployment, choose the gateway server. If you have more than one Gateway Server, select just one, or select the load balancer.

`-cs`,`-cert_srv` *`<hostname>`*

Specifies the BSM certificate server.

`<hostname>` is the FQDN of the BSM certificate server. If your BSM deployment includes a dedicated certificate server, choose the FQDN of the certificate server. Otherwise type the FQDN of the gateway server. The gateway server forwards certificate requests to the data processing server, which by default also acts as certificate server.

If you omit this option, **bsmc-conf** uses the BSM server specified using the `-srv` option.

`-f`,`-force>`

Force mode (for example to reconfigure BSM Connector to connect to a different BSM server).

`-fm`,`-flexmgmt`

Creates the flexible management policy **BSM Connector FlexMgmt Policy**.The policy configures the agent to send the following data to the BSM server:

- Events generated by BSM Connector policies. These events automatically have the **Type** attribute set to **BSMC_Message**.

  The HPOM management server specified using the `-omc_svr` option receives all events that do not have the **Type** attribute set to **BSMC_Message**.

- Metrics data.

- Topology data (both legacy and DDM-discovered data).

`-omc`,`-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
> ```

`-h`,`-help`

Shows tool usage and description.

`-v`,`-version`

Displays the BSM Connector version number.

When you run **bsmc-conf**, the program does the following:

- Replaces all existing values with new or default values.
- Creates a backup of the flexible management policy:

  Windows: `%OvDataDir%\conf\HPOprIA\<`*`policy_ID`*`>_data.bak`

  `%OvDataDir%\conf\HPOprIA\<`*`policy_ID`*`>_header.xml.bak`

  Linux: `/var/opt/OV/conf/HPOprIA/<`*`policy_ID`*`>_data.bak`

  `/var/opt/OV/conf/HPOprIA/<`*`policy_ID`*`>_header.xml.bak`

To reconnect BSM Connector to another BSM server

1. On the BSM Connector server, type the following command to reconnect the system to another BSM server:

   ```
   bsmc-conf -srv <new BSM server> -force
   ```

2. Add the BSM Connector server to the BSM Connector Integrations page in BSM (**Admin > Integrations > BSM Connector Integrations**).

   Do *not* select the **Configure policy management** checkbox.

3. In Operations Management, grant the certificate generated by **bsmc-conf**.
4. Delete the BSM Connector integration from the old BSM server.
5. **Results.** BSM Connector sends all collected data to the new BSM 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 BSM server> -force -flexmgmt -omc_srv <HPOM
   management server>
   ```

2. Add the BSM Connector server to the BSM Connector Integrations page in BSM (**Admin > Integrations > BSM Connector Integrations**).

   Do *not* select the **Configure policy management** checkbox.

3. In Operations Management, grant the certificate generated by **bsmc-conf**.
4. **Results.** The new BSM 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 BSM 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**.

   - Metrics data.
   - Topology data (both legacy and DDM-discovered data).

Log files

bsmc-conf writes information to the following log file:

```
<BSM Connector root dir>/logs/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 user 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.

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 BSM Connector Integrations page in BSM (**Admin > Integrations > BSM Connector Integrations**). 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\HPOprIA\user.bat`

Linux: `/var/opt/OV/installation/HPOprIA/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 (`_`), and periods (`.`).

`-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\HPOprIA\users.properties`

Linux: `/var/opt/OV/conf/HPOprIA/users.properties`

To add local users

1. *Windows.* Run the local user configuration tool:

   a. Open a command prompt and type:

   ```
   cd %OvDataDir%\installation\HPOprIA
   ```

   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/HPOprIA/user.sh -add <username> <password>
   ```

2. *Optional.* Review the log file at:

   `<BSM Connector root dir>/logs/opr-pm-cli.log`

   The program appends log information to the file when you run the program again.

Troubleshooting and Limitations

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

This section includes:

- "Tomcat logging" below
- "Debug trace logging for events" on the next page

Tomcat logging

Tomcat log output is written to the `<BSM Connector root dir>\logs\tomcat.log` file.

Settings for the log file can be configured from the `<BSM Connector root directory>\Tomcat\common\classes\log4j.properties` file.

Debug trace logging for events

It is possible to enable debug trace logging for an event by setting the custom attribute `__TRACE__`. It may have any value. This creates flow trace logs at the `INFO` level for this event.

This can be set on the HPOM server or agent sending the event, or can later be added to the event. Whenever this custom attribute is set on an event, trace output for this event appears in the trace logs:

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

Default event flow trace logging level is set to `INFO`. 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

Managing Policies

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 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

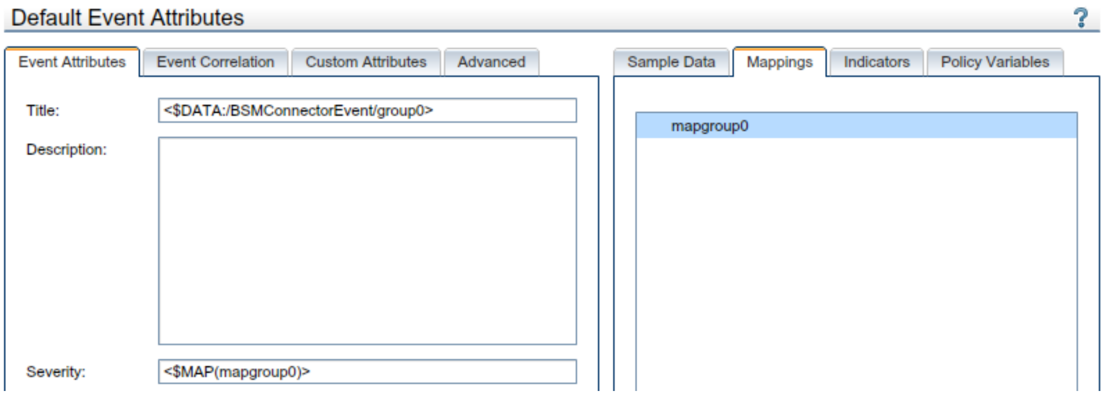
This section includes:

- "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 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.



Title: <$DATA:/BSMConnectorEvent/group0>

Description:

Severity: <$MAP(mapgroup0)>

mapgroup0

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 BSM 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 BSM. Otherwise the events display as unmapped events in the Operations Management Event Browser.

Tasks

Related tasks

- "How to Edit Policies" on the next page
- "How to Copy Policies" on page 107
- "How to Delete Policies" on page 107
- "How to Activate and Deactivate Policies" on page 108
- "How to Export Policies" on page 108
- "How to Import Policies" on page 110
- "How to Configure Policy Management Options" on page 111

UI Descriptions

Policy Management

UI Element	Description
🔄	**Refresh.** Reloads the policy list.

UI Element	Description
✱	**Create.** Opens a policy editor to create a new policy.
✎	**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.
🖥	**Remote Server Management.** Opens the Remote Servers dialog box. For more information about remote servers, see "Remote Servers Overview" on page 185.
⧉	**BSM Integration Preferences.** Opens the BSM Integration Preferences dialog box. For details, see "Configuring Integration Preferences for Inaccessible Profiles" on page 33.
🔧	**Options.** Opens the Options dialog box
Policy Type	The policy type indicates the data source of the policy (for example, 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 **OK** 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 **OK** 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	Database connection URL	If necessary, enter placeholders for the database server name and port, for example enter: `jdbc:mercury:sqlserver://<dbserver>:<dbport>;DatabaseName=EVENT;AuthenticationMethod=type2`
	Database user name	Enter a placeholder for the database user name, for example: `<db user name>`.
	Database password	Remove the password from the policy.
	Server name	Enter a placeholder for the database server. for example: `<db server>`.
Discovery policies (legacy)	User name	Enter a placeholder for the account under which the command runs, for example: `<username>`.
	Password	Remove the password from the policy.
Log file policies	Remote server name	Change the server name to the BSM Connector server.
	Log file name	If necessary, change the path or file name.
Scheduled task policies	Command	If necessary, change the path or file name.
	User name in commands, Perl, or VBScript	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.
Topology policies	Scripts	If necessary, remove sensitive information, or information that is specific to the BSM Connector source server.
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 313

To export policies:

1. Prepare the policies for distribution:

   a. Create copies of the policies. For details, see "How to Copy Policies" on page 107.

   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 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:

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** to start the import process.

   If the same policies 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. For example, configure a remote server in log file policies or replace placeholder information with real data in database policies.
5. *Optional:* Activate the policies. See also "Activation error in the BSM Connector UI" on the next page.

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 and close the dialog box.

UI description

UI Element	Description
Policy Type	The policy type indicates the data source of the policy (for example, 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.

Troubleshooting and Limitations

This section provides help in troubleshooting problems relating to policies and policy management in general.

This section includes:

- "Indicator definitions could not be loaded" below
- "Activation error in the BSM Connector UI" below
- "Number of activated policies" on the next page
- "Number of events per second" on the next page

Indicator definitions could not be loaded

The following error message displays when BSM Connector cannot load indicator definitions from the connected BSM server:

```
Indicator definitions could not be loaded, service returned:
com.hp.opr.policymanagement.exception.PolicyManagementException:
BSM integration is not initialized, possibly BSM Connector is not yet
integrated with BSM
```

The following problems can cause this error message:

- The BSM Connector system is not set up as a BSM Connector integration server in BSM. To correct the problem, set up the BSM Connector system as a BSM Connector integration server in BSM:

  Admin > Integrations > BSM Connector Integrations

  For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

- The BSM Connector integration was deleted from one BSM server and added to another. To correct the problem, restart BSM Connector:

  Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.

  Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

Activation error in the BSM Connector UI

When you activate database policies with placeholder information in the USERNAME or URL fields, the activation will fail and the activation state changes to ❌`activation error` in the UI. The following error message appears in the System.txt log file:BSM Connector

Windows: `%ovdatadir%\log\System.txt`

Linux: `/var/opt/OV/log/System.txt`

```
An error occurred during the initialization of an internal component
for the policy <policy ID> (of type <policy type>).
The component (MonitorDeployment) returned the following error:
Error Code: 55528.  Error Description: Failed to deploy template.
(OpC30-3913)
```

To resolve the problem, complete the following steps:

1. Edit the policies in the UI and replace the placeholder information with real data.
2. Reactivate the policies.

Number of activated policies

You can activate and run up to 60 policies on the BSM Connector server in parallel. If you activate and run more than 60 policies, the performance of BSM Connector may deteriorate.

Number of events per second

BSM Connector event integration policies can send up to 110 events per second to BSM, if BSM Connector is installed on a system that meets the hardware requirements described in the BSM Connector Installation and Upgrade Guide.

If BSM Connector database, log file, and Web service listener policies send fewer than the expected 110 events per second, try the following workarounds:

- Increase the maximum size of the event file. For details, see "Troubleshooting Database Policies" on page 144, "Troubleshooting Log File Policies" on page 231, and "Troubleshooting Web Service Listener Policies" on page 304.
- If the Java Virtual Machine (JVM) crashes, increase the JVM heap size:

  Windows:

  a. Stop the BSM Connector service in the **Administrative Tools > Services**.

  b. Edit the BSM Connector's service key in the registry. Use regedit to open the Registry Editor.

  c. In the Registry Editor, navigate to `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/<`*`BSM Connector service name`*`>/serviceParam`. The default service name is `HP BSM Connector`.

  d. Modify the serviceParam default and change `-Xmx512m` to one of the following:

     2GB: `-Xmx2048m`

     4GB: `-Xmx4096m`

  e. Start the BSM Connector service in the **Administrative Tools > Services**.

  Linux:

  a. Stop BSM Connector, type:

     ```
     /opt/HP/BSMConnector/stop
     ```

  b. Navigate to `<`*`BSM Connector root directory`*`>/bin` and edit the `start-service` script.

  c. In the script, change `-Xmx512m` to one of the following:

2GB: `-Xmx2048m`

4GB: `-Xmx4096m`

d. Start the BSM Connector service, type:

```
/opt/HP/BSMConnector/start
```

Chapter 4

Integrating Database Data into BSM

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 BSM using database policies:

- Events from monitoring applications event tables or views.
- Data from monitoring applications metrics tables.
- Topology from a third-party topology database.

What Data Is Forwarded

Database policies use a user-defined query and enumerating field name, field type, and initial value. While the query provided by the user is used to define a search criterion on the database, the enumerating field is used so that data records are processed only once. Using an initial value enables you to specify an initial threshold value for the data that should be processed.

For example, if **Enumerating Field Type** uses `DATE and` **Initial enumerating value** uses `2003-20-03 12:00:00,` only data records that happened after the specified date are processed in the first run of the policy. In subsequent runs, the highest value for the DATE field found is used to verify that only new data is processed. For a details on how data from the enumerating field is processed, see "Understanding How Data in the Enumerating Field is Processed" on page 120.

You use defaults and rules to control the data that is sent from BSM Connector to BSM.

Data can also be mapped to a topology to forward data to the correct CI hierarchy in BSM. You can configure topology settings for the policy by selecting one of the predefined scripts, or configuring your own topology script during policy creation.

How to Collect Event Data from Databases

This task describes how to collect event data and optionally topology data from databases.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

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 118.

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 119.

5. In the **Mappings** page, configure the default mappings of table columns to custom variables.

   For details, see "Configuring Mappings in Database Policies" on page 125.

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 127.

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 130.

8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in Database Policies" on page 136.

9. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that represents your third-party system.

   For details, see "Configuring Topology Scripts in Database Policies" on page 137.

10. Click **OK** to save the policy and close the editor.

11. *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 and optionally topology data from databases.

For examples and end-to-end workflow information on collecting metrics data with or without topology, see "Collecting and Viewing Metrics Data" on page 53.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click ✳ in the toolbar. Then click **Metrics >** 🛢 **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 118.

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 119.

5. *Optional.* In the **Defaults** page, assign default values to the metric attributes.

   For details, see "Configuring Metrics Defaults in Database Policies" on page 128.

6. 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 134.

7. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that represents your third-party system.

   For details, see "Configuring Topology Scripts in Database Policies" on page 137.

8. Click **OK** 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 Topology Data Only from Databases

You can have BSM Connector report only the topology discovered by the BSM Connector database policies.

This task describes how to collect topology data from database tables.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click ✳ in the toolbar. Then click **Topology >** 🛢 **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 119.

5. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that represents your third-party system.

   For details, see "Configuring Topology Scripts in Database Policies" on page 137.

6. Click **OK** 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. Choose the type of data you want to integrate and then complete the following tasks:

Event data (optionally with topology data)

- "Configuring Database Policy Properties" below
- "Configuring the Data Source in Database Policies" on the next page
- "Configuring Mappings in Database Policies" on page 125
- "Configuring Event Defaults in Database Policies" on page 127
- "Configuring Event Rules in Database Policies" on page 130
- "Configuring Options in Database Policies" on page 136
- "Configuring Topology Scripts in Database Policies" on page 137

Metrics data (optionally with topology data)

- "Configuring Database Policy Properties" below
- "Configuring the Data Source in Database Policies" on the next page
- "Configuring Metrics Defaults in Database Policies" on page 128
- "Configuring Metrics Rules in Database Policies" on page 134
- "Configuring Topology Scripts in Database Policies" on page 137

Topology data only

- "Configuring Database Policy Properties" below
- "Configuring the Data Source in Database Policies" on the next page
- "Configuring Topology Scripts in Database Policies" on page 137

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 **Metrics >** 🛢 **Database**.
- In the BSM Connector user interface, click ✳ in the toolbar. Then click **Topology >** 🛢 **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 115
- "How to Collect Metrics Data from Databases" on page 116
- "How to Collect Topology Data Only from Databases" on page 117

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 423.

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 **Metrics >** **Database**.
- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Topology >** **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 in the Enumerating Field is Processed" on the next page

Database drivers packages supplied with BSM Connector

BSM Connector provides the following database driver packages:

Database Driver	Database	Database Driver String
DB2Driver	IBM DB2	com.mercury.jdbc.db2.DB2Driver
OracleDriver	Oracle	com.mercury.jdbc.oracle.OracleDriver
SQLServerDriver	Microsoft SQL Server	com.mercury.jdbc.sqlserver.SQLServerDriver
SybaseDriver	Sybase	com.mercury.jdbc.sybase.SybaseDriver

Note: The database drivers supplied with BSM Connector should only be used in test environments. For production environments, it is recommended that you install a compatible JDBC database driver to ensure best performance.

Understanding How Data in the Enumerating Field is Processed

Since the data in the enumerating field is not a unique key, the data in the column used for the enumerating field must have constantly increasing values or values that changed since the policy run. For example, if before the first policy run the table contains the following data:

Run 1:

ENUM_FIELD	DATA_NAME
1	Alice
1	Alice
2	Bob

The policy reads all entries, stores "1" as the position where all data was sent, and skips the lines with ENUM_FIELD=2, because there is a possibility that new lines with ENUM_FIELD=2 will be added later.

Run 2:

ENUM_FIELD	DATA_NAME
1	Alice
1	Alice
2	Bob
2	Bob
2	Bob

At the end of the run, the policy runs a query with the filter "where ENUM_FIELD>1". However, it does not send any data because it did not reach the end of the values listed as having ENUM_FIELD=2. The policy cannot send this partial list until a new higher value appears in the table.

Run 3:

ENUM_FIELD	DATA_NAME
1	Alice
1	Alice
2	Bob
2	Bob
2	Bob
3	Charlie

At this stage, the policy runs the same query with the filter "where ENUM_FIELD>1", and sends all the data with ENUM_FIELD equals to 2. It also updates the internal variable for last read position to 2, and skips the last line with ENUM_FIELD equals to 3, until new lines with a value higher than 3 appears.

Tasks

This section includes:

- "How to configure the prerequisites" below
- "How to configure the database source" on page 123

How to configure the prerequisites

- There are several key database driver requirements for using this policy.
  - You can use the database drivers supplied with BSM Connector by default, or you can install or copy a compatible JDBC database driver or database access API into the required BSM Connector directory location. Many database driver packages are available as compressed (zipped) archive files or .jar files. Copy the downloaded driver file into the **<BSM Connector root directory>\WEB-INF\lib** subdirectory. If the file is in zip format, unzip the contents to a temporary directory. Stop and restart the BSM Connector service after copying the driver file to the BSM Connector machine.
  - You must know the syntax for accessing the database driver. Examples of common database driver strings are:
    - **com.mercury.jdbc.sqlserver.SQLServerDriver.** DataDirect driver from DataDirect Technologies. It is a driver for those Microsoft SQL databases that use Windows authentication. For details on installing the driver, see the note below.

      > **Note:** To install the MSSQL JDBC driver:
      >
      > - Download the MSSQL JDBC driver from Microsoft:
      >
      >   Microsoft SQL Server JDBC Driver 3.0: http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=21599
      >
      >   Microsoft JDBC Driver 4.0 for SQL Server: http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774

> Unzip the contents to a temporary directory:
>
> - Copy the **sqljdbc4.jar** file to the **<BSM Connector root directory>\WEB-INF\lib\** subdirectory.
> - Restart the **BSM Connector** service.

- **com.mercury.jdbc.oracle.OracleDriver.** A driver for Oracle databases. This driver is deployed with BSM Connector. When using the driver, the database connection URL has the form of: `jdbc:mercury:oracle://<server name or IP address>:<database server port>;sid=<sid>`
- **oracle.jdbc.driver.OracleDriver.** BSM Connector supports the following categories of JDBC driver that are supplied by Oracle:
  - JDBC thin driver for Oracle 7 and 8 databases.
  - JDBC OCI (thick) driver.

    You can access an Oracle database using an OCI driver. If the port or SID are changed, you only need to make the change in the **tnsnames.ora** file (the database policy editors remain unchanged).

    - On the BSM Connector server, install the version of Oracle client that you are using.
    - Connect to the Oracle database using the Oracle OCI driver.
      - Set **ORACLE_HOME** environment variable (**ORACLE_HOME** is the folder where the Oracle client or database has been installed).
      - Add **ORACLE_HOME\lib** to System PATH (on Windows platforms), or **LD_LIBRARY_PATH** env variable (on UNIX platforms).
      - Set **CLASSPATH** environment variable to use Oracle JDBC driver from **ORACLE_HOME\jdbc\lib**.
    - In the **\oracle\oraX\network\admin\tnsnames.ora** file, configure the service name.
    - Add a database policy to BSM Connector, and configure the following settings in the Basic Settings panel in the Source page:
      - **Database connection URL**: `jdbc:oracle:oci8:@<server name>`
      - **Database driver**: `oracle.jdbc.driver.OracleDriver`
      - Enter the database user credentials in the **Database user name** and **Database password** boxes
- **org.postgresql.Driver.** The database driver for the Postgresql database.

- You must know the syntax for the Database Connection URL. The Database Connection URL 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. Database Connection URLs for this policy are:
  - **jdbc:mercury:sqlserver://<hostname>:1433;DatabaseName=master; AuthenticationMethod=type2**

where <hostname> is the name of the host where the database is running.

- **jdbc:mercury:Sybase://<hostname>:<port>;DatabaseName=<dbname>;**
  where <hostname> is the name of the host where the database is running, <port> is the port on which the database interfaces with the driver, and <dbname> is the name of the Sybase database instance. The default Sybase port is 5000.
- **jdbc:mercury:DB2:// <hostname>:<port>;DatabaseName=<dbname>;**
  where <hostname> is the name of the host where the database is running, <port> is the port on which the database interfaces with the driver, and <dbname> is the name of the DB2 database instance. The default DB2 port is 50000.
- **jdbc:mercury:oracle://<hostname or IP address>:<port>;sid=<sid>**
  where <hostname> is the name of the host where the database is running, <port> is the port on which the database interfaces with the driver, and <sid> is the Oracle system ID.
- **jdbc:oracle:thin:@<hostname>:<port>:<dbname>**
  where <hostname> is the name of the host where the database is running, <port> is the port on which the database interfaces with the driver, and <dbname> is the name of the Oracle database instance.

- 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).
- When Windows authentication is used to connect to the database, configure BSM Connector using the following settings:
  - JDBC Connection string: **jdbc:mercury:sqlserver://<hosthost>:1433; DatabaseName=master;AuthenticationMethod=type2**
  - JDBC driver: **com.mercury.jdbc.sqlserver.SQLServerDriver**.
  - Leave the **Database User name** and **Database Password** fields empty, because the Windows user credentials of the account from which the BSM Connector service is running are used to establish a connection to the database.

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 **Database Connection URL** field, type the URL to the database connection (sometimes referred to as an Authentication string).

   Example: `jdbc:mercury:sqlserver://dbserver.example.com:1433;DatabaseName=BSMEVENTS;AuthenticationMethod=type2`

   b. In the **Database Driver** field, type the driver used to connect to the database. Use the Fully Qualified Class Name of the JDBC driver you are using.

   Example: `com.mercury.jdbc.sqlserver.SQLServerDriver`

   c. Specify how often the policy queries the database. Use the drop-down list to specify increments of seconds, minutes, hours, or days.

   d. Specify how long the policy tries to connect to the database. Use the drop-down list to specify increments of seconds. The timeout period must be shorter than the frequency.

2. Specify the query:

   a. In the **SELECT** field, type the SELECT clause to be used in the SQL query. Enter ***** for all fields or a comma separated list of column names to be retrieved from the database.

   Example: `TITLE,DESCRIPTION,SEVERITY,NODEHINT,CIHINT,TIMERECEIVED,ID,IPADDRESS`

   b. In the **FROM** field, type the FROM clause to be used in the SQL query. Enter a table name or a comma separated list of tables from which the selected columns should be extracted.

   Example: `ALL_EVENTS`

   c. In the **Enumerating field** field, type the name for a database field that can be used to order the events that are returned from the database query.

   Example: `ID`

   d. In **Enumerating field type**, select the type of field used to order the result set. This can be a DATE field, an INTEGER field, a DOUBLE floating point numeral field, or a LONG field.

   e. In the **Initial Enumerating Value** field, type the initial value to be used as a condition for the initial run of this policy.

   f. In the **Max rows to fetch** field, type the maximum number of rows the policy retrieves from the database for each run.

3. Click to retrieve the specified table columns from the database. The results are displayed in the Sample Data tabs of the policy.

Related tasks

- "How to Collect Event Data from Databases" on page 115
- "How to Collect Metrics Data from Databases" on page 116
- "How to Collect Topology Data Only from Databases" on page 117

UI Descriptions

For a description of the Source page, see the following sections:

- "Source Page - Database Policies - Basic and Query Settings" on page 431
- "Source Page - Database Policies - Sample Data" on page 433

Configuring Mappings in Database Policies

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**.

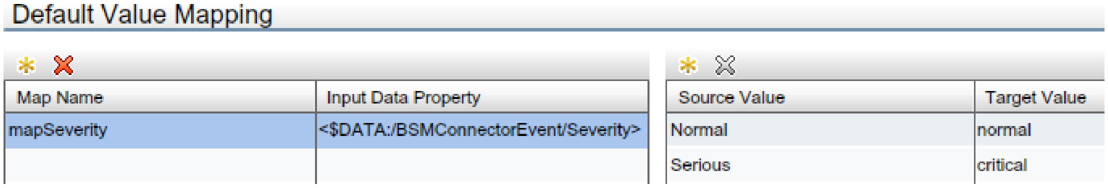
Alternatively, double-click an existing policy to edit it.

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 log file.



Default Value Mapping

Map Name	Input Data Property
mapSeverity	<$DATA:/BSMConnectorEvent/Severity>

Source Value	Target Value
Normal	normal
Serious	critical

Table columns use the following syntax: <$DATA:/BSMConnectorEvent/<*table_column*>>

<*table_column*> is the name of the table column in the third-party database.

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.

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:

- Table columns

  If sample data is available, then the Table Column section shows the table columns specified in the query.

  You can sort the information that appears in the Table Column section so that data appears in either ascending or descending order, indicated by either an up or down arrow at the top of the list. The items in the Table Column section are by default sorted alphabetically in ascending order. To change the sort order, click the arrow at the top of the list.

- Values

The Values section displays the values of a column selected in the Table Columns 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 one or more source and target value pairs to each custom variable.

   - If you are working with sample data, drag the value from the Values list to the Source Value column, and type the target value in the corresponding field.

     Alternatively, click * above the Source Value column and type the source and target values in the corresponding fields.

   - Optionally, use the **Indicators** tab to add indicators to the source or target value fields. After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).

     To insert an indicator in a source or target value field, drag and drop the indicator name (for example, `Normal`) or the indicator name and state (for example, `HTTPServer:Normal`) from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

Related tasks

- "How to Collect Event Data from Databases" on page 115

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Events Only)" on page 410
- "Sample Data Tab- Database Policies" on page 425

- "Indicators Tab" on page 410

Configuring Event Defaults in Database Policies

The Events 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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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 **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:/BSMConnectorEvent/<*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 119.

6. Optionally, 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 Database Policies" on page 125). To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

   Alternatively, type the 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 assigned, 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:/BSMConnectorEvent/SEVERITY>`

   - The source value itself, for example `<$MAP(SEVERITY)>,Critical)>`

Related tasks

- "How to Collect Event Data from Databases" on page 115

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Sample Data Tab- Database Policies" on page 425
- "Mappings Tab (Events Only)" on page 412
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Metrics Defaults in Database Policies

The Default Monitor 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.

To access

In the BSM Connector user interface, click ✳ in the toolbar. Then click **Metrics >** 🛢 **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Monitor Attributes page.

Learn More

Metrics attributes

Each time a metrics policy runs, it extracts raw data from its defined data source and builds a metric structure, which it then forwards to BSM.

A metric structure consists of these attributes:

- Attributes related to the metrics source, category, or status:
  - Monitor name
  - Monitor type
  - Target
  - Time stamp
  - Quality
  - Monitor state
- Attributes related to a specific metric in the source:
  - Name
  - Value
  - CI hint
  - Indicator name

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 default metric attributes common to all metrics collected by this policy, such as monitor name, type, and state.

   > **Tip:** The Policy Constants tab lists the available constants representing metrics quality (status) that are available in BSM Connector. Use drag and drop to add them to the **Quality** attribute.

2. In **Metrics**, click ✱ to add attributes specific to a single metric.

   Alternatively, expand an existing metric group to edit it.

   > **Tip:** After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).

> To insert an indicator, drag and drop the indicator from the Indicators tab to the policy. BSM Connector inserts the indicator name.

3. Optionally, 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: $<*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 119.

4. Optionally, use the **Operations** tab to apply operations to the metric attribute values.

Related tasks

- "How to Collect Metrics Data from Databases" on page 116
- "How to Collect Metrics with Computer - Monitor Topology" on page 55
- "How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60
- "How to Collect Metrics Without Reporting Topology" on page 72

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Default Monitor Attributes and Attributes Tabs" on page 407
- "Sample Data Tab- Database Policies" on page 425
- "Indicators Tab" on page 410
- "Operations Tab (Metrics Only)" on page 412
- "Policy Constants Tab (Metrics Only)" on page 419

Configuring Event Rules in Database Policies

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the source. The settings enable you to configure the event that BSM Connector sends to BSM.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

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 BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to BSM.

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 table column that the policy searches for. You must prefix the column name with `/BSMConnectorEvent/` (for example, `/BSMConnectorEvent/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 "Configuring Event Rules in Database Policies" on the previous page.

   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.

> **Tip:** You can use standard BSM Connector pattern-matching rules when matching values. Select the matches operator and click ▸ in the Operand field to open the pattern matching expression toolbox. The toolbox displays the following:
>
> - **Pattern Matching Expressions.** Click an expression to insert it in the Operand field.
>
> - **Variable Bindings Options.** Variable bindings options include case sensitivity and field separators for the rule. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used. See also "Configuring Options in Database Policies" on page 136.

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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. Optionally, 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:/BSMConnectorEvent/<*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 119.

8. Optionally, 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 Database Policies" on page 125). To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

Alternatively, type the 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 assigned, 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:/BSMConnectorEvent/SEVERITY>`
- The source value itself, for example `<$MAP(SEVERITY)>,Critical)>`

9. Optionally, use the **Pattern Matching Variables** tab to add variables created through 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. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

    It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data from Databases" on page 115
- "How to Collect Metrics Data from Databases" on page 116

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Tab - Rules Only" on page 402
- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Sample Data Tab- Database Policies" on page 425
- "Mappings Tab (Events Only)" on page 412

- "Pattern Matching Variables Tab (Event Rules Only)" on page 419
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Metrics Rules in Database Policies

Rules define what a policy should do in response to a specific type of metric. Each rule consists of a condition and of settings for the metrics generated by the policy. The condition contains the string or pattern that must be matched for the rule to apply. The settings enable you to configure the metrics that BSM Connector sends to BSM.

If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

To access

In the BSM Connector user interface, click ✳ in the toolbar. Then click **Metrics >** 🛢 **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
- "Metrics attributes" below

Rule types

The rule types are:

- **Process on matched rule.** If matched, BSM Connector sends metrics to BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to BSM.

Metrics attributes

Each time a metrics policy runs, it extracts raw data from its defined data source and builds a metric structure, which it then forwards to BSM.

A metric structure consists of these attributes:

- Attributes related to the metrics source, category, or status:
  - Monitor name
  - Monitor type

  - Target
  - Time stamp
  - Quality
  - Monitor state
- Attributes related to a specific metric in the source:
  - Name
  - Value
  - CI hint
  - Indicator name

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.

   The match condition must be a valid boolean expression. The expression can contain one or more operations from the Operations tab. The expression can access the contents of the data that is being processed using the dollar sign ($) notation.
3. *Optional.* If you are creating a rule of the type 'process on matched rule', set **Attributes** for metrics that you want the policy to send. 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: common attributes at the top that apply to all metrics defined this policy, and attributes specific to each metric.

   a. Define the metric attributes common to all metrics collected by this policy, such as monitor name, type, and state.

   > **Tip:** The Policy Constants tab lists the available constants representing metrics quality (status) that are available in BSM Connector. Use drag and drop to add them to the **Quality** attribute.

   b. In **Metrics**, click ✱ to add attributes specific to a single metric.

   Alternatively, expand an existing metric group to edit it.

   > **Tip:** After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator from the Indicators tab to the policy. BSM Connector inserts the indicator name.

   You can create multiple metrics.

Click ▸ to expand a metric attribute group

4. Optionally, 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: $<*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 119.

5. Optionally, use the **Operations** tab to apply operations to the metric attribute values.

Related tasks

- "How to Collect Metrics Data from Databases" on page 116
- "How to Collect Metrics with Computer - Monitor Topology" on page 55
- "How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60
- "How to Collect Metrics Without Reporting Topology" on page 72

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Tab (Rules Only)" on page 410
- "Default Monitor Attributes and Attributes Tabs" on page 407
- "Sample Data Tab- Database Policies" on page 425
- "Indicators Tab" on page 410
- "Operations Tab (Metrics Only)" on page 412
- "Policy Constants Tab (Metrics Only)" on page 419

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**.

Alternatively, double-click an existing policy to edit it.

Click **Options** to open the policy Options page.

Tasks

How to configure options for log file 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 Database Policies" on the previous page.

Related tasks

- "How to Collect Event Data from Databases" on page 115

UI Descriptions

For a description of the Options page, see "Options Page (Events Only)" on page 417.

Configuring Topology Scripts in Database Policies

The topology page enables you to create a topology in BSM's RTSM by selecting a predefined script or creating your own custom script.

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 **Metrics >** 🛢 **Database**.
- In the BSM Connector user interface, click * in the toolbar. Then click **Topology >** 🛢 **Database**.

Alternatively, double-click an existing policy to edit it.

Click **Topology** to open the policy Topology page.

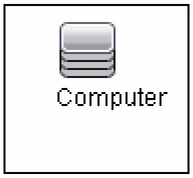
Learn More

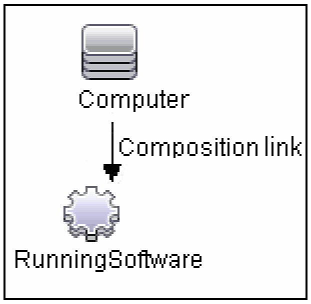
This section includes:

- "Selecting a Topology" on the next page
- "Editing the Topology Script" on page 139
- "Mandatory Values When Reporting Topology Only" on page 141
- "Additional Documentation" on page 141

Selecting a Topology

The following topology scripts are available for integrating events, metrics, and topology:

Topology Script	Events Integration	Metrics Integration	Topology Integration
No Topology Select if you do not want to send any topology (although metrics and event data is still sent).	yes	yes	not available
Computer - Monitor (legacy) Creates a topology with a Computer CI connected to a SiteScope CI with a Monitored By link. **Note:** The Computer - Monitor topology script has been deprecated. For new metrics integrations, use the Computer, Computer - Running Software, or a custom topology script. The Computer - Monitor topology integration requires that the names or IP addresses of the nodes that it adds to the RTSM are accessible through DNS resolution. To successfully add a Node CI specified in a policy's `Target` field to the RTSM, BSM Connector must be able to resolve the node's fully qualified domain name and IP address through a DNS service.	not available	yes	not available
Computer Creates a topology with a Computer CI. 	yes	yes	yes

Topology Script	Events Integration	Metrics Integration	Topology Integration
Computer - Running Software Creates a topology with a Computer CI and a Running Software CI connected to it with a **Composition** relationship. The following illustrates the topology created for the Computer - Running Software integration type which retrieves data from a third-party system: 	yes	yes	yes
Custom You create your own topology if you want the retrieved data to be forwarded to specific CIs and not one of the predefined topology scripts. **Note:** You should only select **Custom** if you are familiar with the Jython language, since you must create the topology script in Jython yourself. Depending on the data type you want to collect, we recommend that you select and edit one of the predefined scripts.	yes	yes	yes

Editing the Topology Script

To configure the topology, you can edit the topology script that appears in the Topology page when creating a policy. You can use the script editor provided, or use any other script editor.

You can choose one of the predefined topologies which are already configured with the necessary information. Following are the guidelines for editing the script if you want to create your own topology.

- The Jython language is sensitive to spaces and tabs. For more information on Jython, see http://wiki.python.org/moin/HowToEditPythonCode.
- You must leave the `import` section of the predefined topology scripts as is and only add to it.
- The main body of the script is mandatory and consists of:

```
def DiscoveryMain(Framework)
```

This main function is responsible for creating Object State Holder Vector (OSHV) results. It holds the CI and CI relationship data, and how to map the incoming samples to the CIs.

- Use the built-in "logger" to debug the topology scripts when samples arrive. You do this by modifying the level and type of information reported to the log file:

  a. Add the logger import statement before system_lib import statement to the topology script, for example:

  ```
  import logger
  …
  import system_lib
  ```

  b. Change the log file settings in the **<BSM Connector root directory>/conf/core/Tools/log4j/PlainJava/bac_integration.properties** file as follows:

  Open the **bac_integration.properties** file in a text editor and locate the following lines in the file:

  ```
  # Jython logger
  log4j.category.PATTERNS_DEBUG=${loglevel}, discovery.appender
  ```

  Change the argument of **log4j.category.PATTERNS_DEBUG** from **${loglevel}** to **DEBUG**, as follows:

  ```
  log4j.category.PATTERNS_DEBUG=DEBUG, discovery.appender
  ```

  c. Save the file. It may take a few seconds for the changes to take effect.

  The debug data is written to the **<BSM Connector root directory>/log/discovery.log** file.

- BSM Connector automatically populates the **Monitored by** attribute of the reported CIs with the values `BSM Connector` and *`<policy name>`*. *`<policy name>`* is the name of the policy as set in the policy's Properties page.

- You can access the names of database columns in database policies. For example, you can access the value of the column `NAME` variable in the following way:

  ```
  name = Framework.getDestinationAttribute("NAME")
  ```

- You may use the following code to set the **Monitored by** attribute on the CIs reported by the script:

  ```
  ems_lib.addMonitoredByForThirdPartySoftware(Framework,
  <computer>)
  ```

- Use the following code to get the routing domain for a Node CI:

  ```
  domainName = system_lib.getDiscoveryDomainByIP(Framework, <IP or
  node DNS name>)
  ```

For more information on the routing domain attribute in RTSM, see the Data Flow Management Guide.

Mandatory Values When Reporting Topology Only

The following values are mandatory when reporting only the topology discovered by the BSM Connector policies, without reporting events or metrics data:

For Topology Script	Field Name	Description
• Computer • Computer - Running Software	`target_name`	Name of the host or machine that generated the event. This can be added manually or taken from: `Framework.getDestinationAttribute ("<someAttribute>")` **Examples:** Log file policies: Framework.getDestinationAttribute("group0") where group0 is the value of the first pattern matching group. Database policies: Framework.getDestinationAttribute("NAME") where NAME is the name of a database column. Web service listener policies: Framework.getDestinationAttribute (“Host") where HOST is the key in the SOAP request <key>Host</key>.
• Computer • Computer - Running Software	`target_ip`	IP of the host or machine. This can be added manually, or calculated using: `HostIPCachingManager.getIPByHostName(target_name)` where `target_name` represents a valid host or machine, or you can use: `HostIPCachingManager.getIPByHostName ("<someAttribute>")`
Computer - Running Software	`name`	Name of Running Software. This can be added manually, or taken from: `Framework.getDestinationAttribute ("<someAttribute>")`

Additional Documentation

For general information on topology scripts, see "Create Jython Code" and "Developing Jython Adapters" in the RTSM Developer Reference Guide.

For information about Java classes that can be used in topology scripts, see "HP Data Flow Management API Reference" in the RTSM Developer Reference Guide.

Tasks

This section includes:

- "How to report topology with event data" below
- "How to report topology with metrics data" below
- "How to report topology without event or metrics data" on the next page

How to report topology with event data

1. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Event >** 🛢 **Database**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.

3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. To create a custom script, see "Editing the Topology Script" on page 139 for more information.

4. Optionally, use the **Sample Data** tab to drag table columns and values to the topology script. Alternatively, you can type the table column directly into the topology script, for example, `ALL_EVENTS`.

   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 119.

How to report topology with metrics data

1. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Metrics >** 🛢 **Database**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.

3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. The Computer - Monitor script does not need to be edited. To create a custom script, see "Editing the Topology Script" on page 139 for more information.

4. Optionally, use the **Sample Data** tab to drag table columns and values to the topology script. Alternatively, you can type the table column directly into the topology script, for example, `ALL_EVENTS`.

   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 119.

How to report topology without event or metrics data

1. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Topology >** **Database**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.

3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. To create a custom script, see "Editing the Topology Script" on page 139 for more information.

4. Map the data discovered by the policy to the relevant attributes in the topology script.

   For a list of mandatory values, see "Mandatory Values When Reporting Topology Only" on page 141.

5. Optionally, use the **Sample Data** tab to drag table columns and values to the topology script. Alternatively, you can type the table column directly into the topology script, for example, `ALL_EVENTS`.

   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 119.

> **Note:** BSM removes CIs from the RTSM if no topology data has been received for them over a period of time. To prevent BSM from deleting the CIs that BSM Connector has sent to BSM, BSM Connector regularly sends a list of active CIs to BSM in addition to the topology data. Inactive CIs are not sent to the RTSM and are marked for deletion in the RTSM.
>
> You can configure BSM Connector to always send CIs regardless of whether topology data has been received for them. This guarantees that CIs reported by BSM Connector are never deleted automatically from the RTSM. This configuration option is known as always touch mode and is recommended for BSM Connector topology-only policies that collect data from incremental data sources such as log files or databases. The first time the policies run, they collect all CIs from the data source. The next time the policies run, they collect new and changed CIs only and then forward only the additions and changes to BSM. The always touch mode ensures that the already existing, unchanged CIs are also reported and are thus not deleted.
>
> For more information, see "Touching of CIs in BSM Connector" on page 23.

Related tasks

- "How to Collect Event Data from Databases" on page 115

- "How to Collect Metrics Data from Databases" on page 116
- "How to Collect Topology Data Only from Databases" on page 117

UI Descriptions

For a description of the Topology page, see the following sections:

- "Topology Page" on page 442
- "Sample Data Tab- Database Policies" on page 425

Troubleshooting Database Policies

This section describes troubleshooting and limitations when working with database policies.

This section includes:

- "Debugging errors" below
- "Database policy file rolling" on the next page
- "Metrics troubleshooting" on page 146
- "Topology troubleshooting" on page 146

Debugging errors

- Check for errors in the following files:

  <BSM Connector root directory>\logs\error.log

  <BSM Connector root directory>\logs\RunMonitor.log

  <BSM Connector root directory>\logs\bac_integration\bac_integration.log.

- You can modify the level and type of information reported to the log files by changing the log file settings in the **<BSM Connector root directory>\conf\core\Tools\log4j\PlainJava\log4j.properties** file. You can instruct the logging mechanism to:

  - Report logged information in less or greater detail than is reported by default.
  - Log all metrics sent by metrics integration policies to BSM.
  - Log all received data from third-party systems.

  To modify log settings:

  a. Open the **log4j.properties** file in a text editor.

  b. To specify that metrics sent by metrics integration policies to BSM be logged:

     i. Locate the following lines in the file:

     ```
     log4j.category.EmsSamplePrinter=${loglevel},
     integration.appender
     log4j.additivity.EmsSamplePrinter=false
     ```

     ii. Change the argument of **log4j.category.EmsSamplePrinter**from **${loglevel}** to

DEBUG, as follows:

```
log4j.category.EmsSamplePrinter=DEBUG, integration.appender
```

iii. Save the file. It may take a few seconds for the changes to take effect.

The results are logged to the **bac_integration.log** file.

c. To specify that all received data from third-party systems be logged:

i. Locate the following lines in the file:

```
log4j.category.EmsEventPrinter=${loglevel}, monitors.appender
log4j.additivity.EmsEventPrinter=false
```

ii. Change the argument of **log4j.category.EmsEventPrinter** from **${loglevel}** to **DEBUG**, as follows:

```
log4j.category.EmsEventPrinter=DEBUG, monitors.appender
```

iii. Save the file. It may take a few seconds for the changes to take effect.

The results are logged to the **RunMonitor.log** file.

> **Tip:** After you have determined the cause of the problem, we recommend that you set log levels to their default settings so as not to overload the system.

- If metrics are created and sent from BSM Connector, but the data is not seen in BSM Service Health, Event Log, or SiteScope reports, look in **<BSM root directory>\log\mercury_wde\wdeIgnoredSamples.log** to make sure the metrics were not dropped due to missing fields or values.
- Change the logging level for Service Health to verify that Service Health received the samples. Open the following file on the Gateway Server machine: **<BSM root directory>\conf\core\tools\log4j\mercury_wde\wde.properties**

  Change the log level parameter to DEBUG in the following lines:

  - `log4j.category.com.mercury.am.platform.wde.decode.IgnoredSamplesLogger=${loglevel}, IgnoredSamples.appender`
  - `log4j.category.com.mercury.am.platform.wde.publish_SamplePublisherSamples=${loglevel}, PublishedSamples.appender`

  Look at the corresponding log files:

  - **<BSM root directory>\logs\mercury_wde\wdeIgnoredSamples.log**
  - **<BSM root directory>\logs\mercury_wde\wdePublishedSamples.log**

Database policy file rolling

A database policy reads the following XML file:

Windows: `%OvDataDir%\tmp\<`*`policyID`*`>.xml`

Linux: `/var/opt/OV/tmp/<`*`policyID`*`>.xml`

The file contains the results of the BSM Connector database queries in an XML file format that database policies can evaluate. The file exists only for policies that integrate events.

To prevent the file from becoming too large, a new version of the file is created when its size reaches 100000 KB. The original file is renamed and stored as a backup file. By default, up to three backup files can be created.

> **Caution:** When the XML file rolls, that is, a new version is created, but the policy has not completed reading all entries, the unread entries are lost and no events are created.

The following defaults are set:

- **Maximum file size:** 100000 KB

  > **Tip:** If not all expected events arrive in BSM, increase the maximum file size to 1000000 KB.

- **Maximum number of backup files:** 3

To change the defaults:

1. Open the following file:

   ```
   <BSM Connector root directory>/groups/master.config
   ```

2. Change the following values as required:

   ```
   _eventRollingMaxFileSizeInKB=100000
   ```

   ```
   _eventRollingMaxBackup=3
   ```

3. Restart the BSM Connector server:

   Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

Metrics troubleshooting

See "Metrics Troubleshooting" on page 87 for more information on troubleshooting and limitations when BSM Connector sends metrics to BSM.

Topology troubleshooting

See "Topology Troubleshooting and Limitations" on page 338 for more information on troubleshooting and limitations when BSM Connector is enabled to report CIs and CIs relationships topology to BSM.

Chapter 5

Integrating Log File Data into BSM

This policy type reads entries in a log file by trying to match text that you choose against a regular expression. When the policy finds a match and certain conditions apply, the policy responds by sending event, metrics, or topology data to BSM. You can define the details of the BSM data sent to BSM based on information in the log file.

Each time that BSM Connector runs this policy, the policy starts from the point in the log file where it stopped reading the last time the policy ran. This ensures that you are notified only of new entries and speeds the rate at which the policy runs.

> **Note:** You must have the format and syntax of the log file that the policy reads. You must construct a regular expression to match on the entries in the log file that contain the data you want to read and forward to BSM. For examples of regular expressions, see "Examples for Log File Policies" on page 226.

What Data Is Collected

The log file policy sends to BSM Connector data that is extracted from any log fle row that matched against the **Log File Pattern** regular expression.

Before setting up the log file policy, you should be clear about the purpose and usage of the data in BSM (for presentation in Operations Management, Service Health, Service Level Management, or reports).

The specific data that is forwarded to BSM is controlled by the type of policy (event, metrics, or topology integration policy) and by the defaults and rules that the policy applies to the data.

Data can also be mapped to a topology to forward data to the correct CI hierarchy in BSM Connector. You can configure topology settings for the policy by selecting one of the predefined scripts, or configuring your own topology script during policy creation.

How to Collect Event Data from Log Files

This task describes how to collect event and optionally topology data from log files.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Event >** ▤ **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 Log File Policy Properties" on page 150.

4. In the **Source** page, define the log file that the policy reads (for example, the path and name of the log file).

   For details, see "Configuring the Data Source in Log File Policies" on page 151.

5. In the **Mappings** page, configure the default mappings of pattern matching groups to custom variables.

   For details, see "Configuring Mappings in Log File Policies " on page 153.

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 Log File Policies" on page 156.

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 Log File Policies" on page 160.

8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in Log File Policies" on page 177.

9. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that represents your third-party system.

   For details, see "Configuring Topology Scripts in Log File Policies" on page 178.

10. Click **OK** to save the policy and close the editor.

11. *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 Log Files

This task describes how to collect metrics and optionally topology data from log files.

For examples and end-to-end workflow information on collecting metrics data with or without topology, see "Collecting and Viewing Metrics Data" on page 53.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click * in the toolbar. Then click **Metrics >** 🗎 **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 Log File Policy Properties" on the next page.

4. In the **Source** page, define the log file that the policy reads (for example, the path and name of the log file).

   For details, see "Configuring the Data Source in Log File Policies" on page 151.

5. *Optional.* In the **Defaults** page, assign default values to the metric attributes.

   For details, see "Configuring Metrics Defaults in Log File Policies" on page 158.

6. In the **Rules** page, define what the policy should do in response to a specific metric.

   For details, see "Configuring Metrics Rules in Log File Policies" on page 163.

7. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that represents your third-party system.

   For details, see "Configuring Topology Scripts in Log File Policies" on page 178.

8. Click **OK** 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 Topology Data from Log Files

You can have BSM Connector report only the topology discovered by the BSM Connector log file policies.

This task describes how to collect topology data from log files.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click ✳ in the toolbar. Then click **Topology >** 🗒 **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 Log File Policy Properties" on the next page.

4. In the **Source** page, define the log file that the policy reads (for example, the path and name of the log file).

   For details, see "Configuring the Data Source in Log File Policies" on page 151.

5. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that

represents your third-party system.

For details, see "Configuring Topology Scripts in Log File Policies" on page 178.

6. Click **OK** 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.

Log File Policy User Interface

Log file policies display different pages depending on the type of data they are integrating. Choose the type of data you want to integrate and then complete the following tasks:

Event data (optionally with topology data)

- "Configuring Log File Policy Properties" below
- "Configuring the Data Source in Log File Policies" on the next page
- "Configuring Mappings in Log File Policies " on page 153
- "Configuring Event Defaults in Log File Policies" on page 156
- "Configuring Event Rules in Log File Policies" on page 160
- "Configuring Options in Log File Policies" on page 177
- "Configuring Topology Scripts in Log File Policies" on page 178

Metrics data (optionally with topology data)

- "Configuring Log File Policy Properties" below
- "Configuring the Data Source in Log File Policies" on the next page
- "Configuring Metrics Defaults in Log File Policies" on page 158
- "Configuring Metrics Rules in Log File Policies" on page 163
- "Configuring Topology Scripts in Log File Policies" on page 178

Topology data only

- "Configuring Log File Policy Properties" below
- "Configuring the Data Source in Log File Policies" on the next page
- "Configuring Topology Scripts in Log File Policies" on page 178

Configuring 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 >** ▤ **Log File**.
- In the BSM Connector user interface, click ✳ in the toolbar. Then click **Metrics >** ▤ **Log File**.

- In the BSM Connector user interface, click ✳ in the toolbar. Then click **Topology >** 📄 **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 Log File Policy Properties" on the previous page.

Related tasks

- "How to Collect Event Data from Log Files" on page 147
- "How to Collect Metrics Data from Log Files" on page 148
- "How to Collect Topology Data from Log Files" on page 149

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 423.

Configuring the Data Source in Log File Policies

The source page of the log file policy editor enables you to specify which log file the policy reads. You can also configure the policy to extract data from the log file. The policy retains that data (as so-called pattern matching groups) 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 >** 📄 **Log File**.
- In the BSM Connector user interface, click ✳ in the toolbar. Then click **Metrics >** 📄 **Log File**.
- In the BSM Connector user interface, click ✳ in the toolbar. Then click **Topology >** 📄 **Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Source** to open the policy Source page.

Learn More

Pattern matching groups

A pattern matching group represents a unit of data that matches a specified pattern and that is then retained for use in event attributes, mappings, and rules.

> **Example:** The log file pattern `(.*) (.*) (.*) (.*)` matches the log file lines

```
Info: user1 logged on
Info: user1 logged off
```

In this example, the parentheses surround the match pattern, and instruct the policy to retain the matched values as pattern matching groups. The patttern `.*` matches all characters, and the space is the delimiter in the log file.

The pattern assigns the string `Info:` to the pattern matching group "group0", `user1` to "group1", `logged` to "group2", and the strings `on` and `off` to "group3". You can then insert the generated pattern matching groups "group0" to "groupn" in event attributes, mappings, and rules.

Tasks

How to configure the log file source

This task describes how configure the log file source and how the policy reads it.

1. Select a server from the server list (only those remote servers that have been configured in BSM Connector are displayed). The BSM Connector server is the default server.

   Alternatively, click **Remote Servers** to add a new server.

2. Type the path to the log file from which you want to extract data:

   - **Remote UNIX.** For reading log files on remote UNIX machines, the path must be relative to the home directory of UNIX user account being used to log on to the remote machine. Click **Remote Servers** and select the UNIX remote server for information about which UNIX user account is being used.

   - **Remote Windows through NetBIOS.** You can also access log files by including the UNC path to the remote log file. For example, `\\remoteserver\sharedfolder\filename.log`.

     This requires that the user account under which BSM Connector is running has permission to access the remote directory using the UNC path.

     If a direct connection using the operating system is unsuccessful, BSM Connector tries to match the `\\remoteserver` with servers currently defined as remote Windows connection profiles (displayed in the remote server list).

     If an exact match is found for `\\remoteserver` in the remote Windows connection profiles, BSM Connector tries to use this connection profile to access the remote log file. If no matching server name is found, the policy reports that the remote log file cannot be found.

     It is not necessary to select a remote Windows server if you are using NetBIOS to connect to remote Windows servers.

   - **Remote Windows through SSH.** Select a remote server from the drop-down list. The path of the log file depends on the type of SSH server installed on the remote Windows server:

     - SSH servers that provide a UNIX-like interface (for example, Cygwin OpenSSH):

       `/cygdrive/<drive_letter>/<directory>/filename.log`

- SSH servers that provide a Windows command prompt (for example, OpenSSH for Windows):

  `<drive_letter>:\<folder>\filename.log`

Optionally, you can use a regular expression to insert date and time variables. For example, you can use a syntax of `s/ex$shortYear$$0month$$0day$.log/` to match date-coded IIS log file names. For details, see "Special Substitution for File Path" on page 225.

3. Select the log file encoding that is used if you are reading a log file whose encoding is different than the BSM Connector system's default encoding.

4. Specify how often the policy reads the log file. Use the drop-down list to specify increments of seconds, minutes, hours, or days.

5. Specify how long the policy should wait before timing out when access to the log file is delayed or not possible. The timeout period must be shorter than the frequency.

6. Specify the text to look for in the log entries, for example `(.*) (.*) (.*) (.*).`

> **Tip:**
>
> - Use parentheses (`()`) to create pattern matching groups.
> - Escape special characters with a backslash (`\`).
>
>   Examples of special characters are the plus sign (`+`), the asterisk (`*`), the question mark (`?`), the at sign (`@`), the exclamation point (`!`), the caret (`^`), brackets (`[]`), and braces (`{}`).
> - For more information about regular expressions, see "Regular Expressions" on page 217.

7. Click to load the log file and apply the log file pattern. Matching results are displayed in the Sample Data tabs of the policy.

Related tasks

- "How to Configure BSM Connector to Access Remote Windows Servers " on page 186
- "How to Configure BSM Connector to Access Remote UNIX Servers " on page 188
- "How to Collect Event Data from Log Files" on page 147
- "How to Collect Metrics Data from Log Files" on page 148
- "How to Collect Topology Data from Log Files" on page 149

UI Descriptions

For a description of the Source page, see "Source Page - Log File Policies" on page 434.

Configuring Mappings in Log File Policies

The Mappings page enables you to map pattern matching groups to custom variables.

To access

In the BSM Connector user interface, click ✱ in the toolbar. Then click **Event >** 🗎 **Log File**.

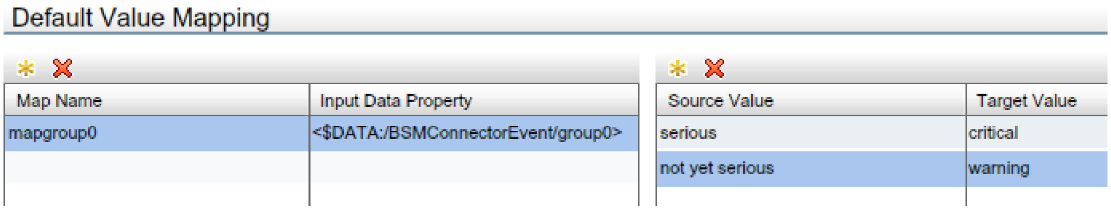
Alternatively, double-click an existing policy to edit it.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

A custom variable consists of a map name, an optional pattern matching group, and one or more source and target value pairs. For example, you can assign the pattern matching group `group1` to the map name `mapgroup0`, 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 `serious` into the event in all places where the variable is used and the source value is `critical` in the log file.



Default Value Mapping

Map Name	Input Data Property
mapgroup0	<$DATA:/BSMConnectorEvent/group0>

Source Value	Target Value
serious	critical
not yet serious	warning

Pattern matching groups use the following syntax: <$DATA:/BSMConnectorEvent/<*pattern_matching_group*>

<*pattern_matching_group*> is the name of the group assigned to a particular content match.

For example, the custom variable `mapgroup0` has the pattern matching group `group0` assigned.

Assigning a pattern matching group to a map name is optional. If you do not assign a pattern matching group 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 log file pattern.

The Pattern Matching Group tab shows the following information, if sample data is available:

- Pattern matching groups

  If sample data is available, then the Pattern Matching Group section shows all groups that match the log file pattern.

  You can sort the information that appears in the Pattern Matching Group section so that data appears in either ascending or descending order, indicated by either an up or down arrow at the top of the list. The items in the Pattern Matching Group section are by default sorted alphabetically in ascending order. To change the sort order, click the arrow at the top of the list.

- Values

  The Values section displays the values of a group selected in the Pattern Matching Group 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 Pattern Matching Group window opens and shows all groups that have the selected value.

When you drag a group from the pattern matching groups 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 group. You can then drag one or more 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 log file mappings

This task describes how to map pattern matching groups to custom variables.

1. Create one or more custom variables.

   If you are working with sample data, drag the group from the pattern matching groups 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. Groups are optional. If you do not assign a group to a variable, you must add the source value directly to the variable when you insert the variable in an event attribute.

2. Add one or more source and target value pairs to each custom variable.

   - If you are working with sample data, drag the value from the Values list to the Source Value column, and type the target value in the corresponding field.

     Alternatively, click * above the Source Value column and type the source and target values in the corresponding fields.

   - Optionally, use the **Indicators** tab to add indicators to the source or target value fields. After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).

     To insert an indicator in a source or target value field, drag and drop the indicator name (for example, `Normal`) or the indicator name and state (for example, `HTTPServer:Normal`) from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

Related tasks

- "How to Collect Event Data from Log Files" on page 147

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Events Only)" on page 410
- "Sample Data Tab - Log File Policies" on page 426
- "Indicators Tab" on page 410

Configuring Event Defaults in Log File Policies

The Events 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 >** ▤ **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 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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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 **Sample Data** tab to drag pattern matching groups and values to the attribute boxes. Alternatively, you can type the path to the pattern matching group directly into the attribute box.

   Pattern matching groups use the following syntax: <$DATA:/BSMConnectorEvent/<*pattern_matching_group*>

   <*pattern_matching_group*> is the name of the group assigned to a particular content match.

   BSM Connector replaces the pattern matching group at runtime with the value of the specified group. If you insert a group value, the value will be used.

   > **Note:**

> - The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server.
> - 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.

6. Optionally, 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 Log File Policies " on page 153). To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

   Alternatively, type the 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(mapgroup0)>`).

   If the custom variable does not have a pattern matching group assigned, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - Name of the pattern matching group, for example `<$MAP(mapgroup0)>`, `<$DATA:/BSMConnectorEvent/group0>`
   - The source value itself, for example `<$MAP(mapgroup0)>,Critical)>`

7. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data from Log Files" on page 147

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Sample Data Tab - Log File Policies" on page 426
- "Mappings Tab (Events Only)" on page 412
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Metrics Defaults in Log File Policies

The Default Monitor 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.

To access

In the BSM Connector user interface, click ✱ in the toolbar. Then click **Metrics >** 📄 **Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Monitor Attributes page.

Learn More

Metrics attributes

Each time a metrics policy runs, it extracts raw data from its defined data source and builds a metric structure, which it then forwards to BSM.

A metric structure consists of these attributes:

- Attributes related to the metrics source, category, or status:
  - Monitor name
  - Monitor type
  - Target
  - Time stamp
  - Quality
  - Monitor state
- Attributes related to a specific metric in the source:
  - Name
  - Value
  - CI hint
  - Indicator name

Tasks

How to configure metrics defaults for log file policies

This task describes how to configure metric attribute defaults for all metrics collected by this policy.

1. Define the default metric attributes common to all metrics collected by this policy, such as monitor name, type, and state.

> **Tip:** The Policy Constants tab lists the available constants representing metrics quality (status) that are available in BSM Connector. Use drag and drop to add them to the **Quality** attribute.

2. In **Metrics**, click ✱ to add attributes specific to a single metric.

   Alternatively, expand an existing metric group to edit it.

   > **Tip:** After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator from the Indicators tab to the policy. BSM Connector inserts the indicator name.

3. Optionally, use the **Sample Data** tab to drag pattern matching groups and values to the attribute boxes. Alternatively, you can type the pattern matching group directly into the attribute box.

   Pattern matching groups use the following syntax: $<*pattern_matching_group*>

   <*pattern_matching_group*> is the name of the group assigned to a particular content match.

   BSM Connector replaces the pattern matching group at runtime with the value of the specified group. If you insert a group value, the value will be used.

   > **Note:**
   >
   > - The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server.
   > - 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.

4. Optionally, use the **Operations** tab to apply operations to the metric attribute values.

Related tasks

- "How to Collect Metrics Data from Log Files" on page 148
- "How to Collect Metrics with Computer - Monitor Topology" on page 55
- "How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60
- "How to Collect Metrics Without Reporting Topology" on page 72

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Default Monitor Attributes and Attributes Tabs" on page 407
- "Sample Data Tab - Log File Policies" on page 426
- "Indicators Tab" on page 410
- "Operations Tab (Metrics Only)" on page 412
- "Policy Constants Tab (Metrics Only)" on page 419

Configuring Event Rules in Log File Policies

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the source. The settings enable you to configure the event that BSM Connector sends to BSM.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

To access

In the BSM Connector user interface, click * in the toolbar. Then click **Event >** ▤ **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 BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to BSM.

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 string in the pattern matching group. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the pattern matching groups and values that the policy searches for in the 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 group that the policy searches for. You must prefix the group name with `/BSMConnectorEvent/` (for example, `/BSMConnectorEvent/group0`).

   If you are working with sample data, you can drag and drop the pattern matching group from the Pattern Matching Group 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 "Configuring Event Rules in Log File Policies" on the previous page.

e. In the **Operand** field, type the value or pattern that you want the policy to compare with the pattern matching group. If you are working with sample data, you can drag the value from the Values list and drop it in the Operand field.

> **Tip:** You can use standard BSM Connector pattern-matching rules when matching values. Select the matches operator and click ▶ in the Operand field to open the pattern matching expression toolbox. The toolbox displays the following:
>
> - **Pattern Matching Expressions.** Click an expression to insert it in the Operand field.
> - **Variable Bindings Options.** Variable bindings options include case sensitivity and field separators for the rule. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used. See also "Configuring Options in Log File Policies" on page 177.

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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
>
> To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. Optionally, use the **Sample Data** tab to drag pattern matching groups and values to the attribute boxes. Alternatively, you can type the path to the pattern matching group directly into the attribute box.

   Pattern matching groups use the following syntax: <$DATA:/BSMConnectorEvent/<*pattern_matching_group*>

   <*pattern_matching_group*> is the name of the group assigned to a particular content match.

   BSM Connector replaces the pattern matching group at runtime with the value of the specified group. If you insert a group value, the value will be used.

   Note:

   - The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server.
   - 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.

8. Optionally, 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 Log File Policies " on page 153). To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

   Alternatively, type the 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(mapgroup0)>`).

   If the custom variable does not have a pattern matching group assigned, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - Name of the pattern matching group, for example `<$MAP(mapgroup0)>, <$DATA:/BSMConnectorEvent/group0>`
   - The source value itself, for example `<$MAP(mapgroup0)>,Critical)>`

9. Optionally, use the **Pattern Matching Variables** tab to add variables created through 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. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data from Log Files" on page 147

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Tab - Rules Only" on page 402
- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Sample Data Tab - Log File Policies" on page 426
- "Mappings Tab (Events Only)" on page 412
- "Pattern Matching Variables Tab (Event Rules Only)" on page 419
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Metrics Rules in Log File Policies

Rules define what a policy should do in response to a specific type of metric. Each rule consists of a condition and of settings for the metrics generated by the policy. The condition contains the string or pattern that must be matched for the rule to apply. The settings enable you to configure the metrics that BSM Connector sends to BSM.

If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

To access

In the BSM Connector user interface, click ✱ in the toolbar. Then click **Metrics >** ▤ **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
- "Metrics attributes" on the next page

Rule types

The rule types are:

- **Process on matched rule.** If matched, BSM Connector sends metrics to BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to BSM.

Metrics attributes

Each time a metrics policy runs, it extracts raw data from its defined data source and builds a metric structure, which it then forwards to BSM.

A metric structure consists of these attributes:

- Attributes related to the metrics source, category, or status:
  - Monitor name
  - Monitor type
  - Target
  - Time stamp
  - Quality
  - Monitor state
- Attributes related to a specific metric in the source:
  - Name
  - Value
  - CI hint
  - Indicator name

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 string in the table column.
2. In the **Rule Content** section, use the **Condition** tab to define the match condition.

   The match condition must be a valid boolean expression. The expression can contain one or more operations from the Operations tab. The expression can access the contents of the data that is being processed using the dollar sign ($) notation.
3. *Optional.* If you are creating a rule of the type 'process on matched rule', set **Attributes** for metrics that you want the policy to send. 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: common attributes at the top that apply to all metrics defined this policy, and attributes specific to each metric.

a. Define the metric attributes common to all metrics collected by this policy, such as monitor name, type, and state.

> **Tip:** The Policy Constants tab lists the available constants representing metrics quality (status) that are available in BSM Connector. Use drag and drop to add them to the **Quality** attribute.

b. In **Metrics**, click ✱ to add attributes specific to a single metric.

Alternatively, expand an existing metric group to edit it.

> **Tip:** After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
>
> To insert an indicator, drag and drop the indicator from the Indicators tab to the policy. BSM Connector inserts the indicator name.

You can create multiple metrics.

Click ▸ to expand a metric attribute group

4. Optionally, use the **Sample Data** tab to drag pattern matching groups and values to the attribute boxes. Alternatively, you can type the pattern matching group directly into the attribute box.

   Pattern matching groups use the following syntax: $<*pattern_matching_group*>

   <*pattern_matching_group*> is the name of the group assigned to a particular content match.

   BSM Connector replaces the pattern matching group at runtime with the value of the specified group. If you insert a group value, the value will be used.

   > **Note:**
   >
   > - The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server.
   > - 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.

5. Optionally, use the **Operations** tab to apply operations to the metric attribute values.

Related tasks

- "How to Collect Metrics Data from Log Files" on page 148
- "How to Collect Metrics with Computer - Monitor Topology" on page 55

- "How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60
- "How to Collect Metrics Without Reporting Topology" on page 72

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Tab (Rules Only)" on page 410
- "Default Monitor Attributes and Attributes Tabs" on page 407
- "Sample Data Tab - Log File Policies" on page 426
- "Indicators Tab" on page 410
- "Operations Tab (Metrics Only)" on page 412
- "Policy Constants Tab (Metrics Only)" on page 419

Policy Rules List

UI Element	Description
✱	**Create New Rule:** Provides the following options: • **Process on matched rule.** If matched, BSM Connector sends metrics to BSM. 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 BSM. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to BSM.
⧉	**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.
⇧	**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.

UI Element	Description
<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	• **Process on matched rule.** If matched, BSM Connector sends metrics to BSM. 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 BSM. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to BSM. 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.

Condition Tab

UI Element	Description
Match	Match condition.

Attributes Tab

Most metrics attributes are represented by string values and must be enclosed in quotation marks (""). You can optionally use the plus sign (+) to add group values (for example, "Disk Usage on " + $group0). For details, see "Configuring Metrics Rules in Log File Policies" on page 163.

The following attributes are represented by numeric values and must *not* be enclosed in quotation marks:

- Time stamp
- Measurement value

Tip: Use the parseDouble() function to convert a quoted string value to a numeric value (for example, parseDouble(“123123”). Alternatively, use parseDouble($group0) if the value in the log file is numeric and without quotation marks. (You may use pattern matching to remove quotation marks from a log file while it is parsed by the policy.)

- Quality

> **Tip:** The Policy Constants tab lists the available constants representing metrics quality (status) that are available in BSM Connector. Use drag and drop to add them to the **Quality** attribute.

UI Element	Description
Monitor Name	Name of a monitor representing a group of metrics that have something in common. **Example:** "Windows CPU Monitor on"+group0
Monitor Type	The monitor type/category. **Example:** "My Integration CPU Monitor "
Target	The target of this monitor (such as the name of a node). **Example:** "mynode.mycompany.com"
Time Stamp	Time stamp in the seconds since Jan 1st 1970 format. Must be a double-precision number. **Example:** parseDouble("123123")
Quality	Metrics quality (status) in BSM Connector terms. Possible values are: QUALITY_ERROR, QUALITY_WARNING, QUALITY_GOOD. **Example:** $group4.equals("error") ? QUALITY_ERROR :$group4.equals("warning") ? QUALITY_WARNING : QUALITY_GOOD
Monitor State	The monitor status. **Example:** "Running..."
Metrics	
✱	**New Item:** Creates new metrics.
✖	**Delete Item:** Deletes the selected metrics.
⇧	**Move Up:** Moves the selected metrics higher in the metrics order.
⇩	**Move Down:** Moves the selected metrics lower in the metrics order.
⊞	**Expand:** Expands the list of metrics to display all details.
⊟	**Collapse:** Collapses the list of metrics to display only the names and hide the details.
Name	Name the Nth metric. **Example:** "CPU Utilization"

UI Element	Description
Value	Value of Nth metric. Must be a double-precision number. **Example:** 80

CI Hint	Contains the CI hint, which BSM uses to identify the CI related to the metric (for example, oraclesid01@@node.example.com or C:@@server.example.com). Use the format `<hint 1>:<hint 2>:...:<hint n>@@<hostname>`.
	Best practices for CI hints
	The CI hint should have sufficient hints to find the corresponding CI.
	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 as a hint using the format `UCMDB:<ci_uuid>`.
	For more information about CI resolution in BSM, see the *Using Operations Management* PDF or online help.
	Note: CI hints are not required when using the Computer - Monitor topology script because the script sets them automatically. However, it is recommended to specify CI hints for custom, Computer, and Computer - Running Software topology scripts.

Indicator Name	Indicator name. Must be enclosed in quotation marks (""). **Example:** "CPULoad"

Sample Data Tab

UI Element	Description
<Search Properties> 🔍 ✖	Entered search string is used to find a pattern matching group. The list changes as you type; only matching items appear. To clear the search results, click ✖.
XML	**Toggle Short/Full Path Notation.** Shows or hides the full path to the pattern matching group. The full path starts with `/BSMConnectorEvents/BSMConnectorEvent`. The Pattern Matching Group section by default shows the short path to the pattern matching group.
🔍	**Find Matching Events.** To find values that belong to more than one pattern matching group, select the value and click 🔍. The Log File Sample Data window opens and shows all pattern matching groups that have the selected value.
⧉	**Toggle Deduplication.** Shows or hides duplicate values.
Pattern Matching Group	Shows all pattern matching groups that match a string in the log file. **Note:** • The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server. • 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.
Values for <...>	Displays the values of the pattern matching group selected in the Pattern Matching Group section.

Indicators Tab

UI Element	Description
🔄	**Refresh.** Loads the configured indicators from the connected BSM server. **Note:** - Loading indicators from the BSM server may take a few seconds. - The BSM Connector server must be configured as a BSM Connector integration server in BSM for the indicators to load successfully. See also "Indicator definitions could not be loaded" on page 112
<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) and event type indicators (ETIs). To insert an indicator in a policy, drag and drop the indicator from the Indicators tab to the relevant field in the policy.

Operations Tab

Expressions and Functions	Description
? :	Conditional operator. Allows usage of conditions and branching logic: "someCondition ? value1 : value2" means "If someCondition is true, assign the value of value1 to result. Otherwise, assign the value of value2 to result." **Example:** $group1.equals("error") ? true : false
+	String concatenation. **Example:** "trap type is " + $trap
<, <=, > , >=, ==, !=	Checks the numerical correctness of the expression. Can be used with numeric values. **Example:** Match: $numberOfLines == 100
&&, ||	To be used to combine any of the above boolean expressions. **Example:** Match: $status.equals("ERROR") \|\| ($numberOfLines == 100)
true, false	Constant Boolean values. **Example:** Match: true

Expressions and Functions	Description
( )	Allows grouping of operators in order to change the order of their execution. **Example:** Monitor Name: ($group1.equals("CPU") ? "CPU Utilization" : $group1.equals("Memory") ? "Memory Usage" : "Unknown") + " " + $group2
boolean contains (String)	Returns true if and only if this string contains the specified sequence of char values. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#contains (java.lang.CharSequence). **Example:** Monitor Name: $group0.contains("monitor")? $group0 : $group0 + "monitor"
boolean endsWith (String)	Tests if this string ends with the specified suffix. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#endsWith (java.lang.String). **Example:** Monitor Name: $group1.endsWith("Operations")? $group1 : $group1 + "Operations"
boolean equals (String)	Compares this string to another string. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#equals (java.lang.Object). **Examples:** Match: "ERROR".equals($status) or Match: $status.equals("ERROR")

Expressions and Functions	Description
boolean equalsIgnoreCase (String)	Compares this String to another String, ignoring case considerations. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#equalsIgnoreCase%28java.lang.String%29 . **Examples:** Match: "ERROR".equalsIgnoreCase($status) or Match: $status.equalsIgnoreCase("ERROR")
boolean exists (String)	Checks for an existence of a property in the processed event and make sure that it is not an empty value. **Example:** Match: exist($status)
String getToken (String, String,int)	Splits input string according to a supplied delimiter (in regular expression format), and returns one of the result strings according to a specified zero-based index. **Example:** getToken($var, "/", 1) will produce "`y`" if $var equals "x/y/z"
int indexOf (String)	Returns the index within this string of the first occurrence of the specified substring. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#indexOf(int). **Example:** Severity: $group0.lastIndexOf("Critical")>-1? "Critical" : "Normal"
int indexOf (String, int)	Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#indexOf (java.lang.String,%20int). **Example:** Severity: $group0.indexOf("Critical",3)>-1? "Critical" : "Normal"
boolean isDouble (String)	Checks if the input string can be interpreted as a double number. **Example:** Match: isDouble($size)

Expres-sions and Functions	Description
boolean isEmpty()	Tests for an empty string (length() == 0). For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#isEmpty(). **Example:** Description: $group1.isEmpty()?$group0 :$group1
boolean isInt (String)	Checks if the input string can be interpreted as an integer number. **Example:** Match: isInt($size)
int lastIndexOf (String, int)	Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#lastIndexOf(java.lang.String,%20int). **Example:** Severity: $group0.lastIndexOf("Critical",2)>-1? "Critical" : "Normal"
int lastIndexOf (String)	Returns the index within this string of the rightmost occurrence of the specified substring. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#lastIndexOf(java.lang.String). **Example:** Severity: $group0.lastIndexOf("Critical")>-1? "Critical" : "Normal"
int length()	Returns the length of this string. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#length(). **Example:** Description: $group1.length() <10 ? $group0+$group1 :$group1
boolean matches (String)	Tells whether or not the string matches the given regular expression. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#matches(java.lang.String). **Example:** Severity: $group0.matches("(.*)Critical(.*)")? "Critical" : "Normal"
double parseDouble (String)	Use to convert strings to numeric values. The input string should be a valid representation of an integer or a floating point number. **Note:** Calling this function on a string that cannot be interpreted as a number causes an error and the incoming data is dropped. **Example:** Match: parseDouble($size) > 10

Expressions and Functions	Description
int parseInt (String)	Use to convert strings to numeric values. The input string should be a valid representation of an integer or a floating point number. **Note:** Calling this function on a string that cannot be interpreted as a number causes an error and the incoming data is dropped. **Example:** Match: parseInt($size) > 10
String resolveHostIP (String)	Performs DNS resolution from a server to its IP address. If the DNS resolution fails, the function returns the value `unknown host`. **Example:** Target: resolveHostIP($host)
String resolveHostName (String)	Performs DNS resolution from an IP address to a fully qualified domain name. If the DNS resolution fails, the function returns the original input host name. **Example:** Target: resolveHostName($host)
boolean startsWith (String)	Tests if this string starts with the specified prefix. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#startsWith(java.lang.String). **Example:** Monitor Name: $group1.startsWith("Operations")? $group1 : "Operations"+$group1
boolean startsWith (String, int)	Tests if the substring of this string beginning at the specified index starts with the specified prefix. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#startsWith(java.lang.String,%20int). **Example:** Monitor Name: $group1.startsWith("Operations",2)? $group1 : "Operations" + $group1

Expressions and Functions	Description
long str_to_seconds (String, String)	Calculates the timestamp (in seconds, since January 1, 1970 format) held in the first String using the format in the second string. True if the date specified in `$time` in `yyyy-MM-dd HH:mm:ss.SSS` format is later than the current time. For more details, see http://download.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html. **Example:** Match: str_to_seconds ($time,"yyyy-MM-dd HH:mm:ss.SSS") > time () **Note:** Use the following symbols to represent time: Year - `y'; Month - `M"; Day of month - `d'; Hour - `H'; Minute - `m'; Second - `s'
String substring (int)	Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#substring (int). **Example:** Title: $group0.substring(2)
String substring (int, int)	Returns a new string that is a substring of this string. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#substring (int). **Example:** Title: $group0.substring(2,9)
int time()	Returns the current time, in seconds, since January 1, 1970 format. **Example:** Match: $timeStampField > (time()-600) True if the value of the `$timeStampField` is newer then ten minutes ago (in seconds, since January 1, 1970 format).
String toLowerCase()	Converts all of the characters in this to lower case using the rules of the default locale. **Example:** Title: $group0.toLowerCase()

Expressions and Functions	Description
String toUpperCase()	Converts all of the characters in this to upper case using the rules of the default locale. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#toUpperCase(). **Example:** Title: $group0.toUpperCase()
String trim()	Returns a copy of the string, with leading and trailing whitespace omitted. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#trim(). **Example:** Category: $group3.trim()

Policy ConstantsTab

UI Element	Description
QUALITY_ERROR	Maps the metrics quality to the critical HI state (that is, the state that corresponds to the critical severity; if critical does not exist on the CI, major, minor, or warning).
QUALITY_WARNING	Maps the metrics quality to the warning HI state (that is, the state that corresponds to the warning severity; if warning does not exist on the HI, the closest state to it except for states representing normal severity).
QUALITY_GOOD	Maps the metrics quality to the normal HI state (that is, the state that corresponds to the normal severity).

Configuring Options in 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 >** ▤ **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 log file 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 Log File Policies" on the previous page.

Related tasks

- "How to Collect Event Data from Log Files" on page 147

UI Descriptions

For a description of the Options page, see "Options Page (Events Only)" on page 417.

Configuring Topology Scripts in Log File Policies

The topology page enables you to create a topology in BSM's RTSM by selecting a predefined script or creating your own custom script.

To access

- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Event >** 📄 **Log File**.
- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Metrics >** 📄 **Log File**.
- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Topology >** 📄 **Log File**.

Alternatively, double-click an existing policy to edit it.

Click **Topology** to open the policy Topology page.

Learn More

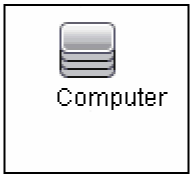
This section includes:

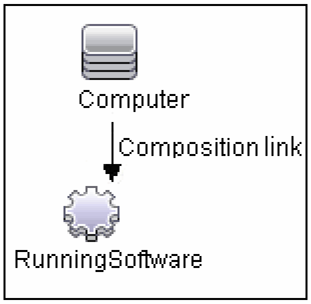
- "Selecting a Topology" below
- "Editing the Topology Script" on page 180
- "Mandatory Values When Reporting Topology Only" on page 182
- "Additional Documentation" on page 182

Selecting a Topology

The following topology scripts are available for integrating events, metrics, and topology:

Topology Script	Events Integration	Metrics Integration	Topology Integration
No Topology Select if you do not want to send any topology (although metrics and event data is still sent).	yes	yes	not available

Topology Script	Events Integration	Metrics Integration	Topology Integration
Computer - Monitor (legacy) Creates a topology with a Computer CI connected to a SiteScope CI with a Monitored By link. **Note:** The Computer - Monitor topology script has been deprecated. For new metrics integrations, use the Computer, Computer - Running Software, or a custom topology script. The Computer - Monitor topology integration requires that the names or IP addresses of the nodes that it adds to the RTSM are accessible through DNS resolution. To successfully add a Node CI specified in a policy's `Target` field to the RTSM, BSM Connector must be able to resolve the node's fully qualified domain name and IP address through a DNS service.	not available	yes	not available
Computer Creates a topology with a Computer CI. 	yes	yes	yes

Topology Script	Events Integration	Metrics Integration	Topology Integration
Computer - Running Software Creates a topology with a Computer CI and a Running Software CI connected to it with a **Composition** relationship. The following illustrates the topology created for the Computer - Running Software integration type which retrieves data from a third-party system: 	yes	yes	yes
Custom You create your own topology if you want the retrieved data to be forwarded to specific CIs and not one of the predefined topology scripts. **Note:** You should only select **Custom** if you are familiar with the Jython language, since you must create the topology script in Jython yourself. Depending on the data type you want to collect, we recommend that you select and edit one of the predefined scripts.	yes	yes	yes

Editing the Topology Script

To configure the topology, you can edit the topology script that appears in the Topology page when creating a policy. You can use the script editor provided, or use any other script editor.

You can choose one of the predefined topologies which are already configured with the necessary information. Following are the guidelines for editing the script if you want to create your own topology.

- The Jython language is sensitive to spaces and tabs. For more information on Jython, see http://wiki.python.org/moin/HowToEditPythonCode.
- You must leave the `import` section of the predefined topology scripts as is and only add to it.
- The main body of the script is mandatory and consists of:

```
def DiscoveryMain(Framework)
```

This main function is responsible for creating Object State Holder Vector (OSHV) results. It holds the CI and CI relationship data, and how to map the incoming samples to the CIs.

- Use the built-in "logger" to debug the topology scripts when samples arrive. You do this by modifying the level and type of information reported to the log file:

  a. Add the logger import statement before system_lib import statement to the topology script, for example:

  ```
  import logger
  …
  import system_lib
  ```

  b. Change the log file settings in the **<BSM Connector root directory>/conf/core/Tools/log4j/PlainJava/bac_integration.properties** file as follows:

  Open the **bac_integration.properties** file in a text editor and locate the following lines in the file:

  ```
  # Jython logger
  log4j.category.PATTERNS_DEBUG=${loglevel}, discovery.appender
  ```

  Change the argument of **log4j.category.PATTERNS_DEBUG** from **${loglevel}** to **DEBUG**, as follows:

  ```
  log4j.category.PATTERNS_DEBUG=DEBUG, discovery.appender
  ```

  c. Save the file. It may take a few seconds for the changes to take effect.

  The debug data is written to the **<BSM Connector root directory>/log/discovery.log** file.

- BSM Connector automatically populates the **Monitored by** attribute of the reported CIs with the values `BSM Connector` and *`<policy name>`*. *`<policy name>`* is the name of the policy as set in the policy's Properties page.

- You can access values such as `group0`, `group1`, and so forth, from log file policies. For example, you can access the value of the second variable in the log file line in the following way:

  ```
  group1 = Framework.getDestinationAttribute("group1")
  ```

- You may use the following code to set the **Monitored by** attribute on the CIs reported by the script:

  ```
  ems_lib.addMonitoredByForThirdPartySoftware(Framework,
  <computer>)
  ```

- Use the following code to get the routing domain for a Node CI:

  ```
  domainName = system_lib.getDiscoveryDomainByIP(Framework, <IP or
  node DNS name>)
  ```

For more information on the routing domain attribute in RTSM, see the Data Flow Management Guide.

Mandatory Values When Reporting Topology Only

The following values are mandatory when reporting only the topology discovered by the BSM Connector policies, without reporting events or metrics data:

For Topology Script	Field Name	Description
• Computer • Computer - Running Software	`target_name`	Name of the host or machine that generated the event. This can be added manually or taken from: `Framework.getDestinationAttribute ("<someAttribute>")` **Examples:** Log file policies: Framework.getDestinationAttribute("group0") where group0 is the value of the first pattern matching group. Database policies: Framework.getDestinationAttribute("NAME") where NAME is the name of a database column. Web service listener policies: Framework.getDestinationAttribute (“Host") where HOST is the key in the SOAP request <key>Host</key>.
• Computer • Computer - Running Software	`target_ip`	IP of the host or machine. This can be added manually, or calculated using: `HostIPCachingManager.getIPByHostName(target_name)` where `target_name` represents a valid host or machine, or you can use: `HostIPCachingManager.getIPByHostName ("<someAttribute>")`
Computer - Running Software	`name`	Name of Running Software. This can be added manually, or taken from: `Framework.getDestinationAttribute ("<someAttribute>")`

Additional Documentation

For general information on topology scripts, see "Create Jython Code" and "Developing Jython Adapters" in the RTSM Developer Reference Guide.

For information about Java classes that can be used in topology scripts, see "HP Data Flow Management API Reference" in the RTSM Developer Reference Guide.

Tasks

This section includes:

- "How to report topology with event data" below
- "How to report topology with metrics data" below
- "How to report topology without event or metrics data" on the next page

How to report topology with event data

1. In the BSM Connector user interface, click ✳ in the toolbar. Then click **Event >** 📄 **Log File**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.

3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. To create a custom script, see "Editing the Topology Script" on page 180 for more information.

4. Optionally, use the **Sample Data** tab to drag pattern matching groups and values to the topology script. Alternatively, type the pattern matching group directly into the topology script, for example, `group0`.

   BSM Connector replaces the pattern matching group at runtime with the value of the specified group. If you insert a group value, the value will be used.

   Note:

   - The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server.
   - 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.

How to report topology with metrics data

1. In the BSM Connector user interface, click ✳ in the toolbar. Then click **Metrics >** 📄 **Log File**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.

3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. The Computer - Monitor script does not need to be edited. To create a custom script, see "Editing the Topology Script" on page 180 for more information.

4. Optionally, use the **Sample Data** tab to drag pattern matching groups and values to the topology script. Alternatively, type the pattern matching group directly into the topology script, for example, `group0`.

   BSM Connector replaces the pattern matching group at runtime with the value of the specified group. If you insert a group value, the value will be used.

Note:

- The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server.
- 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.

How to report topology without event or metrics data

1. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Topology >** ▤ **Log File**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.
3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. To create a custom script, see "Editing the Topology Script" on page 180 for more information.

4. Map the data discovered by the policy to the relevant attributes in the topology script.

   For a list of mandatory values, see "Mandatory Values When Reporting Topology Only" on page 182.

5. Optionally, use the **Sample Data** tab to drag pattern matching groups and values to the topology script. Alternatively, type the pattern matching group directly into the topology script, for example, `group0`.

   BSM Connector replaces the pattern matching group at runtime with the value of the specified group. If you insert a group value, the value will be used.

   Note:

   - The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server.
   - 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.

Note: BSM removes CIs from the RTSM if no topology data has been received for them over a period of time. To prevent BSM from deleting the CIs that BSM Connector has sent to BSM, BSM Connector regularly sends a list of active CIs to BSM in addition to the topology data. Inactive CIs are not sent to the RTSM and are marked for deletion in the RTSM.

You can configure BSM Connector to always send CIs regardless of whether topology data has been received for them. This guarantees that CIs reported by BSM Connector are never deleted automatically from the RTSM. This configuration option is known as always touch mode and is recommended for BSM Connector topology-only policies that collect data from incremental data sources such as log files or databases. The first time the policies run, they collect all CIs from the data source. The next time the policies run, they collect new and changed CIs only and then forward only the additions and changes to BSM. The always touch mode ensures that the already existing, unchanged CIs are also reported and are thus not deleted.

For more information, see "Touching of CIs in BSM Connector" on page 23.

Related tasks

- "How to Collect Event Data from Log Files" on page 147
- "How to Collect Metrics Data from Log Files" on page 148
- "How to Collect Topology Data from Log Files" on page 149

UI Descriptions

For a description of the Topology page, see the following sections:

- "Topology Page" on page 442
- "Source Page - Log File Policies" on page 434

Remote Servers

This section covers the following topics:

- "Remote Servers Overview" below

  This section describes how to set up the connection between and Windows or UNIX remote servers. BSM Connectorcan read log files on remote servers and generate events, BSM Connectormetrics, or topology based on entries in these log files.

- "Secure Shell (SSH) Overview " on page 201

  This section describes how to configure an SSH connection between BSM Connector and Windows or UNIX remote servers. The SSH connection method requires installing an SSH server on each remote server to be connected.

- "UNIX Operating System Adapters" on page 213

  This section describes how to use BSM Connector UNIX operating system adapters to extend BSM Connector to access log files on remote servers that run other versions of UNIX, in addition to those supported by default. This is done by configuring an adapter file to support the particular version of UNIX you want to access.

Remote Servers Overview

BSM Connector must be able to establish a connection to the servers you want to access log files. It must also be authenticated as a user having account permissions to run command line tools on the UNIX remote machine as a remote user.

Microsoft Windows/UNIX Remote server options are used to set up the connection properties, such as credentials and protocols, so that BSM Connector can access log files in remote environments. Multiple policies can access the same remote server. You can also create multiple remote servers for the same host machine.

> **Note:** If multiple Windows remote servers are configured for the same host machine using the NetBIOS method, the connection fails. This is because Windows does not permit multiple connections to a server or shared resource by the same user, using more than one user name (System error 1219).

How to Configure BSM Connector to Access Remote Windows Servers

This task describes the steps involved in configuring BSM Connector to access data on remote Windows servers.

1. *Windows Server 2008 remote servers only:* Disable UAC remote restrictions.

   BSM Connector supports accessing Microsoft Windows Server 2008 remote servers with User Account Control (UAC) enabled or disabled. Where UAC is enabled, you must disable the UAC remote restrictions.

   a. Click **Start**, click **Run**, type `regedit`, and then press ENTER.

   b. Locate and then click the following registry subkey: **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System**

   c. If the LocalAccountTokenFilterPolicy registry entry does not exist, follow these steps:

      - On the **Edit** menu, select **New >DWORDValue**.
      - Type **LocalAccountTokenFilterPolicy**, and then press ENTER.

   d. Right-click **LocalAccountTokenFilterPolicy**, and then click **Modify**.

   e. In the **Value data** box, type **1**, and then click **OK**.

   f. Exit Registry Editor.

2. Enable BSM Connector to access data on remote Windows servers by performing one of the following steps:

   - Define an individual remote Windows server connection profile for each server.

     Accessing remote Windows server data requires authenticated access to the remote server. A Windows server connection profile provides the necessary address and login credentials for BSM Connector to log on to a remote server and to access the Windows performance registry on that remote machine.

     To log on to a remote server using the Windows server connection profile, either:

     - Log on to the remote server as a user with administrator privileges, or
     - Create or modify a user account on the remote server that corresponds with the

connection method and login permissions used in the BSM Connector connection profile for that server.

- Set domain access privileges to permit BSM Connector to access remote servers.

  BSM Connector for Windows automatically generates a list of servers visible in the local domain. BSM Connector running on Windows may be able to use this list to access remote Windows servers without having to create individual connection profiles for each server.

  To set domain privileges, use one of the following methods:

  - **Set the BSM Connector service to run as a user in the Domain Admin group.**

    By default, BSM Connector is installed to run as a `Local System account`. You can set the BSM Connector service to log on as a user with domain administration privileges. This gives BSM Connector access privileges to access server data within the domain.

    To change the user account of the BSM Connector service:

    - In **Administrative Tools**, open **Services**, and select **BSM Connector** from the list of services. The BSM Connector Properties dialog box opens.
    - Click the **Log On** tab, and in the **Log on as** area, enter an account that can access the remote servers.
    - Click **OK** to save your settings and close the BSM Connector Properties dialog box.
    - Right-click **BSM Connector**. Click **Stop** to stop the BSM Connector service.
    - Click **Start**. The BSM Connector service now uses the new account.

  - **Add the server where BSM Connector is running to the Domain Admin group in ActiveDirectory (for Windows 2000 or later).**

    With this option, the BSM Connector service is set to log on as a `Local System account`, but the machine where BSM Connector is running is added to a group having domain administration privileges.

  - **Edit the registry access permissions for all machines in the domain to enable non-admin access.**

    This option requires changes to the registry on each remote machine that you want to access. This means that while the list of servers in the domain includes all machines in the domain, only those remote machines whose registry has been modified can be accessed without use of a connection profile.

  > **Note:** If you configure the BSM Connector service to run as a domain user, BSM Connector uses this account for all Windows-related authorization. You must ensure that this account has the necessary privileges across the domain.

3. Decide which method to use to connect to the Windows remote server. The following methods are available:

   - **NetBIOS.** The default server-to-server communication protocol for Microsoft Windows networks. While NetBIOS provides ease of connectivity, it does have the disadvantage of being relatively vulnerable in terms of network security

> **Note:** If multiple Windows remote servers are configured for the same host computer using the NetBIOS method, the connection fails. This is because Windows does not permit multiple connections to a server or shared resource by the same user, using more than one user name (System error 1219).

- **SSH.** Secure Shell, a more secured communication protocol that can be installed on Microsoft Windows networks. This connection method requires installing an SSH server on each remote server to be connected.

  You can choose between a password-based and a key-based authorization method.

  See "Secure Shell (SSH) Overview " on page 201 for more information about the required SSH setup.

4. Configure the Windows system as a remote server in BSM Connector. For details, see "Configuring Remote Servers " on the next page.

How to Configure BSM Connector to Access Remote UNIX Servers

This task describes the steps involved in configuring BSM Connector to access remote UNIX servers.

To configure access to remote UNIX servers:

1. Define a remote UNIX Server Connection Profiles.

   Accessing remote UNIX server data requires authenticated access to the remote server. A UNIX server connection profile provides the necessary address and login credentials for BSM Connector to log on to a remote server.

   To log on to a remote server using the UNIX server connection profile, do one of the following:

   - Log on to the remote server as a user with administrator privileges.
   - Create or modify a user account on the remote server that corresponds with the connection method and login permissions used in the BSM Connector connection profile for that server.

2. Decide which method to use to connect to the UNIX remote server. The following methods are available:

   - **rlogin.** Logs in to the remote server using the rlogin protocol. You can set up your remote servers to require a password for rlogin, or to enable access without a password (like rsh).
   - **SSH.** Logs in to the remote server using Secure Shell, a more secured communication protocol. This may require additional software and setup depending on the version of UNIX.

     Using SSH requires that digital certificates be installed on each of the servers to which you are connecting.

     You can choose between a password-based and a key-based authentication method.

     See "Secure Shell (SSH) Overview " on page 201 for more information about the required SSH setup.

- **telnet.** Logs in to the remote server using Telnet. Telnet is a popular method for connecting to remote UNIX servers. You can set up your remote servers to require a password for telnet, or to enable access without a password (like rsh).

3. Configure the UNIX system as a remote server in BSM Connector. For details, see "Configuring Remote Servers " below.

Configuring Remote Servers

You can use the Remote Servers page to add, edit, or delete remote servers.

> **Tip:** You can create multiple remote servers for the same host machine.

To access

- Click ✱ in the toolbar, then click **Event >** ▤ **Log File**. In the **Source** page, click **Remote Servers**.
- Click ✱ in the toolbar, then click **Metrics >** ▤ **Log File**. In the **Source** page, click **Remote Servers**.
- Click ✱ in the toolbar, then click **Topology >** ▤ **Log File**. In the **Source** page, click **Remote Servers**.

Alternatively, click ▤ in the toolbar of the BSM Connector user interface.

Tasks

This section includes:

- "How to set up remote Windows servers in BSM Connector" below
- "How to set up remote UNIX servers in BSM Connector" on page 191

How to set up remote Windows servers in BSM Connector

1. Make sure BSM Connector is configured to access the Windows remote server. For details, see "How to Configure BSM Connector to Access Remote Windows Servers " on page 186.
2. In the Remote Servers dialog box, click ✱ in the toolbar, then click **New Microsoft Windows Remote Server**.
3. Configure the following information:

   Configure remote server details

   - Type the **Name** by which the remote server should be known in BSM Connector. Names must be unique.
   - Type a **Description** for the remote Windows server.
   - Specify the remote **Server**. Choose one of the following:
     - IP address or UNC name, for example `192.168.1.2` or `\\server1`.
     - IP host name, for example `server1`.

An IP host name works if the BSM Connector server can translate this common name into an IP address by using a hosts file, DNS, or WINS/DNS integration. For more information about IPv6 support, see "Support for Internet Protocol Version 6 " on page 95.

- Select the connection **Method** for accessing the remote Windows server:
  - **NetBIOS.** The default server-to-server communication protocol for Microsoft Windows networks. While NetBIOS provides ease of connectivity, it does have the disadvantage of being relatively vulnerable in terms of network security

    Note: If multiple Windows remote servers are configured for the same host computer using the NetBIOS method, the connection fails. This is because Windows does not permit multiple connections to a server or shared resource by the same user, using more than one user name (System error 1219).

  - **SSH.** Secure Shell, a more secured communication protocol that can be installed on Microsoft Windows networks. This connection method requires installing an SSH server on each remote server to be connected.

    You can choose between a password-based and a key-based authorization method.

    See "Secure Shell (SSH) Overview " on page 201 for more information about the required SSH setup.

- Type the **Username** for the remote server.

  Note: If the server is within the same domain as the BSM Connector server, include the domain name in front of the user login name (for example `<DOMAIN>\<username>`). If using a local machine login account for machines within or outside the domain, include the machine name in front of the user login name (for example `<machinename>\<username>`).

- Type the **Password** for the remote server or the **Passphrase** for the SSH key file.
- *SSH authentication only:* Path and name of the **SSH Key File** that contains the private key for this connection. The default key file is `<BSM Connector root directory>\groups\identity`. This setting applies only when the authentication method is **Key file**.
- Select the **Remote Server Encoding** if the remote server is running an operating system version that uses a different character encoding than the server on which BSM Connector is running. This enables BSM Connector to display encoded content correctly.

  Default value: Cp1252.

Configure SSH details

- Type the **SSH Port Number** on which the remote SSH server is listening.

  Default value: 22

- Specify the **Connection Limit**, that is the number of open connections that BSM Connector permits for this remote server. If there are many policies configured to use this connection, set the number of open connections high enough to relieve a potential bottleneck.

Default value: 3

- Select **Disable connection cache** to turn off connection caching for this remote server. By default, BSM Connector caches open connections.

  Default value: Not selected

- Select **SSH version 2 only** to force BSM Connector to use SSH protocol version 2 only.

  Default value: Not selected

- Select **SSH keep alive mechanism** to engages a keep alive mechanism for SSH version 2 sessions. This option applies only when using the integrated Java Client.

  Default value: Not selected

4. Click **Save...** to save the remote server configuration.

5. *Optional:* Test the connection to the remote server. Select the remote server that you want to test and click ✓.

How to set up remote UNIX servers in BSM Connector

1. Make sure BSM Connector is configured to access the Windows remote server. For details, see "How to Configure BSM Connector to Access Remote UNIX Servers " on page 188.

2. In the Remote Servers dialog box, click * in the toolbar, then click **New UNIX Remote Server**.

3. Configure the following information:

   Configure remote server details

   - Type the **Name** by which the remote server should be known in BSM Connector. Names must be unique.

   - Type a **Description** for the remote UNIX server.

   - Specify the remote **Server**. Choose one of the following:

     - IP address, for example `192.168.1.2`.

     - IP host name, for example `server1`.

       An IP host name works if the BSM Connector server can translate this common name into an IP address by using a hosts file or DNS.

   - Select the **Operating System** that is running on a remote server. This is required so that the correct information can be obtained from that server. Select an operating system from the list.

     The following operating systems are supported when defining Remote Unix servers: AIX, CentOS Linux, FreeBSD, HP-UX, HP/UX, HP/UX64-bit, Linux, MacOSX, NonStopOS, OPENSERVER, Red Hat Enterprise Linux, SCO, SGI Irix, Solaris Zones, Sun Solaris, SunOS, Tru64 5.x, Tru64 Pre 4.x (Digital), and Ubuntu Linux.

     For servers running versions of UNIX that are not included in the list, see "UNIX Operating System Adapters" on page 213.

   - Select the connection **Method** for accessing the remote UNIX server:

  - ○ **rlogin.** Logs in to the remote server using the rlogin protocol. You can set up your remote servers to require a password for rlogin, or to enable access without a password (like rsh).
  - ○ **SSH.** Logs in to the remote server using Secure Shell, a more secured communication protocol. This may require additional software and setup depending on the version of UNIX.

    Using SSH requires that digital certificates be installed on each of the servers to which you are connecting.

    You can choose between a password-based and a key-based authentication method.

    See "Secure Shell (SSH) Overview " on page 201 for more information about the required SSH setup.
  - ○ **telnet.** Logs in to the remote server using Telnet. Telnet is a popular method for connecting to remote UNIX servers. You can set up your remote servers to require a password for telnet, or to enable access without a password (like rsh).

- Type the **Username** for the remote server.

  > **Note:** If the server is within the same domain as the BSM Connector server, include the domain name in front of the user login name (for example <`DOMAIN>\<username>`). If using a local machine login account for machines within or outside the domain, include the machine name in front of the user login name (for example <`machinename>\<username>`).

- Type the **Password** for the remote server or the **Passphrase** for the SSH key file.
- *SSH authentication only:* Path and name of the **SSH Key file** that contains the private key for this connection. The default key file is `<BSM Connector root directory>\groups\identity`. This setting applies only when the authentication method is **Key file**.
- Select the **Remote Server Encoding** if the remote server is running an operating system version that uses a different character encoding than the server on which BSM Connector is running. This enables BSM Connector to display encoded content correctly.

Configure UNIX shell details

- Specify the **Prompt** output that the remote system sends when it is ready to handle a command (for example, `>`)
- Specify the **Login Prompt** output that the remote system sends when it is waiting for the login to be entered (for example, `User name:`).
- Specify the **Password Prompt** output that the system sends when it is waiting for the password to be entered (for example, `Password:`).
- Specify the **Secondary Prompt** that the system sends if the telnet connection to the remote server causes the remote server to prompt for more information about the connection. Separate multiple prompt string by commas (,).

  Example: For Telnet connections to some remote servers, the remote server may ask what terminal type should be emulated for the connection. In this case, enter `Terminal type?`

as the secondary prompt. The response to the secondary prompt is entered in the **Secondary Response** box below.

- Specify the **Secondary Response** to any secondary prompts required to establish connections with this remote server. Separate multiple responses with commas (,).
- In **Initialize Shell Environment**, type the Shell commands to be run at the beginning of the session. Separate multiple commands with a semicolon (;). This option specifies shell commands to be run on the remote computer directly after a Telnet or SSH session has been initiated. These commands can be used to customize the shell for each BSM Connector remote server. Some examples include:
  - The remote shell may not have the correct path set for BSM Connector scripts to run. The following command adds the directory `/usr/local/bin` into the PATH of the current shell on the remote machine: `export PATH=$PATH:/usr/local/sbin`
  - The remote shell may not be initializing the pseudo terminal correctly. Enter the following command to increase the terminal width to 1024 characters: `stty cols 1024;${SHELL}`

> **Note:** Commands after a shell invocation are not run.
>
> - There have been cases where the remote Telnet Server does not echo back the command line properly. This may cause strange behavior for policies that rely on this behavior. Enter the following command to force the remote terminal to echo: `stty echo`
> - Certain UNIX shells have been known to behave erratically with BSM Connector. This includes bash, ksh, and csh. Enter the following command to change the shell to sh for the BSM Connector connection: /bin/sh

- *NonStop OS only:* Specify **Shell Choice Prompt** output that the system sends when it is waiting for the shell to be selected.

  Default value: >
- *NonStop OS only:* Specify the **Shell Name** to be executed.

  Default value: OSS

Configure SSH details

- Type the **SSH Port Number** on which the remote SSH server is listening.

  Default value: 22
- Specify the **Connection Limit**, that is the number of open connections that BSM Connector permits for this remote server. If there are many policies configured to use this connection, set the number of open connections high enough to relieve a potential bottleneck.

  Default value: 3
- Select **Disable connection cache** to turn off connection caching for this remote server. By default, BSM Connector caches open connections.

  Default value: Not selected
- Select **SSH version 2 only** to force BSM Connector to use SSH protocol version 2 only.

Default value: Not selected

- Select **SSH keep alive mechanism** to engages a keep alive mechanism for SSH version 2 sessions. This option applies only when using the integrated Java Client.

  Default value: Not selected

4. Click **Save...** to save the remote server configuration.
5. Optional: Test the connection to the remote server. Select the remote server that you want to test and click ✓.

Related tasks

- "How to Collect Event Data from Log Files" on page 147
- "How to Collect Metrics Data from Log Files" on page 148
- "How to Collect Topology Data from Log Files" on page 149
- "How to Configure Remote Windows Servers for SSH Access " on page 202
- "How to Configure Remote UNIX Servers for SSH Access " on page 208
- "How to Add an Adapter" on page 214

UI Descriptions

This section includes:

- "Remote Servers List" below
- "Remote Server Configuration" on the next page
- "Unix Shell Details" on page 199
- "SSH Details" on page 200

Remote Servers List

UI Element	Description
🔄	**Refresh Remote Server List.** Updates the list of remote servers.
✳	**New Microsoft Windows/UNIX Remote Server.** Opens the New Remote Server Configuration dialog box enabling you to configure a remote server and add it to the list.
✏	**Edit Remote Server.** Enables you to edit the properties of the selected remote server.
❌	**Delete Remote Server.** Deletes the selected server from the list.
✓	**Test Remote Server Configuration.** Tests the connection to the selected remote server. The test results are displayed in a popup window.
Name	Name by which the remote server is known in BSM Connector.

UI Element	Description
Server	IP address or name of the remote server. You can create two remote servers with the same host name.
Type	Type of remote server: Windows or UNIX.
Description	Description for the remote server.

Remote Server Configuration

UI Element	Description
Name	Name by which the remote server is known in BSM Connector.
Description	Description for the remote server.

UI Element	Description
Server	Windows remote servers: • Real IP address or UNC name of the Windows server. • An IP host name also works if the BSM Connector server can translate this common name into an IP address by using a hosts file, DNS, or WINS/DNS integration. • When specifying a literal IPv6 address as the name for the remote server when using the NetBIOS connection method, you must customize the IPv6 address. a. Replace any colon (":") characters with a dash ("-") character. b. Append the text **.ipv6-literal.net** to the IP address. For example, the IPv6 address: `2004:DB8:2a:1005:230:48ff:fe73:982d` would be: `2004-DB8-2a-1005-230-48ff-fe73-982d.ipv6-literal.net` • You can create multiple remote servers for the same host machine. For example, you can create one remote server that uses the NetBIOS protocol and another that uses SSH for the same host machine, provided the name is unique. • To use the same login credentials to configure multiple servers at the same time, enter the server names or addresses separated by a comma (","), semicolon (";"), or a space. For example, `\\server1,\\server2,\\`. UNIX remote servers: • Real IP address or host name of the UNIX server. • An IP host name works if the BSM Connector server can translate this common name into an IP address by using a hosts file or DNS. • You can create multiple remote servers for the same host machine. For example, you can create one remote server that uses the Telnet protocol and another that uses SSH for the same host machine, provided the name is unique. • To use the same login credentials to configure multiple servers at the same time, enter the server names or addresses separated by a comma (","), semicolon (";"), or a space. For example, `server1`, `server2`. **Note:** • Network address translation (NAT) is not supported for BSM Connector policies that require a remote host definition. BSM Connector is unable to determine if an external (real IP) or internal IP (NAT) is used when you configure a log file policy with an IP address or host. To access servers in a NAT environment, we recommend placing

UI Element	Description
	BSM Connector inside the firewall. - By default, BSM Connector connects to remote servers using IPv4 addresses. If you want your environment to resolve host names to IPv6, configure BSM Connector to prefer IPv6 addresses, specify the hostname in the Server field, and make sure that the host name resolves to both an IPv4 and an IPv6 address. For more information about IPv6 support, see "Support for Internet Protocol Version 6 " on page 95.
Operating System	*UNIX remote servers only.* Operating system that is running on a remote server. This is required so that the correct information can be obtained from that server. Select an operating system from the list. The following operating systems are supported when defining Remote Unix servers: AIX, CentOS Linux, FreeBSD, HP-UX, HP/UX, HP/UX64-bit, Linux, MacOSX, NonStopOS, OPENSERVER, Red Hat Enterprise Linux, SCO, SGI Irix, Solaris Zones, Sun Solaris, SunOS, Tru64 5.x, Tru64 Pre 4.x (Digital), and Ubuntu Linux. For servers running versions of UNIX that are not included in the list, see "UNIX Operating System Adapters" on page 213.

UI Element	Description
Method	**Connection types for accessing Windows remote servers:** • **NetBIOS.** The default server-to-server communication protocol for Microsoft Windows networks. While NetBIOS provides ease of connectivity, it does have the disadvantage of being relatively vulnerable in terms of network security **Note:** If multiple Windows remote servers are configured for the same host computer using the NetBIOS method, the connection fails. This is because Windows does not permit multiple connections to a server or shared resource by the same user, using more than one user name (System error 1219). • **SSH.** Secure Shell, a more secured communication protocol that can be installed on Microsoft Windows networks. This connection method requires installing an SSH server on each remote server to be connected. You can choose between a password-based and a key-based authorization method. See "Secure Shell (SSH) Overview " on page 201 for more information about the required SSH setup. **Connection types for accessing UNIX remote servers:** • **rlogin.** Logs in to the remote server using the rlogin protocol. You can set up your remote servers to require a password for rlogin, or to enable access without a password (like rsh). • **SSH.** Logs in to the remote server using Secure Shell, a more secured communication protocol. This may require additional software and setup depending on the version of UNIX. Using SSH requires that digital certificates be installed on each of the servers to which you are connecting. You can choose between a password-based and a key-based authentication method. See "Secure Shell (SSH) Overview " on page 201 for more information about the required SSH setup. • **telnet.** Logs in to the remote server using Telnet. Telnet is a popular method for connecting to remote UNIX servers. You can set up your remote servers to require a password for telnet, or to enable access without a password (like rsh).

UI Element	Description
Username	User name for the remote server. **Note:** (For Windows only) If the server is within the same domain as the BSM Connector server, include the domain name in front of the user login name. For example: <`DOMAIN>\<username>`. If using a local machine login account for machines within or outside the domain, include the machine name in front of the user login name. For example: <`machinename>\<username>`.
Password **Passphrase**	Password for the remote server or the passphrase for the SSH key file.
SSH Key File	Path and name of the **SSH Key file** that contains the private key for this connection. The default key file is `<BSM Connector root directory>\groups\identity`. This setting applies only when the authentication method is **Key file**.
Remote Server Encoding	Encoding for the remote server, if the remote server is running an operating system version that uses a different character encoding than the server on which BSM Connector is running. This enables BSM Connector to display encoded content correctly. **Default value:** UTF-8

Unix Shell Details

Unix Shell Details

UI Element	Description
Prompt	Prompt output when the remote system is ready to handle a command.
Login Prompt	Prompt output when the system is waiting for the login to be entered.
Password Prompt	Prompt output when the system is waiting for the password to be entered.
Secondary Prompt	Secondary prompts if the telnet connection to the remote server causes the remote server to prompt for more information about the connection. Separate multiple prompt string by commas (,). **Example:** For Telnet connections to some remote servers, the remote server may ask what terminal type should be emulated for the connection. In this case, enter `Terminal type?` as the secondary prompt. The response to the secondary prompt is entered in the **Secondary Response** box below.
Secondary Response	Responses to any secondary prompts required to establish connections with this remote server. Separate multiple responses with commas (,).

Unix Shell Details, continued

UI Element	Description
Mask secondary response input field	Hides the secondary response behind asterisks. If you subsequently clear the check box, the hidden data is deleted. **Default value:** Not selected
Initialize Shell Environment	Shell commands to be run at the beginning of the session. Separate multiple commands with a semicolon (;). This option specifies shell commands to be run on the remote machine directly after a Telnet or SSH session has been initiated. These commands can be used to customize the shell for each BSM Connector remote. Some examples include: • The remote shell may not have the correct path set for BSM Connector scripts to run. The following command adds the directory **/usr/local/bin** into the PATH of the current shell on the remote machine: `export PATH=$PATH:/usr/local/sbin` • The remote shell may not be initializing the pseudo terminal correctly. Enter the following command to increase the terminal width to 1024 characters: `stty cols 1024;${SHELL}` **Note:** Commands after a shell invocation are not run. • There have been cases where the remote Telnet Server does not echo back the command line properly. This may cause strange behavior for policies that rely on this behavior. Enter the following command to force the remote terminal to echo: `stty echo` • Certain UNIX shells have been known to behave erratically with BSM Connector. This includes bash, ksh, and csh. Enter the following command to change the shell to sh for the BSM Connector connection: `/bin/sh`
HP NonStop Shell Settings	
Shell Choice Prompt	(For NonStop OS only) Prompt output when the system is waiting for the shell to be selected. **Default value:** >
Shell Name	(For NonStop OS only) Shell name to be executed. **Default value:** OSS

SSH Details

SSH Details

UI Element	Description
SSH Port Number	Port on which the remote SSH server is listening. **Default value:** 22

SSH Details, continued

UI Element	Description
Connection Limit	Number of open connections that BSM Connector permits for this remote. If there are many policies configured to use this connection, set the number of open connections high enough to relieve a potential bottleneck. **Default value:** 3 **Note:** This setting does not effect running tests for a remote server. Tests always create a new connection.
Disable connection cache	Turns off connection caching for this remote. By default, BSM Connector caches open connections. **Default value:** Not selected
SSH version 2 only	Forces BSM Connector to use SSH protocol version 2 only. **Default value:** Not selected
SSH keep alive mechanism	Engages a keep alive mechanism for SSH version 2 sessions. This option applies only when using the integrated Java Client. **Default value:** Not selected

Secure Shell (SSH) Overview

You can use Secure Shell (SSH) to connect to a server and automatically send a command, so that the server runs that command and then disconnects. This is useful for creating automated processing and scripting.

Secure Shell (SSH), sometimes known as Secure Socket Shell, is a UNIX-based command interface and protocol for securely accessing a remote computer. It is widely used by network administrators to remotely control Web and other kinds of servers. SSH commands are encrypted and secure in several ways. Both ends of the client/server connection are authenticated using a digital certificate, and passwords are protected by encryption. Secure Shell client machines make requests of SSH daemons or servers on remote machines.

Accessing remote servers over SSH has the following basic requirements:

- The servers that you want to access with BSM Connector using SSH must have an SSH daemon (or server) installed and active. For details, see "How to Configure Remote Windows Servers for SSH Access " on the next page and "How to Configure Remote UNIX Servers for SSH Access " on page 208.
- The BSM Connector integrated Java SSH client. BSM Connector includes a SSH client written in Java and native to the BSM Connector application code. For details, see:

  "How to Configure the BSM Connector SSH Client " on page 210

It is recommended to use SSH version 2 (SSH2) for all SSH connections.

How to Configure Remote Windows Servers for SSH Access

This task describes the steps involved in configuring remote Windows servers for SSH access.

The servers that you want to access with BSM Connector using SSH must have an SSH daemon (or server) installed and active. HP has tested and recommends Cygwin OpenSSH and OpenSSH for Windows. HP has also tested SSH Server from F-Secure.

The following is a comparison overview of the OpenSSH packages:

OpenSSH Package	Advantages	Disadvantages	Comments
Cygwin OpenSSH	Provides access to either Windows or UNIX-style scripting on a Windows machine. Provides access to UNIX-style system tools and utilities. BSM Connector can access the remote server both as a Windows Remote and /or a UNIX Remote.	Complicated setup procedure.	Available at http://www.cygwin.com/.
OpenSSH for Windows	Simple setup procedure.	Only provides access to Windows commands, scripts, and utilities.	Formerly Network Simplicity "OpenSSH on Windows". Available at http://sshwindows.sourceforge.net/.

Note:

- OpenSSH for Windows and the Cygwin SSH implementations are incompatible with each other. They should not be installed on the same machine.
- If there is more than one version of the Cygwin utilities or more than one SSH server installed on a machine, there may be conflicts that prevent the SSH connections from working. An error message such as could not find entry point is one indication of this kind of conflict. If you suspect this error, search the machine for multiple copies of cygwin1.dll. It may be necessary to remove all versions of the utilities and then reinstall only a single installation to resolve this problem.

To configure SSH for secure connections:

1. Install and configure an SSH server on each remote server to which you want BSM Connector to connect. For details, see "Install Cygwin OpenSSH on Windows " below and "Install OpenSSH for Windows" on page 207.

2. Configure the SSH client to connect to the remote servers

   After you have set up SSH servers or daemons on remote servers, you must configure the integrated Java SSH client that BSM Connector uses to connect to the remote servers. For task details, see "How to Configure the BSM Connector SSH Client " on page 210.

3. If not already done, configure the Windows remote server to use the SSH connection method.

   After confirming SSH connectivity between BSM Connector and the remote server, set up Windows remote server settings in BSM Connector by selecting **SSH** as the connection method.

   For details, see "Configuring Remote Servers " on page 189.

Install Cygwin OpenSSH on Windows

This task describes the steps involved in installing and configuring a Cygwin OpenSSH server on Windows servers.

Note:

- The following instructions assume that no other Cygwin or other SSH utilities are installed on the machine and that the machine has Internet access.

- The user login account used to install and run the SSH daemon needs adequate permissions to install the necessary programs, configure several file options, and control Windows services. It does not need to be the account that BSM Connector uses to connect to the subject server, although that account must be configured within the Cygwin installation before you can access that server with BSM Connector.

To install and configure a Cygwin OpenSSH server on Windows servers:

1. Create a new System Environment variable with the following definition: `CYGWIN = ntsec tty`.

2. Add the string `;C:\cygwin\bin` to your `PATH` variable. Save the changes to the variables.

3. Download the Cygwin setup program into a temporary folder. For example: `C:\temp`. The setup program is used to select, download, and install different packages and components available with Cygwin.

4. Run the downloaded setup program and choose the **Install from Internet** option when prompted to Choose A Download Source. Click **Next** to continue.

5. If prompted, select a root install directory where the Cygwin package should be installed. This is where the SSH daemon and related files are installed. For example, `C:\cygwin`. Click **Next** to continue.

6. If prompted, select a temporary directory where the Cygwin installation files should be stored.

For example, `C:\temp`. Click Next to continue.

7. If prompted, select an Internet Connection option. Normally, you can use **Direct Connection**. Click **Next** to continue.

8. Select a suitable mirror site from which to retrieve the files using the selection list when prompted. Click **Next** to continue.

9. The Setup program queries the mirror site for the packages available and displays a hierarchy tree of package categories. To view and select the packages to download, click on the plus (+) symbol to the left of the category name to expand any of the package trees. Packages that are selected for download and installation display a version number in the **New** column. If a version number is not displayed for a particular package, it is not downloaded and installed. Click Skip to the left of package name to select the package for download.

   > **Note:** Many of the development (`Devel`) and database (`Database`) tools that may be selected by default for download are not necessary to run the SSH daemon and can be deselected to reduce download time and installation space.

   Select each of the following packages for download and installation:

   - `cygrunsrv` from the `Admin` branch
   - `cygwin-doc` from the `Doc` branch
   - `pdksh` from the `Shells` branch
   - `openssh` and `openssl` from the `Net` branch
   - your choice of UNIX-style text editor from the `Editors` branch (for example: `vim` or `emacs`)

   Then click to download the files as prompted.

10. Depending on your installation options, the Cygwin setup downloads and installs the selected packages. You may be prompted to choose to have a shortcut to the Cygwin terminal window added to the Desktop or Program Start menu. Click to continue and complete the installation.

11. After the Cygwin setup is complete, open a Cygwin terminal window by clicking on the Cygwin desktop shortcut or Program Start menu item.

    > **Note:** Depending on the user profile in the Windows system, the default directory that opens in the terminal window may not be within the root Cygwin installation tree. Use the `pwd` command to display the current directory. Typing in the command string `cd /` normally changes the directory to the Cygwin root, which by default corresponds to the Windows `C:\cygwin` directory.

    Update the default Cygwin group file with the group names in use on the machine and on your network. Use the mkgroup utility to update the default Cygwin group file with the groups defined on the server and in your domain. Examples of the commands to use are as follows:

    ```
    mkgroup -l >> ../etc/group mkgroup -d >> ../etc/group
    ```

Note:

- To have Cygwin recognize both domain and local group accounts, run the mkgroup utility twice, once for local users (`-l` option) and once for domain users (`-d` option). Remember to use `>>` syntax and not just `>,` to append entries to the file.
- If you use both the local and domain options, you must manually edit the `/etc/group` file (using the UNIX style text editor you downloaded) to remove any duplicate group entries. You may also want to remove group entries that should not have access to this machine.

Update the default Cygwin user (`passwd`) file with the users defined on the local machine plus any individual domain users you want to grant access to Cygwin on this machine. Use the mkpasswd utility to update the default Cygwin user file.

Examples of the commands to use are as follows:

```
mkpasswd -l >> ..\etc\passwd mkpasswd -d -u username >>
..\etc\passwd (domain users)
```

Note:

- By default, Cygwin is set to run the OpenSSH daemon as the local user called `SYSTEM`. To have Cygwin recognize both domain and local machine user accounts, run the mkpasswd using the `-l` option to add all local users, and run it with the `-d` and `-u` options to add individual domain users. Remember to use `>>` syntax and not just `>,` to append entries to the file.
- If you use both the local and domain options, you must manually edit the `/etc/passwd` file (using the UNIX style text editor you downloaded) to remove any duplicate user entries. You may also change the default `/home` path and default shell for individual users. This may be necessary to install the RemoteNTSSH package in the `/home/sitescopeaccount/` directory of the user account to be used by BSM Connector.

12. Change the active directory to the `/bin` directory by typing `cd /bin`.
13. Create a symbolic link in the `/bin` directory that points to the Windows Command (CMD) shell by entering the following command line (be sure to include the trailing space and period):

    ```
    ln -s /cygdrive/c/winnt/system32/cmd.exe .
    ```
14. We recommend that you change permissions and ownership of several Cygwin files and directories. Also create a log file for the SSH daemon. Enter the following command lines in the Cygwin terminal command line and press ENTER after each command line entered:

    ```
    cd /
    chmod -R og-w .
    chmod og+w /tmp
    touch /var/log/sshd.log
    ```

Note:

- Exact syntax is required, including spaces.
- Inconsistent and incorrectly assigned file and directory permissions can be one reason that the SSH daemon can not be started or that BSM Connector is unable to connect to and run commands or scripts on the remote server.

15. Configure the SSH daemon to run as a Windows service by entering the following command:

    ```
    ssh-host-config -y
    ```

    When presented with the `CYGWIN=` prompt, type `ntsec tty` to match the environment variable you set at the beginning of this procedure. Normally, this configures the SSH daemon or service to restart automatically if the server needs to be restarted.

16. Configure the encryption keys and files for the SSH daemon using the following command:

    ```
    ssh-user-config -y.
    ```

    Enter required passphrases for several keystore files when prompted. The program asks you to re-enter the passphrase for confirmation.

17. You must change the ownership of several files and folders for use by the SSH daemon. The program does not normally run if the permissions on these files enable them to be changed or run by group or "world" level users. Enter the following command strings to restrict access to these files:

    ```
    chown SYSTEM:Users /var/log/sshd.log /var/empty /etc/ssh_h*
    chmod 755 /var/empty
    ```

18. Check the installation by starting and then stopping the CYGWIN sshd service using the **Programs > Administrative Tools > Services** panel.

    Note: Cygwin includes a server utility to start the SSH daemon. However, there have been a number of situations where this method failed to start the server, whereas using the Windows Services panel was able to start the server.

19. Configure the default shell or command environment for the user account you use for accessing the remote machine with BSM Connector. The shell you select effects what types of scripts or commands can be run remotely using the SSH connection. Use the UNIX-style text editor and edit the **/etc/passwd** file. Find the entry for the BSM Connector login account you intend to use and change the shell from **/bin/bash** to the shell you want to use as described below. This is normally the last entry in the line for that account entry.

    - If you chose to have BSM Connector interact with the remote server using the Windows Command shell, change the default shell entry to **/bin/cmd .** Use this option when you plan to use Windows-style batch files and scripts You must also include the symbolic link to the Windows **cmd.exe** kernel in the **/bin** directory as described in a previous step of this procedure.
    - If you chose to have BSM Connector interact with the remote Windows server using a Cygwin UNIX shell, change the default shell entry to be **/bin/pdksh**. The BSM Connector SSH client may not accurately parse Cygwin's default **bash** shell. You must also configure a Remote UNIX server connection to this (Windows) server that connects to the Cygwin SSH

daemon.

Save the changes to the file.

20. Edit the PATH and the default prompt commands in the `/etc/profile` file to make sure that Cygwin can find certain files and that BSM Connector can parse the output from the remote shell. Use the UNIX-style text editor and edit the `/etc/profile` file. Find the `PATH` definition entry near the top of the file. For example:

    ```
    PATH=/usr/local/bin:/usr/bin:/bin:$PATH
    ```

    Change this to include the following:

    ```
    PATH=.:/usr/local/bin:/usr/bin:/bin:$PATH
    ```

21. To change the default prompt commands, edit the `/etc/profile` file, and find the section similar to the following:

    ```
    ;;
    sh   | -sh    | */sh   |\
    sh.exe   | -sh.exe    | */sh.exe )
    #Set a simple prompt
    PS1='$ '
    ;;
    ```

    Immediately under this entry, add the following:

    ```
    ;;
    pdksh   | -pdksh    | */pdksh   |\
    pdksh.exe   | -pdksh.exe    | */pdksh.exe )
    #Set a simple prompt
    PS1='> '
    ;;
    ```

22. Save the changes to the file.

23. Change the active directory to the home directory of the user you have created for BSM Connector.

    After making these changes and starting the SSH daemon, you should be able to connect to the server using an SSH client.

    > **Note:** Any time you run the `mkpasswd -l /etc/passwd` command (for example, when adding a new user), edit the `/etc/passwd` file again to make sure that the default shell for that user is set to the required value for any account being used by BSM Connector.

Install OpenSSH for Windows

This task describes the steps involved in installing and configuring an OpenSSH server on Windows servers.

The OpenSSH for Windows package is an alternative to the Cygwin SSH package and can be easier to install. Like most products, the Cygwin product and the Open SSH for Windows are subject to change. There are cases where some versions of the Cygwin SSH server have not returned the data needed by BSM Connector. If the OpenSSH for Windows package can solve this problem, you should use this package in place of the Cygwin package.

To install and configure an OpenSSH for Windows server on Windows servers:

1. Download and install the OpenSSH for Windows package.

2. Open a command prompt and change to the installation directory (`C:\Program Files\OpenSSH` is the default installation path).

3. Change the active directory to the `OpenSSH\bin` directory.

4. You must update the default group file with the group names in use on the machine and in your network. Use the `mkgroup` utility to update the default OpenSSH group file with the groups defined on the server and in your domain. Examples of the commands to use are as follows:

   ```
   mkgroup -l >> ..\etc\group mkgroup -d >> ..\etc\group
   ```

   Note:

   - To have OpenSSH recognize both domain and local group accounts, run the **mkgroup** utility twice, once for local users (`-l` option) and once for domain users (`-d` option). Remember to use `>>` syntax and not just `>`, to append entries to the file.

   - If you use both the local and domain options, you must manually edit the `/etc/group` file (using the UNIX style text editor you downloaded) to remove any duplicate group entries. You may also want to remove group entries that are not needed or should not have access to this machine.

5. You must update the default OpenSSH user (`passwd`) file with the users defined on the local machine plus any domain user you want to grant access to the SSH server on this machine. Use the **mkpasswd** utility to update the default user file. Examples of the commands to use are as follows:

   ```
   mkpasswd -l >> ..\etc\passwd
   mkpasswd -d -u username >> ..\etc\passwd
   ```

   Note:

   - To have OpenSSH recognize both domain and local machine user accounts, run the **mkpasswd** utility using the `-l` option to add all local users and run it with the `-d` and `-u` options to add individual domain users. Remember to use `>>` syntax and not just `>`, to append entries to the file.

   - If you use both the local and domain options, you must manually edit the `/etc/passwd` file (using the UNIX style text editor you downloaded) to remove any duplicate user entries. You may also change the default `/home` path and shell for individual users (see instructions below).

6. Check the installation by starting the **OpenSSH Server** service using the **Programs > Administrative Tools > Services** panel.

How to Configure Remote UNIX Servers for SSH Access

This task describes the steps involved in configuring remote UNIX Servers for SSH access.

> **Note:** Setting up the SSH hosts on the remote servers you want to access in the UNIX environment can be very complex and is beyond the scope of this document. Some suggested resources on installation of the OpenSSH daemon are:
>
> - Solaris: http://www.sunfreeware.com/openssh.html
> - RedHat Linux 5.3: http://docs.redhat.com/docs/en-US/Red_Hat_Network_Satellite/5.3/html/Reference_Guide/s1-mon-rhnmd.html
> - RedHat Linux 5.4: http://docs.redhat.com/docs/en-US/Red_Hat_Network_Satellite/5.4/html/Reference_Guide/sect-Reference_Guide-Monitoring-RHN_Monitoring_Daemon_rhnmd.html

To configure SSH for secure connections:

1. Secure Shell daemons or servers (`sshd`) must be installed on each remote server you want to access with BSM Connector.

   For details on the requirements for configuring remote UNIX servers for SSH access with BSM Connector in a UNIX environment, see "SSH Configuration Requirements for UNIX Remote Servers" below.

2. Configure the SSH client to connect to the remote servers

   After you have set up SSH servers or daemons on remote servers, you must configure the integrated Java SSH client that BSM Connector uses to connect to the remote servers. For task details, see "How to Configure the BSM Connector SSH Client " on the next page.

3. If not already done, configure the UNIX remote server to use the SSH connection method.

   After confirming SSH connectivity between BSM Connector and the remote server, set up UNIX remote server settings in BSM Connector by selecting SSH as the connection method in the Remote Server configuration dialog box.

   For details, see "Configuring Remote Servers " on page 189.

SSH Configuration Requirements for UNIX Remote Servers

The following are requirements for configuring remote UNIX servers for SSH access in a UNIX environment:

- Secure Shell daemons or servers (`sshd`) must be installed on each remote server you want to access with BSM Connector.
- The SSH daemons on the remote servers must be running and the applicable communication ports must be open. For example, the default for SSH is port number 22.
- A SSH client must be installed on the server where BSM Connector is running. The BSM Connector integrated Java SSH client fills this requirement.

You should verify SSH client-to-server connectivity from the machine where BSM Connector is running to the remote machine you want to access. You should check SSH connectivity outside of the BSM Connector application before setting up remote server connections using SSH in BSM Connector. For example, if BSM Connector is running on Linux, use the following command line to request an SSH connection using SSH2 to the server <remotehost>:

```
ssh -2 <remotehost>
```

This normally returns text information that indicates the version of SSH protocol that is being used. Also, this attempts to authenticate the current user. Use the `-l username` switch to request a login as a different user.

Once you have confirmed SSH connectivity, create or configure UNIX Remote settings in BSM Connector to use SSH as the connection method.

How to Configure the BSM Connector SSH Client

BSM Connector provides a SSH client written in Java that is integrated into the BSM Connector application. This client significantly reduces the required system resources used by BSM Connector when connecting to servers by using SSH. The Java client supports both SSH version 1 (SSH1) and version 2 (SSH2) protocols as well as both password-based and key-based authentication. The BSM Connector configuration for the client is identical for Linux and Windows BSM Connector.

To configure the BSM Connector SSH client:

1. Configure SSH2 connections.

   Some security vulnerabilities have been found in SSH1 that has resulted in SSH2 being considered the current standard. HP therefore recommends that you configure SSH clients and SSH hosts to use the SSH2 protocol between them to communicate. For details, see "Configure SSH2 Connections" below.

2. Configure key-based authentication.

   By default, the integrated SSH client for BSM Connector uses password authentication. You can configure the integrated SSH client to use key-based authentication, which adds an additional level of security through the use of a passphrase and a public-private key authentication. For details, see "Configure Key-based Authentication" on the next page.

Configure SSH2 Connections

While SSH1 and SSH2 are both Secure Shell protocols, they are considered to be two different protocols and are not compatible with each other. Some security vulnerabilities have been found in SSH1 that has resulted in SSH2 being considered the current standard. Most SSH software supports both protocols. However, to be sure that a request for a SSH connection uses SSH2 instead of SSH1, it is necessary to configure SSH clients and SSH hosts to use the same protocol version between them to communicate. In many cases, SSH1 is the default version used for connections, as it is considered the lowest common denominator between a SSH client and a SSH host.

There are two ways to force SSH2 connections:

- Configure all SSH daemons or servers to accept only SSH2 connection requests. This is the most secure option but may be the most time-consuming unless each server was configured for this option when it was installed and activated.
- Configure the SSH client on the BSM Connector server to only make SSH2 requests. Requires changes only to the client on the BSM Connector server. For the integrated Java SSH client,

this can be controlled by the **SSH version 2 only** setting in the **SSH Details** section on the remote server setup page. For details, see "How to Configure BSM Connector to Access Remote Windows Servers " on page 186 and "How to Configure BSM Connector to Access Remote UNIX Servers " on page 188.

By default, the BSM Connector Java client uses the SSH1 protocol if the server it is trying to connect using SSH1 connections. If this negotiation fails, BSM Connector attempts to connect using version 2 protocol.

Configure Key-based Authentication

The integrated SSH client for BSM Connector can be configured to use one of two authentication options:

- Password Authentication is the default method for SSH connections in BSM Connector.
- Key-Based Authentication adds an additional level of security through the use of a passphrase and a public-private key authentication.

To use Key-Based Authentication for SSH remote servers, you must first create a pair of public/private keys. The public key resides on the remote and the private key is kept on the BSM Connector server. You can copy a BSM Connector SSH key to the remote server, or take the remote server key from a remote server and copy it to BSM Connector.

> **Tip:** It is recommended to maintain one key file on the BSM Connector server and copy it to the remote servers instead of generate a file for each machine and copy them to the BSM Connector machine.

Both Cygwin OpenSSH and OpenSSH for Windows come with a key generation tool called ssh-keygen. The ssh-keygen tool enables you to create both protocol version 1 and version 2 keys.

When setting up a UNIX or Windows remote server using the Internal Java Libraries Client, use the key generation tool called MindTerm to create a public/private key pair for RSA (version 1 and version 2) and DSA (version 2).

Creating a Key on the BSM Connector Server

To create a public or private key pair on the BSM Connector server:

1. Open a command window on the BSM Connector server, and run the following command to launch MindTerm:

   ```
   <BSM Connectorroot directory>\java\bin\java -jar c:\<BSM Connector
   root directory>\
   WEB-INF\lib\mindterm.jar
   ```

2. In MindTerm, select **File > Create Keypair > DSA (or RSA)**. Also select OpenSSH **.pub** format.
3. The key pair is written to the **<USER_HOME>\mindterm** directory.
4. Copy the private key (file not ending in ***.pub**) to the **<BSM Connector root directory>\groups** directory.
5. Copy the **identity.pub** file to the **<USER_HOME>/.ssh** directory on the remote machine and

rename it **authorized_keys** (or **authorized_keys2** for SSH2). You also can add content of **identity.pub** to existing **authorized_keys**/**authorized_keys2** file if you want to allow a number of different users to connect to the server with different keys files.

6. On the remote machine, run the command `chmod 744 authorized_keys` in the **<USER_HOME>/.ssh** directory, and make sure that `User` has read, write, and execute permissions, and that `Group` and `Other` have read permissions on the **authorized_keys** file.
7. Create a remote connection in BSM Connector for the remote server using key file authentication and Internal Java Libraries.

   The public key goes in the **<USER_HOME>/.ssh/authorized_keys** file on the remote machines.

   The private key file can be put into the **<BSM Connector root directory>\groups** directory, and renamed **identity**, which enables BSM Connector to automatically take it without having to specify the file path in the remote server configuration page. Alternatively, you can put the private key in any other BSM Connector directory, or outside of BSM Connector.

   The key generated from MindTerm is in **Openssh** format.

   > **Note:** You must verify that the server key and the MindTerm key are at the same level. For example, if the server key is `768` bit and the MindTerm key is `1024` bit, the authentication procedure fails.

To find out what your server is using:

1. Stop the `sshd` service on the remote server. On a Red Hat Linux server, run the command:

   ```
   /etc/rc.d/init.d/sshd stop
   ```

2. Start the sshd service in debug mode on the remote server. On a Red Hat Linux server, run the command:

   ```
   /usr/sbin/sshd -d
   ```

   You should see output similar to `Generating 768 bit RSA key`.

To convert the openSSH key to SEC SSH format:

1. Create a RSA key in MindTerm (which is an openSSH key pair).
2. Run the following command on the remote server to convert the openSSH key to SEC SSH format:

   ```
   ssh-kegen -e -f <public key>
   ```

3. Leave the private key on the BSM Connector server in the openSSH format.

   > **Note:** When using Key-Based authentication, the Key File supplied must be a version 2 private key.

Creating a Key on a UNIX Remote Server and Copying it to the BSM Connector Server

To set up a connection by taking the remote machine key and put it into BSM Connector:

1. Log on to your UNIX remote server as the user that has root permissions.
2. To generate a public/private RSA key pair for protocol version 1, run the following command:

   ```
   $> ssh-keygen -t rsa
   ```

   If you want to generate key pair for version 2, run the command:

   ```
   $> ssh-keygen -t dsa
   ```

   The possible output is:

   ```
   Enter file in which to save the key (~/.ssh/id_rsa):
   Enter passphrase* (empty for no passphrase):
   Enter same passphrase again:
   ```

   where the passphrase is the password used to decode your private key file; it can be left blank.

   Your identification is saved in `~/.ssh/id_rsa` and the public key in `~/.ssh/id_rsa.pub` (protocol version 1); or `~/.ssh/id_dsa` and `~/.ssh/id_dsa.pub` for protocol version 2.
3. The corresponding public key must be listed in the authorized file on the remote host. Add the content of generated public key to this file (the default authorized_keys file location is the ~/.ssh directory).

   To do this run the commands:

   ```
   $> chmod 700 .ssh
   $> cd .ssh
   $> touch authorized_keys (for ver. 2: touch authorized_keys2)
   $> chmod 600 authorized_keys (for ver. 2: chmod 600 authorized_
   keys2)
   $> cat id_rsa.pub >> authorized_keys (for ver. 2: cat id_dsa.pub >>
   authorized_keys2)
   $> rm id_rsa.pub (for ver. 2: rm id_dsa.pub)
   ```
4. Copy the identification file, private key, to the BSM Connector machine.
5. In BSM Connector, create a new UNIX remote server with the following in the Remote Server Configuration:
   - **Method**. SSH.
   - **SSH authorization method**. Keyfile.
   - **SSH Key file**. Path and name of the file that contains the private key; for example, `C:\private.key`.
6. Test the remote server connection.

UNIX Operating System Adapters

You can use BSM Connector UNIX operating system adapters to extend BSM Connector to access log files on remote servers that run other UNIX platforms, in addition to those supported by default. This is done by configuring an adapter file to support the particular UNIX platform you want to access.

BSM Connector uses adapter files to describe the commands that are needed to access and read log files on servers running different platforms of the UNIX operating system. These adapter files are written in plain text and are stored in the **<BSM Connector root directory>/templates.os** directory. For a list of the default UNIX adapters that are provided with BSM Connector, see "UNIX Adapters Provided with BSM Connector" below.

You can modify existing adapter files to adjust for specific system requirements in your environment, or you can create your own adapter files.

How to Add an Adapter

This task describes the steps involved in adding an adapter to specific versions of UNIX.

1. If the UNIX platform to which you want to add support is similar to one of the default BSM Connector-supported UNIX platforms, make a copy of the adapter file for that UNIX platform and use that as a starting point for your adapter.
2. Modify the adapter file to match the command line requirements for the UNIX platform to which you want BSM Connector to connect.
3. Save your adapter file to the **<BSM Connector root directory>/templates.os** directory. The filename must use the **.config** extension.
4. Restart the BSM Connector service.
5. Start the BSM Connector user interface for the BSM Connector server to which you have added the new adapter file, and log on.
6. Click the **Remote Server Management** toolbar button to open the **Remote Servers** dialog box.
7. Create a new UNIX remote server.
8. In the **Operating system** box, select the name of the UNIX adapter that you have created.
9. Click **OK**. BSM Connector uses the new adapter file to try and retrieve that applicable data from the remote server.

The amount of work required to modify a particular template depends on how different the new UNIX platform is from the supported UNIX platforms.

UNIX Adapters Provided with BSM Connector

The default UNIX adapters that are provided with BSM Connector, include:

Filename	Description
AIX.config	Adapter file for IBM AIX
CentOS.config	Adapter file for CentOS Linux
Digital.config	Adapter file for Digital Tru64 UNIX (Pre 4.x)
FreeBSD.config	Adapter file for FreeBSD 3.x

Filename	Description
HP.config / HP-UX.config	Adapter file for Hewlett-Packard HP/UX
HP64.config	Adapter file for Hewlett-Packard HP/UX 64-bit
ILO.config	Adapter file for Hewlett-Packard Integrated Lights-Out
Linux.config	Adapter file for Linux (Red Hat and others)
MacOSX.config	Adapter file for Apple MacIntosh OS X
NonStopOS.config	Adapter file for Hewlett-Packard NonStop Operating System
OPENSERVER.config	Adapter file for SCO OpenServer
RedHatEnterpriseLinux.config	Adapter file for Red Hat ES Linux
SCO.config	Adapter file for SCO UNIXWare
SGI.config	Adapter file for Silicon Graphics Irix
Sun.config / SunOS.config	Adapter file for Sun Microsystems Solaris
Tru64.config	Adapter file for Compaq Tru64 UNIX 5.x
Ubuntu.config	Adapter file for Ubuntu Linux

Adapter File Format

Each UNIX platform supported for remote access by BSM Connector has an adapter file in the **<BSM Connector root directory>/templates.os** directory. These files use BSM Connector's standard setting file format.

The first group of settings (those settings before the first # sign line) describe the platform:

```
id=yourPlatform
name=your Platform Name
```

The `id` is theBSM Connector internal ID for the OS. This ID must be unique, contain no spaces, and can be alphanumeric.

Tip: We recommend that you use the name of the adapter file as the ID name. For example, if the name of your adapter file is `linux.config`, your ID would be `linux`.

The `name` is the name you want displayed in the **Operating system** drop-down list when adding or editing remote servers.

The rest of the template file contains groups of settings representing a single command, separated by a line of # characters. For example, the following settings represent the command to display the last part of a log file:

```
id=tail
command=/usr/bin/tail -c +<bytes> <file>
```

where:

`id=tail` is the id that BSM Connector uses to look up a command. This must be one of the set of BSM Connector commands (see "Adapter Command List" below). This entry is case sensitive.

For example:

`command=/usr/bin/tail -c +<bytes> <file>` means that the `/usr/bin/tail -c +<bytes> <file>` command is run to display the last number of `<bytes>` of the file `<file>`.

Adapter Command List

BSM Connector requires settings for each the following commands to operate properly. Each command description requires an ID and a command, one or more fields to specify where the data is being read from, and optionally a set of modifiers that are used to filter the output of the command to eliminate certain sets of lines (such as header lines).

Where the variable `column` is used below, it means the number of the column in which the data appears, where columns are space delimited sets of data.

In addition, there are certain fields that can be optionally applied to any command description. For details, see "Optional Adapter Command Details" below.

This section includes:

- "Log File Processing" below
- "Optional Adapter Command Details" below

Log File Processing

ID	Description	Fields
fileExists	Checks that the log file exists.	**match**. The text to match in the log entries.
filesize	Returns the file size to ascertain if the file changed.	**size**. The number in the size column in the command output.
tail	Reads the file content for local file processing.	

Optional Adapter Command Details

The following fields can optionally be applied to any command description:

Process List with Details

ID	Description
startLine	The line number where the command starts looking for data.
endLine	The line number where the command ends looking for data.
skipLine	The pattern that if matched, skips the line.
matchLine	The pattern that if matched, looks for data in that line.

ID	Description
startMatch	The pattern that if matched, starts the command looking for data.
endMatch	The pattern that if matched, ends the command looking for data.
reverseLines	If true, the command output lines are reversed and read back to front. This is useful if there is data at the end of the command and it is too difficult to work out when to start reading.

If a field name has the format, `fieldnameColumnName=COLUMN`, the adapter searches the headers (first line) for `COLUMN` and records the columns containing the data, and then use those settings to read the `fieldname` field. This is useful where the width of the columns varies, and the data has spaces in it.

For example, to read the `my data` information from the following command output:

```
MEM NAME DESC12K my data some of my data
```

you would specify the name field in the command description as:

```
nameColumnName=NAME
```

The adapter reads the header line, finds `NAME`, and records where the previous column ends (`MEM` in this case) and where the specified column ends (`NAME`), and uses that to read, in this case, the text in character columns 6 through 22.

Regular Expressions

BSM Connector uses regular expressions to match log file content.

Regular expressions is a name given to a text parsing tool that was developed for use with scripting languages such as Awk and Perl, as well as several programming environments, such as Emacs, Visual C++, and Java. Regular expressions themselves are not a programming language. They do, however, make use of many special combinations of characters and symbols that often make them more difficult to interpret than some programming languages. The many different combinations of these special characters, known as metacharacters, make regular expressions a very powerful and flexible tool for parsing and isolating specific text within a larger body of text.

Including a regular expression in the **Log File Pattern** text box of a log file policy instructs BSM Connector to parse the text returned to the policy when it is run and look for content that satisfies the pattern defined by the regular expression. This document presents an overview of the syntax and metacharacters used in regular expressions for use in matching content for BSM Connector log file policies.

Defining a Regular Expression

Surrounding the regular expression with parentheses ( ) instructs BSM Connector to create pattern matching groups. BSM Connector retains what was matched between the parentheses and displays the text in the Sample Data tabs. You can then insert the pattern matching groups in event attributes, mappings, and rules. For more information about pattern matching groups, see "Pattern matching groups" on page 151. This is also very useful for troubleshooting your pattern. Parentheses are also used to limit alternations, as discussed below.

Generally, it is best to use an iterative approach when building regular expressions for content matching within BSM Connector. The following are some general steps and guidelines for developing regular expressions for content matches:

- Create a regular expression using literal characters to match a single sample of the data you want to check. For example, `(value: 1022.5)`.
- Iteratively replace literal characters with character classes and metacharacters to generalize the literal into a pattern. For example, the literal in the example above could be changed to: `(value:\s\d\d\d\d\.\d)` to match any four digits, a decimal point, and one more digit.
- Consider that the pattern of the data you want to match may vary. Adjust your pattern to match expected or possible variations in the target data. Continuing with the example used above, the expression `(value:\s\d\d\d\d\.\d)` might become `(value:\s[\d]{1, 8}\.[\d]{1,2})`. This pattern enables variation in the number of digits to the left of the decimal point and the number of digits to the right of the decimal point. It expects that there is a decimal point. See the following sections for more information about the character classes used here.
- Consider that the literal string or pattern you want to match may appear more than once in the content. Identify unique content that precedes the content you want to match, and add regular expression patterns to make sure that the expression matches that unique content before it tries to match the content you are trying to check. In the example used here, the pattern may match the first of several entries that have a similar `(value: numbers)` pattern. Adding a literal to the pattern, that matches some static content that delimits the particular data, can be used to be sure the match is made for the target data. For example, if the data you want to match is preceded by the text `Open Queries`, this literal can be added to the pattern, along with a pattern for any intervening content: `(Open Queries[\s\W]{1,5} value:\s[\d]{1, 8}\.[\d]{1,2})`.

Matching String Literals

Finding and matching an exact or literal string is the simplest form of pattern matching with regular expressions. In matching literals, regular expressions behave much as they do in search/replace in word processing applications. The example above matched the text Web site. The regular expression `(Buy Now)` succeeds if the text returned to the policy contains the characters Buy Now, including the space, in that order.

Note that regular expressions are, by default, case sensitive and literal. This means that the content must match the expression in case and order, including non-alphanumeric characters. For example, a regular expression of `(Website)`, without any modifiers, succeeds only if the content contains the string `Website` exactly but fails even if the content on the page is `website`, `WEBSITE`, or `Web site`. (In the last case the match fails because there is space between the two words but not in the regular expression.)

There are cases where you may want to literally match certain non-alphanumeric characters which are special "reserved" metacharacters used in regular expressions. Some of these metacharacters may conflict with important literals that you are trying to match with your regular expression. For example, the period or dot symbol (`.`), the asterisk (`*`), the dollar sign (`$`), and back slash (\) have special meanings within regular expressions. Because one of these characters may be a key part of a particular text pattern you are looking for, you must "escape" these characters in your regular expression so that the regular expression processing treats them as literal characters rather than interpreting them as special metacharacters. To force any character to be interpreted as a literal rather than a metacharacter, add a back slash in front of that character.

Example - Matching a Literal String

For example, if you wanted to find the string `4.99` on a Web page you might create a regular expression of `(4.99)`. While this matches the string `4.99,` it would also match strings like `4599` and `4Q99` because of the special meaning of the period character. To have the regular expression interpret the period as a literal, escape the period with a forward slash as follows: `(4\.99)`. You can add the back slash escape character in front of any character to force the regular expression processing to interpret the character following the back slash as a literal. In general, use this syntax whenever you want to match any punctuation mark or other non-alphanumeric character.

Using Alternation

Alternation enables you to construct either/or matches where you know that one of two or more strings should appear in the content. The alternation character is the vertical pipe symbol ("|").

The vertical pipe is used to separate the alternate strings in the expression. For example, the regular expression `((e-mail|e-mail|contact us))` succeeds if the content contains any one of the three strings separated by the vertical pipes. The parentheses are used here to delimit alternations. In this example, there are no patterns outside of the alternation that must be matched. In contrast, a regular expression might be written as `((e-mail|e-mail|contact) us)`. In this case, the match succeeds only when any of the three alternates enclosed in the parentheses is followed immediately by a single white space and the word us. This is more restrictive than the previous example, but also shows how the parentheses limit the alternation to the three words contained inside them. The match fails even if one or more of the alternates are found but the word "us" is not the next word.

Matching Patterns with Metacharacters

Often you may not know the exact text you need to match, or the text pattern may vary from one session or from one day to another. Regular expressions have a number of special metacharacters used to define patterns and match whole categories of characters. While matching literal alphanumeric characters seems trivial, part of the power of regular expressions is the ability to match non-alphanumeric characters as well. Because of this, it is important to keep in mind that your regular expressions need to account for the presence of non-alphanumeric characters in the content you are searching. This means that characters such as periods, commas, hyphens, quotation marks, and even white spaces, must be considered when constructing regular expressions.

This section contains the following topics:

- "Metacharacters Used in Regular Expressions" on the next page
- "Defining Character Classes" on the next page
- "Using Quantifiers" on page 221

Metacharacters Used in Regular Expressions

Metacharacter	Description
\s	Matches generic white space (that is, the Spacebar key). This metacharacter is particularly useful when combined with a quantifier to match varying numbers of white space positions that may occur between words that you are looking to match.
\S	Matches characters that are not white space. Note that the `\S` is capitalized as opposed to the small `\s` which is used to match white space.
.	This is the period or dot character. Generally, it matches all characters. Because BSM Connector considers the dot as a form of character class on its own, you should not include it inside the square brackets of a character class.
\n	Matches the linefeed or newline character.
\r	Matches the carriage return character.
\w	Matches non-white space word characters, the same as what is matched by character class [A-Za-z0-9_]. It is important to note that the \w metacharacter matches the underscore character but not other punctuation marks such as hyphens, commas, periods, and so forth.
\W	Matches characters other than those matched by \w (lowercase). This is particularly useful for matching punctuation marks and non-alphabetic characters, such as ~!@#$%^&*()+={[}]:; and including the linefeed character, carriage return, and white space. It does not match the underscore character, which is considered a word constituent matched by \w.
\d	Matches digits only. This is equivalent to the [0-9] character class.
\D	Matches non-numeric characters (what \d does not match) plus other characters. Similar to \W but also matches on alphabetic characters. In BSM Connector, this generally matches everything, including multiple lines, until it encounters a digit.
\b	Requires that the match have a word boundary (usually a white space) at the position indicated by the \b.
\B	Requires that the match not have a word boundary at the position indicated.

Defining Character Classes

An important and very useful regular expression construct is the character class. Character classes provide a set of characters that may be found in a particular position within a regular expression. Character classes may be used to define a range of characters to match a single position or, with the addition of a quantifier, may be used to universally match multiple characters and even complete lines of text.

You form character classes by enclosing any combination of characters and metacharacters in square brackets: [ ]. Character classes create an "any-or-all-of-these" group of characters that may be matched. Unlike literals and metacharacters outside character classes, the physical sequence of characters and metacharacters within a character class has no effect on the search or match sequence. For example, the class [ABC0123abc] matches the same content as [0123abcABC].

The hyphen is used to further streamline character classes to indicate a range of letters or numbers. For example, the class [0-9] includes all digits from zero to nine inclusive. The class [a-z] includes all lowercase letters from a to z. You can also create more restrictive classes with the hyphen, such as [e-tE-T], to match upper or lowercase letters from E to T, or [0-5] to match digits from zero to five only.

You can use the caret character (^) within a character class as a negation or to exclude certain characters from a content match.

Example Character Classes

Example	Description
[a-zA-Z]	This matches any alphabetic character, both upper case and lowercase, from the letter a to the letter z. To match more than one character, append a quantifier after the character class as described below.
[0-9]	This matches any digit from 0 to 9. To match more than one digit, append a quantifier after the character class as described below.
[0-9A-Za-z]	This matches any alphanumeric character, excluding the underscore.
[\w\s]	This matches any alphanumeric character, any white space, or both.

Using Quantifiers

Another set of metacharacters used in regular expressions provides character counting options. This adds a great deal of power and flexibility in content matching. Quantifiers are appended after the metacharacters and character classes described above to specify against which positions the preceding match character or metacharacter should be matched. For example, in the regular expression `((contact|about)\s+us)`, the metacharacter `\s` matches on a white space. The plus sign quantifier following the `\s` means that there must be at least one white space between the words contact (or about) and us.

The following table describes the quantifiers available for use in regular expressions. The Quantifier applies to the single character immediately preceding it. When used with character classes, the quantifier is placed outside the closing square bracket of the character class. For example: `[a-z]+` or `[0-9]*`.

Quantifier	Description
?	The question mark means the preceding character or character class may appear once, but is optional and not required to appear in the position indicated.

Quantifier	Description
*****	The asterisk requires that any number of the preceding character or character class appear in the designated position. This includes zero or more matches. **Note:** Care must be used in combining this quantifier with the dot (.) metacharacter or a character class including the \W metacharacter, as these are likely to "grab" more content than anticipated and cause the regular expression engine to use up all of the available CPU time on the BSM Connector server.
+	The plus sign requires that the preceding character or character class appear at least once.
{min, max}	Using curly braces creates a quantifier range. The range enumerator digits are separated by commas. This construct requires that the preceding character or character class appear at least as many times as specified by the **min** enumerator up to but no more than the value of the **max** enumerator. The match succeeds as long as there are at least as many matches as specified by the **min** enumerator. However, the matching continues up to the number of times specified by the **max** enumerator or until no more matches are found.

Search Mode Modifiers

Regular expressions used in BSM Connector may include optional modifiers outside of the slashes used to delimit the expression. Modifiers after the ending slash affect the way the matching is performed. For example, regular expression of `(website)/i` with the `i` search modifier added makes the match content search insensitive to upper and lowercase letters. This would match either `website`, `Website`, `WEBSite`, or even `WEBSITE`.

With the exception of the `i` modifier, some metacharacters and character classes can override search mode modifiers. In particular, the dot (.) and the `\W` metacharacters can override the `m` and `s` modifiers, matching content across multiple lines despite the modifier.

More than one modifier can be added by concatenating them together after the closing slash of the regular expression. For example: `(matchpattern)/ic` combines both the i and c modifiers.

Regular Expression Match Mode Modifiers

Mode Modifier	Description
/i	Ignore case mode. This makes the search insensitive to upper case and lowercase letters. This is a useful option especially when searching for matches in the text content of Web pages.
/c	The matched pattern must NOT appear anywhere in content that is being searched. This is a "complement" match, returning an error if the pattern IS found, and succeeding if the pattern is NOT found.

Mode Modifier	Description
/m	Match across multiple lines WITHOUT ignoring intervening carriage returns and linefeeds. With this modifier you may still need to account for possible linefeeds and carriage returns with a character class such as [\w\W]* or [\s\S\n\r]*. The .* does not match carriage returns or linefeed characters with this modifier.
/s	Consider the content as being on a single line, ignoring intervening carriage returns and linefeed characters. With this modifier, both the [\w\W]* character class and the .* pattern match across linefeeds and carriage returns.

BSM Connector Date Variables

BSM Connector uses specially defined variables to create expressions that match the current date or time. These variables can be used in the log file pattern field to find date-coded content. The General Date Variables are useful for matching portions of date formats. The Language/Country Specific Date Variables enable you to automatically extend the language used for month names and weekday names to specific countries, based on ISO codes.

This section contains the following topics:

- "General Date Variables" below
- "Language/Country Specific Date Variables" on the next page
- "Special Substitution for File Path" on page 225

General Date Variables

The following table lists the general variables:

Variable	Range of Values
$hour$	0 - 23
$minute$	0 - 59
$month$	1 - 12
$day$	1 - 31
$year$	1000 - 9999
$shortYear$	00 - 99
$weekdayName$	Sun - Sat
$fullWeekdayName$	Sunday - Saturday
$0hour$	00 - 23
$0minute$	00 - 59

Variable	Range of Values
$0day$	01 - 31 (two-digit day format)
$0month$	01 - 12 (two-digit month format)
$monthName$	Jan - Dec (three-letter month format in English)
$fullMonthName$	January - December
$ticks$	milliseconds since midnight, January 1, 1970

For example, if the content match search expression was defined as:

```
Updated on $0month$\/$0day$\/$shortYear$
```

and the content returned by the request includes the string:

```
Updated on 06/01/98
```

then the expression would match when the policy is run on June 1, 1998. The match fails if the content returned does not contain a string matching the current system date or if the date format is different than the format specified.

If you want the time to be before or after the current time, you can add a **$offsetMinutes=mmmm$** to the expression, and this offsets the current time by **mmmm** minutes (negative numbers are permitted for going backwards in time) before doing the substitutions.

For example, if the current day is June 1, 2007, and the search expression is:

```
$offsetMinutes=1440$Updated on $0month$\/$0day$\/$shortYear$
```

the content string that would match would be:

```
Updated on 06/02/07
```

Note: The date is one day ahead of the system date.

Language/Country Specific Date Variables

The following table lists the BSM Connector special variables for use with international day and month name matching. The characters LL and CC are placeholders for two-letter ISO 639 language code characters and two-letter ISO 3166 country code characters (see the notes below the table for more details).

Variable	Range of Values
$weekdayName_LL_CC$	Abbreviated weekday names for the language (LL) and country (CC) specified (see notes below).
$fullWeekdayName_LL_CC$	Full weekday names for the language (LL) and country (CC) specified.

Variable	Range of Values
$monthName_LL_CC$	Abbreviated month names for the language (LL) and country (CC) specified.
$fullMonthName_LL_CC$	Full month names for the language (LL) and country (CC) specified.

CC - an uppercase 2-character ISO-3166 country code. Examples are: DE for Germany, FR for France, CN for China, JP for Japan, BR for Brazil. You can find a full list of these codes at a number of Internet sites, such as: http://www.iso.org/iso/country_codes/iso_3166_code_lists/country_names_and_code_elements.htm.

LL - a lowercase 2-character ISO-639 language code. Examples are: de for German, fr for French, zh for Chinese, ja for Japanese, pt for Portuguese. You can find a full list of these codes at a number of Internet sites, such as: http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt or http://www.dsv.su.se/~jpalme/ietf/language-codes.html.

For example, if the content match expression was defined as:

```
$fullWeekdayName_fr_FR$
```

and the content returned by the request includes the string:

```
mercredi
```

then this expression would match when the policy was run on Wednesday.

If you are not concerned with the country-specific language variations, it is possible to use any of the above variables without including the country code. For example:

```
$fullWeekdayName_fr$
```

could be used to match the same content as `/$fullWeekdayName_fr_FR$/`.

Special Substitution for File Path

BSM Connector Date Variables are useful for matching content as part of a regular expression. The date variables can also be used as a special substitution to dynamically create file paths for log file policies. This is useful for accessing date-coded files and directories where the file path is updated automatically based on system date information. BSM Connector is an example of an application that creates date-coded log files. The log file names include some form of the year, month, and day as part of the file name, such as `File2001_05_01.log,` where the year, month, and date are included.

Based on this example, a new file is created each day. Checking the creation, size, or content of the current days file would normally require the file path of the log file policy to be manually changed each day. Using the BSM Connector date variables and special substitution, BSM Connector can automatically update the file path to the current day's log file. By knowing the pattern used in naming the files, you can construct a special substitution string similar to a regular expression that substitutes portions of the system date properties into the file path.

For example if the absolute file path to the current day's log file in a log file policy is:

```
D:/Production/Webapps/Logs/File2001_05_01.log
```

the log file for the following day would be:

```
D:/Production/Webapps/Logs/File2001_05_02.log
```

You can construct a special substitution expression to automatically update the file path used by the policy, with the following syntax:

```
s/D:\/Production\/Webapps\/Logs\/File$year$_$0month$_$0day$.log/
```

The substitution requires that the expression start with a lower-case `s` and that the expression is enclosed by forward slashes `/.../`. Forward slashes that are part of the file path must be escaped by adding the back slash (\) character as shown. The BSM Connector date variables are separated by the underscore character literals. BSM Connector checks the system time properties each time the policy runs and substitutes with applicable values into the file path before accessing the file.

While the special substitution syntax is similar in syntax to the substitution syntax used in regular expressions, they are not the same. While all of the BSM Connector date variables can be used in match content regular expressions, the special substitution discussed here can not be used as part of a match content expression.

Examples for Log File Policies

Log file policies check for entries in files created by other applications. These files may be data files created by a third-party application or they may be logs created by a custom system specially designed for your environment. Where the logs or files are written with a known, predictable format, BSM Connector can be configured to regularly check the files for new entries and match on specific content strings. The following are several examples of log file entries and simple regular expression patterns that can be used to check the entries. You can use these examples or modify them to work with a specific case.

> **Note:** All regular expressions must be entered on a single line in BSM Connector. Some of the examples below may break across more than one line to fit on this page.

This section contains the following topics:

- "Searching Paths for Log Files" below
- "Matching Comma-Separated Values" on the next page
- "Matching Space Separated Values" on page 228
- "Matching and Retaining the Numbers in a Line of Text and Numbers" on page 228
- "Matching Integers and Floating-point Numbers (Positive or Negative)" on page 229
- "Matching Date and Time-Coded Log Entries" on page 229

Searching Paths for Log Files

UNIX and Windows operating systems treat the case ("N" and "n") of file names in incompatible ways. Windows operating systems are case insenstive which means that when a file is being searched, its case is ignored. UNIX operating systems are case sensitive which means that the case of a name is significant at all times. To avoid log file errors when using regular expressions to search for path names on UNIX operating systems, you should use markers to change the character case in the path expression.

Marker	Description
$L	Enables changing characters between the $L marker and the $E marker to lowercase.
$U	Enables changing characters between the $U marker and the $E marker to upper case.
$E	The end marker used for changing character case.

Example:

If you define the following path expression:

```
s/\/tmp\/logs\/arcv.log.$weekdayName$/
```

for the `/tmp/logs/arcv.log.tue` log file on a Linux machine, you get a log file error because BSM Connector tries to find `tmp/logs/arcv.log.Tue`, and Linux is case sensitive.

To resolve this problem, define the path expression as follows:

```
s/\/tmp\/logs\/arcv.log.$L$weekdayName$$E/
```

The policy converts the characters between `$L` and `$E` to lowercase, `/tmp/logs/arcv.log.tue`.

Conversely, use `$U` and `$E` to enable BSM Connector to change the characters between the markers to upper case. For example, if you define the path expression:

```
s/\/tmp\/logs\/arcv.log.$L$weekdayName$$E/
```

the policy converts the path to `/tmp/logs/arcv.log.TUE`.

You can use `$L` and `$U` multiple times in a path expression, and you can use them both in the same expression.

For example:

```
s/\/tmp\/logs-$L$weekdayName$$E\/arcv.log.$U$weekdayName$$E/
```

converts the path to `/tmp/logs-tue/arcv.log.TUE`

```
s/\/tmp.$L$monthName$$E\/logs-
$L$weekdayName$$E\/arcv.log.$U$weekdayName$$E/
```

converts the path to `/tmp.mar/logs-tue/arcv.log.TUE`

Matching Comma-Separated Values

The following is an example of log file entries that are comma-separated strings of digits and letters:

```
new,open,changed,12,alerts
new,open,changed,13,alerts
new,open,changed,13,alerts
new,open,changed,14,alerts
```

A regular expression to match on log file entries that are comma-separated strings of digits and letters.

```
([\w\d]+,[\w\d]+,[\w\d]+,[\w\d]+,[\w\d]+)[\n\r]?
```

Note: If the file entries include punctuation marks such as an underscore or a colon, add that character explicitly to the `[\w\d]` class pattern. For example, to include a colon character, change each of the `[\w\d]` patterns to `[\w\d:]`.

Matching Space Separated Values

The following is an example of log file entries that are a sequence of strings and digits separated by spaces:

```
requests 12 succeeded 12 failed
requests 12 succeeded 12 failed
requests 11 succeeded 11 failed
requests 12 succeeded 12 failed
requests 10 succeeded 10 failed
```

The following is a regular expression to match on log file entries that are a sequence of strings and digits separated by spaces.

```
([\w\d]+\s+[\w\d]+\s+[\w\d]+\s+[\w\d]+\s+[\w\d]+)[\n\r]?
```

Note: The use of the + character forces the match to include the number of sequences per line included in the match pattern: in this example, five word or number sequences per line of the log file. If the sequences include punctuation marks such as an underscore or colon, add that character explicitly to the `[\w\d]` class pattern. For example, to include a colon character, change each of the `[\w\d]` patterns to `[\w\d:]`.

Matching and Retaining the Numbers in a Line of Text and Numbers

The following is an example of log file entries that are comma separated strings that combine digits and letters:

```
request handle number 12.56, series 17.5, sequence reported 97.45,
15.95 and 19.51
request handle number 15.96, series 27.5, sequence reported 107.45,
25.95 and 19.52
request handle number 11.06, series 36.5, system codes 9.45, 35.95
and 19.53
log reference number 12.30, series 17.5, channel reset values
100.45, 45.95 and 19.54
```

The following is a regular expression to match on log file entries that are comma-separated strings that combine digits and letters and retain the decimal numeric data:

```
[,\D\s]+(\d+\.\d+)[,\D\s]+(\d+\.\d+)[,\D\s]+(\d+\.\d+)[,\D\s]+
(\d+\.\d+)[,\D\s]+(\d+\.\d+)[\n\r]?
```

> **Note:** If the file entries include punctuation marks such as an underscore or colon, add that character explicitly to the `[,\D\s]` class pattern. For example, to include a colon character that appears embedded in the text sequences, change each of the `[,\D\s]` patterns to `[,:\D\s]`.

Matching Integers and Floating-point Numbers (Positive or Negative)

The following is an example of log file entries that are a sequence of integers and floating point numbers that may be negative or positive:

```
12.1987 -71 -199.1 145 -1.00716
13.2987 -72 -199.2 245 -1.00726
14.3987 -73 -199.3 345 -1.00736
15.4987 -74 -199.4 445 -1.00746
```

The following is a regular expression to match on log file entries that are a sequence of 5 integers and floating point numbers that may be negative or positive. The numbers in each entry must be separated by one or more spaces.

```
(-?\d+\.?\d{0,})[\s]+(-?\d+\.?\d{0,})[\s]+(-?\d+\.?\d{0,})[\s]+
(-?\d+\.?\d{0,})[\s]+(-?\d+\.?\d{0,})[\n\r]?
```

Matching Date and Time-Coded Log Entries

Many log files include some form of date and time data with each entry. The following is an example of log file entries that include date and time information together with string data separated by commas:

```
20/04/2003 14:29:22,ERROR,request failed
20/04/2003 14:31:09,INFO,system check complete
20/04/2003 14:35:46,INFO,new record created
```

The following is a regular expression to match on log file entries that are date- and time-coded followed by comma-separated strings of letters and digits. This example uses the BSM Connector date variables to match only on entries that were created on the same day, month, and year as indicated by the system clock of the server where BSM Connector is running.

```
($0day$\/$0month$\/$year$\s+\d+:\d+:\d+,[\w\s]+,[\w\s]+)
```

The following example uses the BSM Connector date variables to match on a more restricted set of entries that were created on the same day, month, year, and within the same hour as indicated by the system clock of the server on which BSM Connector is running.

```
($0day$\/$0month$\/$year$\s+$0hour$:\d+:\d+,[\w\s]+,[\w\s]+)
```

Problems Working with Regular Expressions

This section contains problems encountered when working with regular expressions.

This section contains the following topics:

- "Using the .* construct presents a very large number of possible matches on any page of content" below
- "Regular expression match succeeds as soon as the minimum match requested is satisfied" below
- "Forgetting to account for non-alphanumeric content" below
- "Use of excessive metacharacters can be problematic" below
- "Example Regular Expression Syntax" on the next page

Using the .* construct presents a very large number of possible matches on any page of content

The use of the .* construct is known to cause the regular expression-matching engine used by BSM Connector to take over all available CPU cycles on the BSM Connector server. If this occurs, BSM Connector is unable to function and must be restarted each time the log file policy with the offending regular expression is run, until the expression has been corrected.

Regular expression match succeeds as soon as the minimum match requested is satisfied

After a match is made, no further matching is performed. Therefore, regular expressions are not well suited to count the number of occurrences of a repeating text pattern.

Forgetting to account for non-alphanumeric content

Regular expressions need to be written to account for all of the characters that are and may be present. This includes white space, linefeed, and carriage returns. This is not normally a problem when matching a single-word literal. It can be a challenge when you need to create a match of several words separated by unknown amounts of white space and other non-alphanumeric characters and possibly span more than one line. The `[\s\n\r]+` character class can be useful between words used in the expression. Always check the format of the content you are trying to match to look for patterns and special characters, such as periods, commas, and hyphens, that may cause a seemingly simple match to fail.

Use of excessive metacharacters can be problematic

In some cases, overly generous quantifiers combined with the . or `\W` metacharacters can grab content that you were intending to match with a literal string elsewhere in your regular expression resulting in a match failure. For example, the following might be used to match the log file line: `User "([\W\w\s]*)"`. When the policy performs the check for this regular expression, however, the match grabs the first occurrence of the pattern User "... and continues matching multiple lines of text up to the last quotation mark found in the file. Without some other unique ending delimiter, the `[\W\w\s]*` class and quantifier combination is too excessive. A more successful syntax that narrows the class of expected characters would be: `User "?([:\/\w\s\d\.]*)"?`

Example Regular Expression Syntax

The following are some examples of syntax for use in regular expressions:

Example Expression	Description
`CUSTID\s?=\s?([A-Z0-9]{20,48})`	This example matches an ID string that is made of 20 or more digits and upper-case letters with no spaces or other non-alphanumeric characters. The `\s?` construct permits a white space on either side of the equals sign. Using the parentheses around the character class instructs BSM Connector to retain this value (up to the maximum of 48 characters) as a pattern matching group and the matched value is displayed in the Sample Data tabs.
`"[^"]*"`	This example matches text sequences that are contained between quotation marks. Note the use of the negation caret (`^`) to define a character class of all characters other than the quotation mark.

As with programming and scripting languages, there is almost always more than one way to construct a regular expression to accomplish a particular match. There is not one right way to build regular expressions. You should plan to test and modify regular expressions as necessary until you get the results you need.

Troubleshooting Log File Policies

This section describes troubleshooting and limitations when working with log file policies.

This section includes:

- "Debugging errors" below
- "Log file policy file rolling" on page 233
- "Metrics troubleshooting" on page 233
- "Topology troubleshooting" on page 234

Debugging errors

- Check for errors in the following files:

  <BSM Connector root directory>\logs\error.log

  <BSM Connector root directory>\logs\RunMonitor.log

  <BSM Connector root directory>\logs\bac_integration\bac_integration.log.

- You can modify the level and type of information reported to the log files by changing the log file settings in the **<BSM Connector root directory>\conf\core\Tools\log4j\PlainJava\log4j.properties** file. You can instruct the logging mechanism to:

- Report logged information in less or greater detail than is reported by default.
- Log all metrics sent by metrics integration policies to BSM.
- Log all received data from third-party systems.

To modify log settings:

a. Open the **log4j.properties** file in a text editor.

b. To specify that metrics sent by metrics integration policies to BSM be logged:

   i. Locate the following lines in the file:

   ```
   log4j.category.EmsSamplePrinter=${loglevel},
   integration.appender
   log4j.additivity.EmsSamplePrinter=false
   ```

   ii. Change the argument of **log4j.category.EmsSamplePrinter**from **${loglevel}** to **DEBUG**, as follows:

   ```
   log4j.category.EmsSamplePrinter=DEBUG, integration.appender
   ```

   iii. Save the file. It may take a few seconds for the changes to take effect.

   The results are logged to the **bac_integration.log** file.

c. To specify that all received data from third-party systems be logged:

   i. Locate the following lines in the file:

   ```
   log4j.category.EmsEventPrinter=${loglevel}, monitors.appender
   log4j.additivity.EmsEventPrinter=false
   ```

   ii. Change the argument of **log4j.category.EmsEventPrinter** from **${loglevel}** to **DEBUG**, as follows:

   ```
   log4j.category.EmsEventPrinter=DEBUG, monitors.appender
   ```

   iii. Save the file. It may take a few seconds for the changes to take effect.

   The results are logged to the **RunMonitor.log** file.

> **Tip:** After you have determined the cause of the problem, we recommend that you set log levels to their default settings so as not to overload the system.

- If metrics are created and sent from BSM Connector, but the data is not seen in BSM Service Health, Event Log, or SiteScope reports, look in **<BSM root directory>\log\mercury_wde\wdeIgnoredSamples.log** to make sure the metrics were not dropped due to missing fields or values.
- Change the logging level for Service Health to verify that Service Health received the samples. Open the following file on the Gateway Server machine: **<BSM root directory>\conf\core\tools\log4j\mercury_wde\wde.properties**

  Change the log level parameter to DEBUG in the following lines:

  - `log4j.category.com.mercury.am.platform.wde.decode.IgnoredSamplesLogger=${loglevel}, IgnoredSamples.appender`

  - `log4j.category.com.mercury.am.platform.wde.publish_SamplePublisher Samples=${loglevel}, PublishedSamples.appender`

  Look at the corresponding log files:

  - **<BSM root directory>\logs\mercury_wde\wdeIgnoredSamples.log**
  - **<BSM root directory>\logs\mercury_wde\wdePublishedSamples.log**

Log file policy file rolling

A log file policy reads the following XML file:

Windows: `%OvDataDir%\tmp\<`*`policyID`*`>.xml`

Linux: `/var/opt/OV/tmp/<`*`policyID`*`>.xml`

The file contains the results of the pattern matching (that is, the pattern matching groups) in an XML file format that database policies can evaluate. The file exists only for policies that integrate events.

To prevent the file from becoming too large, a new version of the file is created when its size reaches 100000 KB. The original file is renamed and stored as a backup file. By default, up to three backup files can be created.

> **Caution:** When the XML file rolls, that is, a new version is created, but the policy has not completed reading all entries, the unread entries are lost and no events are created.

The following defaults are set:

- **Maximum file size:** 100000 KB

  > **Tip:** If not all expected events arrive in BSM, increase the maximum file size to 1000000 KB.

- **Maximum number of backup files:** 3

To change the defaults:

1. Open the following file:

   ```
   <BSM Connector root directory>/groups/master.config
   ```

2. Change the following values as required:

   ```
   _eventRollingMaxFileSizeInKB=100000
   ```

   ```
   _eventRollingMaxBackup=3
   ```

3. Restart the BSM Connector server:

   Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

Metrics troubleshooting

See "Metrics Troubleshooting" on page 87 for more information on troubleshooting and limitations when BSM Connector sends metrics to BSM.

Topology troubleshooting

See "Topology Troubleshooting and Limitations" on page 338 for more information on troubleshooting and limitations when BSM Connector is enabled to report CIs and CIs relationships topology to BSM.

Chapter 6

Integrating Data from the Open Message Interface into BSM

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 BSM (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 your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

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 240.

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 241.

5. In the **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 242.

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 245.

7. Click **OK** 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 238 and "opcmsg C API" on page 240.

opcmsg Command Line Tool

The command opcmsg generates a message that BSM Connector open message interface policies evaluate.

The opcmsg command-line interface 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 BSM to identify the node and CI associated with the event.

Option	Description
[-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" on the next page
- "Configuring Rules in Open Message Interface Policies" on page 242
- "Configuring Options in Open Message Interface Policies" on page 245

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 235

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 423.

Configuring Event Defaults in Open Message Interface Policies

The Events 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 severity and category.

   > **Tip:** After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data from the Open Message Interface" on page 235

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Advanced Tab" on page 400
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Rules in Open Message Interface Policies

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the source. The settings enable you to configure the event that BSM Connector sends to BSM.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

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 BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to BSM.

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:** You can use standard BSM Connector pattern-matching rules when matching values. Select the matches operator and click ▶ in the Operand field to open the pattern matching expression toolbox. The toolbox displays the following:
      >
      > - **Pattern Matching Expressions.** Click an expression to insert it in the Operand field.
      > - **Variable Bindings Options.** Variable bindings options include case sensitivity

> and field separators for the rule. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used. See also "Configuring Options in Open Message Interface Policies" on the next page.

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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example "`<$MSG_NODE>`".

Related tasks

- "How to Collect Event Data from the Open Message Interface" on page 235

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Tab - Open Message Interface Policy Rules Only" on page 404
- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

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 235

UI Descriptions

For a description of the Options page, see "Options Page (Events Only)" on page 417.

Chapter 7

Scheduling Tasks

Scheduled task policies enable you to schedule commands or scripts to run on the BSM Connector system, and will send an event to BSM 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 your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

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 250.

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 251.

7. Click **OK** 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 250
- "Configuring Events in Scheduled Task Policies" on page 251

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 423.

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 **Upload** 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 254 .)

  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 **Upload** 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 254 .)

  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 247

UI Descriptions

For a description of the Task page, see "Task Page (Scheduled Task Policies)" on page 441.

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.
- **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 247

UI Descriptions

For a description of the Schedule page, see "Schedule Page - Scheduled Task Policies" on page 429.

Configuring Events in Scheduled Task Policies

Scheduled task policies can send a notification event to BSM 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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Schedule Tasks" on page 247

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 440
- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

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 from the command line.

> **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" below
- "Rule Object" on page 254
- "ConsoleMessage Object" on page 254
- "ExecuteCommand Object" on page 259

Policy Object

This object is used to access the attributes of a policy.

Policy Method:	`CreateObject`
Parameter:	*progID* (string of format: `[Vendor.]Component[.Version]`)

Policy Method:	**`CreateObject`**
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 the Event Browser in response to the success or failure of a scheduled task.

Policy Method:	**`ExecuteEx`**
Parameter:	*command* (string)
Return Type:	VB Script: BSTR Perl: string
VB Script Syntax:	`Policy.ExecuteEx("`*`command`*`")`
Perl Syntax:	`$Policy->ExecuteEx("`*`command`*`");`

Policy Method:	`ExecuteEx`

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 the Event Browser. Multiple uses of the Send method are supported. The same script can then send multiple events to BSM 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 of the event sent to the Event Browser.

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 of the event sent to the Event Browser.

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 that is sent to the Event Browser.

ConsoleMessage Method:	**`Severity`**
Parameter:	*severity* (valid strings are: Unknown|Normal|Warning|Minor|Major|Critical)
Return Type:	VB Script: void Perl: void
VB Script Syntax:	`ConsoleMessage.Severity = "`*`severity`*`"`
Perl Syntax:	`$ConsoleMessage->Severity("`*`severity`*`");`

ConsoleMessage Method:	**`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 of the event sent to the Event Browser. 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 of the event sent to the Event Browser. 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 of the event sent to the Event Browser.

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 in the Event Browser.

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 BSM 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)
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();`

ExecuteCommand Method:	`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 the command 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 8

Integrating SNMP Trap Data into BSM

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 BSM 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 BSM (for presentation in Operations Management, Service Health, Service Level Management, reports, or all).

Event data that is forwarded to BSM 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 BSM.

Note: You can use this policy type to forward information from HP Network Node Manager i (NNMi) to BSM. 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.

To configure HP Operations Agent to subscribe to the Microsoft SNMP trap service:

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.

How to Collect Event Data from SNMP Traps

This task describes how to collect event data from SNMP traps.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click ✱ in the toolbar, then click **Event >** **SNMP**. 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 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 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 Policies" on page 266.

6. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in SNMP Policies" on page 269.

7. Click **OK** 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 Policy Properties" below
- "Configuring Event Defaults in SNMP Policies" below
- "Configuring Rules in SNMP Policies" on the next page
- "Configuring Options in SNMP Policies" on page 269

Configuring SNMP 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**. 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 423.

Configuring Event Defaults in SNMP Policies

The Events 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**. 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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data from SNMP Traps" on page 264

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Advanced Tab" on page 400
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Rules in SNMP Policies

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the source. The settings enable you to configure the event that BSM Connector sends to BSM.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

To access

In the BSM Connector user interface, click * in the toolbar, then click **Event >** **SNMP**. 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 BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to BSM.

Tasks

How to configure rules in SNMP trap policies

This task describes how configure policy rules.

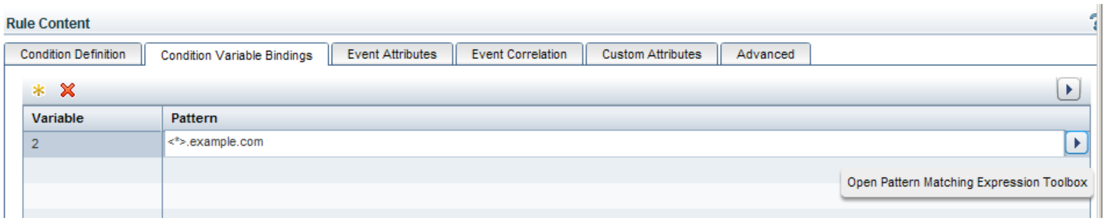
1. In the **Policy Rules** section, click * in the toolbar and select the rule type. Then type a description for the rule. After a rule has been added, you can change the rule type by clicking the current rule type in the list of rules and selecting another rule type from the drop-down list.

   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 Definition** 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`:



Variable	Pattern
2	<*>.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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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.

8. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data from SNMP Traps" on page 264

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Definition Tab - SNMP Trap Policy Rules Only" on page 406
- "Condition Variable Bindings Tab - SNMP Trap Policy Rules Only" on page 407
- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400

- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Options in SNMP 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**. 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 Policies" above.

Related tasks

- "How to Collect Event Data from SNMP Traps" on page 264

UI Descriptions

For a description of the Options page, see "Options Page (Events Only)" on page 417.

Chapter 9

Integrating Data Through the Web Service Listener

Web service policies enable you to collect data from third-party systems through the BSM Connector Web service listener. Event, metrics, and topology entry points into BSM are published for external systems to use.

BSM Connector supplies a WSDL file that can be used to create the client code. The client code reports the event, metrics, and topology data to BSM Connector. The client has several ways to report data to BSM Connector:

- report one metric or an array of metrics
- report data - an array of key value pairs (representing an event, for example)

What Data Is Collected

Web service policies collect data that is extracted from any message received by the BSM Connector Web service listener. The policies process the data by applying policy defaults and rules, and then send the data to BSM.

Data can also be mapped to a topology to forward data to the correct CI hierarchy in BSM. You can configure topology settings for the policy by selecting one of the predefined scripts, or configuring your own topology script during policy creation.

How to Collect Event Data Through the Web Service Listener

This task describes how to collect event data and optionally topology data through the BSM ConnectorWeb service listener.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Event >** **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 Web Service Listener Policy Properties" on page 274.

4. In the **Source** page, define the system ID that connects the messages from the Web service client with the Web service listener policy.

   For details, see "Configuring the Data Source in Web Service Listener Policies" on page 275.

5. In the **Mappings** page, configure the default mappings of Web service <key> elements to custom variables.

   For details, see "Configuring Mappings in Web Service Listener Policies " on page 283.

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 Web Service Listener Policies" on page 285.

7. In the **Rules** page, define what the policy should do in response to a specific Web service message.

   For details, see "Configuring Event Rules in Web Service Listener Policies" on page 290.

8. In the **Options** page, configure several policy behaviors (for example, pattern matching options).

   For details, see "Configuring Options in Web Service Listener Policies" on page 296.

9. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that represents your third-party system.

   For details, see "Configuring Topology Scripts in Web Service Listener Policies" on page 297.

10. Click **OK** to save the policy and close the editor.

11. *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 Web Service Listener

This task describes how to collect metrics and optionally topology data through the BSM Connector Web service listener.

For examples and end-to-end workflow information on collecting metrics data with or without topology, see "Collecting and Viewing Metrics Data" on page 53.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click * in the toolbar. Then click **Metrics >** 🌐 **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 Web Service Listener Policy Properties" on the next page.

4. In the **Source** page, define the system ID that connects the messages from the Web service client with the Web service listener policy.

   For details, see "Configuring the Data Source in Web Service Listener Policies" on page 275.

5. *Optional.* In the **Defaults** page,assign default values to the metric attributes.

   For details, see "Configuring Metrics Defaults in Web Service Listener Policies" on page 287.

6. In the **Rules** page, define what the policy should do in response to a specific metric.

   For details, see "Configuring Metrics Rules in Web Service Listener Policies" on page 293.

7. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that represents your third-party system.

   For details, see "Configuring Topology Scripts in Web Service Listener Policies" on page 297.

8. Click **OK** 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 Topology Data Through the Web Service Listener

You can have BSM Connector report only the topology sent to the BSM Connector Web service listener.

This task describes how to collect topology data through the Web service listener.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Topology >** **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 Web Service Listener Policy Properties" on the next page.

4. In the **Source** page, define the system ID that connects the messages from the Web service client with the Web service listener policy.

   For details, see "Configuring the Data Source in Web Service Listener Policies" on page 275.

5. In the **Topology** page, select or create a script that creates a topology in BSM's RTSM that represents your third-party system.

   For details, see "Configuring Topology Scripts in Web Service Listener Policies" on page 297.

6. Click **OK** 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.

Web Service Listener Policy User Interface

Web service listener policies display different pages depending on the type of data they are integrating. Choose the type of data you want to integrate and then complete the following tasks:

Event data (optionally with topology data)

- "Configuring Web Service Listener Policy Properties" below
- "Configuring the Data Source in Web Service Listener Policies" on the next page
- "Configuring Mappings in Web Service Listener Policies " on page 283
- "Configuring Event Defaults in Web Service Listener Policies" on page 285
- "Configuring Event Rules in Web Service Listener Policies" on page 290
- "Configuring Options in Web Service Listener Policies" on page 296
- "Configuring Topology Scripts in Web Service Listener Policies" on page 297

Metrics data (optionally with topology data)

- "Configuring Web Service Listener Policy Properties" below
- "Configuring the Data Source in Web Service Listener Policies" on the next page
- "Configuring Metrics Defaults in Web Service Listener Policies" on page 287
- "Configuring Metrics Rules in Web Service Listener Policies" on page 293
- "Configuring Topology Scripts in Web Service Listener Policies" on page 297

Topology data only

- "Configuring Web Service Listener Policy Properties" below
- "Configuring the Data Source in Web Service Listener Policies" on the next page
- "Configuring Topology Scripts in Web Service Listener Policies" on page 297

Configuring 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 >** **Web Service Listener**.
- In the BSM Connector user interface, click * in the toolbar. Then click **Metrics >** **Web Service Listener**.
- In the BSM Connector user interface, click * in the toolbar. Then click **Topology >** **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 Web Service Listener Policy Properties" on the previous page.

Related tasks

- "How to Collect Event Data Through the Web Service Listener" on page 271
- "How to Collect Metrics Data Through the Web Service Listener" on page 272
- "How to Collect Topology Data Through the Web Service Listener" on page 273

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 423.

Configuring the Data Source in Web Service Listener Policies

The source page of the Web service listener policy editor enables you to set up the system ID.

To access

- In the BSM Connector user interface, click * in the toolbar. Then click **Event >** **Web Service Listener**.
- In the BSM Connector user interface, click * in the toolbar. Then click **Metrics >** **Web Service Listener**.
- In the BSM Connector user interface, click * in the toolbar. Then click **Topology >** **Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Source** to open the policy Source page.

Learn More

Reporting methods

Choose one of the following reporting methods depending on the type of data you want to send:

- ### For events, metrics, or topology: configure the reportData method

  > **Tip:** For sending metrics data, it is recommended that you use the **reportMetricObject** and **reportMetricsArray** methods. The reportMetricsArray method enables you to submit metrics in bulk instead of one by one. This is a more efficient method than the reportMetricObject method because it causes less network traffic.

  You can configure the **reportData** method while creating your SOAP message to send to BSM Connector.This method contains a data structure that gets an array of key-value objects (see `DataMessage` object below).

  Enter the following service request:

  ```
  <wsdl:message name="reportDataRequest">
    <wsdl:part name="systemId" type="xsd:string" />
    <wsdl:part name="data" type="impl:ArrayOf_tns1_DataMessage" />
  </wsdl:message>
  ```

  where:

  "systemId" is the unique text ID for the Web service listener policy.

  "data" is an array of any data type named `DataMessage` that contains key and value strings:

  ```
  <complexType name="DataMessage">
      <complexContent>
         <extension base="tns1:AbstractMessage">
            <sequence>
              <element name="key" nillable="true" type="soapenc:string" />
              <element name="value" nillable="true" type="soapenc:string" />
            </sequence>
         </extension>
      </complexContent>
  </complexType>
  ```

  The service success response is:

  ```
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <soapenv:Body>
        <ns1:reportMetricObjectResponse
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:ns1="http://webservice.soa.monitors.sitescope.mercury.com">
           <reportMetricObjectReturn xsi:type="xsd:int">1</reportMetricObjectReturn>
        </ns1:reportMetricObjectResponse>
     </soapenv:Body>
  </soapenv:Envelope>
  ```

Example - Sending a common event using the reportData request

The question marks in the following example represent string values.

Example:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:web="http://webservice.soa.monitors.sitescope.mercury.com"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soapenv:Header/>
<soapenv:Body>
  <web:reportData soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <systemId xsi:type="xsd:string">1234</systemId>
       <data xsi:type="rep:ArrayOf_tns1_DataMessage" soapenc:arrayType="mes:DataMessage[]"
         xmlns:rep="http://localhost:8080/SiteScope/services/reportMonitorData"
         xmlns:mes="messages.client.webservice.soa.monitors.sitescope.mercury.com">
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">Title</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">Severity</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">SourceHint</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">CiHint</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">EtiHint</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">ComponentCi</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">HostHint</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">Description</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">Category</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">SubCategory</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
```

```
               <key xsi:type="soapenc:string">Key</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">CloseKey</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
       <data xsi:type="mes:DataMessage">
               <key xsi:type="soapenc:string">LogOnly</key>
               <value xsi:type="soapenc:string">?</value>
       </data>
     </data>
   </web:reportData>
  </soapenv:Body>
</soapenv:Envelope>
```

- ## For metrics: configure the reportMetricObject method

  Use the **reportMetricObject** method to submit a single metric. If you want to submit metrics in bulk, use the **reportMetricsArray** method.

  Enter the following service request:

  ```
  <wsdl:message name="reportMetricObjectRequest">
    <wsdl:part name="metric" type="tns1:MetricMessage" />
  </wsdl:message>
  ```

  where:

  "metric" is a metric of the type `MetricMessage`, which contains various value strings:

  ```
  <complexType name="MetricMessage">
   <complexContent>
      <extension base="tns1:AbstractMessage">
       <sequence>
        <element name="measurementName" nillable="true" type="soapenc:string" />
        <element name="measurementValue" nillable="true" type="soapenc:double" />
        <element name="monitorName" nillable="true" type="soapenc:string" />
        <element name="monitorState" nillable="true" type="soapenc:string" />
        <element name="monitorType" nillable="true" type="soapenc:string" />
        <element name="quality" nillable="true" type="soapenc:int" />
        <element name="measurementETI" nillable="true" type="soapenc:string" />
        <element name="measurementCIHint" nillable="true" type="soapenc:string" />
       </sequence>
     </extension>
    </complexContent>
  </complexType>
  ```

  The service success response is:

  ```
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <soapenv:Body>
        <ns1:reportMetricObjectResponse
  ```

```
         soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:ns1="http://webservice.soa.monitors.sitescope.mercury.com">
         <reportMetricObjectReturn xsi:type="xsd:int">1</reportMetricObjectReturn>
      </ns1:reportMetricObjectResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Example - Sending a metric using the reportMetricObject request

The question marks in the following example represent string values as follows:

- **xsd:double** is a double value
- **xsd:int** is an integer value
- **xsd:string** is a string value

Example:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:web="http://webservice.soa.monitors.sitescope.mercury.com">
<soapenv:Header/>
<soapenv:Body>
  <web:reportMetricObject
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <metric xsi:type="mes:MetricMessage"
    xmlns:mes="messages.client.webservice.soa.monitors.sitescope.mercury.com">
      <sourceTimeStamp xsi:type="xsd:double">?</sourceTimeStamp>
      <targetName xsi:type="xsd:string">?</targetName>
      <timeStamp xsi:type="xsd:double">?</timeStamp>
      <uniqueSystemId xsi:type="xsd:string">1234</uniqueSystemId>
      <measurementCIHint xsi:type="xsd:string">/</measurementCIHint>
      <measurementETI xsi:type="xsd:string">?</measurementETI>
      <measurementName xsi:type="xsd:string">/</measurementName>
      <measurementValue xsi:type="xsd:double">?</measurementValue>
      <monitorName xsi:type="xsd:string">?</monitorName>
      <monitorState xsi:type="xsd:string">?</monitorState>
      <monitorType xsi:type="xsd:string">?</monitorType>
      <quality xsi:type="xsd:int">?</quality>
    </metric>
  </web:reportMetricObject>
</soapenv:Body>
</soapenv:Envelope>
```

- ## For metrics arrays: configure the reportMetricsArray method

The **reportMetricsArray** method enables you to submit metrics in bulk instead of one by one. This is a more efficient method than the **reportMetricObject** method because it involves less network traffic.

However, although you can submit multiple metrics in one Web service request, the policy rules evaluate the metrics individually and not as a whole. For example, if you submit an array of three metrics, the policy rules will be applied three times, not just once.

Enter the following service request:

```
<wsdl:message name="reportMetricsArrayRequest">
  <wsdl:part name="metrics" type="impl:ArrayOf_xsd_anyType"/>
</wsdl:message>
```

where:

"metrics" is an array of metrics of the type `ArrayOf_xsd_anyType`, which contains various value strings:

```
<complexType name="ArrayOf_xsd_anyType">
  <complexContent>
    <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
    </restriction>
   </complexContent>
</complexType>
```

The element `<sourceTimeStamp>` must have the same value for each metric in the array. The source timestamp is a time number (the number of seconds that have elapsed since January 1, 1970 at 00:00:00 GMT (1970-01-01 00:00:00 GMT)), and represents the time at which the metric sample is sent. When BSM Connector receives the service request, it compares the metric sample time with its local time and adjusts the sample time accordingly; that is, BSM Connector calculates the time difference to the third-party system and increases or decreases the sample time accordingly. If you do not want BSM Connector to adjust the sample time, set the element `<sourceTimeStamp>` to zero (0) for all metrics in the array.

The service success response is:

```
<soapenv:Envelope
     xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:reportMetricsArrayResponse
         soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:ns1="http://webservice.soa.monitors.sitescope.mercury.com">
         <reportMetricsArrayReturn xsi:type="xsd:int">1</reportMetricsArrayReturn>
      </ns1:reportMetricsArrayResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Example - Sending an array of metrics using the reportMetricsArray request

The question marks in the following example represent string values as follows:

- **xsd:double** is a double value
- **xsd:int** is an integer value
- **xsd:string** is a string value

Example:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:web="http://webservice.soa.monitors.sitescope.mercury.com"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soapenv:Header/>
<soapenv:Body>
  <web:reportMetricsArray
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <metrics xsi:type="rep:ArrayOf_xsd_MetricMessage"
    soapenc:arrayType="mes:MetricMessage[]">
      <metric xsi:type="mes:MetricMessage"
      xmlns:mes="messages.client.webservice.soa.monitors.sitescope.mercury.com">
        <sourceTimeStamp xsi:type="xsd:double">?</sourceTimeStamp>
        <targetName xsi:type="xsd:string">?</targetName>
        <timeStamp xsi:type="xsd:double">?</timeStamp>
        <uniqueSystemId xsi:type="xsd:string">1234</uniqueSystemId>
        <measurementCIHint xsi:type="xsd:string">/</measurementCIHint>
        <measurementETI xsi:type="xsd:string">?</measurementETI>
        <measurementName xsi:type="xsd:string">/</measurementName>
        <measurementValue xsi:type="xsd:double">?</measurementValue>
        <monitorName xsi:type="xsd:string">?</monitorName>
        <monitorState xsi:type="xsd:string">?</monitorState>
        <monitorType xsi:type="xsd:string">?</monitorType>
        <quality xsi:type="xsd:int">?</quality>
      </metric>
      <metric xsi:type="mes:MetricMessage"
      xmlns:mes="messages.client.webservice.soa.monitors.sitescope.mercury.com">
        <sourceTimeStamp xsi:type="xsd:double">?</sourceTimeStamp>
        <targetName xsi:type="xsd:string">?</targetName>
        <timeStamp xsi:type="xsd:double">?</timeStamp>
        <uniqueSystemId xsi:type="xsd:string">1234</uniqueSystemId>
        <measurementCIHint xsi:type="xsd:string">/</measurementCIHint>
        <measurementETI xsi:type="xsd:string">?</measurementETI>
        <measurementName xsi:type="xsd:string">/</measurementName>
        <measurementValue xsi:type="xsd:double">?</measurementValue>
        <monitorName xsi:type="xsd:string">?</monitorName>
        <monitorState xsi:type="xsd:string">?</monitorState>
        <monitorType xsi:type="xsd:string">?</monitorType>
        <quality xsi:type="xsd:int">?</quality>
      </metric>
    </metrics>
  </web:reportMetricsArray>
</soapenv:Body>
</soapenv:Envelope>
```

Tasks

This section includes:

- "How to configure the connection to the Web service listener" below
- "Event integration only: How to load example data into the policy" on the next page

How to configure the connection to the Web service listener

This task describes how configure the connection between the policy and the Web service listener.

1. **Enable the connection to the BSM Connector reportMonitorData Web service listener**

   To enable the connection to the BSM Connector **reportMonitorData** Web service listener, you must create a client code (in any language) that makes the connection and handles the reporting of the data to BSM Connector through the Web service listener.

   a. Open a Web browser and go to BSM Connector (`https://<BSM Connector host>:30000/bsmconnector/services`).

      Take the WSDL file of the service **reportMonitorData**. The WSDL is an interface file which represents the API of the **reportMonitorData** Web service listener in BSM Connector. The **reportMonitorData** service is the service that listens to incoming messages and forwards them to BSM. This file is used to create the client stubs that connect to the service and report the data.

   b. Generate the stubs using the WSDL file. The generation of the stubs can be to any language. The way to create the files depends on the language that you want to use.

      For example, if you want to use Java as the client code, you must use the WSDL2JAVA task in AXIS package that can be downloaded from their Web site. Run **Java org.apache.axis.wsdl.WSDL2Java <name of saved WSDL file>**. After running this, you get two packages. One package is **com**, which holds the needed objects for sending the data, and the second is **localhost**, which holds the stubs that makes the connection to BSM Connector Web service listener.

   c. Write the actual client code which uses the generated classes to send the data to BSM Connector. In the code, call the **setreportMonitorDataEndpointAddress(<BSM Connector targetHost>)**, which is found in **MonitorDataAcceptorServiceLocator** (one of the generated stubs) to set the BSM Connector address to where you want the data reported.

2. Configure the reporting method. Depending on the type of data you want to send (events, metrics, or topology), choose the appropriate reporting method. For more information on reporting methods, see "Reporting methods" on page 276.

3. Type the system ID in the Source page.

Event integration only: How to load example data into the policy

1. Complete the Source page in the Web service listener policy, and save it.

   Then activate the policy by selecting it in the BSM Connector user interface and clicking ⚙ in the toolbar.

2. Run your Web service client and send SOAP messages for the policy to the Web service listener.

3. Edit the policy by selecting it and clicking ✏ in the toolbar. Navigate to the Source page.

4. Click ⇥ to retrieve the key-value pairs from the SOAP messages. The results are displayed in the Sample Data tabs of the policy.

Related tasks

- "How to Collect Event Data Through the Web Service Listener" on page 271

- "How to Collect Metrics Data Through the Web Service Listener" on page 272
- "How to Collect Topology Data Through the Web Service Listener" on page 273

UI Descriptions

For a description of the Source page, see "Source Page - Web Service Listener Policies" on page 436.

Configuring Mappings in 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 >** **Web Service Listener**.

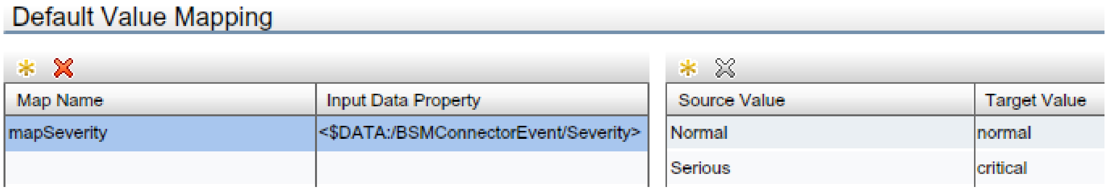
Alternatively, double-click an existing policy to edit it.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

A custom variable consists of a map name, an optional <key> element from the reportData request (that is, the SOAP message sent to the Web service listener), and one or more source and target value pairs. For example, you can assign the <key> value `<$DATA:/BSMConnectorEvent/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 SOAP message.


Default Value Mapping

Map Name	Input Data Property
mapSeverity	<$DATA:/BSMConnectorEvent/Severity>

Source Value	Target Value
Normal	normal
Serious	critical

<key> elements use the following syntax: <$DATA:/BSMConnectorEvent/<*key*>>

<*key*> is the string assigned to the <key> element in the SOAP message (for example, `<key xsi:type="soapenc:string">Severity</key>`)

For example, the custom variable `mapSeverity` has the SOAP message key `Severity` assigned.

Assigning a SOAP message key to a map name is optional. If you do not assign a SOAP message key 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 no SOAP messages have been received for the policy.

The Key tab shows the following information, if sample data is available:

- Keys

  If sample data is available, then the Key section shows all SOAP message keys received by the Web service listener for the specified system ID.

  You can sort the information that appears in the Keys section so that data appears in either ascending or descending order, indicated by either an up or down arrow at the top of the list. The items in the Keys section are by default sorted alphabetically in ascending order. To change the sort order, click the arrow at the top of the list.

- Values

  The Values section displays the values of a key selected in the Keys section. If a value appears more than once, click ⧉ to show or hide duplicate values. To find values that belong to more than one key, select the value and click 🔍. The Key Sample Data window opens and shows all keys that have the selected value.

When you drag a key from the keys 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 key. You can then drag one or more Web service 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 Web service request <key> elements

This task describes how to map <key> elements to custom variables.

1. Create one or more custom variables.

   Click ✱ above the Map Name column and type the variable name in the map name field. Keys are optional. If you do not assign a key to a variable, you must add the source value directly to the variable when you insert the variable in an event attribute.

2. Add one or more source and target value pairs to each custom variable.

   - If you are working with sample data, drag the value from the Values list to the Source Value column, and type the target value in the corresponding field.

     Alternatively, click ✱ above the Source Value column and type the source and target values in the corresponding fields.

   - Optionally, use the **Indicators** tab to add indicators to the source or target value fields. After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).

To insert an indicator in a source or target value field, drag and drop the indicator name (for example, `Normal`) or the indicator name and state (for example, `HTTPServer:Normal`) from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

Related tasks

- "How to Collect Event Data Through the Web Service Listener" on page 271

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Events Only)" on page 410
- "Sample Data Tab - Web Service Listener Policies" on page 427
- "Indicators Tab" on page 410

Configuring Event Defaults in Web Service Listener Policies

The Events 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 >** **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 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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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 **Sample Data** tab to drag keys and values to the policy fields. Alternatively, you can type the path to the key directly into the attribute box.

   <key> elements use the following syntax: <$DATA:/BSMConnectorEvent/<*key*>>

   <*key*> is the string assigned to the <key> element in the SOAP message (for example, `<key xsi:type="soapenc:string">Severity</key>`)

   BSM Connector replaces the key at runtime with the value of the specified key. If you insert a key value, the value will be used.

   Note:

   - The sample data includes a key-value pair that represents the default routing domain configured for the BSM Connector server in BSM under **Admin > Integrations > BSM Connector Integrations**. The default routing domain name is `DefaultDomain`.

     You may insert the default routing domain name in CI-related fields of the policy to help BSM distinguish different routing domains.

   - The Sample Data tab is empty if no sample data has been loaded into the policy or if no SOAP messages have been received for the policy. For information about loading sample data into a Web service listener policy, see "Event integration only: How to load example data into the policy" on page 282.

6. Optionally, 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 Web Service Listener Policies " on page 283). To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

   Alternatively, type the 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(Severity)>`).

   If the custom variable does not have a SOAP message key assigned, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - Name of the SOAP message key, for example `<$MAP(Severity)>, <$DATA:/BSMConnectorEvent/Severity>`

   - The source value itself, for example `<$MAP(Severity)>,critical)>`

7. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example "`<$MSG_NODE>`".

Related tasks

- "How to Collect Event Data Through the Web Service Listener" on page 271

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Sample Data Tab - Web Service Listener Policies" on page 427
- "Mappings Tab (Events Only)" on page 412
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Metrics Defaults in Web Service Listener Policies

The Default Monitor 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.

To access

In the BSM Connector user interface, click * in the toolbar. Then click **Metrics >** **Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Defaults** to open the policy Default Monitor Attributes page.

Learn More

This section includes:

- "Metrics attributes" below
- "Parameters for metrics policies" on the next page

Metrics attributes

Each time a metrics policy runs, it extracts raw data from its defined data source and builds a metric structure, which it then forwards to BSM.

A metric structure consists of these attributes:

- Attributes related to the metrics source, category, or status:
  - Monitor name
  - Monitor type
  - Target
  - Time stamp
  - Quality
  - Monitor state
- Attributes related to a specific metric in the source:
  - Name
  - Value
  - CI hint
  - Indicator name

Parameters for metrics policies

The **reportMetricObject** and **reportMetricsArray** methods send the following keys, which you can reference in a metrics integration policy as `$`*`key`*. For example, if you insert `$MonitorName` in the Monitor Name field, the policy replaces `$MonitorName` with the monitor name from the SOAP message before sending the metric to BSM.

Key from SOAP Message	Policy Parameter
monitorName	$MonitorName
monitorType	$MonitorType
targetName	$TargetName
timeStamp	$TimeStamp
quality	$Quality
monitorState	$MonitorState
measurementName	$MeasurementName
measurementValue	$Value
measurementCIHint	$MeasurementCIHint
measurementETI	$MeasurementETI

Tasks

How to configure metrics defaults for Web service listener policies

This task describes how to configure metric attribute defaults for all metrics collected by this policy.

1. Define the default metric attributes common to all metrics collected by this policy, such as monitor name, type, and state.

   > **Tip:** The Policy Constants tab lists the available constants representing metrics quality (status) that are available in BSM Connector. Use drag and drop to add them to the **Quality** attribute.

2. In **Metrics**, click * to add attributes specific to a single metric.

   Alternatively, expand an existing metric group to edit it.

   > **Tip:** After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator from the Indicators tab to the policy. BSM Connector inserts the indicator name.

3. Optionally, use the **Operations** tab to apply operations to the metric attribute values.

> **Tip:**
>
> You can insert keys directly into the boxes using the $<*key*> syntax.
>
> <*key*> is the string assigned to the <key> element in the SOAP message (for example, `<key xsi:type="soapenc:string">Severity</key>`)
>
> BSM Connector replaces the key at runtime with the value of the specified key. If you insert a key value, the value will be used.

Related tasks

- "How to Collect Metrics Data Through the Web Service Listener" on page 272
- "How to Collect Metrics with Computer - Monitor Topology" on page 55
- "How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60
- "How to Collect Metrics Without Reporting Topology" on page 72

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Default Monitor Attributes and Attributes Tabs" on page 407
- "Indicators Tab" on page 410
- "Operations Tab (Metrics Only)" on page 412
- "Policy Constants Tab (Metrics Only)" on page 419

Configuring Event Rules in Web Service Listener Policies

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the source. The settings enable you to configure the event that BSM Connector sends to BSM.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

To access

- In the BSM Connector user interface, click * in the toolbar. Then click **Event >** **Web Service Listener**.
- In the BSM Connector user interface, click * in the toolbar. Then click **Metrics >** **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 BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to BSM.

Tasks

How to configure rules in Web service listener 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 Web service message. Each policy must have at least one rule.
2. In the **Rule Content** section, use the **Condition** tab to specify the keys and values that the policy searches for in the SOAP message that the policy evaluates. 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 <key> that the policy searches for. You must prefix the <key> with `/BSMConnectorEvent/` (for example, `/BSMConnectorEvent/Severity`).

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 "Configuring Event Rules in Web Service Listener Policies" on the previous page.

e. In the **Operand** field, type the value or pattern that you want the policy to compare with the <key>.

   > **Tip:** You can use standard BSM Connector pattern-matching rules when matching values. Select the matches operator and click ▸ in the Operand field to open the pattern matching expression toolbox. The toolbox displays the following:
   >
   > - **Pattern Matching Expressions.** Click an expression to insert it in the Operand field.
   > - **Variable Bindings Options.** Variable bindings options include case sensitivity and field separators for the rule. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used. See also "Configuring Options in Web Service Listener Policies" on page 296.

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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. Optionally, use the **Sample Data** tab to drag keys and values to the policy fields. Alternatively, you can type the path to the key directly into the attribute box.

   <key> elements use the following syntax: <$DATA:/BSMConnectorEvent/<*key*>>

<*key*> is the string assigned to the <key> element in the SOAP message (for example, `<key xsi:type="soapenc:string">Severity</key>`)

BSM Connector replaces the key at runtime with the value of the specified key. If you insert a key value, the value will be used.

> **Note:**
>
> - The sample data includes a key-value pair that represents the default routing domain configured for the BSM Connector server in BSM under **Admin > Integrations > BSM Connector Integrations**. The default routing domain name is `DefaultDomain`.
>
>   You may insert the default routing domain name in CI-related fields of the policy to help BSM distinguish different routing domains.
>
> - The Sample Data tab is empty if no sample data has been loaded into the policy or if no SOAP messages have been received for the policy. For information about loading sample data into a Web service listener policy, see "Event integration only: How to load example data into the policy" on page 282.

8. Optionally, 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 Web Service Listener Policies " on page 283). To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

   Alternatively, type the 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(Severity)>`).

   If the custom variable does not have a SOAP message key assigned, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - Name of the SOAP message key, for example `<$MAP(Severity)>, <$DATA:/BSMConnectorEvent/Severity>`
   - The source value itself, for example `<$MAP(Severity)>,critical)>`

9. Optionally, use the **Pattern Matching Variables** tab to add variables created through 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. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

    It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data Through the Web Service Listener" on page 271
- "How to Collect Metrics Data Through the Web Service Listener" on page 272

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Tab - Rules Only" on page 402
- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Sample Data Tab - Web Service Listener Policies" on page 427
- "Mappings Tab (Events Only)" on page 412
- "Pattern Matching Variables Tab (Event Rules Only)" on page 419
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Metrics Rules in Web Service Listener Policies

Rules define what a policy should do in response to a specific type of metric. Each rule consists of a condition and of settings for the metrics generated by the policy. The condition contains the string or pattern that must be matched for the rule to apply. The settings enable you to configure the metrics that BSM Connector sends to BSM.

If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

To access

In the BSM Connector user interface, click * in the toolbar. Then click **Metrics >** **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
- "Metrics attributes" below
- "Parameters for metrics policies" below

Rule types

The rule types are:

- **Process on matched rule.** If matched, BSM Connector sends metrics to BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to BSM.

Metrics attributes

Each time a metrics policy runs, it extracts raw data from its defined data source and builds a metric structure, which it then forwards to BSM.

A metric structure consists of these attributes:

- Attributes related to the metrics source, category, or status:
  - Monitor name
  - Monitor type
  - Target
  - Time stamp
  - Quality
  - Monitor state
- Attributes related to a specific metric in the source:
  - Name
  - Value
  - CI hint
  - Indicator name

Parameters for metrics policies

The **reportMetricObject** and **reportMetricsArray** methods send the following keys, which you can reference in a metrics integration policy as `$key`. For example, if you insert `$MonitorName` in the Monitor Name field, the policy replaces `$MonitorName` with the monitor name from the SOAP message before sending the metric to BSM.

Key from SOAP Message	Policy Parameter
monitorName	$MonitorName

Key from SOAP Message	Policy Parameter
monitorType	$MonitorType
targetName	$TargetName
timeStamp	$TimeStamp
quality	$Quality
monitorState	$MonitorState
measurementName	$MeasurementName
measurementValue	$Value
measurementCIHint	$MeasurementCIHint
measurementETI	$MeasurementETI

Tasks

How to configure rules for metrics in Web service listener 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.

   The match condition must be a valid boolean expression. The expression can contain one or more operations from the Operations tab. The expression can access the contents of the data that is being processed using the dollar sign ($) notation.

3. *Optional.* If you are creating a rule of the type 'process on matched rule', set **Attributes** for metrics that you want the policy to send. 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: common attributes at the top that apply to all metrics defined this policy, and attributes specific to each metric.

   a. Define the metric attributes common to all metrics collected by this policy, such as monitor name, type, and state.

      > **Tip:** The Policy Constants tab lists the available constants representing metrics quality (status) that are available in BSM Connector. Use drag and drop to add them to the **Quality** attribute.

   b. In **Metrics**, click * to add attributes specific to a single metric.

      Alternatively, expand an existing metric group to edit it.

      > **Tip:** After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators

> (HIs) and event type indicators (ETIs).
>
> To insert an indicator, drag and drop the indicator from the Indicators tab to the policy. BSM Connector inserts the indicator name.

You can create multiple metrics.

Click ▶ to expand a metric attribute group

4. Optionally, use the **Operations** tab to apply operations to the metric attribute values.

> **Tip:**
>
> You can insert keys directly into the boxes using the $<*key*> syntax.
>
> <*key*> is the string assigned to the <key> element in the SOAP message (for example, `<key xsi:type="soapenc:string">Severity</key>`)
>
> BSM Connector replaces the key at runtime with the value of the specified key. If you insert a key value, the value will be used.

Related tasks

- "How to Collect Metrics Data from Log Files" on page 148
- "How to Collect Metrics with Computer - Monitor Topology" on page 55
- "How to Collect Metrics with Custom, Computer, or Computer - Running Software Topology Data" on page 60
- "How to Collect Metrics Without Reporting Topology" on page 72

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Tab (Rules Only)" on page 410
- "Default Monitor Attributes and Attributes Tabs" on page 407
- "Indicators Tab" on page 410
- "Operations Tab (Metrics Only)" on page 412
- "Policy Constants Tab (Metrics Only)" on page 419

Configuring Options in 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 > 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 log file 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 Web Service Listener Policies" on the previous page.

Related tasks

- "How to Collect Event Data Through the Web Service Listener" on page 271

UI Descriptions

For a description of the Options page, see "Options Page (Events Only)" on page 417.

Configuring Topology Scripts in Web Service Listener Policies

The topology page enables you to create a topology in BSM's RTSM by selecting a predefined script or creating your own custom script.

To access

- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Event > Web Service Listener**.
- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Metrics > Web Service Listener**.
- In the BSM Connector user interface, click ✱ in the toolbar. Then click **Topology > Web Service Listener**.

Alternatively, double-click an existing policy to edit it.

Click **Topology** to open the policy Topology page.

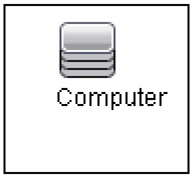
Learn More

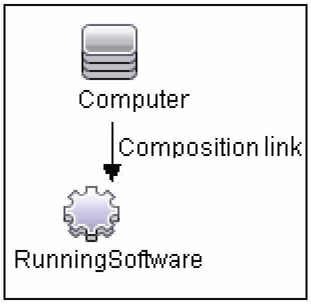
This section includes:

- "Selecting a Topology" below
- "Editing the Topology Script" on the next page
- "Mandatory Values When Reporting Topology Only" on page 301
- "Additional Documentation" on page 302

Selecting a Topology

The following topology scripts are available for integrating events, metrics, and topology:

Topology Script	Events Integration	Metrics Integration	Topology Integration
No Topology Select if you do not want to send any topology (although metrics and event data is still sent).	yes	yes	not available
Computer - Monitor (legacy) Creates a topology with a Computer CI connected to a SiteScope CI with a Monitored By link. **Note:** The Computer - Monitor topology script has been deprecated. For new metrics integrations, use the Computer, Computer - Running Software, or a custom topology script. The Computer - Monitor topology integration requires that the names or IP addresses of the nodes that it adds to the RTSM are accessible through DNS resolution. To successfully add a Node CI specified in a policy's `Target` field to the RTSM, BSM Connector must be able to resolve the node's fully qualified domain name and IP address through a DNS service.	not available	yes	not available
Computer Creates a topology with a Computer CI. 	yes	yes	yes

Topology Script	Events Integration	Metrics Integration	Topology Integration
Computer - Running Software Creates a topology with a Computer CI and a Running Software CI connected to it with a **Composition** relationship. The following illustrates the topology created for the Computer - Running Software integration type which retrieves data from a third-party system: 	yes	yes	yes
Custom You create your own topology if you want the retrieved data to be forwarded to specific CIs and not one of the predefined topology scripts. **Note:** You should only select **Custom** if you are familiar with the Jython language, since you must create the topology script in Jython yourself. Depending on the data type you want to collect, we recommend that you select and edit one of the predefined scripts.	yes	yes	yes

Editing the Topology Script

To configure the topology, you can edit the topology script that appears in the Topology page when creating a policy. You can use the script editor provided, or use any other script editor.

You can choose one of the predefined topologies which are already configured with the necessary information. Following are the guidelines for editing the script if you want to create your own topology.

- The Jython language is sensitive to spaces and tabs. For more information on Jython, see http://wiki.python.org/moin/HowToEditPythonCode.
- You must leave the `import` section of the predefined topology scripts as is and only add to it.
- The main body of the script is mandatory and consists of:

```
def DiscoveryMain(Framework)
```

This main function is responsible for creating Object State Holder Vector (OSHV) results. It holds the CI and CI relationship data, and how to map the incoming samples to the CIs.

- Use the built-in "logger" to debug the topology scripts when samples arrive. You do this by modifying the level and type of information reported to the log file:

  a. Add the logger import statement before system_lib import statement to the topology script, for example:

  ```
  import logger
  …
  import system_lib
  ```

  b. Change the log file settings in the **<BSM Connector root directory>/conf/core/Tools/log4j/PlainJava/bac_integration.properties** file as follows:

  Open the **bac_integration.properties** file in a text editor and locate the following lines in the file:

  ```
  # Jython logger
  log4j.category.PATTERNS_DEBUG=${loglevel}, discovery.appender
  ```

  Change the argument of **log4j.category.PATTERNS_DEBUG** from **${loglevel}** to **DEBUG**, as follows:

  ```
  log4j.category.PATTERNS_DEBUG=DEBUG, discovery.appender
  ```

  c. Save the file. It may take a few seconds for the changes to take effect.

  The debug data is written to the **<BSM Connector root directory>/log/discovery.log** file.

- BSM Connector automatically populates the **Monitored by** attribute of the reported CIs with the values `BSM Connector` and `<policy name>`. `<policy name>` is the name of the policy as set in the policy's Properties page.

- You can access values received through the Web service from Web service listener policies. For example, you can access the value of the key in the SOAP request `<key>Host</key` in the following way:

  ```
  value1 = Framework.getDestinationAttribute(“Host")
  ```

- You may use the following code to set the **Monitored by** attribute on the CIs reported by the script:

  ```
  ems_lib.addMonitoredByForThirdPartySoftware(Framework,
  <computer>)
  ```

- Use the following code to get the routing domain for a Node CI:

  ```
  domainName = system_lib.getDiscoveryDomainByIP(Framework, <IP or
  ```

```
node DNS name>)
```

Web service listener policies that integrate events only:

The sample data includes a key-value pair that represents the default routing domain configured for the BSM Connector server in BSM under **Admin > Integrations > BSM Connector Integrations**. The default routing domain name is `DefaultDomain`.

You may insert the default routing domain name in CI-related fields of the policy to help BSM distinguish different routing domains.

For more information on the routing domain attribute in RTSM, see the Data Flow Management Guide.

Mandatory Values When Reporting Topology Only

The following values are mandatory when reporting only the topology discovered by the BSM Connector policies, without reporting events or metrics data:

For Topology Script	Field Name	Description
• Computer • Computer - Running Software	`target_name`	Name of the host or machine that generated the event. This can be added manually or taken from: `Framework.getDestinationAttribute ("<someAttribute>")` **Examples:** Log file policies: Framework.getDestinationAttribute("group0") where group0 is the value of the first pattern matching group. Database policies: Framework.getDestinationAttribute("NAME") where NAME is the name of a database column. Web service listener policies: Framework.getDestinationAttribute (“Host") where HOST is the key in the SOAP request <key>Host</key>.
• Computer • Computer - Running Software	`target_ip`	IP of the host or machine. This can be added manually, or calculated using: `HostIPCachingManager.getIPByHostName(target_name)` where `target_name` represents a valid host or machine, or you can use: `HostIPCachingManager.getIPByHostName ("<someAttribute>")`
Computer - Running Software	`name`	Name of Running Software. This can be added manually, or taken from: `Framework.getDestinationAttribute ("<someAttribute>")`

Additional Documentation

For general information on topology scripts, see "Create Jython Code" and "Developing Jython Adapters" in the RTSM Developer Reference Guide.

For information about Java classes that can be used in topology scripts, see "HP Data Flow Management API Reference" in the RTSM Developer Reference Guide.

Tasks

This section includes:

- "How to report topology with event data" below
- "How to report topology with metrics data" on the next page
- "How to report topology without event or metrics data" on the next page

How to report topology with event data

1. In the BSM Connector user interface, click ✳ in the toolbar. Then click **Event >** **Web Service Listener**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.

3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. To create a custom script, see "Editing the Topology Script" on page 299 for more information.

4. Optionally, use the **Sample Data** tab to drag keys and values to the topology script. Alternatively, type the key directly into the topology script, for example `Severity`. In this example, `Severity` is the string assigned to the <key> element in the SOAP request: `<key xsi:type="soapenc:string">Severity</key>`.

   BSM Connector replaces the key at runtime with the value of the specified key. If you insert a key value, the value will be used.

   > **Note:**
   >
   > - The sample data includes a key-value pair that represents the default routing domain configured for the BSM Connector server in BSM under **Admin > Integrations > BSM Connector Integrations**. The default routing domain name is `DefaultDomain`.
   >
   >   You may insert the default routing domain name in CI-related fields of the policy to help BSM distinguish different routing domains.
   >
   > - The Sample Data tab is empty if no sample data has been loaded into the policy or if no SOAP messages have been received for the policy. For information about loading sample data into a Web service listener policy, see "Event integration only: How to load example data into the policy" on page 282.

How to report topology with metrics data

Tip:

You can insert keys directly into the topology script, for example `Severity`. In this example, `Severity` is the string assigned to the <key> element in the SOAP request: `<key xsi:type="soapenc:string">Severity</key>`.

BSM Connector replaces the key at runtime with the value of the specified key. If you insert a key value, the value will be used.

1. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Metrics >** **Web Service Listener**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.
3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. The Computer - Monitor script does not need to be edited. To create a custom script, see "Editing the Topology Script" on page 299 for more information.

How to report topology without event or metrics data

Tip:

You can insert keys directly into the topology script, for example `Severity`. In this example, `Severity` is the string assigned to the <key> element in the SOAP request: `<key xsi:type="soapenc:string">Severity</key>`.

BSM Connector replaces the key at runtime with the value of the specified key. If you insert a key value, the value will be used.

1. In the BSM Connector user interface, click ✱ in the toolbar. Then click **Topology >** **Web Service Listener**.

   Click **Topology** to open the policy Topology page.

2. Select a script template and if necessary, click **Load Template**.
3. Edit the script if necessary to meet your needs.

   The predefined scripts Computer and Computer - Running Software contain instructions that help you adapt the script to your environment. To create a custom script, see "Editing the Topology Script" on page 299 for more information.

4. Map the data discovered by the policy to the relevant attributes in the topology script.

   For a list of mandatory values, see "Mandatory Values When Reporting Topology Only" on page 301.

Note: BSM removes CIs from the RTSM if no topology data has been received for them over a period of time. To prevent BSM from deleting the CIs that BSM Connector has sent to BSM, BSM Connector regularly sends a list of active CIs to BSM in addition to the topology data. Inactive CIs are not sent to the RTSM and are marked for deletion in the RTSM.

You can configure BSM Connector to always send CIs regardless of whether topology data has been received for them. This guarantees that CIs reported by BSM Connector are never deleted automatically from the RTSM. This configuration option is known as always touch mode and is recommended for BSM Connector topology-only policies that collect data from incremental data sources such as log files or databases. The first time the policies run, they collect all CIs from the data source. The next time the policies run, they collect new and changed CIs only and then forward only the additions and changes to BSM. The always touch mode ensures that the already existing, unchanged CIs are also reported and are thus not deleted.

For more information, see "Touching of CIs in BSM Connector" on page 23.

Related tasks

- "How to Collect Event Data Through the Web Service Listener" on page 271
- "How to Collect Metrics Data Through the Web Service Listener" on page 272
- "How to Collect Topology Data Through the Web Service Listener" on page 273

UI Descriptions

For a description of the Topology page, see the following sections:

- "Topology Page" on page 442
- "Sample Data Tab - Web Service Listener Policies" on page 427

Troubleshooting Web Service Listener Policies

This section describes troubleshooting and limitations when working with Web service listener policies.

This section includes:

- "Debugging errors" on the next page
- "Web service listener policy file rolling" on page 306
- "Checking received Web service requests" on page 307
- "Client test tools" on page 307
- "Metrics troubleshooting" on page 309
- "Topology troubleshooting" on page 309

Debugging errors

- Check for errors in the following files:

  <BSM Connector root directory>\logs\error.log

  <BSM Connector root directory>\logs\RunMonitor.log

  <BSM Connector root directory>\logs\bac_integration\bac_integration.log.

- You can modify the level and type of information reported to the log files by changing the log file settings in the **<BSM Connector root directory>\conf\core\Tools\log4j\PlainJava\log4j.properties** file. You can instruct the logging mechanism to:

  - Report logged information in less or greater detail than is reported by default.
  - Log all metrics sent by metrics integration policies to BSM.
  - Log all received data from third-party systems.

  To modify log settings:

  a. Open the **log4j.properties** file in a text editor.

  b. To specify that metrics sent by metrics integration policies to BSM be logged:

     i. Locate the following lines in the file:

     ```
     log4j.category.EmsSamplePrinter=${loglevel},
     integration.appender
     log4j.additivity.EmsSamplePrinter=false
     ```

     ii. Change the argument of **log4j.category.EmsSamplePrinter**from **${loglevel}** to **DEBUG**, as follows:

     ```
     log4j.category.EmsSamplePrinter=DEBUG, integration.appender
     ```

     iii. Save the file. It may take a few seconds for the changes to take effect.

     The results are logged to the **bac_integration.log** file.

  c. To specify that all received data from third-party systems be logged:

     i. Locate the following lines in the file:

     ```
     log4j.category.EmsEventPrinter=${loglevel}, monitors.appender
     log4j.additivity.EmsEventPrinter=false
     ```

     ii. Change the argument of **log4j.category.EmsEventPrinter** from **${loglevel}** to **DEBUG**, as follows:

     ```
     log4j.category.EmsEventPrinter=DEBUG, monitors.appender
     ```

     iii. Save the file. It may take a few seconds for the changes to take effect.

     The results are logged to the **RunMonitor.log** file.

  > **Tip:** After you have determined the cause of the problem, we recommend that you set log levels to their default settings so as not to overload the system.

- If metrics are created and sent from BSM Connector, but the data is not seen in BSM Service Health, Event Log, or SiteScope reports, look in **<BSM root directory>\log\mercury_wde\wdeIgnoredSamples.log** to make sure the metrics were not dropped due to missing fields or values.
- Change the logging level for Service Health to verify that Service Health received the samples. Open the following file on the Gateway Server machine: **<BSM root directory>\conf\core\tools\log4j\mercury_wde\wde.properties**

  Change the log level parameter to DEBUG in the following lines:

  - `log4j.category.com.mercury.am.platform.wde.decode.IgnoredSamplesLogger=${loglevel}, IgnoredSamples.appender`
  - `log4j.category.com.mercury.am.platform.wde.publish_SamplePublisherSamples=${loglevel}, PublishedSamples.appender`

  Look at the corresponding log files:

  - **<BSM root directory>\logs\mercury_wde\wdeIgnoredSamples.log**
  - **<BSM root directory>\logs\mercury_wde\wdePublishedSamples.log**

Web service listener policy file rolling

A Web service listener event integration policy reads the following XML file:

Windows: `%OvDataDir%\tmp\<policyID>.xml`

Linux: `/var/opt/OV/tmp/<policyID>.xml`

The file contains the received SOAP messages in an XML file format that Web service listener policies can evaluate. The file exists only for policies that integrate events.

To prevent the file from becoming too large, a new version of the file is created when its size reaches 100000 KB. The original file is renamed and stored as a backup file. By default, up to three backup files can be created.

> **Caution:** When the XML file rolls, that is, a new version is created, but the policy has not completed reading all entries, the unread entries are lost and no events are created.

The following defaults are set:

- **Maximum file size:** 100000 KB

  > **Tip:** If not all expected events arrive in BSM, increase the maximum file size to 1000000 KB.

- **Maximum number of backup files:** 3

To change the defaults:

1. Open the following file:

   ```
   <BSM Connector root directory>/groups/master.config
   ```

2. Change the following values as required:

```
_eventRollingMaxFileSizeInKB=100000
```

```
_eventRollingMaxBackup=3
```

3. Restart the BSM Connector server:

   Windows: Restart the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Restart the BSM Connector main process, type `/opt/HP/BSMConnector/stop` followed by `/opt/HP/BSMConnector/start`.

Checking received Web service requests

Check the following Web service listener XML file for any Web service requests:

Windows: `%OvDataDir%\tmp\<`*`policyID`*`>.xml`

Linux: `/var/opt/OV/tmp/<`*`policyID`*`>.xml`

The file contains the received SOAP messages in an XML file format that Web service listener policies can evaluate. The file exists only for policies that integrate events.

Client test tools

After creating a Web service listener policy in BSM Connector, you can check connectivity to the Web service listener by using the client check tools. This tools send messages to the BSM Connector **reportMonitorData** Web service. The messages can be metrics or event messages.

The client test tools are located on the BSM Connector server in:

`<`*`BSM Connector root directory`*`>/conf/ems/webservice/test_client`

After running the tools, access BSM and see if the data that you sent is displayed.

- ### test_data_client

  Use the **test_data_client** tool to send events or metrics:

  Usage: **test_data_client.bat|sh sysId=<*The Web Server system id*> [host=<*BSM Connector host*>] [port=<*BSM Connector port*>] [isSSL=<false/true>] key1=value1 key2=value2 ...**

  where:

sysId	Required. System ID of the policy that receives the messages. **Default value:** none
host	Address of the BSM Connector host that receives the messages. **Default value:** localhost
port	Port at which BSM Connector receives the messages. **Default value:** 8080

isSSL	Is BSM Connector configured for SSL. **Possible values:** true, false **Default value:** false
key[*n*]	Required. Key-value pair, for example **key1=value1**. **Default value:** none

- ## test_metrics_client

Use the **test_metrics_client** tool to send metrics:

Usage: **sysId=<*The Web Server system id*> [host=<*BSM Connector host*>] [port=<*BSM Connector port*>] [isSSL=<false/true>] metric1..N=Quality=<QUALITY_GOOD/QUALITY_WARNING/QUALITY_ERROR>||MeasurementCIHint=<*?*>||MeasurementETI=<*?*>||MeasurementName=<*?*>||MeasurementValue=<*double value*>||MonitorName=<*?*>||MonitorState=<*?*>||MonitorType=<*?*>||TargetName=<*?*>||SourceTimeStamp=<*double value*>||TimeStamp=<*double value*>)**

where:

sysId	Required. System ID of the policy that receives the messages. **Default value:** none
host	Address of the BSM Connector host that receives the messages. **Default value:** localhost
port	Port at which BSM Connector receives the messages. **Default value:** 8080
isSSL	Is BSM Connector configured for SSL. **Possible values:** true, false **Default value:** false
metric*n*	
Quality	Quality in BSM Connector terms. Possible values are: QUALITY_ERROR, QUALITY_WARNING, QUALITY_GOOD.
MeasurementCIHint	CI resolution hint that is used to identify monitored CIs and relate metrics to them.

MeasurementETI	The display name of the ETI.
MeasurementName	Name the metric.
MeasurementValue	Value of metric. Must be a double-precision number.
MonitorName	Logical monitor name.
MonitorState	The monitor status.
MonitorType	The monitor type.
TargetName	The target of this monitor (such as the name of host machine).
SourceTimeStamp	Source stamp in the seconds since Jan 1st 1970 format. Must be a double-precision number.
TimeStamp	Time stamp in the seconds since Jan 1st 1970 format. Must be a double-precision number.

Metrics troubleshooting

See "Metrics Troubleshooting" on page 87 for more information on troubleshooting and limitations when BSM Connector sends metrics to BSM.

Topology troubleshooting

See "Topology Troubleshooting and Limitations" on page 338 for more information on troubleshooting and limitations when BSM Connector is enabled to report CIs and CIs relationships topology to BSM.

Chapter 10

Integrating XML File Data into BSM

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 events from third-party systems that can store their event information in XML format.

XML file policies process XML files and send events to BSM when certain conditions apply. You can define the attributes of the BSM event based on information in the XML file. This enables you to process events generated by other applications and to convert them to BSM events.

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 event information is embedded in XML elements and attributes.

> **Tip:** If the application does not store its events in XML files, you may write a program or script that extracts the events from wherever they are stored, formats the data using XML syntax, and generates an XML file with the events. If you have control over the XML elements that are used in the XML file, choose XML elements and attributes that map to event 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 your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, click ✳ in the toolbar, then click **Event >** XML **XML**. The XML file policy editor opens.

   Alternatively, double-click an existing XML file 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 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 313.

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 314.

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 317.

7. In the **Rules** page, define what the policy should do in response to a specific type of event.

   For details, see "Configuring Rules in XML File Policies" on page 319.

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 322.

9. Click **OK** 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.

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 314
- "Configuring Event Defaults in XML File Policies" on page 317
- "Configuring Rules in XML File Policies" on page 319
- "Configuring Options in XML File Policies" on page 322

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**. The XML file policy editor opens.

Alternatively, double-click an existing XML file 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 XML File Policy Properties" above.

Related tasks

- "How to Collect Event Data from XML Files" on page 311

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 423.

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**. The XML file policy editor opens.

Alternatively, double-click an existing XML file policy to edit it.

Click **Source** to open the policy Source page.

Learn More

Requirements for XML source files

XML files must meet the following criteria so that they can be processed correctly by XML file policies:

- The root element is optional.
- If a root element exists, it must not be closed by an end tag.
- All other XML elements must be complete.

The following example XML 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 event tags in the Source tab of XML file policies, only those events are processed by XML file policies.

```
<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>
```

Tasks

How to configure the XML source

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 event 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 event 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. You can define more than one event 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.

Related tasks

- "How to Collect Event Data from XML Files" on page 311

UI Descriptions

For a description of the Source page, see "Source Page - XML File Policies" on page 437.

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 **XML**. The XML file policy editor opens.

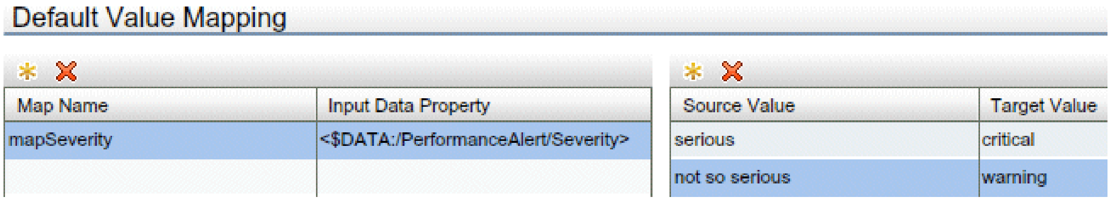
Alternatively, double-click an existing XML file policy to edit it.

Click **Mappings** to open the policy Mappings page.

Learn More

Mappings overview

A custom variable consists of a map name, an optional XML property (XML elements or attributes), 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 `Warning`. You can then assign the target value `Major` to the variable so that BSM Connector inserts the value `Major` into the event in all places where the variable is used and the source value is `Warning` in the XML log file.



Default Value Mapping

✱ ✖

Map Name	Input Data Property
mapSeverity	<$DATA:/PerformanceAlert/Severity>

✱ ✖

Source Value	Target Value
serious	critical
not so serious	warning

XML properties use the following syntax: <$DATA:/<*XML_property*>>

<*XML_property*> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute.

For example, the custom variable `mapSeverity` has the following XML property: `<$DATA:/Performance_Alert/Severity>` where `Severity` is a child element of `Performance_Alert`.

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.

> **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 event tags.

The Sample Data tab shows the following information if sample data is available:

- XML Properties section

  If sample data is available, then the XML Properties section of the Sample Data tab shows all XML elements and attributes that match an XML event tag. (You can identify attributes based on the preceding at sign (@).)

  The XML Properties section by default shows the short path to the XML property or value. To view the full path, click XML. The full path begins with the XML event tag specified in the Source tab.

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.

- The Values 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 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 one or more source and target value pairs to each custom variable.

   - If you are working with sample data, drag the value from the Values list to the Source Value column, and type the target value in the corresponding field.

     Alternatively, click * above the Source Value column and type the source and target values in the corresponding fields.

   - Optionally, use the **Indicators** tab to add indicators to the source or target value fields. After loading the indicators from the connected BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).

     To insert an indicator in a source or target value field, drag and drop the indicator name (for example, `Normal`) or the indicator name and state (for example, `HTTPServer:Normal`) from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

Related tasks

- "How to Collect Event Data from XML Files" on page 311

UI Descriptions

For a description of the Mappings page, see the following sections:

- "Mappings Page (Events Only)" on page 410
- "Sample Data Tab - XML File Policies" on page 428
- "Indicators Tab" on page 410

Configuring Event Defaults in XML File Policies

The Events 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**. The XML file policy editor opens.

Alternatively, double-click an existing XML file 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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
   >
   > To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. 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:/<*XML_property*>>

<*XML_property*> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute.

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.

6. 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). To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

   Alternatively, type the 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(map@Severity)>`).

   If the custom variable does not have an XML property assigned, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - XML path to the source value, for example `<$MAP(map@Severity, <$DATA/Performance_Alert/Severity>)>`

   - The source value itself, for example `<$MAP(map@Severity, Warning)>`

7. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

   It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data from XML Files" on page 311

UI Descriptions

For a description of the Defaults page, see the following sections:

- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399
- "Advanced Tab" on page 400
- "Sample Data Tab - XML File Policies" on page 428
- "Mappings Tab (Events Only)" on page 412
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Rules in XML File Policies

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the source. The settings enable you to configure the event that BSM Connector sends to BSM.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

To access

In the BSM Connector user interface, click * in the toolbar, then click **Event >** **XML**. The XML file policy editor opens.

Alternatively, double-click an existing XML file 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 BSM. 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 BSM.
- **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to BSM.

Tasks

How to configure 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 "Configuring Rules in XML File Policies" on the previous page.

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.

> **Tip:** You can use standard BSM Connector pattern-matching rules when matching values. Select the matches operator and click ▸ in the Operand field to open the pattern matching expression toolbox. The toolbox displays the following:
>
> ○ **Pattern Matching Expressions.** Click an expression to insert it in the Operand field.
>
> ○ **Variable Bindings Options.** Variable bindings options include case sensitivity and field separators for the rule. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used. See also "Configuring Options in XML File Policies" on page 322.

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 BSM server, the Indicators tab shows a hierarchy of configuration item types with the associated health indicators (HIs) and event type indicators (ETIs).
>
> To insert an indicator, drag and drop the indicator, indicator state, or both from the Indicators tab to the policy. Use the **Select target drop format** panel to toggle between inserting indicator names only and inserting indicator names and states.

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. 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:/<*XML_property*>>

<*XML_property*> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute.

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.

8. 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 Rules in XML File Policies" on page 319). To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

   Alternatively, type the 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(map@Severity)>`).

   If the custom variable does not have an XML property assigned, use the following syntax:

   <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

   - XML path to the source value, for example `<$MAP(map@Severity, <$DATA/Performance_Alert/Severity>)>`
   - The source value itself, for example `<$MAP(map@Severity, Warning)>`

9. Optionally, use the **Pattern Matching Variables** tab to add variables created through 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. Optionally, use the **Policy Variables** tab to add policy variables to event attributes. BSM Connector replaces the variables with the appropriate values in the generated event.

    It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces, for example `"<$MSG_NODE>"`.

Related tasks

- "How to Collect Event Data from XML Files" on page 311

UI Descriptions

For a description of the Rules page, see the following sections:

- "Rules Page - Policy Rules" on page 424
- "Condition Tab - Rules Only" on page 402
- "Event Attributes Tab" on page 390
- "Event Correlation Tab" on page 395
- "Custom Attributes Tab" on page 399

- "Advanced Tab" on page 400
- "Sample Data Tab - XML File Policies" on page 428
- "Mappings Tab (Events Only)" on page 412
- "Pattern Matching Variables Tab (Event Rules Only)" on page 419
- "Indicators Tab" on page 410
- "Policy Variables Tab (Events Only)" on page 420

Configuring Options in XML 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 >** **XML**. The XML file policy editor opens.

Alternatively, double-click an existing XML file policy to edit it.

Click **Options** to open the policy Options page.

Tasks

How to configure options for XML 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 XML File Policies" above.

Related tasks

- "How to Collect Event Data from XML Files" on page 311

UI Descriptions

For a description of the Options page, see "Options Page (Events Only)" on page 417.

Chapter 11

Pattern Matching in Policy Rules

BSM Connector uses pattern matching in the following ways:

- To extract data from log files and make this data available as pattern matching groups.

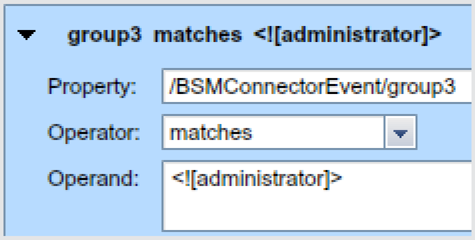
  A pattern matching group represents a unit of data that matches a specified pattern and that is then retained for use in event attributes, mappings, and rules.

  > **Example:** The log file pattern `(.*) (.*) (.*) (.*)` matches the log file lines
  >
  > ```
  > Info: user1 logged on
  > Info: user1 logged off
  > ```
  >
  > In this example, the parentheses surround the match pattern, and instruct the policy to retain the matched values as pattern matching groups. The patttern `.*` matches all characters, and the space is the delimiter in the log file.
  >
  > The pattern assigns the string `Info:` to the pattern matching group "group0", `user1` to "group1", `logged` to "group2", and the strings `on` and `off` to "group3". You can then insert the generated pattern matching groups "group0" to "groupn" in event attributes, mappings, and rules.

  For more information about the pattern matching syntax used to create pattern matching groups in log file policies, see "Regular Expressions" on page 217.

- To determine if an event should be sent to BSM or should be suppressed.

  BSM Connector compares the pattern that you specify with the data contained in XML elements or attributes, in pattern matching groups, in database columns, or in variable bindings. When the pattern is matched, the policy sends or suppresses an event, depending on the rule type.

  > **Example:** A log file policy contains a rule of the type "event on matched condition". A condition compares the pattern `<![administrator]>` with the value of "group3".
  >
  > 
  >
  > ▼ group3 matches <![administrator]>
  >
  > Property: /BSMConnectorEvent/group3
  >
  > Operator: matches
  >
  > Operand: <![administrator]>
  >
  > If the value of "group3" 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 BSM.
  >
  > For more information about the pattern matching syntax used to in rule conditions, see "Pattern Matching in Policy Rules".

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 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 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:

- Service ID
- Message type
- Category
- Application

- Object
- 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.

Chapter 12

Discovering 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 the following BSM applications:

- Operations Management (for example, for Topology-based Event Correlation (TBEC), context-specific tools, service health monitoring (Health Perspective))
- Service Health
- Service Health Analyzer
- Service Level Management

BSM Connector uses the following technologies to discover topology:

- **Data Flow Management (DFM).** DFM is a component of BSM that collects discovery data for BSM.

  BSM Connector uses DFM technology to collect topology based on DFM discovery scripts in policies. The policies report data based on the topology settings script you select and edit for the policy. The data they report is tightly integrated with BSM. You can create a custom topology script or use a predefined script to forward the relevant data.

  > **Note:** BSM Connector uses DFM technology internally; it does not include a full version of DFM and a DFM license is therefore not required.

- **Custom Topology Discovery and Synchronization.** Custom 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 on the BSM Connector system to the CI and CI relation model used in the RTSM. The processed and transformed topology data is then forwarded to BSM and stored in the RTSM.

  The advantage of using custom 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.

  For details, see "Local Topology Synchronization" on page 342.

- **HP Operations Agent Discovery Agent and Agent Repository (Legacy).** The discovery agent is an extension to the HP Operations Agent.

  The discovery agent runs discovery policies that execute discovery scripts on the BSM Connector server. The discovery agent stores the data that it discovers in the agent repository (agtrep), which is a local data store of configuration items and their relationships. The agent then publishes details of new, changed, and removed CIs to the BSM server, but does not resend details of unchanged CIs.

> **Note:** BSM Connector supports agent-based discovery only for discovery policies that have been migrated from BSM Integration Adapter or imported from HP Operations Manager (HPOM). You can edit migrated discovery policies. HPOM service auto-discovery policies cannot be edited because they are incompatible with the BSM Connector legacy discovery policy editor.

For details on HP Operations Agent discovery, see the HP Operations Agent Reference Guide.

Topology Discovery

BSM Connector log file, database, and Web service listener policies enable you to create topology data by mapping the data to events and metrics collected from log files, databases, and through the Web service listener. In addition, you can report topology data to BSM without reporting event or metrics data. To collect the data, you can select an out-of-the-box topology script or create your own custom script.

For data sources other than log files, databases, and the Web service listener, you can create topology policies that run custom topology scripts.

You develop topology scripts using the Jython language. For details on how to work in Jython, you can refer to these Web sites:

- http://www.jython.org
- http://www.python.org

How to Report Topology Data

BSM Connector provides several options for reporting topology data. The options that are available to you depend on the type of data you want to integrate and on the data source:

- **Topology mapped to event data from log files, databases, and the Web service listener.** Create a log file, database, or Web service listener policy for event integration and complete all pages, including the Topology page.

  For details, see the following tasks:

  "How to Collect Event Data from Log Files" on page 147

  "How to Collect Event Data from Databases" on page 115

  "How to Collect Event Data Through the Web Service Listener" on page 271

- **Topology mapped to metrics data from log files, databases, and the Web service listener.** Create a log file, database, or Web service listener policy for metrics integration and complete all pages, including the Topology page.

  For details, see the following tasks:

  "How to Collect Metrics Data from Log Files" on page 148

  "How to Collect Metrics Data from Databases" on page 116

  "How to Collect Metrics Data Through the Web Service Listener" on page 272

- **Topology without event or metrics data from log files, databases, and the Web service listener.** Create a log file, database, or Web service listener policy for topology integration.

  For details, see the following tasks:

  "How to Collect Topology Data from Log Files" on page 149

  "How to Collect Topology Data Only from Databases" on page 117

  "How to Collect Topology Data Through the Web Service Listener" on page 273

- **Topology without event or metrics data from other sources** (for example, SNMP, open message interface, and XML files). Create a custom topology policy:

  a. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

     Admin > Integrations > BSM Connector Integrations

     For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

  b. In the BSM Connector user interface, click ✳ in the toolbar. Then click **Topology >** **Custom**.

     Alternatively, double-click an existing policy to edit it.

  c. 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 Custom Topology Policy Properties" on the next page.

  d. In the **Topology** page, create a script that creates a topology in BSM's RTSM that represents your third-party systems.

     For details, see "Configuring Scripts in Custom Topology Policies" on the next page.

  e. Click **OK** to save the policy and close the editor.

  f. *Optional.* If the list of policies does not refresh automatically in the BSM Connector user interface, click ⟳ in the toolbar.

  > **Note:** BSM removes CIs from the RTSM if no topology data has been received for them over a period of time. To prevent BSM from deleting the CIs that BSM Connector has sent to BSM, BSM Connector regularly sends a list of active CIs to BSM in addition to the topology data. Inactive CIs are not sent to the RTSM and are marked for deletion in the RTSM.
  >
  > You can configure BSM Connector to always send CIs regardless of whether topology data has been received for them. This guarantees that CIs reported by BSM Connector are never deleted automatically from the RTSM. This configuration option is known as always touch mode and is recommended for BSM Connector topology-only policies that collect data from incremental data sources such as log files or databases. The first time the policies run, they collect all CIs from the data source. The next time the policies run, they collect new and changed CIs only and then forward only the additions and changes to BSM. The always touch mode ensures that the already existing, unchanged CIs are also reported and are thus not deleted.

> For more information, see "Touching of CIs in BSM Connector" on page 23.

Custom Topology Policy User Interface

This section includes:

- "Configuring Custom Topology Policy Properties" below
- "Configuring Scripts in Custom Topology Policies" below

Configuring Custom Topology 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 **Topology >** 📄 **Custom**.

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 Report Topology Data" on page 334

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 423.

Configuring Scripts in Custom Topology Policies

The topology page enables you to create a topology in BSM's RTSM by creating your own custom script.

To access

In the BSM Connector user interface, click ✱ in the toolbar. Then click **Topology >** 📄 **Custom**.

Alternatively, double-click an existing policy to edit it.

Click **Topology** to open the policy Topology page.

Learn More

This section includes:

- "General Script Guidelines" below
- "Additional Documentation" on the next page

General Script Guidelines

- The Jython language is sensitive to spaces and tabs. For more information on Jython, see http://wiki.python.org/moin/HowToEditPythonCode.

- You must leave the `import` section of the predefined topology scripts as is and only add to it.

- The main body of the script is mandatory and consists of:

  ```
  def DiscoveryMain(Framework)
  ```

  This main function is responsible for creating Object State Holder Vector (OSHV) results. It holds the CI and CI relationship data, and how to map the incoming samples to the CIs.

- Use the built-in "logger" to debug the topology scripts when samples arrive. You do this by modifying the level and type of information reported to the log file:

  a. Add the logger import statement before system_lib import statement to the topology script, for example:

     ```
     import logger
     …
     import system_lib
     ```

  b. Change the log file settings in the **<BSM Connector root directory>/conf/core/Tools/log4j/PlainJava/bac_integration.properties** file as follows:

     Open the **bac_integration.properties** file in a text editor and locate the following lines in the file:

     ```
     # Jython logger
     log4j.category.PATTERNS_DEBUG=${loglevel}, discovery.appender
     ```

     Change the argument of **log4j.category.PATTERNS_DEBUG** from **${loglevel}** to **DEBUG**, as follows:

     ```
     log4j.category.PATTERNS_DEBUG=DEBUG, discovery.appender
     ```

  c. Save the file. It may take a few seconds for the changes to take effect.

     The debug data is written to the **<BSM Connector root directory>/log/discovery.log** file.

- You may use the following code to set the `monitored_by` attribute on the CIs reported by the script:

```
ems_lib.addMonitoredByForThirdPartySoftware(Framework,computer)
```

- Use the following code to get the routing domain for a Node CI:

```
domainName = system_lib.getDiscoveryDomainByIP(Framework, <IP or
node DNS name>)
```

For more information on the routing domain attribute in RTSM, see the Data Flow Management Guide.

Additional Documentation

For general information on topology scripts, see "Create Jython Code" and "Developing Jython Adapters" in the RTSM Developer Reference Guide.

For information about Java classes that can be used in topology scripts, see "HP Data Flow Management API Reference" in the RTSM Developer Reference Guide.

Tasks

How to report topology without event or metrics data

1. Configure the frequency of the script.
2. Configure the script.

Related tasks

- "How to Report Topology Data" on page 334

UI Descriptions

For a description of the Topology page, see "Topology Page" on page 442.

Topology Troubleshooting and Limitations

This section describes troubleshooting and limitations when BSM Connector is enabled to report CIs and CIs relationships topology to BSM.

This section includes:

- "Opening Logs in BSM Connector to Debug Mode" on the next page
- "Opening Logs in BSM to Debug Mode" on the next page
- "No Topology Reported" on page 340
- "Jython Scripts Troubleshooting" on page 340
- "RTSM Troubleshooting" on page 341
- "Limitations" on page 341

Opening Logs in BSM Connector to Debug Mode

The following log files in BSM Connector contain information relating to the BSM integration.

- **<BSM Connector root directory>/logs/bac_integration/bac_integration.log**
- **<BSM Connector root directory>/logs/bac_integration/discovery.log**
- **<BSM Connector root directory>/logs/bac_integration/probeGW-taskResults.log**

  This log file does not exist by default. To create it, open it in debug mode as described below.

Open the relevant log files to debug mode using the instructions listed below.

To open logs in BSM Connector in debug mode:

1. In the **<BSM Connector root directory>/conf/core/Tools/log4j/PlainJava** folder, open the **bac_integration.properties** file and change the debug level to the following appenders:

```
# discovery logger
log4j.category.com.hp.ucmdb.discovery=DEBUG, discovery.appender
log4j.additivity.com.hp.ucmdb.discovery=false

# Jython logger
log4j.category.PATTERNS_DEBUG=DEBUG, discovery.appender
log4j.additivity.PATTERNS_DEBUG=false
```

2. 2. Add the following appender to the **probeGW-taskResults.log** file:

```
#################################################################################
# Topology task results
#################################################################################
log4j.category.ProbeTaskResultsLog=DEBUG, PROBE_TASK_RESULT_FILE
log4j.additivity.ProbeTaskResultsLog=false
log4j.appender.PROBE_TASK_RESULT_FILE=org.apache.log4j.RollingFileAppender
log4j.appender.PROBE_TASK_RESULT_FILE.MaxFileSize=10MB
log4j.appender.PROBE_TASK_RESULT_FILE.MaxBackupIndex=5
log4j.appender.PROBE_TASK_RESULT_FILE.File=../${log.file.path}/probeGW-taskResults.log
log4j.appender.PROBE_TASK_RESULT_FILE.Threshold=DEBUG
log4j.appender.PROBE_TASK_RESULT_FILE.Append=true
log4j.appender.PROBE_TASK_RESULT_FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.PROBE_TASK_RESULT_FILE.layout.ConversionPattern=<%d> %-4r [%-5p] (%F:%L) -
%m%n
log4j.appender.PROBE_TASK_RESULT_FILE.encoding=UTF-8
```

Opening Logs in BSM to Debug Mode

The following log files in BSM contain information relating to the integration with BSM Connector.

Open the relevant log files to debug mode using the instructions listed below.

- **<BSM root directory>/log/odb/odb/mam.autodiscovery.log**
- **<BSM root directory>/log/odb/odb/cmdb.reconciliation.log**
- **<BSM root directory>/log/odb/odb/cmdb.reconciliation.datain.ignored.log**
- **<BSM root directory>/log/odb/odb/discoveryservlet.log**

  This log file does not exist by default. To create it, open it in debug mode as described below.

To open logs in BSM (on the Data Processing server machine in a distributed environment) to debug mode:

Change the RTSM debug level in the following BSM files:

- **<BSM root directory>/odb/conf/log/reconciliation.properties**
- **<BSM root directory>/odb/conf/log/mam.properties**
- **<BSM root directory>/odb/conf/log/mam.web.properties**

No Topology Reported

1. Open all BSM Connector and BSM logs to debug mode. For details, see "Opening Logs in BSM Connector to Debug Mode" on the previous page and "Opening Logs in BSM to Debug Mode" on the previous page.
2. Check if there are **/bin** files left in the **<BSM Connector root directory>/cache/topologyresultsData/merged** directory.
3. Check for errors in the logs in the following order:
   - **bac_integration.log.** Topology is not sent due to general errors or syntax problems in the scripts.
   - **discovery.log.** Get the complete picture of the topology reported from BSM Connector to BSM via Discovery Agent.
   - **probeGW-taskResults.log.** Shows topology Discovery Agent actually sends to the BSM server after doing the necessary validation.

     If we see information written in this log but we don't see the topology in BSM/RTSM it means there were errors in server side (BSM side).
   - **mam.autodiscovery.log.** Discovery manager log on BSM. Responsible for handling all tasks from server side.
   - **cmdb.reconciliation.log.** Reconciliation mechanism is processing each bulk sent to BSM and performs identification process in order to identify similar object.
   - **cmdb.reconciliation.datain.ignored.log.** Topology that is ignored by server and is not created.
   - **discoveryservlet.log.** Contains servlet log which is responsible for getting the data from discovery agent (client) and send the reply (success/failure).
4. It is recommended to open logs that are in XML format (probeGW-taskResults.log) using an XML editor.

Jython Scripts Troubleshooting

The scripts that enable BSM Connector to act as a discovery probe are stored on the BSM server. When BSM Connector is registered to BSM, BSM Connector downloads the appropriate script from the BSM server. It then uses the script to perform the discovery while integrating the event and metrics data collected from the third-party system.

Use the following procedure to make sure that the required Jython scripts are downloaded and updated correctly:

- Check that the **<BSM Connector root directory>/discovery/scripts** directory is not empty.
- Check that there are two Jython scripts for each type, for example, **system_lib.py** and **system_lib.py.old**.

  If you encounter a script with the suffix ***.py.11**, the scripts were not updated correctly.

To download all scripts again, do the following:

1. Stop BSM Connector:

   Windows: Stop the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Stop the BSM Connector main process, type `/opt/HP/BSMConnector/stop`.

2. Delete all files in the following directory:

   <BSM Connector root directory>/discovery/hsqldb

3. Start BSM Connector:

   Windows: Start the **HP BSM Connector** service in the **Administrative Tools > Services**.

   Linux: Start the BSM Connector main process, type `/opt/HP/BSMConnector/start`.

RTSM Troubleshooting

To get the properties of a CI reported to a Data Flow database with JMX:

1. Open **http://<BSM_machine>:21212/jmx-console/** in a Web browser.
2. Click **UCMDB:service= Model Services**.
3. Invoke method: **retrieveObjectProperties**.

Limitations

If an integration discovers a node with multiple IP addresses, BSM Connector may create more than one instance of the node with different IP addresses attached to different instances. For example: for a node A with IP addresses A1 and A2, BSM Connector may create two nodes A:A1 and A:A2. This may happen in the following scenarios:

- Different IP addresses discovered in different topology runs of a policy.
- Discovery of a large bulk of topology data where different IP addresses appear in different places of the bulk (not together).

Possible solutions: To avoid or overcome the limitation, the following approaches are possible:

- To avoid the limitation, wherever possible, make sure that node information appears together in one place of the discovered topology bulk. After nodes with multiple IP addresses have been fully populated, they can be referenced using partial information; for example, after a node with several IP addresses has been created in the RTSM, only its name and a single IP address can be used to populate a running software CI for the node.

  For more information on the node identification and reconciliation rule, as well as the minimal set of attributes required to identify a node CI correctly, access **RTSM Administration > CI Type Manager > Select Node CIT > Details tab > Identification**.

- To overcome the limitation, apply manual reconciliation in RTSM.

  For more information about reconciliation in the RTSM, see the section on "Reconciliation" in the Data Flow Management Guide.

Local Topology Synchronization

Local topology synchronization enables you to filter and process the discovered data on the BSM Connector system before sending it to the RTSM. By applying mapping rules, you can transform the discovered data to CIs and CI relations that can then be stored in the RTSM.

You can also map and transform the discovered data by developing appropriate Jython scripts. However, mapping rules use XML syntax and are therefore easier to create and maintain. Keeping the model information separate from the discovery script also makes updating the files easier, if the discovery methods or the data model changes.

Local topology synchronization is not to be confused with *BSM Connector topology synchronization*. During BSM Connector topology synchronization, the BSM Connector server sends a list of active CIs to the RTSM to update their last access time in the RTSM. CIs that have had no activity over a period of time become deletion candidates in the RTSM. BSM Connector topology synchronization by default takes place once a week. You can change the interval in BSM (select **Admin > Integrations > BSM Connector Integrations**).

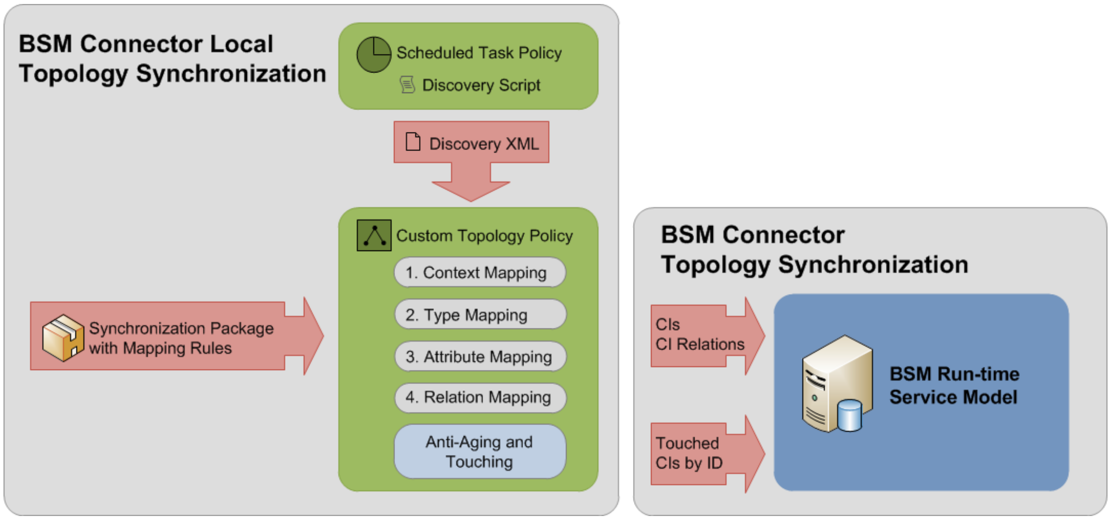
You can apply local topology synchronization only to topology data discovered by custom topology policies. Database, log file, Web service listener, and legacy service discovery policies do not support local topology synchronization.

Topology Synchronization Data Flow

Local topology synchronization relies on the discovered data being available in an XML file that conforms to the BSM Connector topology discovery syntax. To create the XML file, write a discovery script (using your preferred scripting or programming language) that can be run by a scheduled task policy according to a defined schedule.

The discovery XML is then transformed 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 also need a custom topology policy that applies the synchronization package to the discovery XML file. Each time the custom topology policy runs, the mapping rules are applied to the whole discovery XML file, not only to appended entries.

Before sending the mapped topology data to the RTSM, BSM Connector checks whether topology data has been received for a CI or not, and then updates its local cache of changed CIs. New and changed CIs are then sent to BSM and stored in the RTSM. The next time BSM Connector synchronizes the topology data with the RTSM (by default once a week), it sends the list of active CIs to BSM in addition to the topology data.



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 custom topology policy.

Topology Synchronization Packages

A synchronization package consists of the following files:

- **package.xml.** The package descriptor file defines 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 the RTSM. 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 the RTSM.
- **attributemapping.xml.** The attribute mapping file defines the mapping between the attributes of a discovered CI and the attributes of a CI in the RTSM.

  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 the RTSM 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 placed in the following directory on the BSM Connector system:

```
<BSM Connector root directory>/conf/topology/xml/sync-
packages/<synchronization package directory>/
```

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:

  `<`*`BSM Connector root directory`*`>/conf/topology/xml/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 BSM</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 the RTSM.

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 the RTSM.

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 the RTSM.

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 the RTSM.

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 the RTSM 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>
```

Topology Discovery Syntax

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

This section includes:

- "Configuration Item XML Schema Definition (XSD)" below
- "Configuration Item XML Element Description" on page 349
- "Example Discovery XML" on page 350

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="DeleteInstance" />
            <xs:element ref="NewRelationship" />
            <xs:element ref="DeleteRelationship" />
         </xs:choice>
      </xs:complexType>
      <xs:key name="InstanceKey">
```

```
         <xs:selector xpath="NewInstance|DeleteInstance">
         </xs:selector>
         <xs:field xpath="Key"></xs:field>
      </xs:key>
      <xs:keyref refer="InstanceKey" name="InstanceKeyRef">
         <xs:selector xpath="NewInstance|DeleteInstance">
         </xs:selector>
         <xs:field xpath="@ref"></xs:field>
      </xs:keyref>
      <xs:keyref refer="InstanceKey" name="InstanceRef">
         <xs:selector
xpath="NewRelationship/*/Instance|DeleteRelationship/*/Instance">
         </xs:selector>
         <xs:field xpath="@ref"></xs:field>
      </xs:keyref>
   </xs:element>
   <xs:element name="NewInstance" type="InstanceType" />
   <xs:element name="DeleteInstance" 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:element name="DeleteRelationship" 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.

Element	Description
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* and *DeleteInstance* elements. You must not specify a *NewInstance* and *DeleteInstance* with the same key in the same XML document.
Attributes	Contains *Attribute* elements.
Attribute	Has a `name` attribute and a `value` attribute.
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.
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.

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="contains">
        <Instance ref="myUniqueID2"/>
      </Relations>
      <Relations type="contains">
        <Instance ref="myUniqueID3"/>
      </Relations>
   </GenericRelations>
  </NewRelationship>
</Service>
```

How to Configure Local Topology Synchronization

1. **Create the discovery script.**

   a. Create a script (or program) that runs on the BSM Connector system 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.

      For details on the XML syntax, see "Topology Discovery Syntax" on page 347

b. *Optional:* Create a scheduled task policy that runs the discovery script according to a defined schedule.

   For details on scheduled task policies, see "Scheduling Tasks" on page 247.

2. **Create the synchronization 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 344.

   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 page 344.

   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 345.

   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 345.

   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 346.

   g. *Optional:* Use the supplied XML schema definitions to validate the correctness of the XML mapping files:

      ```
      <BSM Connector root directory>/conf/topology/xml/schemas
      ```

   h. Copy the directory that contains the XML mapping files from the temporary location to the following directory on the BSM Connector system:

```
<BSM Connector root directory>/conf/topology/xml/sync-packages/
```

3. Create a custom topology policy.

a. In the BSM Connector user interface, click ✳ in the toolbar. Then click **Topology >** 📄 **Custom**.

b. In the Properties page, type a name for the policy. You can use spaces in the name. The equal sign (=) is not allowed.

c. In the Topology page, configure the frequency of the script and the topology script itself.

> **Note:** An example topology script is available in a Jython file attached to this PDF. To view the attachment, select **View > Navigation Panels > Attachments**, and select **jython_toposync_custom_topo.py**.

In the `Globals` section of the example script, replace the following values:

```
configDir = "C:\\BSMConnector\\conf\\topology\\xml"
```

Replace "`C:\\BSMConnector\\conf\\topology\\xml`" with the path to the topology synchronization files on the BSM Connector system. By default, the files are installed to `<BSM Connector root directory/conf/topology/xml`.

```
inputFile = "C:\\temp\\test.xml"
```

Replace "`C:\\temp\\test.xml`" with the path and name of the discovery XML file that the discovery script creates. For details, see "Topology Discovery Syntax" on page 347.

```
runProps.setProperty("opr.toposync.taskName", "SyncMyTopology")
```

Replace "`SyncMyTopology`" with a name of your choice for the synchronization task you want to run.

```
runProps.setProperty("opr.toposync.selectedBundles",
"mysyncpkg")
```

Replace "`mysyncpkg`" with the name of your synchronization package. To specify more than one synchronization package, separate individual packages with semi-colons.

Synchronization packages must be stored in the following directory on the BSM Connector system:

```
<BSM Connector root directory>/conf/topology/xml/sync-
packages/
```

For details on custom topology scripts, see "Configuring Scripts in Custom Topology Policies" on page 336.

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" below
- "XPath Navigation" on page 372

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" below.

Mapping File Syntax

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

- "Rules" on the next page
- "Rule Conditions" on the next page
- "Operator Elements" on page 356
- "Operand Elements" on page 359
- "Mapping Elements" on page 365
- "Filtering" on page 366
- "Type Mapping" on page 367
- "Attribute Mapping" on page 368
- "Relation Mapping" on page 370

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 366 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 Operations Management. 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 Operations Management. 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 BSM.

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 372.

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 372.

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 the next page.

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

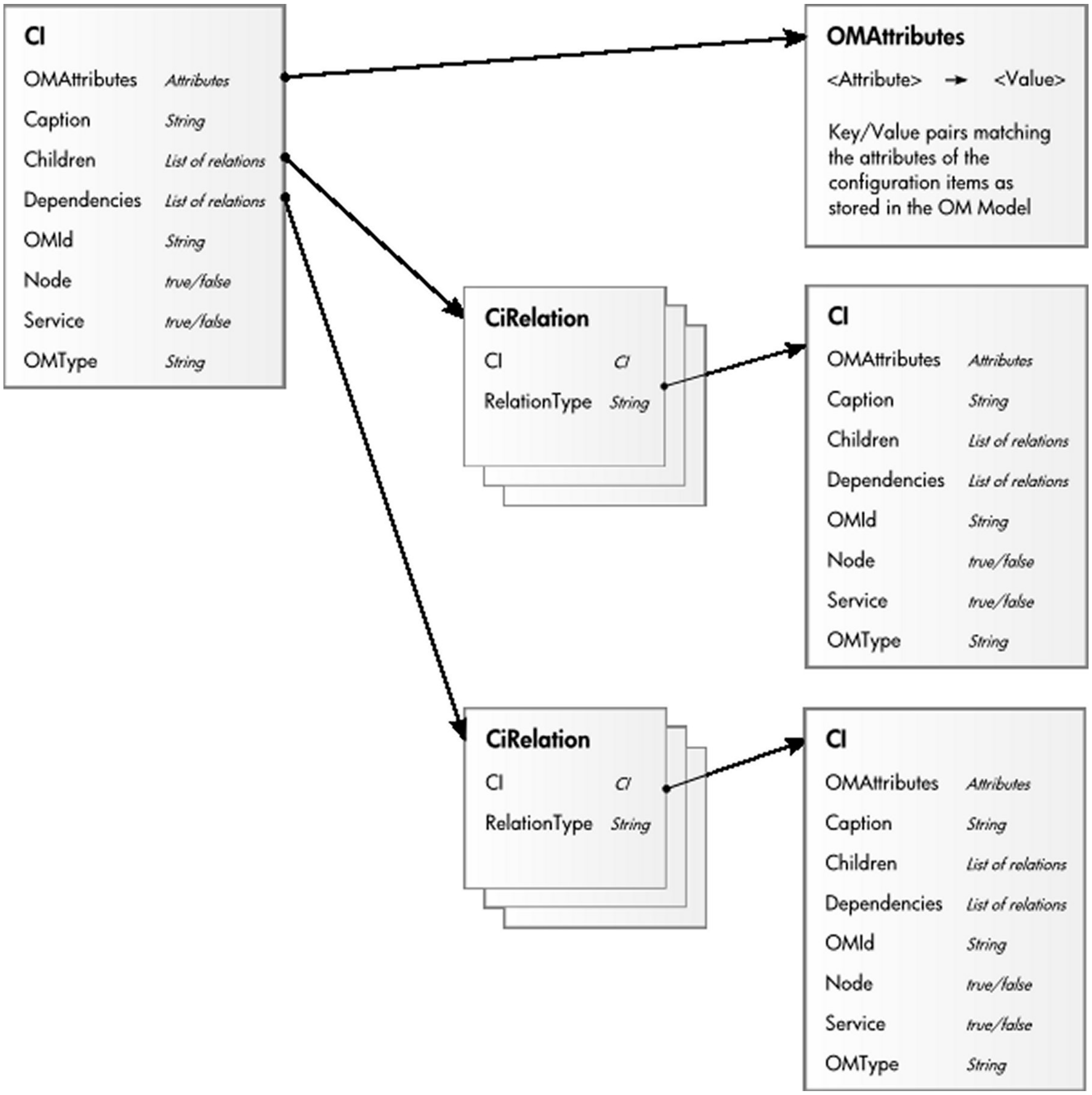
Relationship type as stored in the RTSM.

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`.

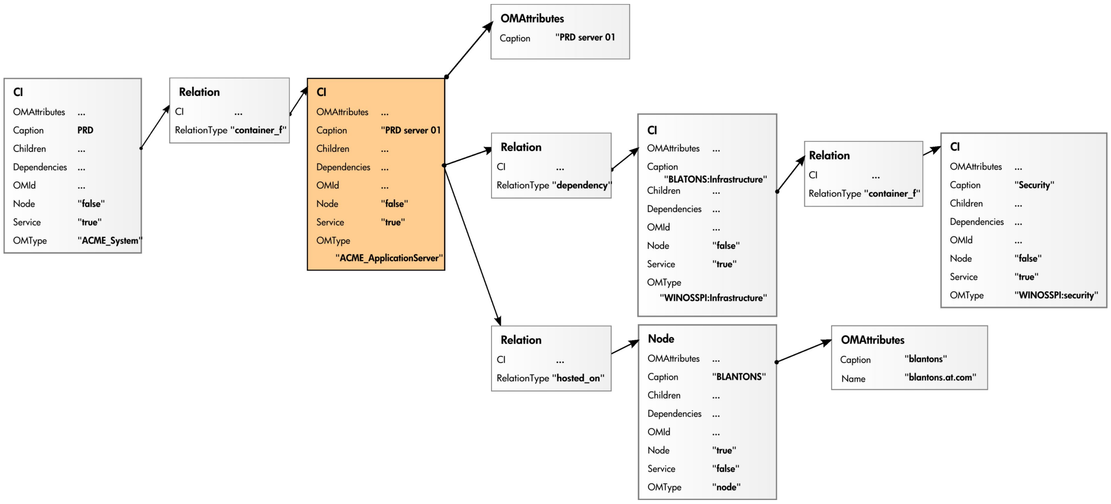
The following figure illustrates the data structure exposed to the navigation.



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 Expressions, Meaning and Examples

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.

Topology Synchronization Troubleshooting

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

This section includes:

- "Validating XML Mapping Files" below
- "Mapping Log Files" on the next page
- "Writing Rules" 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:

```
<BSM Connector root directory>/conf/topology/xml/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

To change the log level of the mapping engine, follow these steps:

1. Open the following file in a text editor:

   ```
   <BSM Connector root
   directory>/conf/core/Tools/log4j/PlainJava/toposyncsa.log4j.properties
   ```

2. Locate the line starting with `loglevel=`
3. Set the log level to any of the following values (for example, `loglevel=INFO`):

   `DEBUG` designates fine-grained informational events that are most useful to debug an application.

   `INFO` designates informational messages that highlight the progress of the application at coarse-grained level.

   `WARN` designates potentially harmful situations.

   `ERROR` designates error events that might still allow the application to continue running.

   `FATAL` designates very severe error events that will presumably lead the application to abort.

The mapping engine generates the following log file:

```
<BSM Connector root directory>/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.

Legacy Discovery

Legacy discovery in BSM Connector uses the discovery functionality of HP Operations Agent. The discovery agent runs discovery policies that execute discovery scripts on the BSM Connector server. These policies automatically populate the BSM Run-time Service Model (RTSM) with discovered configuration items (CIs) and CI relationships. Discovered attributes may be hardware resources, operating system attributes, applications, and other information that can be retrieved from an object and mapped to CI attributes in the (RTSM). The discovery scripts run according to a schedule that you specify for each policy.

> **Note:** BSM Connector supports agent-based discovery only for discovery policies that have been migrated from BSM Integration Adapter or imported from HP Operations Manager (HPOM). You can edit migrated discovery policies. HPOM service auto-discovery policies cannot be edited because they are incompatible with the BSM Connector legacy discovery policy editor.

How to Discover Topology Data

This task describes how to configure discovery policies.

1. *Prerequisite:* Make sure your BSM Connector is set up as a BSM Connector integration server in BSM:

   Admin > Integrations > BSM Connector Integrations

   For details on configuring a BSM Connector integration server in BSM, see the BSM online help or the BSM Application Administration Guide.

2. In the BSM Connector user interface, select a discovery policy and click ✏ in the toolbar, or double-click a discovery policy. The discovery policy editor opens.

3. Complete the information in the following tabs:

   - **Properties** include information that is related to the policy itself (for example, the name and description of the policy).
   - **Command** defines the script or program that the HP Operations agent runs to discover configuration items.
   - **Schedule** defines the time, date, and frequency of discovery.

4. Click **OK** to save the policy and close the editor.

Discovery Policy User Interface

This section includes:

- "Configuring Discovery Policy Properties" below
- "Configuring the Command in Discovery Policies" below
- "Configuring Schedules in Discovery Policies" on page 386

Configuring Discovery 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, select a discovery policy and click ✏ in the toolbar, or double-click a discovery policy. The discovery policy editor opens.

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 Discover Topology Data" on the previous page

UI Descriptions

For a description of the Properties page, see "Properties Page" on page 423.

Configuring the Command in Discovery Policies

Configuring a discovery policy includes creating a script (or program) that the HP Operations agent can run on the BSM Connector system to discover configuration items (CIs) and CI relations. This discovery script must write details of each discovered CI in XML to the standard output stream (STDOUT). The agent stores these details in the agent repository, which is a local data store of CIs that exist in the BSM Connector environment. The agent publishes details of new, changed, and removed CIs to the BSM RTSM, but does not resend details of unchanged CIs.

To access

In the BSM Connector user interface, select a discovery policy and click ✏ in the toolbar, or double-click a discovery policy. The discovery policy editor opens.

Click **Command** to open the policy Command page.

Learn More

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="DeleteInstance" />
            <xs:element ref="NewRelationship" />
            <xs:element ref="DeleteRelationship" />
         </xs:choice>
      </xs:complexType>
      <xs:key name="InstanceKey">
         <xs:selector xpath="NewInstance|DeleteInstance">
         </xs:selector>
         <xs:field xpath="Key"></xs:field>
      </xs:key>
      <xs:keyref refer="InstanceKey" name="InstanceKeyRef">
         <xs:selector xpath="NewInstance|DeleteInstance">
         </xs:selector>
         <xs:field xpath="@ref"></xs:field>
      </xs:keyref>
      <xs:keyref refer="InstanceKey" name="InstanceRef">
         <xs:selector
xpath="NewRelationship/*/Instance|DeleteRelationship/*/Instance">
         </xs:selector>
         <xs:field xpath="@ref"></xs:field>
      </xs:keyref>
   </xs:element>
   <xs:element name="NewInstance" type="InstanceType" />
   <xs:element name="DeleteInstance" 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:element name="DeleteRelationship" 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

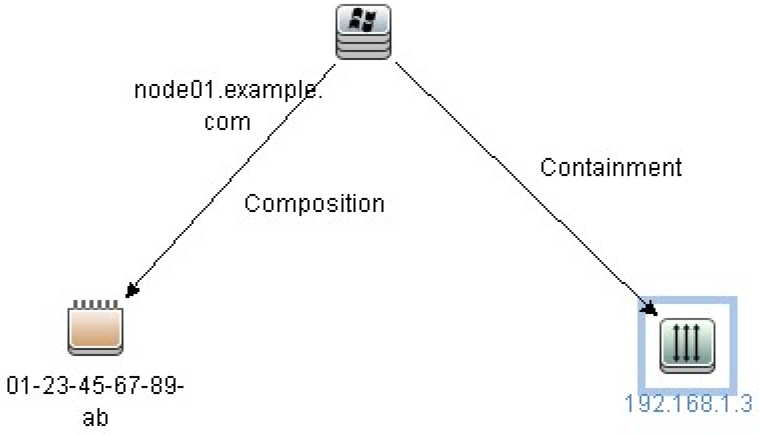
The following table describes the elements that the XML document can contain.

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* and *DeleteInstance* elements. You must not specify a *NewInstance* and *DeleteInstance* with the same key in the same XML document.
Attributes	Contains *Attribute* elements.

Element	Description
Attribute	Has a `name` attribute and a `value` attribute. Attributes with the following names have a special meaning: • `hpom_citype` specifies the CI type as stored in the RTSM (for example, `nt`). The `default` synchronization package on the BSM server assigns the context `IntegrationAdapter` to all CIs that have a `hpom_citype` attribute so that they are included for topology synchronization. CIs that do not have this attribute are filtered out and excluded from topology synchronization. • `hpom_rootcontainer` specifies the full ID of the CI that contains or hosts this CI. Maps to the CI attribute `Container`. Creates a composition relationship. • 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`).
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. 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.

Example Discovery XML

The following example XML creates a node instance with related IP address and MAC address instances. The IP address instance has a containment relationship and the MAC address has a composition relationship to the node instance:



```
<Service>
   <NewInstance>
      <Std>DiscoveredElement</Std>
      <Key>CBA3A4AC-2EC5-11E0-8071-4876DFD72085</Key>
      <Virtual />
      <Attributes>
         <!-- use nt as CI type attribute -->
         <Attribute name="hpom_citype" value="nt" />
         <!-- use node01 as name attribute -->
         <Attribute name="ucmdb_name" value="node01" />
         <!-- use node01.example.com as primary_dns_name attribute -->
         <Attribute name="ucmdb_primary_dns_name"
value="node01.example.com" />
         <!-- use node01.example.com as user_label attribute -->
         <Attribute name="ucmdb_user_label" value="node01.example.com"
/>
         <!-- use "Windows 2003 R2 64 Bit" as discovered_os_name
attribute -->
         <Attribute name="ucmdb_discovered_os_name" value="Windows
2003 R2 64 Bit" />
      </Attributes>
   </NewInstance>
   <NewInstance>
      <Std>DiscoveredElement</Std>
      <Key>E0AD966E-2EC5-11E0-B5C8-4E76DFD72085</Key>
      <Virtual />
      <Attributes>
         <Attribute name="hpom_citype" value="ip_address" />
         <!-- use 192.168.1.3 as name attribute -->
         <Attribute name="ucmdb_name" value="192.168.1.3" />
         <Attribute name="ucmdb_routing_domain" value="DefaultDomain"
/>
      </Attributes>
   </NewInstance>
   <NewInstance>
      <Std>DiscoveredElement</Std>
      <Key>157C8328-2EC6-11E0-931A-C176DFD72085</Key>
      <Virtual />
      <Attributes>
         <!-- relation to the MAC address is composition  -->
         <!-- this is handled by the attribute hpom_rootcontainer  -->
         <Attribute name="hpom_rootcontainer" value="CBA3A4AC-2EC5-
11E0-8071-4876DFD72085" />
         <Attribute name="hpom_citype" value="interface" />
         <Attribute name="ucmdb_name" value="Intel(R) PRO/1000 MT
Network Connection" />
         <Attribute name="ucmdb_mac_address" value="01-23-45-67-89-ab"
/>
         <Attribute name="ucmdb_interface_description" value="Intel(R)
PRO/1000 MT Network Connection" />
         <Attribute name="ucmdb_description" value="Intel(R) PRO/1000
MT Network Connection" />
      </Attributes>
   </NewInstance>
   <NewRelationship>
   <!-- create relationship of type containment between node (node01)
 -->
   <!-- and IP address (192.168.1.3)   -->
      <Parent>
         <Instance>
               <Key>CBA3A4AC-2EC5-11E0-8071-4876DFD72085</Key>
               <Virtual/>
               <Std>DiscoveredElement</Std>
         </Instance>
      </Parent>
      <GenericRelations>
         <Relations type="containment">
            <Instance>
               <Key>E0AD966E-2EC5-11E0-B5C8-4E76DFD72085</Key>
               <Virtual/>
               <Std>DiscoveredElement</Std>
            </Instance>
         </Relations>
      </GenericRelations>
   </NewRelationship>
</Service>
```

Tasks

How to configure the command in discovery policies

1. Type the name of your discovery script in the **Command** box. For example, you could specify the command line "`$ACTION_DIR/custom_discovery.cmd`".

`$ACTION_DIR` represents the folder that contains instrumentation on the BSM Connector system:

Windows: `%OvDataDir%\bin\instrumentation`

Linux: `/var/opt/OV/bin/instrumentation`

Escape any backslashes (\) with a second backslash (\\).

> **Tip:** Click ▸ in the Command box to open the edit assistant. The edit assistant enables you to insert and remove quotation marks, and to insert the variables `$ACTION_DIR` and `$DATA_DIR` more easily.

2. *Optional.* By default, the HP Operations Agent starts the discovery script under the same account as the agent is running under, which is Local System or root by default. You can specify a different user and password if you want the script to run under a different account.

> **Tip:** The maximum frequency that you can schedule for a discovery policy is hourly. This frequency may not be convenient when you are developing and testing your policy. However, after you deploy your policy to a node, you can run the policy on demand using the command `ovagtrep -run <policy name>` on the node.

Related tasks

- "How to Discover Topology Data" on page 378

UI Descriptions

For a description of the Command page, see "Command Page (Legacy Discovery Policies)" on page 390.

Configuring Schedules in Discovery Policies

Use the Schedule tab of the discovery policy editor to specify a discovery schedule for the current policy. By default, a discovery policy runs the discovery script every Sunday at 00:00.

> **Tip:** You can associate a single schedule only with a discovery policy. To run the same discovery script according to different schedules, create a policy for each schedule. This enables you to, for example, run a discovery script Monday to Friday at 15:00, and Saturday and Sunday at 16:00.

To access

In the BSM Connector user interface, select a discovery policy and click ✎ in the toolbar, or double-click a discovery policy. The discovery policy editor opens.

Click **Schedule**.

Tasks

How to configure the schedule in discovery policies

Indicate when the discovery script should run:

- **Days of Week.** Specify the day or multiple days of the week that the discovery script should run.

  To specify the day of the week, click the day in the **Days of Week** display. To select multiple days, click a day, press **Ctrl** or **Shift**, and select additional days. A dark-gray bar indicates your choice.

- **Hours of Day.** Specify the specific hour or multiple hours of day that the discovery script should run.

  To specify the hour of the day, click the hour in the **Hours of Day** display. To select multiple hours, click an hour, press **Ctrl** or **Shift**, and select additional hours. A dark-gray bar indicates your choice.

- **Minute of Hour.** Specify the minute or multiple minutes within the hour that the discovery script should run. The maximum frequency is once in five minutes.

  To specify the minute of the hour, click one of the five-minute intervals in the **Minute of Day** display. To select multiple intervals, click an interval, press **Ctrl** or **Shift**, and select additional intervals. A dark-gray bar indicates your choice.

> **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 ✖.

Your choices are summarized in the **Schedule Summary** at the bottom of the page.

Related tasks

- "How to Discover Topology Data" on page 378

UI Descriptions

For a description of the Schedule page, see "Schedule Page - Legacy Discovery Policies" on page 429.

Appendix 13

UI Descriptions

Command Page (Legacy Discovery Policies)

UI Element	Description
Command	Name of the discovery script. Use the variable `$ACTION_DIR` to represent the folder that contains instrumentation on managed nodes. For example, you could specify the command line "`$ACTION_DIR/custom_discovery.cmd`". `$ACTION_DIR` represents the folder that contains instrumentation on the BSM Connector system: Windows: `%OvDataDir%\bin\instrumentation` Linux: `/var/opt/OV/bin/instrumentation` Escape any backslashes (\) with a second backslash (\\). **Tip:** Click ▸ in the Command box to open the edit assistant. The edit assistant enables you to insert and remove quotation marks, and to insert the variables `$ACTION_DIR` and `$DATA_DIR` more easily.
Username	User name of the account under which the discovery script is started. By default, the agent starts the service discovery script under the same account as the agent is running under, which is Local System by default.
Password	Password of the account under which the discovery script is started.

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.

UI Element	Description
Severity	Severity assigned to the event.
Severity	Severity assigned to the event (Critical, Major, Minor, Warning, Normal, Unknown).
Severity	Severity assigned to the event. Accept the severity that is set in the event defaults or choose a specific event severity: Critical, Major, Minor, Warning, Normal.
Time Created	*Database, log file, 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) • **Pattern matching.** You can use the function `<$DATETIME(FORMAT, VALUE)>` to specify a pattern (`FORMAT`) that matches the time string in the policy source (`VALUE`). You can use standard pattern-matching rules when matching values. By default, pattern matching for the time format is case sensitive. The default field separators are the space and the tab characters. `FORMAT` must be enclosed in quotation marks (`"FORMAT"`) and accepts the following variables: `H` (hours), `M` (minutes), `S` (seconds). If H, M, or S is not set, the hour, minute, or second displays as zero. `d` (day), `m` (month), `y` (year). If d or m is not set, the day or month display as one. If y is not set, the current year is assumed. If y is less than 100, the current millennium is assumed; for example, if y matches 10, the year displays as 2010. It is not possible to match a year earlier than 1970. `p` (P.M.) If p is set, BSM Connector adds 12 hours to the hours that precede the variable. `VALUE` is one of the following: ▪ **Database policies.** `VALUE` is the table column or value to match. Table columns use the following syntax: <$DATA:/BSMConnectorEvent/<*table_column*>> <*table_column*> is the name of the table column in the third-party database.

UI Element	Description
	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>)>` - **Log file policies.** `VALUE` is the pattern matching group or value to match. Pattern matching groups use the following syntax: <$DATA:/BSMConnectorEvent/<*pattern_matching_group*> <*pattern_matching_group*> is the name of the group assigned to a particular content match. Examples: To match the time format `06/15/2010 11:12:13` in the pattern matching group `<$DATA:/BSMConnectorEvent/group3>`, type `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M>:<#.S>$", <$DATA:/BSMConnectorEvent/group3>)>` To match the time format `11:12 15.06.2010` in the pattern matching group `<$DATA:/BSMConnectorEvent/group3>`, type `<$DATETIME("^<#.H>:<#.M> <#.d>.<#.m>.<#.y>$", <$DATA:/BSMConnectorEvent/group3>)>` To match the time format `06/15/2010 1:35 PM` in the pattern matching group `<$DATA:/BSMConnectorEvent/group3>`, type `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M> <2*.p>$", <$DATA:/BSMConnectorEvent/group3>)>` - **Web service listener policies.** `VALUE` is the <key> element or value to match. <key> elements use the following syntax: <$DATA:/BSMConnectorEvent/<*key*>> <*key*> is the string assigned to the <key> element in the SOAP message (for example, `<key xsi:type="soapenc:string">Severity</key>`) Examples:

UI Element	Description
	To match the time format `06/15/2010 11:12:13` in the <key> element `<$DATA:/BSMConnectorEvent/TIMERECEIVED>`, type `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M>:<#.S>$", <$DATA:/BSMConnectorEvent/TIMERECEIVED>)>` To match the time format `11:12 15.06.2010` in the <key> element `<$DATA:/BSMConnectorEvent/TIMERECEIVED>`, type `<$DATETIME("^<#.H>:<#.M> <#.d>.<#.m>.<#.y>$", <$DATA:/BSMConnectorEvent/TIMERECEIVED>)>` To match the time format `06/15/2010 1:35 PM` in the <key> element `<$DATA:/BSMConnectorEvent/TIMERECEIVED>`, type `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M> <2*.p>$", <$DATA:/BSMConnectorEvent/TIMERECEIVED>)>` ■ **XML file policies.** `VALUE` is the XML property or value to match. XML properties use the following syntax: <$DATA:/<*XML_property*>> <*XML_property*> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute. Examples: To match the time format `06/15/2010 11:12:13` in the XML property `<$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 BSM. 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.
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)).

UI Element	Description
ETI	Contains the event type indicator (ETI) resolution hint, which BSM 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 BSM 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 related CI hint, which BSM uses to identify the CI related to the event (for example, oraclesid01@@node.example.com or C:@@server.example.com). Use the format `<hint 1>:<hint 2>:...:<hint n>@@<hostname>`.
	Contains the CI hint, which BSM uses to identify the CI related to the metric (for example, oraclesid01@@node.example.com or C:@@server.example.com). Use the format `<hint 1>:<hint 2>:...:<hint n>@@<hostname>`.
	Best practices for CI hints
	The related CI hint should have sufficient hints to find the corresponding CI.
	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 `webspheréas` 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 as a hint using the format `UCMDB:<ci_uuid>`.
	For more information about CI resolution in BSM, see the *Using Operations Management* PDF 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 BSM'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 CI hint. 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 CI hint, BSM 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 Operations Management. **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	Sets the event's lifecycle status to Closed before sending it to the Event Browser in Operations Management.
Send with closed status	Sets the event's lifecycle status to Closed before sending it to the Event Browser in Operations Management. 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.

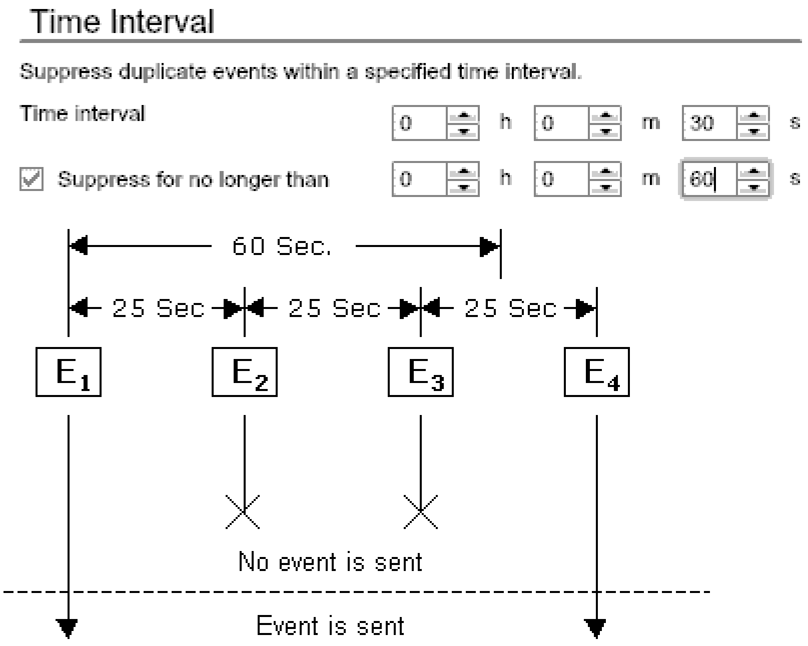
In scheduled task policies, you can set only the following event correlation attributes:

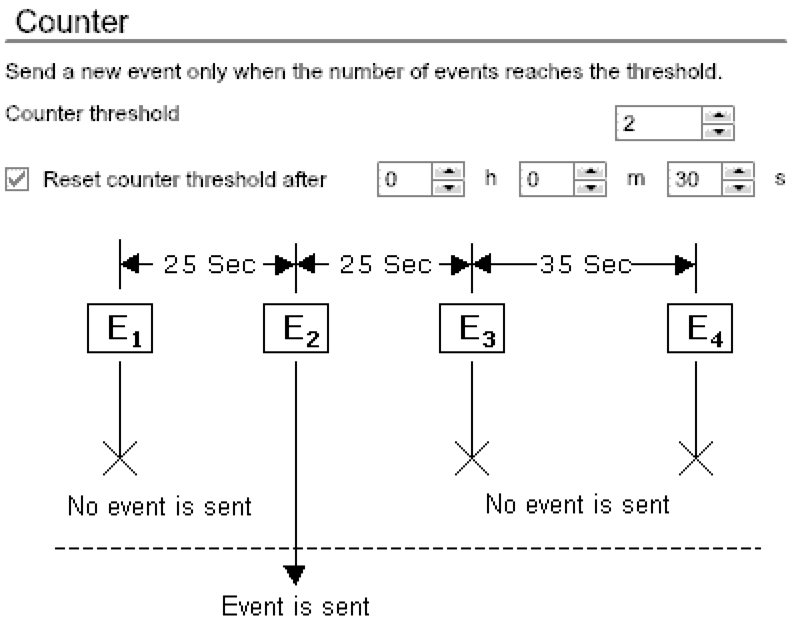
- Event Key
- Suppress Deduplication on Server

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 Operations Management Event Browser 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>:<5*>`
	This pattern is evaluated by first replacing the variables with the values that they resolve to, for example:
	`critical:cabbage.example.com:<5*>`
	This pattern is then compared using pattern matching rule against the event keys for all events in the Operations Management Event Browser. The pattern above would match the following event keys:
	`critical:cabbage.example.com:12345`
	`critical:cabbage.example.com:TEST1`

Suppress Deduplication on Server	Stops automatic discarding of new events that are duplicates of existing events.

UI Element	Description
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.  Time Interval Suppress duplicate events within a specified time interval. Time interval 0 h 0 m 30 s ☑ Suppress for no longer than 0 h 0 m 60 s 60 Sec. 25 Sec 25 Sec 25 Sec E_1 E_2 E_3 E_4 No event is sent Event is sent

UI Element	Description
	The $\boxed{E_x}$ 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**  Counter Send a new event only when the number of events reaches the threshold. Counter threshold 2 ☑ Reset counter threshold after 0 h 0 m 30 s 25 Sec 25 Sec 35 Sec E_1 E_2 E_3 E_4 No event is sent No event is sent Event is sent The $\boxed{E_x}$ 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,

UI Element	Description
	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 the Operations Management Event Browser. **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_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.

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 URL	*Rules only:* 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 (for example, `http://nnm-i.example.com:8004/nnm/launch?cmd=showForm&objtype=Incident&objuuid=$OPC_CUSTOM[nnm.incident.uuid]&menus=true`). Event drilldown information enables BSM 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 BSM 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.
Appli-cation	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:* 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 BSM to identify the node and CI associated with the event.

UI Element	Description
Use event default	*Rules only:* Indicates the agent MSI settings configured in the event defaults: **Agent MSI not set.** 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.
Agent MSI	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	Divert an event to the MSI instead of to the server when an event is requested by an external application.
Copy events	Send 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
- Log file policies
- 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:* Table column that the policy searches for. You must prefix the column name with `/BSMConnectorEvent/` (for example, `/BSMConnectorEvent/Severity`). *Log file policies:* Pattern matching group that the policy searches for. You must prefix the group name with `/BSMConnectorEvent/` (for example, `/BSMConnectorEvent/group0`). *Web service listener policies:* <key> element that the policy searches for. You must prefix the <key> element with `/BSMConnectorEvent/` (for example, `/BSMConnectorEvent/Severity`). *XML file policies:* XML property 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`).
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.)

UI Element	Description
Operand	*Database policies:* 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 Table Column Values list and drop it in the Operand field. *Log file policies:* Value or pattern that you want the policy to compare with the pattern matching group. If you are working with sample data, you can drag the value from the Pattern Matching Group Values list and drop it in the Operand field. *Web service listener policies:* Value or pattern that you want the policy to compare with the <key>. *XML file policies:* 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 XML Values list and drop it in the Operand field. **Tip:** You can use standard BSM Connector pattern-matching rules when matching values. Select the matches operator and click ▶ in the Operand field to open the pattern matching expression toolbox. The toolbox displays the following: • **Pattern Matching Expressions.** Click an expression to insert it in the Operand field. • **Variable Bindings Options.** Variable bindings options include case sensitivity and field separators for the rule. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used. See also "Configuring Options in Log File Policies" on page 177"Configuring Options in XML File Policies" on page 322"Configuring Options in Database Policies" on page 136"Configuring Options in Web Service Listener Policies" on page 296"Configuring Options in Open Message Interface Policies" on page 245.

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. **Tip:** You can use standard BSM Connector pattern-matching rules when matching values. Select the matches operator and click ▸ in the Operand field to open the pattern matching expression toolbox. The toolbox displays the following: • **Pattern Matching Expressions.** Click an expression to insert it in the Operand field. • **Variable Bindings Options.** Variable bindings options include case sensitivity and field separators for the rule. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used. See also "Configuring Options in Log File Policies" on page 177"Configuring Options in XML File Policies" on page 322"Configuring Options in Database Policies" on page 136"Configuring Options in Web Service Listener Policies" on page 296"Configuring Options in Open Message Interface Policies" on page 245.

Condition Definition Tab - SNMP Trap 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 Trap. 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 Trap 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. **Tip:** You can click the ▶ button to open the pattern matching expression toolbox.

Defaults and Rules Pages (Metrics)

Default Monitor Attributes and Attributes Tabs

Most metrics attributes are represented by string values and must be enclosed in quotation marks (""). You can optionally use the plus sign (+) to add group values (for example, "Disk Usage on " + $group0). For details, see "UI Descriptions" on page 389.

The following attributes are represented by numeric values and must *not* be enclosed in quotation marks:

- Time stamp
- Measurement value

Tip: Use the parseDouble() function to convert a quoted string value to a numeric value (for example, parseDouble("123123"). Alternatively, use parseDouble($group0) if the value in the log file is numeric and without quotation marks. (You may use pattern matching to remove quotation marks from a log file while it is parsed by the policy.)

- Quality

> **Tip:** The Policy Constants tab lists the available constants representing metrics quality (status) that are available in BSM Connector. Use drag and drop to add them to the **Quality** attribute.

UI Element	Description
Monitor Name	Name of a monitor representing a group of metrics that have something in common. **Example:** "Windows CPU Monitor on"+group0
Monitor Type	The monitor type/category. **Example:** "My Integration CPU Monitor "
Target	The target of this monitor (such as the name of a node). **Example:** "mynode.mycompany.com"
Time Stamp	Time stamp in the seconds since Jan 1st 1970 format. Must be a double-precision number. **Example:** parseDouble("123123")
Quality	Metrics quality (status) in BSM Connector terms. Possible values are: QUALITY_ERROR, QUALITY_WARNING, QUALITY_GOOD. **Example:** $group4.equals("error") ? QUALITY_ERROR :$group4.equals("warning") ? QUALITY_WARNING : QUALITY_GOOD
Monitor State	The monitor status. **Example:** "Running..."
Metrics	
✳	**New Item:** Creates new metrics.
✖	**Delete Item:** Deletes the selected metrics.
↑	**Move Up:** Moves the selected metrics higher in the metrics order.
↓	**Move Down:** Moves the selected metrics lower in the metrics order.
⊞	**Expand:** Expands the list of metrics to display all details.
⊟	**Collapse:** Collapses the list of metrics to display only the names and hide the details.
Name	Name the Nth metric. **Example:** "CPU Utilization"

UI Element	Description
Value	Value of Nth metric. Must be a double-precision number. **Example:** 80

CI Hint	Contains the related CI hint, which BSM uses to identify the CI related to the event (for example, oraclesid01@@node.example.com or C:@@server.example.com). Use the format `<hint 1>:<hint 2>:...:<hint n>@@<hostname>`.
	Contains the CI hint, which BSM uses to identify the CI related to the metric (for example, oraclesid01@@node.example.com or C:@@server.example.com). Use the format `<hint 1>:<hint 2>:...:<hint n>@@<hostname>`.
	Best practices for CI hints
	The related CI hint should have sufficient hints to find the corresponding CI.
	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 as a hint using the format `UCMDB:<ci_uuid>`.
	For more information about CI resolution in BSM, see the *Using Operations Management* PDF or online help.
	Note: CI hints are not required when using the Computer - Monitor topology script because the script sets them automatically. However, it is recommended to specify CI hints for custom, Computer, and Computer - Running Software topology scripts.

Indicator Name	Indicator name. Must be enclosed in quotation marks (""). **Example:** "CPULoad"

Condition Tab (Rules Only)

UI Element	Description
Match	Match condition.

Indicators Tab

UI Element	Description
🔄	**Refresh.** Loads the configured indicators from the connected BSM server. **Note:** - Loading indicators from the BSM server may take a few seconds. - The BSM Connector server must be configured as a BSM Connector integration server in BSM for the indicators to load successfully. See also "Indicator definitions could not be loaded" on page 112
🔧	*Event integration policies only:* Shows or hides the **Select drop target format** drop-down panel: - **Use Indicator States.** Click to change the drop target format to indicator states only. - **Use Indicator Names and States.** Click to change the drop target format to indicator names and states.
<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) and event type indicators (ETIs). To insert an indicator in a policy, drag and drop the indicator from the Indicators tab to the relevant field in the policy.

Mappings Page (Events Only)

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.
Data Input 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. *Log file policies:* Pattern matching group assigned to the custom variable. BSM Connector replaces the pattern matching group at runtime with the value of the specified group. If you insert a value, the value will be used. *Web service listener policies:* Key assigned to the custom variable. BSM Connector replaces the key at runtime with the value of the specified key. 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:/<*XML_property*>> <*XML_property*> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute. 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.
✱	**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. *Log file policies:*Original value of the pattern matching group. *Web service listener policies:*Original value associated with the <key> element in the SOAP message. *XML file policies:*Original value of the XML element or attribute.

Target Value	*Database policies:*New value of the table column. *Log file policies:*New value of the pattern matching group. *Web service listener policies:*New value associated with <key>. *XML file policies:*New value of the XML element or attribute.

Mappings Tab (Events Only)

UI Element	Description
<Mappings>	Displays the mapping definitions configured for the policy. For details, see "Configuring Mappings in XML File Policies" on page 314"Configuring Mappings in Log File Policies " on page 153"Configuring Mappings in Database Policies" on page 125"Configuring Mappings in Web Service Listener Policies " on page 283.

Operations Tab (Metrics Only)

Expressions and Functions	Description
? :	Conditional operator. Allows usage of conditions and branching logic: "someCondition ? value1 : value2" means "If someCondition is true, assign the value of value1 to result. Otherwise, assign the value of value2 to result." **Example:** $group1.equals("error") ? true : false
+	String concatenation. **Example:** "trap type is " + $trap
<, <=, > , >=, ==, !=	Checks the numerical correctness of the expression. Can be used with numeric values. **Example:** Match: $numberOfLines == 100
&&, ||	To be used to combine any of the above boolean expressions. **Example:** Match: $status.equals("ERROR") \|\| ($numberOfLines == 100)
true, false	Constant Boolean values. **Example:** Match: true
( )	Allows grouping of operators in order to change the order of their execution. **Example:** Monitor Name: ($group1.equals("CPU") ? "CPU Utilization" : $group1.equals("Memory") ? "Memory Usage" : "Unknown") + " " + $group2

Expressions and Functions	Description
boolean contains (String)	Returns true if and only if this string contains the specified sequence of char values. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#contains(java.lang.CharSequence). **Example:** Monitor Name: $group0.contains("monitor")? $group0 : $group0 + "monitor"
boolean endsWith (String)	Tests if this string ends with the specified suffix. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#endsWith(java.lang.String). **Example:** Monitor Name: $group1.endsWith("Operations")? $group1 : $group1 + "Operations"
boolean equals (String)	Compares this string to another string. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#equals(java.lang.Object). **Examples:** Match: "ERROR".equals($status) or Match: $status.equals("ERROR")
boolean equalsIgnoreCase (String)	Compares this String to another String, ignoring case considerations. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#equalsIgnoreCase%28java.lang.String%29. **Examples:** Match: "ERROR".equalsIgnoreCase($status) or Match: $status.equalsIgnoreCase("ERROR")
boolean exists (String)	Checks for an existence of a property in the processed event and make sure that it is not an empty value. **Example:** Match: exist($status)
String getToken (String, String ,int)	Splits input string according to a supplied delimiter (in regular expression format), and returns one of the result strings according to a specified zero-based index. **Example:** getToken($var, "/", 1) will produce `"y"` if $var equals "x/y/z"

Expressions and Functions	Description
int indexOf (String)	Returns the index within this string of the first occurrence of the specified substring. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#indexOf(int). **Example:** Severity: $group0.lastIndexOf("Critical")>-1? "Critical" : "Normal"
int indexOf (String, int)	Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#indexOf(java.lang.String,%20int). **Example:** Severity: $group0.indexOf("Critical",3)>-1? "Critical" : "Normal"
boolean isDouble (String)	Checks if the input string can be interpreted as a double number. **Example:** Match: isDouble($size)
boolean isEmpty()	Tests for an empty string (length() == 0). For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#isEmpty(). **Example:** Description: $group1.isEmpty()?$group0 :$group1
boolean isInt (String)	Checks if the input string can be interpreted as an integer number. **Example:** Match: isInt($size)
int lastIndexOf (String, int)	Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#lastIndexOf(java.lang.String,%20int). **Example:** Severity: $group0.lastIndexOf("Critical",2)>-1? "Critical" : "Normal"
int lastIndexOf (String)	Returns the index within this string of the rightmost occurrence of the specified substring. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#lastIndexOf(java.lang.String). **Example:** Severity: $group0.lastIndexOf("Critical")>-1? "Critical" : "Normal"

Expressions and Functions	Description
int length()	Returns the length of this string. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#length(). **Example:** Description: $group1.length() <10 ? $group0+$group1 :$group1
boolean matches (String)	Tells whether or not the string matches the given regular expression. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#matches(java.lang.String). **Example:** Severity: $group0.matches("(.*)Critical(.*)")? "Critical" : "Normal"
double parseDouble (String)	Use to convert strings to numeric values. The input string should be a valid representation of an integer or a floating point number. **Note:** Calling this function on a string that cannot be interpreted as a number causes an error and the incoming data is dropped. **Example:** Match: parseDouble($size) > 10
int parseInt (String)	Use to convert strings to numeric values. The input string should be a valid representation of an integer or a floating point number. **Note:** Calling this function on a string that cannot be interpreted as a number causes an error and the incoming data is dropped. **Example:** Match: parseInt($size) > 10
String resolveHostIP (String)	Performs DNS resolution from a server to its IP address. If the DNS resolution fails, the function returns the value `unknown host`. **Example:** Target: resolveHostIP($host)
String resolveHostName (String)	Performs DNS resolution from an IP address to a fully qualified domain name. If the DNS resolution fails, the function returns the original input host name. **Example:** Target: resolveHostName($host)
boolean startsWith (String)	Tests if this string starts with the specified prefix. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#startsWith(java.lang.String). **Example:** Monitor Name: $group1.startsWith("Operations")? $group1 : "Operations"+$group1

Expressions and Functions	Description
boolean startsWith (String, int)	Tests if the substring of this string beginning at the specified index starts with the specified prefix. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#startsWith(java.lang.String,%20int). **Example:** Monitor Name: $group1.startsWith("Operations",2)? $group1 : "Operations" + $group1
long str_to_ seconds (String, String)	Calculates the timestamp (in seconds, since January 1, 1970 format) held in the first String using the format in the second string. True if the date specified in `$time` in `yyyy-MM-dd HH:mm:ss.SSS` format is later than the current time. For more details, see http://download.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html. **Example:** Match: str_to_seconds ($time,"yyyy-MM-dd HH:mm:ss.SSS") > time() **Note:** Use the following symbols to represent time: Year - \`y'; Month - \`M"; Day of month - \`d'; Hour - \`H'; Minute - \`m'; Second - \`s'
String substring (int)	Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#substring(int). **Example:** Title: $group0.substring(2)
String substring (int, int)	Returns a new string that is a substring of this string. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#substring(int). **Example:** Title: $group0.substring(2,9)
int time()	Returns the current time, in seconds, since January 1, 1970 format. **Example:** Match: $timeStampField > (time()-600) True if the value of the `$timeStampField` is newer then ten minutes ago (in seconds, since January 1, 1970 format).
String toLowerCase()	Converts all of the characters in this to lower case using the rules of the default locale. **Example:** Title: $group0.toLowerCase()

Expressions and Functions	Description
String toUpperCase()	Converts all of the characters in this to upper case using the rules of the default locale. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#toUpperCase(). **Example:** Title: $group0.toUpperCase()
String trim()	Returns a copy of the string, with leading and trailing whitespace omitted. For more details, see http://download.oracle.com/javase/7/docs/api/java/lang/String.html#trim(). **Example:** Category: $group3.trim()

Options Page (Events Only)

UI Element	Description
Log Local Events	Defines which events, if any, are logged on the node from which they originated. These events are logged on the local node in the log file `<data_dir>\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 (that is, they do not cause an event to be sent to the Event Browser).
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 the Event Browser when an 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 the Event Browser 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:** If several XML file policies forward unmatched events to BSM, you could receive multiple events about a single input event. **Note:** Nodes create an event about an unmatched event only if the input event is unmatched in all SNMP trap policies on the node. Nodes send only one event for each unmatched input event.
are sent to the Event Browser	Sends unmatched events to the Event Browser.
are sent to the Closed Events Browser	Sends unmatched events to the Closed Events Browser.
are ignored	Ignores unmatched events.
Pattern Matching Options	Defines case sensitivity and field separators 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.

UI Element	Description
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 (Event Rules 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 327.

Policy Constants Tab (Metrics Only)

UI Element	Description
QUALITY_ ERROR	Maps the metrics quality to the critical HI state (that is, the state that corresponds to the critical severity; if critical does not exist on the CI, major, minor, or warning).
QUALITY_ WARNING	Maps the metrics quality to the warning HI state (that is, the state that corresponds to the warning severity; if warning does not exist on the HI, the closest state to it except for states representing normal severity).
QUALITY_ GOOD	Maps the metrics quality to the normal HI state (that is, the state that corresponds to the normal severity).

Policy Variables Tab (Events Only)

Variable	Description
`<$#>`	Returns the number of variables in an enterprise-specific SNMP event (generic event 6 Enterprise specific ID). Sample output: `2`
`<$*>`	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`
`<$@>`	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>`	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>`	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`
`<$\>+1>`	Returns all attributes greater than *n* as *name:value* string. Sample output: `.1.2: asparagus.example.com`
`<$+2>`	Returns the nth variable binding as *name:value*. Sample output: `.1.2: artichoke.example.com`
`<$\>-n >`	Returns all attributes greater than *n* as *[seq] name (type): value* strings. Sample output: `[2] .1.2 (OctetString): cauliflower.example.com`
`<$-2>`	Returns the nth variable binding as *[seq] name-type:value*. Sample output: `[2] .1.2 (OctetString): brusselsprouts.example.com`
`<$A>`	Returns the node that produced the event. Sample output: `eggplant.example.com`
`<$C>`	Returns the community of the event. Sample output: `public`
`<$E>`	Returns the enterprise ID of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1`
`<$e>`	Returns the enterprise object ID. Sample output: `.1.3.6.1.4.1.11.2.17.1`
`<$DATABASE_SERVER>`	Returns the name of the database server as specified in the **Server name** field in the **Source** page. Sample output: `dbserver.example.com`
`<$DATABASE_TABLES>`	Returns the names of the queried tables as specified in the **FROM** field in the **Source** page. Sample output: `ALL_EVENTS`

Variable	Description
<$F>	Returns the textual name of the remote postmaster daemon's computer if the event was forwarded. Sample output: `cress.example.com`
<$G>	Returns the generic event ID. Sample output: `6`
<$LOGFILE>	Returns the name of the log file that contains the input event. Sample output: `program_log.txt`
<$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`
<$MSG_ID>	Returns the unique identity number of the event, as generated by the HP Operations Agent. Note that identity numbers are not generated for suppressed messages. Sample output: `6e998f80-a06b-71d0-012e-0f887a7c0000`
<$MSG_NODE>	Returns the IP address of the node on which the original event took place. Sample output: `192.168.1.123`
<$MSG_NODE_NAME>	Returns the name of the node on which the original event took place. This is the hostname that the agent resolves for the node. This variable is not fixed, however, and can be changed by a policy on a per-event basis. For example, if the policy is receiving SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated.
<$MSG_SERVICE>	Returns the service name associated with the event.
<$MSG_OBJECT>	Delivers the name of the object associated with the event.
<$MSG_OBJECT>	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`

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.
`<$N>`	Returns the event name (textual alias) of the event format specification used to format the event, as defined in the Event Configurator. Sample output: `OV_Node_Down`
`<$NAME>`	Returns the name of the policy that sent the event. Sample output: `cpu_util`
`<$O>`	Returns the name (object identifier) of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1.0.58916865`
`<$o>`	Returns the numeric object identifier of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1.0.58916865`
`<$OPTION(N)>`	Returns the value of an optional variable that is set by `opcmsg` (for example, `<$OPTION(A)>`, `< $OPTION(B)>`, and so on.).
`<$PROG>`	Returns the name of the program executed by the scheduled task policy Sample output: `check_for_upgrade.bat`
`<$R>`	Returns the true source of the event. This value is inferred through the transport mechanism which delivered the event. Sample output: `carrot.example.com`
`<$r>`	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`
`<$S>`	Returns the specific event ID. Sample output: `5891686`
`<$s>`	Returns the event's severity. Sample output: `Normal`
`<$T>`	Returns the event time stamp. Sample output: `0`
`<$USER>`	Returns the name of the user under which the scheduled task was executed. Sample output: `administrator`
`<$V>`	Returns the event type, based on the transport from which the event was received. Currently supported types are SNMPv1, SNMPv2, CMIP, GENERIC, and SNMPv2INFORM. Sample output: `SNMPv1`
`<$X>`	Returns the time the event was received using the local time representation. Sample output: `17:24:58`
`<$x>`	Returns the date the event was received using the local date representation. Sample output: `03/27/10`

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.
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.
Sample Type	*Metrics integration policies only.* The type of metrics sample collected: ss_t (System). Other sample types are currently not supported.

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 BSM. 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 BSM. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to BSM. *Metrics policies:* **Create New Rule:** Provides the following options: • **Process on matched rule.** If matched, BSM Connector sends metrics to BSM. 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 BSM. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to BSM.
⧉	**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.
⇧	**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.

UI Element	Description
▽	**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 BSM. 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 BSM. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send an event to BSM. The three rule types of metrics policies are: • **Process on matched rule.** If matched, BSM Connector sends metrics to BSM. 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 BSM. • **Suppress on unmatched rule.** If not matched, BSM Connector stops processing and does not send metrics to BSM. 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 ✖.
XML	**Toggle Short/Full Path Notation.** Shows or hides the full path to the table column. The full path starts with `/BSMConnectorEvents/BSMConnectorEvent`. The Table Column section by default shows the short path to the table column.

UI Element	Description
🔍	**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.
Table Column	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 119.
Values for <...>	Displays the values of the column selected in the Table Column section.

Sample Data Tab - Log File Policies

UI Element	Description
<Search Properties> 🔍 ❌	Entered search string is used to find a pattern matching group. The list changes as you type; only matching items appear. To clear the search results, click ❌.
XML	**Toggle Short/Full Path Notation.** Shows or hides the full path to the pattern matching group. The full path starts with `/BSMConnectorEvents/BSMConnectorEvent`. The Pattern Matching Group section by default shows the short path to the pattern matching group.
🔍	**Find Matching Events.** To find values that belong to more than one pattern matching group, select the value and click 🔍. The Log File Sample Data window opens and shows all pattern matching groups that have the selected value.
⧉	**Toggle Deduplication.** Shows or hides duplicate values.

UI Element	Description
Pattern Matching Group	Shows all pattern matching groups that match a string in the log file. **Note:** • The default pattern matching group `host_name` contains the fully qualified domain name of the BSM Connector server. • 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 log file pattern specified in the source page. See also "Configuring the Data Source in Log File Policies" on page 151.
Values for <...>	Displays the values of the pattern matching group selected in the Pattern Matching Group section.

Sample Data Tab - Web Service Listener Policies

UI Element	Description
<Search Properties> 🔍 ✖	Entered search string is used to find a key. The list changes as you type; only matching items appear. To clear the search results, click ✖.
XML	**Toggle Short/Full Path Notation.** Shows or hides the full path to the key. The full path starts with `/BSMConnectorEvents/BSMConnectorEvent`. The Key section by default shows the short path to the key.
🔍	**Find Matching Events.** To find values that belong to more than one key, select the value and click 🔍. The Web Service Sample Data window opens and shows all keys that have the selected value.
⧉	**Toggle Deduplication.** Shows or hides duplicate values.

UI Element	Description
Key	Shows all keys that have been received by the Web service listener for this policy. **Note:** - The sample data includes a key-value pair that represents the default routing domain configured for the BSM Connector server in BSM under **Admin > Integrations > BSM Connector Integrations**. The default routing domain name is `DefaultDomain`. You may insert the default routing domain name in CI-related fields of the policy to help BSM distinguish different routing domains. - The Sample Data tab is empty if no sample data has been loaded into the policy or if no SOAP messages have been received for the policy. For information about loading sample data into a Web service listener policy, see "Event integration only: How to load example data into the policy" on page 282.
Values for <...>	Displays the values of the key selected in the Keys 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 ✖.
XML	**Toggle Short/Full Path Notation.** Shows or hides the full path to the XML property or value. The full path begins with the XML event 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 event tag. (You can identify attributes based on the preceding at sign (@).) **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 event tags.

UI Element	Description
Values for <...>	Displays the values of the XML property selected in the XML Properties section.

Schedule Page

Schedule Page - Legacy Discovery Policies

UI Element	Description
✖	Clears the selection.
☑	Selects all days, hours, or minutes.
Days of Week	Days of the week from Sunday to Saturday.
Hours of Day	1 to 12 AM and 1 to 12 PM.
Minute of Hour	0 to 59 minutes.
Schedule Summary	Summary of the specified schedule

Schedule Page - Scheduled Task Policies

UI Element	Description
✖	
☑	**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. - **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: <> h <> m <> s	Interval in hours, minutes, and seconds.
Advanced (daily execution)	
Minute of Hour	0 to 59 minutes.
Hours of Day	1 to 12 AM and 1 to 12 PM.

UI Element	Description
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 - Basic and Query Settings

UI Element	Description
Basic Settings	
Database Connection URL	URL to a database connection (sometimes referred to as an Authentication string). One way to create a database connection is to use ODBC to create a named connection to a database. For example, first use the ODBC control panel to create a Data Source Name (DSN) called `test` under the system DSN tab. Then, enter `jdbc:odbc:test` as the connection URL. Alternatively, use the supplied Microsoft SQL or Oracle driver to connect to the database.
Database Driver	Driver used to connect to the database. Use the fully qualified class name of the JDBC driver you are using.
Database User Name	User name used to log on to the database.
Database Password	Password used to log on to the database.
Server Name	Name of the database server. The value entered here is returned by the policy variable `<$DATABASE_SERVER>`, which can be used in the Source CI attribute to identify the source of the database data.

UI Element	Description
Frequency	How often the policy queries the database (in seconds, minutes, hours, days, or weeks). **Default value:** 10 minutes **Minimum value:** 3 seconds
Timeout	Amount of time, in seconds, that BSM Connector waits before the policy query times out. **Default value:** 60 seconds **Note:** The timeout period must be shorter than the frequency.
Query Settings	
SELECT	SELECT clause to be used in the SQL query. Enter ***** for all fields or a comma separated list of column names to be retrieved from the database. When specifying the SELECT clause, the column used as the enumerating field must appear in the clause.
FROM	FROM clause to be used in the SQL query. Enter a table name or a comma separated list of tables from which the selected columns should be extracted.
WHERE	WHERE clause to be used in the SQL query. This is an optional field which enables you to define the select criteria. Leaving it empty results in retrieving all the rows from the table defined in the FROM option.
Enumerating field	Name for a database field that can be used to order the data that is returned from the database query. For details on how data from the enumerating field is processed, see "UI Descriptions" on page 389 above. **Note:** The column used as enumerating field must be included in the SELECT clause.

<table>
<tr><th>UI Element</th><th>Description</th></tr>
<tr><td><b>Enumerating field type</b></td><td>The type of field used to order the result set. This can be a DATE field, an INTEGER field, a DOUBLE floating point numeral field, or a LONG field.
The following table maps SQL types to the required enumerating field type.
<table>
<tr><th>SQL Type</th><th>Enumerating Field Type</th></tr>
<tr><td>SMALLINT</td><td>INTEGER</td></tr>
<tr><td>INTEGER</td><td>INTEGER / LONG</td></tr>
<tr><td>BIGINT</td><td>LONG</td></tr>
<tr><td>NUMERIC</td><td>LONG</td></tr>
<tr><td>DOUBLE</td><td>DOUBLE</td></tr>
<tr><td>DECIMAL</td><td>DOUBLE</td></tr>
<tr><td>FLOAT</td><td>DOUBLE</td></tr>
<tr><td>TIMESTAMP</td><td>TIMESTAMP</td></tr>
<tr><td>DATE</td><td>TIMESTAMP</td></tr>
</table></td></tr>
<tr><td><b>Initial Enumerating Value</b></td><td>Initial value to be used as a condition for the initial run of this policy instance. For example, if you specify the <b>Enumerating field type</b> as a field type <code>DATE</code> and you enter a value of <code>2000-01-31 12:00:00</code> in the <b>Initial Enumerating Value</b> field, only records that were added to the database after the specified date are forwarded.</td></tr>
<tr><td><b>Max rows to fetch</b></td><td>Maximum number of rows the policy retrieves from the database for each cycle.
<b>Default value</b>: 5000 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.</td></tr>
</table>

Source Page - Database Policies - Sample Data

UI Element	Description

📄	Retrieves the specified table columns from the database. **Note:** BSM Connector can only load a maximum of 50 MB of sample data.
🔍	Opens the Database Sample Data dialog box. This dialog box displays the table columns and values returned by the database query.
<>	Displays the message `Example data loaded from database` when the database query was successful.

Source Page - Log File Policies

UI Element	Description
Server	Server on which this policy reads the log file. The drop-down list displays on those remote servers that have been configured in BSM Connector. **Default value:** BSM Connector Server (the server on which BSM Connector is installed)
Remote Servers	Opens the Remote Servers dialog box. For details, see "Configuring Remote Servers " on page 189.

Log File Name	Path and name of the log file from which you want to extract data.
	• **Remote UNIX.** For reading log files on remote UNIX machines, the path must be relative to the home directory of UNIX user account being used to log on to the remote machine. Click **Remote Servers** and select the UNIX remote server for information about which UNIX user account is being used.
	• **Remote Windows through NetBIOS.** You can also access log files by including the UNC path to the remote log file. For example, `\\remoteserver\sharedfolder\filename.log`.
	This requires that the user account under which BSM Connector is running has permission to access the remote directory using the UNC path.
	If a direct connection using the operating system is unsuccessful, BSM Connector tries to match the `\\remoteserver` with servers currently defined as remote Windows connection profiles (displayed in the remote server list).
	If an exact match is found for `\\remoteserver` in the remote Windows connection profiles, BSM Connector tries to use this connection profile to access the remote log file. If no matching server name is found, the policy reports that the remote log file cannot be found.
	It is not necessary to select a remote Windows server if you are using NetBIOS to connect to remote Windows servers.
	• **Remote Windows through SSH.** Select a remote server from the drop-down list. The path of the log file depends on the type of SSH server installed on the remote Windows server:
	▪ SSH servers that provide a UNIX-like interface (for example, Cygwin OpenSSH):
	`/cygdrive/<drive_letter>/<directory>/filename.log`
	▪ SSH servers that provide a Windows command prompt (for example, OpenSSH for Windows):
	`<drive_letter>:\<folder>\filename.log`
	Optionally, you can use a regular expression to insert date and time variables. For example, you can use a syntax of `s/ex$shortYear$$0month$$0day$.log/` to match date-coded IIS log file names. For details, see "Special Substitution for File Path" on page 225.

Log File Encoding	Log file encoding that is used if you are reading a log file whose encoding is different than the BSM Connector machine's default encoding. **Default value:** Cp1252
Frequency	How often the policy reads the log file (in seconds, minutes, hours, days, or weeks). **Default value:** 10 minutes **Minimum value:** 3 seconds

Timeout	Maximum number of seconds that the policy attempts to access the log file. **Default value:** 60 seconds
Log File Pattern	
Log File Pattern	Text to look for in the log entries. You can also use a regular expression in this entry to match text patterns. Surround values you wish to extract with parenthesis, for example `(.*) (.*) (.*)(.*)`. Escape special characters with a backslash (`\`). Examples of special characters are the plus sign (`+`), the asterisk (`*`), the question mark (`?`), the at sign (`@`), the exclamation point (`!`), the caret (`^`), brackets (`[]`), and braces (`{}`). For more information about regular expressions, see "Regular Expressions" on page 217.
Sample Data	
📄	Loads the log file and applies the log file pattern. **Note:** BSM Connector can only load a maximum of 50 MB of sample data.
<>	Displays the message `Example data loaded from file '<file>'` if the log file could be loaded successfully.

Source Page - Web Service Listener Policies

UI Element	Description
System ID	Text system ID for the Web service listener policy instance. Each received message from the third-party system holds a system ID. Each Web service listener policy receives messages only with a system ID that matches the system ID defined in the policy. The system ID must be unique for all policies. Enter the system ID that represents the messages that you want this policy to receive.
Sample Data (for event integration only)	
📄	Retrieves the specified key-value pairs from the SOAP messages. **Note:** BSM Connector can only load a maximum of 50 MB of sample data.
🔍	Opens the Web Service Sample Data dialog box. This dialog box displays the keys and values of the received SOAP messages.
<>	Displays the message `Example data loaded` when the policy receives messages through the Web service listener that match the system ID defined in the policy.

Source Page - XML File Policies

UI Element	Description
Log File Path / Name	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. **Tip:** • You can use Windows environment variables (for example winnt or clusterlog) to make your policies more flexible. The proper syntax for these variables is <$*variablename*>, for example <$winnt>. • You can also call a 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. **Note:** BSM Connector cannot process log files that are larger than 2 GB.
Polling Interval	Determines how often the policy reads the XML file. This period of time is the polling interval. The polling interval should be as large as possible, although this depends on the amount of new data written to the file and the read mode that you choose. Set the interval to no less than 30 seconds; usually 5 minutes is appropriate. Note, however, that a policy begins to evaluate data *after* the first polling interval passes. A shorter polling interval is better when you are testing a policy. Default value: 5 minutes
Logfile Character Set	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 BSM. 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	BSM Connector sends an event if the specified XML file does not exist. Default value: not selected
Close after reading	The policy keeps the XML file open (and retains its file handle) after reading it. Do not use a polling interval of less than one minute when this option is selected. If you do not select this option and the name of the XML file changes, the policy continues to read the original XML file instead of processing any new XML file with the specified name. Consider the following example: a policy reads the log file `syslog.log`. Mondays at 23:59, the file is renamed to `syslog.monday`, and a new version of `syslog.log` is created for the Tuesday log. Without Close after reading being selected, the policy continues to read `syslog.monday` because the file handle refers to the original, renamed file. Default value: not selected

<table>
<tr>
<td><b>Read Mode</b></td>
<td>
<p>The read mode of an XML file policy indicates whether the policy processes the entire file or only new entries.</p>
<table>
<tr>
<td><p><b>Read from last position.</b> 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.</p><p>Choose this option if you are concerned only with entries that occur when the policy is enabled.</p></td>
<td><p><b>Advantage:</b> No chance of reading the same entry twice. (Unless the file decreases in size because some entries were deleted.)</p><p><b>Disadvantage:</b> Entries written to file while the policy is disabled or the agent is not running are not processed by the policy.</p></td>
</tr>
<tr>
<td><p><b>Read from beginning (first time).</b> 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.</p><p>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.</p></td>
<td><p><b>Advantage:</b> Every existing and future entry in the file will be processed by the policy.</p><p><b>Disadvantage:</b> Duplicate entries can occur if an activated policy is deactivated and reactivated, or if the agent stops and restarts.</p></td>
</tr>
<tr>
<td><p><b>Read from beginning (always).</b> 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.</p><p>Choose this option if you the policy reads a file that is overwritten, rather than appended.</p></td>
<td><p><b>Advantage:</b> Ensures that files that are overwritten are correctly processed.</p><p><b>Disadvantage:</b> Only valid for files that are overwritten, rather than appended.</p></td>
</tr>
</table>
<p><b>Note:</b> Every policy reads the same XML files independently from any other policies. This means, for example, that if "Policy 1" with read mode <b>Read from beginning (first time)</b> is activated and "Policy 2" with the same read mode</p>
</td>
</tr>
</table>

	already exists, "Policy 1" still reads the entire file after it has been activated. Default value: Read from last position
Sample Data	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 BSM 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	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.

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.

Task Page (Scheduled Task Policies)

UI Element	Description
📄	**Load.** 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 254 .) **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 254 .) **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 (`).

Topology Page

UI Element	Description
Frequency	*Custom topology discovery only:* How often the policy runs the topology script. Use the drop-down list to specify increments of seconds, minutes, hours, days or weeks. **Default value:** 10 minutes **Minimum value:** 15 seconds
<script>	*Custom topology discovery only:* The contents of the script are visible in this box. You can edit the script contents in this field using the script editor provided by BSM Connector, or you can copy it into your preferred text editor, edit it, and then copy it back into this box. **Note:** The topology script is sensitive to spaces and tabs.

UI Element	Description
Script Template	*Database, log file, and Web service listener topology discovery only:* Script to create the topology in BSM for the data retrieved from the connected third-party system. The script is based on the Jython scripting language (Python enabled by Java). • *Metrics and event integration only:* **No Topology**. No script is selected and no topology is sent (although data is still sent). • *Metrics integration only:* **Computer - Monitor.** BSM Connector reports this data to the Computer CI, a descendant of the Node CI. • **Computer**. Creates a topology with a Computer CI. • **Computer - Running Software.** Creates a topology with a Computer CI and a Running Software CI connected to it with a `Composition` relationship. • **Custom**. You create your own topology if you want the retrieved data to be forwarded to specific CIs and not the standard Computer or Running Software CIs. **Note:** You should only select **Custom** if you are familiar with the Jython language, since you must create the topology script in Jython yourself. Depending on the data type you want to collect, we recommend that you select and edit one of the predefined scripts.
Load Template	*Database, log file, and Web service listener topology discovery only:* Loads the required script for the topology you selected in the **Script template** option. If you select Computer - Monitor, Custom, or No Topology, there is no script to load and the Load template button is not available. **Caution:** When you load a script, the currently loaded script is overwritten and all of you changes are lost.