HP Business Service Management

For the Windows, Linux operating systems

Software Version: 9.21

BSM Extensibility Guide

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notice

Trademark Notices

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

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

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

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

iPod is a trademark of Apple Computer, Inc.

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

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

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

UNIX® is a registered trademark of The Open Group.

Acknowledgements

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

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

This product includes software developed by the MX4J project (mx4j.sourceforge.net).

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.

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.

This document was last updated: Friday, November 16, 2012

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

Extensibility Guide Overview

This guide describes how to customize Business Service Management applications, extend their functionality, and use APIs to perform operations. The guide is intended for administrators and integrators who need to establish advanced configurations and extensions. If you are an administrator and need to set up BSM, first refer to the BSM Application Administration Guide.

This guide provides instructions for working with the following applications:

- **Service Health**. For details, see "Service Health Rules API" on page 22 and "Service Health External APIs" on page 52.

- **Service Level Management**. For details, see "SLM External API " on page 64, "SLM Rules API" on page 78, and "SLM Web Services API" on page 126.

- **End User Management**. For details, see "EUM Admin Open API" on page 133.

- **Operations Management**. For details, see "Extending Operations Management" on page 135.

- **Reports**. For details, see "Generic Reporting Engine API" on page 419.

Part 1

Service Health

Chapter 1

Service Health Rules API

> **Note:** In BSM versions 9.00 and later, the rules that calculate indicator statuses and values based on samples ("API Sample Rule" on page 27 and "API Duration-Based Sample Rule" on page 29) are used to calculate metric-based health indicators (HIs).
>
> Throughout the Rules API documentation, you will see references to various methods used to calculate KPIs. **In BSM versions 9.00 and later, when calculating sample-based values, these methods are used to calculate metric-based HIs.**

This chapter describes how to use the Rules API to create new business rules. Business rules are used to calculate Key Performance Indicators (KPIs). A KPI must have an associated business rule that defines how the KPI is calculated. The default Service Health rules appear in the section List of Calculation Rules in Service Health in the BSM Application Administration Guide.

It is recommended to create rules with the Rules API. The Rules API enables you to create rules using the Groovy scripting language with Groovy runtime environment 1.7.3 and earlier. Users of the Rules API should be familiar with Groovy and Java, and with BSM administration and applications.

The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
**\\< Gateway Server root directory>
\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html**.

Service Health API Rules

These are the types of Service Health API rules:

- **Group and Sibling Rule**. This rule calculates KPIs based on data received from other KPIs, rather than from original sample data. For details, see "API Group and Sibling Rule" on page 24.

- **Sample Rule**. This rule calculates KPIs based on original data taken from sample fields; the number of samples included in the calculation is limited by a maximum number of samples rule parameter. For details, see "API Sample Rule" on page 27.

- **Duration-Based Sample Rule**. This rule calculates KPIs based on original data taken from sample fields; a duration parameter defines which samples are included in the calculation. For details, see "API Duration-Based Sample Rule" on page 29.

Creating API Rules

Rules can be created using the Rules API in these ways:

- Using the CI Indicators tab to create a rule for a specific KPI.

- Using a text file to create a new rule for multiple KPIs.

- Using a clone of an API rule in the Rule Repository to create a new rule.

These ways are described in "Creating Rules with the Rules API" on page 31.

Tooltips and Log Files

To display KPI information in tooltips when working with the Rules API, see "How to Work with Tooltip Entries" on page 37.

You can write to log files from the Rules API code, as described in "How to Write to Log Files From the Rules API Code" on page 38.

API Group and Sibling Rule

An API Group and Sibling Rule calculates KPIs based on data received from other indicators, rather than from original sample data. The received data can come from the KPIs of child CIs, or from other KPIs or HIs associated with the same CI.

> **Note:** If you are creating a sibling rule, make sure that the KPI is calculated after its sibling KPIs, as defined by the KPI's Calculation Order field. For details, see KPIs Repository page in the BSM Application Administration Guide.

Group and Sibling Rule Methods and Fields

The Group and Sibling rule implements the Rules API Interface **GroupAndSiblingCalculator**, using the following guidelines:

- In this interface, the only method is **calculateKPI**. The method signature is:

```
public void calculateKPI(CI ci, KPI kpi)
```

- The **calculateKPI** method includes the parameters **ci** and **kpi**, which represent the current CI, and the KPI whose value the API rule calculates.

  - The **ci** parameter type is **CI**, and is used as an accessor to KPIs of child CIs or sibling KPIs, or HIs on the CI.

  - The **kpi** parameter type is **KPI**, and is used to set calculation results.

In the following illustration, the Calculated KPI is calculated based on the sibling or child KPIs, and it is represented by the **kpi** parameter.

The CI to which the Calculated KPI is assigned, is represented by the **ci** parameter, and it is an accessor to the other KPIs or HIs.
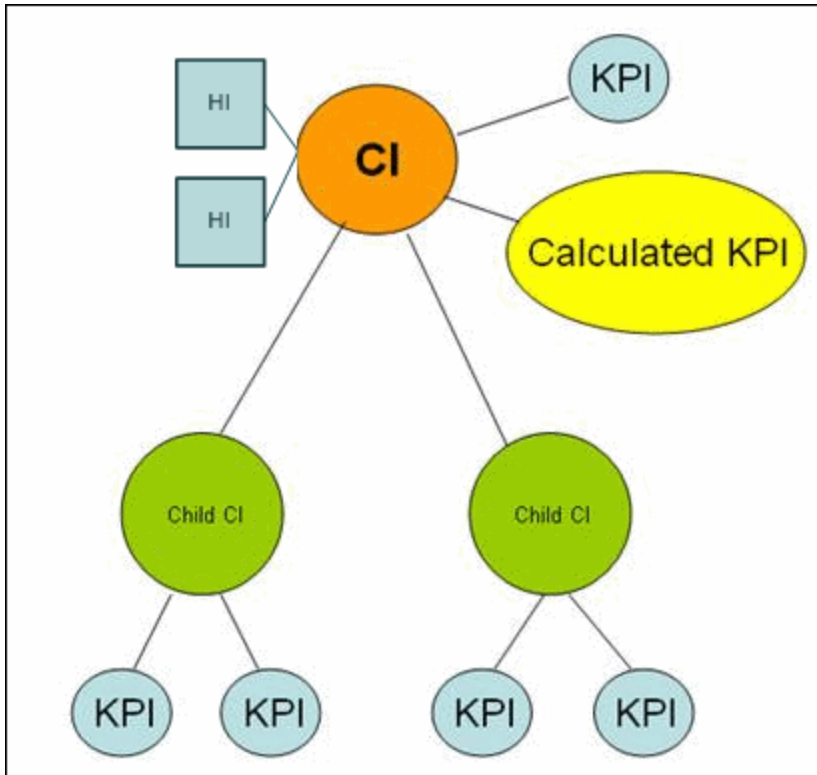
The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:

\\<Gateway Server root directory>
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html.

For detailed examples of Group and Sibling rules, see

API rules can be defined within the Service Health CI Indicators tab or Rule Repository, or using a text file template, as described in

Defining a Group and Sibling Rule in the CI Indicators tab or Rule Repository

To define a Group and Sibling rule using the CI Indicators tab or within the Rule Repository, enter the **calculateKPI** method implementation in the **KPI Calculation Script** area.

The parameters **ci** and **kpi** of the **calculateKPI** method are available for use in this script.

For detailed instructions, see or

Accessing a Specific Child KPI in the CI Indicators Tab

When creating a Group rule for a specific KPI in the CI Indicators tab, to access a specific child KPI, the API includes a mechanism to simplify the code. When defining your KPI Calculation Script, you can enter the format **"<CI name>"."<KPI name>"**.

For an example of this, see in

Defining a Group and Sibling Rule Using a Text File

To define a Group and Sibling rule using a text file, use the
DashboardGroupAndSiblingTemplate.groovy template as described in "How to Create a Text File-Based API Rule" on page 33.

Within the text file, enter the **calculateKPI** method body.

API Sample Rule

A Sample rule calculates KPIs based on original data taken from sample fields; the number of samples included in the calculation is limited by a maximum number of samples parameter.

Sample Rule Methods and Fields

The Sample rule implements the Rules API Interface **LeafCalculator**, using the following guidelines:

- In this interface, the only method is **calculateKPI.** The method signature is:

```
public void calculateKPI(CI ci, KPI kpi, List<Sample> samples)
```

- The **calculateKPI** method includes the parameters **ci**, **kpi, and samples.** These represent the current CI, the KPI whose value the rule calculates, and the samples to be used in the rule calculation based on the **Maximum number of samples** parameter. (If this parameter value is 1, list one sample in this field.)

  - The **kpi** parameter type is **KPI**, and is used to set calculation results.

  - The **samples** parameter is a **List** of **Sample** objects, which hold sample field values.

- The rule must also set the **sampleFields** field to define which sample fields are held by the **Sample** object. These values are the values used by the rule.

For detailed examples of Sample rules, see "Examples - API Sample Rule" on page 41.

API rules can be defined within the Service Health CI Indicators tab or the Rule Repository, or using a text file template, as described in "Creating Rules with the Rules API" on page 31.

The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
**\\<Gateway Server root directory>\
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html**.

Defining a Sample Rule in the CI Indicators tab or Rule Repository

To define a Sample rule using the CI Indicators tab or within the Rule Repository, fill in the fields as follows:

- **Sample Fields**. List the sample fields which are held by the **Sample** object; separate between the sample names with a comma (for example: `"u_iStatus", "dResponseTime"`).

- **KPI Calculation Script**. Enter the **calculateKPI** method implementation; do not enter the method signature. The parameters **ci**, **kpi, and samples** of the **calculateKPI** method are available for use in this script.

- **Maximum number of samples**. By default only the most recent sample is included (default=1). You can use this field to change this setting.

For detailed instructions, see "How to Define an API Rule in the CI Indicators Tab" on page 32 or "How to Define an API Rule in the Rule Repository" on page 36.

Defining a Sample Rule Using a Text File

To define a Sample rule using a text file template, use the
DashboardSampleRuleTemplate.groovy template file as described in "How to Create a Text
File-Based API Rule" on page 33.

Within the text file, enter the **calculateKPI** method body, and define the **sampleFields** field.

API Duration-Based Sample Rule

A Duration-Based Sample rule calculates KPIs based on original data taken from sample fields; the duration rule parameter defines which samples are included in the calculation. For example, if duration is defined as fifteen minutes, all samples collected during the last fifteen minutes are included in the calculation.

Duration-Based Sample Rule Methods and Fields

The Duration-Based Sample rule implements the Rules API Interface **LeafCalculator**, using the following guidelines:

- In this interface, the only method is **calculateKPI**. The method signature is:

```
public void calculateKPI(CI ci, KPI kpi, List<Sample> samples)
```

- The **calculateKPI** method includes the parameters **ci**, **kpi, and samples.** These represent the current CI, the KPI whose value the rule calculates, and the list of samples to be used in the rule calculation.

  - The **kpi** parameter type is **KPI**, and is used to set calculation results.

  - The **samples** parameter is a **List** of **Sample** objects, which hold sample field values.

- The rule must also set the **sampleFields** field to define which sample fields are held by the **Sample** object. These values are the values used by the rule.

For detailed examples of this rule, see "Examples - API Sample Rule" on page 41.

API rules can be defined using the Service Health CI Indicators tab, using a text file, or within the Rule Repository, as described in "Creating Rules with the Rules API" on page 31.

The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
\\<Gateway Server root directory>
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html.

Defining a Duration-Based Sample Rule in the CI Indicators tab or Rule Repository

To define a Duration-Based Sample rule using the CI Indicators tab or within the Rule Repository, fill in the fields as follows:

- **Sample Fields**. List the sample fields which are held by the **Sample** object; separate between the sample names with a comma (for example: `"u_iStatus", "dResponseTime"`).

- **KPI Calculation Script**. Enter the method implementation; do not enter the method signature. The parameters **ci**, **kpi, and samples** of the **calculateKPI** method are available for use in this script.

- **No data timeout** and **duration**. (Optional) You can define the timeout period and duration parameters, as described in List of Rule Parameters.

For detailed instructions, see "How to Define an API Rule in the CI Indicators Tab" on page 32 or "How to Define an API Rule in the Rule Repository" on page 36.

Defining a Duration-Based Sample Rule Using a Text File

To define a Duration-Based Sample rule using a text file template, use the **DashboardDurationBasedSampleRuleTemplate.groovy** template file as described in "How to Create a Text File-Based API Rule" on page 33.

Within the text file, enter the **calculateKPI** method body, and define the **sampleFields** field.

Creating Rules with the Rules API

There are a number of ways to create rules using the Rules API, as described in the following section.

Define a rule for a specific KPI using the CI Indicators tab

Each Service Health KPI has these applicable API rules: API Group and Sibling Rule, API Sample Rule, or API Duration-Based Sample Rule. From the CI Indicators tab, you can assign one of the API rules to a KPI, and enter a calculation script (and other rule details) to define rule logic for that KPI.

You can then edit the rule details in the CI Indicators tab at any time to change the rule logic for the KPI.

For details, see "How to Define an API Rule in the CI Indicators Tab" on the next page.

Create a rule using a text file

For each of the API rules (Group and Sibling Rule, Sample Rule, or Duration-Based Sample Rule) there is a corresponding template file, located in the **<Data Processing server root directory>\BLE\rules\groovy\templates** directory. You can use one of the template files to create a text file defining a new rule. You then add this rule to the Rule Repository, and it can be applied like any out-of-the-box rule.

The API code cannot be seen or changed within Service Health, but only within the text file. If you make changes to the code within the text file, these changes are applied to all instances where the rule has been assigned, after you reload Service Health rules.

For details, see "How to Create a Text File-Based API Rule" on page 33.

Define a rule within the Rule Repository

The Rule Repository contains three API rules: API Group and Sibling Rule, API Sample Rule, or API Duration-Based Sample Rule. You can use the Rule Repository to clone an API rule and enter a calculation script (and other rule details) to define the rule logic.

After the rule is applied to a KPI, you can edit rule details within the CI Indicators tab at any time to change the rule logic for a specific KPI.

For details, see "How to Define an API Rule in the Rule Repository" on page 36.

How to Define an API Rule in the CI Indicators Tab

Each KPI has three applicable API rules. Within the CI Indicators tab, assign one of the API rules to a KPI, and enter the calculation script (and other rule details) to define the rule logic for that KPI.

1. ### Assign an API rule to a KPI

   To assign an API rule for a specific KPI assigned to a CI, select **Admin > Service Health > CI Indicators**. Select **New KPI** to assign a new KPI to the CI, or **Edit KPI** to modify an existing KPI. For details on this process, see How to Assign KPIs and HIs to CIs in the BSM Application Administration Guide.

   From the list of applicable business rules, select one of the API rules: API Group and Sibling Rule, API Sample Rule, or API Duration-Based Sample Rule. For a description of the rule types see "Service Health Rules API" on page 22.

2. ### Define the KPI's rule logic

   Depending on the type of rule you are creating, define the rule methods and fields as described in:

   - "API Group and Sibling Rule" on page 24

   - "API Sample Rule" on page 27

   - "API Duration-Based Sample Rule" on page 29

How to Create a Text File-Based API Rule

There are three rule template files corresponding to the three API rules; each template implements the rule's interface.

Create a text file defining a new rule using one of the templates, and then add the new rule to the Business Rule Repository. The rule can then be applied like any out-of-the-box rule.

The API code cannot be seen or changed within Service Health, but only within the text file. If you make changes to the code within the text file, these changes are applied to all instances where the rule has been assigned, after you reload Service Health rules.

1. ## Create a text file for a rule

   Based on the type of rule you want to create, copy and rename one of the template files located in the **<Data Processing server root directory>\BLE\rules\groovy\templates** directory.

   Within your copy of the template, define the rule methods and fields as described in:

   - "API Group and Sibling Rule" on page 24

   - "API Sample Rule" on page 27

   - "API Duration-Based Sample Rule" on page 29

   Save the file to the **<Data Processing server root directory>\BLE\rules\groovy\rules** directory.

   You must now add a rule in the Rule Repository that uses the rule logic in the text file.

2. ## Add a rule in the rule repository

   a. Select **Admin> Service Health > Repositories > Business Rules > New Rule**. For details on adding rules, see How to Customize a Business Rule Template in the Repository in the BSM Application Administration Guide.

   b. In the **Name** field, type the name of the rule you want to create (mandatory).

   c. In the **Class Name** field, type **groovy:** <file name>. Note that the file name must be identical (case sensitive) to the file name in the **<Data Processing server root directory>\BLE\rules\groovy\rules** directory.

   d. Create Rule parameters depending on your API rule type, as follows:

      ○ In the **Rule parameters** area, click **New**.

      ○ For API Sample rules:

      In the **Name** field type **Maximum number of samples**. In the **Type** field, select **Integer**. In the **Default Value** field, type **1**.

      Click **OK** to Save.

      ○ For API Duration-Based Sample rules:

      In the **Name** field type **duration**. In the **Type** field, select **Long**. In the **Default Value** field, type **990**.

      Click **OK** to Save.

Repeat these steps to add the **No Data Timeout** rule parameter (Type: Long; Default Value = 990).

e. Create Threshold parameters: critical, major, minor, warning, informational, and operator. (Skip this step if you are defining a Group and Sibling rule that does not have Thresholds, where status is calculated by the rule code.)

○ In the **Threshold parameters** area, click **New**.

○ In the **Name** field, type **critical**. In the **Type** field, select **Float**.

When defining the **operator** parameter, select **String** in the **Type** field.

○ Click **OK** to save.

Repeat the above steps for each of the other Threshold parameters (major, minor, warning, informational, and operator).

The following image shows a Sample rule after the rule parameter has been added:

The following image shows a Duration-Based Sample rule after the rule parameters have

been added:

3. **Add the rule to the KPI's applicable rules list**

Add the new rule to the list of applicable rules already attached to the relevant KPI. For details, see the Main Settings Area > Applicable Rules parameter in New KPI/Edit KPI Dialog Box in the BSM Application Administration Guide.

4. **Add tooltip parameters to the new tooltip**

When a rule is created using this procedure, a corresponding tooltip is with no tooltip parameters. For instructions on adding tooltip parameters to the new tooltip, see "How to Work with Tooltip Entries" on page 37.

5. **Reload rules after editing the text file**

If you make changes to the text file at any time after the rule is created, perform the following steps to apply the changes.

a. In the browser, access JMX port <29810 + workerID> (for example, 29811 for worker _1).

b. Within **BSM-Platform**, select the service called **MarbleWorker** and invoke the **reloadRules** method. This method is applied to all the customers served by this worker.

How to Define an API Rule in the Rule Repository

Within the Business Rule Repository, create an API rule that can be applied to multiple KPIs. This is done by cloning one of the three API rules, and setting default rule values for specific rule parameters. After the rule is applied to a KPI, you can edit its script within the CI Indicators tab at any time to change the rule logic for the specific KPI.

1. ## Clone an API rule

   Select **Admin> Service Health > Repositories > Business Rules**. In the Business Rule Repository page, clone one of the following rules: API Group and Sibling Rule, API Sample Rule, or API Duration-Based Sample Rule.

   For details on cloning a rule, see How to Customize a Business Rule Template in the Repository in the BSM Application Administration Guide.

2. ## Edit rule details

   a. Open the new rule for editing.

   b. In the **Name** field, rename the cloned rule.

   c. Edit the **KPI Calculation Script** rule parameter. In the **Default Value** field, enter the rule calculation script. The code that you enter is the default code for this rule, and appears in the CI Indicators tab for all KPIs assigned this rule. (Do not change any other fields.)

   d. If you are creating a Sample rule or Duration-Based Sample rule, edit the **Sample Fields** rule parameter. The sample fields that you enter are the default sample fields for this rule, and appear in the CI Indicators tab for all KPIs assigned this rule. (Do not change any other fields.)

   For details on these rule parameters, see the following sections (depending on the type of rule you are creating):

   ○ "API Group and Sibling Rule" on page 24

   ○ "API Sample Rule" on page 27

   ○ "API Duration-Based Sample Rule" on page 29

   The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
   **\\<Gateway Server root directory>\
   AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_
   API\index.html**.

3. ## Add the rule to the KPI's applicable rules list

   Add the new rule to the list of applicable rules already attached to the relevant KPI. For details, see the Applicable Rules parameter in New KPI/Edit KPI Dialog Box in the BSM Application Administration Guide.

How to Work with Tooltip Entries

The following section describes how to work with tooltip entries to display information calculated by the Rules API.

1. Select **Admin > Service Health > Repositories > Business Rules**. In the Rule Repository page, add any required tooltip entries for the new rule. The following table lists common tooltip entries and their corresponding value sources and formatting methods:

Tooltip Parameter	Value Source	Formatting Method
Business Rule	NODE.DIM.RULE.ID_CUST	ruleIDtoString
CI name	NODE.PROPS.BamNodeNameKey	toLowerCase
Held status since	NODE.DIM.RESULT.LastStatusChange	returnDateAsString
Status	NODE.DIM.RESULT.Status	getStatusString
Value	NODE.DIM.RESULT.Value	returnNumOfDigitAfterPoint

For details, see New/Edit Tooltip Parameter Dialog Box in the BSM Application Administration Guide.

2. If you have used the **kpi.setTooltip** method, you must set a corresponding tooltip entry in the Rule Repository as described above. In the **Value Source** field, type the name of the tooltip entry exactly as used in the code, and leave the **Formatting Method** field empty.

For example, if your code contains the method invocation **kpi.setTooltip("total_sales", value),** type **NODE.DIM.RESULT.total_sales** in the **Value Source** field.

How to Write to Log Files From the Rules API Code

Within your API rules, you can write to log files from rule methods using a **logger** object. There are five log levels: debug, info, warn, error and fatal. Each of these uses a specific logger method.

By default, only log method invocations of error and fatal severity are written to the log files. You can modify this within the log configuration files.

To write to log files using the Rules API:

1. Within the rule method, implement one of the following methods (listed in ascending order of severity):

   ■ **logger.debug("<API rule name> : log message");**

   ■ **logger.info("<API rule name> : log message");**

   ■ **logger.warn("<API rule name> : log message");**

   ■ **logger.error("<API rule name> : log message");**

   ■ **logger.fatal("<API rule name> : log message");**

   Type the name of your API rule inside the log message to identify each log message with its source rule.

2. The Rules API log files are found in the **<Data Processing server root directory>\HPBSM\log\marble_worker_<worker#>\RulesAPI** directory.

   Open one of the following files to view the log messages (depending on your rule type):

   ■ **groupAndSiblingRule.log** (for API Group and Sibling rules)

   ■ **sampleRule.log** (for API Sample rules)

   ■ **durationBasedSampleRule.log** (for API Duration-Based sample rules)

To modify the severity level written to a log file:

1. By default, only log method invocations of error and fatal severity are written to log files. To modify this setting, open the log configuration file located in **<Data Processing server root directory>HPBSM\conf\core\Tools\log4j\marble_worker\dashboard_rules.properties**.

2. In the line corresponding with your rule type, replace the string **${loglevel}** with the severity level you want logged (either DEBUG, INFO, WARN, ERROR, or FATAL). Edit one of the following lines, depending on your rule type:

   ■ Group and Sibling rules:
   log4j.category.com.mercury.am.rules.dashboard.blDashboardRules.
   simplifiedRule.groupAndSiblingRule.DashboardGroupAndSiblingRule = **${loglevel}**,
   bam.app.rules.api.group.appender

   ■ Sample rules: log4j.category.com.mercury.am.rules.dashboard.blDashboardRules.
   simplifiedRule.leaf.DashboardSimplifiedSampleBasedRule = **${loglevel}**,
   bam.app.rules.api.leafsample.appender

- Duration-Based Sample rules:
  log4j.category.com.mercury.am.rules.dashboard.blDashboardRules.
  simplifiedRule.leaf.DashboardSimplifiedTimeBasedRule = **${loglevel}**,
  bam.app.rules.api.leafduration.appender

How to Include a CI Property in Rules API Calculations

Within your API rules, you can include CI properties using the CI class **getPropertyValue** method, and the KPI class **getCiProperty** method. Only CI properties with one of the following qualifiers can be accessed with this method:

- **BLE_ATTRIBUTE** - SLM and Service Health

- **BLE_ONLINE_ATTRIBUTE** - Service Health only

To add this attribute to a CI class you must export the class, edit the class definition, and import it back to the server. When you open the exported class for editing, add the following xml to the required attribute:

```
<Attribute name="<attribute-name>" type="double" display-name="<attribute-display-name>">

  <Attribute-Qualifiers>

    <Attribute-Qualifier name="BLE_ATTRIBUTE"/>

  </Attribute-Qualifiers>

</Attribute>
```

Examples - API Sample Rule

This section provides examples of API Sample Rules. The following examples are described:

- "Example - Average Availability Rule" below

- "Example - Average Performance Rule" below

- "Example - Average Performance Rule Using a Rule Parameter Filter" on the next page

Example - Average Availability Rule

The following rule calculates average availability of samples, based on the u_iStatus sample field.

The rule logic is (available samples / total samples) * 100.

```
// This rule uses the u_iStatus sample field.
def sampleFields = ["u_iStatus"];
public void calculateKPI(CI ci, KPI kpi, List<Sample> samples) {
    // Keep total number of samples for this calculation cycle.
    def totalSamples = samples.size();
    // Create a variable to count available samples.
    def availableSamples = 0;
    /**
    * Go over the given samples. If a sample's u_iStatus is equal
to 0,
    * the sample is considered available.
    */
    samples.each {Sample currentSample->
    if (currentSample.u_iStatus == 0) {
        // Increase the count of available samples.
        availableSamples++;
        }
    }
    if (totalSamples > 0) {
        // Set KPI value, converted to percentage.
    kpi.setValue ((availableSamples/totalSamples)*100.0);
    }
}
```

Example - Average Performance Rule

The following rule calculates average performance in seconds, based on the dResponseTime and u_iStatus sample fields.

Only samples with a u_iStatus value of 0 (available samples) are used in the calculation. The rule logic is: sum(dResponseTime) / available samples.

```
// This rule uses the u_iStatus and dResponseTime sample field.
def sampleFields = ["u_iStatus", "dResponseTime"];
public void calculateKPI(CI ci, KPI kpi, List<Sample> samples) {
    // Create a variable to count available samples.
    def availableSamples = 0;
    // Create a variable to sum response times of available
samples.
    def totalResponseTime = 0;
    /**
     * Go over the given samples. If a sample's u_iStatus is equal
to 0,
     * the sample is considered available.
     */
    samples.each {Sample currentSample ->
        if (currentSample.u_iStatus == 0) {
            // Increase the count of available samples.
            availableSamples++;
            // Add the current sample's dResponseTime value to
totalResponseTime.
            totalResponseTime += currentSample.dResponseTime
        }
    }
    if (availableSamples > 0) {
        // Set KPI value, converted to percentage.
        kpi.setValue((totalResponseTime / availableSamples))
    }
}
```

Example - Average Performance Rule Using a Rule Parameter Filter

The following rule calculates average performance in seconds, based on the dResponseTime and u_iStatus sample fields.

Only samples with a u_iStatus value of 0 (available samples) are used in the calculation.

The rule uses an optional rule parameter: Response time limit. If this rule parameter value has been set in the Service Health Admin, samples with a dResponseTime value greater then the rule parameter value are not used in the calculation.

Note: A rule parameter with the same name must be set for the rule in the Rule Repository. For details, see New/Edit Rule Parameter or New/Edit Rule Threshold Dialog Box in the BSM Application Administration Guide.

The rule logic is: sum(dResponseTime) / available samples.

```
/ This rule use the u_iStatus and dResponseTime sample fields.
def sampleFields = ["u_iStatus", "dResponseTime"];
public void calculateKPI(CI ci, KPI kpi, List<Sample> samples) {
    // Create a variable to count available samples.
    def availableSamples = 0;
    // Create a variable to sum response times of available
samples.
    def totalResponseTime = 0;
    /**
     * Get the value of the rule parameter named "Response time
limit"
     * from the KPI, as defined for the KPI in Service Health
Admin.
     * This rule parameter is optional, so responseTimeLimit can be
null.
     */
    Long responseTimeLimit = kpi.getRuleParameter("Response time
limit")

    /**
     * Go over the given samples. If a sample's u_iStatus is equal
to 0,
     * the sample is considered available.
     */
    samples.each {Sample currentSample ->
        if (currentSample.u_iStatus == 0) {
            /**
            * Check the value of the rule parameter.
            * If it is not null (meaning the user has set a value),
            * and the sample's dResponseTime is greater than the
            * rule parameter value, the value is not valid.
            */
            boolean isSampleValid = true;
            if (responseTimeLimit != null) {
                // Check if ResponseTime exceeds the rule parameter
value.
                if (currentSample.dResponseTime >
responseTimeLimit) {
                    // The sample is not valid.
                    isSampleValid = false;
                }
            }
            if (isSampleValid) {
                // Increase the count of available samples.
                availableSamples++;
                // Add the sample's dResponseTime value to
totalResponseTime.
                totalResponseTime += currentSample.dResponseTime
            }
```

```
        }
    }
    if (availableSamples > 0) {
        // Set KPI value, converted to percentage.
        kpi.setValue((totalResponseTime / availableSamples))
    }
}
```

Examples - API Group and Sibling Rule

This section provides examples of API Group and Sibling Rules. The following examples are described:

-

-

-

-

-

-

-

Example - Worst Child Rule

The following rule finds the worst status from all of the KPIs of the calculated CI's child CIs, which are of the same type as the calculated KPI, based on active statuses only. Active statuses are **Critical**, **Major**, **Minor**, **Warning**, and **OK**.

```
public void calculateKPI(CI ci, KPI kpi) {
    // Get the calculated KPI's type ID (as defined in the Service
Health KPI Repository).
    int kpiId = kpi.getType();
    // Get a list of all of the KPIs of the calculated CI's child
CIs, which are of the same
    // type as the calculated KPI.
    List<KPI> childKpiList = ci.getChildrenKPIsByID(kpiId);
    // Create a variable to set the status of the calculated KPI,
    // only if an active status is found.
    boolean isActiveStatusFound = false;
    // Set the current worst status to OK; if a worse status is
found this will be updated.
    Status worstStatus = Status.OK;
    // Go over the list of child KPIs.
    childKpiList.each{KPI childKPI->
        // Get the child KPI's status.
        Status childKpiStatus = childKPI.status;
        // Check if the child KPI's status is an active status.
        if(childKpiStatus.isActive()){
            // Mark that an active status was found.
            isActiveStatusFound = true;
            // Check if the child KPI's status is worse than the
current worst status.
            if(childKpiStatus.isWorse(worstStatus)){
                // Update the worst status.
```

```
            worstStatus = childKpiStatus;
            }
        }
    }
    // Check if an active status was found in the child KPI.
    if(isActiveStatusFound){
        // Set the calculated KPI status.
        kpi.setStatus(worstStatus);
    }
}
```

Example - Worst Sibling Status Rule

The following rule finds the worst status from sibling KPIs, based on active statuses only. Active statuses are **Critical**, **Major**, **Minor**, **Warning**, and **OK**.

```
public void calculateKPI(CI ci, KPI kpi) {
    // Get a list of all the KPIs for the CI.
    List<KPI> ciKpiList = ci.getAllKPIs();
    /**
     * Create a variable to set the status of the calculated KPI,
     * only if an active status is found.
     */
    boolean isActiveStatusFound = false;
    // Set the current worst status to OK; if a worse status is
found this will be updated.
    Status worstStatus = Status.OK;
    // Go over the list of the CI's KPIs.
    ciKpiList.each {KPI ciKPI ->
        /**
         * Check that the CI's KPI is not the calculated KPI.
         * This is needed because getAllKPIs method returns all the
KPIs for the CI.
         */
        if (ciKPI !=  kpi) {
            /**
             * The ciKPI represents a sibling KPI of the calculated
KPI.
             * Get the sibling KPI's status.
             */
            Status siblingKpiStatus = ciKPI.status;
            // Update worstStatus if necessary.
            if (siblingKpiStatus.isActive()) {
                isActiveStatusFound = true;
                if (siblingKpiStatus.isWorse(worstStatus)) {
```

```
                            worstStatus = siblingKpiStatus;
                        }
                    }
                }
            }
        // Check if an active status was found in the sibling KPI.
        if (isActiveStatusFound) {
            // Set the calculated KPI's status.
            kpi.setStatus(worstStatus);
        }
    }
```

Example - Specific Child CI Group Rule

The following rule calculates KPI status based on the Availability KPI of a specific child CI (RTSM ID = "96c2df2b544683c7f79bb382d1d7b3a9").

If the child CI's Availability KPI value is 100, the calculated KPI's status is set to OK. All other values set the KPI's status to **Critical**.

Status is set only if the child CI exists, has the Availability KPI, and its Availability KPI has value.

```
public void calculateKPI(CI ci, KPI kpi) {
    /**
    * Get the Availability KPI for the child CI "tx_10 from
virtual_host_3".
    * The RTSM ID of "tx_10 from virtual_host_3" is
"96c2df2b544683c7f79bb382d1d7b3a9".
    *
    * Note: Within the UI, the following line can be written as
    * KPI childKPI = "tx_10 from virtual_host_3"."Availability"
    */
    KPI childKPI = ci.getChildKpiByChildId(KpiType.Availability,
"96c2df2b544683c7f79bb382d1d7b3a9");

    // Check if childKPI is not null. It is null if no child CI
with this RTSM ID exists, or if this CI does not have the
Availability KPI.
    if (childKPI != null) {

        // Check if the child KPI has a value.
        if (childKPI.valueExist) {
            if (childKPI.value == 100.0) {
                kpi.status = Status.OK
            }
            else {
```

```
                    kpi.status = Status.CRITICAL
                }
            }
        }
}
```

Example - Sibling Rule Based on Availability and Performance KPIs

The following rule calculates KPI status based on the status of sibling Availability and Performance KPIs.

If these KPIs do not exist or do not have active status, no status is set.

If these sibling KPIs exist and are both OK, the calculated KPI status is set to OK. Otherwise, its status is set to **Critical**. (Active statuses are **Critical**, **Major**, **Minor**, **Warning**, and **OK**.)

```
public void calculateKPI(CI ci, KPI kpi) {
    /**
     * Get the sibling KPI of type Availability.
     * If Availability KPI does not exist, null will be returned.
     */
    KPI availabilityKPI = ci.getKPI(KpiType.Availability);
    // Get the sibling KPI of type Performance.
    KPI performanceKPI = ci.getKPI(KpiType.Performance);
    if (availabilityKPI != null && performanceKPI != null) {
        // Both KPIs exist for this CI. Check if the KPIs status is
active.
        if (availabilityKPI.status.isActive() &&
performanceKPI.status.isActive()) {
            // Check the KPI's status.
            if (availabilityKPI.status == Status.OK &&
                performanceKPI.status == Status.OK) {
                /**
                 * Both statuses are active and both are OK. Set
this KPI's status to OK.
                 */
                kpi.status = Status.OK
            }
            else {
                /**
                 * Both statuses are active, and not both are OK.
                 * Set this KPI's status to CRITICAL
                 */
                kpi.status = Status.CRITICAL
```

```
            }
        }
    }
}
```

Example - Group Average Value by CI Type

The following rule calculates the average status of the KPIs of child CIs, which are of the same CI type as the calculated KPI.

Only child CIs of type "bpm_tx_from_location" are used in the calculation. If there are no child CIs of this type, or no child CI KPIs have value, no value is set for the KPI.

```
public void calculateKPI(CI ci, KPI kpi) {
    // Get the calculated KPI's type ID (as defined in the Service
Health KPI Repository).
    int kpiId = kpi.getType();
    // Get a list of the KPIs of the child CIs, which are of the
same CI type as the calculated
    // KPI, whose CI type is  "bpm_tx_from_location".
    List<KPI> bpmTxFromLocationChildKpiList =
ci.getChildrenKPIsByIDAndCiType(kpiId, "bpm_tx_from_location")
    // Create a variable to sum the total values from child KPIs.
    // If no child exists or no child has value the variable will
remain null.
    Double totalChildValue = null;
    // Write information to the log file.
    logger.debug("DashboardGroupAvgValueByCiTypeRule : number of
child CIs with type bpm_tx_from_location: " +
bpmTxFromLocationChildKpiList.size())
    // Go over the list of child KPIs.
    bpmTxFromLocationChildKpiList.each {KPI childKPI ->
        // Sum values of the child KPIs using the Utils class,
which handles null values.
        totalChildValue = Utils.sum(totalChildValue,
childKPI.value);
    }
    // Set the calculated KPI's value to the average value, using
the Utils class.
    // If totalChildValue is null, null value will be set.
    kpi.value = Utils.divide(totalChildValue,
bpmTxFromLocationChildKpiList.size());
}
```

Example - Worst Health Indicator Rule

The following rule finds the worst status from all of the health indicators (HIs) of the calculated CI, based on active statuses only. Active statuses are **Critical**, **Major**, **Minor**, **Warning**, and **OK**.

```
public void calculateKPI(CI ci, KPI kpi) {
    // Get all health indicators.
    List<HI> his = ci.getHIs();
    // Create a variable to set the status of the calculated KPI,
    // only if an active status is found.
    boolean isActiveStatusFound = false;
    // Set the current worst status to OK;
    // if a worse status is found this will be updated.
    Status worstHiStatus = Status.OK;
    his.each {HI hi ->
        Status hiStatus = hi.getStatus();
        // Check if the current HI status is an active status.
        if (hiStatus.isActive()) {
            // Mark that an active status was found.
            isActiveStatusFound = true;
            // Check if the child KPI's status is worse than the
current worst status.
            if (hiStatus.isWorse(worstHiStatus)) {
                // Update the worst status.
                worstHiStatus = hiStatus;
            }
        }
    }
    // Check if an active status was found in the child KPI.
    if (isActiveStatusFound) {
        // Set the calculated KPI status.
        kpi.setStatus(worstHiStatus);
    }
}
```

Example - Using Groovy Closure

The following rule sets the calculated KPI's status to Critical, if at least one Availability KPI with Major status exists for the calculated CI's child CIs.

This rule illustrates Groovy Closure. Refer to http://groovy.codehaus.org/Closures for more information.

```
public void calculateKPI(CI ci, KPI kpi) {
    /**
```

```
    * Use Groovy Closure with the CI class getChildrenKPIs method,
    * to get List of KPIs from the CI`s child CIs, where
    * 1. KPI type is Availability
    * 2. Status is MAJOR

    Closure description:
    { KPI childKPI ->
        childKPI.type == KpiType.Availability.getID("DASHBOARD") &&
childKPI.status == Status.MAJOR
    }
    The Closure defines one parameter named childKPI of type KPI.
    Each KPI from the CI`s child CIs will be passed to the Closure
by the getChildrenKPIs method.
    The Closure body returns a boolean value based on the logical
expression result.
    Each KPI that the Closure body will return true for, will be
part of the returned List
    The expression KpiType.Availability.getID("DASHBOARD") returns
an int representing the Availability KPI ID from the Service Health
KPI Repository.
    */

    List<KPI> kpiList = ci.getChildrenKPIs {KPI childKPI ->
        childKPI.type == KpiType.Availability.getID("DASHBOARD") &&
childKPI.status == Status.MAJOR
    }
    // Check if such a KPI exists.
    if (kpiList.isEmpty()) {
        // No such KPI exists.
        // Write to a log file at debug level.
        logger.debug "Closure Rule: no Availability KPI with  MAJOR
status exist"
    }
    else {
        //  At least one Availability KPI with MAJOR status exists.
        logger.debug("Closure Rule: At least one Availability KPI
with  MAJOR status exist")
        // Set calculated KPI status to CRITICAL.
        kpi.status = Status.CRITICAL;
    }
}
```

Chapter 2

Service Health External APIs

This section includes:

- "Retrieve Indicator Data API" on the next page. You can use this API to access KPI over time statuses, KPI definitions, and indicator statuses.

- "Reset Health Indicator State API" on page 59. In certain event flows, you might have an HI showing that a problem has occurred but no event has closed the problem, even though the problem was fixed. After dealing with the problem, you can use this API to reset the HI's state to **Normal.**

- "Service Health Database Query API" on page 61. You can use this API to query the database and return a list of views in XML format.

Retrieve Indicator Data API

The following external API can be used to access KPI over time statuses, KPI definitions, and indicator statuses.

This section includes the following topics:

- "Get KPI Over Time Statuses" below
- "Get KPI Definitions" on page 55
- "Get Indicator Statuses" on page 56

The service log file is located under: **<Gateway server root directory>\log\EJBContainer\serviceHealthExternalAPI.log**.

Return values are supported in XML and JSON formats.

Authentication should be done using basic access authentication method. For details and examples refer to http://en.wikipedia.org/wiki/Basic_access_authentication.

Get KPI Over Time Statuses

You can use the following to get KPI over time statuses.

API Syntax

```
http://<Gateway Server>/topaz/servicehealth/customers/<Customer Id>/
kpiOverTime?ciIds=<CI ID>&startDate=<Start Date>&endDate=<End Date>
```

The API uses the following parameters:

- **customerId.** Customer ID (use **1** for non-HP SaaS deployment).
- **ciId**. Mandatory; use comma-separated CI IDs.
- **startDate.** Mandatory; start time for the KPI status (value representing the date in seconds since January 1 1970).
- **endDate**. Mandatory; end time for the KPI status (value representing the date in seconds since January 1 1970).
- **view.** Optional; retrieve the results in the context of a local impact view (default is global view). For details, see Local Impact Views in the BSM User Guide.
- **kpiId.** Optional; use comma separated KPI internal IDs as in the repository UI (default is empty for all KPIs). For details, see List of Service Health KPIs in the BSM Application Administration Guide.

The following is an example of the API and its output:

```
http://host.devlab.ad/topaz/servicehealth/customers/1/kpiOverTime?
ciIds=0b656ce308022a6739e3e726497fda6a&startDate=1296499370
&endDate=1296501466
```

```
<kpiStatuses>
    <kpiStatus>
        <ciId>0b656ce308022a6739e3e726497fda6a</entityId>
        <ciDisplayLabel>ATM 1610</ciDisplayLabel>
        <kpiType>6</kpiType>
        <kpiDisplayName>Application Performance</kpiDisplayName>
        <timeStamp>1296499370</timeStamp>
        <status>20</status>
        <statusDisplayName>OK</statusDisplayName>
        <duration>311</duration>
    </kpiStatus>
    <kpiStatus>
        <ciId>0b656ce308022a6739e3e726497fda6a</entityId>
        <ciDisplayLabel>ATM 1610</ciDisplayLabel>
        <kpiType>6</kpiType>
        <kpiDisplayName>Application Performance</kpiDisplayName>
        <timeStamp>1296499681</timeStamp>
        <status>-2</status>
        <statusDisplayName>No Data</statusDisplayName>
        <duration>1785</duration>
    </kpiStatus>
    <kpiStatus>
        <ciId>0b656ce308022a6739e3e726497fda6a</entityId>
        <ciDisplayLabel>ATM 1610</ciDisplayLabel>
        <kpiType>6</kpiType>
        <kpiDisplayName>Application Performance</kpiDisplayName>
        <timeStamp>1296501466</timeStamp>
        <status>20</status>
        <statusDisplayName>OK</statusDisplayName>
        <duration>13334</duration>
    </kpiStatus>
    <kpiStatus>
        <ciId>0b656ce308022a6739e3e726497fda6a</entityId>
        <ciDisplayLabel>ATM 1610</ciDisplayLabel>
        <kpiType>7</kpiType>
        <kpiDisplayName>Application Availability</kpiDisplayName>
        <timeStamp>1296428400</timeStamp>
        <status>0</status>
        <statusDisplayName>Critical</statusDisplayName>
        <duration>69663</duration>
    </kpiStatus>
</kpiStatuses>
```

The output fields are as follows:

Field	Description
ciId	CI ID

Field	Description
ciDisplayLabel	CI display label
kpiType	KPI ID (see "Get KPI Definitions" below below)
kpiDisplayName	KPI display name
timeStamp	Start time for the KPI status; value representing the date in seconds since January 1 1970
status	KPI status (see Get Indicator Statuses below)
statusDisplayName	KPI status display name
duration	Duration of the KPI's status in seconds.

Return Codes

The API returns the following return codes:

Name	Error Code	Description
BAD_REQUEST	400	• Start date is after the end date • Start date is in the future • startDate, endDate or ciIDs are missing
UNAUTHORIZED	401	User has no permission for the selected view
INTERNAL_ SERVER_ERROR	500	• Result size has exceeded the maximum quota • General failure

Get KPI Definitions

You can use the following to retrieve the KPIs defined in the system.

API Syntax

```
http://<Gateway Server>/topaz/servicehealth/customers/<CustomerId>/
repositories/indicators/kpis/<kpiId>
```

The API uses the following parameters:

- **customerId.** Customer ID (use **1** for non-HP SaaS deployment).

- **kpiIds.** Optional; leave empty for all KPIs (default), or enter a KPI internal ID as in the repository UI, to select a specific KPI. For details, see List of Service Health KPIs in the BSM Application Administration Guide.

The following is an example of the API and its output:

```
http://host.devlab.ad/topaz/servicehealth/customers/1/repositories/
indicators/kpis/
```

```
<kpis>
    <kpi>
        <id>1</id>
        <name>Legacy System</name>
    </kpi>
    <kpi>
        <id>1311</id>
        <name>Value</name>
    </kpi>
    <kpi>
        <id>1310</id>
        <name>Exceptions</name>
    </kpi>
</kpis>
```

The output fields are as follows:

Field	Description
id	KPI internal ID as in the repository UI; for details see List of Service Health KPIs in the BSM Application Administration Guide.
name	KPI name

Return Codes

The API returns the following return codes:

Name	Error Code	Description
NOT_FOUND	404	KPI not found
INTERNAL_ SERVER_ERROR	500	General failure

Get Indicator Statuses

You can use the following to retrieve indicator statuses.

API Syntax

```
http://<Gateway Server>/topaz/servicehealth/customers/<CustomerId>/
repositories/indicators/statuses
```

The API uses the following parameters:

- **customerId.** Customer ID (use **1** for non-HP SaaS deployment).

The following is an example of the API and its output:

```
http://host.devlab.ad/topaz/servicehealth/customers/1/repositories/
indicators/statuses
```

```
<targets>
    <target>
        <id>20</id>
    <name>OK</name>
    </target>
    <target>
        <id>15</id>
        <name>Warning</name>
    </target>
    <target>
        <id>10</id>
        <name>Minor</name>
    </target>
    <target>
        <id>5</id>
        <name>Major</name>
    </target>
    <target>
        <id>0</id>
        <name>Critical</name>
    </target>
    <target>
        <id>-1</id>
        <name>Info</name>
    </target>
    <target>
        <id>-2</id>
        <name>No Data</name>
    </target>
    <target>
        <id>-4</id>
        <name>Downtime</name>
    </target>
</targets>
```

The output fields are as follows:

Field	Description
id	KPI status internal ID
name	KPI status name

Return Codes

The API returns the following return codes:

Name	Error Code	Description
INTERNAL_ SERVER_ERROR	500	General failure

Reset Health Indicator State API

In certain event flows, you might have an HI showing that a problem has occurred but no event has closed the problem, even though the problem was fixed. After dealing with the problem, you might want to reset the HI's state to **Normal** (default). For details on resetting HI state within Service Health, see Health Indicator Component in the BSM User Guide.

The Reset HI State API enables users outside of the BSM user interface to reset event-based HIs to their default state, using the HTTP-based REST protocol.

You can reset all HIs on a specific CI, or reset a specific HI.

This REST API is case-sensitive, and uses the **PUT** method.

> **Note:** This API can impact the overall performance of your system; consult with HP Professional Services before using the API.

API Syntax

- To reset all HIs related to a CI:

```
http://<Gateway
Server>/topaz/servicehealth/customers/<CustomerId>/cis/<CI
ID>/his/reset
```

- To reset a specific HI:

```
http://<Gateway
Server>/topaz/servicehealth/customers/<CustomerId>/cis/<CI
ID>/his/<HI name>/reset
```

- To reset a specific subcomponent of an HI:

```
http://<Gateway
Server>/topaz/servicehealth/customers/<CustomerId>/cis/<CI
ID>/his/<HI name>/reset?subcomponent=<subcomponent name>
```

HI name refers to the name of the HI as defined in the indicator repository, and not to the HI's display label.

Return Codes and Log File

The API returns the following return codes:

Name	Error Code	Description
OK	200	Success
UNAUTHORIZED	401	The user is not authorized for the customer

Name	Error Code	Description
NOT_FOUND	404	• CI not found • HI not found • Bad request (syntax error)
INTERNAL_ SERVER_ERROR	500	• RTSM error • Repositories error • Online engine error

The service log file is located under: **<Gateway server root directory>\log\EJBContainer\serviceHealthExternalAPI.log.**

In addition, the service writes to the Audit log on each HI reset.

Service Health Database Query API

You can use the Service Health API to query the database and return a list of views in XML format.

> **Tip:** You can use XSLT to convert the XML output into any other format (commonly text or HTML). For example, using basic XSLT transformations, you can produce HTML reports that are formatted to fit on mobile devices. These reports can be served via a mobile portal to display critical Business Service Management views on users' mobile phones.

Query Syntax

The basic syntax of the query is as follows:

```
http://<Gateway Server>/topaz/bam/BAMOpenApi?customerId=<customer
ID>&userName=<user name>&password=<password>&command=<command
parameter>
```

Depending on the **command** parameter defined, additional parameters may also be included.

Main Parameters Used in the Query

The following table lists the parameters that must be defined in the query.

Parameter	Description
customerID	BSM customers should specify 1. HP Software-as-a-Service customers should specify their unique customer ID.
userName	Specify a user name defined in BSM. The query does not encrypt the login credentials.
password	Specify the password for the user name provided. The query does not encrypt the login credentials.
command	Specify one of the following values: **getViews** – Specify to retrieve all views from the Run-time Service Model (RTSM). No other parameters are required. **getNodes** – Specify to retrieve all child nodes of a specified view (you must also specify the view for which to retrieve child nodes in the **viewName** parameter); if using this command parameter you can also set the following parameters: **showTooltip**, **depth**, **layout**, **xsltURL**, **responseContentType**
viewName	If the **getNodes** command parameter is defined, include this parameter in the query and specify the view to retrieve. You can set the value to **ticker_all_views** to retrieve all views and their nodes.

Parameter	Description
showTooltip	If the **getNodes** command parameter is defined, you can include this parameter in the query to specify whether to display Service Health's KPI tooltip data, either **true** to display data or **false** to not. The default value is **false**.
depth	If the **getNodes** command parameter is defined, you can include this parameter in the query to specify the number of levels in the view to display. The default value is **1**.
layout	If the **getNodes** command parameter is defined, you can include this parameter in the query to specify the layout for the query results, either **hierarchical** or **flat.** In flat mode all nodes are retrieved in a flat list, and in hierarchical mode nodes are retrieved in the same hierarchy as in the view. The default value is **flat**.
xsltURL	If the **getNodes** command parameter is defined, you can include this parameter in the query to specify a URL to an .xslt file that transforms the .xml-format result of the query.
responseContentType	If the **getNodes** command parameter is defined, and the **xsltURL** parameter is included in the query, you can include this parameter in the query to specify the response MIME type.

Query Examples

Below are examples of queries and the data they return.

The following query returns a flat list of all views in the Run-time Service Model (RTSM):

```
http://myserver/topaz/bam/BAMOpenApi?customerId=1
&userName=admin&password=admin&command=getViews
```

The following query returns a hierarchical tree showing KPI status and tooltip information for the Service Measurements view, to a depth of three child nodes:

```
http://myserver/topaz/bam/BAMOpenApi?customerId=1&userName=
admin&password=admin&command=getNodes&viewName=Service%20
Measurements&showTooltip=true&depth=3&layout=hierarchical
```

Part 2

Service Level Management

Chapter 3

SLM External API

You can use the SLM external API to retrieve SLA configuration properties and SLA calculation results, in order to consume this data in external applications.

The API enables you to access and process SLM data from outside BSM. For details, see:

The service log file is located under: **<Gateway server root directory>\log\EJBContainer\slmExternalAPI.log**.

Return values are supported in XML and JSON formats.

Authentication should be done using basic access authentication method. For details and examples refer to http://en.wikipedia.org/wiki/Basic_access_authentication.

Get SLA Configuration Data

You can use the following to get SLA configuration data.

API Syntax

```
http://<Gateway Server>/topaz/slm/customers/<CustomerId>/sla/<slaId>
```

The API uses the following parameters:

- **customerId.** Customer ID (use **1** for non-HP SaaS deployment).

- **slaId.** SLA RTSM ID (non-mandatory parameter; leave empty to get all SLAs in the system).

The following is an example of the API output:

```
<sla>
    <id>c808d3ddce3d849c1cc0d667bec7f6cc</id>
    <name>TestSLA3</name>
    <description></description>
    <details></details>
    <timezone>Central Standard Time</timezone>
    <type>SLA</type>
    <creator>administrator</creator>
    <url></url>
    <classification>Formal</classification>
    <customer>
        <customerId>e35cded4b3a628a8da4b9cf352007467</customerId>
        <customerName>IT Department</customerName>
    </customer>
    <provider>
        <providerId>ae9ea545c5f99adb52d7edd641bbf8ef</providerId>
        <providerName>Customers</providerName>
    </provider>
    <state>Running</state>
    <startDate>1290938400</startDate>
    <endDate>1608631200</endDate>
    <targets>
        <target>0</target>
        <target>5</target>
        <target>20</target>
        <target>10</target>
        <target>15</target>
    </targets>
    <trackingPeriods>

     <trackingPeriod>3ef2dfe04fa07c349e72ca9e7b04e2af</trackingPeri-
od>
```

```
    <trackingPeriod>bffe0ea9334ff7f309e969eec3c9c266</trackingPeri-
od>

    <trackingPeriod>a562b09777271425835abadb12f32e73</trackingPeri-
od>

    <trackingPeriod>f9f77b20f986fbb2eb064b0a30ece93d</trackingPeri-
od>

    <trackingPeriod>4ecf36e0eb88816745a8849db029c73f</trackingPeri-
od>
    </trackingPeriods>
    <calendars>
        <calendar>ecf840eef788851986195301aba206fd</calendar>
    </calendars>
</sla>
```

The output fields are as follows:

Field	Description
slaId	SLA ID
name	SLA name
description	SLA description
details	SLA details
type	OLA, SLA, or UC
creator	Username of the user which created the SLA
timezone	SLA time zone
url	Link to the contract details from an external source.
classification	Formal or informal
customer	CI ID and name of the organization specified as the customer of the SLA.
provider	CI ID and name of the organization specified as the provider of the SLA.
state	Preliminary, Running, or Terminated
startDate	SLA start date, long value representing the date in milliseconds since January 1, 1970. 00:00:00 GMT.
endDate	SLA end date, long value representing the date in milliseconds since January 1, 1970.
targets	IDs of the SLA targets

Field	Description
trackingPeriods	IDs of the tracking periods
calenders	IDs of the SLA calendars

Return Codes

The API returns the following return codes:

Name	Error Code	Description
UNAUTHORIZED	401	User has no view permission for selected SLA
NOT_FOUND	404	SLA not found
INTERNAL_ SERVER_ERROR	500	General failure

Get SLA Calculation Results

You can use the following to retrieve the SLA calculation results; this API supports filtering by CI, KPI, tracking periods, and calendars.

API Syntax

```
http://<Gateway Server>/topaz/slm/customers/
<CustomerId>/ciSummary/<view>/sla/ <slaID>?
cal-
endar=<calendarId>&startDate=<startDate>&ciIds=<ciIds>&kpiId=<kpiId>
```

The API uses the following parameters:

- **customerId.** Customer ID (use **1** for non-HP SaaS deployment).

- **view.** Mandatory; report view. Use one of the following: hour, day, week, month, quarter, year, weekToDate, monthToDate, quarterToDate, yearToDate or slaToDate.

- **slaId.** SLA ID (use **"all"** for all SLAs).

- **calendars.** Optional, SLA calendar ID (leave empty for all SLA defined calendars, see "Get Calendars" on page 70).

- **startDate.** Mandatory for closed periods only, in seconds.

- **endDate.** Mandatory for closed periods only, in seconds.

- **ciIds.** Optional. Comma separated CI IDs; leave empty for all SLA CIs (or SLA nodes when slaId is **"all"**).

- **kpiId.** Optional. KPI internal ID (as it appears in the KPI repository UI); leave empty for all KPIs. This API does not support Outage KPI.

The following is an example of the API output:

```
<slaStatus>
    <slaId>2c4d69f2784021a6a94b7a40100ba31f</slaId>
    <ciId>b8ae7539b0cb338f3bce84b4866647eb</id>
    <startDate>1293832800</startDate>
    <endDate>1294225200</endDate>

    <trackingPeriod>a562b09777271425835abadb12f32e73</trackingPerio-
d>
    <calendar>dd9ef8826c553d3e217fa0b3bf03f0a0</calendar>
    <kpiType>220</kpiType>
    <kpiDisplayName>SLM Status</kpiDisplayName>
    <kpiStatus>0</kpiStatus>
    <statusDisplayName>Failed</statusDisplayName>
    <kpiValue>0</kpiValue>
</slaStatus>
```

The output fields are as follows:

Field	Description
slaId	SLA ID
ciId	CI ID
startDate	Period start time, long value representing the date in seconds since January 1 1970
endDate	Period end time, long value representing the date in seconds since January 1 1970
trackingPeriod	Tracking period of the sample: hour, day, week, month, quarter, year, SLA period (see "Get Tracking Periods" on page 72).
calendar	Calendar of the sample (see "Get Calendars" on the next page).
kpiType	KPI ID (see "Get KPIs" on page 74).
kpiDisplayName	KPI display name
kpiStatus	KPI status ID (see "Get Indicator Statuses" on page 76).
statusDisplayName	KPI status display name
kpiValue	KPI numeric value

Return Codes

The API returns the following return codes:

Name	Error Code	Description
BAD_REQUEST	400	- Unsupported view - Start date is after the end date - startDate, endDate are missing for closed view (closed period)
UNAUTHORIZED	401	User has no view permission for selected SLA
INTERNAL_ SERVER_ ERROR	500	- Result size has exceeded the maximum quota - General failure

Get Calendars

You can use the following to retrieve the calendars defined in the system.

API Syntax

```
http://<Gateway Server>/topaz/slm/customers/
<CustomerId>/repositories/calendars/<calendarId>
```

The API uses the following parameters:

- **customerId.** Customer ID (use **1** for non-HP SaaS deployment).

- **calendarId.** Calendar ID (leave empty for all calendars in the system).

The following is an example of the API output:

```
<calendars>
    <calendar>
        <id>5c6c09ec3d61db775333bb5beb5ea863</id>
        <name>testcalc</name>
        <description></description>
        <scheduleDescription>Friday 3:00 AM - 3:30 AM, Wednesday
2:30 PM - 3:00         PM, Thursday 5:30 PM - 6:00 PM, Wednesday
3:30 AM - 4:30 AM, Tuesday 11:00 AM - 11:30 AM, Friday 10:00 AM -
10:30 AM, Monday 2:00 AM - 2:30 AM, Wednesday 7:00 AM - 7:30 AM,
Thursday 10:30 AM - 11:00 AM, Sunday, Monday 6:30 AM - 7:00 AM,
Friday 2:00 PM - 2:30 PM, Monday 8:00 PM - 8:30 PM, Thursday 9:00
PM - 9:30 PM
        </scheduleDescription>
    </calendar>
    <calendar>
        <id>ecf840eef788851986195301aba206fd</id>
        <name>24x7</name>
        <description>24 hours, 7 days a week</description>
        <scheduleDescription>Sunday, Monday, Tuesday, Wednesday,
Thursday, Friday, Saturday 12:00 AM - 12:00 AM
        </scheduleDescription>
    </calendar>
    <calendar>
        <id>86594b724cb03e2647b2a86b08103a28</id>
        <name>Business Hours</name>
        <description>8:00-17:00, Monday-Friday</description>
        <scheduleDescription>Monday, Tuesday, Wednesday, Thursday,
Friday 8:00 AM - 5:00 PM
        </scheduleDescription>
    </calendar>
</calendars>
```

The output fields are as follows:

Field	Description
id	Calendar ID
name	Calendar display name
description	Calendar description
scheduleDescription	A string representing the calendar scheduling configuration.

Return Codes

The API returns the following return codes:

Name	Error Code	Description
NOT_FOUND	404	Calendar not found
INTERNAL_ SERVER_ERROR	500	General failure

Get Tracking Periods

You can use the following to retrieve the tracking periods defined in the system.

API Syntax

```
http://<Gateway Server>/topaz/slm/customers/
<CustomerId>/trackingPeriods/<trackingPeriodId>
```

The API uses the following parameters:

- **customerId.** Customer ID (use **1** for non-HP SaaS deployment).

- **trackingPeriodId.** Tracking period ID (leave empty for all tracking periods in the system).

The following is an example of the API output:

```
<trackingPeriods>
    <trackingPeriod>
        <id>a562b09777271425835abadb12f32e73</id>
        <name>Month</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>dd613d616284c14ec848a4fa5d939655</id>
        <name>Hour</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>2e57ee4d9c4f6b50a1d01385a861aa4b</id>
        <name>Day</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>4ecf36e0eb88816745a8849db029c73f</id>
        <name>Week</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>3ef2dfe04fa07c349e72ca9e7b04e2af</id>
        <name>Quarter</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>f9f77b20f986fbb2eb064b0a30ece93d</id>
        <name>Year</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>bffe0ea9334ff7f309e969eec3c9c266</id>
        <name>SLA period</name>
    </trackingPeriod>
</trackingPeriods>
```

The output fields are as follows:

Field	Description
id	Tracking period ID
name	Tracking period display name

Return Codes

The API returns the following return codes:

Name	Error Code	Description
INTERNAL_ SERVER_ERROR	500	General failure

Get KPIs

You can use the following to retrieve the KPIs defined in the system.

API Syntax

```
http://<Gateway Server>/topaz/slm/customers/
<CustomerId>/repositories/indicators/kpis/<kpiId>
```

The API uses the following parameters:

- **customerId.** Customer ID (use **1** for non-HP SaaS deployment).
- **kpiId.** KPI ID (leave empty for all KPIs in the system).

The following is an example of the API output:

```
<kpis>
    <kpi>
        <id>1</id>
        <name>Legacy System</name>
    </kpi>
    <kpi>
        <id>1311</id>
        <name>Value</name>
    </kpi>
    <kpi>
        <id>1310</id>
        <name>Exceptions</name>
    </kpi>
    <kpi>
        <id>615</id>
        <name>Operational Status</name>
    </kpi>
    <kpi>
        <id>6</id>
        <name>Application Performance</name>
    </kpi>
    <kpi>
        <id>7</id>
        <name>Application Availability</name>
    </kpi>
    <kpi>
        <id>1500</id>
        <name>Generic</name>
    </kpi>
</kpis>
```

The output fields are as follows:

Field	Description
id	KPI ID
name	KPI display name

Return Codes

The API returns the following return codes:

Name	Error Code	Description
NOT_FOUND	404	KPI not found
INTERNAL_ SERVER_ERROR	500	General failure

Get Indicator Statuses

You can use the following to retrieve indicator statuses.

API Syntax

```
http://<Gateway Server>/topaz/slm/customers/
<CustomerId>/repositories/indicators/targets
```

The API uses the following parameter:

customerId. Customer ID (use **1** for non-HP SaaS deployment).

The following is an example of the API output:

```
<targets>
    <target>
        <id>20</id>
        <name>Exceeded</name>
    </target>
    <target>
        <id>15</id>
        <name>Met</name>
    </target>
    <target>
        <id>10</id>
        <name>Minor Breached</name>
    </target>
    <target>
        <id>5</id>
        <name>Breached</name>
    </target>
    <target>
        <id>0</id>
        <name>Failed</name>
    </target>
    <target>
        <id>-4</id>
        <name>Downtime</name>
    </target>
    <target>
        <id>-2</id>
        <name>No Data</name>
    </target>
</targets>
```

The output fields are as follows:

Field	Description
id	Target/Status ID (as it appears in the calculation results)
name	Target/Status display name

Return Codes

The API returns the following return codes:

Name	Error Code	Description
INTERNAL_ SERVER_ERROR	500	General failure

Chapter 4

SLM Rules API

> **Note:** In BSM versions 9.00 and later, the rules that calculate indicator statuses and values based on samples ("API Sample Rule" on page 89 and "API Duration-Based Sample Rule" on page 91) are used to calculate metric-based health indicators (HIs).
>
> Throughout the Rules API documentation, you will see references to various methods used to calculate KPIs. **In BSM versions 9.00 and later, when calculating sample-based values, these methods are used to calculate metric-based HIs.**

This chapter describes how to use the Rules API to create business rules to calculate Key Performance Indicators (KPIs). The default Service Level Management rules appear in the section List of Service Level Management Business Rules in the BSM Application Administration Guide.

The recommended way to create new rules is with the Rules API. The Rules API enables you to create rules using the Groovy dynamic scripting language with Groovy runtime environment 1.7.3 and earlier. Users of this API should be familiar with Groovy and Java syntax, and with BSM administration and applications.

The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
**\\<Gateway Server root directory>\
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html**.

> **Note:** Extensive use of rules which use the Rules API will affect the performance of SLA calculation.

Types of API Rules

The following are the API rules in Service Level Management:

- **Simplified Average Rule and Simplified Duration-Based Average Rule.** These rules calculate health indicator average values based on data taken from sample fields, and enables you to manipulate these values. These rules are simpler to define than other API rules, and only require you to enter the sample field you want used in the calculation. For details, see "API Simplified Average Rules" on page 80.

- **Group and Sibling Rule**. This rule calculates KPIs based on aggregated values received from other KPIs, rather than from original sample data. The received data can come from the KPIs of child CIs (group), or from another KPI associated with the same CI (sibling). For details, see "API Group and Sibling Rule" on page 81.

- **Sample Rule**. This rule calculates KPIs based on data taken from sample fields. For details, see "API Sample Rule" on page 89.

- **Duration-Based Sample Rule**. This rule calculates KPIs based on data taken from sample fields, and uses both the sample's value and its duration within the rule calculation (for example,

availability over time or backlog over time). For details, see "API Duration-Based Sample Rule" on page 91.

- **Outage by Samples Rule**. This rule calculates outages based on data received from samples. For details, see "API Outage by Samples Rule" on page 95.

For details on how sample rules are calculated, see "Sample Rule Calculation Mechanism - Overview" on page 84.

Creating API Rules

Rules can be created using the Rules API in three ways:

- Using the KPI Definition page to create a rule for a specific KPI.

- Using a text file to create a new rule for multiple KPIs.

- Using a clone of an API rule template in the rule repositories.

These methods are described in "Creating Rules with the Rules API" on page 97.

Tooltips and Log Files

To display KPI information in tooltips when working with the Rules API, see "How to Work with Tooltip Entries" on page 103.

You can write to log files from the Rules API code, as described in "How to Write to Log Files From the Rules API Code" on page 105.

API Simplified Average Rules

The API Simplified Average Rule and API Simplified Duration-Based Average Rule calculate health indicator average values based on data taken from sample fields, and enable you to easily manipulate these calculations.

API Simplified Average Rule

The Simplified Average Rule calculates an average of sample field values, uses the rule parameters: Enumerator and Denomination.

- In the **Enumerator** area, enter the name of the sample field you want to use in the calculation in the following format: **sample.<field name>**.

  To calculate the average value of the sample field dValue, type `sample.dValue` in the Enumerator. The rule takes the values of this field from the samples that come in during the calculation cycle, and calculates their average. This average becomes the value of the health indicator for the cycle.

  You can also use mathematical expressions to manipulate these calculations. For example, suppose you have an EMS integration sending samples with two values: attr1 representing sent text messages, and attr2 representing failed text messages. To calculate the average number of successful messages (attr1-attr2), type `sample.attr1-sample.attr2` in the Enumerator.

- You can use the **Denomination** area to perform actions on the above value; the number or value you enter in this area is used to divide the Enumerator result.

  For example, if you enter `sample.dValue` in the Enumerator, and `2` in the Denomination, the average of the dValue fields on the samples is divided by two, and the health indicator value is half the average of the sample fields.

API Simplified Duration-Based Average Rule

The Simplified Duration-Based Average Rule calculates an average of sample field values, taking the duration values from the samples themselves.

In the **Enumerator** field, enter the name of the sample field you want to use in the calculation in the following format: **Sample.<field name>**.

The rule takes the durations from the samples and calculates the average of sample field value, divided by their durations.

For example, if the value of the specified field was 10 for 40 minutes and 5 for 20 minutes, the rule calculates (10x40) + (5x2) / 60 = 8.33. The health indicator's average value for this hour will be 8.33.

API Group and Sibling Rule

An API Group and Sibling Rule calculates KPIs based on data received from other KPIs, rather than from original sample data. The received data can come from the KPIs of child CIs, or from other KPIs or HIs associated with the same CI.

The KPI is calculated based on the aggregated values of the group or sibling KPIs or HIs. The calculated KPI results represent the aggregated results.

Note: If you are creating a sibling rule, make sure that the KPI is calculated after its sibling KPIs, as defined by the KPI's Calculation Order field. For details, see KPIs Repository page.

Group and Sibling Rule Methods and Fields

The Group and Sibling rule implements the Rules API Interface **GroupAndSiblingCalculator**, using the following guidelines:

- In this interface, the only method is **calculateKPI**. The method signature is:

  ```
  public void calculateKPI(CI ci, KPI kpi)
  ```

- The **calculateKPI** method includes the parameters **ci** and **kpi**, which represent the current CI, and the KPI or HI whose value the API rule calculates.

- The **ci** parameter type is **CI**, and is used as an accessor to KPIs of child CIs or sibling KPIs, or HIs on the CI. The **KPI** objects returned by **CI** are used to get the aggregated values of these KPIs or HIs.

- The **kpi** parameter type is **KPI**, and is used to set calculation results.

In the following illustration, the Calculated KPI is calculated based on the sibling or child KPIs, and it is represented by the **kpi** parameter.

The CI to which the Calculated KPI is assigned, is represented by the **ci** parameter, and it is an accessor to the other KPIs and HIs.
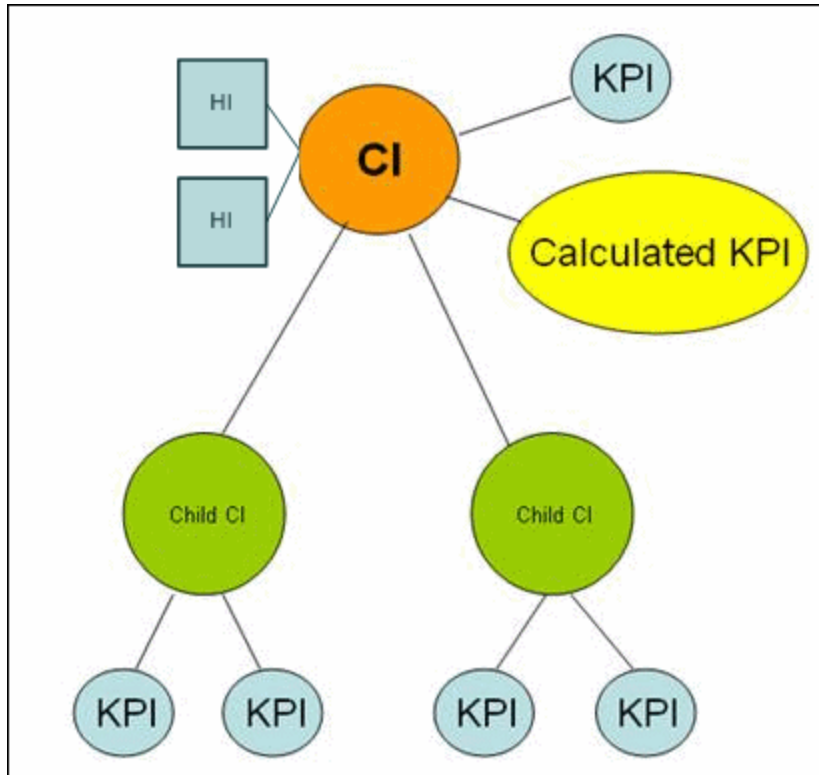
The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:

\\<Gateway Server root directory>
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html.

For examples of Group and Sibling rules, see "Examples - API Group and Sibling Rule" on page 108.

API rules can be defined within the KPI Definition page or Rule Repository, or using a text file template, as described in "Creating Rules with the Rules API" on page 97.

Defining a Group and Sibling Rule in the KPI Definition Page or Rule Repository

To define a Group and Sibling rule using the KPI Definition page or within the Rule Repository, enter the **calculateKPI** method implementation in the **KPI Calculation Script** area.

The parameters **ci** and **kpi** of the **calculateKPI** method are available for use in this script.

For detailed instructions, see "How to Define an API Rule for a Specific KPI or Outage" on page 98 or "How to Define an API Rule Within the Rule Repository" on page 102.

Accessing a Specific Child KPI in the KPI Definition Page

When creating a Group rule for a specific KPI in the KPI Definition page, to access a specific child KPI, the API includes a mechanism to simplify the code. When defining your KPI Calculation Script, you can enter the format **"<CI name>"."<KPI name>"**.

For an example of this, see "Examples - API Group and Sibling Rule" on page 108.

Defining a Group and Sibling Rule Using a Text File

To define a Group and Sibling rule using a text file, use the
SlmGroupAndSiblingTemplate.groovy template as described in "How to Create a Text File-Based API Rule" on page 99.

Within the text file, enter the **calculateKPI** method body.

Sample Rule Calculation Mechanism - Overview

The API Sample rule and Duration-Based Sample rule calculate KPIs based on sample values, for each combination of calendar and tracking period. This calculation process in divided into calculation cycles.

Each calculation cycle, the following steps occur:

1. The samples that are in the Profile database for the calculation cycle time are retrieved. For example, if a KPI is calculated for 10:00-11:00 and the cycle duration is 5 minutes, the samples that are in the database with the timestamp 10:00-10:05 are used in the first cycle's calculation.

2. The samples go through a filtering mechanism which determines which samples are included in the calculation.

3. The KPI is calculated for the cycle based on the samples that passed through the filtering mechanism. For details, see "Sample Rules: Calculating the KPI Based on Samples" on the next page.

4. Aggregated KPI results are calculated based on the results for the cycle, and the previous aggregated results for the full calculation period. These aggregated KPI results are displayed in the Service Level Management reports. For details, see "Sample Rules: Calculating the KPI's Aggregated Results" on page 86.

Sample Rules: Calculating the KPI Based on Samples

KPI results are calculated for each calculation cycle by the **calculateKPI** method, based on the **samples** parameter. The **samples** parameter is a **List** of **Sample** objects, which hold sample field values.

The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
**\\<Gateway Server root directory>\
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html**.

The **calculateKPI** method can be used to set KPI value, keys, and tooltips.

The value, keys, and tooltips set by the **calculateKPI** method are available to the **calculateAggregatedKPI** method, as described in "Sample Rules: Calculating the KPI's Aggregated Results" on the next page.

Setting KPI Value

KPI value should be used when calculating KPI value directly from the sample fields. For details on setting the value, refer to the **setValue** method documentation.

For an example which uses KPI value, see "Example - Sample-Based Maximum Response Time Rule" on page 113.

Setting KPI Calculation Keys

Each KPI can hold keys which are used as helpers to calculate value or tooltips. For example, when calculating average response time (total response time / total number of samples), the **setKey** method is used to set two keys for the current cycle: total response time, and total number of samples. These keys are aggregated and used to calculate the aggregated value of the KPI.

For an example which uses KPI calculation keys, see "Example - Sample-Based Average Response Time Rule" on page 109.

Setting KPI Tooltips

KPI tooltips should be used when calculating KPI tooltips directly from the sample fields. For details on setting the tooltips, refer to the **setTooltip** method documentation.

Sample Rules: Calculating the KPI's Aggregated Results

Using the values, tooltips, and keys set by the **calculateKPI** method, the **calculateAggregatedKPI** method calculates aggregated values, keys, and tooltips.

The aggregated results are based on the KPI calculation results for the cycle, and the aggregated calculation results from the previous cycles.

The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
**\\<Gateway Server root directory>\
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html**.

The **calculateAggregatedKPI** method can be used to set KPI aggregated value, keys, and tooltips.

Setting KPI Aggregated Value

The KPI aggregated value can be calculated based on KPI value (for example, minimum calculation), or based on aggregated keys (for example, average response time). The KPI aggregated value is displayed in the Service Level Management report. For details on setting the aggregated value, refer to the **setAggregatedValue** method documentation.

For an example which calculates KPI aggregated value based on KPI value, see "Example - Sample-Based Maximum Response Time Rule" on page 113.

For an example which calculates KPI aggregated value based on KPI aggregated keys, see "Example - Sample-Based Average Response Time Rule" on page 109.

Setting KPI Aggregated Calculation Keys

The aggregated keys are used as helpers to calculate aggregated value or aggregated tooltips. The aggregated keys are calculated based on aggregated keys and keys from the current cycle.

For example, when calculating average response time (total response time / total number of samples), the **setAggregatedKey** method is used to set two aggregated keys: total response time, and total number of samples. These value are then used to calculate the aggregated KPI value.

For details on setting the aggregated keys, refer to the **setAggregatedKey** method documentation.

For an example which uses KPI aggregated calculation keys, see "Example - Sample-Based Average Response Time Rule" on page 109.

Setting KPI Status

If your rule sets KPI status, use the **setStatus** method within the **calculateAggregatedKPI** method.

Setting KPI Aggregated Tooltips

The KPI aggregated tooltips can be calculated based on KPI tooltips, or aggregated keys. The KPI aggregated tooltips are displayed in the Service Level Management report.

For details on setting the aggregated tooltips, refer to the **setAggregatedTooltip** method documentation.

When to Use Sample or Duration-Based Sample Rules

Use a Sample rule to calculate KPIs based on sample values and number of samples (when needed), without using sample duration in the calculation.

Use a Duration-based Sample rule to calculate KPIs based on sample values and sample duration, without using the number of samples in the calculation.

Example of Average Response Time Calculation

Average response time can be calculated using a Sample rule or a Duration-Based Sample rule.

At 10:00 a sample is received with 100 seconds as response time, and at 10:55 a sample is received with 50 seconds as response time. The rule calculation is for 10:00-11:00.

The API Sample rule calculates average response time (total response time / number of samples) = 75 seconds.

The API Duration-Based Sample rule calculates weighted response time based on the value of the samples, and on their durations. The duration of the first sample is 55 minutes and the duration of the second sample is 5 minutes. The weighted average response time for the period is (55x100 + 5x50) / 60 = 95.83 seconds.

API Sample Rule

A Sample rule calculates KPIs based on aggregated data taken from sample fields. For details on when to use the Sample rule, see "When to Use Sample or Duration-Based Sample Rules" on the previous page.

For details on how the rule is calculated, see "Sample Rule Calculation Mechanism - Overview" on page 84.

Sample Rule Methods and Fields

The Sample rule implements the Rules API Interface **SlmSamplesAggregatedCalculator**, which contains the following methods:

```
public void calculateKPI(CI ci, SlmKPI kpi, List<Sample> samples)

public void calculateAggregatedKPI(CI ci, SlmKPI kpi)

public boolean isSampleValid(CI ci, SlmKPI kpi, Sample sample)
```

- The **calculateKPI** method calculates the KPI for each calculation cycle. This method includes the parameters **ci**, **kpi**, and **samples**. These represent the current CI, the KPI whose value the rule calculates, and the samples to be used in the rule calculation.

- The **kpi** parameter type is **SlmKPI**, and is used to set calculation results.

- The **samples** parameter is a **List** of **Sample** objects, which hold sample field values.

- The **calculateAggregatedKPI** method calculates the aggregated KPI. This method includes the parameters **ci** and **kpi**.

- The **isSampleValid** method is used to filter samples. If a sample is not valid, is not included in the calculation. Samples that are valid are included in the **samples** parameter of the **calculateKPI** method.

- The rule must also set the **sampleFields** field to define which sample fields are held by the **Sample** object. These values are the values used by the rule.

The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
**\\<Gateway Server root directory>\
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html**.

For detailed examples of Sample rules, see "Examples - API Sample Rule" on page 109.

API rules can be defined within the KPI Definition page or the Rule Repository, or using a text file template, as described in "Creating Rules with the Rules API" on page 97.

Defining a Sample Rule in the KPI Definition Page or Rule Repository

To define a Sample rule using the KPI Definition page or within the Rule Repository, fill in the fields as follows:

- **Sample Fields**. List the sample fields whose values can be included in the calculation; separate between the sample names with a comma (for example: `"u_iStatus"`,

`"dResponseTime"`).

- **KPI Calculation Script**. Enter the **calculateKPI** method implementation; do not enter the method signature. The parameters **ci**, **kpi,** and **samples** of the **calculateKPI** method are available for use in this script.

- **Aggregated Calculation Script**. Enter the **calculateAggregatedKPI** method implementation; do not enter the method signature. The parameters **ci** and **kpi** of the **calculateAggregatedKPI** method are available for use in this script.

- **Sample Filter Script**. This field contains the default implementation of the **isSampleValid** method (by default, all samples are included in the calculation). You can edit this field to exclude samples from the calculation.

For detailed instructions, see "How to Define an API Rule for a Specific KPI or Outage" on page 98 or "How to Define an API Rule Within the Rule Repository" on page 102.

Defining a Sample Rule Using a Text File

To define a Sample rule using a text file template, use the **SlmSampleRuleTemplate.groovy** template file as described in "How to Create a Text File-Based API Rule" on page 99.

Within the text file, define the **sampleFields** field, the **calculateKPI** method body, and the **calculateAggregatedKPI** method body.

The **isSampleValid** method has a default implementation of return true (all samples are included in the calculation). To override, uncomment the method and enter your implementation.

API Duration-Based Sample Rule

A Duration-Based Sample rule calculates KPIs based on aggregated data taken from sample fields. The duration-based rule uses each sample's duration within the rule calculation (for example, availability over time or backlog over time).

For details on when to use the Duration-Based Sample rule, see "When to Use Sample or Duration-Based Sample Rules" on page 88. For details on how the rule is calculated, see "Sample Rule Calculation Mechanism - Overview" on page 84.

Duration-Based Sample Rule Methods and Fields

The Duration-Based Sample rule implements the Rules API Interface **SlmSamplesTimeBasedAggregatedCalculator**, which contains the following methods:

```
public void calculateKPI(CI ci, SlmKPI kpi, List<Sample> samples)

public void calculateAggregatedKPI(CI ci, SlmKPI kpi)

public boolean isSampleValid(CI ci, SlmKPI kpi, Sample sample)

public boolean isSampleAndDurationValid(CI ci, SlmKPI kpi, Sample
sample)
```

- The **calculateKPI** method calculates the KPI for each calculation cycle. This method includes the parameters **ci**, **kpi**, and **samples**. These represent the current CI, the KPI whose value the rule calculates, and the samples to be used in the rule calculation.

  - The **kpi** parameter type is **SlmKPI**, and is used to set calculation results.

  - The **samples** parameter is a **List** of **Sample** objects, which hold sample field values and sample durations. The **samples** parameter contains the samples that passed through a filter mechanism, as described in "Filtering with the Duration-Based Sample Rule" on page 94. The last sample used in the previous calculation cycles can also by included, as described in "Duration-Based Sample Continuity" on page 93.

- A sample's duration is defined as the interval from the sample timestamp to one of the following (whichever event occurs first):

  - The timestamp of the next sample, if the next sample is filtered using the **isSampleAndDurationValid** method.

  - The timestamp of the next sample within the cycle.

  - The end of the cycle.

- The **calculateAggregatedKPI** method calculates the aggregated KPI. This method includes the parameters **ci** and **kpi**.

- The **isSampleValid** and **isSampleAndDurationValid** method are used for filtering, as described in "Filtering with the Duration-Based Sample Rule" on page 94.

The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
**\\<Gateway Server root directory>\
AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html**.

- The rule must also set the **sampleFields** field to define which sample fields are held by the **Sample** object. These values are the values used by the rule.

For detailed examples of Duration-Based Sample rules, see "Examples - API Duration-Based Sample Rule" on page 115.

API rules can be defined using the KPI Definition page or the Rule Repository, or using a text file template, as described in "Creating Rules with the Rules API" on page 97.

Defining a Duration-Based Sample Rule in the KPI Definition Page or Rule Repository

To define a Duration-Based Sample rule using the KPI Definition page or within the Rule Repository, fill in the fields as follows:

- **Sample Fields**. List the sample fields whose values can be included in the calculation; separate between the sample names with a comma (for example: `"u_iStatus"`, `"dResponseTime"`).

- **KPI Calculation Script**. Enter the **calculateKPI** method implementation; do not enter the method signature. The parameters **ci**, **kpi,** and **samples** of the **calculateKPI** method are available for use in this script.

- **Aggregated Calculation Script**. Enter the **calculateAggregatedKPI** method implementation; do not enter the method signature. The parameters **ci** and **kpi** of the **calculateAggregatedKPI** method are available for use in this script.

- **Sample Filter Script** and **Sample and Duration Filter Script**. These fields contain the default implementation of the **isSampleValid** and **isSampleAndDurationValid** methods (by default, all samples are included in the calculation). You can edit these fields to exclude samples from the calculation.

For detailed instructions, see "How to Define an API Rule for a Specific KPI or Outage" on page 98 or "How to Define an API Rule Within the Rule Repository" on page 102.

Defining a Duration-Based Sample Rule Using a Text File

To define a Duration-Based Sample rule using a text file template, use the **SlmDurationBasedSampleRuleTemplate.groovy** template file as described in "How to Create a Text File-Based API Rule" on page 99.

Within the text file, define the **sampleFields** field, the **calculateKPI** method body, and the **calculateAggregatedKPI** method body.

The **isSampleValid** and **isSampleAndDurationValid** methods have a default implementation of return true (all samples are included in the calculation). To override, uncomment the method and enter your implementation.

Duration-Based Sample Continuity

In the Duration-Based Sample rule, the first sample included in the **samples** parameter (for the **calculateKPI** method), is the last sample from the previous cycle's calculation. This enables a sample's value to be used over more than one calculation cycle.

For example, for the calculation cycle 10:05-10:10 there is one sample in the database (Sample1) with the timestamp 10:08. The **samples** parameter contains two samples: the last sample from the previous cycle (duration=3 minutes), and the sample from the current cycle (duration=2 minutes).
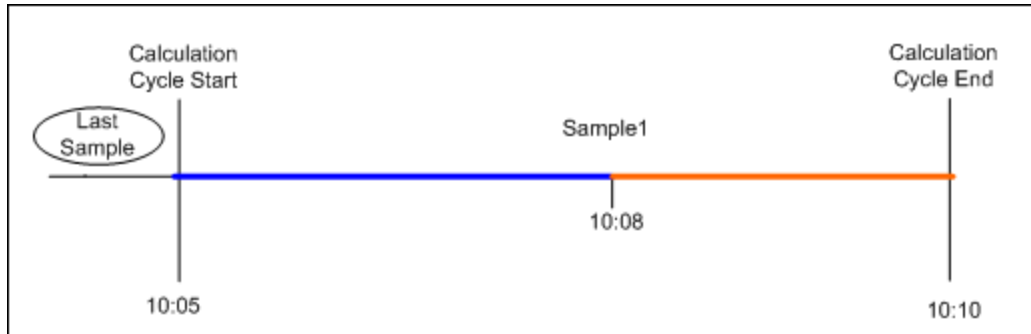
If there are no samples in the current cycle, the **samples** parameter contains the last sample from the previous cycle (duration=5 minutes).
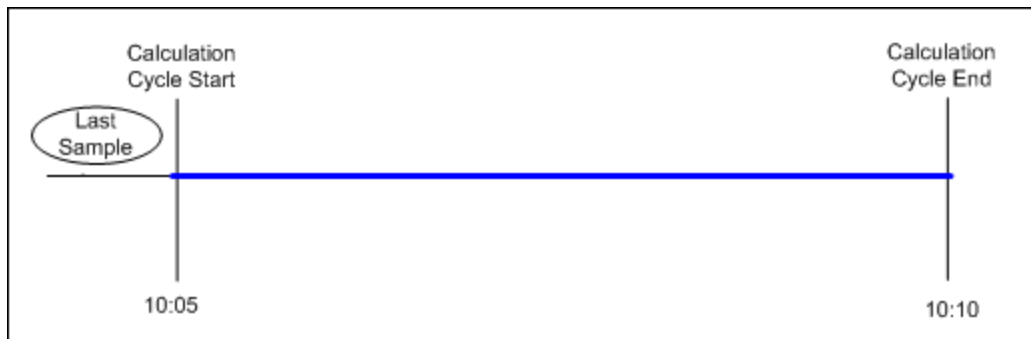
The last sample does not continue to the next calculation cycle if one of the following conditions are true:

- **No data timeout.** The last sample's timestamp has reached the no data timeout limit. For example, if the last sample's timestamp is 09:00 and the no data timeout is one hour, the sample is not included in the 10:00-10:05 calculation cycle, and all subsequent calculation cycles.

- **Downtime event.** A downtime event has occurred in the time between the last sample's timestamp and the current calculation cycle. For example, if the last sample's timestamp is 09:00 and a downtime is configured to 10:00-10:30, the sample is not part of the 10:30-10:35 calculation cycle, and all subsequent calculation cycles.

Filtering with the Duration-Based Sample Rule

By default, the API Duration-Based Sample rule includes all samples in rule calculations. The **isSampleValid** and **isSampleAndDurationValid** methods enable filtering of samples when using this rule.

- When using the **isSampleValid** method, the duration of a filtered sample is added to the duration of the previous valid sample.
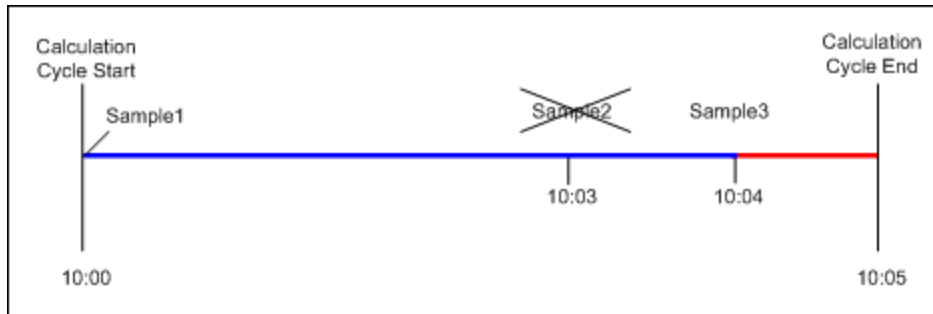
In the above example, Sample2 is filtered out using the **isSampleValid** method. The **samples** parameter contains Sample1 with 4 minutes duration and Sample3 with 1 minute duration. The total duration of all the samples in the cycle is 5 minutes.

- When using the **isSampleAndDurationValid** method, the duration of a filtered sample is not added to the duration of the previous valid sample, and it is therefore not included in the calculation.
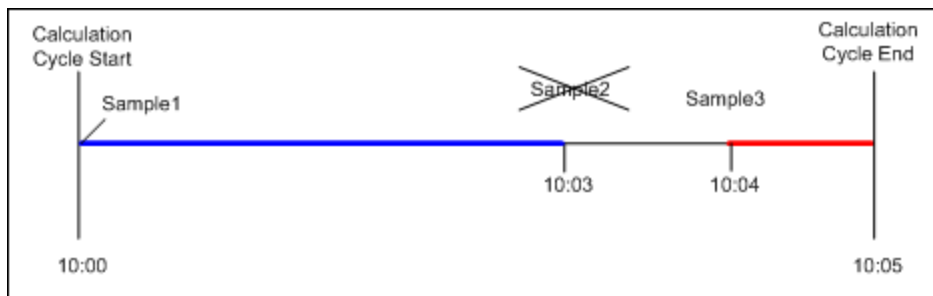
In the above example, Sample2 is filtered out using the **isSampleAndDurationValid** method. The **samples** parameter contains Sample1 with 3 minutes duration and Sample3 with 1 minute duration. The total duration of all the samples in the cycle is 4 minutes.

API Outage by Samples Rule

An API Outage by Samples rule calculates outages based on data received from samples. An outage begins when a specific number of consecutive failures is calculated based on consecutive samples, and ends when a sample representing no failures is used in calculation.

Rule parameters (minimum number of failures, minimum duration, and maximum duration) define when an outage begins and when it ends, as described in List of Service Level Management Business Rule Parameters in the BSM Application Administration Guide. The Outage by Samples rule parameters are demonstrated in "Examples - API Outage by Samples Rule" on page 122.

Each sample can represent a number of failures (zero or more). Using the API Outage by Samples rule, you can calculate how many failures a sample represents based on the sample values. The outage is then calculated using the outage rule parameters.

Outage by Samples Rule Methods and Fields

The Outage by Samples rule implements the Rules API Interface **OutageBySamplesCalculator**, which contains the following methods:

```
public void calculateOutage(Outage outage, Sample sample)

public boolean isSampleValid(Sample sample)
```

- The **calculateOutage** method calculates the number of failures represented by a **Sample**. This method includes the parameters **outage** and **sample**.

  - The **outage** parameter type is **Outage**, and is used to set the number of failures for the given **sample**.

  - The **sample** parameter type is **Sample**, which holds sample field values.

- The **isSampleValid** method is used to filter samples. If a sample is not valid, it is not included in the **calculateOutage** method calculation.

  The Rules API classes are documented in Javadoc format in the *HP Rules API Reference*. These files are located in the following folder:
  **\\<Gateway Server root directory>\
  AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_
  API\index.html**.

- The rule must also set the **sampleFields** field to define which sample fields are held by the **Sample** object. These values are the values used by the rule.

For examples of the Outage by Samples rule, see "Examples - API Outage by Samples Rule" on page 122.

API rules can be defined using the KPI Definition page or the Rule Repository, or using a text file template, as described in "Creating Rules with the Rules API" on page 97.

Defining an Outage by Samples Rule in the KPI Definition Page or Rule Repository

To define an Outage by Samples rule using the KPI Definition page or within the Rule Repository, fill in the fields as follows:

- **Sample Fields**. List the sample fields whose values can be included in the calculation; separate between the sample names with a comma (for example: `"u_iStatus"`).

- **Outage Calculation Script**. Enter the **calculateOutage** method implementation; do not enter the method signature. The parameters **outage** and **sample** of the **calculateOutage** method are available for use in this script.

- **Sample Filter Script**. This field contains the default implementation of the **isSampleValid** method (by default, all samples are included in the calculation). You can edit this field to exclude samples from the calculation.

For detailed instructions, see "How to Define an API Rule for a Specific KPI or Outage" on page 98 or "How to Define an API Rule Within the Rule Repository" on page 102.

Defining an Outage by Samples Rule Using a Text File

To define an Outage by Samples rule using a text file template, use the **SlmOutageBySampleTemplate.groovy** template file as described in "How to Create a Text File-Based API Rule" on page 99.

Within the text file, define the **sampleFields** field and the **calculateOutage** method body.

The **isSampleValid** method has a default implementation of return true (all samples are included in the calculation). To override, uncomment the method and enter your implementation.

Creating Rules with the Rules API

There are a number of ways to create rules using the Rules API, as described in the following section.

> **Note:** The following three ways are also applicable to defining Outage calculations, using the API Outage by Samples Rule.

Define a rule for a specific KPI

Service Level Management KPIs can have the following API rules: API Group and Sibling Rule, API Sample Rule, and API Duration-Based Sample Rule.

Using the **New SLA/Edit SLA Wizard > Configure SLA Indicators** page, you can assign one of the API rules to a KPI, and enter rule details to define rule logic for that KPI.

You can then edit the rule details in the **Configure SLA Indicators** page at any time to change the rule logic for the KPI.

For details, see

Create a rule for multiple KPIs using a text file

For each of the API rules there is a corresponding template file, located in the **<Data Processing server root directory>\BLE\rules\groovy\templates** directory. You can use one of the template files to create a text file defining a new rule. You then add this rule to the Rule Repository, and it can be applied like any out-of-the-box rule.

The API code cannot be seen or changed within Service Level Management, but only within the text file. If you make changes to the code within the text file, these changes are applied to all instances where the rule has been assigned, after you restart the offline engine.

For details, see

Defining a rule within the rule repository

The Rule Repository contains the following API rules: API Group and Sibling Rule, API Sample Rule, and API Duration-Based Sample Rule. You can use the Rule Repository to clone an API rule and enter rule details to define the rule logic.

After the rule is applied to a KPI, you can edit rule details within the **Configure SLA Indicators** page at any time to change the rule logic for a specific KPI.

For details, see

How to Define an API Rule for a Specific KPI or Outage

Each KPI or Outage has applicable API rules. Using the KPI Definition page, assign one of the API rules to a KPI or Outage, and enter the rule details to define the rule logic.

1. ## Assign an API rule to a KPI or outage

   - To assign an API rule for a specific KPI assigned to a CI, edit an SLA using the **Agreements Manager >New SLA/Edit SLA Wizard > Configure SLA Indicators** page. Select **Add KPI** to assign a new KPI to the CI, or modify an existing KPI. For details, see Add KPI for CI/Edit KPI for CI Dialog Box in the BSM Application Administration Guide.

     From the list of applicable business rules, select one of the API rules: API Group and Sibling Rule, API Sample Rule, or API Duration-Based Sample Rule. (API Sample Rule and API Duration-Based Sample Rule are only applicable for monitor CIs.) For a description of the rule types see "SLM Rules API" on page 78.

   - To assign an API Outage rule to a CI's Outage, edit an SLA using the **Agreements Manager > New SLA/Edit SLA Wizard > Configure SLA Indicators** page. For details, see Add Outages KPI/Edit Outages KPI Dialog Box in the BSM Application Administration Guide.

     Click the Edit button to edit the Outage. From the list of applicable business rules, select API Outage by Samples Rule. (This rule is only applicable for monitored CIs.)

2. ## Define the KPI or outage rule logic

   Depending on the type of rule you are creating, define the rule methods and fields as described in:

   - "API Group and Sibling Rule" on page 81

   - "API Sample Rule" on page 89

   - "API Duration-Based Sample Rule" on page 91

   - "API Outage by Samples Rule" on page 95

How to Create a Text File-Based API Rule

There are rule template files corresponding to the API rules; each template implements the rule's interface.

Create a text file defining a new rule using one of the templates, and then add the new rule to the Business Rule Repository. The rule can then be applied like any out-of-the-box rule.

The API code cannot be seen or changed within Service Level Management, but only within the text file. If you make changes to the code within the text file, these changes are applied to all instances where the rule has been assigned, after you restart the offline engine.

1. ## Create a text file for a rule

   Based on the type of rule you want to create, copy and rename one of the template files located in the **<Data Processing server root directory>\BLE\rules\groovy\templates** directory.

   Within your copy of the template, define the rule methods and fields as described in:

   - "API Group and Sibling Rule" on page 81
   - "API Sample Rule" on page 89
   - "API Duration-Based Sample Rule" on page 91
   - "API Outage by Samples Rule" on page 95

   Save the file to the **<Data Processing server root directory>\BLE\rules\groovy\rules** directory.

   You must now add a rule in the Rule Repository that uses the rule logic in the text file.

   - For rules other than Outage, follow the instructions in "Add a rule in the rule repository (for Group and Sibling, Sample, or Duration-Based Sample rules)" below.
   - For Outage rules, follow the instructions in "Add an Outage rule in the rule repository" on the next page.

2. ## Add a rule in the rule repository (for Group and Sibling, Sample, or Duration-Based Sample rules)

   a. Select **Admin > Service Level Management > Repositories > Business Rules > New Rule**. For details on adding rules, see How to Customize a Business Rule Template in the Repository in the BSM Application Administration Guide.

   b. In the **Name** field, type the name of the rule you want to create (mandatory).

   c. In the **Class Name** field, type **groovy:** <file name>. Note that the file name must be identical (case sensitive) to the file name in the **<Data Processing server root directory>\BLE\rules\groovy\rules** directory.

   d. For API Duration-Based Sample rules, create the following rule parameter:

      ○ In the **Rule parameters** area, click **New**.

      ○ In the **Name** field type **No Data Timeout**. In the **Type** field, select **Long**. In the **Default Value** field, type **3600**.

e. Click **OK** to save.

The following image shows a Duration-Based Sample rule after the rule parameter has been added:

3. Add an Outage rule in the rule repository

a. Within **Admin> Service Level Management > Repositories > Business Rules**, select the **Synthetic WS Outage** rule (316) and clone the rule.

b. In the **Name** field, change the name of the rule (mandatory).

c. In the **Class Name** field, type **groovy:** <file name>. Note that the file name must be identical (case sensitive) to the file name in the **<Data Processing server root directory>\BLE\rules\groovy\rules** directory.

d. You can edit the **Description**, but do not change any other fields.

e. Click OK to save.

4. Add the rule to the list of applicable rules for a KPI or Outage

Add the new rule to the list of applicable rules already attached to the relevant KPI. The relevant KPI for Outage rules is the **Outages** KPI (200).

For details, see the Main Settings Area > Applicable Rules parameter in New KPI/Edit KPI Dialog Box in the BSM Application Administration Guide.

5. Restart the offline engine after editing the text file

If you make changes to the text file at any time after the rule is created, perform the following steps to apply the changes.

a. In the browser, enter the following:

```
http://<Business Service Management Data Processing server
name>:11021
```

b. Select **Foundations:NannyManager**.

c. Invoke **showServiceInfoAsHTML**, and restart the offline engine.

How to Define an API Rule Within the Rule Repository

Within the Business Rule Repository, create an API rule that can be applied to multiple KPIs or Outages. This is done by cloning one of the four API rules, and setting default rule values for specific rule parameters. After the rule is applied to a KPI or Outage, you can edit its script within the KPI Definition page at any time to change the rule logic for the specific KPI or Outage.

1. Clone an API rule

Select **Admin > Service Level Management > Repositories > Business Rules**.In the Business Rule Repository page, clone one of the following rules: API Group and Sibling Rule, API Sample Rule, API Duration-Based Sample Rule, or API Outage by Samples Rule.

For details on cloning a rule, see How to Customize a Business Rule Template in the BSM Application Administration Guide.

2. Edit rule details

a. Open the new rule for editing.

b. In the **Name** field, rename the cloned rule.

c. Within the **Rule Parameters**, set the **Default value** for each rule parameter defining your rule logic, as described in the following sections:

- "API Group and Sibling Rule" on page 81
- "API Sample Rule" on page 89
- "API Duration-Based Sample Rule" on page 91
- "API Outage by Samples Rule" on page 95

For example, to define the **KPI Calculation Script**, edit the **KPI Calculation Script** rule parameter. In the **Default Value** field, enter the rule calculation script. The code that you enter becomes the default code for this rule, and appears in the KPI Definition page for all KPIs assigned this rule. (Do not change any other fields.)

3. Add the rule to the list of applicable rules for a KPI or Outage

Add the new rule to the list of applicable rules already attached to the relevant KPI. The relevant KPI for Outage rules is the **Outages** KPI (200).

For details, see the Applicable Rules GUI parameter in New KPI/Edit KPI Dialog Box in the BSM Application Administration Guide.

How to Work with Tooltip Entries

If you have used the **kpi.setTooltip** method, you must set a corresponding tooltip entry in the Infrastructure Settings.

1. Select **Admin > Platform > Setup and Maintenance > Infrastructure Settings > Service Level Management**.

2. Edit the **Additional Values** as follows:

   a. Scroll down to the KPI where you want to add the tooltip.

   b. After the last entry for the KPI, add a line in the following format:

   <Additional-Value calculate="true" id="<id>" name="<name>"/>

   In the above format, <id> is one number higher than the current highest ID (if the highest ID is 5, type 6), and <name> is the name of the tooltip exactly as used in the code.

   For example, if your code contains the method invocation **kpi.setTooltip("total_sales", value),** and the KPI is **Availability,** type the following:
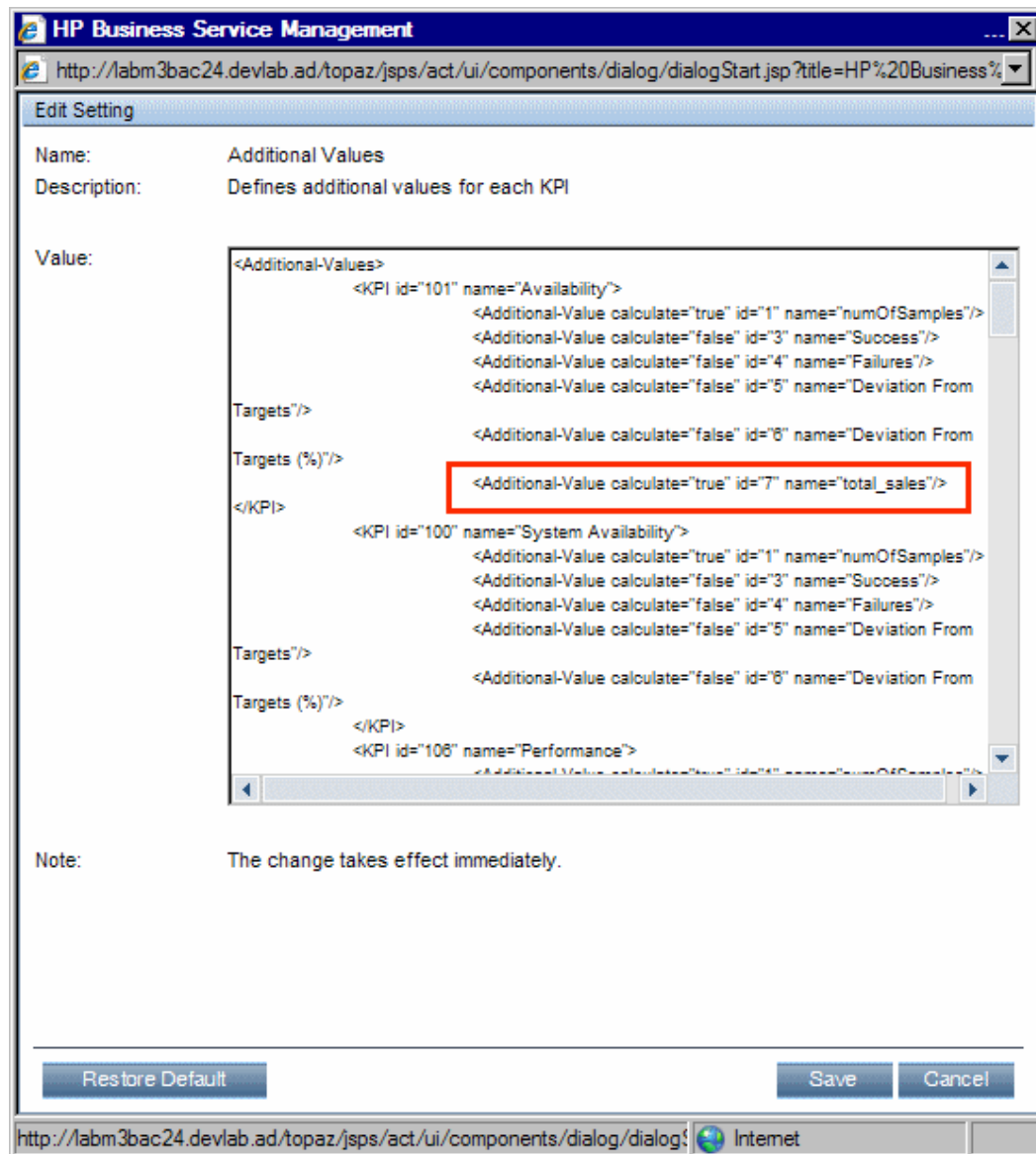
c. Click **Save**.

How to Write to Log Files From the Rules API Code

Within your API rules, you can write to log files from rule methods using a **logger** object. There are five log levels: debug, info, warn, error, and fatal. Each of these uses a specific logger method.

By default, only log method invocations of error and fatal severity are written to the log files. You can modify this within the log configuration files.

To write to log files using the Rules API:

1. Within the rule method, implement one of the following methods (listed in ascending order of severity):

   - **logger.debug("<API rule name> : log message");**

   - **logger.info("<API rule name> : log message");**

   - **logger.warn("<API rule name> : log message");**

   - **logger.error("<API rule name> : log message");**

   - **logger.fatal("<API rule name> : log message");**

   Type the name of your API rule inside the log message to identify each log message with its source rule.

2. The Rules API log files are found in the **<Data Processing server root directory>\HPBSM\log\offline_engine\RulesAPI** directory.

   Open one of the following files to view the log messages (depending on your rule type):

   - **groupAndSiblingRule.log** (for API Group and Sibling rules)

   - **sampleRule.log** (for API Sample and API Duration-Based Sample rules)

   - **OutageRule.log** (for API Outage by Samples rules)

To modify the severity level written to a log file:

1. By default, only log method invocations of error and fatal severity are written to log files. To modify this setting, open the log configuration file located in **<Data Processing server root directory>HPBSM\conf\core\Tools\log4j\offline_engine\slm_rules.properties**.

2. In the line corresponding with your rule type, replace the string **${loglevel}** with the severity level you want logged (type one of the following: **DEBUG**, **INFO**, **WARN**, **ERROR**, or **FATAL**). Edit one of the following lines, depending on your rule type:

   - Group and Sibling rules:
   log4j.category.com.mercury.am.bac.slm.rules.group.common.SlmGroupAndSiblingRule = **${loglevel}**, slm.rules.api.group.appender

   - Sample rules and Duration-Based Sample rules:
   log4j.category.com.mercury.am.bac.slm.rules.leaf.simplified.SlmSimplifiedLeafRule = **${loglevel}**, slm.rules.api.sample.appender

   - Outage by Samples rules:

log4j.category.com.mercury.am.bac.slm.rules.outages.simplified.SimplifiedOutageRule =
${loglevel}, slm.rules.api.outage.appender

How to Include a CI Property in Rules API Calculations

Within your API rules, you can include CI properties using the CI class **getPropertyValue** method, and the KPI class **getCiProperty** method. SLM rules can only access CI properties which have with the qualifier **BLE_ATTRIBUTE**.

To add this attribute to a CI class you must export the class, edit the class definition, and import it back to the server. When you open the exported class for editing, add the following xml to the required attribute:

```
<Attribute name="<attribute-name>" type="double" display-name="<attribute-display-name>">

  <Attribute-Qualifiers>

    <Attribute-Qualifier name="BLE_ATTRIBUTE"/>

  </Attribute-Qualifiers>

</Attribute>
```

Examples - API Group and Sibling Rule

Both Service Level Management and Service Health implement the same interface to calculate API Group and Sibling rules.

For detailed examples showing API Group and Sibling rules, see "Examples - API Group and Sibling Rule" on page 45.

Note that the statuses used in the **Status** class are different in Service Level Management and Service Health. For example, `Status.OK` in Service Health code is equivalent to `Status.EXCEEDED` in Service Level Management code. The following table shows the parallel statuses:

Service Health Status	Service Level Management Status
OK	EXCEEDED
WARNING	MET
MINOR	MINOR_BREACHED
MAJOR	BREACHED
CRITICAL	FAILED

Examples - API Sample Rule

This section provides examples of API Sample Rules. The following examples are described:

- "Example - Sample-Based Average Response Time Rule" below

- "Example - Sample-Based Average Response Time Rule with Filter" on page 112

- "Example - Sample-Based Maximum Response Time Rule" on page 113

Example - Sample-Based Average Response Time Rule

The following rule calculates average response time, based on the dResponseTime sample field. The rule result (aggregated value) is the average response time calculated based on all the samples that exist for the calculation period.

The rule logic is (total response time in seconds / total number of samples).

The rule uses the **SlmKPI** keys and aggregated keys to aggregate the total response time and the total number of samples, to calculate the rule result.

```
// Define the sample fields that will be used in the rule
calculation.
def sampleFields = ["dResponseTime"];

/*
* Implementation of the SlmSamplesAggregatedCalculator interface
method.
* For more information refer to the Rules API documentation.
*/
public void calculateKPI(CI ci, SlmKPI kpi, List<Sample> samples) {
/*
* Sum the field dResponseTime from all given samples. This
represents the total
* response time for all the samples in the current calculation
cycle.
*/
def totalTime =  Utils.sumField(samples, "dResponseTime");
/*
* Keep the total response time, converted to seconds, on a kpi key
named
* totalResponseTime. This key is used by the calculateAggregatedKPI
method.
    */
kpi.key.totalResponseTime = Utils.divide(totalTime,1000.0);
/*
* Keep the number of samples for the current calculation cycle on a
kpi key named
* totalSamples. This key is used by the calculateAggregatedKPI
method.
```

```
     */
kpi.key.totalSamples = samples.size();
}

/*
* Implementation of the SlmSamplesAggregatedCalculator interface
method.
*/
public void calculateAggregatedKPI(CI ci, SlmKPI kpi) {
/*
* Keep the aggregated response time on a kpi aggregated key named
* totalResponseTime, by adding the current aggregated
totalResponseTime from the
* kpi aggregated key, and the current calculation cycle response
time taken from the
* kpi key named totalResponseTime (calculated in the calculateKPI
method).
*/
kpi.aggregatedKey.totalResponseTime = Utils.sum
(kpi.key.totalResponseTime, kpi.aggregatedKey.totalResponseTime).
/*
* Keep the aggregated total samples on a kpi aggregated key named
totalSamples,
* by adding the current aggregated total samples from the kpi
aggregated key, and
* the current calculation cycle total samples, taken from the kpi
key named
* totalSamples (calculated in the calculateKPI method).
*/
kpi.aggregatedKey.totalSamples = Utils.sum(kpi.key.totalSamples,
kpi.aggregatedKey.totalSamples)
/*
* Set the aggregated value of the KPI by dividing the two
aggregated values.
* This aggregated value will be displayed in the SLA reports.
*/
kpi.aggregatedValue = Utils.divide
(kpi.aggregatedKey.totalResponseTime,
kpi.aggregatedKey.totalSamples)
}
```

Calculation - Sample-Based Average Response Time Rule

The following calculation illustrates the Sample-Based Average Response Time rule. Between 10:00 and 11:00, 9 samples have arrived:

10:00 {Sample Fields: dResponseTime = 60}
10:04 {Sample Fields: dResponseTime = 30}
10:11 {Sample Fields: dResponseTime = 30}
10:25 {Sample Fields: dResponseTime = 25}

10:27 {Sample Fields: dResponseTime = 35}
10:35 {Sample Fields: dResponseTime = 10}
10:36 {Sample Fields: dResponseTime = 20}
10:38 {Sample Fields: dResponseTime = 30}
10:52 {Sample Fields: dResponseTime = 75}

Each calculation cycle is 5 minutes long. The rule calculation is as follows:

Cycle	Keys set by calculateKPI		Aggregated keys and value set by calculateAggregatedKPI		
	totalResponseTime	totalSamples	totalResponseTime	totalSamples	Aggregated Value
10:00-10:05	90	2	90	2	45
10:05-10:10	Null	0	90	2	45
10:10-10:15	30	1	120	3	40
10:15-10:20	Null	0	120	3	40
10:20-10:25	Null	0	120	3	40
10:25-10:30	60	2	180	5	36
10:30-10:35	Null	0	180	5	36
10:35-10:40	60	3	240	8	30
10:40-10:45	Null	0	240	8	30
10:45-10:50	Null	0	240	8	30
10:50-10:55	75	1	315	9	35
10:55-11:00	null	0	315	9	35
					Final result: 35

Example - Sample-Based Average Response Time Rule with Filter

This rule is the same as the previous rule ("Example - Sample-Based Average Response Time Rule" on page 109), with the addition of a sample filter.

The code for the sample fields is as follows:

```
// This rule uses the dResponseTime sample field and u_iStatus
sample field.
def sampleFields = ["dResponseTime", "u_iStatus"];
```

This rule uses an additional method, as follows:

```
/*
* Override the default implementation of the
SlmSamplesAggregatedCalculator
* interface method.
*
* Filter samples that hold u_iStatus field value which is not 0 or
2.
*/
public boolean isSampleValid(Sample sample) {
        //Get the value of the sample's u_iStatus field.
        def avialFieldValueObj = sample.u_iStatus;
        return (avialFieldValueObj == 0 || avialFieldValueObj == 2)
}
```

Calculation - Sample-Based Average Response Time Rule with Filter

The following calculation illustrates the Sample-Based Average Response Time rule with filter.

For 10:00 - 11:00, 9 samples exist on the Profile Database:

10:00 {Sample Fields: dResponseTime = 60, u_iStatus = 0}
10:04 {Sample Fields: dResponseTime = 30, u_iStatus = 1}
10:11 {Sample Fields: dResponseTime = 30, u_iStatus = 2}
10:25 {Sample Fields: dResponseTime = 25, u_iStatus = 1}
10:27 {Sample Fields: dResponseTime = 35, u_iStatus = 0}
10:35 {Sample Fields: dResponseTime = 10, u_iStatus = 1}
10:36 {Sample Fields: dResponseTime = 20, u_iStatus = 0}
10:38 {Sample Fields: dResponseTime = 30, u_iStatus = 2}
10:52 {Sample Fields: dResponseTime = 75, u_iStatus = 1}

The samples in bold do not pass the filter. The following 5 samples are taken into calculation:

10:00 {Sample Fields: dResponseTime = 60, u_iStatus = 0}
10:11 {Sample Fields: dResponseTime = 30, u_iStatus = 2}

10:27 {Sample Fields: dResponseTime = 35, u_iStatus = 0}
10:36 {Sample Fields: dResponseTime = 20, u_iStatus = 0}
10:38 {Sample Fields: dResponseTime = 30, u_iStatus = 2}

The calculation result is = (60+30+35+20+30)/5 = 35

Example - Sample-Based Maximum Response Time Rule

The following rule calculates maximum response time, based on the dResponseTime sample field. The rule result (aggregated value) is the maximum response time calculated based on all the samples that exist for the calculation period. The rule uses the **SlmKPI** value to keep the maximum value for each calculation cycle.

```
// Define the sample fields that will be used in the rule
calculation.
def sampleFields = ["dResponseTime"];

/*
* Implementation of the SlmSamplesAggregatedCalculator interface
method. For more information refer to the Rules API documentation.
*/
public void calculateKPI(CI ci, SlmKPI kpi, List<Sample> samples) {
        /**
        * Find the maximum value of the dResponseTime field from
all given samples,
        * and set it as the KPI value for the current calculation
cycle.
        */
        kpi.value = Utils.getMax(samples, "dResponseTime");
}
    /**
    * Implementation of the SlmSamplesAggregatedCalculator
interface method.
    */
    public void calculateAggregatedKPI(CI ci, SlmKPI kpi) {
        /**
        * Keep the aggregated maximum response time on the KPI
aggregated value,
        * by replacing the current aggregated value with the
maximum between the KPI
        * aggregated value and the KPI value (calculated in the
calculateKPI method).
        * This aggregated value will be displayed in the SLA
reports.
        */
        kpi.aggregatedValue = Utils.max(kpi.value,
kpi.aggregatedValue)
}
```

Calculation - Sample-Based Maximum Response Time Rule

The following calculation illustrates the Sample-Based Maximum Response Time rule.

Between 10:00 and 11:00, 9 samples have arrived:

10:00 {Sample Fields: dResponseTime = 60}
10:04 {Sample Fields: dResponseTime = 30}
10:11 {Sample Fields: dResponseTime = 30}
10:25 {Sample Fields: dResponseTime = 25}
10:27 {Sample Fields: dResponseTime = 35}
10:35 {Sample Fields: dResponseTime = 10}
10:36 {Sample Fields: dResponseTime = 20}
10:38 {Sample Fields: dResponseTime = 30}
10:52 {Sample Fields: dResponseTime = 75}

The result of the rule calculation is 75.

Examples - API Duration-Based Sample Rule

This section provides examples of API Duration-Based Sample Rules. The following examples are described:

Example - Duration-Based Average Response Time Rule

The following rule calculates the weighted average response time based on the dResponseTime sample field and sample duration, for all the samples that exist for the calculation period.

The rule logic is sum (sample response time * sample duration) / sum (samples duration).

The rule uses **SlmKPI** keys and aggregated keys to aggregate the total weighted response time and the total duration of samples, in order to calculate rule results.

```
// Define the sample fields that will be used in the rule
calculation.
def sampleFields = ["dResponseTime"];

/*
* Implementation of the SlmSamplesTimeBasedAggregatedCalculator
interface method.
* For more information refer to the Rules API documentation.
*/
public void calculateKPI(CI ci, SlmKPI kpi, List<Sample> samples) {
    /**
    * Set the KPI key totalDuration to sum duration of all given
samples.
    * This key is used by the calculateAggregatedKPI method.
    */
    kpi.key.totalDuration = Utils.sumDuration(samples)
    // Iterate over all samples that arrived in the current
calculation cycle.
    samples.each {Sample sample ->
        /**
        * Calculate weighted response time for each sample by
multiplying
        * sample duration with sample dResponseTime field value.
        */
        def weightedResponseTime = Utils.multiply(sample.duration,
```

```
sample.dResponseTime);
        /**
        * Keep the total weighted response time for all given
samples
        * on a KPI key named totalResponseTime.
        * This key is used by the calculateAggregatedKPI method.
        */
        kpi.key.totalResponseTime = Utils.sum
(kpi.key.totalResponseTime, weightedResponseTime)
        }
}

/*
* Implementation of the SlmSamplesTimeBasedAggregatedCalculator
interface method.
*/
public void calculateAggregatedKPI(CI ci, SlmKPI kpi) {
    /**
     * Keep the aggregated response time on a kpi aggregated key
named
     * totalResponseTime, by adding the current aggregated
totalResponseTime, and the
     * current calculation cycle response time taken from the
totalResponseTime kpi key
     * (calculated by the calculateKPI method).
     */
    kpi.aggregatedKey.totalResponseTime = Utils.sum
(kpi.key.totalResponseTime, kpi.aggregatedKey.totalResponseTime)
    /**
     * Keep the aggregated total duration on a kpi aggregated key
named totalDuration,
     * by adding the current aggregated total duration from the kpi
aggregated key,
     * and the current calculation cycle total duration.
     */
    kpi.aggregatedKey.totalDuration = Utils.sum
(kpi.key.totalDuration, kpi.aggregatedKey.totalDuration)
    /**
     * Set the aggregated value of the KPI by dividing the two
aggregated values.
     * This aggregated value will be displayed in the SLA reports.
     */
    kpi.aggregatedValue = Utils.divide
(kpi.aggregatedKey.totalResponseTime,
kpi.aggregatedKey.totalDuration)
}
```

Calculation - Duration-Based Average Response Time Rule

The following calculation illustrates the Duration-Based Average Response Time rule.

For 10:00 - 11:00, 5 samples exist on the Profile Database:

10:00 Sample1 {Sample Fields: dResponseTime = 60}
10:04 Sample2 {Sample Fields: dResponseTime = 30}
10:25 Sample3 {Sample Fields: dResponseTime = 25}
10:38 Sample4 {Sample Fields: dResponseTime = 30}
10:52 Sample5 {Sample Fields: dResponseTime = 75}

Each calculation cycle is 5 minutes long. The rule calculation is as follows:

Cycle	samples parameter		Keys set by calculateKPI		Aggregated keys and value set by calculateAggregatedKPI		
	Sample	Sample Duration	totalResponseTime	totalDuration	totalResponseTime	totalDuration	Aggregated Value
10:00-10:05	Sample1	240	16200	300			
	Sample2	60			16200	300	54.000
10:05-10:10	Sample2	300	9000	300	25200	600	42.000
10:10-10:15	Sample2	300	9000	300	34200	900	38.000
10:15-10:20	Sample2	300	9000	300	43200	1200	36.000
10:20-10:25	Sample2	300	9000	300	52200	1500	34.800
10:25-10:30	Sample3	300	7500	300	59700	1800	33.167
10:30-10:35	Sample3	300	7500	300	67200	2100	32.000
10:35-10:40	Sample3	180	8100	300			
	Sample4	120			75300	2400	31.375
10:40-10:45	Sample4	300	9000	300	84300	2700	31.222
10:45-10:50	Sample4	300	9000	300	93300	3000	31.100
10:50-10:55	Sample4	120	17100	300			
	Sample5	180			110400	3300	33.455
10:55-11:00	Sample5	300	22500	300	132900	3600	36.917
						Result:	36.917

Example - Duration-Based Average Response Time Rule with isSampleValid Method Filter

This rule is the same as "Example - Duration-Based Average Response Time Rule" on page 115, with the addition of the **isSampleValid** method filter.

The code for the sample fields is as follows:

```
// This rule uses the dResponseTime sample field and u_iStatus
sample field.
def sampleFields = ["dResponseTime", "u_iStatus"];
```

This rule uses an additional method, as follows:

```
/*
* Override default implementation of the
SlmSamplesTimeBasedAggregatedCalculator
* interface method.
*
* Filter samples that hold u_iStatus field with value of 6.
```

```
*/
public boolean isSampleValid(Sample sample) {
        //Get the value of the sample's u_iStatus field.
        def avialFieldValueObj = sample.u_iStatus;
        return (avialFieldValueObj != 6)
}
```

Calculation - Duration-Based Average Response Time Rule with isSampleValid Method Filter

The following calculation illustrates the Duration-Based Average Response Time rule with **isSampleValid** method filter.

For 10:00 - 11:00, 5 samples exist on the Profile Database:

10:00 Sample1 {Sample Fields: dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {Sample Fields: dResponseTime = 30, u_iStatus = 2}
10:25 Sample3 {Sample Fields: dResponseTime = 25, u_iStatus = 6}
10:38 Sample4 {Sample Fields: dResponseTime = 30, u_iStatus = 0}
10:52 Sample5 {Sample Fields: dResponseTime = 75, u_iStatus = 2}

Sample3 did not pass the **isSampleValid** method filter, so the following 4 samples are taken into calculation:

10:00 Sample1 {Sample Fields: dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {Sample Fields: dResponseTime = 30, u_iStatus = 2}
10:38 Sample4 {Sample Fields: dResponseTime = 30, u_iStatus = 0}
10:52 Sample5 {Sample Fields: dResponseTime = 75, u_iStatus = 2}

Note that after the filtering, the interval between Sample3 and Sample4 is considered part of the duration of Sample2.

The rule calculation is as follows:

| Cycle | samples parameter | | Keys set by calculateKPI | | Aggregated keys and value set by calculateAggregatedKPI | | |
	Sample	Sample Duration	totalResponseTime	totalDuration	totalResponseTime	totalDuration	Aggregated Value
10:00-10:05	Sample1	240	16200	300			
	Sample2	60			16200	300	54.000
10:05-10:10	Sample2	300	9000	300	25200	600	42.000
10:10-10:15	Sample2	300	9000	300	34200	900	38.000
10:15-10:20	Sample2	300	9000	300	43200	1200	36.000
10:20-10:25	Sample2	300	9000	300	52200	1500	34.800
10:25-10:30	Sample2	300	9000	300	61200	1800	34
10:30-10:35	Sample2	300	9000	300	70200	2100	33.428
10:35-10:40	Sample2	180	9000	300			
	Sample4	120			79200	2400	33
10:40-10:45	Sample4	300	9000	300	88200	2700	32.666
10:45-10:50	Sample4	300	9000	300	97200	3000	32.4
10:50-10:55	Sample4	120	17100	300			
	Sample5	180			114300	3300	34.636
10:55-11:00	Sample5	300	22500	300	136800	3600	38
							Result: 38

Example - Duration-Based Average Response Time Rule with isSampleAndDurationValid Method Filter

This rule is the same as "Example - Duration-Based Average Response Time Rule" on page 115, but with the **isSampleAndDurationValid** method filter.

The code for the sample fields is as follows:

```
// This rule uses the dResponseTime sample field and u_iStatus
sample field.
def sampleFields = ["dResponseTime", "u_iStatus"];
```

This rule uses an additional method, as follows:

```
/*
 * Override default implementation of the
SlmSamplesTimeBasedAggregatedCalculator
 * interface method.
 *
 * Filter samples that hold u_iStatus field value which is not 0 or
2.
 */
public boolean isSampleAndDurationValid(CI ci, SlmKPI kpi, Sample
sample) {
        //Get the value of the sample's u_iStatus field.
        def avialFieldValueObj = sample.u_iStatus;
        return (avialFieldValueObj == 0 || avialFieldValueObj == 2)
}
```

Calculation - Duration-Based Average Response Time Rule with isSampleAndDurationValid Method Filter

The following calculation illustrates the Duration-Based Average Response Time rule with **isSampleAndDurationValid** method filter.

For 10:00 - 11:00, 5 samples exist on the Profile Database:

10:00 Sample1 {Sample Fields: dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {Sample Fields: dResponseTime = 30, u_iStatus = 2}
10:25 Sample3 {Sample Fields: dResponseTime = 25, u_iStatus = 2}
10:38 Sample4 {Sample Fields: dResponseTime = 30, u_iStatus = 1}
10:52 Sample5 {Sample Fields: dResponseTime = 75, u_iStatus = 2}

Sample4 did not pass the **isSampleAndDurationValid** method filter, so the following 4 samples are taken into calculation:

10:00 Sample1 {Sample Fields: dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {Sample Fields: dResponseTime = 30, u_iStatus = 2}

10:25 Sample3 {Sample Fields: dResponseTime = 25, u_iStatus = 2}
10:52 Sample5 {Sample Fields: dResponseTime = 75, u_iStatus = 2}

Note that after the filtering, the interval between Sample4 and Sample5 is *not* considered part of the duration of Sample3, so the total duration for the hour is 46 minutes.

The rule calculation is as follows:

	samples parameter		Keys set by calculateKPI		Aggregated keys and value set by calculateAggregatedKPI		
Cycle	Sample	Sample Duration	totalResponseTime	totalDuration	totalResponseTime	totalDuration	Aggregated Value
10:00-10:05	Sample1	240	16200	300			
	Sample2	60			16200	300	54.000
10:05-10:10	Sample2	300	9000	300	25200	600	42.000
10:10-10:15	Sample2	300	9000	300	34200	900	38.000
10:15-10:20	Sample2	300	9000	300	43200	1200	36.000
10:20-10:25	Sample2	300	9000	300	52200	1500	34.800
10:25-10:30	Sample3	300	7500	300	59700	1800	33.167
10:30-10:35	Sample3	300	7500	300	67200	2100	32.000
10:35-10:40	Sample3	180	4500	180	71700	2280	31.477
10:40-10:45	empty			0	71700	2280	31.477
10:45-10:50	empty			0	71700	2280	31.477
10:50-10:55	Sample5	180	13500	180	85200	2460	34.634
10:55-11:00	Sample5	300	22500	300	107700	2760	36.917
							Result: 39.021

Example - Duration-Based Average Response Time Rule with isSampleAndDurationValid and isSampleValid Method Filters

This rule uses both the **isSampleAndDurationValid** and the **isSampleValid** method filters. These are described in the following sections:

- "Example - Duration-Based Average Response Time Rule with isSampleValid Method Filter" on page 117

- "Example - Duration-Based Average Response Time Rule with isSampleAndDurationValid Method Filter" on the previous page

For 10:00 - 11:00, 5 samples exist on the Profile Database:

10:00 Sample1 {Sample Fields: dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {Sample Fields: dResponseTime = 30, u_iStatus = 2}
10:25 Sample3 {Sample Fields: dResponseTime = 25, u_iStatus = 6}
10:38 Sample4 {Sample Fields: dResponseTime = 30, u_iStatus = 1}
10:52 Sample5 {Sample Fields: dResponseTime = 75, u_iStatus = 2}

Sample3 did not pass the **isSampleValid** method filter, and Sample4 did not pass the **isSampleAndDurationValid** method filter. The following 3 samples are taken into calculation:

10:00 Sample1 {Sample Fields: dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {Sample Fields: dResponseTime = 30, u_iStatus = 2}
10:52 Sample5 {Sample Fields: dResponseTime = 75, u_iStatus = 2}

After the filtering, the interval between Sample3 and Sample4 is considered part of the duration of Sample2, but the interval between Sample4 and Sample5 is not considered part of the duration of Sample4 (filtered by **isSampleAndDurationValid**).

The rule calculation is as follows:

Cycle	samples parameter		Keys set by calculateKPI		Aggregated keys and value set by calculateAggregatedKPI		
	Sample	Sample Duration	totalResponseTime	totalDuration	totalResponseTime	totalDuration	Aggregated Value
10:00-10:05	Sample1	240	16200	300			
	Sample2	60			16200	300	54
10:05-10:10	Sample2	300	9000	300	25200	600	42
10:10-10:15	Sample2	300	9000	300	34200	900	38
10:15-10:20	Sample2	300	9000	300	43200	1200	36
10:20-10:25	Sample2	300	9000	300	52200	1500	34.8
10:25-10:30	Sample2	300	9000	300	61200	1800	34
10:30-10:35	Sample2	300	9000	300	70200	2100	33.429
10:35-10:40	Sample2	180	5400	180	75600	2280	33.157
10:40-10:45					75600	2280	33.157
10:45-10:50					75600	2280	33.157
10:50-10:55	Sample5	180	13500	180	89100	2460	36.219
10:55-11:00	Sample5	300	22500	300	111600	2760	40.434
							Result: 40.434

Examples - API Outage by Samples Rule

This section provides examples of Outage by Samples Rules. The following examples are described:

- "Example - Outage by Samples Rule and Calculation with Default Rule Parameters" below

- "Example - Outage by Sample Calculation with Minimum Duration of 900 Seconds" on the next page

- "Example - Outage by Sample Calculation with Maximum Duration of One Hour" on page 124

- "Example - Outage by Sample Calculation with a Sample Representing Two Failures" on page 125

Example - Outage by Samples Rule and Calculation with Default Rule Parameters

The following section illustrates the Outage by Samples rule, based on the default Outage rule parameters:

Minimum number of failures: 2
Minimum duration: 0
Max duration: undefined

A sample represents one failure if the sample's u_iStatus field value is not 0 or 2. The rule also filters out samples whose u_iStatus field value is 6.

```
// Define the sample fields that will be used in the rule
calculation.
def sampleFields = ["u_iStatus"];

/*
* Implementation of the OutageBySamplesCalculator interface method.
* If the sample's u_iStatus field value is not 0 or 2, the sample
represents 1 failure.
* In any other case the sample represents no failures.
*
* For more information refer to the Rules API documentation.
*/
public void calculateOutage(Outage outage, Sample sample) {
    // Take the sample field's u_iStatus value.
    def statusFieldValue = sample.u_iStatus;
    if(statusFieldValue != 0 && statusFieldValue != 2){
        outage.setNumberOfFailures 1;
    }
}

/*
* Override default implementation of the OutageBySamplesCalculator
interface method.
```

```
*
* If the sample's u_iStatus field value is not 0 or 2, the sample
represents 1 failure.
*/
public boolean isSampleValid(Sample sample) {
    def statusFieldValue = sample.u_iStatus;
    return (statusFieldValue != 6)
}
```

The following calculation illustrates the above Outage by Samples rule.

For 10:00 - 11:00, 6 samples exist on the Profile Database:

10:10 Sample1 {Sample Fields: u_iStatus = 1}
10:20 Sample2 {Sample Fields: u_iStatus = 1}
10:25 Sample3 {Sample Fields: u_iStatus = 6}
10:30 Sample4 {Sample Fields: u_iStatus = 0}
10:35 Sample5 {Sample Fields: u_iStatus = 6}
10:40 Sample6 {Sample Fields: u_iStatus = 2}

Sample3 and Sample5 did not pass the **isSampleValid** method. The following samples are passed to the **calculateOutage** method:

10:10 Sample1 {Sample Fields: u_iStatus = 1}
10:20 Sample2 {Sample Fields: u_iStatus = 1}
10:30 Sample4 {Sample Fields: u_iStatus = 0}
10:40 Sample6 {Sample Fields: u_iStatus = 2}

For Sample1 and Sample2, the following line is invoked inside the **calculateOutage** method:
outage.setNumberOfFailures 1;

The result of the calculation is the following:
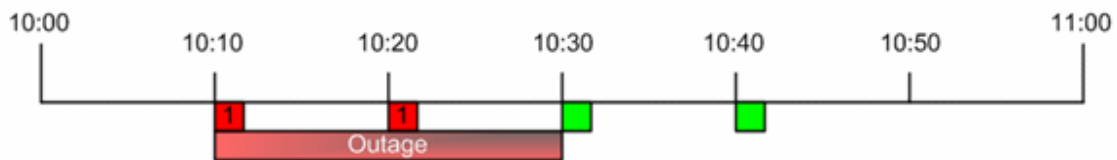
An outage is reported with a duration of 20 minutes, from 10:10 - 10:30.

Example - Outage by Sample Calculation with Minimum Duration of 900 Seconds

The following calculation illustrates the Outage by Samples rule, based on the following Outage rule parameters:

Minimum number of failures: 2
Minimum duration: 900 (seconds)
Max duration: undefined

For 10:00 - 11:00, 6 samples exist on the Profile Database:

- 10:10 - Sample representing 1 failure.

- 10:15 - Sample representing 1 failure.

- 10:20 - Sample representing no failures.

- 10:30 - Sample representing 1 failure.

- 10:40 - Sample representing 1 failure.

- 10:50 - Sample representing no failures.

The result of the calculation is the following:
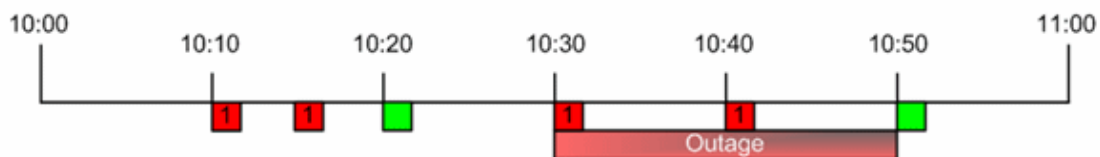
An outage is reported with a duration of 20 minutes, from 10:30 - 10:50. There is no outage between 10:10 - 10:20 because the outage duration did not reach the minimum outage duration parameter.

Example - Outage by Sample Calculation with Maximum Duration of One Hour

The following calculation illustrates the Outage by Samples rule, based on the following Outage rule parameters:

Minimum number of failures: 2
Minimum duration: 0 (default)
Max duration: 1 (hour)

For 10:00 - 12:00, 4 samples exist on the Profile Database:

- 10:00 - Sample representing 1 failure.

- 10:30 - Sample representing 1 failure.

- 11:00 - Sample representing 1 failure.

- 11:30 - Sample representing 1 failure.

The result of the calculation is the following:
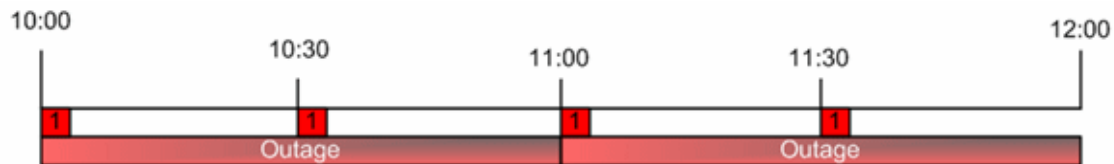
Two outages are reported with a duration of 1 hour; the first outage from 10:00 - 11:00, and the second outage from 11:00 - 12:00.

Example - Outage by Sample Calculation with a Sample Representing Two Failures

The following calculation illustrates the Outage by Samples rule, based on the following Outage rule parameters:

Minimum number of failures: 2
Minimum duration: 0 (default)
Max duration: undefined

For 10:00 - 11:00, two samples exist on the Profile Database:

- 10:00 - Sample representing two failures.

The following line of code is invoked in the **outage** parameter in **calculateOutage** method:
`outage.setNumberOfFailures 2;`

- 10:30 - Sample representing no failures.

The result of the calculation is the following:
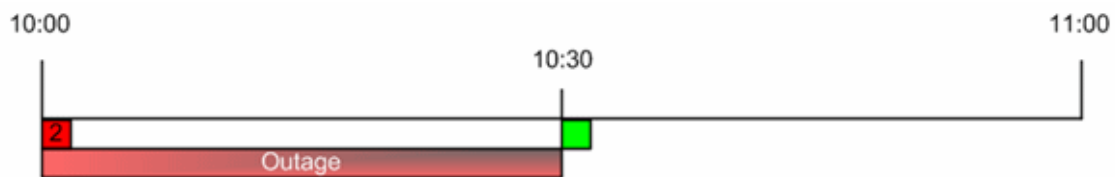
An outage is reported with a duration of 30 minutes, from 10:00 - 10:30.

Chapter 5

SLM Web Services API

The SLM Web Services API is an integration tool enabling administration of SLAs from an application either internal or external to BSM. The SLM Web services support the creation and management of SLAs, also in an HP Software-as-a-Service (HP SaaS) deployment.

For details, see the following:

- "Using the SLM Web Services" on the next page
- "SLM Web Services' Operations " on page 128

Prerequisite Knowledge

Users of the API should be familiar with Service Level Management administration and SOAP concepts.

Permissions

The administrator provides login credentials for connecting with the Web services. The credentials must be those of a user with Administrator permissions, or the SLA owner.

For details on setting permissions in the Permissions Manager, see Permissions Overview in the BSM Platform Administration Guide.

Supported Operations

The following Web service operations are supported:

- Create SLA (with no CI)
- Get SLA general properties
- Update SLA general properties
- Delete SLA
- Add Service to SLA
- Delete Service from SLA
- Get Services that are included in the SLA

Using the SLM Web Services

The SLM Web Services API enables submitting a service request. The engine returns an error description if it cannot parse the statement or does not run successfully. If there is no error, the results of the request are returned.

The Web services are described in a SOAP WSDL file, located at:

```
http://<server>:8080/slm_ws/services/SlmServices?wsdl
```

The port specification is only necessary for non-standard installations. Consult your system administrator for the correct port number.

Developers can use a development environment to generate code from WSDL for calling the Web services. The WSDL describes the interface operations and operation parameters.

SLM Web Services' Operations

This section describes the operations for the SLM Web services.

- "createSLA" below
- "updateSLA" below
- "deleteSLA" on the next page
- "getSLAProperties " on the next page
- "addServicesForSLAWithOfferings" on the next page
- "deleteServiceFromSLA" on page 130
- "getSLAServicesFullPath" on page 130
- "getServiceSLAs" on page 131

createSLA

Creates a new SLA with the specified SLA properties.

Operation signature:
```
String createSLA(String customerId, SlaPropertiesDTO properties)
```

Operation arguments:

Argument	Description
customerId	Typically 1 (apart from in an HP SaaS environment).
properties	The properties used for initializing the new SLA, as follows: name,description,agreementDetails,type(OLA, SLA,UC), classification (internal, external); startDate,endDate,timeZoneId,customerId, providerId,trackingPeriods

Return-value: Returns the ID of the SLA in the RTSM.

Exception: Throws `SlmWebServiceException` if the user cannot create another SLA; for example, because the user does not have the necessary permissions to create the SLA, or if the number of allowed SLAs has been reached.

updateSLA

Updates the SLA with the new properties.

Operation signature:
```
updateSLA(String customerId, String SLAId, SlaPropertiesDTO properties)
```

Operation arguments:

Argument	Description
customerId	Typically 1 (apart from in an HP SaaS environment).
SLAId	The RTSM ID of the SLA to update.
properties	The properties to be updated.

Exceptions:

- Throws `SlaDoesNotExistsException` if an SLA with the given ID does not exist.

- Throws `SlmWebServiceException` if a system problem occurs that prevents the operation to execute successfully.

deleteSLA

Deletes the specified SLA.

Operation signature:
```
deleteSLA(String customerId, String SLAId)
```

Operation arguments:

Argument	Description
customerId	Typically 1 (apart from in an HP SaaS environment).
SLAId	The RTSM ID of the SLA you want to delete.

Exception: Throws `SlmWebServiceException` when any type of error occurs.

getSLAProperties

Retrieves the properties of the specified SLA.

Operation signature:
```
SlaPropertiesDTO getSLAProperties(String customerId, String SLAId)
```

Operation arguments:

Argument	Description
customerId	Typically 1 (apart from in an HP SaaS environment).
SLAId	The RTSM ID of the SLA with the properties you want to retrieve.

Exception: Throws `SlmWebServiceException` when any type of error occurs.

addServicesForSLAWithOfferings

Adds the specified services and their impact sub-tree to the SLA. The matching service offering is used for each service.

Operation signature:
```
String[] addServicesForSLAWithOfferings(String customerId, String
slaId, ServiceWithOffering[] servicesWithOffering)
```

Operation arguments:

Argument	Description
customerId	Typically 1 (apart from in an HP SaaS environment).
SLAId	The RTSM ID of the SLA to which you want to add services.
servicesWithOffering	The services to add to the SLA with matching service offering names. Each `ServiceWithOffering` includes: serviceId, serviceName, offeringName **Note:** • offeringName must be provided. • You must provide at least one of serviceId or serviceName. If `serviceId` is missing, the server will try to obtain it using the `serviceName`.

Return-value: Returns an array of validation error messages (empty array if none).

Exception: Throws `SlmWebServiceException` if a system problem occurs that prevents the operation from executing successfully.

deleteServiceFromSLA

Removes the specified services and their paths from the specified SLA.

Operation signature:
```
deleteServiceFromSLA(String customerId, String SLAId, Service[]
services)
```

Operation arguments:

Argument	Description
customerId	Typically 1 (apart from in an HP SaaS environment).
SLAId	The RTSM ID of the SLA from which you want to remove the services.
services	The services you want to remove from the SLA. Each service includes: `serviceId, serviceName` If `serviceId` is missing, the server will try to obtain it using the serviceName.

Exceptions: `Throws SlmWebServiceException` when any type of error occurs.

getSLAServicesFullPath

Retrieves the services of the specified SLA.

Operation signature:
```
ServiceFullPath[] getSLAServicesFullPath(String customerId, String
SLAId)
```

Example: Example of SLA with services 4, 5:

```
{id1, id2, id3, id4}
{id1, id2, id5}
```

Operation arguments:

Argument	Description
customerId	Typically 1 (apart from in an HP SaaS environment).
SLAId	The RTSM ID of the SLA

Return-value: Returns the full paths of the services in the SLA.

Exceptions: Throws `SlmWebServiceException` when any type of error occurs.

getServiceSLAs

Returns the SLAs that include the specified service.

Operation signature:
```
String[] getServiceSLAs(String customerId, String serviceId)
```

Operation arguments:

Argument	Description
customerId	Typically 1 (apart from in an HP SaaS environment).
serviceId	The RTSM ID of the service.

Return-value: Returns an array of SLA IDs that include the specified service.

Exceptions: Throws `SlmWebServiceException` when any type of error occurs.

Part 3

End User Management

Chapter 6

EUM Admin Open API

The EUM Administration Open API enables you to perform operations on EUM configurations without using the EUM Administration user interface in Business Service Management. The API supports retrieving, updating and creating Business Process Monitor and Real User Monitor configurations.

The EUM Admin Open API is a RESTful Web service. It contains an integrated server side module that is compliant to JAX-RS v1.0 standard. In addition, a java client based on the Apache Wink framework is provided.

The EUM Admin Open Api classes are documented in Javadoc format in the BSM EUM Administration API Reference. These files are located in the following folder:

\\< Gateway Server root directory>\AppServer\webapps\site.war\amdocs\eng\doc_lib\API_docs\EUM_API\EUM_Administration_API_CSH.htm

Part 4

Operations Management

Chapter 7

Extending Operations Management

This Operations Manager i Extensibility Guide provides information to customize and extend the functionality of HP Business Service Management (BSM) Operations Management provided by the Operations Manager i (OMi) license.

The guide is divided into the following sections.

- Section I: "Content Development" on page 139

- Section II: "Populating the Run-time Service Model" on page 176

- Section III: "Event Processing Interface" on page 245

- Section IV: "Integrating the Operations Management UI with Other Applications" on page 268

- Section V: "Automating Operator Functions and Event Change Detection" on page 280

- Section VI: "Integrating External Event Processes" on page 329

Section I: "Content Development" on page 139 provides steps required for content developers to add management capabilities for a new application, using the fictitious ACME environment as a simple example. The example illustrates the various integration steps required in order to make management information of the new application available in Operations Management.

Section II: "Populating the Run-time Service Model" on page 176 provides information for developers to create their own topology synchronization mapping rules, to augment the out-of-the-box mapping rules to populate the Run-Time Service Model (RTSM) with configuration items (CIs) and CI relationships from nodes and services in HP Operations Manager (HPOM).

The ACME environment example, introduced in Section I: "Content Development" on page 139, is developed further to illustrate how to create topology synchronization rules specific to a particular service model.

Section III: "Event Processing Interface" on page 245 describes the role of event processing scripts and custom actions for modifying and enhancing events during event processing.

Section IV: "Integrating the Operations Management UI with Other Applications" on page 268 describes how to integrate parts of the Operations Management user interface with an external application using a drill-down URL launch.

Section V: "Automating Operator Functions and Event Change Detection" on page 280 provides integrators with information to allow them to programmatically automate operator functions and detect event changes using the Event Web Service. Everything that an operator can do in the console while working on events can be done programmatically, to improve efficiency.

Section VI: "Integrating External Event Processes" on page 329 describes the Event Synchronization Web Service interface that enables integrations with external applications, where the aim is to integrate external processes into event processing. The interface enables notification of forwarded events and subsequent changes to be received programmatically.

The main use case for the Event Synchronization Web Service interface is to synchronize events and changes to events with an external manager such as an ITIL incident manager, for example, HP Service Manager or BMC Remedy Service Desk.

Prerequisites

The prerequisites for extending and customizing OMi are as follows:

- You should be familiar with the relevant HP Software products and components, including the associated documentation. As an example, the following list includes the essential steps required to extend the integration between HPOM:

  - **Run-Time Service Model** (RTSM). Detailed administrative and operational knowledge is assumed. The *HP Business Service Management Modeling Guide* is an essential reference for RTSM concepts, including CI types, CI attributes, and views. For details on importing data from Excel Workbooks and other external sources, see "Import From Excel Workbook Discovery" and "Importing Data from External Sources" in the HP Universal CMDB Discovery and Integration Content Guide. These guides are available on the HP Software Product Manuals Web site (http://h20230.www2.hp.com/selfsolve/manuals) under the Universal CMDB (Application Mapping) product.

  - **HP Operations Manager** (HPOM) for Windows and UNIX. In particular, take a look at the Service Tree in HPOM, and decide which services should be synchronized with the RTSM.

    > **Note:** Services that are used only to build up the parents of a tree (for example, Applications, and Systems Infrastructure) probably do not need to be synchronized, because the RTSM uses graphs and does not require a tree structure.

  - **Operations Manager i** (OMi). For example, the following steps must be executed in OMi:

    i. Using the HPOM service tree as reference, mark the services that you want to synchronize with a Context in the `contextmapping.xml` file.

    ii. Choose the CI type for each HPOM service. Enter the mapping from an HPOM service to a CI type in the `typemapping.xml` file.

    iii. Make sure that all required CI Attributes (key attributes) are filled from HPOM service attributes in the `attributemapping.xml` file.

    iv. Create all relationships in accordance with the RTSM model in the `relationmapping.xml` file.

    > **Note:** Some CI types (for example, Running Software) have keys that consist of simple CI attributes and relations. For these CI types, the `attributemapping.xml` must set all key attributes, and the `relationmapping.xml` must create the correct relations, to instantiate the CI correctly.
    >
    > You can refer to the standard topology synchronization packages, such as `default`, `operations-agent`, and `nodegroups`, as examples of how to work with synchronization packages. You can find these under:
    >
    > *<HPBSM root directory>*`/conf/opr/topology-sync/sync-packages`

    This information is contained in the BSM Extensibility Guide.

  - **HP Business Service Management** (BSM).

- Knowledge of Groovy scripting and syntax is required for creating Groovy scripts. Groovy is supported for scripting, and Groovy scripts are used in the topology synchronization process, for Event Processing Interface (EPI) and custom action scripts, for automating operator functions, and for integrating external event processes.

- Knowledge of the XPath query language is required to navigate through the CI data structure in the mapping engines.

Chapter 8

Content Development

This section describes the steps required to:

- Customize the existing monitoring configuration data supplied out-of-the-box according to customer requirements.

- Add monitoring capabilities for new applications and elements of the IT environment.

The steps are illustrated using a simple example for the ACME environment.

This section is structured as follows:

Integration Content

When you integrate a new application area into a monitoring solution, the following areas are important to the integration:

- "Topology" below

- "Event Type and Health Indicators" below

- "Correlation Rules" on the next page

- "Additional Event Processing" on the next page

- "Tools" on the next page

- "View Mappings" on the next page

- "Graphs" on page 142

Topology

Topology data is contained in a Run-time Service Model (RTSM). The RTSM contains definitions of configuration item types (CI types), and how those CI types can potentially be related to other CI types. Configuration items (CIs) are instances of CI types.

To integrate a new application into a BSM Operations Management monitoring solution, and create a topology view of the new application, it may be necessary to:

- Create new CI types for the new application.

- Identify the key attribute values for the new CI types.

- Establish the relationships (for example, membership, dependency, and composition relationships) for the new application.

- Create CIs and CI relationships in the RTSM.

The effort required to integrate the topology data for a new application depends on what data already exists. For example, integrating an application where you can re-use existing RTSM objects will require less effort than integrating one where you need to start at the beginning, and define all the RTSM CI types and their relations.

For more details about the role of topology data in integrating a new application, see "Topology" on page 143.

Event Type and Health Indicators

To benefit from the advanced health-based monitoring features for your new application area, offered to you by a BSM monitoring solution running Operations Management, it is necessary to:

- Populate the RTSM with CIs, and Operations Management events must be mapped to the correct CIs in the RTSM.

- Transform received events into data about the service health of CIs. This involves analyzing incoming events for the various CI types and creating meaningful event type indicators (ETIs) and health indicators (HIs).

- Assign HIs to health-based key performance indicators (KPIs).

For more details, see "Event Type and Health Indicators" on page 153.

Correlation Rules

The event management process is simplified for you by not only consolidating events from all sources in a central console, but also correlating events using topology-based event correlation (TBEC).

TBEC rules make associations between a known root-cause event and related symptom events. Symptom events are those events that occur as a consequence of the root-cause event. TBEC greatly reduces the number of events displayed in the browser by avoiding duplication and overload. This enables you to manage the problem of large numbers of similar (related) symptom events in a large network.

HI and ETI values are used to represent the events that have an impact on the configuration item types specified in the TBEC rule. These values are used to create correlation rules.

For more details about correlation rules, see "Correlation Rules and Mapping" on page 165.

Additional Event Processing

You can perform additional event processing to modify and enrich events using Groovy scripts.

The Event Processing Interface (EPI) lets you run a number of user-defined Groovy scripts during event processing.

You can also configure custom actions to apply to events.

For more details about additional event processing, see "Additional Event Processing" on page 167.

Tools

You can configure tools to help you manage and monitor specific events, and to solve event-related problems associated with your new application area.

For an example of a tool configured for the new application, see "Tools" on page 168.

View Mappings

You can map configuration item types to RTSM views using the RTSM Modeling Studio, so that views are available for selection and use in the Health Top View pane.

For more details about mapping CI types to RTSM views, see "View Mappings" on page 169.

Graphs

Graphs and charts provide you with additional data to help you visualize and analyze performance-related problems and trends affecting the configuration items impacted by an event for your new application area.

For more information about graphs, see "Graphs" on page 170.

Topology

This section shows how to integrate a new application, and create a topology view of the new environment, using an example application environment called "ACME".

This section is structured as follows:

Integrating New Applications

To integrate a new application into a BSM monitoring solution running Operations Management, and create a topology view of the new application, it is necessary to:

- Create new CI types for the new application (if existing CI types cannot be reused)

- Set the key attribute values for the new CI types.

- Establish the relations for the new application.

- Create CIs and CI relationships in the RTSM.

Create New CI Types for the New Application

The first step in integrating your new application is to create new CI types for the application.

The following graphic shows the topology model of the "ACME" environment.
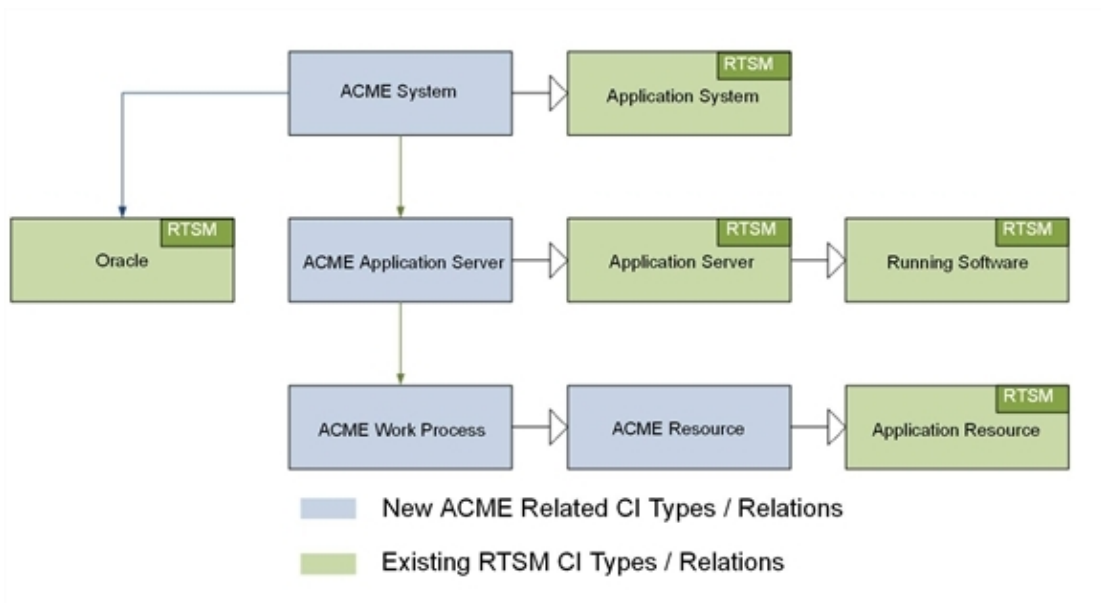
In this example ACME environment, an **ACME system** contains various **ACME application servers**. These application servers employ **ACME work processes** to execute user requests.

The ACME environment uses an **Oracle database** to store all information. The graphic shows four new CI types, depicted in blue:

- ACME System

- ACME Application Server

- ACME Work Process

- ACME Resource

These new CI types are child elements of CI types that already exist in the RTSM (shown in green).

Comprehensive details about how to create new CI types, and how to work with the concepts of the RTSM, are described in the BSMthe Modeling Guide.

To create a new CI type, do the following:

1. Navigate to the CI Type Manager:

   RTSM Administration > Modeling > CI Type Manager

2. In the CI Types pane, activate the CI Types tree by selecting **CI Types** from the drop-down menu.

3. In the CI Types tree, navigate to the folder where you want to add your new application, for example:

   Configuration Item > Infrastructure Element > Application System

4. Right-click, and click the New (⊛) button. The Create Configuration Item Type dialog box opens.

5. In the Name field, enter the name of the CI type to be created: **acme_system**.

   In the Display Name field, enter the display name of the CI type: **ACME System**.

Optional. In the Description field, enter a description of the CI type you are creating.

Click **Next** to proceed to the next page of the Create Configuration Item Type dialog, where you set the key attributes for the new CI you created, as described in "Set Key Attribute Values for New CI Types" below below.

Set Key Attribute Values for New CI Types

It is essential to identify the new CI types with unique key attributes. By setting unique key attributes, you can make sure there are no duplicate CIs created, for instance, by different discovery sources.

The following table shows a list of potential key attributes for an ACME environment.

List of CI Type Attributes for the ACME Environment

CI Type Display Name	Attribute	Description	Value
ACME System	`name`	Name of an ACME system	System ID
ACME Application Server	`name`	A unique name that identifies an ACME server within an ACME System landscape	ACME server name
	`root_ container`	The container host	CI reference (CI ID) of the host
ACME Work Process	`name`	A logical single-instance representation of a certain type of work process	Work process category; batch, dialog or spool

1. In the Create Configuration Item Type-Attributes dialog, set the key attributes for the CI type, for example, ACME System.

   To identify your new CI type with an existing attribute, click in the column marked **Key** on the same row as the attribute you want to set as a key attribute. A small key icon then appears. (Click again if you do not wish to set that attribute.)
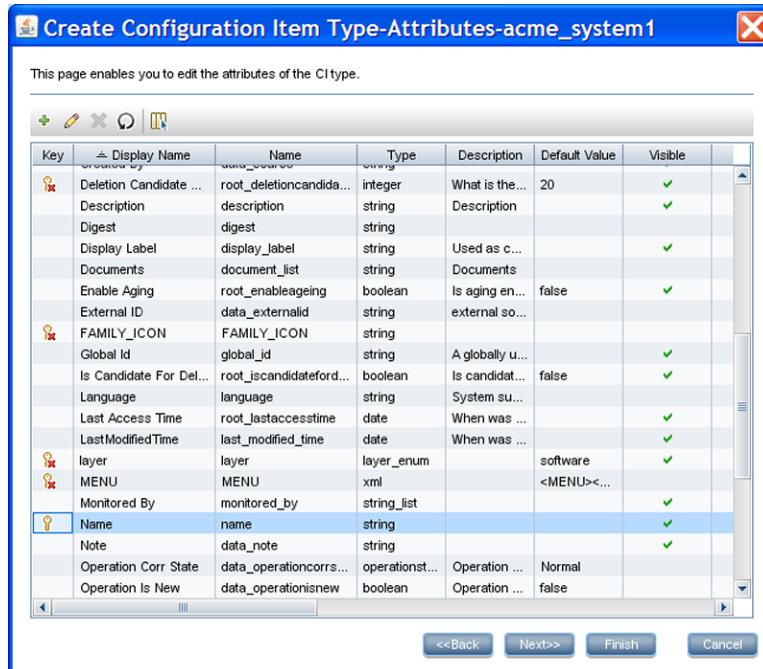
2. In addition to setting key attributes for the new CI type, you may want to create your own attributes used solely for that CI type.

   To create a new attribute:

   a. In the Create Configuration Item Type-Attributes dialog, click the Add ( ) button. The Add Attribute dialog box opens.
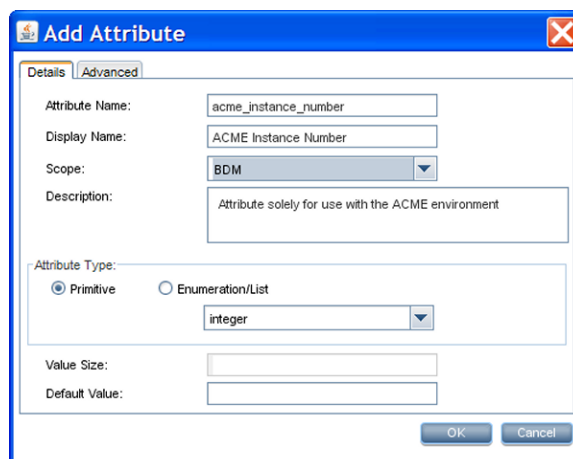
   b. In the Attribute Name field, enter the name of the new attribute: `acme_instance_ number`

   In the Display Name field, enter the display name of the new attribute: **ACME Instance Number**

   In the Scope field, choose the scope: **BDM**

   Optional. In the Description field, enter a description for the new attribute.

    c.   Set the attribute type, and if applicable, fill out the Value Size and Default Value fields.

    d.   Click **OK**.

3. Set the newly created attribute as a key attribute for the ACME System CI type, as described in step 1.

4. Click **Finish**.

Establish Relationships for the New Application

The next step in integrating the new application is to establish the relationships for the new application.

In the out-of-the-box RTSM model, there is a **membership** relationship between the ACME system and the ACME application server. Our example ACME system depends on an Oracle database. This **dependency** relationship, however, does not exist in the out-of-the-box RTSM model. You must create this relationship using the CI Type Manager:

RTSM Administration > Modeling. CI Type Manager

To create the dependency relationship between the ACME system and the Oracle database, do the following:

1. In the CI Type Manager, select **ACME System** and **Oracle**.

2. Right-click, and select **Add/Remove Relationship**.

3. In the Add/Remove Relationship dialog, check the box for Dependency in the **ACME System > Oracle** column, to establish the relationship that our ACME system depends on an Oracle database.
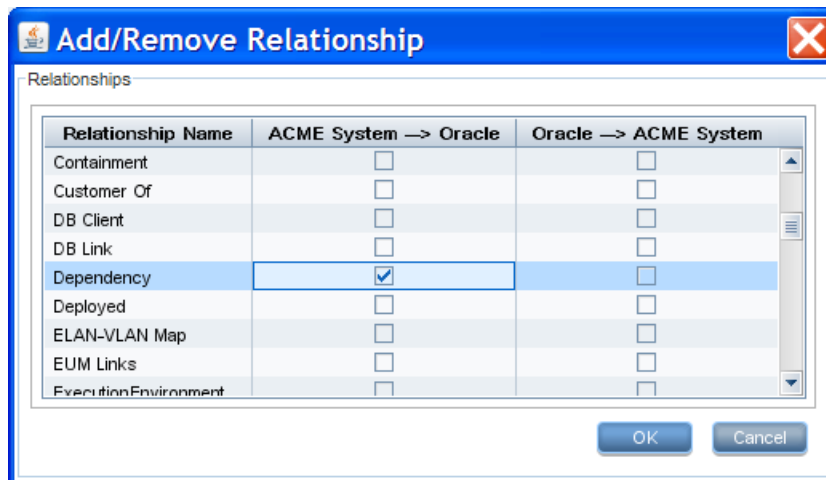
4. Click **OK**.

Creating CIs and CI Relationships in the RTSM

The next step in integrating the new application is to create CIs and CI relationships in the RTSM.

There are three methods to create CIs and CI relationships:

- Create the CIs and CI relationships using discovery features of the RTSM and HP Data Flow Management (DFM).

- Create the CIs and CI relationships using topology synchronization. Topology synchronization reuses the HPOM service model to create corresponding CIs and CI relationships. This, of course, requires that a suitable service model in HPOM has been created.

  For more details about topology synchronization, see "Populating the Run-time Service Model" on page 176.

- Create CIs manually in the RTSM. This option is really only practical if you need to create just a limited number of CIs, and that these CIs are stable in nature, and are not expected to change.

Creating CIs Using RTSM/HP Data Flow Management Discovery

HP Data Flow Management automatically discovers and maps logical application assets in Layers 2 to 7 of the Open System Interconnection (OSI) Model. The discovery technology is based on discovery patterns.

The Data Flow Management licensing structure is as follows:

- UCMDB Foundation License. The Foundation license includes UCMDB as the backbone component for BTO products. This version enables data flow between multiple instances of UCMDB, and integration with BTO products to enable solution deployment.

- UCMDB Integration License. The Integration license adds third party integrations on top of the UCMDB Foundation license.

- UCMDB DDM Advanced License. The DDM Advanced license includes all discovery capabilities to discover the IT infrastructure elements and feed that information as CIs and Relationships to the RTSM. DDM Advanced license also allows users to extend the RTSM model and write their own discovery patterns.

Using Data Flow Management, it is also possible to query external data sources:

- Comma Separated Value (CSV) Files

- Properties Files

- Databases

For more details, see the Data Flow Management Guide and the RTSM Developer Reference Guide in the HP Business Service Management documentation library.

Creating CIs Using Topology Synchronization

Many HPOM customers use HPOM service views or the Service Navigator to visualize dependencies between IT resources and IT services to their operators.

If you take this route, then you would use either the service discovery features of the HP Operations Smart Plug-ins or your own discovery mechanisms to create the service tree.

If this is the case, then you can use the topology synchronization feature to create CIs and CI relationships, based on the HPOM service model and topology synchronization mapping rules. Out-of-the-box mapping rules are provided that are able to map service models created by the following HP Operations Smart Plug-ins, all of which are enabled to work with Operations Management:

- HP Operations Smart Plug-in for Databases (Oracle and MS SQL Server only)

- HP Operations Smart Plug-in for IBM WebSphere Application Server

- HP Operations Smart Plug-in for BEA WebLogic Application Server

- HP Operations Smart Plug-in for Microsoft Active Directory

- HP Operations Smart Plug-in for Microsoft Exchange Server

- HP Operations Smart Plug-in for Virtualization Infrastructure

- HP Operations Smart Plug-in for Systems Infrastructure

- HP Operations Smart Plug-in for Cluster Infrastructure

You may have invested significant effort in creating your own custom service model, together with a corresponding manual or automatic service model creation process. You can reuse that model to create corresponding RTSM configuration items automatically. All you need to do is to write corresponding topology synchronization mapping rules for your model.

For more details about topology synchronization, see .

Considerations when Choosing a Discovery Method

The following considerations are useful in helping you decide which discovery option to use to populate the RTSM.

- When to Use the Data Flow Management Guide

  Use the Data Flow Management Guide if:

  - You already use the Data Flow Management Guide to populate the RTSM, or are planning to use it.

  - You do **not** already have a service model in HPOM. We recommend using the Data Flow Management Guide, as it is the preferred discovery method for populating the RTSM for Operations Management.

- When to Use Topology Synchronization

  Use out-of-the-box topology synchronization rules provided in the synchronization packages if:

  - You are not using (and have no plans to use) the Data Flow Management Guide to populate the RTSM, *and*

  - You already have a service model in HPOM containing services corresponding to your ACME topology.

- When to Create CIs Manually

  You can create CIs manually if the CIs you want to create are limited in number, and are stable in nature, so are unlikely to change.

Enrichment Rules
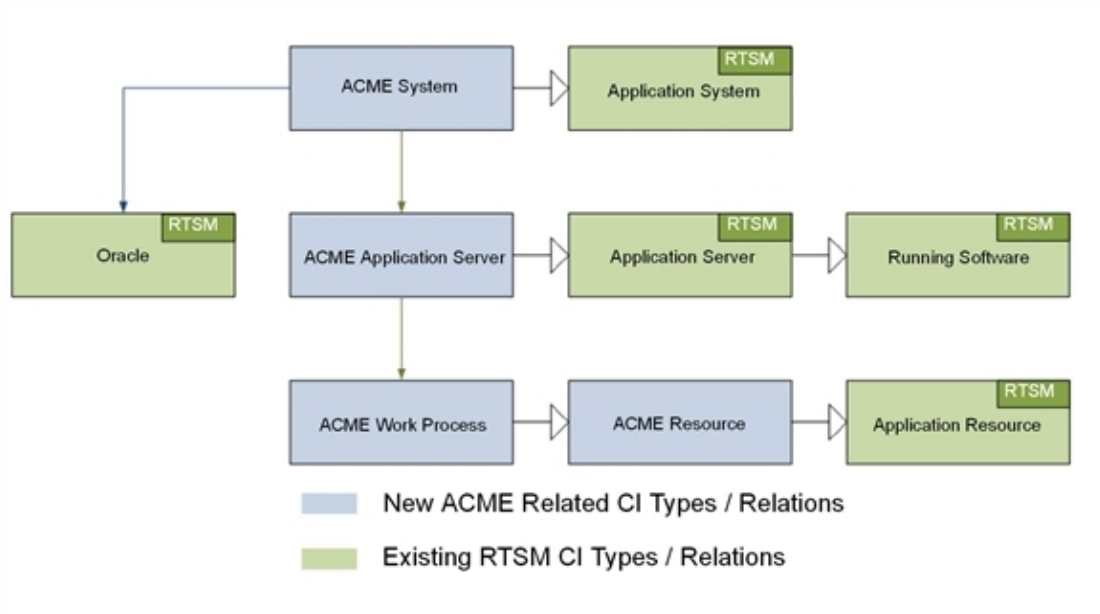
The illustration shows the ACME model, where there is a cross-domain relation between an ACME system and an Oracle database. If there are insufficient key attributes for the ACME discovery to create the Oracle database CI, this dependency relation can be generated by an enrichment rule.

More information about how to create and maintain enrichment rules, see the section entitled "Enrichment Manager" of the Modeling Guide.

Topology View of the New Application

Use the RTSM Modeling Studio to create a view to display the ACME topology. The following screenshot shows the ACME topology view in the RTSM Modeling Studio.
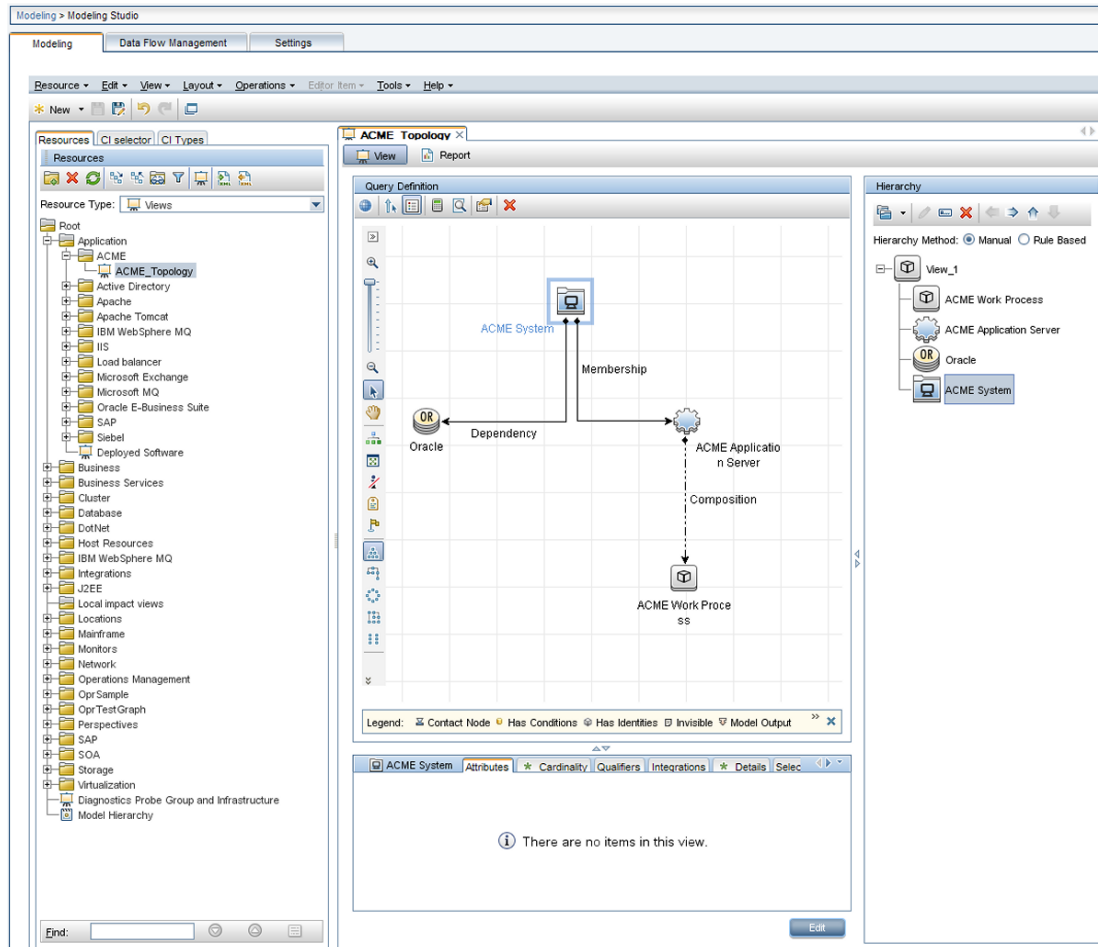
ACME topology view in RTSM Modeling Studio

Impact Propagation

Impact modeling is performed using calculated relations. Impact relationships are important for KPI calculations. For a detailed description of impact modeling in the RTSM, including concepts, see the Modeling Guide.

> **Note:** You can create a specific view to verify the impact propagation within your topology. Create a new view and choose `impacted by` as the relationship type between each CI type.

The following graphic shows a calculated relation (a triplet), which uses a propagation rule to create an impact relation between the ACME Application Server and the ACME Work Process.
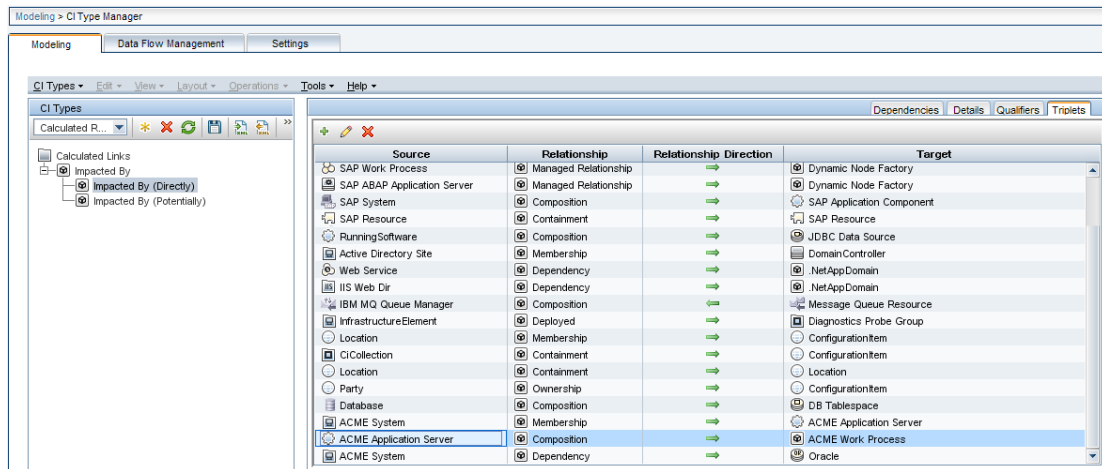
Creating an impact relation between the ACME Application Server and the ACME Work Process

Event Type and Health Indicators

This section shows how to enrich events for your new application to benefit from the advanced event correlation and health-based monitoring features offered as part of a BSM solution.

In our ACME example, we assume that HPOM provides a Smart Plug-In for an ACME landscape. If there were no HP Operations Smart Plug-in, you would have to analyze the ACME application itself to determine its interfaces, and also create policies to monitor such a landscape.

To be able to use the advanced event correlation and health-based monitoring capabilities, it is necessary to:

- Populate the RTSM with CIs, and map Operations Management events to the correct CIs in the RTSM.

- Analyze incoming events for the various CI types and create meaningful event type indicators (ETIs) and health indicators (HIs).

- Assign HIs to health-based key performance indicators (KPIs).

Mapping Events to CIs

The fundamental requirement underlying all advanced event and health monitoring features provided by Operations Management is that events are mapped to the correct CIs in the RTSM.

If events are not mapped to the correct CIs:

- Setting event type or health indicators in HPOM messages will have no effect.

- Correlation rules will never trigger.

- The Health Perspective view will show either incorrect health and KPI data or none at all.

Therefore, it is essential that:

- The RTSM is populated with CIs.

- Adjustments are made (if necessary) to the event integrations in such a way that events can be mapped to these CIs.

Smart mapping technology is employed to map events to CIs. The system looks for hints in certain event attributes, compares these hints with CI attributes, and then maps an event to the best matching CI.

Most events contain at least the DNS name of the affected host (in the HPOM message node name field). As this DNS name is used as one possible hint, the smart mapping will almost always be able to map an incoming event to at least the host CI (assuming, of course, that the host CI exists in the RTSM).

However, to make use of the complex IT topology model, it is important to map:

- Database events to corresponding database CIs.

- Events related to other applications to their respective CIs.

To achieve this, an evaluation is made of the following additional event attributes:

- Application

- Object

- HPOM service ID or the CI Resolution hint attributes

  If the CI Resolution hint attribute is set, the HPOM service ID attribute is ignored.

The more identifying hints that can be provided in these attributes, the higher the likelihood that the event will be mapped to the correct CI.

Hints must be specified using a certain format to allow smart mapping to identify what belongs to one attribute.

The default format is:

- `<hint 1>:<hint 2>:...:<hint n>` for the Application and Object attribute

- `<hint 1>:<hint 2>:...:<hint n>@@<hostname>` for the HPOM service ID and CI Resolution hint attributes

The separator (:) can be configured in the BSM Infrastructure Settings. For details, see the HP Business Service Management online help.

All the hints are then evaluated to find a matching CI in the RTSM. The hints in the Application, Object and HPOM service ID attributes are evaluated for backward compatibility reasons. In the past, many HP Operations Smart Plug-ins used these fields to transport information about which object (or configuration item, in RTSM terms) the event is related to. If this information is sufficient to identify the correct CI, then there is no need to change anything.

However, if you find that an event is related to an incorrect CI, then you should set the necessary hints in the CI Resolution hint attribute. You can do this by setting a custom message attribute (CMA) called `RelatedCiHint` in the HPOM message in HPOM.

Setting the Custom Message Attribute RelatedCiHint

The following graphic shows an example of how to set the CMA `RelatedCiHint` in the HPOM message:
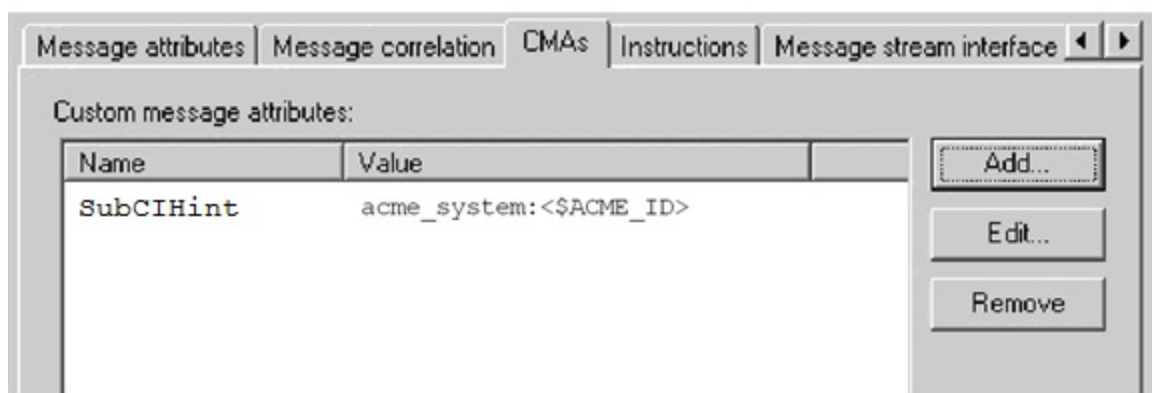
Setting custom message attribute RelatedCiHint in the HPOM message

The following considerations regarding best practice are important to bear in mind when setting the `RelatedCiHint` variable:

- The CMA `RelatedCiHint` should have sufficient hints to find the corresponding CI.

- It is necessary to differentiate between CIs that have a composition relationship to a host, and those that do not have such a relation.

Setting the Custom Message Attribute SubCiHint

The following graphic shows an example of how to set the CMA `RelatedCiHint` in the HPOM message:
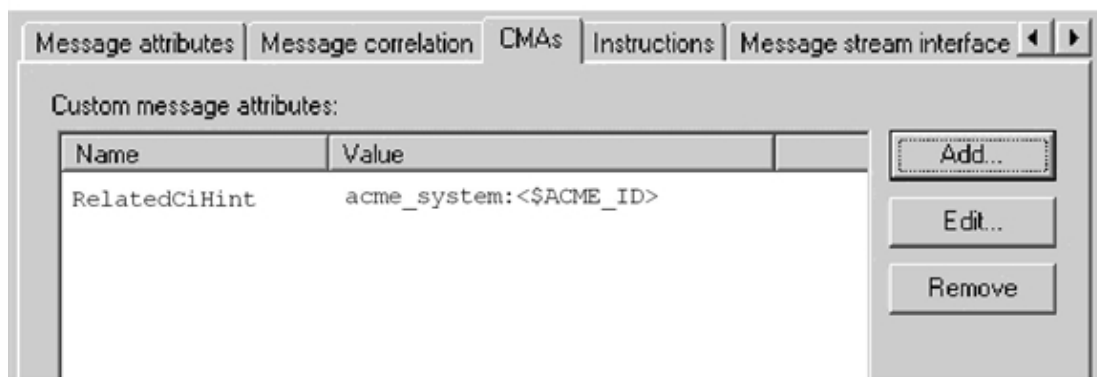
Setting custom message attribute SubCiHint in the HPOM message

The following considerations regarding best practice are important to bear in mind when setting the `SubCiHint` variable:

- The CMA `SubCiHint` should have sufficient hints to find the corresponding CI.

- It is necessary to differentiate between CIs that have a composition relationship to a host, and those that do not have such a relation.

RelatedCiHint Values

In general, the `RelatedCiHint` variable should have the following value:

- **For "hosted on" CIs**

  ```
  <CI-Typename>:<key-attribute-1>:<key-attribute-2>:<key-attribute-
  n>@@hostname
  ```

  Typically, a "hosted on" CI is a sub-type of "software element". For example, a CI of type `websphereas` has a container-link relation to the host.

  Another example is the exchange server role CI type `exchangeclientaccessserver`. The root-container for this CI type is a software element, and for that CI type the root-container is host.

- **For virtual CIs**

```
<CI-Typename>:<key-attribute-1>:<key-attribute-2>:<key-attribute-n>
```

A virtual CI does not have a strong containment relation (container-link or root-container) to a host.

An example of a typical virtual CI type is `cluster`. This CI type does not have a strong containment relation to a host.

Creating Event Type and Health Indicators

After mapping the monitored HPOM events to the correct CIs in the RTSM, the next step is to create event type indicators and health indicators.

Analyze Events and Define Indicators

Incoming events for the various CI types need to be analyzed, so that meaningful event type indicators (ETIs) and health indicators (HIs) can be created.

ETIs are attributes of events (they do not exist as instances in their own right). ETIs are used to categorize incoming events according to the type of occurrence in the managed IT environment. For example, you can configure messages in HPOM to include the custom message attribute (CMA) `EtiHint`, which is used to set event type attributes. If the CMA is not configured, ETIs can be set using applicable mapping rules. At least one value is required for an ETI, which is used to describe the event occurrence in the environment. An example would be `Lost database Connection:`*Occurred*.

Any occurrence on the monitored system of a given type causing an Operations Management event must be assigned the same ETI. After defining appropriate correlation rules, events are correlated based on the ETIs. The correlation rules relate types of events that can occur on the CI.

HIs determine and display the health of specified aspects of a monitored CI. An HI is used to indicate if a hardware resource is available, and uses one value to represent the normal state of the CI. An example would be `ACME System Status:`*Available*. One or more values are used to indicate abnormal states for the CI, such as `ACME System Status:`*Unavailable*.

HIs can also indicate the state of a software application, for example, when the load on certain process is normal, high or exceeded. An example of an abnormal state would be `Job Queue Length:`*Too Long* for the `ACME Work Process` CI type.

Only events that provide CI state information can set a health indicator. Health indicators are assigned to a specific configuration item type through the associated ETI.

HIs also provide the data needed by a key performance indicator (KPI) to calculate the availability and performance of monitored resources. See "Assigning HIs to KPIs" on page 162, which shows how to assign HIs from the ACME example to health-based KPIs.

The new CI types for our ACME example environment were introduced in the section "Integrating New Applications" on page 143, and illustrated in "Integrating New Applications" on page 143.

For these CI types, there are specific ETIs and HIs. "Overview of ETIs and HIs" on the next page shows the result of an investigation into which ETIs/HIs are important within an ACME environment.

Overview of ETIs and HIs

CI Type Display Name	Category	Name	Value	Severity	Policy
ACME System	HI	ACME System Status	Available	Normal	ACME_ SystemStatus001
ACME System	HI	ACME System Status	Unavailable	Critical	ACME_ SystemStatus001
ACME Application Server	HI	ACME Application Server Status	Available	Normal	ACME_ AppServerStatus_ 001
ACME Application Server	HI	ACME Application Server Status	Unavailable	Major	ACME_ AppServerStatus_ 001
ACME Work Process	ETI	Job Aborted	Occurred		ACME_opcmsg_001
ACME Work Process	ETI	Job Start Passed	Occurred		ACME_opcmsg_002
ACME Work Process	HI	Job Queue Length	Normal	Normal	ACME_JobQueue001
ACME Work Process	HI	Job Queue Length	Too Long	Major	ACME_JobQueue001
ACME Work Process	ETI	Lost Database Connection	Occurred		ACME_LogFile001

Note: Health indicators should show the current state of the health of the monitored object. Therefore, events should only set HIs if there is continuous monitoring of the health of the monitored object.

You create HIs and ETIs in the following area of the BSM user interface:

Admin > Operations Management > Monitoring > Indicators

For full details about how to create HIs and ETIs, see the HP Business Service Management online help.

Here is an example of how you would create an HI for the CI type ACME System:

1. Navigate to the following location:

   Admin > Operations Management > Monitoring > Indicators

2. In the CI Types pane, right-click on the CI type you want to set an indicator for, in this example,

ACME System.

3. Click the New ( ) button, and select the kind of indicator you want to create: either Health Indicator or Event Type Indicator. In this case, click **Health Indicator**. The New Health Indicator dialog opens.

4. In the General area of the New Health Indicator dialog, enter the following information:

   In the Display Name field, enter the display name of the HI to be created: `ACME System Status`.

   By default, the Name field is filled automatically. For example, if you enter `ACME System Status` as the Display Name for the target HP Service Manager server, `ACME_Service_Status` is automatically inserted in the Name field. Of course, you can specify your own name in the Name field, if you want to change it from the one suggested automatically.

   Optional. In the Description field, enter a description of the CI type you are creating.

   In the Application field, select the option which is appropriate for your HI from one of the following applications: Service Health, SLM, Both Service Health and SLM.
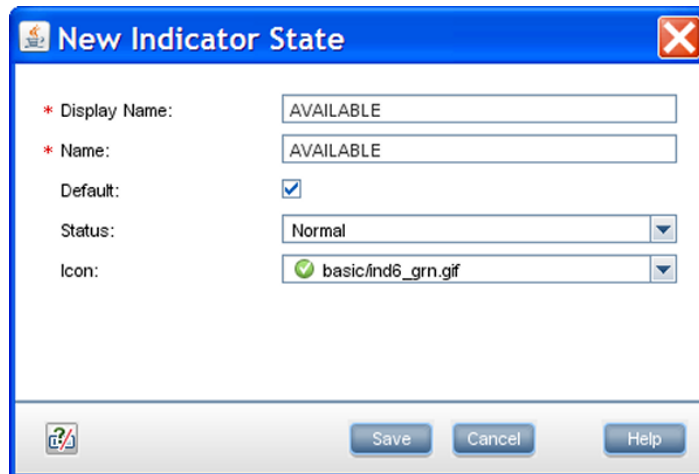
5. In the States area of the New Health Indicator dialog, you can add indicator states by clicking the New ( ) button, or edit an existing state.

   In this example, you are adding new indicator states. You will make one state the default state with Normal severity with the value `AVAILABLE`, and another state with the severity Critical with the value `UNAVAILABLE`.

   To add the default indicator state with the value `AVAILABLE` and with normal severity, do the following:

   a. Select the New ( ) button. The Edit Indicator State dialog opens.

   b. In the **Display Name** field, enter the display name for the HI indicator state: `AVAILABLE`.

   c. In the **Name** field, enter the system name of the HI indicator state: `AVAILABLE`.

   d. Check the **Default** checkbox to make this the default (normal case) state.

   e. In the Status field, choose `Normal.`

   f. In the **Icon** field, choose an icon for this normal state.

   g. Click **Save**.

6. To add the indicator state with the value `UNAVAILABLE` and with critical severity, do the following:

   a. Click the New ( ) button.

   b. In the **Display Name** field, enter the display name for the HI indicator state: `UNAVAILABLE`.

   c. In the **Name** field, enter the system name of the HI indicator state: `UNAVAILABLE`.

   d. Uncheck the **Default** checkbox.

   e. In the **Status** field, choose **Critical**.

   f. In the **Icon** field, choose the icon for this critical state.

   g. Click **Save**.

7. Repeat this procedure for other HIs and ETIs you want to create for your CI types.

Assigning Event Type Indicators to an Event

There are two ways to assign ETIs to an event:

- Use ETI mapping rules

- Set the custom message attribute `EtiHint` within an HPOM policy

Which option you want to use highly depends on the possibility you have to modify the incoming HPOM message. We recommend that you use CMAs to set ETIs for incoming HPOM messages/Operations Management events. However, it is also possible to assign an ETI to an event with ETI mapping rules.

Setting ETIs with the CMA "EtiHint"

"Overview of ETIs and HIs" on page 157 shows a list of the analyzed ETIs and HIs for the ACME monitoring system. The last column gives the information about which HPOM policy creates the event. These HPOM policy conditions must be enriched with the CMA `EtiHint`.

The following graphic shows an example configuration of the CMA `EtiHint`.
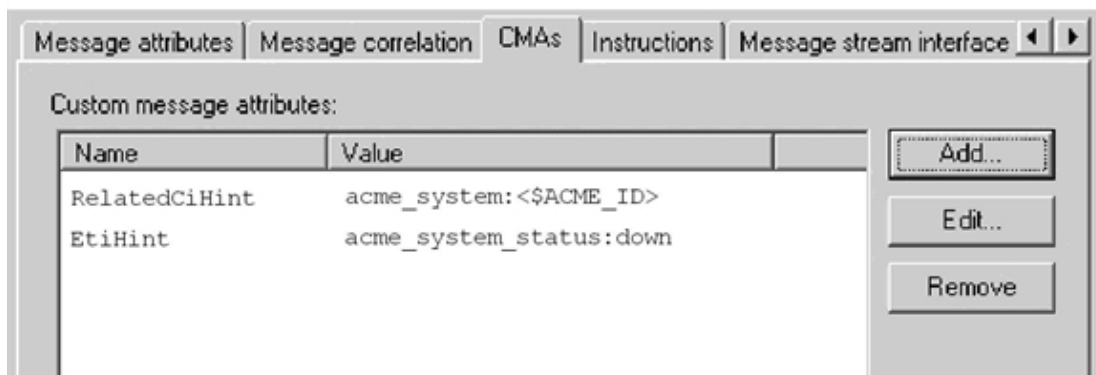
Setting ETIs with ETI Mapping Rules

In the ACME example, all messages are sent using HPOM policies. Therefore, the CMAs are used to set the ETIs. However, it is also possible to use mapping rules to set ETIs for incoming HPOM messages.

The following graphic shows an example of a filter configuration for `db connection down` related messages.

The following graphic illustrates the configuration of the mapping rule itself for the `Lost Database Connection` ETI, using the filter configuration shown above.

With this mapping rule, you match every incoming message where `Category=ACME`, `Sub Category=Work Process` and `Title=db connection lost` to the ETI `Lost Database Connection` for CI type `ACME Work Process`.

Assigning HIs to KPIs

The next step is to assign HIs to health-based KPIs. HIs provide the data that KPIs need to calculate the availability and performance of monitored resources represented by CIs. You assign HIs to KPIs in the following area of BSM:

Admin > Service Health > Assignments > KPI Assignments

For details about KPI assignments, see the HP Business Service Management online help.

The HIs from the ACME example, shown in "Overview of ETIs and HIs" on page 157, mostly represent the availability status of the specific CI. Therefore, those HIs are assigned to the Operations Availability KPI.

The following graphic illustrates the assignment of the HI `Job Queue Length` to the Operations Performance KPI. As a result of this configuration, the HI `Job Queue Length` is also shown in the Health Perspective for events which are related to CIs of type `ACME Work Process`.

In the Condition area of the Add/Edit KPI Assignment for CI Type dialog, you can limit an assignment to a subset of CIs, for example, those monitored by a certain application, or with a

specific set of CI attributes. For full details, refer to the *Indicator Assignments and Propagation* section of the HP Business Service Management online help.
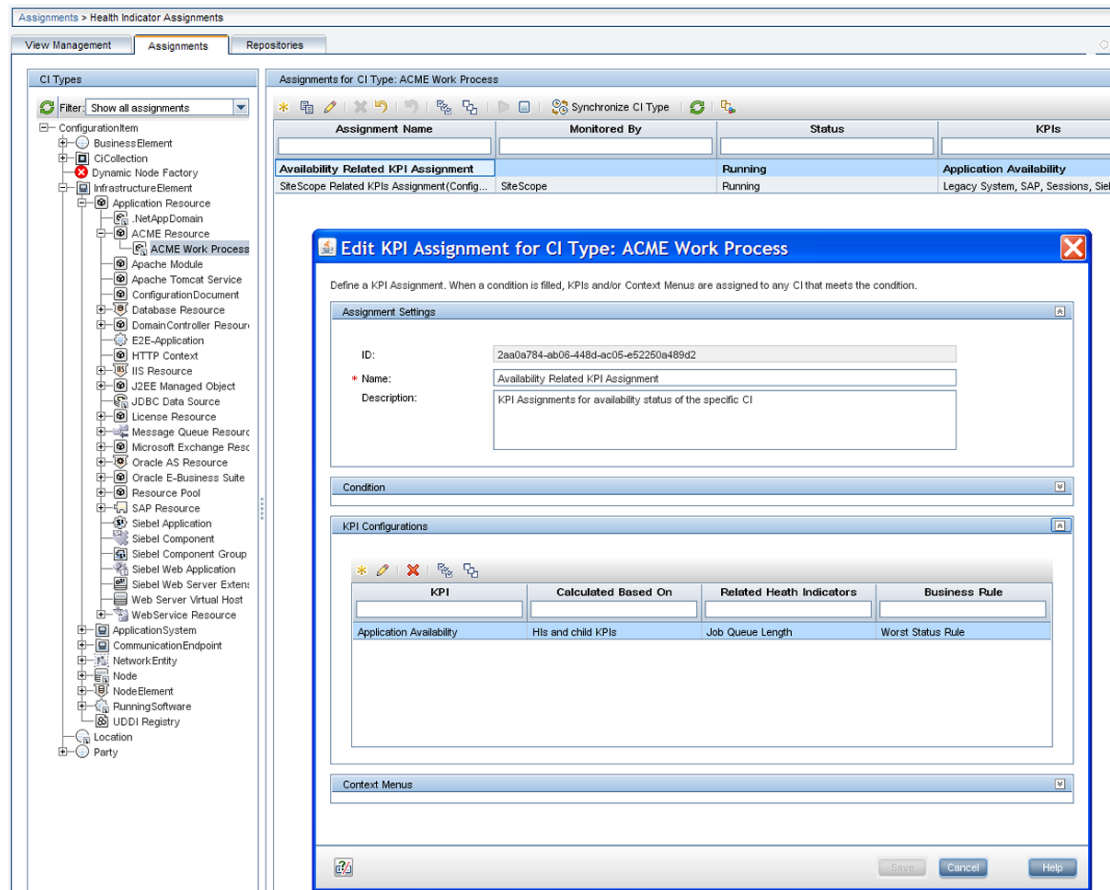
When you have finished with assigning HIs to KPIs, perform a CI type synchronization for the assignment to take effect on existing CIs.

Custom Message Attributes in HPOM

The following table lists the supported custom message attributes that can be specified in HPOM messages and affect the value of the corresponding Operations Management event property.

CMAs in HPOM

CMA	Description
Description	Text used to describe the cause of the event.
NoDuplicateSuppression	Event is not suppressed when it is a duplicate of an existing event.
Solution	Text field used to document solutions to help operators solve the problem indicated by the event.

CMA	Description
SubCategory	Name of the logical subgroup (category) to which the event belongs (for example, Oracle (database), Accounts (security), or Routers (network)).
EtiHint	Event information used to identify the ETI for each CI.
NodeHint	Event information used to identify node CI related to the event.
RelatedCiHint	Event information used to identify the CI related to the event.
SourceCiHint	Event information used to identify the source CI related to the event.
SourcedFromExternalId	Event sourced from external system with specified ID.
SourcedFromExternalUrl	Event sourced from web application with specified URL.
SourcedFromDnsName	Event sourced from external system with specified DNS name.

Correlation Rules and Mapping

You can define correlation rules that correlate related events occurring in the same or different domains of the managed IT environment. Topology-Based Event Correlation (TBEC) is used to automatically identify and display the real cause of problems. Events that are only symptoms of the cause event can be filtered out using the Top Level Items filter, resulting in a clearer overview of the actual problems that need to be solved. Correlating events reduces the number of events displayed in the Event Browser and helps operators to locate the cause of the problems more quickly and efficiently.

You should think about which events of the ACME landscape are symptoms or causes of other events that you receive and check whether correlation rules can be defined.

For example, as illustrated in the following graphic, the ACME landscape has a dependency to an Oracle database. We assume here that the monitoring of such a database is realized through the HP Operations Smart Plug-In for Oracle (SPI for Oracle) and that an event is received when the database is no longer available. At the same time, one of the ACME policies detects a lost database connection as well, so it makes sense to define a correlation rule for these two related events. The following graphic shows such a rule, which defines that the event with the ETI `Lost Database Connection Occurred` is the symptom event, and the event with the ETI `Database Server Status Down` is the cause event.
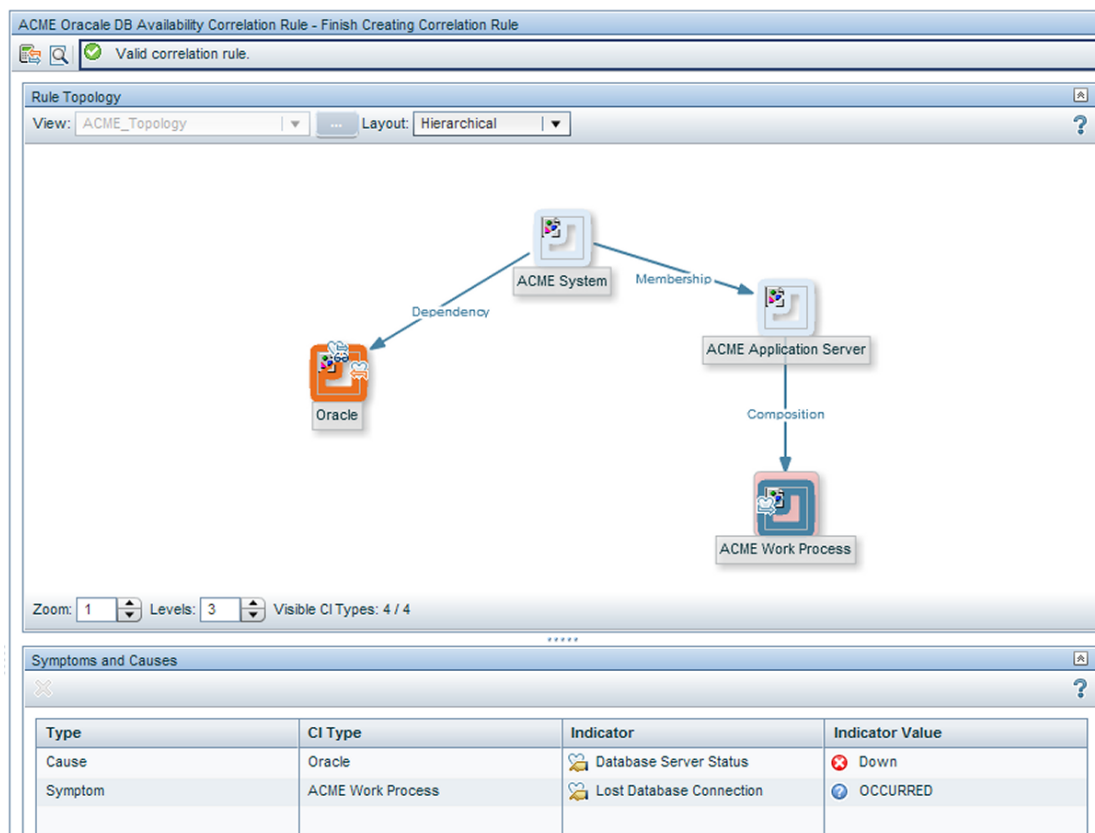
Another example where defining a correlation rule would be appropriate is a scenario where a `long job queue` event is the cause of many `Job Start Passed` messages.

Of course, you can define more elaborate TBEC rules for multiple symptoms and multiple causes.

For full details about correlation rules and mapping, see the Event Correlation section of the HP Business Service Management online help.

Additional Event Processing

You can perform additional event processing to modify and enrich events. For example, you can enrich the Title and Description fields of an event with either or both node and CI labels by using the variable placeholders: `${node.label}` and `${relatedCi.label}`.

The Event Processing Interface (EPI) lets you run any user-defined Groovy scripts during event processing. You may want to enrich events, for example, from an external SQL database.

For an overview of the event processing pipeline, the Event Processing Interface, and EPI scripts, see "Event Processing Interface" on page 246.

You can also configure custom actions to apply to events. You can configure Groovy scripts to make custom actions available in the Event Browser.

For an overview of custom actions and scripts, see "Scripts For Custom Actions" on page 256

For details about how to create EPI and custom action scripts, including some sample Groovy scripts provided with the product, see "Creating EPI and Custom Action Scripts" on page 257.

Tools

You can configure tools to help domain operators manage and monitor specific events and CIs, and to solve problems that occur in your new application area.

By assigning tools to a particular CI type, you can make sure that the assigned tools are always available in the context of any event that has an impact on any instance of the selected CI type.

In the ACME example, we provide a cross-URL launch from the Operations Management event browser to the ACME administrator console.

For more details about tools, see the Operations Management online help.

The following graphic shows an example of a URL launch to the ACME monitoring console. The tool uses the custom event attributes `App_Host` and `App_Port` to generate the appropriate URL.
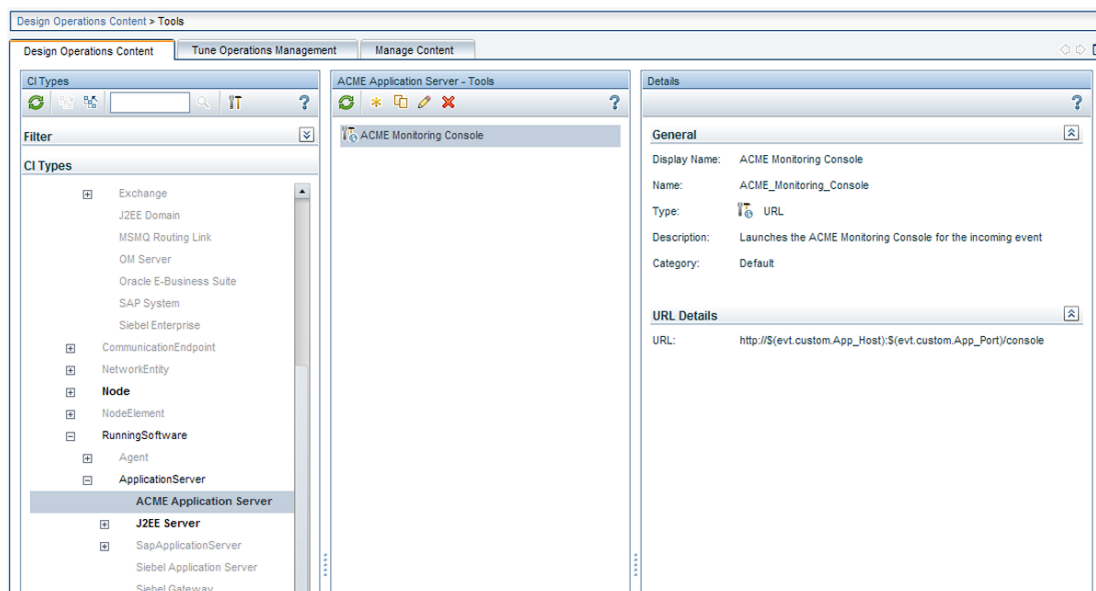
View Mappings

To ensure that views are available for selection and use in the Health Top View pane, the configuration item impacted by the event selected in the event browser must be explicitly mapped to at least one existing view. The RTSM Modeling Studio enables users to create new views in the RTSM. You can associate CI types to more than one view. Multiple views are listed by precedence, and the view with the highest precedent is shown as the default view.

With the View Mappings manager you can map CI types to RTSM views. Only those views that are mapped to the CI type related to the selected event are available for display in the Selected Views drop-down list in the Health Top View pane.

The following graphic shows the View Mappings pane. The CI type `ACME System` is mapped to (has an association with) the view `ACME_Topology`.
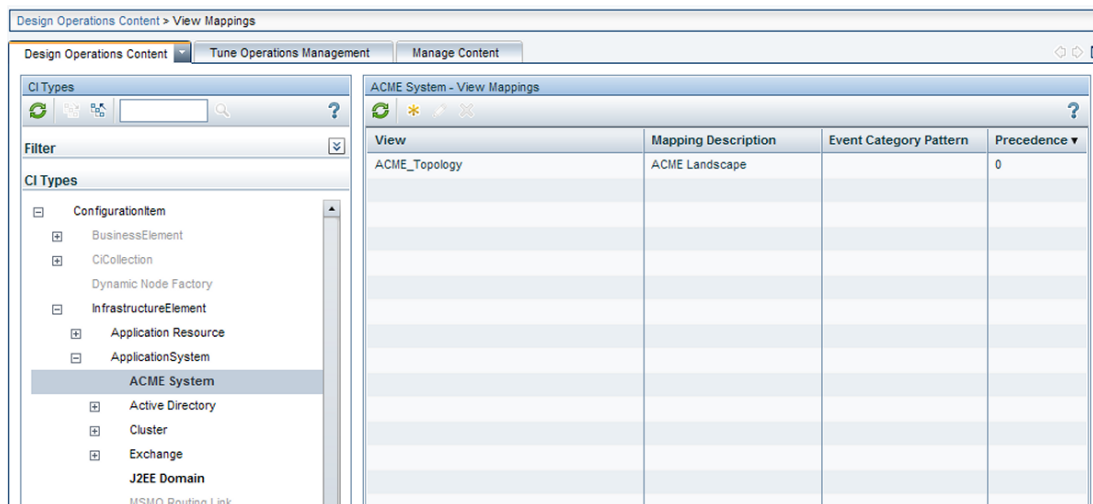
Graphs

You can associate performance graphs with a specific CI type so that graphs and charts of a particular type are always available in the context of any event that has an impact on the selected CI type.

To do this for your new ACME application, you need to:

- Create a new graph template or edit an existing graph template, using the Performance Graphs Designer. For details, see the Operations Management online help.

- Map suitable graph families or categories to the ACME CI types.

Integrating Performance Data

The process of integrating performance data involves:

- Gathering metrics

- Storing the metrics

- Using the metrics to make graphs

For example, performance data for HPOM can be collected by the embedded performance component (EPC) of the HPOM agent or by the HP Performance Agent.

Data collected by these agents can be used in HPOM measurement threshold policies to create alarm messages, and historical data can be displayed using HP Performance Manager or the Performance Grapher of Operations Management, helping operators to analyze and fix problems.

An easy way to integrate performance data is to use the HPOM measurement threshold policy type.

The HPOM measurement threshold policy enables you to collect performance data from several sources:

- External / Program – data sent from an external program (using the `opcmon` command line interface or API)

- MIB – collect metrics from a SNMP Management Information Base (MIB)

- Real Time Performance Measurement – collect metrics from Windows Performance Counters

- WMI – collect metrics from Windows Management Instrumentation

The metrics collected from these sources can be stored easily in the Embedded Performance Component using the `Store in Embedded Performance Component` option.

Packaging Content

This section describes how to transfer the content that you create on one BSM Operations Management instance to another.

If you want to use the content you create on another BSMOperations Management instance, you need to:

- Create and export the RTSM package.

- Create and export the ACME content pack.

- Copy topology synchronization files.

- Upload the files to the other BSMOperations Management instance.

Create RTSM Packages

RTSM packages are essential parts of content. You can use RTSM packages to manage view models and configuration item types. For example, you can use packages for exporting, importing, and updating content, either on the same server or between different instances of BSMOperations Management.

The workflow for creating an ACME RTSM package is summarized below:

1. Create the ACME RTSM package.

   For more information about how to create a content pack, see the HP Business Service Management online help.

2. Add the CI types (see "Topology" on page 143), and views (see "View Mappings" on page 169) that you created in previous steps.

The following graphic shows the selected items which are part of the ACME RTSM package. Usually CI types, TQLs and views are part of such a package.
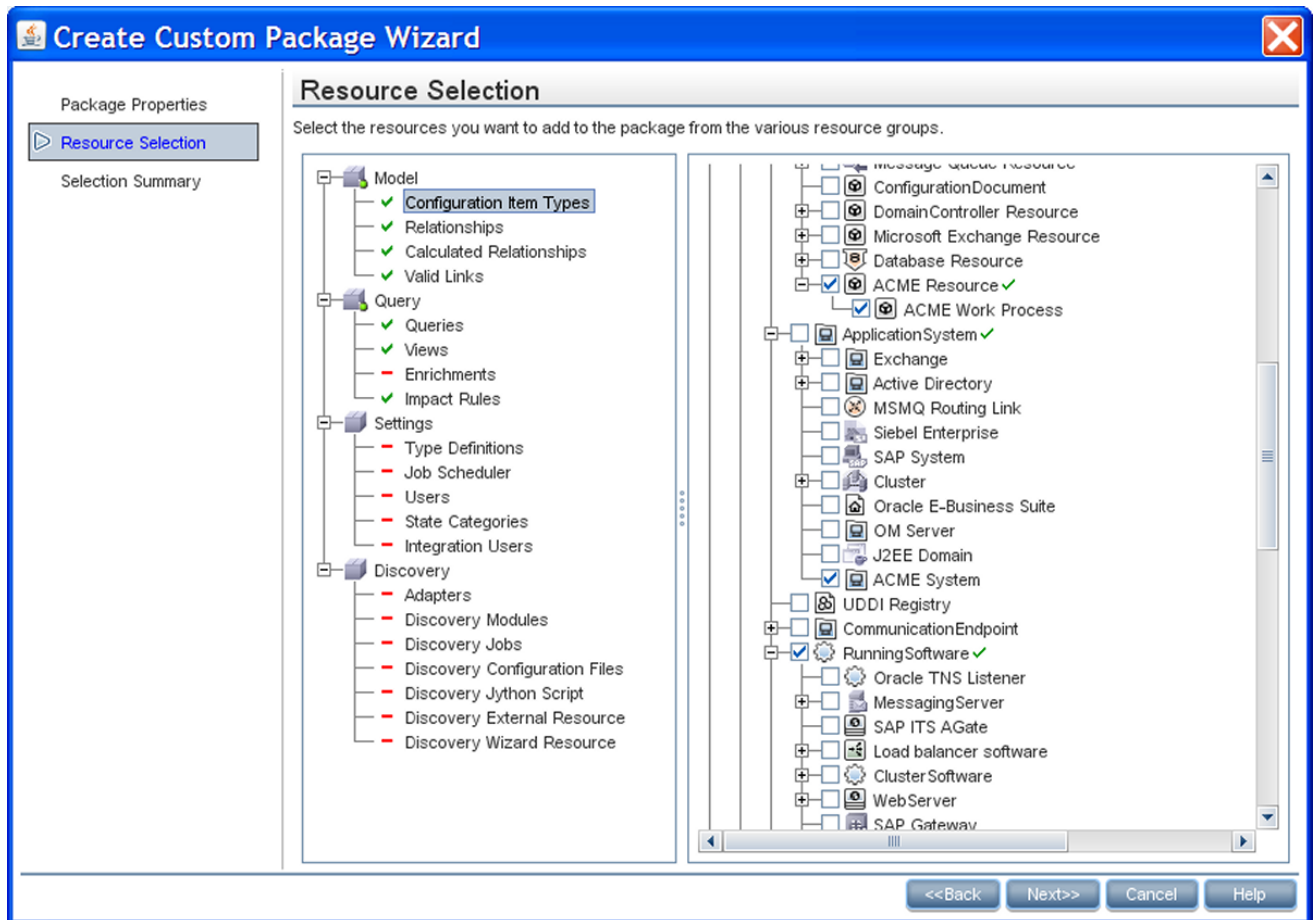
Save Topology Synchronization Files

Collect and save the topology synchronization files so that you can copy them to another instance.

You can find topology synchronization files in the following location:

`<HPBSM root directory>/conf/opr/topology-sync/sync-packages`

For more details about how to create topology synchronization mapping rules, see "Populating the Run-time Service Model" on page 176.

Create a Content Pack

A content pack can contain a complete snapshot of all (or any part of) the rules, tools, graphs, mappings, and assignments that you define and configure to help users manage your IT environment.

You can share content between content packs. In your content pack definition, you can include content directly in your content pack. Content you include in your content pack definition may depend on referenced content. Referenced content is content not explicitly included in your content pack, but referenced in other content pack definitions. For example, when you include a correlation rule as content in your content pack definition, it may require indicators that are referenced to another content pack definition.

How referenced content is handled is guided by the principle that referenced content should be added to a content pack definition only if it is not included in other content pack definitions. Therefore, a content pack definition is very likely to have dependencies to other content pack definitions. Dependencies to other content packs mean that you have to consider what impact the dependencies have. For example, when importing a content packs on another system, the order of loading content packs could be important, and, of course, the referenced content must be present on the target system.

The handling of referenced content is summarized in the following use cases:

- Case 1: Referenced content is not included in other content pack definitions. In this case, the content is added to the content pack you are creating.

- Case 2: Referenced content is included in another content pack definition. In this case, rather than adding the content to your current content pack, a dependency to the referenced content is marked to the other content pack definition.

- Case 3: Referenced content is included in more than one content pack definition. This is similar to Case 2, and the dependency to the referenced content is marked to one of the content pack definitions where the referenced content is included.

The workflow for creating an ACME content pack is summarized below:

- Create the ACME content pack.

  For more information about how to create a content pack, see the HP Business Service Management online help.

- Add all correlation rules, HIs, ETIs, KPI assignments, tools, graphs and graph assignments, and view mappings that you created in previous steps. EPI scripts and custom actions can also be defined in content packs.

- *Optional*. Include content from any dependent content pack definitions.

- Export the content pack XML file (ACME.xml, for the ACME environment content pack) using the content manager. You can exchange content between instances of BSMOperations Management by defining and creating packages using content management tools. The package you create can be exported to a file which you can then use to deploy the same content on another instance of BSMOperations Management.

The following graphic shows a selection in the content manager within the ACME environment content pack.
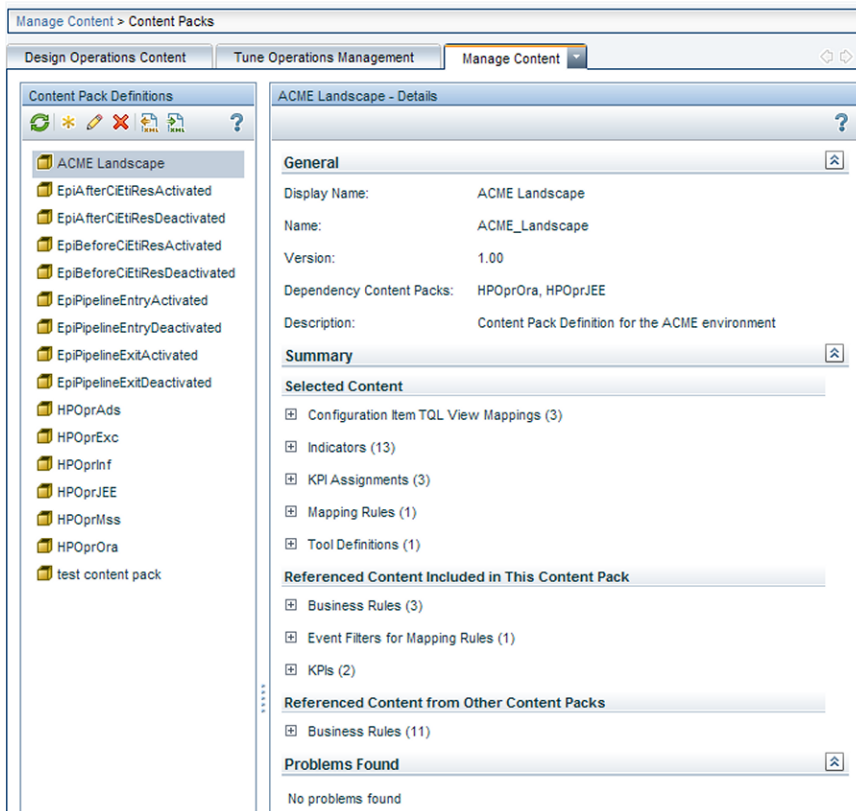
Upload the Content

Copy the exported files to another BSM Operations Management instance, and upload them to the other system in the following sequence:

- Upload the RTSM packages.

- Copy topology synchronization files.

- Upload the content packs.

Upload RTSM Packages

To upload an RTSM package:

1. Place the zip-file in a suitable directory of your choice in the file system.

2. Use the RTSM package manager to upload the RTSM package.

Copy Topology Synchronization Files

Copy the topology synchronization files to the following location on the destination system:

<HPBSM root directory>`/conf/opr/topology-sync/sync-packages`

Upload Content Packs

To upload a content pack:

1. Place the exported XML file in a suitable directory of your choice on the file system.

2. Upload the content pack using the content manager.

Chapter 9

Populating the Run-time Service Model

This section provides information for developers to create their own topology synchronization mapping rules to populate the Runtime Service Model (RTSM (Run-time Service Model)) with configuration items (CIs) and CI relationships from nodes and services in HPOM.

This section reuses the ACME environment example, introduced in Section I: Content Development, which illustrates how to create topology synchronization rules specific to a particular service model.

This section is structured as follows:

- "Topology Synchronization Overview" on the next page
- "Synchronization Packages" on page 194
- "Scripting" on page 205
- "Testing and Troubleshooting" on page 209
- "Mapping Engine and Syntax" on page 220

Topology Synchronization Overview

You can use topology synchronization instead of, or in conjunction with, HP DFM (Data Flow Management) discovery to create CIs in the RTSM (Run-time Service Model). Accurate and up-to-date CI topology data in the RTSM is essential for Topology-based Event Correlation (TBEC), context-specific tools, and BSM-wide service health monitoring (Health Perspective).

Note: Alternatively, you can import existing topology information into the RTSM. For details on importing data from Excel Workbooks and other external sources, see "Import From Excel Workbook Discovery" and "Importing Data from External Sources " in the HP Universal CMDB Discovery and Integration Content Guide. This guide is available on the HP Software Product Manuals Web site (http://h20230.www2.hp.com/selfsolve/manuals) under the Universal CMDB (Application Mapping) product.

Topology synchronization can be defined as a set of rules that determine how services, nodes, and node groups monitored by HP Operations Manager (HPOM) are mapped to OMi configuration items (CIs) in the operational database (RTSM).

Topology synchronization is a standard part of the OMi license. It offers a solution to populate the RTSM with CIs from service models (or alternatively, service trees or service graphs) defined in HPOM. Topology synchronization is especially interesting for HPOM customers who have created their own service model.

Mapping rules used for topology synchronization are contained in topology synchronization packages. The installation copies the topology synchronization rules to the local file system. The rules are then uploaded to the database when dynamic synchronization is used for the first time. You can also write your own topology synchronization rules to populate the RTSM based on your service model or the discovered topology data.

Topology synchronization uses the RTSM API, and uses Wiseman, JAXB, JDOM, JXPath, SPRING, Hibernate, and Groovy scripting technology.

There are two methods available for topology synchronization:

- "Dynamic Topology Synchronization" on the next page

  Note: Dynamic topology synchronization is the recommended method for topology synchronization, and should be used with all versions of HPOM that support dynamic topology synchronization. For details of supported HPOM versions, see the *HP Business Service Management Release Notes*.

  Dynamic topology synchronization receives discovered topology data from *multiple* HPOM and updates CIs and CI relationships in the RTSM database as soon as changes are discovered.

- "Basic Topology Synchronization" on page 184

  Note: Use basic topology synchronization only with those versions of HPOM that do not support dynamic topology synchronization. For details of supported HPOM versions, see the *HP Business Service Management Release Notes*.

Basic topology synchronization enables the topology data from HPOM services, nodes, and node groups to be transferred from a *single* HPOM server to Operations Management using the HPOM web service.

Both topology synchronization methods use the same synchronization packages to map discovery data to CIs and CI relationships in the RTSM.

Dynamic Topology Synchronization

Dynamic topology synchronization makes it possible to share topology data from HPOM between any supported combination of multiple HPOM and Operations Management instances, in a distributed, hierarchical environment.

An Operations Management instance can be configured to receive topology data from multiple HPOM and other Operations Management instances. An Operations Management instance can also be configured to forward topology data to other Operations Management instances.

Topology data is forwarded to the Operations Management instance dynamically, that is, as soon as a topology change is discovered by an HP Operations Agent (and written to the HPOM service model). Dynamic topology synchronization enables near-real-time discovery of infrastructure changes. For example, if a node is added to the environment, this change is immediately forwarded, and the database updated.

For details of supported HPOM versions, see the *HP Business Service Management Release Notes*.

When you first configure dynamic topology synchronization, the source system (HP Operations Agent or HPOM) sends all its topology data to one or more target systems (HPOM or Operations Management). After this first synchronization, dynamic topology synchronization forwards only the discovered topology changes. Changes in topology of an Operations Management instance are *not* forwarded, and *not* synchronized back to an HPOM system using dynamic topology synchronization.

For details about database synchronization, refer to the HP Business Service Management RTSM documentation.

Once configured, dynamic topology synchronization runs continuously in the background. Also running continuously in the background is a process that prevents aging in the RTSM by touching all elements from the previous synchronization. This is equivalent to running basic topology synchronization in touch mode.

For more details about RTSM aging, refer to the Modeling Guide.

To create CIs in the RTSM, based on mapping rules and using HPOM topology data as sources, dynamic topology synchronization does the following:

- Forwards the discovered topology data from configured source forwarding servers to configured target servers, using HTTPS-based communication requiring trusted certificate exchange.

- Receives discovered topology data asynchronously.

- Converts the topology data into a format compatible with Operations Management.

- Uploads the topology data to the RTSM.

- Updates the data in the RTSM on demand.

- Performs delta detection and deletes elements from the RTSM that are deleted in HPOM.

- Provides a mapping table for CI resolution.

> **Note:** The definition of particular CI types is especially important when synchronizing the topologies of Operations Management and HPOM because the topology synchronization process cannot create new CI types. If the topology synchronization process attempts to map to a CI type that does not exist in the RTSM, the topology synchronization process aborts.

Dynamic Topology Synchronization Architecture

The following graphic provides an overview of the dynamic topology synchronization architecture.
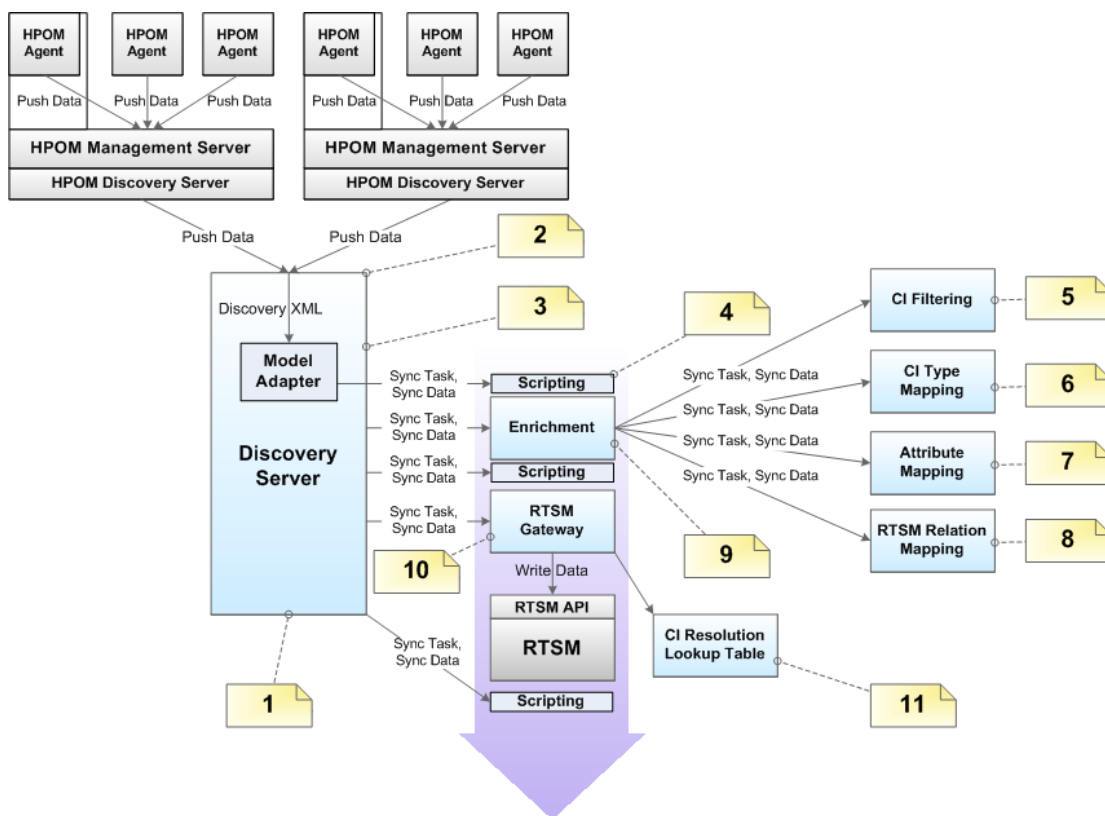
The reference numbers in the diagram refer to the following notes:

Reference	Refers to	Note
1	Discovery Server	- Controls the synchronization process and passes data from one component to the other. - Runs in `hpsbm_wde` process on the gateway servers.

Reference	Refers to	Note
2	Receiving topology data on the Discovery Server	Receive topology data in the form of a discovery XML forwarded from an HPOM server or another OMi server.
3	Model Adapter	Convert the discovery XML to the synchronization data structure.
4	Scripting	Execute Groovy scripts to manipulate synchronization data.
5	CI Filtering	Use the mapping engine to specify which CIs are part of the synchronized model, and which are not.
6	CI Type Mapping	Map Service Type Definitions (STDs) for RTSM (Run-time Service Model) types.
7	Attribute Mapping	Map HPOM relations that are not typed to RTSM relations.
8	RTSM Relation Mapping	- Create the relations for the RTSM. - Map the root container to the CI to be able to create the CI in the RTSM according to the model.
9	Enrichment	Control the whole enrichment process and return the modified synchronization data.
10	RTSM Gateway	Write the synchronization data to the RTSM.
11	CI Resolution Mapping Table	For each synchronized CI, update a mapping table for the CI Resolution component that maps the HPOM Service ID to the RTSM CI ID.

Running Dynamic Topology Synchronization

Before configuring dynamic forwarding of topology (node and service) data to Operations Management from HPOM management servers, complete the following configuration steps in Operations Management:

1. Add the HPOM management server as a connected server in Operations Management. For details, see the Operations Management online help.

2. Establish a trust relationship between the Data Processing Server and the HPOM management server. For details, see the Operations Management online help.

3. *Optional:* Use the `opr-sdtool.bat` command-line tool to upload new or changed synchronization packages from the file system to the database. For details, see the "Uploading and Downloading Synchronization Packages" on page 192.

4. On the HPOM management server, configure forwarding of topology data as described in the following section..

The following sections describe how to configure topology synchronization:

- "How to Configure Dynamic Topology Synchronization on HPOM for Windows Systems" below

- "How to Migrate from Scheduled Synchronization on HPOM for Windows Systems" on the next page

- "How to Configure Dynamic Topology Synchronization on HPOM for UNIX or Linux Systems" on page 183

- "How to Migrate from Scheduled Synchronization on HPOM for UNIX or Linux Systems" on page 183

How to Configure Dynamic Topology Synchronization on HPOM for Windows Systems

This section describes how to configure dynamic topology synchronization on HPOM for Windows management servers. For further details, see the HPOM for Windows documentation.

To forward topology data to Operations Management, complete the following steps on the HPOM for Windows management server from which you want to receive topology information:

1. *Prerequisite:* Make sure that the minimum patch level for the HPOM for Windows management server is installed:

   - Version 8.16: Patch OMW_00121 or superseding patch.

   - Version 9.00: Patch OMW_00122 or superseding patch.

2. *Prerequisite:* Configure trusted certificates for multiple servers.

   In an environment with multiple servers, you must configure each server to trust certificates that the other servers issued.

3. In the console tree, right-click **Operations Manager**, and then click **Configure→Server…**. The Server Configuration dialog box opens.

4. Click **Namespaces**, and then click **Discovery Server**. A list of values appears.

5. Add the hostname of the server to **List of target servers to forward discovery data**. If there is more than one target server, separate the hostnames with semicolons, for example:

   ```
   server1.example.com;server2.example.com
   ```

   If the target server uses a port other than port 383, append the port number to the hostname, for example:

   ```
   server1.example.com:65530;server2.example.com:65531
   ```

6. Make sure that the value of **Enable discovery WMI listener** is true. This is the default value.

7. Click **OK** to save your changes and close the Server Configuration dialog box.

8. Restart the `OvAutoDiscovery Server` process for your changes to take effect.

9. Start the initial synchronization of topology data:

   a. In the console tree, select **Tools > HP Operations Manager Tools**.

   b. Right-click **Synchronize Topology** and select **All Tasks > Launch Tool…**.

The tool `startInitialSync.bat` is started and begins to send all the topology data to the configured target management servers.

How to Migrate from Scheduled Synchronization on HPOM for Windows Systems

This section describes how to migrate from scheduled synchronization on HPOM for Windows management servers. For further details, see the HPOM for Windows documentation.

To migrate from scheduled synchronization, complete the following steps on the HPOM for Windows management server from which you want to receive topology information:

1. *Prerequisite:* Make sure that the minimum patch level for the HPOM for Windows management server is installed:

   - Version 8.16: Patch OMW_00121 or superseding patch.

   - Version 9.00: Patch OMW_00122 or superseding patch.

2. Clear the agent repository cache on the HPOM management server using the following command:

   ```
   %OvBinDir%\ovagtrep -clearall
   ```

3. Remove the service auto-discovery policies from the HPOM management server node, type:

   ```
   %OvBinDir%\ovpolicy -remove DiscoverOM
   ```

   ```
   %OvBinDir%\ovpolicy -remove DiscoverOMTypes
   ```

4. Synchronize the policy inventory on the HPOM management server:

   a. In the console tree, right-click the management server.

   b. Select **All Tasks > Synchronize inventory > Policies**.

      The management server creates a deployment job to retrieve the inventory from the local agent.

5. Make sure the listener process is running:

   a. In the console tree, right-click **Operations Manager**, and select **Configure Server.**

      The Server Configuration dialog box opens.

   b. Click **Namespaces**, and select **Discovery Server**.

      A list of values appears.

   c. Set the value of **Enable discovery WMI listener** to true. This is the default value.

   d. Click **OK** to save your changes and close the Server Configuration dialog box.

   e. Restart the OvAutoDiscovery Server process for your changes to take effect using the following commands:

      ```
      net stop "OvAutoDiscovery Server"
      ```

      ```
      net start "OvAutoDiscovery Server"
      ```

6. Start the initial synchronization of topology data:

a. In the console tree, select **Tools > HP Operations Manager Tools**.

b. Right-click **Synchronize Topology** and select **All Tasks > Launch Tool...**.

The tool startInitialSync.bat is started and begins to send all the topology data to the configured target servers.

How to Configure Dynamic Topology Synchronization on HPOM for UNIX or Linux Systems

This section describes how to configure dynamic topology synchronization on HPOM for UNIX or Linux management servers. For further details, see the HPOM for UNIX or Linux documentation.

To forward topology data to Operations Management, complete the following steps on the HPOM for UNIX or Linux management server from which you want to receive topology information:

1. *Prerequisite:* Make sure that the minimum patch level for the HPOM 9.10 for UNIX or Linux management server is installed:

   ■ HP-UX: Patch PHSS_42736 or superseding patch.

   ■ Linux: Patch OML_00050 or superseding patch.

   ■ Solaris: Patch ITOSOL_00772 or superseding patch.

2. *Prerequisite:* Make sure that the HP Operations Agent version on the HPOM for UNIX or Linux management server is 8.60.500 or higher. (Older agents require the agent hotfix QCCR1A100254 and agtrep must be configured to send complete instance data.)

3. *Prerequisite:* Configure trusted certificates for multiple servers.

   In an environment with multiple servers, you must configure each server to trust certificates that the other servers issued.

4. Type the following command to enable topology synchronization:

   `/opt/OV/contrib/OpC/enableToposync.sh -online -target <`*`comma_`*
   `separated_server_list``>`

   Replace `<`*`comma_separated_server_list`*`>` with the fully qualified domain name of the target management server. If you have more than one target management server, separate each server name with a comma (,). Do not include spaces in the server list.

   This command restarts the service discovery server. The source management server begins to send any topology data changes immediately.

5. Type the following command to start the initial synchronization of topology data:

   `/opt/OV/bin/OpC/startInitialSync.sh`

How to Migrate from Scheduled Synchronization on HPOM for UNIX or Linux Systems

This section describes how to migrate from scheduled synchronization on HPOM for UNIX or Linux management servers. For further details, see the HPOM for UNIX or Linux documentation.

To migrate from scheduled synchronization, complete the following steps on the HPOM for UNIX or Linux management server from which you want to receive topology information:

1. *Prerequisite:* Make sure that the minimum patch level for the HPOM for Windows management server is installed:

   - HP-UX: Patch PHSS_42736 or superseding patch.

   - Linux: Patch OML_00050 or superseding patch.

   - Solaris: Patch ITOSOL_00772 or superseding patch.

2. Clear the agent repository cache on the management server using the following command:

   ```
   /opt/OV/bin/ovagtrep -clearall
   ```

3. Remove the service auto-discovery policies from the management server node using the following command:

   ```
   /opt/OV/bin/ovpolicy -remove DiscoverOM
   ```

   ```
   /opt/OV/bin/ovpolicy -remove DiscoverOMTypes
   ```

4. Deassign the service auto-discovery policies from the management server node using the following command:

   ```
   /opt/OV/bin/OpC/utils/opcnode -deassign_pol node_name=<management_
   server> net_type=NETWORK_IP pol_name=DiscoverOMTypes
   pol_type=svcdisc
   ```

   ```
   /opt/OV/bin/OpC/utils/opcnode -deassign_pol node_name=<management_
   server> net_type=NETWORK_IP pol_name=DiscoverOM
   pol_type=svcdisc
   ```

   ```
   /opt/OV/bin/OpC/opcragt -dist <management_server>
   ```

   Replace *<management_server>* with the name of the management server.

5. Type the following command to enable topology synchronization:

   ```
   /opt/OV/contrib/OpC/enableToposync.sh -online
   ```

   This command restarts the service discovery server. The source management server begins to send any topology data changes immediately.

6. Type the following command to start the initial synchronization of topology data:

   ```
   /opt/OV/bin/OpC/startInitialSync.sh
   ```

Basic Topology Synchronization

> **Note:** Use basic topology synchronization only with those versions of HPOM that do not support dynamic topology synchronization. For details of supported HPOM versions, see the *HP Business Service Management Release Notes*.

Basic topology synchronization is a command-line tool, running on demand on the BSM data processing server.

To create CIs in the RTSM (Run-time Service Model), based on mapping rules, and using the HPOM nodes, node groups and the HPOM service model as sources, basic topology synchronization does the following:

- Transfers the complete service model hierarchy (infrastructure-based service management data) from HPOM to Operations Management, using the HPOM web service.

- Converts the service management data into a format compatible with Operations Management.

- Uploads the service management data to the RTSM (Run-time Service Model).

- Updates the data in the RTSM on demand.

- Performs delta detection and deletes elements from the RTSM that are deleted in HPOM.

- Provides a mapping table for CI resolution.

Note: The definition of particular CI types is especially important when synchronizing the topologies of Operations Management and HPOM because the topology synchronization process cannot create new CI types. If the topology synchronization process attempts to map to a CI type that does not exist in the RTSM, the topology synchronization process aborts.

Basic Topology Synchronization Architecture

The following graphic provides an overview of the topology synchronization architecture.
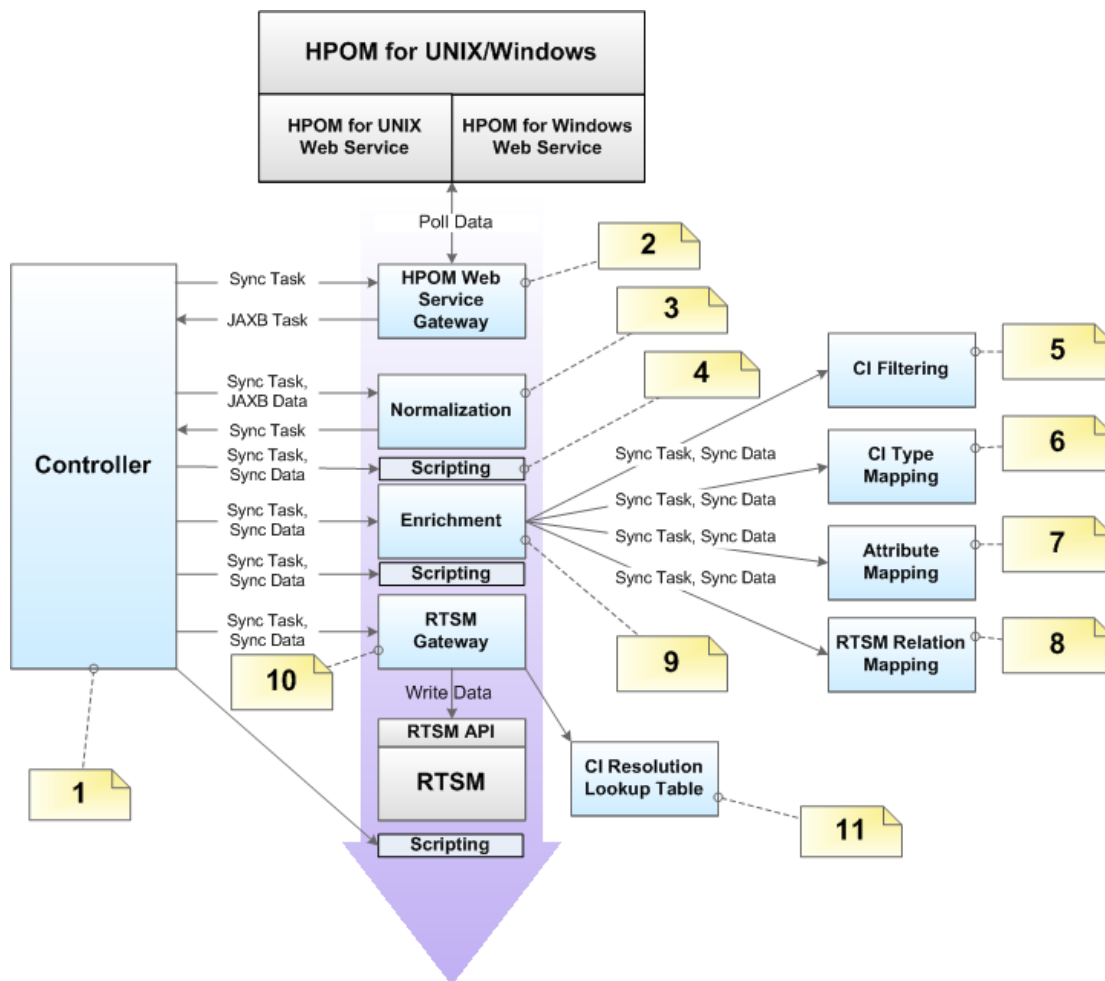
The numbers in the diagram refer to the following notes:

Reference	Refers to	Note
1	Controller	• Controls the synchronization process and passes data from one component to the other. • Provides an executable class to launch the synchronization from the command line.
2	HPOM Web Service Gateway	Send the (pull) request to the HPOM Web Service and return the JAXB objects.
3	Normalization	Convert the JAXB result to the internal synchronization data structure.
4	Scripting	Execute Groovy scripts to manipulate synchronization data.
5	CI Filtering	Use the mapping engine to specify, which CIs are part of the synchronized model, and which are not.

Reference	Refers to	Note
6	CI Type Mapping	Map Service Type Definitions (STDs) for RTSM (Run-time Service Model) types.
7	Attribute Mapping	Map untyped HPOM relations to typed RTSM relations.
8	RTSM Relation Mapping	• Create the relations for the RTSM. • Map the root container to the CI to be able to create the CI in the RTSM according to the model
9	Enrichment	Control the whole enrichment process and return the modified sync data.
10	RTSM Gateway	Write the synchronization data into the RTSM.
11	CI Resolution Lookup Table	For each synchronized CI, update a mapping table for the CI Resolution component that maps the HPOM Service ID to the RTSM CI ID.

Running Basic Topology Synchronization

You start basic topology synchronization by running the `opr-startTopologySync` command-line tool, on demand, on the BSM data processing server.

The `opr-startTopologySync` tool has a `.bat` file extension for Windows systems, and a `.sh` file extension for Linux systems.

> **Note:** The binary file to use if you want to set up a topology synchronization task in the Windows scheduler is:
> *<HPBSM root directory>*/opr/bin/startTopologySync.bat

You can run the `opr-startTopologySync.bat` tool in two modes:

- Normal mode
- Touch mode

Normal Mode

The normal mode loads the complete service model and synchronizes all configured data from *one*HPOM server (configured in Infrastructure Settings) to the RTSM (Run-time Service Model). The normal mode also performs delta detection and deletes elements from the RTSM that have been deleted in HPOM.

To run the `opr-startTopologySync.bat` tool in normal mode, enter the following command:

Windows: *<HPBSM root directory>*/bin/opr-startTopologySync.bat

Linux: *<HPBSM root directory>*/bin/opr-startTopologySync.sh

Touch Mode

The touch mode prevents aging in the RTSM by touching all elements from the previous synchronization. In touch mode, no new CIs are created in the RTSM, and no CIs are deleted.

To run the `opr-startTopologySync.bat` tool in touch mode, enter the following command:

Windows: `<HPBSM root directory>/bin/opr-startTopologySync.bat –touch`

Linux: `<HPBSM root directory>/bin/opr-startTopologySync.sh –touch`

For more details about RTSM aging, refer to the HP Business Service Managementthe Modeling Guide.

Skip Services Option

There is a command-line option `–skipservices` to skip the loading of services from the HPOM web service. You may want to use this option if your HPOM service model is really large, and as a result may fail to load.

The following example shows the command you would enter to run the `opr-startTopologySync.bat` tool in normal mode with the option to skip the loading of HPOM services:

Windows: `<HPBSM root directory>/bin/opr-startTopologySync.bat –skipservices`

Linux: `<HPBSM root directory>/bin/opr-startTopologySync.sh –skipservices`

Comparing Basic and Dynamic Topology Synchronization

The following table provides a comparison between basic and dynamic topology synchronization.

Basic	Dynamic
Run from the command-line.	Runs automatically after the initial manual synchronization. The results are always available without manual intervention.
Receives topology data from a single HPOM server.	Able to receive topology data from multiple HPOM servers, and can forward data to other connected servers.
Schedule-based: runs at scheduled intervals only.	Asynchronous: runs asynchronously, when a change in topology is detected.
Updates are delayed until the next scheduled run.	Updates are written immediately to the RTSM (Run-time Service Model) after a change occurs in HPOM.

Always retrieves the complete topology data from the HPOM service model. Resource consumption is therefore relatively high.	Receives only changes (deltas) in topology data. This means that only the changes need to updated in the RTSM, therefore resource consumption is much lower than with basic topology synchronization.
Requires addditional port to access the web service.	Single port HTTPS communication.

Synchronization Packages and Mapping

Topology synchronization uses synchronization packages to create CIs in the RTSM (Run-time Service Model). Topology synchronization packages contain the mapping between one or more services, nodes, or node groups on the HPOM side to one or more CIs on the RTSM side.

A topology synchronization package consists of XML configuration files (see "Mapping Files" on page 197). These files are responsible for transforming HPOM services, nodes and node groups into CIs in the RTSM, and synchronizing those CIs with data from specified services, nodes, and node groups in HPOM.

There are two types of topology synchronization packages:

- Standard packages provided out-of-the-box as part of the installation package.

  For more information, see "Standard Out-of-the-box Synchronization Packages" on page 195.

- Additional, out-of-the-box packages that are aligned with a subset of HPOM SPIs and available content packs.

  For more information, see "Additional Out-of-the-box Synchronization Packages" on page 196.

You can also create your own topology synchronization packages. An example of how to configure topology synchronization and create your own synchronization package is provided in the section "Configuring Topology Synchronization: ACME Example" on page 198.

For an out-of-the-box installation, synchronization packages are automatically loaded from the file system to the RTSM. When you make new synchronization packages, or changes to existing packages, you need to run a command-line tool to upload the new or changed packages to the RTSM. For details, see "Uploading and Downloading Synchronization Packages" on page 192.

For more information about synchronization packages and mapping, see "Synchronization Packages" on page 194.

Scripting and Topology Data

Scripting enables you to perform additional processing and customizing during the synchronization process before the mapping and before and after the upload of topology data from HPOM to the RTSM (Run-time Service Model).

Groovy scripts are supported to manipulate the synchronization data during the synchronization process. Groovy scripts can be placed into a topology synchronization package. Scripting is required, for example, if you want to create two CIs out of one HPOM service, which is not possible using the XML mapping files alone.

For more information, see "Scripting" on page 205.

CI Resolution Using a Mapping Table

Topology synchronization creates a mapping table for all CIs synchronized from HPOM. This mapping table can be used as a short-cut for CI resolution. The service ID from the HPOM service is searched in the table, and mapped to a CI in the RTSM (Run-time Service Model). When use of the mapping table is enabled, the table is analyzed first before CI resolution is used. If the mapping table yields no match, CI resolution then continues with the mapping process, including Smart Message Mapping.

The use of this mapping table is enabled by default (the **Use Topology Sync Shortcut** setting in the CI Resolver settings is set to **true**).

You would typically enable use of the mapping table in situations where there is a direct, one-to-one relationship between a service in the HPOM service tree and a CI for that service in the RTSM.

There are situations in which you would not use the mapping table shortcut, for example, where the service tree structure and RTSM structure are quite different, and there is no longer a one-to-one relationship between a service and a CI for that service in the RTSM. There may be many CIs in the RTSM that provide information about the cause of a service failure, and CI resolution is the quickest, most reliable way to find the most appropriate CI for the service object.

If you do not want to use the mapping table, you can disable it in the CI Resolver settings in the BSM Infrastructure Settings Manager:

Infrastructure Settings > Applications > Operations Management > Operations Management - CI Resolver Settings > Use Topology Sync Shortcut

Topology Synchronization File Locations

Initial product installation copies topology synchronization files to the locations specified in this section on the local file system on the BSM data processing server.

Basic Topology Synchronization

You can find the topology synchronization files for basic topology synchronization in the following locations:

Binaries:

- *<HPBSM root directory>*`/opr/lib/opr-ts-*.jar`

- **Linux:**

  - *<HPBSM root directory>*`/bin/opr-startTopologySync.sh`

  - *<HPBSM root directory>*`/bin/opr-sdtool.sh`

    <HPBSM root directory>`/bin/opr-startTopologySync.sh`

- **Windows:**

  - *<HPBSM root directory>*`\bin\opr-sdtool.bat`

  - *<HPBSM root directory>*`\bin\opr-startTopologySync.bat`

Log Files:

<HPBSM root directory>`/log/opr-topologysync`

Log File configuration Files:

<HPBSM root directory>`/conf/core/Tools/log4j/opr-topologysync/opr-`
`topologysync.properties`

Topology Synchronization Packages:

<HPBSM root directory>`/conf/opr/topology-sync/sync-packages`

Schema Files:

<HPBSM root directory>`/conf/opr/topology-sync/schemas`

Dynamic Topology Synchronization

You can find the topology synchronization files for dynamic topology synchronization in the following locations:

Binaries:

Windows: *<HPBSM root directory>*`/bin/opr-sdtool.bat`

Linux: *<HPBSM root directory>*`/bin/opr-sdtool.sh`

<HPBSM root directory>`/opr/lib/opr-ts-*.jar`

<HPBSM root directory>`/opr/lib/OvSvcDiscServer.jar`

Log Files:

<HPBSM root directory>`/log/wde/opr-svcdiscserver.log`

`<OvDataDir>/log/OvSvcDiscServer.log`

Log File Configuration Files:

<HPBSM root directory>`/conf/core/Tools/log4j/wde/opr-`
`svcdiscserver.properties`

Topology Synchronization Packages:

<HPBSM root directory>`/conf/opr/topology-sync/sync-packages`

Schema Files:

<HPBSM root directory>`/conf/opr/topology-sync/schemas`

Topology Synchronization Settings

For the successful synchronization of Operations Management, HPOM make sure that the following settings are correctly configured:

- **HPOM Topology Synchronization Connection Settings**: (*basic topology synchronization only*)

Basic topology synchronization needs to read the topology data from the HP Operations Manager web service (WS) during synchronization. The settings to enable this are made in the HPOM Connection Settings.

You can access the connection settings here:

I**nfrastructure Settings > Applications > Operations ManagementOperations Management** - **HPOM Topology Synchronization Connection Settings**

For more information, see the Operations Management online help.

- **HPOM Topology Synchronization Settings**

  In the HPOM Topology Synchronization settings, for both basic and dynamic topology synchronization, you can:

  - Enable or disable dump data

  - Enable or disable the use of Groovy Scripts

  - Specify which topology synchronization packages to use

  For basic topology synchronization only:

  You can also enable IP address resolution during synchronization for HPOM nodes that do not have any information regarding the IP address

  > **Note:** These settings are mandatory for the correct configuration of Operations Management and successful synchronization of the object topology in the environments monitored by Operations Management and HPOM.

  You can access the HPOM Topology Synchronization settings here:

  I**nfrastructure Settings > Applications > Operations ManagementOperations Management** - **HPOM Topology Synchronization Settings**

Uploading and Downloading Synchronization Packages

For both basic and dynamic topology synchronization, the command-line tool `opr-sdtool` uploads new or changed synchronization packages from the file system to the database, or

downloads synchronization packages from the database to files.

Uploading Synchronization Packages

Topology synchronization always uses the synchronization package that is loaded in the database. As a consequence, if you made changes to the synchronization package files, you must upload the synchronization package again to the database.

To upload new or changed synchronization packages to the database, run the following command:

Windows: `opr-sdtool.bat –uploadpackage` *`<path of synchronization package>`*

Linux: `opr-sdtool.sh –uploadpackage` *`<path of synchronization package>`*

Downloading Synchronization Packages

You can also use the download option to ensure that the latest version of the synchronization packages is available for editing on each BSM server when BSM is installed in a distributed environment.

To download synchronization packages from the database, run the following command:

Windows: `opr-sdtool.bat -downloadpackage [<`*`syncPackageName`*`>] [-path [<`*`downloadPath`*`>]]`

Linux: `opr-sdtool.sh -downloadpackage [<`*`syncPackageName`*`>] [-path [<`*`downloadPath`*`>]]`

If you do not specify the name of the synchronization package, the tool downloads all packages that are currently in the database.

The tool by default downloads the packages to *`<HPBSM root directory>`*`/conf/opr/topology-sync/sync-packages` unless you specify a download path.

Uploading HPOM SPI Service Type Definitions to the Database

HPOM Smart Plug-ins (SPI) Service Type Definitions are used to process received services from agents.

For both basic and dynamic topology synchronization, the command-line tool `opr-sdtool` is provided to upload new or changed service type definitions from HPOM Smart Plug-ins (SPIs) to the database, using the `–uploadstd` command-line option.

To upload new or changed service type definitions from HPOM SPIs to the database, run the following command:

Windows: `opr-sdtool.bat –uploadstd` *`<path of MofFile>`*

Linux: `opr-sdtool.sh –uploadstd` *`<path of MofFile>`*

Synchronization Packages

This section describes the topology synchronization packages that contain the rules for mapping HPOM topology data to CIs in the RTSM (Run-time Service Model).

The chapter is structured as follows:

- "Synchronization Packages Overview" below
- "Standard Out-of-the-box Synchronization Packages" on the next page
- "Additional Out-of-the-box Synchronization Packages" on page 196
- "Package Descriptor File: package.xml" on page 196
- "Mapping Files" on page 197
- "Configuring Topology Synchronization: ACME Example" on page 198
- "Customizing Synchronization and Scripting" on page 203
- "Synchronization Package Locations" on page 204

Synchronization Packages Overview

Topology synchronization packages contain the mapping between one or more service models, nodes, or node groups on the HPOM side to one or more CIs on the RTSM (Run-time Service Model) side.

A topology synchronization package contains a set of XML configuration files that define the mapping rules (context, CI type, attributes, and so on) during topology synchronization. The configuration files are used to:

- Transform topology data (for example, HPOM services, nodes, and node groups) into CIs in the RTSM.
- Synchronize CIs in the RTSM with topology data from HPOM.

A topology synchronization package must include the package descriptor file (`package.xml`) to define the synchronization package (see "Package Descriptor File: package.xml" on page 196).

Mapping files that can be part of a synchronization package are:

- `contextmapping.xml`
- `typemapping.xml`
- `attributemapping.xml`
- `relationmapping.xml`

For more information about the XML configuration files, see "Mapping Files" on page 197.

For basic information on mapping, see "Mapping Engine and Syntax" on page 220.

Groovy scripts can also be placed into a topology synchronization package to manipulate the synchronization data during the synchronization process, or to carry out post-synchronization

activities, for example, for auditing purposes. You can include the following Groovy scripts in a topology synchronization package:

- `preEnrichment.groovy`

- `preUpload.groovy`

- `postUpload.groovy`

For more information about Groovy scripts, see "Groovy Scripts" on page 205.

Standard Out-of-the-box Synchronization Packages

You can specify the content you want to update when synchronizing the topology between Operations Management and HPOM.

There are three out-of-the-box topology synchronization packages:

- `default`

  Contains basic type mappings for nodes, and basic attribute mappings for nodes and node groups.

  Does not create any CIs in the RTSM (Run-time Service Model).

  Should not be removed from the list of enabled packages.

- `operations-agent`

  In addition to creating the host CI itself, creates a CI instance of type `hp_operations_agent` for each HPOM managed node with an agent, and relates it to the host CI. Also creates CIs of type `omserver` and relates it to its host and to all the `hp_operations_agent` CIs.

- `nodegroups`

  In addition to creating the host CI itself, maps HPOM node groups to the RTSM (Run-time Service Model) CI type `ci_group`, creates instances of the CI type `ci_group`, and creates relations for the contained nodes. Also creates CIs of type `ipaddress` and `interface` and relates them to their host.

In the **Packages for Topology Sync** setting in HPOM Topology Synchronization settings, you can list the packages whose contents should be updated during the topology synchronization process:

Infrastructure Settings > Applications > Operations Management > Operations Management - HPOM Topology Synchronization Settings > Packages for Topology Sync

The entries in the list must be separated by a semicolon (;) as illustrated in the following example:

`default;nodegroups;operations-agent`

By default, packages are located in the following directory:

`<HPBSM root directory>/conf/opr/topology-sync/sync-packages`

Additional topology synchronization packages are provided in the content packs.

Additional Out-of-the-box Synchronization Packages

Additional topology synchronization packages are provided out-of-the-box in content packs. Content packs include the following:

- ActiveDirectory

- Exchange

- MS SQL Server

- Oracle

- J2EE (includes WebSphere and WebLogic content)

- Infrastructure (includes UNIX and Windows operating systems, Virtualization Systems, and Cluster Systems)

These additional topology synchronization packages are not enabled by default. To enable them:

1. Load the content pack(s) you wish to use.

2. Enable the synchronization packages manually in the Infrastructure Settings:

   Infrastructure Settings > Applications > Operations Management > Operations Management - HPOM Topology Synchronization Settings > Packages for Topology Sync

Topology synchronization packages are written to the following directory:

<HPBSM root directory>`/conf/opr/topology-sync/sync-packages`

For example, the Oracle content pack uses the package (and directory) name `HPOprOra`. This is the name you enter in the list if you want the mapping rules to be executed during topology synchronization. If we add the Oracle package to the list of standard out-of-the-box packages we had in the example in the section "Standard Out-of-the-box Synchronization Packages" on the previous page), the list would look like this:

`default;nodegroups;operations-agent;HPOprOra`

> **Note:** If you are adding custom packages, note that the package name is the same as the name of the directory in which the package is located. Be aware that when a synchronization package is removed, CIs added to the RTSM (Run-time Service Model) by previous runs of that synchronization package, and that have not been reconciled with other CIs, are also removed.

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:

  <HPBSM root directory>`/con/opr/topology-sync/sync-packages`

- `<Description>` Description of the package.

- `<Priority>` Priority level of the package.

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

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

Mapping Files

The following mapping files can be included in a topology synchronization packages.

Context Mapping (Filtering): contextmapping.xml

You can determine which elements of an HPOM service tree you want to include in the topology synchronization for mapping in the RTSM (Run-time Service Model) by configuring the filtering file, `contextmapping.xml`. Filtering involves assigning a context to those CIs you want to synchronize. Configuring the context enables you to apply mapping rules selectively to CIs of the same context.

For example, specified HPOM services are tagged, and all subsequent mapping rules contained in other configuration files are applied to those tagged services. A service that has no context assigned is not included for synchronization.

Type Mapping: typemapping.xml

The type mapping file `typemapping.xml` defines the mapping from a service in HPOM based on their attributes to the type of a CI in the RTSM.

Attribute Mapping: attributemapping.xml

The attribute mapping file `attributemapping.xml` defines the mapping between the attributes of a service in HPOM 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.

Relation Mapping: relationmapping.xml

Using the relation mapping file `relationmapping.xml`, you can define the CI relationships created in the RTSM between specified HPOM services.

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

Configuring Topology Synchronization: ACME Example

This section provides a walk-through of how to configure topology synchronization, using the fictitious "ACME" content area as an example.

The following graphic shows a service tree from HPOM discovery.
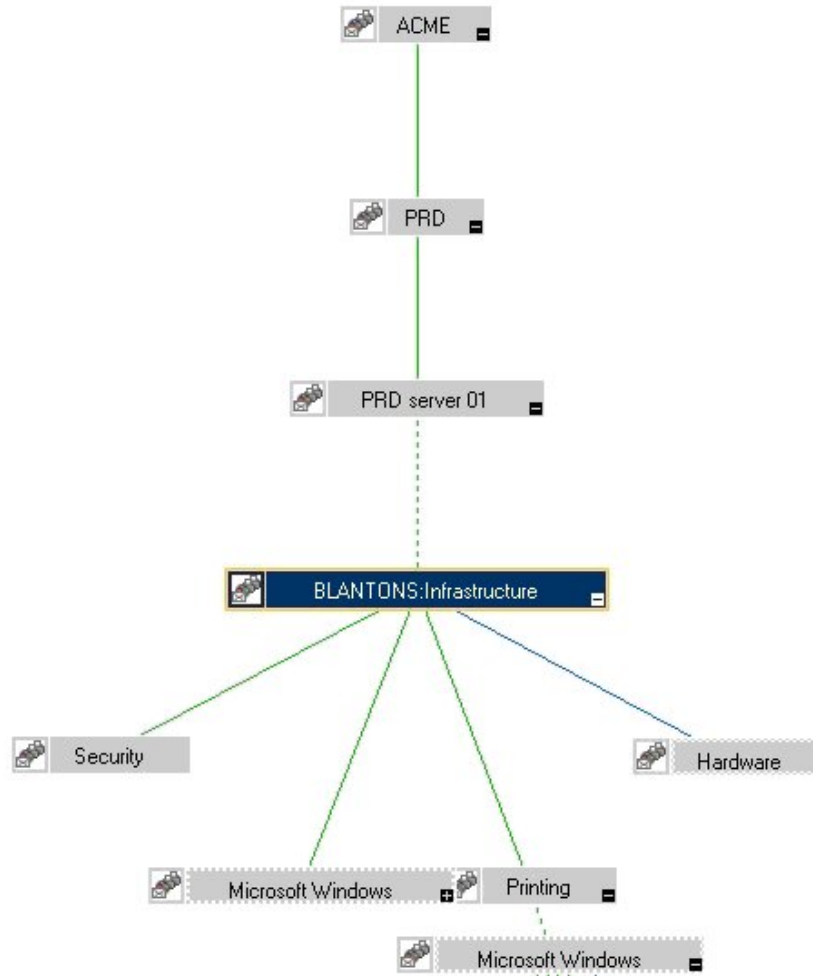
Configure Package Descriptor File: package.xml

Configure the file `package.xml` to define the name and provide a description of your topology synchronization package, together with a priority level. For the ACME topology synchronization package, the `package.xml` file looks like this:

```
<Package>
   <Name>ACME</Name>
   <Description>Service to RTSM Mapping for ACME
Landscape.</Description>
```

```
        <Priority>5</Priority>
</Package>
```

Configure Context Mapping (Filtering) File: contextmapping.xml

Configure the file `contextmapping.xml` to tag which elements included in topology data you want to include in the topology synchronization for mapping in the RTSM (Run-time Service Model). The mapping rules contained in other configuration files are applied to the tagged elements.

The following graphic represents the view for ACME in the RSTM.
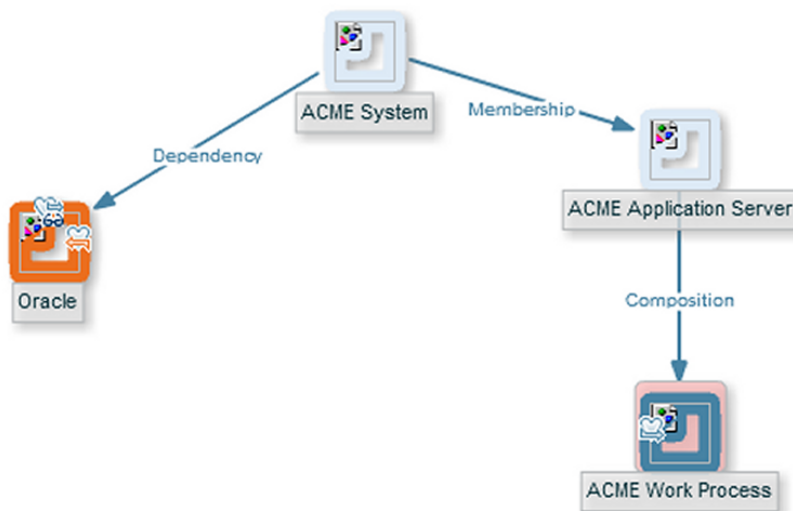
An example configuration of `contextmapping.xml` follows, where a context called `ACME_Landscape` is assigned to those HPOM service elements, of type `ACME_System` and `ACME_Application_Server`, for which you want to create CIs in the RTSM:

```
<?xml version="1.0" encoding="UTF-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd">
<!-- CONFIGURE THE CIs THAT DEFINE THE CONTEXT FOR THE MAPPING -->
  <Rules>
    <Rule name="Filter ACME Items">
      <Condition>
      <!-- Select all Service Elements of interest
         further refinements will be made later -->
        <Or>
            <Equals>
              <OMType />
               <Value>ACME_System</Value>
            </Equals>
            <Equals>
              <OMType />
               <Value>ACME_Application_Server</Value>
            </Equals>
```

```
        </Or>
      </Condition>
      <MapTo>
        <Context>ACME_Landscape</Context>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

Configure Type Mapping File: typemapping.xml

Configure the type mapping file `typemapping.xml` to define the mapping between the service type definition of a service in HPOM and the type of a CI in the RTSM (Run-time Service Model).

For the ACME example, the type mapping is defined in the following table.

HPOM Service Type	CI Type (CI Name)
ACME_System	acme_system
ACME_Application_Server	acme_appserver

Here is an example configuration of the type mapping file `typemapping.xml` for the ACME synchronization package, using the context `ACME_Landscape`. HPOM service elements of type `ACME_System` are mapped to CIs of type `acme_system` in the RTSM. HPOM service elements of type `ACME_Application_Server` are mapped to CIs of type `acme_appserver` in the RTSM.

```
<?xml version="1.0" encoding="UTF-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd">
  <Rules Context="ACME_Landscape">
    <Rule name="Map ACME System">
      <Condition>
        <Equals>
          <OMType />
          <Value>ACME_System</Value>
        </Equals>
      </Condition>
      <MapTo>
        <CMDBType>
          <Value>acme_system</Value>
        </CMDBType>
      </MapTo>
    </Rule>
    <Rule name="Map ACME Application Server">
      <Condition>
        <Equals>
          <OMType />
          <Value>ACME_Application_Server</Value>
        </Equals>
```

```
            </Condition>
            <MapTo>
                <CMDBType>
                    <Value>acme_appserver</Value>
                </CMDBType>
            </MapTo>
        </Rule>
    </Rules>
</Mapping>
```

Configure Attribute Mapping File: attributemapping.xml

Configure the attribute mapping file `attributemapping.xml` to define the mapping between the attributes of a service in HPOM and the attributes of a CI in the RTSM (Run-time Service Model).

For the ACME example, the following table shows which HPOM service attributes are mapped to which CI attributes in the RTSM.

Service Type	Service Attribute Name	CI Type	CI Attribute Name
ACME_System	Caption	ALL	display_label
ACME_Application_Server	OMId	ALL	data_name

Here is an excerpt of the corresponding configuration of the attribute mapping file `attributemapping.xml` for the ACME synchronization package. This shows two attribute mappings:

- For the HPOM service elements of type `ACME_system`, the HPOM service attribute `Caption` is mapped to the CI attribute `display_label` in the RTSM.

- For the HPOM service elements of type `ACME_Application_Server`, the HPOM service attribute `OMId` is mapped to the CI attribute `data_name` in the RTSM.

```
<?xml version="1.0" encoding="UTF-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd">
  <Rules Context="ACME_Landscape">
      <Rule name="Map ACME System attributes">
          <Condition>
              <Equals>
                  <OMType />
                  <Value>ACME_System</Value>
              </Equals>
          </Condition>
          <MapTo>
              <CMDBAttribute>
                  <Name>display_label</Name>
                  <SetValue>
                      <Caption />
                  </SetValue>
              </CMDBAttribute>
```

```
            </MapTo>
        </Rule>
        <Rule name="Map ACME Application Server attributes">
            <Condition>
                <Equals>
                    <OMType />
                    <Value>ACME_Application_Server</Value>
                </Equals>
            </Condition>
            <MapTo>
                <CMDBAttribute>
                    <Name>data_name</Name>
                    <SetValue>
                        <OMId />
                    </SetValue>
                </CMDBAttribute>
            </MapTo>
        </Rule>
    </Rules>
</Mapping>
```

Configure Relation Mapping File: relationmapping.xml

Configure the relation mapping file `relationmapping.xml` to define the CI relationships created in the RTSM (Run-time Service Model) between specified HPOM services.

Here is an example configuration of the relation mapping file `relationmapping.xml` for the ACME synchronization package. This shows the creation of the following relations:

- The `root_container` CI attribute of CIs with the HPOM service type `ACME_Application_Server` is set to the host. Additionally, a `container_f` relation is created implicitly between the host and the CI.

- A composition relation `container_f` between HPOM service elements of type `ACME_System` and `ACME_Application_Server`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd">
  <Rules Context="ACME_Landscape">
      <Rule name="Create relation ACME Application Server to node">
          <Condition>
              <StartsWith>
                  <OMType />
                  <Value>ACME_Application_Server</Value>
              </StartsWith>
          </Condition>
          <MapTo>
              <RootContainer>
                  <DependencyCI relationType="hosted_on">
                      <True />
```

```
                         </DependencyCI>
                    </RootContainer>
               </MapTo>
          </Rule>
          <Rule name="Create relation between ACME Application Server and
     ACME System">
               <Condition>
                    <And>
                         <Equals>
                              <OMType />
                              <Value>ACME_Application_Server</Value>
                         </Equals>
                         <Equals>
                              <AncestorCI relationType="container_f">
                                   <Equals>
                                        <OMType/>
                                        <Value>ACME_System</Value>
                                   </Equals>
                              </AncestorCI>
                                <ParentCI/>
                         </Equals>
                    </And>
               </Condition>
               <MapTo>
                    <RelationFrom>
                         <From>
                           <AncestorCI relationType="container_f">
                                   <Equals>
                                        <OMType/>
                                        <Value>ACME_System</Value>
                                   </Equals>
                              </AncestorCI>
                         </From>
                         <Type>member</Type>
                    </RelationFrom>
               </MapTo>
          </Rule>
     </Rules>
</Mapping>
```

Customizing Synchronization and Scripting

Scripting enables you to perform additional processing and customization during the synchronization process before the mapping and before and after the upload of topology data to the RTSM (Run-time Service Model). One pre-mapping, one pre-upload, and one post-upload script can be associated with each synchronization package.

For details, see "Scripting" on page 205.

Synchronization Package Locations

The `sync-packages` directory contains dedicated subdirectories for each synchronization package. It is recommended but not essential that you use directory names that match the synchronization package name.

Synchronization packages are deployed by placing them into the following directory:

`<HPBSM root directory>/conf/opr/topology-sync/sync-packages/<SyncPackageName>`

Scripting

Scripting enables you to perform additional processing and customizing during the synchronization process:

- Pre-mapping scripts are executed before the mapping rules are applied.

- Pre-upload scripts are executed after mapping, but before the upload of data to the RTSM (Run-time Service Model).

- Post-upload scripts are executed after the upload of data to the RTSM.

One script of each type can be associated with each synchronization package. These optional script files are located in the associated synchronization package directory. For details of synchronization package locations, see "Synchronization Package Locations" on the previous page.

Associating script files with synchronization packages simplifies the distribution of scripts and enables script development to be handled independently of the working environment. The execution of synchronization scripts follows the settings of the synchronization packages:

- Only scripts in active synchronization packages are executed.

- Scripts are executed in the order of the priority of the synchronization packages.

> **Caution:** Script execution is potentially insecure. In particular, the use of `scriptInterface.exec(...)` commands can cause damage to an installation. To enhance security, script access for editing is allowed on the file system level only. This makes sure that only users with log-on credentials to the OMi host can edit scripts. This protects the scripts by the log-on security of the OMihost.

Groovy Scripts

Groovy scripting is supported. Groovy is a high level, object-oriented scripting language for the Java platform, which complies down to Java bytecode.

Groovy is aimed at Java developers, and is tightly integrated with the Java platform. It provides you with a similarly powerful and concise coding syntax to that provided by languages such as Python or Ruby, while enabling you to stay on the Java Virtual Machine (JVM). Java beans are supported, and Java and Groovy classes are interchangeable inside the JVM. As Groovy integrates well with Java code and libraries, and enables you to reuse the semantics of Java, and all J2SE and J2EE APIs, you do not have to learn, implement and maintain new semantics and APIs.

For more information about Groovy and documentation describing the Groovy language, visit:

http://groovy.codehaus.org

There are three Groovy scripts that can be placed into a topology synchronization package, and these are identified using fixed names within synchronization package directories. Each script runs at a defined point in the synchronization process:

- `preEnrichment.groovy` — script to be executed before mapping

  The `preEnrichment.groovy` script is executed before starting the topology synchronization's mapping process.

- `preUpload.groovy` — script to be executed before upload

  The `preUpload.groovy` script is executed after the mapping process but before writing any data to the RTSM (Run-time Service Model) (for example, to create additional CIs or add extra details to existing CI instances).

- `postUpload.groovy` — script to be executed after upload

  The `postUpload.groovy` script is executed after saving the uploaded data in the RTSM, to modify data saved during the upload process (for example, for logging or auditing purposes).

The upload is performed between execution of the `preUpload.groovy` scripts and the `postUpload.groovy` scripts.

Enabling and Disabling Scripts

By default, the use of Groovy scripts is enabled (in the HPOM Topology Synchronization settings, the `Enable usage of Groovy scripts` value is set to `true`).

To help identify the cause of synchronization failure, you can disable scripting. If there is an error in a script, disabling scripting should make successful synchronization more likely.

To disable topology synchronization package script execution, change the `Enable usage of Groovy scripts` setting from `true` to `false` in the HPOM Topology Synchronization settings in the Infrastructure Settings manager:

Infrastructure Settings > Applications > Operations Management > Operations Management - HPOM Topology Synchronization > Enable usage of Groovy scripts

Groovy Script Location

The Groovy scripts must be located in the same directory as the topology-synchronization mapping rules:

<HPBSM root directory>`/conf/opr/topology-sync/sync-packages/`*<SyncPackageName>*

Script Variables

Each script has two predefined variables:

- `ScriptInterface`

  Object Type: `com.hp.opr.ts.scripting.ScriptInterface`

  Description: Enables access to CI information function calls to manipulate synchronization data and control the synchronization.

  The object type implements the following interface:

  `com.hp.opr.ts.interfaces.scripting.IScriptingInterface`

- `SyncData`

  Object Type: `com.hp.opr.ts.common.data.sync.SyncData`

  Description: Provides access to the data that is synchronized.

  The object type implements the following interface:

  `com.hp.opr.ts.interfaces.data.sync.ISyncData`

For more information about the interfaces and object types required for developing your own topology synchronization scripts, see the Java API Documentation document located in the following directory:

`<HPBSM root directory>/opr/api/doc/opr-ts-interfaces-javadoc.zip`

Scripts within a synchronization package share the same variable scope. That means variables assigned in `preEnrichment.groovy` can be later used in the corresponding `preUpload.groovy` and `postUpload.groovy`. Scripts from different synchronization packages do not share variables with the same name, which avoids name clashes and undesired side effects.

Handling Errors

Errors in scripts result in the generation of exceptions. The error handling is around each script invocation. By default, an exception in a script aborts the synchronization. This behavior can be changed by calling the command:

`scriptInterface.setAbortSyncOnError(boolean)`

When set to false, you can enforce a script failure using the method `abortSync("...")`. For example, your script checks conditions, and because of these forced failures, a synchronization cannot be completed.

The following table shows the relationship between synchronization status (successful or unsuccessful synchronization) and script behavior.

Status	Script behavior
Synchronization OK	Scripting completed without errors and without forced synchronization interruption within a script. Scripting completed without errors even if an exception is thrown, and `AbortSyncOnError` is set to false.
Synchronization failed	A script execution caused an exception or the script forced a failure because of a scripting condition using the `abortSync(String)` command.

Sample Script: preUpload.groovy

The following script is an extract of a sample preUpload.groovy script:

```
import com.hp.opr.ts.interfaces.data.ci.*  ;
import com.hp.opr.ts.common.data.ci.*  ;
import java.util.*;
```

```
import java.lang.String;


List resourceGroups = new LinkedList ();
List haMembers      = new LinkedList ();

// Get all HPOM services, hosts and node groups
for (ICi ci : syncData.getConfigurationItems()) {


  if (ci.getOmTypeId() == "Class_RG") {
// If type is "Class_RG", then create a CI of type IP for all entries
  // of the HPOM attribute ip address

    scriptInterface.logInfo ("add resource group");
    resourceGroups.add (ci);
  }



}



// Create ip-ci and relationship to the cluster package
for (ICi ipCi : resourceGroups) {
    HashMap hm = new HashMap();
// Get HPOM service-specific attributes
    hm = ipCi.getOmAttributes();



// Create CI for ip-address attribute
    ICi newCi = scriptInterface.createCi();
    newCi.setContext ("cluster");
        newCi.setCmdbAttribute ("ip_address", hm.get("ipaddr"));
        newCi.setCmdbAttribute ("ip_domain", "\${DefaultDomain}");
        newCi.setCmdbTypeId ("ip");
        scriptInterface.logInfo ("create relationship between two ip-
ci: "
        + hm.get("ipaddr") + " and cluster package " );
// Create the "contained" relationship between the cluster package
// and ip
    scriptInterface.createCmdbRelation(ipCi, newCi, "contained");


    }
}
```

Testing and Troubleshooting

This chapter contains information on:

- "Validating XML Configuration Files" below

- "Dumping Synchronization Data" on page 211

- "Writing Rules" on page 214

- "Log Level Configuration" on page 215

- "Troubleshooting, Common Issues, and Tips" on page 217

Validating XML Configuration 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. You can use Eclipse or another editor of your choice that is capable of validating an XML file against a schema.

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.

XSD Files

The schema files are stored in the following directory:

`<HPBSM root directory>/conf/opr/topology-sync/schemas`

The files are:

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

`containmentrelations.xsd`
    Validates the `containmentrelations.xml` file.

`datadump.xsd`
    Validates synchronization data files that are created through enabling data dumps or used as input for the enrichment simulator.

`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`

`nodetypes.xsd`

Validates the node type mapping file `nodetypes.xml` file in each synchronization package.

Validating Files Automatically

Each configuration 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.

Validating Files Manually

With a modern XML editor, you can validate a file against a schema. Eclipse, for example, can validate an XML file against a schema, if the top level element of the document contains a reference to an XSD file. To enable validation, add the following attributes to the top level element of an XML file:

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="<path or URL to schema file>"
```

Replace *<path or URL to schema file>* with the respective path or URL to the schema file against which you want to validate. For example, for a `contextmapping.xml` file, add the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="<HPBSM root
directory>/conf/opr/topology-sync/schemas/mapping.xsd">
...
</Mapping>
```

After you have added the reference, the Eclipse editor validates the file and suggests valid elements when pressing **CTRL**+**SPACE** during editing. See the following figure for an example.
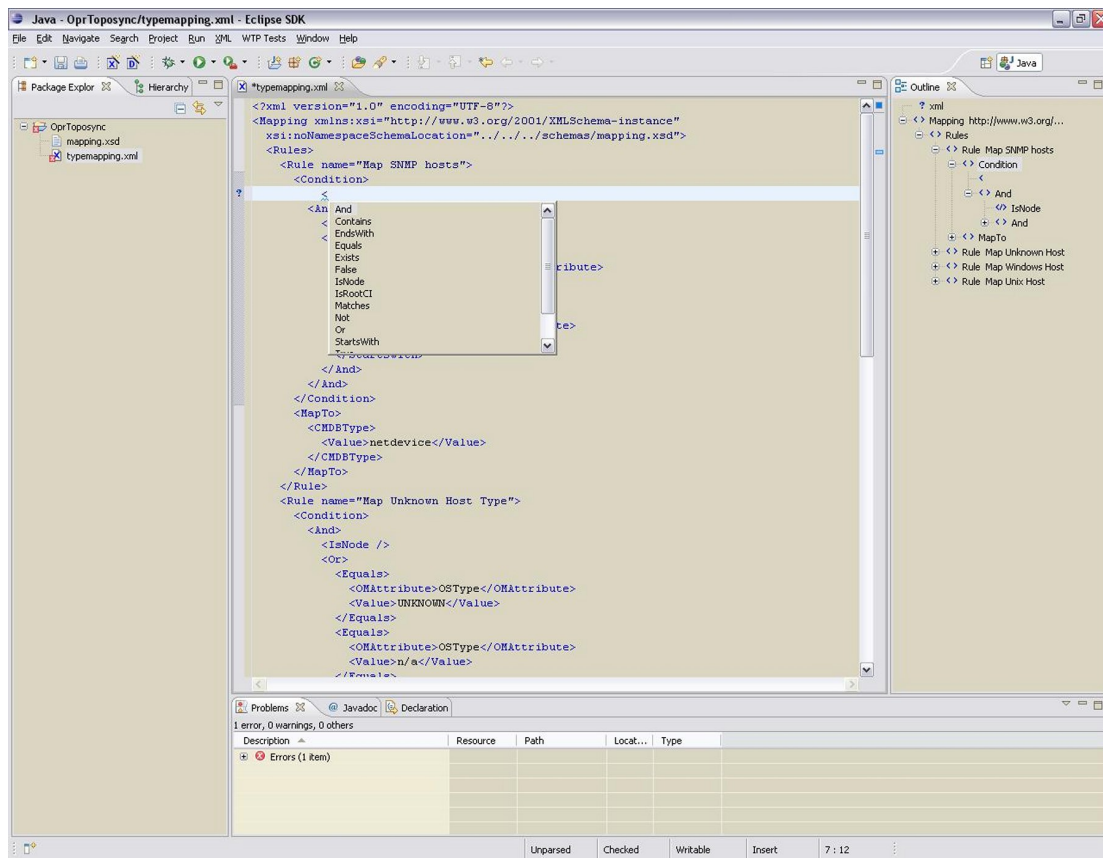
Note: You may have to reopen the XML file after you have added the XSD reference to the XML file before Eclipse starts to validate it and provides suggestions.

Dumping Synchronization Data

You can use a dump of the synchronization data to:

- Troubleshoot mapping rules to discover incorrect mappings.

- Compare the data sent to the data in the RTSM (Run-time Service Model), and the data changed and added during the mapping.

- Create a dump file to check XPath expressions of rules.

Creating a Synchronization Data Dump

A synchronization data dump contains the synchronized topology data in XML files using the data format as exposed to the XPath Expression matching in the mapping rules.

There are two separate dumps:

- The first is recorded following CI data normalization.

- The second is recorded following the processing of the mapping rules.

To activate the creation of synchronization data dumps:

1. Navigate to the HPOM Topology Synchronization settings in the Infrastructure Settings
   Manager:

   **Infrastructure Settings > Applications > Operations Management > Operations
   Management - HPOM Topology Synchronization Settings > Dump data**

2. Change the value of **Dump data** to **true**.

3. Run the Topology Sync tool with the following command:

   Windows: <*HPBSM root directory*>/bin/opr-startTopologySync.bat

   Linux: <*HPBSM root directory*>/bin/opr-startTopologySync.sh

Data Dump Example

Here is an example extract from a data dump after mapping has been performed:

```
<CI>
  <OMId>Root</OMId>
  <OMType />
  <Caption>Root</Caption>
  <Node>false</Node>
  <Service>false</Service>
  <OMAttributes />
  <CMDBId />
  <CMDBAttributes />
  <CMDBType />
  <RootContainerId />
  <Children>
    <RelationType>container_f</RelationType>
    <CI>
      <Context>operations-agent</Context>
      <OMId>03a2f7b2-ec88-7539-0532-c5b07da188dd</OMId>
      <OMType>agent</OMType>
      <Caption>Operations-agent on met</Caption>
      <Node>false</Node>
      <Service>false</Service>
      <OMAttributes>
        <AgentId>03a2f7b2-ec88-7539-0532-c5b07da188dd</AgentId>
        <Name>met.deu.hp.com</Name>
      </OMAttributes>
      <CMDBId />
      <CMDBAttributes>
        <data_name>03a2f7b2-ec88-7539-0532-c5b07da188dd</data_name>
      </CMDBAttributes>
      <CMDBType>hp_operations_agent</CMDBType>
      <RootContainerId>{8BB8864B-CEC9-4B26-BD4C-41F2C97C108E}
</RootContainerId>
      <Dependencies>
        <RelationType>hosted_on</RelationType>
        <CI>
```

```
          <Context>VISPI</Context>
          <Context>nodegroups</Context>
          <OMId>{8BB8864B-CEC9-4B26-BD4C-41F2C97C108E}</OMId>
          <OMType>node</OMType>
          <Caption>met</Caption>
          <Node>true</Node>
          <Service>false</Service>
          <NodeGroupList>
            <NodeGroupID>OpenView_Windows2000</NodeGroupID>
            <NodeGroupID>Root_Nodes</NodeGroupID>
          </NodeGroupList>
          <MACAddressList />
          <OMAttributes>
            <AgentId>03a2f7b2-ec88-7539-0532-c5b07da188dd</AgentId>
            <CommType>HTTPS</CommType>
            <DiscoveryDomain>${DefaultDomain}</DiscoveryDomain>
            <Domain>deu.hp.com</Domain>
            <Name>met.deu.hp.com</Name>
            <OSType>Windows_32</OSType>
            <OSVersion>2000 (5.0)</OSVersion>
            <SystemType>x86/x64 Compatible</SystemType>
            <VirtualNodeType>0</VirtualNodeType>
          </OMAttributes>
          <CMDBId />
          <CMDBAttributes>
            <host_dnsname>met.deu.hp.com</host_dnsname>
            <host_hostname>met</host_hostname>
            <host_key>met.deu.hp.com</host_key>
            <host_os>2000 (5.0)</host_os>
          </CMDBAttributes>
          <CMDBType>nt</CMDBType>
          <RootContainerId />
        </CI>
      </Dependencies>
    </CI>
  </Children>
</CI>
```

Viewing a Synchronization Data Dump

To view synchronization data dumps, navigate to the directory:

<HPBSM root directory>/opr/tmp/datadump

The directory contains the following subdirectories:

- pre-enrichment

  Contains the synchronization data after the CI data structure has been normalized. The data
  reflects what has been loaded from HPOM to the RTSM (Run-time Service Model).

- `post-enrichment`

  Contains the synchronization data after the mapping rules have been executed on the normalized data.

- `ws-data`

  Contains raw data which was read from the HPOM web service. For each HPOM node, node group, and service, there is an XML file called `Caption_OMId.xml`.

  Only in the case where writing to the RTSM failed, the XML file is written to the `post-ucmbd` directory.

Validating Mapping Rules

To validate mapping rules, complete the following steps:

1. Compare file differences.

   Using a file comparison tool of your choice you can easily see what has been changed during enrichment.

2. Validate XPath expressions.

   You can validate XPath Expressions that are used in mapping rules by loading the normalized synchronization data dumps into an XML editor that supports XPath queries.

   > **Note:** An XML document must have a single root element (`<ci>`) in the data dumps. When running XPath queries in the mapping rules, this root element does not exist. For testing with dump files, when you create absolute expressions, prepend the expression `/ci` to your test expression.

Writing Rules

This section contains a set of guidelines for writing rules.

Simplifying 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. See "Validating XML Configuration Files" on page 209 for more information.

Avoiding 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).

Matching 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.

Avoiding 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.

Log Level Configuration

Topology synchronization logs details of the synchronization process in log files. You can change the level of detail for debugging purposes.

Service Discovery Server Log Level Configuration

Dynamic topology synchronization only. The service discovery server supports the following log levels:

- Log level 1 logs errors only.

- Log level 3 logs errors and information (including raw data received from the agent).

- Log level 10 logs tracing information for debugging purposes, for example method parameters.

To change the log level of the service discovery server:

1. In a command prompt, run the following command:

   `ovconfchg -edit`

   You can see the log in a Notepad window.

2. Edit the file by adding `LOG_LEVEL=10` to the `[om.svcdiscsserver]` namespace.

The service discovery server generates the following log file:

Windows: `%OvShareDir%\server\log\OvSvcDiscServer.log`

Linux: `/var/opt/OV/shared/server/log/OvSvcDiscServer.log`

Mapping Log Level Configuration

To change the log level of the mapping engine, follow these steps:

1. Open the following file in a text editor:

   <HPBSM root directory>`/conf/core/Tools/log4j/wde/opr-svcdiscserver.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:

<HPBSM root directory>`/log/wde/opr-svcdiscserver.log`

Troubleshooting, Common Issues, and Tips

The log files specified under "Topology Synchronization File Locations" on page 190 are a good starting point for troubleshooting.

The most common issues are listed in the following table. The issues apply to topology synchronization generally, unless otherwise specified.

Symptom	Cause	Solution
Topology synchronization fails.	Required patches for HPOM for Windows were not installed. See the BSM Readme for details, including information about any required agent hotfixes or patches.	Operations Manager for Windows: Install Patches OMW_00138 or superseding and OMW_00123 for HPOM 8.1x for Windows. Install Patches OMW_00139 or superseding and OMW_00124 for HPOM 9.00 for Windows. See the *BSM 9.10 Release Notes* for details.
		Operations Manager for UNIX or Linux: Install Patch PHSS_42736 or superseding for HPOM 9.10 for HP-UX. Install Patch OML_00050 or superseding for HPOM 9.10 for Linux. Install Patch ITOSOL_00772 or superseding for HPOM 9.10 for Solaris.
Basic topology synchronization fails.	Port for the web service was not configured correctly.	Make sure the port for the web service is correctly configured.
	User name / password are wrong.	Format for HPOM for Windows: DOMAIN\Username. User must have at least PowerUser rights and must be a member of HP-OVE-Admins.
Dynamic topology synchronization fails.	Synchronization package was changed on disk, but not uploaded to the database.	Run the `opr-sdtool` command-line tool to upload changes to synchronization packages to the database (see "Uploading and Downloading Synchronization Packages" on page 192).

Symptom	Cause	Solution
Result of *dynamic* topology synchronization is incomplete or empty.	Discovery policies (`DiscoverOMTypes` and `DiscoverOM`) are deployed prior to configuring the Operations Management instance as a target server in HPOM.	On the HPOM management server, run the command `ovagtrep -publish`. This command resends all topology data to the HPOM or Operations Management instances.
All of a sudden, no more node CIs are created and the synchronization fails.	The `default` synchronization package was removed from the Topology Synchronization settings.	Check if the `default` synchronization package was removed from the Topology Synchronization settings. The `default` package must always be present in the semicolon-separated list.
Warnings in the log file.	Model-related issues.	No immediate action required, however topology synchronization performance can be affected.
You created your own synchronization package but you only get a cryptic RTSM (Run-time Service Model) exception in the log file.	Mapping-related issues.	Enable the data dump option and check if the file in the `<HPBSM root directory>/opr/tmp/datadump/post-enrichment` directory contains all expected attributes for the CIs of your synchronization package.

Limitations

This section describes some known limitations relating to topology synchronization.

Delta Detection Limitations

If an attribute of a service or node in HPOM changes, and that attribute or node is mapped to a key attribute in the RTSM (Run-time Service Model), the delta detection does not delete and replace the old RTSM (Run-time Service Model) CI instance, but generates an additional new instance.

Here is an example to illustrate this. Consider a managed node in HPOM for Windows that has an agent ID of `aaaaa-bbbb-cccc-dddd`. This node maps to a host CI in the RTSM (Run-time Service Model) and an agent CI with the key `aaaaa-bbbb-cccc-dddd`.

In HPOM, a change is made to the agent ID, so that the agent ID is now `aaaaa-bbbb-cccc-eeee`. A new agent CI with the key attribute `aaaaa-bbbb-cccc-eeee` is created, but the old one is not deleted. So there are now two RTSM (Run-time Service Model) instances relating to the same (changed) managed node.

Workaround: To overcome this limitation, you need to manually delete the old RTSM (Run-time Service Model) instance.

Topology Synchronization Limitation

Event changes from the HPOM discovery server are registered with the WMI Listener, and forwarded through the WMI to the RTSM (Run-time Service Model). Under certain circumstances, for example, due to high loads on the WMI, it could happen that not all events arrive in Operations Management, and so are not reflected in the RTSM (Run-time Service Model). As a result, this could lead to a temporary discrepancy between the status in HPOM and the RTSM (Run-time Service Model). This discrepancy will be reconciled the next time the tool `startInitialSync.bat` runs.

Mapping Engine and Syntax

Mapping is the mechanism used to map services, attributes, or nodes within HPOM to CIs in the RTSM (Run-time Service Model). 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 240

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 222

- "Operand Elements" on page 225

- "Mapping Elements" on page 232

- "Filtering" on page 232

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 232 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 HPOM service is `testtype`.

```
<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.

```
<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://java.sun.com/j2se/1.5.0/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. This operator can be used only for dynamic topology synchronization, as basic topology synchronization uses a different mechanism to delete CIs. Basic topology synchronization ignores the IsDeletionCI operator.

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 (Run-time Service Model). 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://java.sun.com/j2se/1.5.0/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 240.

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 240.

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.

Condition Examples

Check if the type of the current HPOM service is `testtype`.

```
<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.

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

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>
```

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>
```

All CIs that are assigned to the service type definition `exch_spi_std_server_role` and that are below a service with a service type definition `exch_spi_std_server` are assigned to the `exchange` context.

Type Mapping

The service mapping maps the HPOM service type definitions to their CMDB types.

Mapping

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

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.

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>
```

All CIs that have an OM Type `Exch2k7_ByServer` and that have the context `exchange` assigned are mapped to the CMDB Type `exchangeserver`.

```
<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>
```

All nodes that have an attribute `OSType` that starts with the string `Windows` are mapped to the CMDB type `nt`.

Attribute Mapping

The attribute mapping file `attributemapping.xml` defines the mapping between the attributes of a service in HPOM and the attributes of a CI in the RTSM (Run-time Service Model).

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 in the RTSM

```
<CMDBAttribute>
     <Name>[Attribute Name]</Name>
     <SetValue>
          [Operands]
     </SetValue>
</CMDBAttribute>
```

Mapping to String values of a maximum length in the RTSM

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

Mapping to Integer values in the RTSM

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

Mapping to Boolean values in the RTSM

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

Mapping to Long values in the RTSM

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

Mapping to Date values in the RTSM

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

Mapping to Float values in the RTSM

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

Mapping to Byte values in the RTSM

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

Mapping to Double values in the RTSM

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

Mapping to StringList values in the RTSM

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

Mapping to IntList values in the RTSM

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

Attribute Mapping Example

```
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd">
  <Rules>
    <Rule name="Map Display Label">
      <Condition>
        <True/>
      </Condition>
      <MapTo>
        <CMDBAttribute>
          <Name>display_label</Name>
          <SetValue>
            <Caption/>
          </SetValue>
        </CMDBAttribute>
      </MapTo>
    </Rule>
  </Rules>
  <Rules Context="exchange">
    <Rule name="Set data_name key attribute">
      <Condition>
        <True/>
      </Condition>
      <MapTo>
        <CMDBAttribute>
          <Name>data_name</Name>
          <SetValue>
            <OMId/>
          </SetValue>
        </CMDBAttribute>
      </MapTo>
    </Rule>
    <Rule name="Set host_key key attribute for nodes">
      <Condition>
        <IsNode/>
      </Condition>
      <MapTo>
        <CMDBAttribute>
          <Name>host_key</Name>
          <SetValue>
            <OMId/>
          </SetValue>
        </CMDBAttribute>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

For all CIs (no matter which context is assigned) the CMDB attribute `display_label` is set to the Caption of the OM CI. CIs that are assigned to the context `exchange` have `data_name` and for nodes the `host_key` attribute set to the OM ID.

Relation Mapping

Using the relation mapping you can create relations between CIs. For topology synchronization, the OM associations are not synchronized as relations by default. You must explicitly define these relations.

```
<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. The topology sync 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.

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>
```

The CI with the STD `Exch2k7_ByServer` gets an 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 an OM Type that starts with `Exch2k7_Role_`.

The same node is also the root container CI.

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 "The following figure illustrates the data structure exposed to the navigation." on the next page.

CI Data Structure

OMAttributes

Contains a map of all original RTSM (Run-time Service Model) 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.
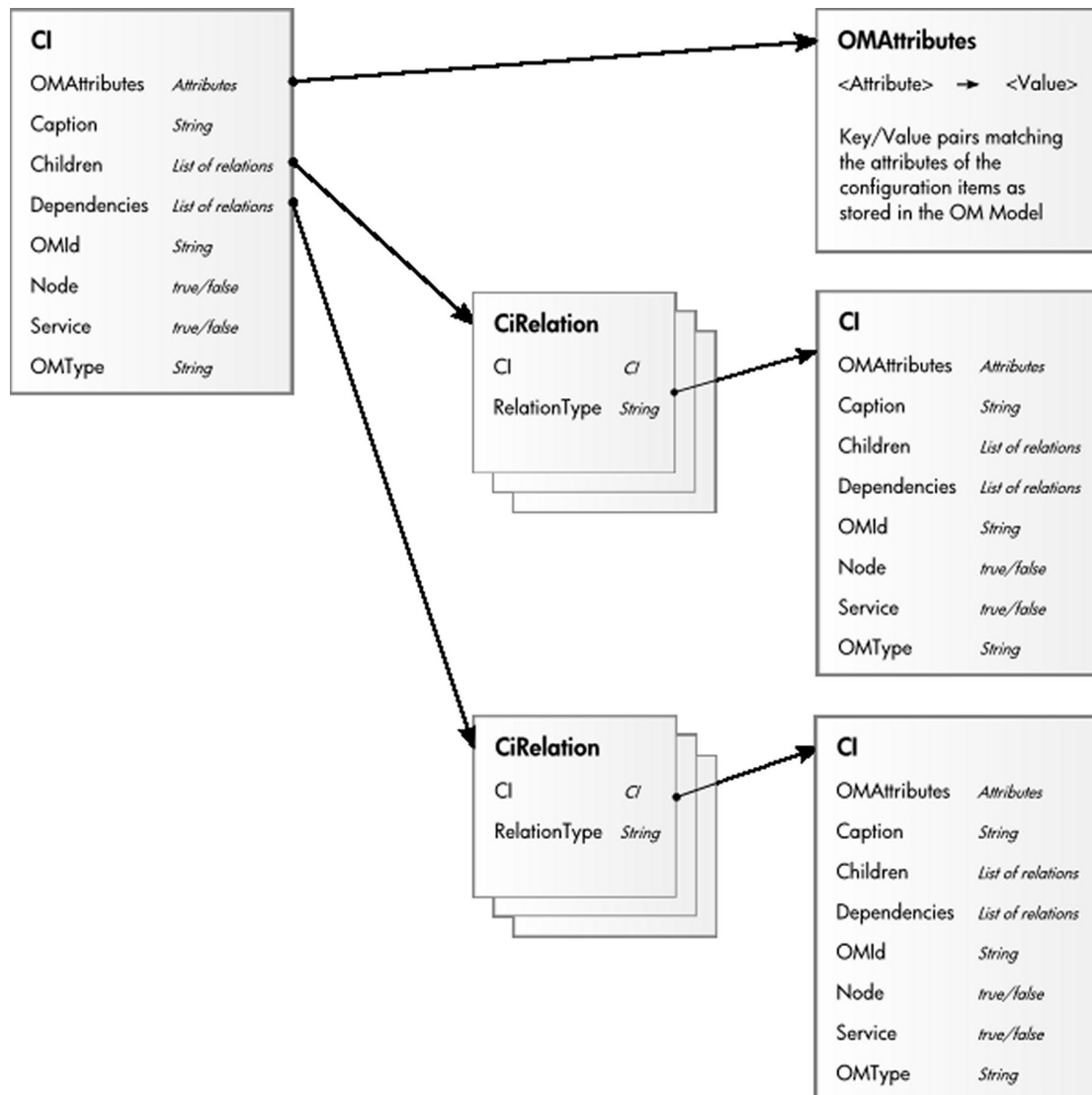
Example of an XPath-Navigated Data Structure

An example of an XPath-navigated data structure is shown in "The following figure illustrates some XPath examples." 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.
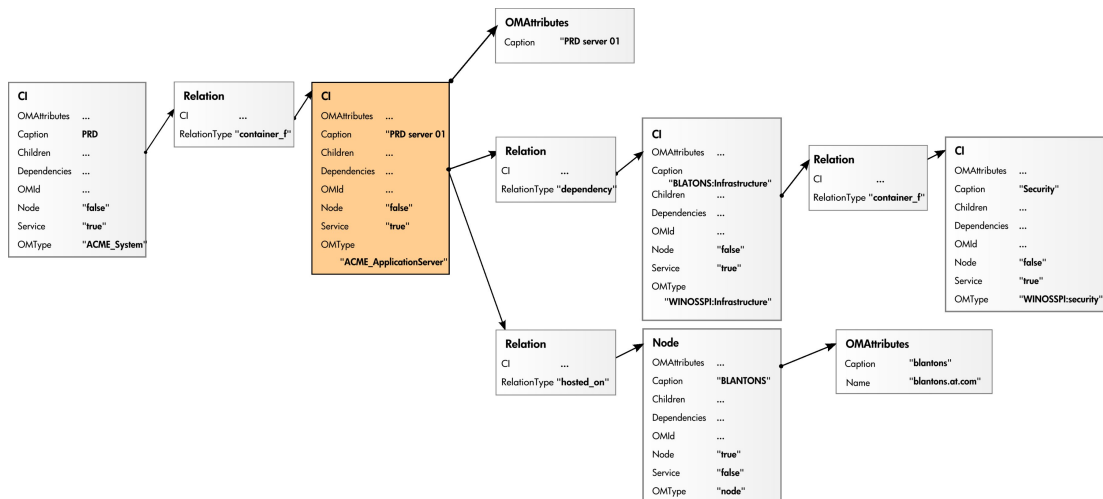
XPath Expressions and Example Values

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.

Chapter 10

Event Processing Interface

This section describes the role of event processing scripts and custom actions for modifying and enhancing events during event processing.

This section is structured as follows:

- "Event Processing Interface" on the next page
- "Scripts For Custom Actions" on page 256
- "Creating EPI and Custom Action Scripts" on page 257
- "EPI Troubleshooting" on page 267

Event Processing Interface

This section provides an overview of the Event Processing Interface (EPI), explaining the different steps in the event pipeline, and appropriate entry points for running scripts for modifying and enriching events.

Event Processing Interface and Scripts

The EPI enables you to run user-defined Groovy scripts during event processing. With these scripts, you can modify and enhance events with external data. For example, you could enrich events with additional data from an external SQL database, or an Excel list.

"Event processing with pipeline entry points for scripts" below provides an overview of the event processing, showing the event pipeline, and how the EPI script execution can be integrated into this process.
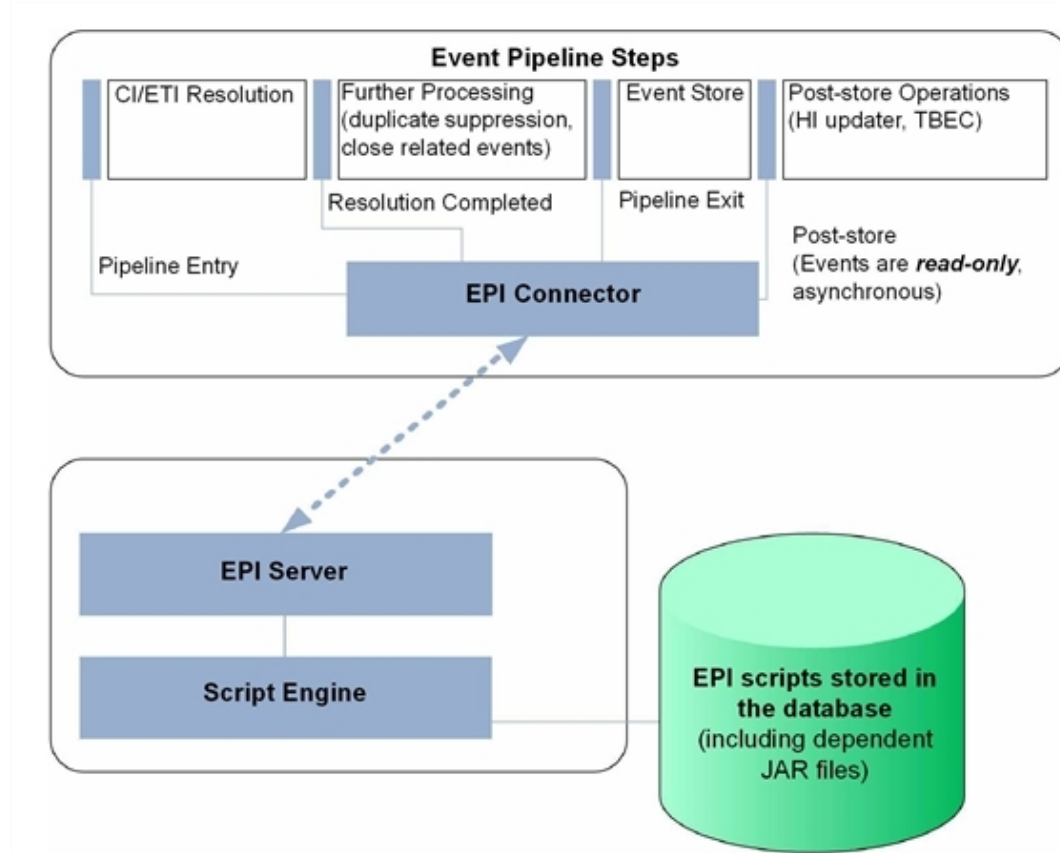
Event processing with pipeline entry points for scripts

Scripts are stored in the database. Any JAR files or libraries that are referenced or used by the scripts are also loaded into the database.

For each pipeline step, you can set up one or more scripts. There is no limit on the number of scripts that can be executed. However, be aware that the time it takes to process events in the pipeline increases with the number of scripts that are executed.

> **Note:** You should design and execute scripts in the context of overall event processing. In other words, be aware of the interaction of the scripts with other event processing settings in the Settings Manager, for example for duplicate events suppression and closing related events.

Entry Points for Running EPI Scripts

The event pipeline represents the various steps of event processing. There are four points within the event pipeline at which EPI scripts can be executed:

- Before CI/ETI resolution

- After CI/ETI resolution

- Before the event is stored in the database

- After the event is stored in the database

Before CI/ETI Resolution

Scripts can be executed directly before the event enters the event pipeline, so before the resolution of CIs and ETIs takes place.

For example, you may want to execute a script at this point that sets further hints that affect the resolution of CIs and ETIs. An entry point further along the event pipeline would be too late to influence the resolution of CIs and ETIs.

After CI/ETI Resolution

Scripts can be executed directly after CI/ETI resolution, but before further processing, such as duplicate events suppression and closing related events automatically.

For example, you may want to execute a script at this entry point in the event pipeline if you want to influence how duplicate events are handled. It could be that you have duplicate events suppression enabled in general, but you are interested in changing the duplicate events suppression setting for a particular type of event, while leaving it unchanged for all other event types. So you could execute a script at this entry point that disables duplicate events suppression for the specified event type. Any entry point further along the event pipeline would be too late to influence duplicate events suppression behavior.

Before Storing Events

Scripts can be executed after all event processing has taken place, but before the event is stored in the database.

For example, at this entry point to the event pipeline, you could execute a script that makes changes to some text, or inserts a link to a knowledge base, and so on, before the event gets stored in the database.

After Storing Events

Scripts can be executed after the event has been stored in the database. In this case, the scripts are all executed in read-only mode, since as soon as an event has been stored on the database, it can no longer be modified.

For example, you may want to execute a script at this entry point in the event pipeline to forward events of a particular type, that have already been stored in the database, to another application. Or you could execute a script that writes specified events stored in the database to an audit log.

Specifying a Script

You configure the EPI scripts in the following area of the configuration user interface:

Admin > Operations Management > Event Automation > Event Processing Customizations

You can create, copy, edit, and delete EPI scripts. You can also determine the order in which the scripts are executed.

For more details about creating scripts, see "Creating EPI and Custom Action Scripts" on page 257.

For details about applying and managing scripts to event processing, refer to the Operations Management online help.

EPI scripts can be defined in content packs and can be imported/exported using the Content Manager.

EPI Script Execution

The steps involved in script execution are as follows:

1. The EPI server reads the script and calls the `init()` function.

2. The EPI server calls the `process()` function, with the list of events specified as parameter. The script stays in memory, and the `process()` function is called when matching events arrive.

3. The EPI server calls the `destroy()` function if the script is unloaded, for example, if the script is disabled by the user, or at system shutdown.

   The `destroy()` function is also called if the script is modified in the Event Processing Customizations manager. In this case the `destroy()` function is called on the old, unmodified script, followed by a call to the `init()` function on the new, modified script.

The call to the `process()` function of a script conforms to the ACID (Atomicity, Consistency, Isolation, Durability) principle that ensures that database transactions are processed reliably.

- **Atomicity**

If a script cannot be successfully executed (does not return without errors, and no exception is thrown), any changes made so far are rolled back. Subsequent events will get the event list as it was before the script in error was executed.

- **Consistency**

  Values that violate the consistency of an event can be over-ridden by the system. For example, setting an user ID and a group ID where the user ID is not a member of the group ID is such a violation, and can be over-ridden.

- **Isolation**

  Scripts with read/write access to events are not executed in parallel, but sequentially.

- **Durability**

  Changes made to events by executing scripts are stored in the database.

EPI Script for Event Enrichment

It is possible to access CI-related data available in the RTSM for a resolved CI and use this information to enrich an event. An EPI groovy script in conjunction with the RTSM JAVA API forms the basis of this approach. See the "Script Example" on page 253 section for a working example.

The available data for event enrichment can be any attribute listed under:

Admin > RTSM Administration > CI Type Manager > *<selected CI Type>* > Attributes

Learn More

Basic Script Design

One of the requirements with regards to accessing the RTSM via the JAVA API is that a connection needs to be established first, including providing credentials for a valid user. Establishing the connection involves processing overhead and it is not practical to open and close a new connection for each event being processed. For EPI scripts, you need to minimize the processing time for each event being handled.

To minimize processing time, you can pre-load a simple script-internal HashMap containing CI ObjectIDs linked to a string containing the corresponding value of the CI attribute used for creating our event CA. The pre-loaded HashMap resides in memory, and enables fast retrieval of a small subset of RTSM information.

The `init()` method is used to establish a connection to the RTSM. A separate thread is started in which the HashMap is initialized. This thread is also synchronized with the main thread (via a synchronization object) so that the HashMap can be momentarily blocked and reloaded after a specified wait period (for example 10 minutes).

The `process()` method is used to:

- Retrieve the `ObjectID` of the event's related CI
- Access the HashMap to obtain the "owner" information
- Invoke `event.AddCustomAttribute` to create the CA

The `destroy()` method (called when the script is unloaded or de-activated) is used to release the memory used by the HashMap.

Building the HashMap

To retrieve a set of CIs from the RTSM, a TQL query is executed using the HP UCMDB API. For full documentation on the available APIs, see HP UCMDB API Reference. These files are located in the following folder:

<BSM GW Server install directory>\AppServer\webapps\site.war\amdocs\eng\doc_lib\API_docs\UCMDB_JavaAPI

The script illustrates how to execute a "named query" (a query that has already been defined in the RTSM).

After CIs are retrieved via the query, the HashMap must be initialized. For each CI in the query result, the `ObjectID` and the corresponding string value of the attribute (obtained via the `ci.getPropertyValue` method) are loaded. Any string attribute can be used.

> **Note:** The query should return CIs which actually have the desired attribute defined (and ideally populated) in the RTSM.

The background thread loads a parallel map (`newMap`) in the part of the code that is not synchronized with the main thread. When the thread gets the lock (is synchronized), it initializes the "run-time" HashMap (`ownerMap`) from the temporary one. This is a safe and effective approach to keeping the HashMap up to date to reflect ongoing changes in the RTSM topology

How to Use the OwnerResolver Script

The following procedure describes how to use a script to enrich events during event processing after the CI has been resolved with CI attributes from the RTSM:

1. Define a TQL query in **RTSM Administration > Modeling Studio** that returns the CIs of interest.
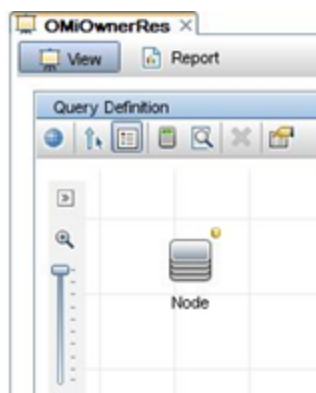
In this example, the `OMiOwnerRes` query returns CIs of type `Node` that have a value of `NOT null` for the attribute `DiscoveredContact`.

> **Note:** `DiscoveredContact` is a standard attribute of the `Node` CIs type. You need to manually add values to the `DiscoveredContact` attribute for each node CI so that the example returns contact names in the Contact column of the Event Browser.

The `Node` CI Type conditions are set in the Query Node Properties dialog box as follows:

2. To make the `DiscoveredContact` attribute available to the RTSM JAVA API, you must mark it for calculation under "Element Layout":
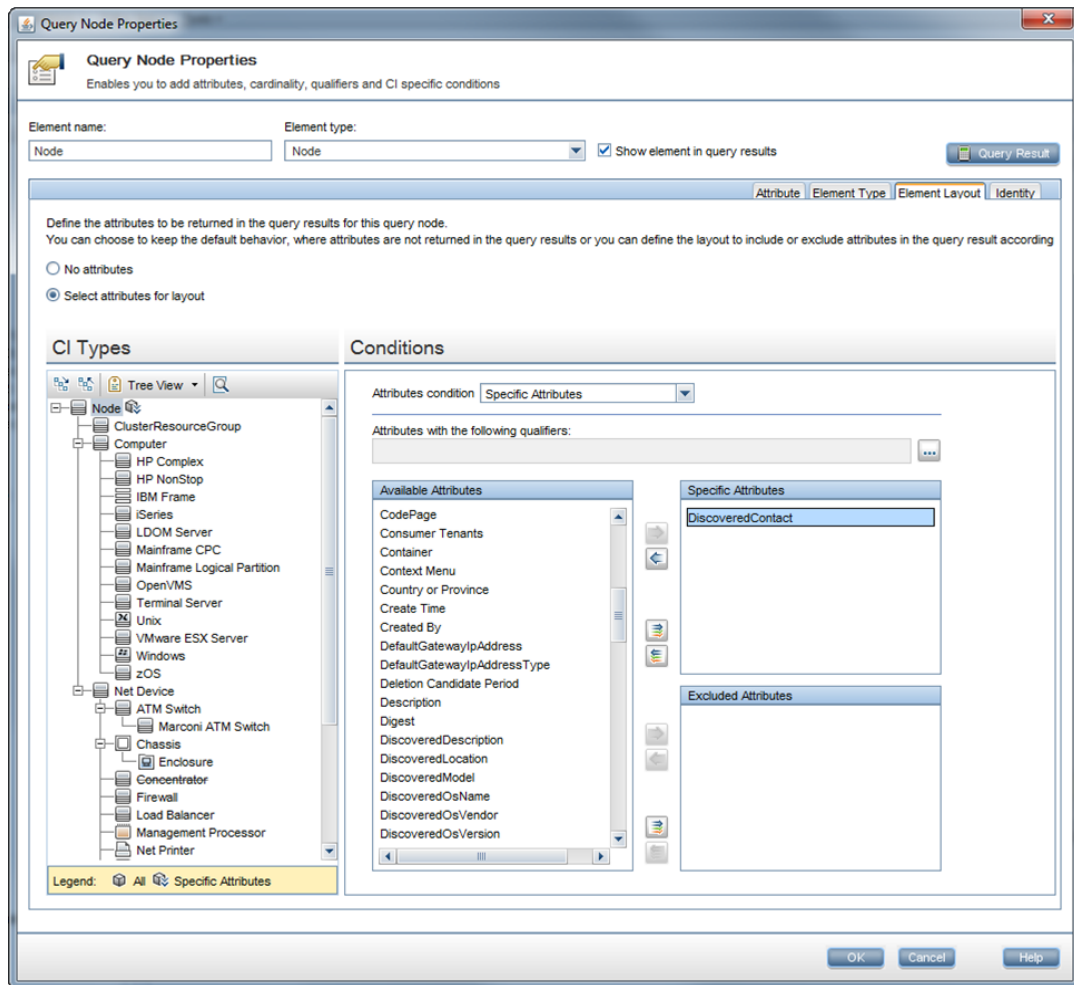
3. Install the script (in this case, with appropriate hostname, credentials, query name, and CI attribute name specified) in the **After CI/ETI Resolution** pipeline step.

> **Note:** No special classpath is required.

4. Exposed the CONTACT CA for use in your browser by configuring the CA in the Available Custom Attributes Operations Management Infrastructure Settings:

   a. Select **Admin > Platform > Setup and Maintenance > Infrastructure Settings**

   b. Select **Applications** and use the list to set the administration context to **Operations Management**

   c. Open the Available Custom Attributes item in the Custom Attributes Setting pane.

   d. Add CONTACT as a new value.

Edit Setting

Name:	Available Custom Attributes
Description:	Semicolon-separated list of Custom Attributes that can be added as Event Browser columns.
Value:	stomer;Region;Manager;Company;CONTACT
Note:	The change takes effect after closing all open web browsers and logging in again.

5. Modify your browser columns to show the new CA (CONTACT).

Title	Related CI	State	CONTACT
Interface error rate exceeds threshold of 2 errors/sec.	11223344556a		
CPU exceeds 95 percent for 10 minutes.	tucana		Worf
Hello from ovlabdt2 (CI attribute modified)	ovlabdt2 (OMW Svr)		Jean-Luc Picard
Hello from ovlabdt2	ovlabdt2 (OMW Svr)		William Riker
Average query time has exceeded threshold of 10 seconds.	web_order		
Memory utilitzation has exceeded 90 percent for longer than 5 minutes!	orion		Geordi
SetEventKPIs	SA_7900_2		Data

The `DiscoveredContact` attribute from the RTSM is used to populate the CONTACT column.

Some fields in the CONTACT column are empty because the related CI Type is not `Node` (or child CI Type of Node). As a result, there is nothing in the HashMap because this CI Type is not included in the TQL used to build the HashMap.

Script Example

To enrich events during event processing after the CI has been resolved with CI attributes from the RTSM, the following definition can be used in your EPI Groovy scripts:

```
def ucmdbService
UcmdbService service = null
```

The following OwnerResolver script template illustrates how the definition can be used:

```
import com.hp.ucmdb.api.*
import com.hp.ucmdb.api.topology.*

class OwnerResolver
{
  def ucmdbService
  final String QUERY_NAME = "<myQueryName>"

  HashMap<String,String> ownerMap = new HashMap<String,String>()
  boolean running = true
  UcmdbServiceProvider provider = null
  UcmdbService service = null
  Object syncObject = new Object()
```

```
void init()
{
  ucmdbService.getCmdbService()

  Thread.start {
    while (running)
    {
      TopologyQueryService tqs = service.getTopologyQueryService()
      Topology topology = tqs.executeNamedQuery(QUERY_NAME)
      HashMap<String,String> newMap = new HashMap<String,String>()
      topology.getAllCIs().each{ ci -> newMap.put(ci.getId().
        getAsString(), ci.getPropertyValue("discovered_contact")) }
      synchronized (syncObject)
      {
        ownerMap = newMap
        try
        {
          syncObject.wait(600000)
        }
        catch (InterruptedException e)
        {
          // ignore
        }
      }
    }
  }
}

void destroy()
{
  ownerMap = null
  running = false
  synchronized (syncObject)
  {
    syncObject.notifyAll()
  }
}

void process(eventList)
{
  synchronized (syncObject)
  {
    eventList.each { event ->
      def ciId = event.getRelatedCiId()
      if (ciId != null)
      {
        def owner = ownerMap.get(ciId)
        if (owner != null)
          event.addCustomAttribute('CONTACT', owner)
      }
    }
```

```
        }
    }
}
```

Tips and Limitations

Some tips for using scripts for retrieving RTSM data:

- Use conditions in your query to limit the CIs returned to only those that have some value in the attribute you want to use. This avoids loading CIs with no corresponding attribute value.

- Be mindful of how many CIs the query returns. Do a **Calculate Query Result Count** in RTSM Administartion before using the query within the script. The more CIs it returns, the more memory will be required for the HashMap.

- The script name (saved under **Admin > Operations Management > Event Automation > Event Processing Customizations**) must be the same as the class name specified in the script.

- Use the actual CI attribute name in the script, NOT the display name (in the `getPropertyValue` method)

- You can find the time taken by your script for each event from the following log file:

  `<BSM install dir>/log/opr-epi-server/opr-epi-server.log`

Creating Scripts

EPI scripts and custom actions scripts share the same script definition format.

For more details about creating scripts, see "Creating EPI and Custom Action Scripts" on page 257.

Scripts For Custom Actions

This section describes how to configure scripts for custom actions. Custom actions let you define your own actions to apply to events. You can configure Groovy scripts to make custom actions available in the Event Browser.

- "Specifying Custom Actions Scripts" below

- "Creating Scripts" below

Specifying Custom Actions Scripts

> **Note:** To be able to specify or execute Operations Management custom actions, users must have the appropriate permissions granted in the BSM User Management settings. For details about how to do this, see the BSM online help.

The Custom Actions manager enables you to setup scripts to run custom actions on events. As a simple example, you can add a text string to certain events to make them easier to identify in the Event Browser.

You specify custom actions in Groovy scripts. You configure the scripts for custom actions in the following area of the configuration user interface:

Admin > Operations Management > Operations Console > Custom Actions

After a custom action is configured in Operations Management, the script is available in the list of scripts in the Scripts pane. The script can be triggered from an event from the context menu **custom action list**. The selected custom action is launched in the context of the CI associated with the selected event. If a custom action is run from a non-assigned event, that event is automatically assigned to the user that executed the custom action, and a corresponding entry is made in the Event History.

You can create, copy, edit, and delete scripts for custom actions. You can also stop the execution of a script.

For more details about creating scripts, see "Creating EPI and Custom Action Scripts" on the next page.

For details about how to configure custom actions, refer to the Operations Management online help.

Custom action scripts can be defined in content packs and can be imported/exported using the Content Manager.

Creating Scripts

EPI scripts and custom actions scripts share the same script definition format.

For more details about creating scripts, see "Creating EPI and Custom Action Scripts" on the next page.

Creating EPI and Custom Action Scripts

This section describes how to create EPI and custom action scripts.

This section is structured as follows:

- "Script Definition Attributes" below

- "Groovy Script API" below

- "Script Definition Format" on the next page

- "EPI Groovy Script Samples" on the next page

- "Custom Actions Groovy Script Samples" on page 264

Script Definition Attributes

A script requires a script definition, and for this you need to specify script definition attributes.

A script definition consists of the following attributes:

- **Name**: The internal script name (**not** the file name of the script).

- **Classpath / JAR files**: One or more JAR files can be uploaded together with the script. The content of the JAR file is available on the classpath during execution of the script. The order of the jar files on the classpath can be changed by moving jar files in the UI up or down.

- **Filter**: *Optional, EPI scripts only*. A filter can be referenced by name. Only events that match the filter are passed to the script.

- **Read-only**: *Optional*. Scripts specified with the read-only attribute do not modify events. These scripts are executed asynchronously to read/write scripts. This asynchronous execution speeds up overall event processing, so it is recommended as a best practice to set the read-only attribute for scripts that are not intended to modify events.

- **Active**: Enables or disables the script for execution. When a script is enabled, the `init()` function is called. Similarly, when a script is disabled, the `destroy()` function is called.

- **Timeout**: *Optional*. The maximum time for each script invocation. This is independent of the number of events. The default value for the timeout is 0, which means that the script execution will never timeout.

  For synchronous scripts, if the timeout is reached, script execution is aborted, and all changes made to the events are rolled back.

  For asynchronous scripts, if the timeout is reached, script execution is aborted.

Groovy Script API

If you intend to create your own EPI or custom action scripts, you must implement a Groovy script that at a minimum contains calls to the `init()`, `process()`, and `destroy()` functions. The Groovy script API with full documentation of all arguments and types can be found in the Java API Documentation delivered with the product.

The Java API Documentation includes the following information needed for creating scripts:

- Event class: this is the main interface

- Complete list of event attributes that are available for modification

You can find the Java API Documentation at the following location:

<HPBSM root directory>/opr/api/doc/opr-external-api-javadoc.zip

Script Definition Format

The basic format of a script definition for EPI scripts and custom action scripts looks like this:

```
import com.hp.opr.api.scripting.Event;

def init()
{
  // This method is called when the script is loaded (for example,
  // when it is enabled in the configuration user interface).
}

def destroy()
{
  // This method is called when the script is unloaded (for example,
  // if it isdisabled in the configuration user interface)
}

def process(List<EpiEvent> events)
{
  // This method is called when events are processed. The list is of
  // type java.util.List.


  // In this method, properties of the events can be changed. If the
  // script is in read-only mode, the UnsupportedOperationException is
  // thrown.

  // See the Java API documentation for opr-external-api.jar for a
  // list of event attributes that are available for modification.
}
```

EPI Groovy Script Samples

This section contains the sample EPI Groovy scripts shipped with the product.

You can find the EPI Groovy script examples in the following directory:

<HPBSM root directory>/opr/examples/epi_scripts

SimpleExampleEPI.groovy

This is a simple example that sets all possible event attributes to some sample values.

```
import java.util.Date;
import java.util.List;

import com.hp.opr.api.scripting.Action;
import com.hp.opr.api.scripting.Event;
import com.hp.opr.api.scripting.EventActionFlag;
import com.hp.opr.api.scripting.LifecycleState;
import com.hp.opr.api.scripting.MatchInfo;
import com.hp.opr.api.scripting.NodeInfo;
import com.hp.opr.api.scripting.PolicyType;
import com.hp.opr.api.scripting.Priority;
import com.hp.opr.api.scripting.ResolutionHints;
import com.hp.opr.api.scripting.Severity;

/*
 * This example set all possible event attribute to some example
values.
 */

class SimpleExample
{
        def init()
        {
        }

        def destroy()
        {
        }

        def process(List<Event> events)
        {
                        events.each {
                                event -> modifyEvent(event);
                        }
        }

        def modifyEvent(Event event)
        {
                String application = event.getApplication();
                event.setApplication("Modified by EPI: " + application);

                long groupId = event.getAssignedGroupId();
                event.setAssignedGroupId(groupId);

                int assignedUserId = event.getAssignedUserId();
                event.setAssignedUserId(assignedUserId);

                Action autoAction = createSampleAction();
                event.setAutoAction(autoAction);
```

```
String category = event.getCategory();
event.setCategory("Modified by EPI: " + category);

String correlationKeyPattern = event.getCloseKeyPattern();
event.setCloseKeyPattern("Modified by EPI: " +
correlationKeyPattern);

String description = event.getDescription();
event.setDescription("Modified by EPI: " + description);

String etiInfo = event.getEtiHint();
event.setEtiHint(etiInfo);

String correlationKey = event.getKey();
event.setKey("Modified by EPI: " + correlationKey);

MatchInfo matchInfo = createSampleMatchInfo();
event.setMatchInfo(matchInfo);

event.setNoDedup(true);

ResolutionHints hints = createSampleResolutionHints();

event.setNodeHints(hints);

String object = event.getObject();
event.setObject("Modified by EPI: " + object);

String omServiceId = event.getOmServiceId();
event.setOmServiceId(omServiceId);

String omUser = event.getOmUser();
event.setOmUser(omUser);

String originalText = event.getOriginalData();
event.setOriginalData("Modified by EPI: " + originalText);

String originalId = event.getOriginalId();
event.setOriginalId(originalId);

event.setPriority(Priority.HIGHEST);

String ciInfo = event.getRelatedCiHint();
event.setRelatedCiHint("Modified by EPI: " + ciInfo);

event.setSeverity(Severity.CRITICAL);

String solution = event.getSolution();
event.setSolution("Modified by EPI: " + solution);
```

```
            ResolutionHints sourceCiHints = createSampleResolutionHints();
            event.setSourceCiHints(sourceCiHints);

            event.setState(LifecycleState.IN_PROGRESS);

            String subCategory = event.getSubCategory();
            event.setSubCategory("Modified by EPI: " + subCategory);

            event.setTimeReceived(new Date());

            String title = event.getTitle();
            event.setTitle("Modified by EPI: " + title);

            String type = event.getType();
            event.setType("Modified by EPI: " + type);

            Action userAction = createSampleAction();
            event.setUserAction(userAction);
    }

    def ResolutionHints createSampleResolutionHints()
    {
            ResolutionHints hints = new ResolutionHints(false);

            hints.setCoreId("CoreId");
            hints.setDnsName("mydqdn.com");
            hints.setHint("My Hint");
            hints.setIpAddress("0.0.0.0");
            return hints;
    }

    def MatchInfo createSampleMatchInfo()
    {
            MatchInfo matchInfo = new MatchInfo(false);

            matchInfo.setConditionId("conditionId");
            matchInfo.setPolicyName("policyName");
            matchInfo.setPolicyType(PolicyType.CONSOLE);
            return matchInfo;
    }

    def Action createSampleAction()
    {
            NodeInfo actionNodeInfo = new NodeInfo(false);

            Action action = new Action(false);
            actionNodeInfo.setCoreId("CoreId");
            actionNodeInfo.setDnsName("myfqdn.com");
            actionNodeInfo.setIpAddress("0.0.0.0");
```

```
                    action.setCall("Call");
                    action.setNode(actionNodeInfo);
                    action.setStatus(EventActionFlag.AVAILABLE);
                    return action;
            }
    }
```

RegExample.groovy

This example script flips every second word with its previous word.

```
import java.util.Date;
import java.util.List;

import com.hp.opr.api.scripting.Action;
import com.hp.opr.api.scripting.Event;
import com.hp.opr.api.scripting.EventActionFlag;
import com.hp.opr.api.scripting.LifecycleState;
import com.hp.opr.api.scripting.MatchInfo;
import com.hp.opr.api.scripting.NodeInfo;
import com.hp.opr.api.scripting.PolicyType;
import com.hp.opr.api.scripting.Priority;
import com.hp.opr.api.scripting.ResolutionHints;
import com.hp.opr.api.scripting.Severity;

/*
 * This script flips every second word with its previous word.
 */
class RegExpExample
{
        def init()
        {
        }

        def destroy()
        {
        }

        def process(List<Event> events)
        {
                events.each {
                        event -> event.setTitle(event.getTitle().replaceAll(/(\w+)\s+(
/, '$2 $1'));
                        }
        }
}
```

ResolveLocationFromDB.groovy

This script matches an IP address to a node name to resolve a location from the database.

To make this example work with a MS SQL Server, you must do the following:

1. Create a new DB `asset_db` (or adjust the name below).

2. Create a new table `LocationMapping` (or adjust the name below) with the following attributes:

   - `ip (varchar(50), primary key)`

   - `location (varchar(50))`

   - `phone (varchar(50))`

   - `contact (varchar(50))`

3. Add rows to the table for each IP address that matches the node hint of your events.

4. Adjust the database parameters in `Sql.newInstance` below.

5. Add this script as an EPI script. Upload a JTDS driver. You can find a JTDS driver here:

   <HPBSM root directory>`/lib/jtds-1.0.jar`

```
import groovy.sql.Sql;

class ResolveLocationFromDB
{
  def connection;
  void init()
  {
    connection = Sql.newInstance
("jdbc:jtds:sqlserver://localhost/asset_db",
                               'sa', 'installed',
                               "net.sourceforge.jtds.jdbc.Driver");
  }

  void process(eventList)
  {
    try {
      for (event in eventList) {
      def nodeHints = event.getNodeHints();
      def nodeName = nodeHints.getDnsName();
      if(nodeName != null) {
        def ipaddress = InetAddress.getByName(nodeName).hostAddress
        connection.eachRow('select * from LocationMapping', {
          if (it.ip == ipaddress) {
            if (event.getDescription() == null)
              event.setDescription("CI located in building: " +
                                    it.location)
            else
              event.setDescription(event.getDescription() +
                                    "\nCI located in building: " +
                                    it.location)
            event.addCustomAttribute('phone', it.phone)
            event.addCustomAttribute('contact', it.contact)
            event.addCustomAttribute('location', it.location)
```

```
            }
        });
      }
    }
  }

  finally {
  }
}

  void destroy()
  {
    connection.close();
  }
}
```

Custom Actions Groovy Script Samples

This section contains some examples of Groovy scripts for custom actions.

You can find the custom actions Groovy script examples in the following directory:

<HPBSM root directory>/opr/examples/ca_scripts

SimpleExample.groovy

Here is a simple example of a custom actions script that modifies an event:

```
import com.hp.opr.api.scripting.Event;
import com.hp.opr.api.scripting.Priority;
import com.hp.opr.api.scripting.Severity;

class SimpleExample
{
      def init()
      {
            // Nothing to initialize
      }
      def destroy()
      {
            // Nothing to destroy
      }
      def process(List<Event> events)
      {
            events.each {
                  event -> modifyEvent(event);
            }
      }
      def modifyEvent(Event event)
      {
```

```
                    event.addCustomAttribute("CA_SCRIPT", "MODIFIED");
                    event.setSeverity(Severity.CRITICAL);
                    event.setPriority Priority.HIGHEST;
            }
    }
```

TranslateEventTitle.groovy

This script translates a title from English to German.

> **Note:** You must download the google Translate API to run this custom action script. The Google Translate API can be found here:
>
> ```
> http://google-api-translate-java.googlecode.com/files/google-api-
> translate-java-0.92.jar
> ```
>
> Attach/upload this jar file together with this script as a custom action.

```
import java.util.List;
import com.hp.opr.api.scripting.Event;
import com.google.api.translate.Language;
import com.google.api.translate.Translate;
class TranslateTitleToGerman
{
        def init()
        {
                // Set the HTTP referrer to your website address.
                Translate.setHttpReferrer("***MY_REFERER***");

                // Set a proxy address for outgoing connections. If no proxy is
required
                // you can just delete the next two lines.
                System.setProperty("http.proxyHost", "***PROXY_HOST***");
                System.setProperty("http.proxyPort", "***PROXY_PORT***");
        }

        def destroy()
        {
                // Nothing to destroy or shutdown
        }

        def process(List events)
        {
                for (Event e : events)
                {
                        translateTitle(e);
                }
        }

        def translateTitle(Event event)
```

```
        {
            String translatedText =
                    com.google.api.translate.Translate.execute(
                        event.getTitle(),
                        Language.ENGLISH, Language.GERMAN);
            event.setTitle(translatedText);
        }
    }
```

EPI Troubleshooting

This section contains information to help you troubleshoot EPI and custom action script execution.

Log Files

A good starting point for troubleshooting EPI and custom action script execution is to look at the following log file:

`<HPBSM root directory>\log\opr-scripting-host\opr-scripting-host.log`

Debugging

To perform a debug, do the following:

1. Go to the following location:

   `<HPBSM root directory>/conf/core/Tools/log4j/opr-scripting-host/opr-scripting-host.properties`

2. In the `opr-scripting-host.properties` file, set the `loglevel` to the desired value.

   The debug loglevel (`loglevel=DEBUG`) is useful for finding problems.

Log File Entries

The log file entries can provide helpful information about scripts and their execution. For example, you can see:

- When a script was loaded or shut down.
- When a script execution reached its timeout.
- Statistics about the execution of a script, for example, the time taken for script execution.

Chapter 11

Integrating the Operations Management UI with Other Applications

This section describes how to integrate parts of the Operations Management user interface with an external application using a drill-down URL launch.

This section is structured as follows:

URL Launch of the Event Browser

You can launch the Event Browser using a URL link. This is particularly interesting for integrators who want to integrate parts of the Operations Management user interface (UI) with an external application. So an operator of an external application with a graphical user interface can drill down into the Operations Management UI. For example, it is possible to have a portal application, where operators can launch an Event Browser and the Event Details in a browser within their application.

This section describes how to specify such a URL launch and includes the following sections.

- "Specifying a URL Launch" below

- "Parameters and Parameter Values" on the next page

- "Defining Columns" on page 272

- "Setting Filters" on page 274

- "URL launch of the Event Details" on page 279

Specifying a URL Launch

To perform a URL launch of the Event Browser, you can either use the default URL, or specify additional parameters. The optional parameters you specify in the URL enable you to define how you want the Event Browser to appear and behave.

Default URL Launch

When you use the default URL launch, an Event Browser opens with default Operations Management UI settings.

> **Note:** In an Event Browser launched with the default URL, any changes you make (for instance, visible columns and columns widths) are *not* automatically saved.

To launch the Event Browser with default UI settings, enter the URL as follows:

```
http://<hostname:port>/opr-console/opr-evt-browser
```

> **Note:** You must specify the port number if the port you are using is not the default HTTP port (80). If you are using the default port, you need not specify the port number.

Specifying Optional Parameters

You can specify additional parameters to define what you want to display in the Event Browser.

The available parameters, together with their possible values, are given in "Parameters for the URL Launch of the Event Browser" on the next page.

You specify additional parameters for the URL launch in the following way:

```
http://<hostname:port>/opr-console/opr-evt-browser?<set_parameters_
here>
```

> **Note:** You must specify the port number if the port you are using is not the default HTTP port (80). If you are using the default port, you need not specify the port number.

> **Note:** You place the character "?" in front of the first argument of the URL, and thereafter separate each argument by the "&" character.

Here is an example of a URL launch with optional parameters:

```
http://my.example.com:8080/opr-console/opr-evt-
browser?sortField=severity&sortOrder=desc&filter_severities=critical,
major,minor
```

Parameters and Parameter Values

"Parameters for the URL Launch of the Event Browser" below contains the parameters available for the URL launch of the Operations Management Event Browser:

Parameters for the URL Launch of the Event Browser

Parameter	Description	Possible Values	Default Values
activeUpdates	Specifies whether the Event Browser gets updates periodically from the server. Without periodic updates, be aware that you may not get all data into the browser.	One of: • true • false	true
checkParams	Determines whether unknown URL launch parameters are validated. If you do not specify the checkParams parameter, unknown parameters (for example, misspelled parameters) are not validated; the Event Browser ignores unknown parameters and opens without error notifications. Known parameters are always validated. If you set the checkParams parameter to true, the Event Browser validates all parameters and reports unknown parameters as errors.	One of: • true • false	false

Parameter	Description	Possible Values	Default Values
`columns`	Defines the columns to be displayed in the Event Browser.	One or more of: `<Column key>` (see "Fixed Columns Keys to Define Which Columns are Displayed" on the next page)	Default set of columns
`context`	Defines the context of the browser settings. Editing browser settings (for example, visible columns or column widths) when a context is selected, means that the settings are saved under the given context. The next time you open the browser with the same context, the browser settings are restored. If you do not specify the `context` parameter, the browser settings are not saved. Example usage: `context=myOwnContext`	Any user-defined string	No default value set
`headerText`	Sets a header text for the Event Browser.	Any user-defined string	`Event Browser`
`showClosed`	Launches the closed events browser configuration when the system starts.	One of: ● `true` ● `false`	`false`
`showDetails`	Launches the Event Browser with embedded event details already open.	One of: ● `true` ● `false`	`false`

Parameter	Description	Possible Values	Default Values
`sortField`	Defines the column by which the Event Browser is sorted.	One of: *`<Column key>`* (see "Fixed Columns Keys to Define Which Columns are Displayed" below)	`timeReceived`
`sortOrder`	Defines the sort order of the column specified by the `sortField` parameter, in either ascending or descending order.	One of: ● `asc` ● `desc`	`desc`

Defining Columns

The fixed column keys for available columns are defined in "Fixed Columns Keys to Define Which Columns are Displayed" below. "Custom Attribute Column Key" on page 274 contains a special kind of column key, that unlike the other column keys, is customer configurable. All the column keys are case-insensitive, which means that for example "ID", "Id" and "id" all map to the ID column.

For more detailed explanations about the column keys, refer to the Operations Management online help.

Fixed Columns Keys to Define Which Columns are Displayed

Column Key	Explanation
`annotations`	Annotations
`application`	Application
`automaticAction`	Automatic action
`category`	Category
`ciType`	CI type
`controlTransferred`	Shows whether control of the event has been transferred
`correlation`	Correlation (symptom or cause)
`description`	Description
`duplicateCount`	Number of duplicates
`eti`	Event type indicator value

Column Key	Explanation
externalId	External event ID
group	Assigned group
id	ID
node	Node
nodeHint	Node hint
object	Object
operatorAction	Operator action
originatingServer	Originating server
ownedInOM	Owned in HP Operations Manager (HPOM)
priority	Priority
receivedOnCiDowntime	Received during CI downtime
relatedCi	Related CI
relatedCiHint	Related CI Hint
sendingServer	Sending server
severity	Severity
solution	Solution
sourceCi	Source CI
sourceCiHint	Source CI hint
state	Lifecycle state
subCategory	Subcategory
timeCreated	Time when the event was created
timeReceived	Time when the event was received
timeStateChanged	Time when lifecycle state of the event was last changed
title	Title
type	Type
user	Assigned user

Custom Attribute Column Key

Column Key	Explanation
ca_<columnname>	Custom attribute column, where <columnname> is the name of the custom attribute. This is a configurable column key that you configure in the Available Custom Attributes setting in Operations Management - Custom Attribute Settings. For details, refer to the Operations Management online help.

Setting Filters

You can narrow down the number of events in the Event Browser by using filters. You can specify filters for the following:

- String attributes

- Time properties

- Flag attributes

- Event priorities

"Filter Parameters for the URL Launch of the Event Browser" below contains the filter parameters available for the URL launch of the Event Browser.

Filter Parameters for the URL Launch of the Event Browser

Filter Parameter	Description	Values
filter_assignment	Applies an assignment filter to the Event Browser.	One or more of: • me • my_ workgroups • others • nobody
filter_severities	Applies a severity filter to the Event Browser.	One or more of: • unknown • normal • warning • minor • major • critical

Filter Parameter	Description	Values
`filter_states`	Applies a lifecycle state filter to the Event Browser.	One or more of: • `open` • `in_ progress,` • `resolved`

Filtering by String Attributes

You can filter events by string attributes.

You specify filters to filter by properties of type string using the following format:

`filter_<stringAttributeName>_<filterType>`

Possible values for string attribute names and filter types are listed in "Possible Filter Types and Values for String Attributes" below.

Possible Filter Types and Values for String Attributes

Possible string attribute names	`application`
	`ca_<customAttribute>`
	`category`
	`correlationKey`
	`description`
	`object`
	`originalText`
	`relatedCiHint`
	`subCategory`
	`title`
	`type`
Possible filter types	`contains`
	`equals`
	`isEmpty`
	`isNotEmpty`
	`notContains`
	`notEquals`

Here is an example of filtering by string attributes, where we are interested only in returning events that have a filter category called `Network`:

```
filter_category_equals=Network
```

Filtering by Time Properties

You can also filter events by time properties.

You can set the filter so that the Event Browser displays only those events for which the time attribute `timeAttributeName` is between `fromTime` and `toTime`.

You specify filters to filter by time using the following format:

`filter_<`*`timeAttributeName`*`>=fromTime-toTime`

where:

<`*`timeAttributeName`*`>` can be one of: `timeReceived`, `timeCreated`, `timeStateChanged`.

You must specify the time in the following format:

<*`yyyyMMddHHmmss`*>

where:

- `yyyy` is the year
- `MM` is the month value (01-12)
- `dd` is the day value (01-31)
- `HH` is the hour value (00-23)
- `mm` is the minute values (00-59)
- `ss` is the second value (00-59)

Filtering by Priorities

You can also filter events by priorities.

You can set the filter so that Event Browser displays the events according to their priorities.

You specify filters to filter by priorities using the following format:

`filter_priorities=<`*`priority_level`*`>`

where:

<*`priority_level`*`>` can be one or more of the following in a comma-separated list: `none`, `lowest`, `low`, `medium`, `high`, `highest`.

As an example, if you want the Event Browser is to display only those events with the priorities `highest` and `high`, then you would specify the filter as follows:

`filter_priorities=highest,high`

Filtering by CIs and CI Types

You can set a filter so that the Event Browser displays only the events that are related to a given CI. To specify such a filter, use the following format:

```
filter_relatedCi_equals=<CI Id>
```

where:

`<CI Id>` is the ID of the CI.

Possible filter operations are: `equals`, `isempty`, `notisempty`

You can also set a filter so that the Event Browser displays only events where the CI type of the related CI matches the CI type specified. To specify such a filter, use the following format:

```
filter_ciType_<operator>=<type>
```

where `<type>` is the specified CI type, and `<operator>` can have one of the following values: `equals`, `is_derived`.

Filtering by Global CI ID

You can filter events by global CI ID. The global CI ID is the global ID of a CI in an external (non-BSM) database, such as a Content Management System (CMS) database.

When specifying the `globalCiId` parameter, the local (ODB) CI ID is looked up and a filter for that CI ID is set up.

To specify such a filter, use the following format:

```
filter_globalCiId_equals=<global id>
```

where:

`<global id>` is the global ID of the CI.

Possible filter operations are: `equals`, `isempty`, `notisempty`

The filter specification `filter_globalCiId_equals=<global id>` returns the same result as the filter specification `filter_relatedCi_equals=<CI Id>`.

Filtering by ETIs and ETI Values

You can set a filter so that the Event Browser displays only the events that match a given ETI.

To specify a filter by ETI, use the following format:

```
filter_eti_equals=<ETI Id>
```

where:

`<ETI Id>` is the UUID of the ETI.

Possible filter operations are: `equals`, `isempty`, `notisempty`, `isoneof`.

When using the `isoneof` filter operation, a comma-separated list of ETI UUIDs is expected, for example:

`filter_eti_isoneof=<ETI Id1,ETI Id2,ETI Id3,...>`

For the filter operations `isempty` and `notisempty`, no argument is expected.

You can also set a filter so that the Event Browser displays only events that set a specific ETI Value. To specify such a filter, use the following format:

`filter_etiValue_<operator>=<ETI value Id>`

where:

`<ETI value Id>` is the UUID of the ETI Value.

Possible filter operations are: `equals`, `isempty`, `notisempty`, `isoneof`.

When using the `isoneof` filter operation, a comma-separated list of ETI value UUIDs, is expected, for example:

`filter_etiValue_isoneof=<ETI value Id1,ETI value Id2,ETI value Id3,...>`

For the filter operations `isempty` and `notisempty`, no argument is expected.

Filtering by Other Event Characteristics

You can also set a filter to group events that share the same characteristic. For example, you may want to see only those events that have duplicates.

To filter events to group them by event characteristics, there are a number of boolean flag attributes that you can specify. These boolean flag attributes specify whether the event possesses a particular characteristic, for example, whether it has symptoms, or annotations. You can set the filter so that the Event Browser displays only those events that match the boolean flag attribute for the specified characteristic.

To set a filter to display events that share a particular characteristic, you specify the boolean flag attribute for that characteristic using the following format:

`filter_<flagAttributeName>`

where:

`<flagAttributeName>` can have one of the following values: `hasSymptoms`, `hasCause`, `hasDuplicates`, `hasAnnotations`.

You can also specify a combination of characteristics, as in the following example where the filter is set to display only those events that have both duplicates and symptoms:

`http://<my.example.com:8080>/opr-console/opr-evt-browser?filter_ hasDuplicates&filter_hasSymptoms`

URL launch of the Event Details

In a similar way to launching the Event Browser using a URL, you can also launch the Event Details using the following URL:

```
http://<hostname:port>/opr-console/opr-evt-details?eventId=<id_of_the_event>
```

where:

<id_of_the_event> is the ID of the event that you want to display in the Event Details.

Here is an example of the URL for a direct launch of the Event Details, showing the parameter `eventId` set to the ID of the event that you want to display:

```
http://my.example.com:8080/opr-console/opr-evt-details?eventId=e004e66b-cada-407f-84ac-32f2d613eec4
```

Chapter 12

Automating Operator Functions and Event Change Detection

This section provides integrators with information to allow them to programmatically automate operator functions and detect event changes. Everything that an operator can do in the console while working on events can be done programmatically, to improve efficiency.

For information about integrating applications for event synchronization, see "Forwarding Events and Synchronizing Event Changes" on page 330.

This section is structured as follows:

- "Automating Operator Functions using the Event Web Service Interface" on the next page

- "REST Web Service Command-Line Utility" on page 303

- "Event Web Service Query Language" on page 312

Automating Operator Functions using the Event Web Service Interface

An interface is provided for integrators to integrate events into other applications, enabling them to programmatically automate operator functions and detect event changes. Most operations that an operator can do in the console while working on events can be done programmatically, to improve efficiency, and enable integration with external applications.

The interface used for integrating events into other applications, and automating operator functions, is the Event Web Service. This is a REST-based web service supporting the event model, that also provides subscription support through Atom feed functionality. You can read an Atom feed in your browser, where you can see a list of events, and you can also create and update events using the Atom service.

For convenience, the link to the URL for launching the Event Browser (for drilling down into the Operations Management user interface from an external application) is included in the event web service data. For details of how to specify and launch the drill-down URL, see "URL Launch of the Event Browser" on page 269.

An integrator would typically be interested in:

- "How to Access the Event Web Service" below

- "How to Detect New Events" on page 283

- "How to Detect Event Changes" on page 285

- "How to Modify Events" on page 286

- "How to Create New Events" on page 291

- "Event Web Service Security" on page 295

How to Access the Event Web Service

Your entry point to the event web service interface is the Service Document, using the following base URL:

- Standard environments:

  `http://<bsmserver.example.com>/opr-console/rest/`

- Secure environments:

  `https://<bsmserver.example.com>/opr-console/rest/`

where:

<bsmserver.example.com> is the name of the gateway server.

For access to the Event Web Service, you need to be a valid user, and need to provide your user name and password as credentials for user authentication. Only authorized users can view events, change events or run actions.

The Service Document lists the URLs of different OPR Event services. These services are listed in "List of OPR Event Services in the Service Document" on the next page.

In the list of URLs in the Service Document, only two of them are actual links:

- Events Service:

  `http://<bsmserver.example.com>/opr-console/rest/<version>/event_list`

  `<version>` is the version of the BSM release (for example, `9.10`). If the version number is omitted, versions lower than 9.10 are addressed.

  This shows a list of all events.

- Event Changes Service:

  `http://<bsmserver.example.com>/opr-console/rest/event_change_list`

  This shows a list of changes to events.

All the other URLs require a parameter to be specified (for instance, the event ID). In case of annotations, for the event for which you want the annotation, you need to specify the annotation ID. In "List of OPR Event Services in the Service Document" below, variables that you need to specify for each URL are shown in curly brackets, such as {event}, {annotation}, or {custom_attribute}. An example URL specifying the event with the ID `532d3674-684f-419f-a752-b8681ee01a72` would look like this:

`http://<bsmserver.example.com>/opr-console/rest/9.1/event_list/532d3674-684f-419f-a752-b8681ee01a72`

List of OPR Event Services in the Service Document

OPR Event Service	URL
Annotation Service:	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/annotation_list/{annotation}`
Annotations Service:	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/annotation_list`
Automatic Action Service:	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/auto_action`
Custom Attribute Service:	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/custom_attribute_list/{custom_attribute}`
Custom Attributes Service	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/custom_attribute_list`
Event Change Service	`http://<bsmserver.example.com>/opr-console/rest/event_change_list/{event_change}`
Event Changes Service	`http://<bsmserver.example.com>/opr-console/rest/event_change_list`

OPR Event Service	URL
Event Service	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}`
Events Service	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list`
History Line Service	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/history_line_list/{history_line}`
History Lines Service	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/history_line_list`
Operator Action Service	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/user_action`
Symptom Service	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/symptom_list/{symptom}`
Symptoms Service	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/symptom_list`

In general, for most of the OPR Event services, you can carry out the four supported operations (read, create, update and delete). Exceptions to this are:

- Automatic Action Service and Operator Action Service. For these services you can:
  - Get the state of an action by performing a read operation.
  - Start an action (if it is not already running) by performing a create operation.
  - Stop an action (if it is already in progress) by performing a delete operation.
- Services where only read operations are possible:
  - Event Change Service
  - Event Changes Service
  - History Line Service
  - History Lines Service

At first glance, the two services Event Change and Event Changes return very similar results to the services History Line and History Lines. However, the former services list all event changes, and are not specific to any particular events. The latter services are event-specific, and list changes to events relating to particular events. For more details about History Lines, see "History Lines" on page 326.

How to Detect New Events

To return a list of all events, in the Service Document, click the URL for the event list:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
```

By default, clicking this URL opens the event list in XML format. You can add HTTP query parameters to this base URL to change the way you want to view the data in the list:

- `alt=atom`: This parameter presents the list of events in an Atom feed format. How the event data is displayed is determined by meta-data (for example, categories, author, and so on). For further details about the `alt` media type, see "Media Type" on page 317.

- `alt=json`: This parameter presents the list of events in JSON format.

- `alt=xml`: This parameter presents the list of events in XML format. This the default for a web browser.

Receiving Events as Atom Feeds

Most browsers come with a very basic internal feed reader for Atom and RSS feeds. As the feed readers vary in their features and in the way they represent data, the same data may be represented differently by different readers (and you would typically also access the data in a different way, too).

The two most commonly available browsers with feed readers are Mozilla Firefox and Microsoft Internet Explorer. The following examples show how the Atom web service works with these browsers.

Event lists in an Atom feed are ordered as "last changed", so a new event, or an event that was modified last would be at the top of the list.

To return a list of events in an atom feed format, follow these steps:

1. Enter the URL for the event list in your web browser, specifying the `alt=atom` parameter as follows:

   ```
   http://<bsmserver.example.com>/opr-console/rest/9.10/event_
   list/?alt=atom
   ```

2. You now see a list of events. Firefox and Internet Explorer represent the data in a slightly different way.

   - **In Firefox**:

     You see a list of events where the title and description of each event are shown.

     If you click on the title of an event in the atom feed, you start the URL launch of the Event Details for that event. For more information about the URL launch of the Event Browser and Event Details, see "URL Launch of the Event Browser" on page 269.

     > **Note:** If you want to see a more detailed list of the events, right-click the page and select **View Page Source**. This displays the whole atom feed in XML format. Now you can see all other properties and information regarding the events that are listed.

   - **In Internet Explorer**:

     You see a list of events where the title and description of each event are shown. In addition to the list, you see a filter box on the right, where you can sort the list by date and title. You can also filter the events by specific categories.

If you click on the title of an event in the atom feed, you start the URL launch of the Event Details to drill down into the Operations Management user interface. For more information about the URL launch of the Event Browser and Event Details, see "URL Launch of the Event Browser" on page 269.

Only basic details about the events are shown, and to see all the details, right-click on the page and select **View Source**. This opens the page in your XML editor.

Specifying Parameters to Filter the Event List

The Event Web Service provides a number of parameters for filtering the event list by specific criteria.

For example, you can filter results by matching specific event parameters to a condition in a URL query parameter. You can also choose to reduce the size of the list by displaying only a restricted number of events, and page through them. The default page size is 20.

Another example would be to specify a time parameter which lists only those events that have been created or updated after a specified time. When an application that is monitoring new and changed events using the web service shuts down, it will want to remember the time of the last changed event. In this way when it restarts it can query for only those new and changed events since this timestamp.

To return a list of events after a specific date, you specify the `watermark` parameter.

Alternatively, you can get a list of events from a certain sequence number onwards by specifying the `sequence_number` parameter.

The following example URL returns the first 40 currently open events since 15:59:17 on March 4, 2010 in an atom feed format.

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list/?alt=atom&watermark=2010-03-04T15:59:17%2B02:00&page_
size=40&start_index=1
```

For details about the URL query language, query filter and parameters, see "Event Web Service Query Language" on page 312.

How to Detect Event Changes

To return a list of event changes, in the Service Document, click the URL for the event change list:

```
http://<bsmserver.example.com>/opr-console/rest/event_change_list
```

You may want to return a list of all event changes since the last time you got a list of events. The following example URL returns a list of all event changes since 12:29:54 on March 10, 2010 in an atom feed format:

```
http://<bsmserver.example.com>/opr-console/rest/event_change_
list?alt=atom&watermark=2010-03-10T15:59:17%2B01:00
```

How to Modify Events

You can modify events (read, create, update and delete items) using a REST client within your web browser, or using the `RestWsUtil` command-line utility.

> **Note:** In this section, we describe a standard HTTP environment. For secure environments, use HTTPS.

> **Note:** With BSM 9.10 and higher, Event Web Service modify operations must be secured by setting the `X-Secure-Modify-Token` HTTP header on modify requests (PUT, POST, and DELETE). This header provides enhanced security protection against malicious exploits of web applications. For more information, see "Event Web Service Security" on page 295.

Modifying Events Using a REST Client

In general, when using a REST client to update an event, it is usual to do the following:

- Call the REST service by entering the URL of the event in question (you need to specify the event ID in the URL).

- Retrieve the event with an HTTP GET request.

- Edit the elements of the XML document you want to change.

- Send back the changed parts of the event using an HTTP PUT request.

- Reload the XML to see the changes.

Two examples of REST clients that can be used to modify events are the RESTClient and the Mozilla Firefox Poster Extension.

Modifying Events Using RESTClient

Here, we will look at how to use the RESTClient to modify events. The RESTClient is available as an open source download. You can find information about where to download the RESTClient from this location:

`http://code.google.com/p/rest-client/`

The RESTClient user interface is divided into two parts:

- HTTP Request: top half of the user interface, used for entering the URL for the event you want to modify, to retrieve the event with an HTTP GET request, and for sending the changed XML using an HTTP PUT request.

- HTTP Response: bottom half of the user interface, used for returning the response.

We will use an example of how to change the title and severity of an event using the RESTClient. The steps required are as follows:

1. Get the event ID of the event you want to modify from the event list feed.

2. In the URL field of the RESTClient user interface, enter the URL (specifying the event ID) of the event you want to modify. The syntax is as follows:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list/<event_ID>
```
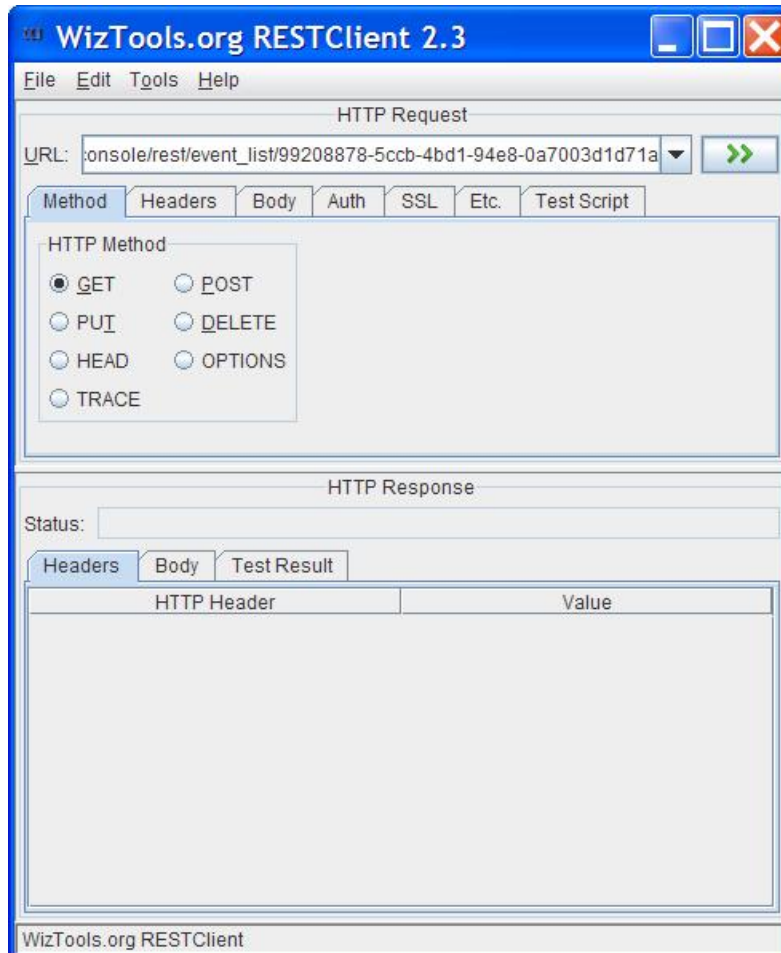

3. Select the **GET** radio button in the HTTP Method box, and click the ⟩⟩ button.

4. The result is returned in the lower, HTTP Response section. Click the **Body** tab to read the response XML for the event you want to edit.

5. Copy the XML. Click the **Body** tab in the upper, HTTP Request section, and paste the XML into the text box.

6. Edit the event properties you want to change to modify the event. It is not necessary to send the complete XML back to the server. As long as you preserve the XML structure, you can choose to send only those XML elements that you want to update for that event.

7. When you are done with your changes, select the **Method** tab again, and then click the **PUT** radio button.

8. Click the ⟩⟩ button to submit the changes. When the changes have been applied, you will see an HTTP 200 OK message in the Response area. Check the Atom feed to verify the

changes you made.

Modifying Events Using the Firefox Poster Extension

To illustrate how you can modify events using a REST client, Here we will look at how to use the Mozilla Firefox Poster Extension to modify events. The Poster Extension is a simple REST client that you first need to install as a plug-in for Firefox.

We will use an example of how to change the title and severity of an event using the Firefox Poster Extension. The steps required are as follows:

1. First install the Mozilla Firefox Poster Extension if it is not already installed.

2. Open a Firefox browser, and select **Tools > Poster**. This opens a Poster dialog box.

3. Get the event ID of the event you want to modify from the event list feed.

4. In the URL field of the Poster dialog box, enter the URL including the event ID of the event you want to modify. The syntax is as follows:

   ```
   http://<bsmserver.example.com>/opr-console/rest/9.10/event_
   list/<event_ID>
   ```
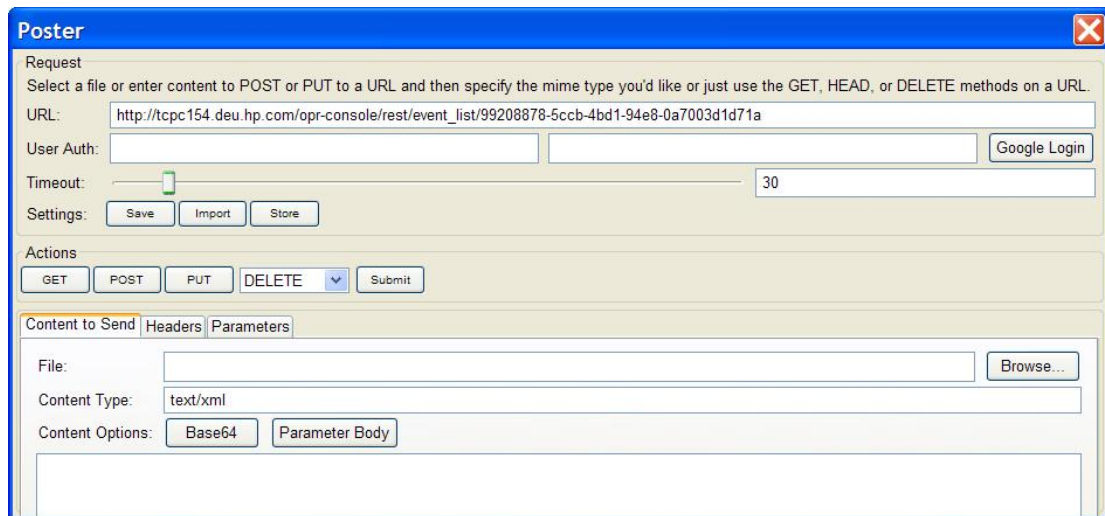

5. Click **GET** to receive the event as XML. A window entitled Response opens, and if there are no errors, the status should be stated as `200 OK` and you should be able to see the full XML of the event you want to modify.
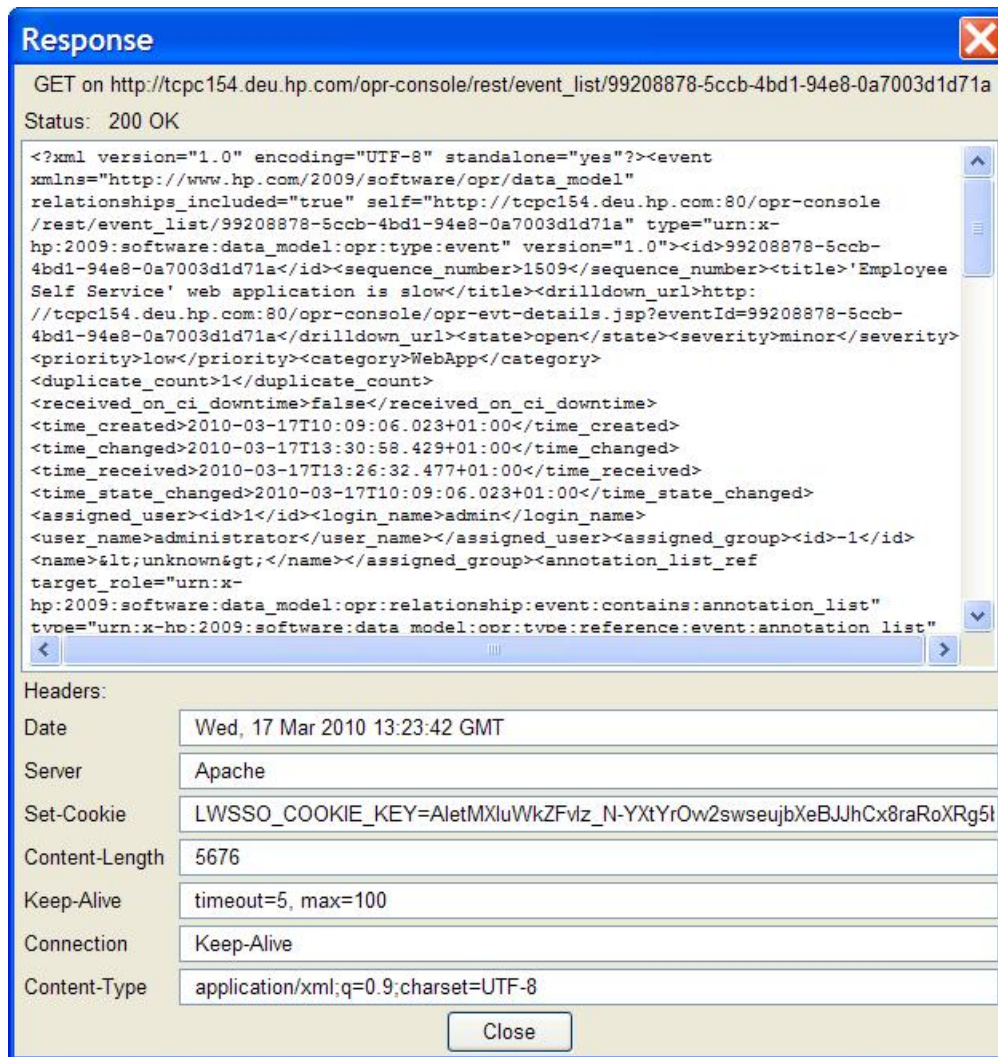
6. Copy the full XML from the Response window and paste it into content text field of the Content to Send tab in the Poster dialog box. You no longer need the Response window, so you can close it now.

7. Next, you can edit the XML according to the changes you want to make to the event. It is not necessary to send the complete XML back to the server. As long as you preserve the XML structure, you can choose to send only those XML elements that you want to update for that event.

> **Note:** Not all properties can be updated. For a list of editable properties, see "Editable Properties" on page 325. Also check the latest Java API documentation, which is contained in a zip file that you can find in the following location:
>
> *<HPBSM root directory>*`/opr/api/doc/opr-external-api-javadoc.zip`
>
> Unzip the contents of the zip file to a suitable location to see the API documentation.

In this example, we will change the title of the event, and also the event severity, directly in the XML. For example, the event's original title was: "`'Employee Self Service' web application is slow`" and we want to edit the event title by inserting the word "very":

`...<title>'Employee Self Service' web application is **very** slow</title>...`

Similarly, we could change the event severity from minor to warning:

`...<severity>**warning**</severity>...`

8. Once you have made your changes to the event, make sure that the Content Type field is set to **application/xml** (you must type this in). Then click **PUT** to save your changes. A Response window opens, and if there are no errors, you see an HTTP `200 OK` message. Check the Atom feed to verify the changes you made.
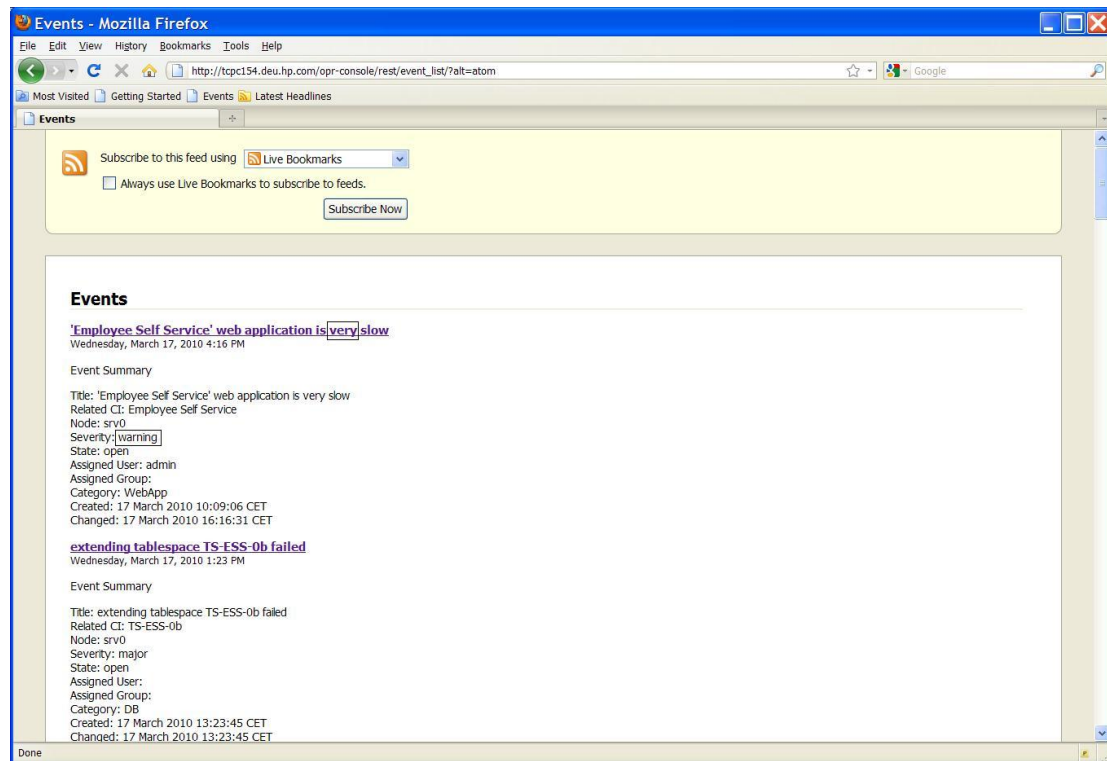
More About Firefox Poster Extension

You can find more information about Firefox Poster Extension at the following location:

`http://code.google.com/p/poster-extension/`

Modifying Events Using the RestWsUtil Utility

A REST web service utility is provided to allow you to execute REST web service operations against the event web service from the command-line. With this utility, you can execute one of the four following REST web service operations:

CRUD Operation	HTTP Method
Create	POST
Read	GET
Update	PUT
Delete	DELETE

You can find the utility in the following location:

<HPBSM root directory>`/opr/bin`

You can modify events using the `RestWsUtil` command-line utility.

We will look at a very simple example of how you would modify an event using the utility.

Example: How to Change an Event Title

To change the title of an event using the `RestWsUtil` utility, follow these steps:

1. Get the event ID of the event you want to modify.

2. Write the XML for the event you want to modify to an XML file (called, for example, update.xml) in the **<*HPBSM root directory*>/opr/bin** directory.

   The contents of the `update.xml` file look like this:

   ```
   <event xmlns="http://www.hp.com/2009/software/opr/data_
   model"><title>
   New title goes here</title></event>
   ```

   Edit the title.

3. To update and save the change, enter the following command in a command prompt:

   **<*HPBSM root directory*>/opr/bin>RestWsUtil -update update.xml -
   username <*login name*> -password <*password*> -url
   http://<*bsmserver.example.com*>/opr-console/rest/9.10/event_
   list/<*event_ID*>**

   Where:

   <HPBSM root directory> is the directory where BSM is installed, *<login name>* is the user name required for authentication, and *<event_ID>* is the ID of the event you want to modify.

For further details and examples of how to use the `RestWsUtil` command-line utility, see "REST Web Service Command-Line Utility" on page 303.

How to Create New Events

You can create new events using the `RestWsUtil` command-line utility.

> **Note:** To create events, users must have been granted the `Event Submission` permission

> in the BSM User Management settings. For details about how to do this, see the HP Business Service Management online help.

Example: How to Create a New Event

To create an new event using the `RestWsUtil` utility, follow these steps:

1. Create an XML file in the following folder (called, for example, `create.xml`):

   `<HPBSM root directory>/opr/bin/create.xml`

2. Define the properties of the new event by adding the following lines to the file:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>New event title</title>
   <severity>normal</severity>
   <priority>low</priority>
   <state>open</state>
</event>
```

3. Open a command prompt and type:

   `<HPBSM root directory>/opr/bin>RestWsUtil -create create.xml -username <login name> -password <password> -url http://<bsmserver.example.com>/opr-console/rest/9.10/event_list`

   Where:

   `<HPBSM root directory>` is the directory where BSM is installed, `<login name>` is the user name required for authentication.

For further details and examples of how to use the `RestWsUtil` command-line utility, see "REST Web Service Command-Line Utility" on page 303.

Advanced Modification of Event Properties

An event has a selected number of list properties that can be modified by the OPR Event services listed in the Service Document, described in the section "How to Access the Event Web Service" on page 281.

To generate an XML list of the custom attribute properties of the specified event, you call the following URL:

`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/<event_ID>/custom-attribute-list/`

Where:

`<bsmserver.example.com>` is the name of the gateway server, and `<event_ID>` is the ID of the event for which you want to list the custom attributes.

The REST response to such a URL call (in this case for an event with the ID 26629e00-2d8d-71dd-1aa2-1039228c0111) could look like this:

```
<custom_attribute_list
      xmlns="http://www.hp.com/2009/software/opr/data_model"
      self="http://<bsmserver.example.com>/ws/rest/event_
list/26629e00-2d8d-71dd-1aa2-1039228c0111/custom_attribute_list"
      type="urn:x-hp:2009:software:data_model:opr:type:event:custom_
attribute_list"
      version="1.0">
  <custom_attribute
      self="http://<bsmserver.example.com>/ws/rest/event_
list/26629e00-2d8d-71dd-1aa2-1039228c0111/custom_attribute_
list/drilldown.url.ci"
      type="urn:x-hp:2009:software:data_model:opr:type:event:custom_
attribute"
      version="1.0">
    <name>drilldown.url.ci</name>
    <value>http://url.to/drill/down/ci</value>
  </custom_attribute>
  <custom_attribute
      self="http://<bsmserver.example.com>/ws/rest/event_
list/26629e00-2d8d-71dd-1aa2-1039228c0111/custom-attribute-
list/drilldown.url.event"
      type="urn:x-hp:2009:software:data_model:opr:type:event:custom_
attribute"
      version="1.0">
    <name>drilldown.url.event</name>
    <value>http://url.to/drill/down/event</value>
  </custom_attribute>
</custom_attribute_list>
```

Using the Event Web Service, you can create and edit list items, and also delete items from the custom attributes list.

> **Note:** You cannot create or delete an event itself using the RestWsUtil command-line utility.

To create a custom attribute item for a specified event, you send an HTTP POST request with the corresponding XML object to the event custom attribute list URL. When you call the custom attribute list URL for your specified event, you see that your new item has been added to the list of custom attributes for that event.

To edit an item in the list of custom attributes, you do something very similar. You send an HTTP PUT Request specifying an existing item name and a changed value for that item. The Event Web Service updates the value to the new value.

The HTTP response could look like this:

```
<custom_attribute
      xmlns="http://www.hp.com/2009/software/opr/data_model"
      self="http://<bsmserver.example.com>/ws/rest/event_list/26629e00-
2d8d-71dd-1aa2-1039228c0111/custom_attribute_list/mynewattribute"
      type="urn:x-hp:2009:software:data_model:opr:type:event:custom_
attribute"
```

```
      version="1.0">
  <name>mynewattribute</name>
  <value>Hello</value>
</custom_attribute>
```

Similarly, to delete an item from the list of custom attributes, you send an HTTP DELETE request to the following URL:

`http://<bsmserver.example.com>/opr-console/rest/9.10/event_ list/<event_ID>/custom_attribute_list/<custom_attribute_name>`

Where:

`<custom_attribute_name>` is the name of the custom attribute selected for deletion.

Bulk Update of Events

In addition to updating events individually with reference to the specific event ID, you can also update events in bulk. This is done not by addressing a specific event, but by addressing the event list with a query (see "Event Web Service Query Language" on page 312), and specifying the update in the payload of a PUT request.

For example:

URL: `http://my.host.com/opr-console/rest/9.10/event- list&query=title%20LIKE%20'%25db down%25'`

HTTP Method: PUT

Pay-load:	`<event xmlns="http://www.hp.com/2009/software/opr/data_ model">`
	`<severity>major</severity>`
	`</event>`

The above request sets all events with `db down` in the title to severity `major`.

HTTP Method: PUT

Pay-load:	`<event xmlns="http://www.hp.com/2009/software/opr/data_ model">`
	`<state>closed</state>`
	`</event>`

The above request closes all events with `db down` in the title.

Bulk Insert of Events

In addition to inserting individual events, a list of events may be inserted in a single Web Service call. This is done by executing a POST request with the following properties:

- An `event_list` rather than just a single event, for example:

  ```
  <event_list xmlns="http://www.hp.com/2009/software/opr/data_model">
      <event>
  ```

```
        <title>Major Event</title>
        <severity>major</severity>
    </event>
    <event>
        <title>Minor Event</title>
        <severity>minor</severity>
    </event>
</event_list>
```

- A Content-Type of "`application/xml; type=collection`" or "`text/xml;`
`type=collection`".

    When using the `RestWsUtil` utility, you can use the `-content_type` option to specify the content type.

Event Web Service Security

To access the Event Web Service, you need to be a valid user, and need to provide your user name and password as credentials for user authentication. Only authorized users can view events, change events or run actions.

Event Web Service clients must secure modify operations (PUT, POST, and DELETE) by setting the `X-Secure-Modify-Token` HTTP header. This header provides enhanced security protection against malicious exploits of web applications. For details, see "Securing Modify Operations" on the next page.

If the CA SiteMinder Web Agent is configured for `CssChecking=YES`, characters configured in the CA SiteMinder Web Agent `BadUrlChars` parameter are rejected by the CA SiteMinder Web Agent. See "Environments with CA SiteMinder" on page 301 for more information about how to avoid these characters in URLs.

Changing Error Verbosity

The Event Web Service returns error codes that help administrators identify the cause of possible problems. Administrators can control the types of error codes returned from the Event Web Service by changing the level of verbosity in the Infrastructure Settings Manager. The following options are available:

Option	Description
Standard	Default. Returns to the caller of the web service the appropriate HTTP error code, as listed in the HTTP 1.1 standard, along with a standard error text message describing the error.
Verbose	Recommended for development environments. Returns a detailed message describing the reason for the error.
Brief	Returns only error codes 400 (Bad Request) or 503 (Service Unavailable), depending on the type of the error, and an error identifier. A detailed error message may be obtained by locating the identifier and detailed error message in the error logs.

1. Navigate to the Web Service Settings in the Infrastructure Settings Manager:

   Infrastructure Settings > Applications > Operations Management > Operations Management - Web Service Settings > Error Response Verbosity

2. *Optional.* Change the default value **Standard** to another value.

3. Review the error log file on the BSM Gateway Server:

   <HPBSM root directory>/log/opr-ws-response.log

Securing Modify Operations

With BSM 9.10 and higher, Event Web Service modify operations must be secured by setting the X-Secure-Modify-Token HTTP header on modify requests (PUT, POST, and DELETE). This header provides enhanced security protection against malicious exploits of web applications. Enhanced protection for modify operations is by default enabled in the Web Service Settings in the Infrastructure Settings Manager. You can disable the setting for backwards compatibility (see).

Set the X-Secure-Modify-Token HTTP Header

Event Web Service clients must first obtain the secureModifyToken cookie, and then set the value of the cookie in the X-Secure-Modify-Token HTTP header:

1. Obtain the secureModifyToken cookie before executing any modify requests (PUT, POST, or DELETE).

   The recommended approach to obtain the secureModifyToken cookie at client startup is to execute an HTTP GET request for the Event Web Service Service Document at /opr-console/rest.

   After the HTTP GET operation has completed the cookie is set. During the life of the client and the single sign-on session, the value of the cookie may change. Before each modify operation the HTTP client should get the current value of the cookie from the client. It is not recommended to save this value in a local variable for later use, as it may change during the life of the HTTP client.

2. Set the X-Secure-Modify-Token HTTP header.

   The X-Secure-Modify-Token HTTP header must be set by all Event Web Service clients when executing modify operations (PUT, POST, or DELETE). The value to set this HTTP header to is specified in the secureModifyToken cookie.

> **Note:** The client must set *all* cookies returned by the server in subsequent requests. For example, LWSSO_COOKIE_KEY and JSESSIONID are additional cookies returned by the server and must be set in subsequent requests to the server. If a new session is established, the secureModifyToken previously obtained through a GET request becomes invalid. Therefore the other cookies are also required.

Sample Code Using a Standard Java HTTP Client

The following sample code first gets the value of the `secureModifyToken` cookie and then sets the `X-Secure-Modify-Token` HTTP header.

Get the secureModifyToken cookie

The following method gets all the cookies from an initial GET request. The standard Java HTTP client does not automatically manage cookies for the user like the Apache HttpClient does. The cookies need to be obtained and managed separately.

```
private static List<HttpCookie> getCookies(final String path)
{
   final URL url = new URL(path);
   final HttpURLConnection connection = url.openConnection();
   final List<HttpCookie> result = new ArrayList<HttpCookie>();

   connection.setRequestMethod("GET");

   // Set the username and password for the request
   byte[] encodedUserPassword = Base64.encodeBase64((username + ":" +
password).getBytes());
   connection.setRequestProperty("Authorization", "Basic " + new
String(encodedUserPassword));

   connection.connect();
   int response = connection.getResponseCode();
   if (response == 200)
   {
      for (int i=1; (final String headerName =
connection.getHeaderFieldKey(i)) != null; i++)
      {
         if (headerName.equals("Set-Cookie"))
         {
            final String cookieString = connection.getHeaderField(i);
            final List<HttpCookie> cookies = HttpCookie.parse
(cookieString);
            if (cookies != null && !cookies.isEmpty())
               result.addAll(cookies);
         }
      }
   }
   return result;
}
```

Set the X-Secure-Modify-Token HTTP header

The following code adds the cookies to the POST request and the HTTP header `X-Secure-Modify-Token` if the cookie `secureModifyToken` exists.

```
final URL url
final List<HttpCookie> cookies = getCookies("http://" + localHostName
```

```
+ ":" + port + "/opr-console/rest");

final URL url = new URL("http://" + localHostName + ":" + port +
"/opr-console/rest/9.10/event_list");
final HttpURLConnection connection = url.openConnection();

// Set the cookies and HTTP header for the request
for (HttpCookie cookie : cookies)
{
   // add the cookies to the request
   connection.addRequestProperty("Cookie", cookie.getName() + "=" +
cookie.getValue());
   if (cookie.getName().equalsIgnoreCase("secureModifyToken"))
   {
      // add the HTTP header
      connection.setRequestProperty("X-Secure-Modify-Token",
cookie.getValue());
   }
}
...
```

Sample Code Using an Apache HttpClient

The following sample code first gets the value of the `secureModifyToken` cookie and then sets the `X-Secure-Modify-Token` HTTP header.

Get the secureModifyToken cookie

The following method may return null, in which case it should be assumed the target web service does not require the `X-Secure-Modify-Token` HTTP header. (For example, BSM versions lower than 9.10 do not require this HTTP header.)

```
private static String getSecureModifyToken(final HttpClient client,
final String url)

{
   int rc = -1;

   String secureModifyToken = null;

   // get the service document from the base path
   final HttpMethod getMethod = new GetMethod(url);
   getMethod.setFollowRedirects(true);
   getMethod.setDoAuthentication(true);
   getMethod.setRequestHeader("Accept", "text/plain, text/xml,
application/xml, application/atomsvc+xml");
   getMethod.setRequestHeader("Accept-Language", System.getProperty
("user.language", "en") + "-"
      + System.getProperty("user.country", "US"));
   try
   {
      client.executeMethod(getMethod);
```

```
        rc = getMethod.getStatusCode();
    }
    catch (IOException ioe)
    {
        // ignore any errors for backwards compatibility
    }
    if (rc == HttpStatus.SC_OK)
    {
        // look for the secureModifyToken
      Cookie[] cookies = client.getState().getCookies();
      if (cookies != null && cookies.length > 0)
      {
          for (Cookie cookie : cookies)
          {
              if (SECURE_MODIFY_TOKEN.equalsIgnoreCase(cookie.getName()))
                  secureModifyToken = cookie.getValue();
          }
      }
    }
    return secureModifyToken;
}
```

Set the X-Secure-Modify-Token HTTP header

```
HttpClient client = new HttpClient();
client.getState().setCredentials(new AuthScope(hostname, port),
      new UsernamePasswordCredentials(username, password));
final String secureModifyToken = getSecureModifyToken(client,
      "http://" + localHostName + ":" + port + "/opr-console/rest");
String url = "http://" + localHostName + ":" + port + "/opr-
console/rest/9.10/event_list";
PostMethod method = new PostMethod(url);
if (secureModifyToken != null)
   method.setRequestHeader("X-Secure-Modify-Token",
secureModifyToken);
...
```

Sample Code Using an Apache Wink RestClient

The following sample code first gets the value of the `secureModifyToken` cookie and then sets the `X-Secure-Modify-Token` HTTP header.

Get initial cookies

The following method gets all the cookies from an initial GET request. The Apache Wink RestClient does not automatically manage cookies for the user like the Apache HttpClient does. The cookies need to be obtained and managed separately.

```
private static Set<Cookie> getCookies(final String url, final
RestClient client)
{
    final Set<Cookie> cookies = new HashSet <Cookie>();
```

```
    final Resource resource = client.resource(url);

    // Set the username and password for the request
    byte[] encodedUserPassword = Base64.encodeBase64((username + ":" +
password).getBytes());
    resource.header("Authorization", "Basic " + new String
(encodedUserPassword));
    final ClientResponse response = resource.get();


    final MultivaluedMap<String, String> headers = response.getHeaders
();
    if (headers != null)
    {
        for (final Map.Entry<String, List<String>> header :
headers.entrySet())
        {
            if ("Set-Cookie".equalsIgnoreCase(header.getKey()))
            {
                for (final String value : header.getValue())
                {
                    if (value != null && value.length() > 0)
                    try
                    {
                        cookies.add(Cookie.valueOf(value));
                    }
                    catch (IllegalArgumentException e)
                    {
                        // ignore this entry
                    }
                }
            }
        }
    }
    return cookies;
}
```

Set the X-Secure-Modify-Token HTTP header

The following code adds the cookies to the REST resource and the HTTP header `X-Secure-Modify-Token` if the cookie `secureModifyToken` exists.

```
final RestClient client = new RestClient();
final Set<Cookie> cookies = getCookies("http://" + localHostName + ":"
+ port + "/opr-console/rest", client);

String url = "http://" + localHostName + ":" + port + "/opr-
console/rest/9.10/event_list";
final Resource resource = client.resource(url);

// Set the username and password for the request
byte[] encodedUserPassword = Base64.encodeBase64((username + ":" +
```

```
password).getBytes());
resource.header("Authorization", "Basic " + new String
(encodedUserPassword));

// Set the cookies and HTTP header for the request
for (Cookie cookie : cookies)
{
   // add the cookies to the resource
   resource.cookie(cookie);
   if (cookie.getName().equalsIgnoreCase("secureModifyToken"))
   {

      // add the HTTP header
      resource.header("X-Secure-Modify-Token", cookie.getValue());
   }
}
...
```

Disable Enhanced Security Protection

Event Web Service clients that set the `X-Secure-Modify-Token` HTTP header may fail when communicating with Operations Management Event Web Services 9.0x and lower. You may therefore disable enhanced security protection in the Web Service Settings in the Infrastructure Settings Manager.

1. Navigate to the Web Service Settings in the Infrastructure Settings Manager:

   Infrastructure Settings > Applications > Operations Management > Operations Management - Web Service Settings > Secure Modify

2. Change the default value true to **false**.

3. *Optional.* Implement the following additional measures for end users to reduce their exposure to malicious attacks when using Operations Management:

   - Do not allow your web browser to save user names and passwords.

   - Do not use the same web browser to access Operations Management and the Internet at the same time (tabbed browsing). While logged in to Operations Management, the web browser should not be used to browse other web sites.

   - HTML-enabled applications that integrate web browsers (for example email or newsreader applications) pose additional risks because simply viewing an email message or a news message may lead to the execution of an attack. Caution should be taken when using client workstations connected to Operations Management and to such applications.

Environments with CA SiteMinder

If the CA SiteMinder Web Agent is configured for `CssChecking=YES`, characters configured in the CA SiteMinder Web Agent `BadUrlChars` parameter are rejected by the CA SiteMinder Web Agent. By default the following characters are included in this list:

- Single quotation mark (')

- Greater than sign (>)

- Less than sign (<)

Event Web Service clients must not use these characters in the query parameter section of the URL:

- **String literals:** As the single quotation mark (') may be rejected in environments with CA SiteMinder, surround string literals with double quotation marks (""). See "Value Types" on page 322 for more information.

  As CA SiteMinder blocks the % encoding, the string literal itself must escape the offending characters using the dollar sign ($) instead of percent (%), for example:

  - Replace single quotation marks (') with $60.

  - Replace greater than signs (>) with $3E.

  - Replace less than signs (<) with $3C

  See "URL Escape Codes" on page 323 for more information.

- **Operators:** As the greater than sign (>) and less than sign (<) may be rejected by the CA SiteMinder agent, use the GT and LT aliases instead. See "Operator Aliases" on page 322 for more information.

REST Web Service Command-Line Utility

A REST web service command-line utility is provided which you can use to:

- Carry out simple testing of the Event Web Service.

- Perform the four basic operations: create, read, update and delete.

You can find the command-line utility in the following location:

Windows: *<HPBSM root directory>*/opr/bin/RestWsUtil.bat

Linux: *<HPBSM root directory>*/opr/bin/RestWsUtil.sh

You can use the `RestWsUtil` utility to perform the four basic operations: create, read, update and delete. Create and update require an input file with the payload to send to the REST web service. Read and delete do not set the payload. The utility uses basic authentication and has parameters to specify a username and password.

How to Call the Utility Help

To call the help for the utility, type the following command:

`RestWsUtil -help`

The usage is as follows:

```
Usage: RestWsUtil          (-h | -version |

                           (-r | -d | ((-c | -u) <filename> [-content_
                           type <type>])

                           [-o <filename>]

                           (([-ssl] [-server <server>] [-p <port>]) | [-
                           u <URL>]))

                           [-username <login name>] [-password
                           <password>] [-v]
```

The options for the `RestWsUtil` utility are as follows:

`-c,-create <in_file>`	Create the resource in the specified XML document. HTTP POST operation.
`-content_type <type>`	HTTP header content-type value to set for create and update operations. Default is `application/xml`.
`-d,-delete`	Delete the specified resource. The URL directly addresses the resource to delete. HTTP DELETE operation.
`-h,-help`	Print this message and exit.
`-o,-output <out_file>`	Name of output file for text returned from the web service request. Default is `stdout`.

`-p,-port <port>`	Set the port number. Default port is `80` for HTTP and `443` for HTTPS. This option cannot be specified in conjunction with the option `url`.
`-password <password>`	Password for the specified user.
`-r,-read`	Read the specified resource. HTTP GET operation.
`-server <server>`	Set target gateway server. The value may be a hostname or IP address of a gateway server. This option cannot be specified in conjunction with the option `url`.
`-ssl`	Set the protocol to HTTPS. Default is to use HTTP. This option cannot be specified in conjunction with the option `url`.
`-update <in_file>`	Update the resource with the specified XML document changes. HTTP PUT operation.
`-url <URL>`	URL of the gateway server. This option cannot be specified in conjunction with the options `ssl`, `server`, or `port`.
`-username <login name>`	User login name required for authentication.
`-v,-verbose`	Print verbose output.
`-version`	Print the version information and exit.

Possible exit status values are:

0	Successful completion.
1	Failure.
3	Redirection (300-399).
4	Client error (400-499).
5	Internal Server Error (500-599).

Examples

Some examples of usage of the `RestWsUtil` utility follow.

Reading Events

Here is an example of using the `RestWsUtil` command-line utility to read events and to send the results to an output file called `test.xml`.

```
RestWsUtil -r -username admin -o test.xml -verbose

Password: ******

INFO: Read the resource located at: http://bsmserver.example.com/opr-console/rest/9.10/event_
list

INFO: Operation successful.
```

Reading Custom Attributes of an Event

To read the custom attributes of an event, you need to specify the complete URL, including the ID of the event you want to read the attributes for.

The following example sends the output to `stdout`, which is the default.

```
RestWsUtil -r
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/0695624b-93fa-40b1-8b0fc9b4ea07a4ec/custom_attribute_list
-username admin -verbose
Password: ******
INFO: Read the resource located at: http://localhost/opr-console/rest/9.10/event_
list/0695624b-93fa-40b1-8b0f-c9b4ea07a4ec/custom_attribute_list

<?xml version="1.0"encoding="UTF-8"standalone="yes"?>
<custom_attribute_list xmlns="http://www.hp.com/2009/software/opr/data_model"
   self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-93fa-40b1-8b0f-
c9b4ea07a4ec/custom_attribute_list"
   type="urn:x-hp:2009:software:data_model:opr:type:event:custom_attribute_list"version="1.0">
  <custom_attribute
    self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-93fa-40b1-
8b0fc9b4ea07a4ec/custom_attribute_list/CiResolverSimilarityMetric"
    type="urn:x-hp:2009:software:data_model:opr:type:event:custom_attribute"version="1.0">
    <name>CiResolverSimilarityMetric</name>
    <value>100</value>
  </custom_attribute>
</custom_attribute_list>

INFO: Operation successful.
```

Reading Annotations of an Event

To read annotations of an event, you need to specify the complete URL, including the ID of the event you want to read the annotations for.

The following example sends the output to `stdout`, which is the default.

```
-username admin -verbose
Password: ******
INFO:  Read the resource located at:
       http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list

<?xml version="1.0" encoding="UTF-8"?>
<annotation_list xmlns="http://www.hp.com/2009/software/opr/data_model"
   self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list"
   type="urn:x-hp:2009:software:data_model:opr:type:event:annotation_list" version="1.0">
  <annotation
    self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list/74b869a8-399c-42cb-854c-03b9ced975c3"
    type="urn:x-hp:2009:software:data_model:opr:type:event:annotation" version="1.0">
    <id>74b869a8-399c-42cb-854c-03b9ced975c3</id>
    <event_ref
      target_role="urn:x-hp:2009:software:data_model:opr:relationship:event:is_related_
to:event"
      type="urn:x-hp:2009:software:data_model:opr:type:reference:event:event_ref"
version="1.0">
      <target_id>10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_id>
      <target_type>urn:x-hp:2009:software:data_model:opr:type:event</target_type>
      <target_href>http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-
41d6-b933-ed84b4f5e43e</target_href>
    </event_ref>
    <author>admin</author>
    <time_created>2010-07-13T14:18:14.860+02:00</time_created>
    <text>Some annotation text</text>
```

```
    </annotation>
</annotation_list>


INFO:  Operation successful.
```

Creating a Custom Attribute

Here is an example showing how to create a new custom attribute called `MyNewCA` for a specified event.

```
RestWsUtil -c newca.xml
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/0695624b-93fa-40b1-8b0fc9b4ea07a4ec/custom_attribute_list
-username admin -verbose
Password: *******
INFO: Create the resource located at: newca.xml

<?xml version="1.0"encoding="UTF-8"standalone="yes"?>
<custom_attribute xmlns="http://www.hp.com/2009/software/opr/data_model"
    self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-93fa-40b1-8b0f-
c9b4ea07a4ec/custom_attribute_list/MyNewCA"
    type="urn:xhp:2009:software:data_model:opr:type:event:custom_attribute"version="1.0">
  <name>MyNewCA</name>
  <value>100</value>
</custom_attribute>


INFO: Operation successful.
```

The new custom attribute and its value is written to `stdout`.

The contents of the `newca.xml` file look like this:

```
<custom_attribute xmlns="http://www.hp.com/2009/software/opr/data_model">
  <name>MyNewCA</name>
  <value>100</value>
</custom_attribute>
```

Creating an Annotation

Here is an example showing how to create an annotation. The new annotation is written to the `newanno.xml` file.

```
RestWsUtil -c newanno.xml
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list
-username admin -verbose
Password: ******
INFO:  Create the resource located at: newanno.xml

<?xml version="1.0" encoding="UTF-8"?>

<annotation
  xmlns="http://www.hp.com/2009/software/opr/data_model"
  self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list/62f86310-38ce-4793-ab3f-23ecfb3f2a67"
  type="urn:x-hp:2009:software:data_model:opr:type:event:annotation" version="1.0">
  <id>62f86310-38ce-4793-ab3f-23ecfb3f2a67</id>
  <event_ref target_role="urn:x-hp:2009:software:data_model:opr:relationship:event:is_related_
to:event"
    type="urn:x-hp:2009:software:data_model:opr:type:reference:event:event_ref" version="1.0">
    <target_id>10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_id>
    <target_type>urn:x-hp:2009:software:data_model:opr:type:event</target_type>
    <target_href>http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-
```

```
41d6-b933-ed84b4f5e43e</target_href>
  </event_ref>
  <author>admin</author>
  <time_created>2010-07-13T14:56:56.642+02:00</time_created>
  <text>Some new annotation text</text>
</annotation>

INFO:  Operation successful.
```

The contents of the `newanno.xml` file look like this:

```
<annotation xmlns="http://www.hp.com/2009/software/opr/data_model" >
  <author>admin</author>
  <text>Some new annotation text</text>
</annotation>
```

Updating a Custom Attribute

Here is an example showing how to update a custom attribute called `MyNewCA` with a new value for a specified event.

```
RestWsUtil -update updateca.xml
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/0695624b-93fa-40b1-8b0fc9b4ea07a4ec/custom_attribute_list/MyNewCa
-username admin -verbose
Password: ******
INFO: Update the resource with changes located at: updateca.xml

<?xml version="1.0"encoding="UTF-8"standalone="yes"?>
<custom_attribute xmlns="http://www.hp.com/2009/software/opr/data_model"
   self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-93fa-40b1-8b0f-
c9b4ea07a4ec/custom_attribute_list/MyNewCA"
   type="urn:x-hp:2009:software:data_model:opr:type:event:custom_attribute"version="1.0">
  <name>MyNewCA</name>
  <value>999</value>
</custom_attribute>

INFO: Operation successful.
```

The updated value for the custom attribute is written to `stdout`.

The contents of the `updateca.xml` file look like this:

```
<custom_attribute xmlns="http://www.hp.com/2009/software/opr/data_model">
  <value>999</value>
</custom_attribute>
```

Updating an Annotation

Here is an example showing how to update an annotation. The updated annotation is written to the `updateanno.xml` file.

```
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list/582f0488-15ad-40ac-907f-fec21041b5c0
-username admin -verbose
Password: ******
INFO: Update the resource with changes located at: updateanno.xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<annotation xmlns="http://www.hp.com/2009/software/opr/data_model"
  self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list/582f0488-15ad-40ac-907f-fec21041b5c0"
  type="urn:x-hp:2009:software:data_model:opr:type:event:annotation" version="1.0">
```

```
<id>582f0488-15ad-40ac-907f-fec21041b5c0</id>
<event_ref
  target_role="urn:x-hp:2009:software:data_model:opr:relationship:event:is_related_to:event"
  type="urn:x-hp:2009:software:data_model:opr:type:reference:event:event_ref" version="1.0">
  <target_id>10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_id>
  <target_type>urn:x-hp:2009:software:data_model:opr:type:event</target_type>
  <target_href>http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-
41d6-b933-ed84b4f5e43e</target_href>
</event_ref>
<author>admin</author>
<time_created>2010-07-13T15:56:31.220+02:00</time_created>
<text>Updated annotation text</text>
</annotation>

INFO: Operation successful.
```

The contents of the `updateanno.xml` file look like this:

```
<annotation xmlns="http://www.hp.com/2009/software/opr/data_model">
  <text>Updated annotation text</text>
</annotation>
```

Updating the Title of an Event

Here is an example showing how to modify the title of an event. The modified title is written to the `update.xml` file.

```
RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/d36a157e-7312-4302-a2da-e5b7230b0e21
Password: *******

INFO:  Operation successful.
```

The contents of the `update.xml` file look like this:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <title>New title goes here</title>
</event>
```

Updating the Lifecycle State of an Event

Here is an example showing how to modify the lifecycle state of an event. This works in the same way as modifying the title of an event. The modified lifecycle state is written to the `update.xml` file.

```
RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/d36a157e-7312-4302-a2da-e5b7230b0e21
Password: *******

INFO:  Operation successful.
```

The contents of the `update.xml` file look like this:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <state>in_progress</state>
</event>
```

Updating the Lifecycle State and the Severity of an Event

Here is an example showing how to modify both the lifecycle state and the severity of an event. This works in the same way as any other updates to an event. The modified lifecycle state and

severity is written to the `update.xml` file.

```
RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/d36a157e-7312-4302-a2da-e5b7230b0e21
Password: *******

INFO:  Operation successful.
```

The contents of the `update.xml` file look like this:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <state>in_progress</state>
  <severity>critical</severity>
</event>
```

Updating the Event with Transfer Control to Connected Server Information

Here is an example showing how to update an event with the information that control of the event has been transferred to a connected server. This information is written to the `update.xml` file.

```
RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/d36a157e-7312-4302-a2da-e5b7230b0e21
Password: *******
INFO:  Operation successful.
```

The contents of the `update.xml` file look like this:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <control_transferred_to>
    <name>logger</name>
  </control_transferred_to>
</event>
```

Bulk Event Update: Updating the State and the Severity of an Event

Here is an example where all events with the title set to "DB down" have their severity set to `in_progress`. A list of the updated events is returned to the caller.

```
RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list?query=title='DB down'
Password: *******
INFO:  Operation successful.
```

The contents of the `update.xml` file look like this:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <state>in_progress</state>
  <severity>critical</severity>
</event>
```

Deleting a Custom Attribute

Here is an example of how to delete a custom attribute called `MyNewCA` from the list of custom attributes for a selected event.

```
RestWsUtil -d
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/0695624b-93fa-40b18b0fc9b4ea07a4ec/custom_attribute_list/MyNewCa -username admin -verbose
Password: ******
INFO: Deleting resource located at: http://localhost/opr-console/rest/9.10/event_
list/0695624b-93fa-40b1-8b0f-c9b4ea07a4ec/custom_attribute_list/MyNewCa
```

```
<?xml version="1.0"encoding="UTF-8"standalone="yes"?>
<custom_attribute xmlns="http://www.hp.com/2009/software/opr/data_model"
  self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-93fa-40b1-8b0f-
c9b4ea07a4ec/custom_attribute_list/MyNewCa"
  type="urn:x-hp:2009:software:data_model:opr:type:event:custom_attribute"version="1.0">
  <name>MyNewCa</name>
  <value>999</value>
</custom_attribute>

INFO: Operation successful.
```

Deleting an Annotation

Here is an example of how to delete an annotation from the list of annotations for a selected event.

```
RestWsUtil -d
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list/582f0488-15ad-40ac-907f-fec21041b5c0
-username admin -verbose
Password: ******
INFO: Deleting resource located at: http://mambo.mambo.net/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list/582f0488-15ad-40ac-907f-fec21041b5c0

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<annotation xmlns="http://www.hp.com/2009/software/opr/data_model"
  self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list/582f0488-15ad-40ac-907f-fec21041b5c0"
  type="urn:x-hp:2009:software:data_model:opr:type:event:annotation" version="1.0">
  <id>582f0488-15ad-40ac-907f-fec21041b5c0</id>
  <event_ref
    target_role="urn:x-hp:2009:software:data_model:opr:relationship:event:is_related_to:event"
    type="urn:x-hp:2009:software:data_model:opr:type:reference:event:event_ref" version="1.0">
    <target_id>10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_id>
    <target_type>urn:x-hp:2009:software:data_model:opr:type:event</target_type>
    <target_href>http://mambo.mambo.net:80/opr-console/rest/9.10/event_list/10d9a54f-54b9-
41d6-b933-ed84b4f5e43e</target_href>
  </event_ref>
  <author>admin</author>
  <time_created>2010-07-13T15:56:31.220+02:00</time_created>
  <text>Updated annotation text</text>
</annotation>

INFO: Operation successful.
```

Creating an Event

Here is an example of how to create a new event.

```
RestWsUtil -create create.xml -username admin -password ***** -url
http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
```

The contents of the `create.xml` file look like this:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>New event title</title>
   <severity>normal</severity>
   <priority>low</priority>
   <state>open</state>
</event>
```

Creating a List of Events

Here is an example of how to create a list of events.

```
RestWsUtil -create createlist.xml -content_type "application/xml; type=collection" -username
admin -password ***** -url http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
```

The contents of the `createlist.xml` file look like this:

```
<event_list xmlns="http://www.hp.com/2009/software/opr/data_model">
   <event>
      <title>Major Event</title>
      <severity>major</severity>
   </event>
   <event>
      <title>Minor Event</title>
      <severity>minor</severity>
   </event>
</event_list>
```

> **Note:** The HTTP response code from this call will be `202 Accepted`. This means that the POST operation looks OK and the request will be submitted, but the resource has not yet been created. New events must be submitted through the event pipeline, and may or may not be eventually stored in the event database. For example, the submitted event may be de-duplicated in the event pipeline. Submission of events through this interface is subject to all the standard restrictions and must be processed through the event pipeline.

Event Web Service Query Language

The Event Web Service uses standard query parameters for its URL query language.

Query parameters are used to modify the response from a resource in some way. For all resources, the response media type can be controlled. If a resource is a collection, the set of entries returned in the collection can be filtered in a variety of ways. Query parameters can be combined in interesting ways to further constrain a response.

The Event Web Service offers a number of parameters for filtering the event list by specific criteria. For example, you can filter the results by matching specific event parameters to a condition in a URL query parameter. It is also possible to reduce the list size by displaying only a certain number of list items and page through them. You can also send a time and date parameter which returns only those events that have been updated after the specified time and date.

HTTP Query Parameters

This section describes the HTTP query parameters that are supported by the Event Web Service.

These query parameters apply only to collection resources and are only meaningful when used with the HTTP GET method, or when updating events in bulk with the HTTP PUT method. They are specified within the HTTP query portion of the URL that addresses the resources.

The following are a list of parameters that may be specified in the HTTP query portion of the URL. Query parameters can be specified in combination, separated by the ampersand (&) operator.

Query Parameter	Description
watermark	Parameter for filtering by date and time. For details, see "Filtering by Date and Time: watermark" on the next page.
query	Parameter for filtering by event attributes. For details, see "Filtering by Event Attributes: query" on page 314.
start_index	Paging query parameter used to define the item from which to start the query. For details, see "Paging" on page 315.
page_size	Paging query parameter used to define how many items are displayed per page of an atom feed. For details, see "Paging" on page 315.
order_by	Ordering parameter used to specify that the response feed should be ordered by the indicated field. For details, see "Ordering" on page 316.

Query Parameter	Description
`order_direction`	Ordering parameter used to specify whether the response feed should be ordered in either descending or ascending order. For details, see "Ordering" on page 316.
`include_closed`	Parameter used to specify whether closed events are included when querying the Events Service (`event_list`). Default value: `false` For details, see "Data Inclusion" on page 317.
`include_relationships`	Parameter used to specify whether relationships are included when querying the Events Service (`event_list`). Default value: `true` For details, see "Data Inclusion" on page 317.

A client can specify that the response feed should be filtered by only including resources where the meta-data or data for the resource matches certain criteria. This can be specified using the query parameter within the HTTP query portion of the URL.

"Query Filter Criteria Properties" on page 318 lists the available query filter criteria properties and supported operators.

Filtering by Date and Time: watermark

A client can specify that the response feed should be filtered based on time. These query parameters apply only to collection resources and are only meaningful when used with the HTTP GET method.

You can query items by time and date using the `watermark` parameter. If the `watermark` parameter is specified, only events that have been created or updated after the specified time are returned.

For instance, `watermark=2009-01-01T00:00:00Z` indicates that the only resources to be included in the response feed have been updated after the beginning of 2009.

The `watermark` query parameter is used to obtain the most recently updated resources in a collection. Like all times specified in a query, the value for the `watermark` parameter must be specified in the XML Schema dateTime format. If there are no resources in the collection that have been updated after the specified date, the response feed will be empty. A poorly formatted value for the query parameter (for example, one that does not conform to the XML Schema dateTime format) will cause a fault response.

For more details about XML Schema data types, refer to the XML schema document, which you can find in the following location:

http://www.w3.org/TR/xmlschema-2

Additionally when specifying a time zone other than GMT using "Z", you must specify a "+" character. Since this is in the URL it must be URL encoded.

For a list of URL escape codes for characters that must be escaped, see "URL Escape Codes for Characters that Must Be Escaped" on page 324.

For full details about URL encoding, refer to the Uniform Resource Locators (URL) Specification at the following location:

http://www.rfc-editor.org/rfc/rfc1738.txt

The need to escape certain characters means that, for example, any "plus" character must be replaced with "%2B". An example would be:

```
query=time_createdGT2009-10-19T17:06:54.453%2B02:00
```

An example of a URL call where only events after 13:59:17 on March 19, 2010, are listed would look like this:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?alt=atom&watermark=2010-03-19T13:59:17%2B02:00
```

Filtering by Event Attributes: query

The `query` parameter facilitates filtering the request using specific event attribute values. The criteria for the `query` filter are:

- A filter property specifying an event attribute

- A supported operator

- A value for the property

Simple Filters

As a simple example, to filter all the queries where the `severity` property equals `critical`, the web service client would need to request the following URL:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?alt=atom&query=severity%20EQ%20"critical"
```

Compound Filters

You can compose multiple filters for more complex filter queries. Filter query parameters can be used in combination, separated by logical ANDs, NOTs, and ORs. Logical ANDs are resolved first. To ensure the correct processing of the query, use parentheses.

Some examples of compound filters follow.

The first example shows two simple filters made into a compound filter query using the logical AND:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=assigned_user%20EQ%20"admin"%20AND%20title%20EQ%20"My
Title"
```

The following example shows a compound filter query made up of a combination of a simple and a complex filter, using a logical AND:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=assigned_user%20EQ%20"admin"%20AND%20related_ci[target_
id%20EQ%20"ffca22eea15268533029f17b4c01b008"]
```

The next example shows how you would compose a compound filter so that the logical ORs are resolved first.

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=assigned_user%20EQ%20"admin"%20AND%20(title%20EQ%20"My
Title"%20OR%20state%20EQ%20"closed")
```

The next example shows how you would compose a compound filter so that the logical NOTs are resolved first.

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=assigned_user%20EQ%20"admin"%20AND%20(title%20EQ%20"My
Title"%20NOT%20state%20EQ%20"closed")
```

Paging

You can optimize the query result by specifying some additional URL parameters. Paging query parameters are used to filter the entries in a response feed to a subset of those that would be returned without the query parameters. Paging query parameters apply to collection resources and will have no affect on an individual resource. They are only meaningful when used with the HTTP GET method. Paging query parameters are applied to a response feed after considering all other filtering query parameters.

There are two query parameters that work together to provide an interface that a client can use to page through a feed with a large number of entries: `start_index`, and `page_size`.

start_index

The `start_index` query parameter is used to specify the index of the first entry returned in a response feed. So you can define the item from which to start the query with the `start_index` parameter. The first entry in a feed has an index of 1, the second entry an index of 2, and so on. The default value for `start_index` is always 1 if the query parameter is not specified. The request returns a fault if a value less than 1 is specified. A value greater than the number of entries in the collection will return an empty response feed.

page_size

The `page_size` query parameter is used to specify the number of entries returned at one time. So you can define how many items are displayed (on one page of an atom feed) with the `page_size` parameter. The default for the Event Web Service is set to 20 items.

The minimum value is 1 and the maximum value can be any number that the service is able to support. There is no default value that all applications are expected to use when the `page_size` parameter is not set. The request should be faulted if a value less than 1 is specified.

Combining start_index and page_size

By combining the `start_index` and the `page_size` parameters, you can view the result distributed over more than one page. The value of `start_index` must be greater than 0.

For example, if you call the URL with a `page_size` of 5, but do not define the `start_index` value, the first five items (items 1 - 5) will be returned.

If you call the URL with a `page_size` of 5, and define the `start_index` value as 6, five items starting from the sixth item (items 6 - 10) will be returned on the second page. So the result of the query would be that 10 items are returned, distributed over two pages of an atom feed.

So, to return the first page (items 1 - 5) of query results filtered by the `severity` parameter set to equal `warning`, you would call the following URL:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?alt=atom&query=severity%20EQ%20"warning"&page_size=5
```

To get the second page (items 6 - 10) of query results, you would call the following URL:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?alt=atom&query=severity%20EQ%20"warning"&page_size=5&start_
index=6
```

A client using paging can keep the page size constant for every request or can vary the page size for every request. The pages can also overlap. In the examples above with `start_index` set to 3 and `page_size` set to 5, the requested page will overlap with the first page. There is no guarantee that a client will not see duplicate entries, or that a client may miss some entries due to additions or deletions between page requests.

Links are set in all feeds to indicate how to get to the first or last page. When not all of the entries are returned due to paging, links also indicate how to get the next and previous pages. The links are described by setting the `rel` attribute to "first", "last", "next", or "previous". These links are described in RFC5005, which you can access at the following location:

http://tools.ietf.org/html/rfc5005

Ordering

A client can specify that a response feed should be returned with certain ordering criteria. These query parameters apply only to collection resources and are only meaningful when used with the HTTP GET method.

order_by

The `order_by` query parameter is used to specify that the response feed should be ordered by the indicated field. The field may be any simple data or meta-data property of the resource.

If the `order_by` query parameter is a time or the `sequence_number`, then the default ordering is descending, so that the newest entries appear first, otherwise it is ascending.

order_direction

The `order_direction` query parameter is used to specify that the response feed should have an order direction as indicated. There are only two valid values for this query parameter: "ascending", and "descending". Any other value causes a fault response. The default value if none of the ordering query parameters is specified is descending (the APP specification indicates that a feed is ordered descending by update time).

Data Inclusion

With data inclusion query parameters, a client can control the amount of relevant data returned by the query, thereby directly influencing the performance of the response.

include_closed

The `include_closed` query parameter is used to specify whether closed events are included when querying the Events Service (`event_list`). The default value of this parameter is `false`. If set to `true`, closed events are included in the query, otherwise only events that do not have a lifecycle state of `closed` are returned from the Event Web Service.

include_relationships

The `include_relationships` query parameter is used to specify whether relationships are included when querying the Events Service (`event_list`). The default value of this parameter is `true`. If set to `false`, relationships are not included in the query. For example, if `related_ci` or `source_ci` is set in the event, and the `include_relationships` query parameter is set to `false`, only the CI IDs are returned from the Event Web Service. Key attributes, including the container CI (`part_of`), are not resolved or returned, in contrast to the case when the `include_relationships` parameter is set to the default value `true`.

Media Type

A client can request that a response be returned with a specified media type.

alt

The `alt` query parameter is used by a client to tell a server that it would like to have the response returned using the specified media type. This query parameter applies to all resources, including collections, and can be used with any HTTP method that returns a response. The value of the parameter is a media type for example, application/atom+xml, application/json, application/xml, and so on). Usually, a service will have a limited set media types that it is able to format a given resource. The list of supported media types is available from the Atom response format (for example, `alt=atom`, `alt=json`,`alt=xml`), which is supported by most resources (see "How to Detect New Events" on page 283).

The `alt` query parameter expresses the same client desire as the Accept HTTP header. The advantage of using the HTTP header is that many clients already have built-in support for the Accept header. The advantage of using the `alt` query parameter is that a link with a specific response format can be passed around in email, chat, twitter, documents, etc. without also having to say that you need to set the Accept header to a certain value. When the service receives a

message having both the Accept header and the `alt` query parameter set, the `alt` query
parameter takes precedence.

Query Filter Criteria Properties

"Query Filter Criteria Properties" below lists the available query filter criteria properties and
supported operators.

> **Note:** If you want to filter using complex (nested) attributes, you need to use brackets ([ ]) to
> show the nesting. When specifying complex attributes, the bracket characters ([ ]) must be
> escaped, for example, like this: `query=related_ci%5Btarget_`
> `id="ffca22eea15268533029f17b4c01b008"%5D`. For more information about filtering
> using complex attributes, see "Complex Attributes" on page 325.

Query Filter Criteria Properties

Property	Supported Operators	Type of Value	Supports `order_by`	Default for `order_direction`
`application`	`EQ, NE, LIKE, IS NULL, IN`	String Literal	yes	ascending
`assigned_group`	`EQ, NE, IN`	String Literal	yes	ascending by ID
`assigned_group[id]`	`EQ, NE, IN`	String Literal	yes	ascending by ID
`assigned_group[name]`	`EQ, NE, IN`	String Literal	yes	ascending by ID
`assigned_user`	`EQ, NE, IN`	String Literal	yes	ascending by ID
`assigned_user[id]`	`EQ, NE, IN`	String Literal	yes	ascending by ID
`assigned_user[login_ name]`	`EQ, NE, IN`	String Literal	yes	ascending by ID
`category`	`EQ, NE, LIKE, IS NULL, IN`	String Literal	yes	ascending
`cause[target_id]`	`EQ, IS NULL, IN`	String Literal (UUID)		
`close_key_pattern`	`EQ, NE, LIKE, IS NULL, IN`	String Literal		

Property	Supported Operators	Type of Value	Supports order_by	Default for order_direction
control_transferred_to[dns_name]	EQ, NE, LIKE	String Literal		
control_transferred_to[external_id]	EQ, NE, LIKE, IS NULL, IN	String Literal		
control_transferred_to[id]	EQ, NE, LIKE, IS NULL, IN	String Literal (UUID)		
control_transferred_to[state]	EQ, NE, IS NULL, IN	String Literal (UUID)		
control_transferred_to[state]	EQ, NE, IS NULL, IN	String Literal (UUID)		
ca[<CA name>]	EQ, NE, LIKE, IN	String Literal		
ca[<CA name>]	EQ, NE, LIKE, IN	String Literal		
custom_attribute_list[<CA_name>]	EQ, NE, LIKE, IN	String Literal		
description	EQ, NE, LIKE, IS NULL, IN	String Literal	yes	ascending
duplicate_count	EQ, LT, GT, LTE, GTE, NE, IN	Integer	yes	ascending
eti_hint	EQ, NE, LIKE, IS NULL, IN	String Literal		
eti_value_id	EQ, NE, LIKE, IS NULL, IN	String Literal		
id	EQ, NE, IN	String Literal (UUID)		
instruction_available		Boolean		
key	EQ, NE, LIKE, IS NULL, IN	String Literal		

Property	Supported Operators	Type of Value	Supports order_by	Default for order_ direction
log_only	EQ	Boolean		
node[target_global_id]	EQ, NE, IS NULL, IN	String Literal		
node[target_id]	EQ, NE, IS NULL, IN	String Literal		
node_hints[hint]	EQ, NE, LIKE, IS NULL, IN	String Literal		
node_hints[node_core_id]	EQ, NE, LIKE, IS NULL, IN	String Literal		
node_hints[node_dns_name]	EQ, NE, LIKE, IS NULL, IN	String Literal		
node_hints[node_ip_address]	EQ, NE, LIKE, IS NULL, IN	String Literal		
object	EQ, NE, LIKE, IS NULL, IN	String Literal	yes	ascending
om_service_id	EQ, NE, LIKE, IS NULL, IN	String Literal		
om_user	EQ, NE, LIKE, IS NULL, IN	String Literal	yes	ascending
original_data	EQ, NE, LIKE, IS NULL, IN	String Literal		
original_id	EQ, NE, LIKE, IS NULL, IN	String Literal		
priority	EQ, NE, LIKE, IS NULL, IN	String Literal	yes	ascending
received_as_notify	EQ	Boolean		
received_on_ci_downtime	EQ	Boolean		
related_ci[sub_component]	EQ, NE, LIKE, IS NULL, IN	String Literal		
related_ci[sub_component]	EQ, NE, LIKE, IS NULL, IN	String Literal		

Property	Supported Operators	Type of Value	Supports `order_by`	Default for `order_direction`
`related_ci[target_global_id]`	EQ, NE, IS NULL, IN	String Literal		
`related_ci[target_id]`	EQ, NE, LIKE, IS NULL, IN	String Literal		
`related_ci_hints[hint]`	EQ, NE, LIKE, IS NULL, IN	String Literal		
`sequence_number`	EQ, LT, GT, GTE, LTE, NE, IN	Integer	yes	descending
`severity`	EQ, NE, IS NULL, IN	String Literal	yes	ascending
`skip_duplicate_suppression`	EQ	Boolean		
`solution`	EQ, NE, LIKE, IS NULL, IN	String Literal	yes	ascending
`source_ci[target_global_id]`	EQ, NE, IS NULL, IN	String Literal		
`source_ci[target_id]`	EQ, NE, IS NULL, IN	String Literal		
`source_ci_hints[hint]`	EQ, NE, LIKE, IS NULL, IN	String Literal		
`source_ci_hints[node[core_id]]`	EQ, NE, LIKE, IS NULL, IN	String Literal		
`source_ci_hints[node[dns_name]]`	EQ, NE, LIKE, IS NULL, IN	String Literal		
`source_ci_hints[node[ip_address]]`	EQ, NE, LIKE, IS NULL, IN	String Literal		
`sourced_from[dns_name]`	EQ, NE, LIKE, IS NULL, IN	String Literal		
`sourced_from[external_id]`	EQ, NE, LIKE, IS NULL, IN	String Literal		
`sourced_from[id]`	EQ, NE, IS NULL, IN	String Literal (UUID)		

Property	Supported Operators	Type of Value	Supports order_by	Default for order_direction
state	EQ, NE, IS NULL, IN	String Literal	yes	ascending
sub_category	EQ, NE, LIKE, IS NULL, IN	String Literal	yes	ascending
time_changed	LT, GT, GE, LE	Datetime	yes	descending
time_created	LT, GT, GE, LE	Datetime	yes	descending
time_received	LT, GT, GE, LE	Datetime	yes	descending
time_state_changed	LT, GT, GE, LE	Datetime	yes	descending
title	EQ, NE, LIKE, IS NULL, IN	String Literal	yes	ascending
type	EQ, NE, LIKE, IS NULL, IN	String Literal	yes	ascending

Operator Aliases

"Operator Aliases" below lists the verbose aliases for operators.

Operator Aliases

Operator	Alias
=	EQ
!=	NE
<	LT
<=	LTE
>	GT
>=	GTE
LIKE	n/a

Value Types

The query filter criteria properties listed in "Query Filter Criteria Properties" on page 318 can take the following types of values:

Value Types

Value Type	Description
String Literals	If you want to filter using string literals, you need to surround them with double quotation marks ("").
	If you need to use one of the characters that must be escaped in a string literal (see "URL Escape Codes for Characters that Must Be Escaped" on the next page, you need to escape the character using the dollar sign ($) instead of percent (%); for example, use `query=title EQ "$3CMy title$3E"` instead of `query=title EQ '%3CMy title%3E'`.
	Note: OMi 9.0x and lower only supports single quotation marks (') and the percent (%) escape codes. String literal escape codes using the dollar sign ($) are not supported in OMi 9.0x. OMi 9.10 and higher supports single (') *and* double (") quotation marks. It is recommended that Event Web Service clients use double quotation marks.
Datetime	Must be specified in the XML Schema dateTime format. For more details about XML Schema data types, refer to the XML schema document, which you can find in the following location: http://www.w3.org/TR/xmlschema-2
	Additionally, when specifying a time zone other than GMT using "Z", you must specify a "+" character. Since this is in the URL it must be URL encoded. See "URL Escape Codes for Characters that Must Be Escaped" on the next page.
Integer	Positive and negative natural numbers, including 0.

Query Using the POST Method

A limitation may exist for the maximum length of the URLs accepted by servers. If this length is exceeded, a 414 status is returned indicating `Request-URI Too Long`. It is more common for an intermediary to exhibit this type of limitation. When a client receives this status in a response, it is very probably caused by a long query expression. The client should re-try the request with the following changes:

- Change the HTTP method from `GET` to `POST`

- Set the Content-Type header to `application/x-www-form-urlencoded`

- Remove the `query` query parameter from the URL and set the body of the request to the `query` query parameter. For example, the message body could be as follows: `query=severity='critical'.`

- Re-send the request

URL Escape Codes

"URL Escape Codes for Characters that Must Be Escaped" on the next page lists the characters that must be escaped in URLs.

If you need to escape a character in a string literal, you need to use the dollar sign ($) instead of percent (%); for example, use `query=title EQ "$3CMy title$3E"` instead of `query=title EQ '%3CMy title%3E'`.

URL Escape Codes for Characters that Must Be Escaped

Character	URL Escape Codes	String Literal Escape Code
SPACE	%20	$20
<	%3C	$3C
>	%3E	$3E
#	%23	$23
%	%25	$25
+	%2B	$2B
{	%7B	$7B
}	%7D	$7D
|	%7C	$7C
\	%5C	$5C
^	%5E	$5E
~	%7E	$7E
[	%5B	$5B
]	%5D	$5D
`	%60	$60
;	%3B	$3B
/	%2F	$2F
?	%3F	$3F
:	%3A	$3A
@	%40	$40
=	%3D	$3D
&	%26	$26
$	%24	$24

White Spaces in URLs

You can use the following characters or strings to represent white space in the query portion of a URL:

- White space ( )

- Plus sign (+)

- URL escape code (%20)

- String literal escape code ($20)

Complex Attributes

If you want to filter using complex (nested) attributes, you need to use square brackets ([ ]) to show the nesting.

Complex attribute filters allow multiple nesting for subjacent attributes and can also be combined with other filters.

Here are examples of URL calls containing a complex attribute:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=related_ci%5Btarget_
id="ffca22eea15268533029f17b4c01b008"%5D
```

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=custom_attribute_list%5BMyCaName%20NE%20""%5D
```

> **Note:** When specifying complex attributes, the bracket characters ([ ]) must be escaped. For more information about URL escape codes, see "URL Escape Codes" on page 323.

Editable Properties

Event properties that you can edit within the Event Web Service are listed in "List of Editable Event Properties" below.

> **Note:** This is a summary of the com.hp.opr.ws.model.event Java API Documentation. For the latest information about editable properties, refer to the Java API Documentation.
>
> See "File Locations" on page 328 for access information for the Java API Documentation.

List of Editable Event Properties

Name of Tag	Type	Description	Comment
title	String	Title of the event	

Name of Tag	Type	Description	Comment
`description`	String	Description of the event	
`solution`	String	A solution	
`state`	String	Lifecycle state of the event	Value can be one of: `closed`, `in_progress`, `open`, `resolved`
`severity`	String	Severity of the event	Value can be one of: `major`, `minor`, `critical`, `warning`, `normal`, `unknown`
`priority`	Integer	Priority of the event	
`assigned_user`	User	User assigned to the event	Example: `<name>User</name>`
`assigned_group`	Group	User group assigned to the event	Example: `<name>Users</name>`
`cause`	ID	Identifier of the cause	
`control_transferred_to`	ID or name	ID or name of the connected server	Set in the `control_transferred_to` structure
`symptom_list`	List	List of symptoms	Symptoms can only be created, updated, or deleted using the Symptom web service
`custom_attribute_list`	List	List of custom attributes	Custom attributes can only be created, updated, or deleted using the Custom Attribute web service
`annotation_list`	List	List of annotations	Annotations can only be created, updated, or deleted using the Annotation web service
`auto_action`	None	Automatic action	POST to run the action, DELETE to stop the action using the Automatic Action web service only. Note that the body of the XML is empty.
`user_action`	None	Operator action	POST to run the action, DELETE to stop the action using the Operator Action web service only. Note that the body of the XML is empty.

History Lines

History lines enable you to track changes for an event. History lines provide information about the types, names, and previous and current values of event properties that changed for a specific

event. Concurrent changes are represented as a single history line. They are available in read-only mode, and are generated automatically.

Recorded Property Changes

Event properties that are recorded when they are changed in the form of a history line are listed in "Recorded Property Changes" below. In general, the following properties are recorded for each change: property name, previous value, current value, and time of change. For details on each of these properties, see the appropriate Javadoc for the Change Type listed in the package `com.hp.opr.api.ws.model.event.property`.

Recorded Property Changes

Name	ChangeType
`annotation`	`OprAnnotationPropertyChange`
`application`	`OprStringPropertyChange`
`assigned_group`	`OprGroupPropertyChange`
`assigned_user`	`OprUserPropertyChange`
`category`	`OprStringPropertyChange`
`cause`	`OprStringPropertyChange`
`control_transfer_to`	`OprContrlTransferredToPropertyChange`
`correlation_rule`	`OprCorrelationRulePropChange`
`custom_attribute`	`OprCustomAttributePropertyChange`
`description`	`OprStringPropertyChange`
`duplicate_count`	`OprIntegerPropertyChange`
`eti_hint`	`OprStringPropertyChange`
`key`	`OprStringPropertyChange`
`key_pattern`	`OprStringPropertyChange`
`node_hint`	`OprStringPropertyChange`
`object`	`OprStringPropertyChange`
`om_service_id`	`OprStringPropertyChange`
`om_user`	`OprStringPropertyChange`
`original_data`	`OprStringPropertyChange`
`original_id`	`OprStringPropertyChange`
`priority`	`OprPriorityPropertyChange`

Name	ChangeType
`related_ci_hint`	`OprStringPropertyChange`
`severity`	`OprSeverityPropertyChange`
`solution`	`OprStringPropertyChange`
`source_ci_hint`	`OprStringPropertyChange`
`state`	`OprStatePropertyChange`
`sub_category`	`OprStringPropertyChange`
`time_created`	`OprTimePropertyChange`
`time_received`	`OprTimePropertyChange`
`title`	`OprStringPropertyChange`
`type`	`OprStringPropertyChange`

Event Changes List

History lines are provided for a specific event and no filtering support is provided in this web service endpoint.

The Event Changes List returns all history lines available. You can access the event changes list by calling the following URL:

`http://<bsmserver.example.com>/opr-console/rest/event_changes_list`

The change list can be filtered using the standard HTTP query parameters. This list is used to detect event changes as they occur.

In order to receive the history lines from a specific date, you must specify a watermark query. See "Filtering by Date and Time: watermark" on page 313 for more details about watermarks and specifying date and time.

File Locations

File locations for reference material for the Event Web Service interface are as follows:

- Event Web Service Java API Documentation:

  `<HPBSM root directory>/opr/api/doc/opr-external-api-javadoc.zip`

- Event Web Service Schema:

  `<HPBSM root directory>/opr/api/schema/OprDataModel.xsd`

Chapter 13

Integrating External Event Processes

This section describes the interface that enables integrations with external applications, where the aim is to integrate external processes into event processing. The interface enables notification of forwarded events and subsequent changes to be received programmatically.

The main use for the interface is to synchronize events and changes to events with an external manager such as an ITIL incident manager, for example, HP Service Manager or BMC Remedy Service Desk.

This section is structured as follows:

Forwarding Events and Synchronizing Event Changes

This section describes how events are forwarded to an external event processing application, such as an incident manager like HP Service Manager or BMC Remedy Service Desk, and how forwarded events and subsequent event changes are synchronized back from the external application.

Event Forwarding and Synchronization Process

Synchronizing events and event changes between applications depends on:

- Forwarding events and subsequent changes to an external event process

- Synchronizing event changes back from an external process

Forwarding Events and Event Changes to an External Event Process

The following graphic shows an overview of the event forwarding architecture, and the various routes that the forwarded event can take to reach the incident manager.
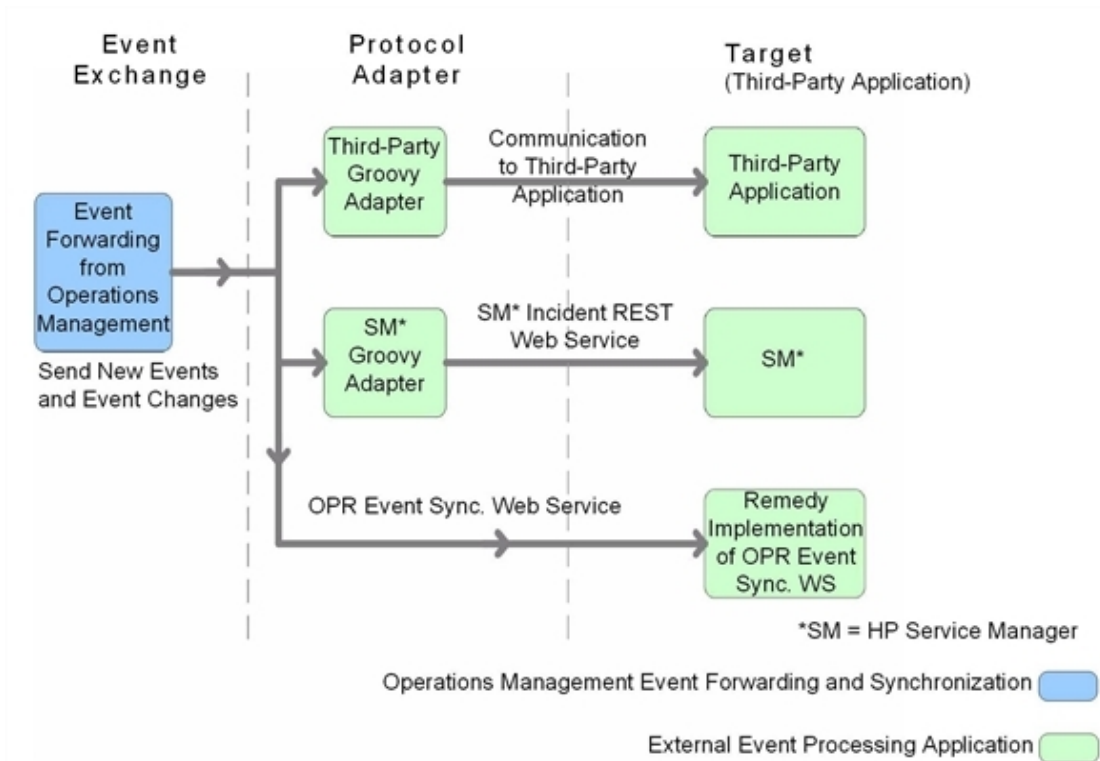
The target servers that are to receive the forwarded events and subsequent event changes must be configured using the Connected Servers manager. You can also configure which events to forward

based upon a filter, and which connected server to forward the events to. You can configure filters in the Event Forwarding manager.

For details about how to configure connected servers and forwarding rules, see the Operations Management online help.

Events matching the filter are pushed to the target connected server, together with any subsequent changes to the event. The following four forwarding modes are supported:

- Notify: events that match the filter are forwarded to the specified connected server.

- Notify and Update: same as Notify, but with subsequent changes to the events also forwarded to the target connected server.

- Synchronize: same as Notify and Update, but supports bi-directional synchronization, with the target connected server expected to synchronize back any changes it made to the event.

- Synchronize and Transfer Control: same as Synchronize, but control of the event is transferred to the connected server. Only Operations Management users with special permission are allowed to close the event after control is transferred, for example, an Administrator. It is expected the connected server will synchronize back the closed state when the external event (incident) is closed. In addition to being available as an option for a forwarding rule, an operator may manually transfer control via the context menu of the Event Browser.

Delivery of forwarded events and subsequent event changes is guaranteed. If the target connected server is down when the event is forwarded or changes occur, the request is queued and delivered when the target connected server becomes available again.

There are two ways to integrate an external event process:

- Using a Groovy script adapter

- Using the Event Synchronization Web Service

Groovy Script Adapter

You can write and modify Groovy scripts so that you can customize out-of-the-box adapters, and create new adapters. The following Groovy script adapters are provided out-of-the-box:

- Service Manager Adapter

- Sample Logfile Adapter (see "Sample Groovy Script: Logfile Adapter" on page 335)

If a Groovy script is configured for the connected server, then a call is made to the Groovy script to forward matching events, and subsequent changes to those events, to the target connected server. The script is so designed that it uses the API that is best suited to deliver the event and event changes to the target connected server. For example, Groovy scripts used for the Service Manager Adapter use the Apache Wink REST client API to execute REST web service calls to HP Service Manager.

For more details about how to use Groovy scripts for integrations, see "Integrating External Event Processes Using Groovy Scripts" on page 335.

Event Synchronization Web Service

Instead of implementing a Groovy script, an integrator may implement an Event Synchronization Web Service endpoint to receive event forwarding requests and subsequent updates directly from Operations Management. If the connected server is configured within Operations Management to call an event REST web service rather than a Groovy script, then it is also expected that an OPR Event-compliant REST web service is implemented by the target server and available at the connected server endpoint.

To forward events and their subsequent changes, the following standard REST web service HTTP method calls are made by Operations Management:

- POST: A POST call forwards an event with the payload of an OPR event (the payload is the body of the request). The base URL configured for the target connected server, appended with the `/event` parameter, is used to address the endpoint. The new event created is expected in the response payload.

- POST: A POST call forwards a list of events with the payload of an OPR event list. The base URL configured for the target connected server, appended with the `/event` parameter is used to address the endpoint. This endpoint is optional. If provided, the user may select **Supports Bulk Transfer** in the Connected Servers configuration. Each event in the list has a unique `sequence_number`. The expected response payload in an OPR event list containing the events that are created. The corresponding events must have the `sequence_number` set to identify which events were created. Events that are not identified in the response are retried.

- POST: A POST call forwards event changes with the payload of an OPR event change. The base URL configured for the target connected server, appended with the `/event_change/<external_event_ID>` parameter, is used to address the endpoint. The expected response payload is an OPR event change object.

- POST: A POST call receives bulk event changes with the payload of OPR event change list. This endpoint is optional. It is only required if the option **Supports Bulk Transfer** is selected in the Connected Servers configuration. The base URL configured for the target connected server, appended with the `/event_change` parameter, is used to address the endpoint. The expected response payload is an OPR event change list. Each OprEventChange item in the list has an event reference with the event ref with the `target_id` set to the OPR event ID and the `global_target_id` set to the external event ID. Each OprEventChange item also has a sequence number set to it's sequence in the list. The response must set the sequence number to denote which event changes have been successfully applied. Event changes missing from the list (identified by the corresponding sequence number) are retried.

- GET: A GET call is used to get the current state of the external event. The base URL configured for the target connected server, appended with the parameters `/event/<external_event_ID>`, is used to address the endpoint. The expected response payload is an OPR event object.

- HEAD: A HEAD call is used to ping the service. This is used by the Connected Servers manager to check the web service credentials specified by the end user. The base URL configured for the target connected server is used to address the endpoint.

For details about how to use the Event Synchronization Web Service, see "Event Synchronization Web Service Interface" on page 352.

Receiving Event Changes Back from an External Event Process

The following graphic shows an overview of how the target application synchronizes changes back to the OPR Event Synchronization Web Service.
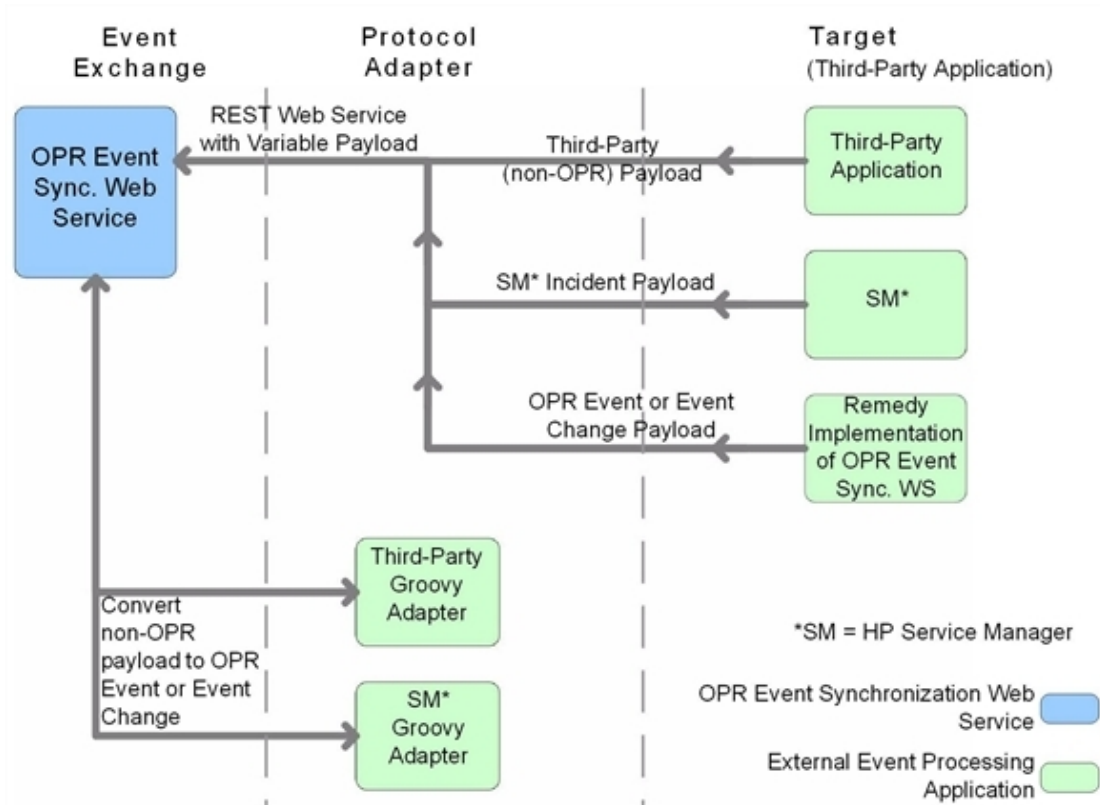
When configuring forwarding types for connected servers, you need to consider whether the target server should synchronize back any changes to Operations Management. If so, you need to specify either the `Synchronize` forwarding type, or the `Synchronize and Transfer Control` forwarding type when configuring a forwarding rule.

You can configure a target connected server to support transfer of control. If Supports Synchronize and Transfer of Control is selected during the configuration of the target server, the server is available in the Event Browser context menu, enabling an operator to transfer control of the event to this target connected server. If transfer of control is not selected, then the target server does not appear in the context menu.

If it is expected that the target connected server should synchronize back any changes, the Event Synchronization Web Service is available to receive these changes. You can customize the payload of the web service using a Groovy script adapter, in the same way as you can when configuring event forwarding.

HP OMi additionally provides the following WS PUT method endpoints for use by external applications to synchronize back external event changes.

PUT: A PUT call is used to bulk update external events. The URL used by Operations Management to address the endpoint for bulk updating external events is:
`http://gatewayhost/opr-gateway/rest/synchronization/event/`

With a Groovy Script configured for the Connected Server, the payload is defined by the Groovy script `receiveChanges()` method.

Without a Groovy Script configured for the Connected Server, the payload must be an `OprEventList` object.

For details about how to use the Event Synchronization Web Service, see "Event Synchronization Web Service Interface" on page 352.

Performing a URL Launch of the Event Browser from an External Application

Any operator who needs to drill down into the Operations Management user interface from an external application using a URL launch of the Event Browser must:

- Be a valid Operations Management user.

- Have the same user name set up in Operations Management as the user name of the operator logged on to the calling application and performing the call.

  If Single Sign-On (SSO) authentication is configured, set up the operator in Operations Management with the *same* user name that is used by operator to log onto the calling application and to perform the URL call. (The password of the Operations Management operator can be empty or any string.) After successfully logging into the calling application, the operator can launch the Operations Management Event Browser without further authentication.

  If the calling application is not configured to use SSO authentication, set up the operator in Operations Management with the *same* user name that is used by the operator of the calling application and specify a valid password. The operator is required to enter user name and password when launching the Operations Management Event Browser.

- Have the permission `Events assigned to user` including the required actions granted. You can optionally grant the permission to view events not assigned to the user.

Without this valid user name, or if a user does not have the required viewing rights, an attempt to perform a URL launch of the Operations Management Event Browser from the calling application results in an empty browser window.

For details about how you would set up a URL launch of the Operations Management Event Browser, see "URL Launch of the Event Browser" on page 269.

Integrating External Event Processes Using Groovy Scripts

Groovy scripting is supported. You can use a Groovy script adapter to forward events and event changes to target servers. This section provides information about the main integration points you need to consider when using and developing Groovy scripts for integrations with external event processing applications.

For more information about Groovy and documentation describing the Groovy language, visit:

http://groovy.codehaus.org

You can also find answers to frequently asked questions in the section "Groovy Scripts and Programming" on page 375.

Sample Groovy Script: Logfile Adapter

A sample Groovy script is provided out-of-the-box for use as a template. It is called `LogfileAdapter.groovy` and can be found in the following directory:

`<HPBSM root directory>``/conf/opr/integration/sample`

You can configure a target connected server in the Connected Servers manager using the Logfile Adapter as the Groovy script adapter. You can use any hostname and port number for the configuration when using the LogfileAdapter, but note that you cannot use the DNS name of the Operations Manager i server. You must use another name, for example, `localhost`. Events and event changes forwarded to this adapter are logged to this log file:

`<HPBSM root directory>``/log/opr/integration/LogfileAdapter.log`

Developers and integrators can use this adapter for testing, and as a template to create other adapters.

A template groovy script for use as a starting point when developing new groovy scripts for external event integration on OMi is available from the following location:

Note: *`<HPBSM root directory>`*`/opr/examples/external-event-adapter` contains

There are clearly marked sections with "TODO" and steps to take when developing a new integration.

Configure a Connected Server using the Logfile Adapter

Synchronizing events and event changes between an Operations Management instance and a third-party event process depends upon Operations Management forwarding events to the external event processing application, with these events and event changes being sent back from the external application. The first step to achieve this is to configure a target connected server in the Connected Servers manager.

For full details about how to configure a connected server, see the Operations Management online help.

To configure a target connected server using the Logfile Adapter as Groovy script adapter, perform the following steps:

1. Navigate to the Connected Servers manager in the Operations Management user interface:

   Admin > Operations Management > Setup > Connected Servers

2. Click the New ✳ button to open the Create New Server Connection dialog box.

3. In the **Display Name** field, enter a name for the target connected server. The Name field is filled automatically. For example, if you enter `Logger Example` as the Display Name for the target HP Service Manager server, `Logger_Example` is automatically inserted in the Name field.

   > **Note:** Make a note of the name of the new target server (in this example, `Logger_Example`). You need to provide it later on as the `username` when configuring the HP Service Manager server to communicate with the server hosting Operations Management.

   Optional: Enter a description for the new target server.

   Make sure that you check the Active checkbox.

   Click **Next**.

4. Select **External Event Processing** to choose the server type suitable for an external event processing application.

   Click **Next**.

5. Enter the Fully Qualified DNS Name of the Logfile target server, for example, `localhost`

   Click **Next**.

6. Next, you need to establish the type of integration. In the Integration Type dialog box, you can choose between using a Groovy script adapter, or the Event Synchronization Web Service. In this example, we want to choose the a Groovy script adapter.

   a. Select **Call Groovy Script Adapter**.

   b. In the External Event Processing Type field, select **sample**.

   c. In the Groovy Script File Name field, select **LogfileAdapter.groovy**.

      (Leave the Groovy Classpath field blank, as in this case, no external resources are required.)

   d. Click **Next**.

7. The Outgoing Connection dialog box is for providing the credentials (user name, password, port number) to enable connection to the target server, and to forward events to that server. In the case of the Logfile Adapter, you do not need to provide the user name, password, or port number, and you do not have to choose the HTTP setting. You can leave these fields blank for this exercise.

   In the Outgoing Connection dialog box, do the following:

   a. Make sure that the Enable Synchronize and Transfer Control checkbox is checked. When the Synchronize and Transfer Control flag is set, an Operations Management operator is

then able to transfer ownership of the event to the target connected server. If the flag is not set, then the option Synchronize and Transfer Control does not appear in the list of forwarding types when configuring forwarding rules.

   b. Click **Next**.

8. The Event Drilldown dialog opens. If, in addition to purely forwarding events to the target connected server, you also want to be able to drill-down into the external event processing application, you need to specify the fully qualified DNS name, and port of the Operations Management system where you want to perform event drill-down. For this example, enter the following:

- Fully qualified DNS name: **test.host.com**

- Port: **80**

- Click **Next**.

9. The next thing to do is to enable event changes to be received back from the connected server. For this you need to provide credentials for the connected server to access the server hosting Operations Management.

   a. In the Incoming Connection dialog box, enter a password that the external application requires to connect to the server hosting Operations Management, `HPpasswd1_` in this example.

   b. Click **Finish**.

The target Logger Example server appears in the list of Connected Servers.

Configure an Event Forwarding Rule

The next step is to configure an event forwarding rule that determines which events are forwarded to the `Logger Example` server.

Refer to the Operations Management online help for full details about configuring filters.

To configure a forwarding rule, carry out the following steps:

1. Navigate to the Event Forwarding manager in the Operations Management user interface:

   Admin > Operations Management > Event Automation > Event Forwarding

2. Click the New ✳ button to open the Create New Forwarding Rule dialog box.

3. In the Display Name field, enter a name for the forwarding rule, in this example `Forward Major (Sync and Transfer Control)`.

   Optional. Enter a description for the forwarding rule you are creating.

   Make sure the Active checkbox is checked. A rule must be active in order for its status to be available in the target connected server.

4. Click the browse button next to the Event Filter field. The Select an Event Filter dialog opens.

5. In the Select an Event Filter dialog, click the New ✳ button to open the Filter Configuration dialog.

6. In the Filter Display Name field, enter a name for the new filter, in this example, `FilterMajor`.

   Deselect the checkboxes for all severity levels except for the severity Major.

   Click **OK**.

7. You should see your new filter in the Select an Event Filter dialog (select it, if it is not already highlighted).

   Click **OK**.

8. Under Target Servers, select the connected target server you configured in the section "Sample Groovy Script: Logfile Adapter" on page 335. In this example, this is `Logger Example`.

   Click the Add ( ) button next to the target servers selection field. You can now see the connected server's details.

   Click **OK**.

The new forwarding rule is now available.

Groovy Script Interface

If you intend to use Groovy scripts to integrate with external event processes, such as an integration with an incident manager, then you must implement a Groovy script that implements the methods defined by the following interface:

`com.hp.opr.api.ws.adapter.ExternalProcessAdapter`

This is provided in the JAR file `opr-external-api.jar`.

The Groovy script interface with full documentation of all arguments and types can be found in the Javadoc documentation delivered with the product. For the location of the Javadoc API Documentation, see "Javadoc API Documentation" on the next page.

Important File Locations

The locations for the API library, Javadoc API Documentation, and the OPR Event Schema are given in this section.

API Library

The API library contains Java Architecture for XML Binding (JAXB) annotated classes for the OPR Event and Event Change objects. If you are programming in Groovy or Java, you can use these classes directly. Otherwise, you may need to use the OPR Event Schema (see "OPR Event Schema" on the next page).

You can find the API library in the following installation location:

<HPBSM root directory>/lib/opr-external-api.jar

Javadoc API Documentation

The external API interface is documented in detail in the Javadoc API Documentation. You can find it in the following installation location:

<*HPBSM root directory*>`/opr/api/doc/opr-external-api-javadoc.zip`

OPR Event Schema

If you are using a programming language other that Java, you may need to use the OPR Event Schema to generate classes to marshall and unmarshall Event and Event Change objects.

You can find the OPR Event Schema in the following installation location:

<*HPBSM root directory*>`/opr/api/schema/OprDataModel.xsd`

Event Integration Groovy Scripts

Event integration Groovy scripts are stored in the database and accessed from the following location:

Admin > Operations Management > Setup > Connected Servers

In the Connected Servers pane, click the ▤ button to open the Event Forwarding Scripts Configuration dialog box from which you can select an existing script or create a new one.

The out-of-the-box scripts are:

Logfile Adapter, type: `sample:LogfileAdapter`

Service Manager Adapter, type: `sm:ServiceManagerAdapter`

Upgrade Adapter, type: `upgrade:UpgradeAdapter`

Groovy Script Methods

If you are using a Groovy script to integrate external event processes, that Groovy script must implement the methods defined by the `ExternalProcessAdapter` interface.

The methods that must be implemented by the Groovy script are listed in this section.

init

The `init(def args)` method is called when the Groovy script is first loaded. The loaded script is cached and called many times. It is only reloaded at startup and whenever the script or the configuration is changed in the Connected Servers manager. A separate instance is created for each connected server that uses this script.

The `init(def args)` method has one argument, and the properties listed in "Properties for init() Method" on the next page.

Properties for init() Method

Property	Description
Read Properties	
`installDir`	BSMOperations Management root directory.
`logger`	Type `org.apache.commons.logging.Log`. Used to access the logs.
`connectedServerId`	ID of the target connected server.
`connectedServerName`	Name of the target connected server.
`connectedServerDisplayName`	Display Name of the target connected server.
`node`	DNS name of the target connected server.
`port`	Type Integer. Port number of the web service, if any, for the target connected server.
`nodeSsl`	Type Boolean. Indicates if the target connected server web service requires SSL.
`drilldownNode`	DNS name of the target connected server where drill-down may be done.
`drilldownPort`	Type Integer. Drill-down port, if any, for the drilldown service on the target connected server drill-down node.
`drilldownNodeSsl`	Type Boolean. Indicates if the drill-down service requires SSL.
`maxTimeout`	Number of milliseconds to use for timeout on connections. You configure this in the Connected Servers manager.
Write Properties	
None	There are no write properties for this method.

destroy

The `destroy()` method is called when the Groovy script is no longer needed. It has no arguments.

ping

The `ping(def args)` method is called by the Connected Servers manager to determine whether the configuration parameters are correct. If the connected server is reachable using the specified read properties, the `ping` method returns true, otherwise it returns false.

lists the properties for the ping()method:

Properties for ping() Method

Property	Description
Read Properties	

Property	Description
`installDir`	BSMOperations Management root directory.
`logger`	Type `org.apache.commons.logging.Log`. Used to access the logs.
`connectedServerName`	Name of the target connected server.
`connectedServerDisplayName`	Display Name of the target connected server.
`node`	DNS name of the target connected server.
`port`	Type Integer. Port number of the web service, if any, for the target connected server.
`nodeSsl`	Type Boolean. Indicates if the target connected server web service requires SSL.
`credentials`	Type `java.net.PasswordAuthentication` Credentials, if any, for the target connected server.
`locale`	Type: Locale The desired locale of the properties that are set.
Write Properties	
`outputDetail`	Detailed results of the ping operation. This is displayed in the Connected Servers configuration dialog box, and should be in the desired locale.

Methods for Forwarding Events to a Connected Server

The methods for forwarding events to an external event processing application, configured as a target connected server, are listed in this section.

forwardEvent

The `forwardEvent(def args)` method is called to forward the event to the target connected server. This method has one argument, with the properties listed in "Properties for forwardEvent() Method" below:

Properties for forwardEvent() Method

Property	Description
Read Properties	
`credentials`	Type: `java.net.PasswordAuthentication` Credentials, if any, for the target connected server.

Property	Description
`event`	Type: `com.hp.opr.api.ws.model.event.OprEvent` This is the event to be forwarded.
`info`	Type: `com.hp.opr.api.ws.model.event.c-i.OprForwardingInfo` Holds information regarding the type of forwarding, that is: `Notify`, `Notify and Update`, `Synchronize`, or `Synchronize and Transfer Control`.
`cau-seExternalRefId`	If this is a symptom, the cause is set, and the cause has been forwarded to the target server, this property holds the external reference ID of the cause, otherwise null is returned.
Read Methods	
`credentials`	Type: `java.net.PasswordAuthentication` Credentials, if any, for the target connected server.
`getEvent(id, includeRefs)`	Allows getting a specified event. If the event exists and the application has access to it, it is returned, otherwise null is returned. If the caller wishes the references to be resolved, set `includeRefs` to true, otherwise false. See Event WS for more details on this flag.
`getCi(id)`	Obtains details of a configuration item, including all properties. Arguments are of type String. The method returns the specified configuration item, including all properties. Return value is of type `OprConfigurationItem`.
Write Properties	
`drilldownUrlPath`	*Optional*. If set, enables drill-down into the external application in the Event Browser. This is the base path of the URL, and does not include node or port
`externalRefId`	Must be set if forward is successful. Can be any string value.

forwardEvents

The `forwardEvents(def args)` method is called to forward a list of events to the target connected server. This method is passed a single argument, that contains a list of events to forward, as well as some utility methods. There is also a method to set the forward result of each event in the list. If the forward result is not set for one of the events it will remain in the forwarding queue and will be retried on the next call to the groovy script. The result for each individual event may be set to success or failed. This method is optional.

The method `forwardEvent()` has utility methods to enable the groovy script to easily obtain the `OprForwardingInfo`. `BulkForwardEventArgs` does not directly provde this utility method.

To obtain the `OprForwardingInfo` for each event, the you can call the method `getForwardingInfo(final String serverId)` on each event. The `serverId` is the ID of this connected server and was passed to the groovy script in the `InitArgs`.

This method has one argument (`BulkForwardEventArgs`), with the properties listed in "Properties for forwardEvents() Method" below:

Properties for forwardEvents() Method

Property	Description
Read Properties	
`credentials`	Type: `java.net.PasswordAuthentication` Credentials, if any, for the target connected server.
`events`	Type: `com.hp.opr.api.ws.model.event.OprEventList` This is the list of events to be forwarded.
Read Methods	
`getEvent(id, includeRefs)`	Enables getting a specified event. If the event exists and the application has access to it, it is returned, otherwise null is returned. If the caller wishes the references to be resolved, set `includeRefs` to true, otherwise false. See Event WS for more details on this flag.
`getCi(id)`	Obtains details of a configuration item, including all properties. Arguments are of type String. The method returns the specified configuration item, including all properties. Return value is of type `OprConfigurationItem`.
`getCauseExternalRefId (symptomId)`	Obtains an external reference ID. Arguments are of type String.The method returns the `cause external reference ID`, if any exists for the specified symptom.
Write Methods	
`setForwardSuccess (eventId, externalRefId, crossLaunchUrlPath)`	This method enables setting the success for each individual event, along with the external reference ID and any cross launch URL for that external event.
`setForwardFailed (reason)`	If the bulk forward fails, this method can be called, passing a `Throwable` which describs the reason for the failure to deliver. The reason may be null. The entire list is marked as failed and not retried.
`setForwardFailed (eventId, reason)`	Sets an individual event in the list to failed. This event is not retried. An optional reason may be specified or set to null. If neither `setSuccess()` or `setFailed()` is called for a particular event, the event is retried later.

forwardChange

The `forwardChange(def args)` method is called to forward the event changes to the target connected server. This method has one argument, with the properties listed in "Properties for forwardChange() Method" below:

Properties for forwardChange() Method

Property	Description
Read Properties	
changes	Type: com.hp.opr.api.ws.model.event.OprEventChange The event changes to be forwarded.
credentials	Type: java.net.PasswordAuthentication Credentials, if any, for the target connected server.
externalRefId	Type: String The ID of the external event that should be updated.
info	Type: com.hp.opr.api.ws.model.event.c-i.OprForwardingInfo Holds information regarding the type of forwarding, that is: `Notify`, `Notify and Update`, `Synchronize`, or `Synchronize and Transfer Control`.
event	Returns a copy of the event in its current state. The attributes of the event reflect the current state of the event at the time this call was made, and not the state when the change occurred.
cau-seExternalRefId	If this is a symptom, the cause is set, and the cause has been forwarded to the target server, this property holds the external reference ID of the cause, otherwise null is returned.
Read Methods	
getEvent(id, includeRefs)	Allows getting a specified event. If the event exists and the application has access to it, it is returned, otherwise null is returned. If the caller wishes the references to be resolved, set `includeRefs` to true, otherwise false. See Event WS for more details on this flag.
getCi(id)	Obtains details of a configuration item, including all properties. Arguments are of type String. The method returns the specified configuration item, including all properties. Return value is of type `OprConfigurationItem`.
Write Properties	
None	There are no write properties for this method.

forwardChanges

The `forwardChanges(def args)` method is called to forward the event changes to the target connected server. This method is passed a single argument, that contains a list of event changes to forward, as well as some utility methods. There is also a set of methods to set the forward result of each event change item in the list. If the forward result is not set for one of the event changes it will remain in the forwarding queue and will be retried on the next call. The result for each individual event change may be set to success or failed. This method is optional.

This method has one argument (`BulkForwardChangeArgs`), with the properties listed in "Properties for forwardChanges() Method" below:

Properties for forwardChanges() Method

Property	Description
Read Properties	
`changes`	Type: `com.hp.opr.api.ws.model.e-vent.OprEventChangeList` The event changes to be forwarded.
`credentials`	Type: `java.net.PasswordAuthentication` Credentials, if any, for the target connected server.
Read Methods	
`getCi(id)`	Obtains details of a configuration item, including all properties. Arguments are of type String. The method returns the specified configuration item, including all properties. Return value is of type `OprForwardingInfo`.
`getEvent(id, includeRefs)`	Allows getting a specified event. If the event exists and the application has access to it, it is returned, otherwise null is returned. If the caller wishes the references to be resolved, set `includeRefs` to true, otherwise false. See Event WS for more details on this flag.
`get-CauseExternalRefId (symptomId)`	Obtains an external reference ID. Arguments are of type String.The method returns the `cause external reference ID`, if any exists for the specified symptom.
`OprForwardingInfo getForwardingInfo (change)`	Obtains details of the forwarding information, including all properties. Arguments are of type String. This method gets the `OprForwardingInfo` for the specified event change. The external reference ID can be obtained from this forwarding info.
Write Methods	

Property	Description
`setForwardSuccess()`	Sets success for all changes in the list.
`setForwardSuccess (changesId)`	Sets success for the specified change in the list.
`setForwardFailed (reason)`	Sets all changes in the list to failed. A reason may be specified in the form of an Exception, or null may be specified. If set to failed, the changes are not retried.
`setForwardFailed (changeId, reason)`	Set the specified change to failed. This change is not retried. If neither `setSuccess()` or `setFailed()` is called for a particular change, the change is retried later.

Methods for Receiving Synchronization Data from a Connected Server

The methods for receiving synchronization data from an external event processing application, configured as a target connected server, are listed in this section.

> **Note:** These methods are only needed if the connected server supports the synchronizing back of event changes from the connected server.

receiveChange

The `receiveChange(def args)` method is called when the connected server sends an event change to the Event Synchronization Web Service. This method has one argument with the properties listed in "Properties for receiveChange() Method" below:

Properties for receiveChange() Method

Property	Description
Read Properties	
`event`	Type: OprEvent A read only copy of the event in its current state, before any changes are applied. This is for read-only. To update the event set one of the writable properties of the arguments.
`exter-nalEventChange`	Type: String The payload of the web service call (PUT or POST) received from the connected server when a change is synchronized back from the connected server.

Property	Description
`info`	Type: `com.hp.opr.api.ws.model.event.c-i.OprForwardingInfo` Holds information regarding the type of forwarding, that is: `Notify`, `Notify and Update`, `Synchronize`, or `Synchronize and Transfer Control`.
`locale`	Type: Locale The desired locale of the web service caller.
`externalRefId`	Type: String The ID of the external event that should be updated.
`causeExternalRefId`	If this is a symptom, the cause is set, and the cause has been forwarded to the target server, this property holds the external reference ID of the cause, otherwise null is returned.
Read Methods	
`getEvent(id, includeRefs)`	Allows getting a specified event. If the event exists and the application has access to it, it is returned, otherwise null is returned. If the caller wishes the references to be resolved, set `includeRefs` to true, otherwise false. See Event WS for more details on this flag.
`getCi(id)`	Obtains details of a configuration item, including all properties. Arguments are of type String. The method returns the specified configuration item, including all properties. Return value is of type `OprConfigurationItem`.
Write Properties	
`eventId`	ID of the event.
`title`	Title of the event.
`description`	Description of the event.
`solution`	Solution of the event.
`severity`	Severity of the event.
`priority`	Priority of the event.
`state`	State of the event. If set to `closed` or `resolved`, the state of the event is set to `closed`. Setting the event to any other value sets the state of the event to `open`.
`cause`	ID of the cause event.

Property	Description
`assignedUser`	Type: `com.hp.opr.api.ws.model.event.OprUser` Name of the user who is responsible for solving the event's underlying problem.
`assignedGroup`	Type: `com.hp.opr.api.ws.model.event.OprGroup` Name of the group to which the event's assigned user belongs.

Additional Methods

Additional methods are provided for transferring the control of an event, annotations, custom attributes, configuration items, and for accessing the HTTP requests and responses.

Control Transfer

The methods available for transferring control of an event are listed here.

- `requestControl`

  The `requestControl` method is a request to take control of the event. Once a request has been made, it is not possible for the event to be owned by another server. The request is queued, and when completed, the caller receives a change notification for the `control_transferred_to` event property.

- `returnControl`

  The `returnControl` method returns the control of the event back from the external event process to Operations Management. The caller must have control of the event in order to return control. The request is queued, and when completed, the caller receives a change notification for the `control_transferred_to` event property.

Annotations

The methods available for annotations are listed here. Annotations may be added or updated, but not deleted.

- `addAnnotation(def author, def text)`

  The `addAnnotation(def author, def text)` method adds an annotation. Arguments are of type String.

- `updateAnnotations(def id, def author, def text)`

  The `updateAnnotation(def id, def author, def text)` method updates an annotation. Arguments are of type String.

Custom Attributes

The methods available for custom attributes are listed here.

- `addCustomAttribute(def name, def value)`

  The `addCustomAttribute(def name, def value)` method adds a custom attribute. Arguments are of type String.

- `updateCustomAttribute(def name, def value)`

  The `updateCustomAttribute(def name, def value)` method updates a custom attribute. Arguments are of type String.

HTTP Requests and Responses

Three methods are provided for accessing the HTTP request or response.

- `getHttpRequestHeader(def name)`

  The `getHttpRequestHeader(def name)` method returns a list of header values. Arguments are of type String.

- `setHttpResponseStatus(def httpResponseCode, def httpResponseMessage)`

  The `setHttpResponseStatus(def httpResponseCode, def httpResponseMessage)` method can be called to control the response status and payload. Default is 202 and no payload. The code is of type Integer, message is of type String.

- `setHttpResponseHeader(def name, def value)`

  The `setHttpResponseHeader(def name, def value)` method enables setting of the specified HTTP response header. Arguments are of type String.

receiveChanges

The `receiveChanges(def args)` method is called when the connected server sends multiple event changes to the Event Synchronization Web Service. This method is passed a single argument, that contains the payload sent by the external processing server to the Event Synchronization WS endpoint /opr-gateway/rest/9.10/event_change, as well as some utility methods. It is expected the payload contains a list of changes. It is up to the groovy script method to interpret the list of changes and apply them to an event. This is done through a utility method to create a ReceiveChangeArgs object for each event that is to be changed and setting the changes in that object.

This method has one argument (`BulkReceiveChangeArgs`) with the properties listed in "Properties for receiveChanges() Method" below:

Properties for receiveChanges() Method

Property	Description
Read Properties	
`externalEventChanges`	Type: String The payload of the web service call (PUT or POST) received from the connected server when one or more changes are synchronized back from the connected server.

Property	Description
locale	Type: Locale The desired locale of the web service caller.
Read Methods	
getEvent(id, includeRefs)	Allows getting a specified event. If the event exists and the application has access to it, it is returned, otherwise null is returned. If the caller wishes the references to be resolved, set includeRefs to true, otherwise false. See Event WS for more details on this flag.
getCi(id)	Obtains details of a configuration item, including all properties. Arguments are of type String. The method returns the specified configuration item, including all properties. Return value is of type OprConfigurationItem.
getgetHttpRequestHeader (name)	Obtains the specified header. Header may be specified more than once. All instances specified are returned.
getReceiveChangeArgs (id)	Gets the ReceiveChangeArgs object for the specified event. It is automatically added to the list of events to be changed. The methods provided by ReceiveChangeArgs can be called to apply the appropriate changes to the specified event.

Methods for Retrieving Data from a Connected Server

The methods for retrieving data from a connected server are listed in this section.

getExternalEvent

The getExternalEvent() method is called when the operator requests the current representation of the external event. When the operator opens the Event Details in the Event Browser and selects the External Info tab, or uses the context menu in the Event Browser, the getExternalEvent() method is called to get the latest copy of the external event, and display selected fields in the Event Browser. This method has one argument with the properties listed in "Properties for getExternalEvent() Method" below:

Properties for getExternalEvent() Method

Property	Description
Read Properties	
externalRefId	The external reference ID, identifying the external event to get.
credentials	Type: java.net.PasswordAuthentication Credentials, if any, for the target connected server.

Property	Description
locale	Type: Locale
	The desired locale of the properties that are set.
Write Properties (All are of type String)	
title	*Optional.* Title of the external event.
description	*Optional.* Description of the external event.
assignedUser	*Optional.* User assigned to the external event.
assignedGroup	*Optional.* User group assigned to the external event.
severity	*Optional.* Severity of the event.
priority	*Optional.* Priority of the event.
state	*Optional.* State of the event.

Method for Supplying Data to a Connected Server

There is also a method for supplying data to a target connected server.

toExternalEvent

The toExternalEvent() method converts an OPR Event into an external event object. This method is called when a connected server queries the Event Synchronization Web Service for the latest copy of the OPR Event. This enables an integrator to convert the event to an external event representation, that is, to create the response payload of the GET from (.../event/<event id>).

This method has one input parameter: com.hp.opr.api.ws.model.event.OprEvent

The return value of this method is a string. This is the payload that is returned to the application that called the web service.

Management of Groovy Script Methods

The Groovy script methods run under a separate Java Security Manager. The following permissions are denied by the Java Security Manager:

- access to the System.exit() call

- java.lang.RuntimePermission only for target exitVM:

  http://java.sun.com/javase/6/docs/api/java/lang/RuntimePermission.html

Event Synchronization Web Service Interface

The Event Synchronization Web Service interface can be used to integrate with external (third-party) event processes, such as an incident manager like HP Service Manager. The interface is used to forward events and event changes from the Operations Management OPR client to the third-party application, and to synchronize back from the third-party client events and event changes that the external application makes to the events.

An external event process may integrate directly with Operations Management event processing, through the implementation of an OPR-compliant Event Synchronization Web Service, as opposed to implementing a Groovy script. The external application would need to implement a web service endpoint to receive OPR events and event changes, and, if synchronization is supported by the external application, a REST web service client to submit events and synchronize back event changes to the Event Synchronization Web Service.

Forward Events and Event Changes from OPR Client

For the OPR client to be able to forward events and event changes to a third-party event processing application, that third-party application must implement an OPR-compliant Event Synchronization Web Service.
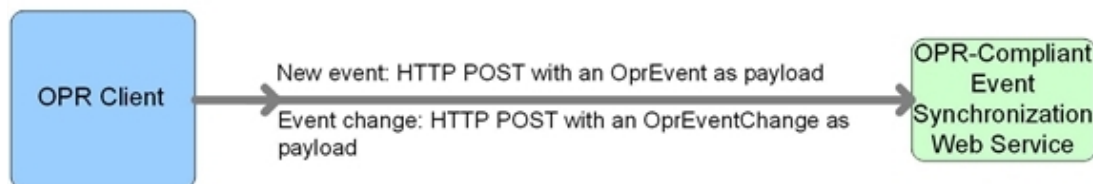
Forward Event

When forwarding events from the OPR client to a third-party application, the following data is relevant:

- HTTP Method: POST

- Base URL: `http://<base configured URL of the connected server>/event`

- Expected Payload: The expected response payload is an `OprEvent` object.

Forward Event Change

When forwarding event changes from the OPR client to a third-party application, the following data is relevant:

- HTTP Method: POST

- Base URL: `http://<base configured URL of the connected server>/event_change/<external_event_ID>`

- Expected Payload: The expected response payload is an `OprEventChange` object.

Forward Events

When forwarding events in bulk from the OPR client to a third-party application, the following data is relevant:

- HTTP Method: POST

- Base URL: `http://<base configured URL of the connected server>/event`

- Expected Payload: The payload is an `OprEventList` object.

- Payload returned: Upon success the target service must return a list of the events sent to it with the ID set to the external Event ID. The `sequence_number` must match the `sequence_number` of the event sent in the POST. For those events which do not have a matching event returned, they are retried in a subsequent request. If an event is to be rejected, an HTTP error must be returned for the entire payload. If an HTTP error 500-599 is returned, each event is individually retried as a non-bulk request. For an HTTP error 400-499, all events in the request are marked as failed and no retry is done.

Forward Event Changes

When forwarding event changes in bulk from the OPR client to a third-party application, the following data is relevant:

- HTTP Method: POST

- Base URL: `http://<base configured URL of the connected server>/event_change/`

- Expected Payload: The expected response payload is an `OprEventChangeList` object. Each `OprEventChange` item in the list must have the `event_ref` property set. The `global_id` is expected to be set to the ID on the target system, for example, when Operations Managementcalls this web service on the target system, the `target_id` is set to the Operations Managementevent ID and the `target_global_id` is set to the target object ID.

- Payload returned: Upon success the target service must return a list of the event changes sent to it with the same `sequence_number`. The `sequence_number` must match the `sequence_number` of the event change sent in the POST. For those changes which do not have a matching change returned, they are retried in a subsequent request. If an event change is to be rejected, an HTTP error must be returned for the entire payload. If an HTTP error 500-599 is returned, each event change is individually retried as a non-bulk request. For an HTTP error 400-499, all event changes in the request are marked as failed and no retry is done.

Web Service GET Request

When there is a need to get the external event from the third-party application (for example, when the External Info tab is selected from the Event Browser), the following data is relevant:

- HTTP Method: GET

- Base URL: `http://<base configured URL of the connected server>/event/<external_event_ID>`

- Expected Payload: The payload is the external event in the form of an `OprEvent`.

Web Service Ping Request

You can send a ping request *to* a third-party application. When sending a ping request to a third-

party application, the following data is relevant:

- HTTP Method: HEAD to base URL

- Base URL:

  - Base URL for forwarding events: `http://<base configured URL of the connected server>`

- Expected Payload: None

Synchronize Event Changes Back from External Client

If synchronization is supported by the external application, a REST web service client is needed to synchronize back event changes to the Event Synchronization Web Service. The payload in this back synchronization is expected to comply with native OPR objects. With a Groovy script configured for connected server, the Groovy script interprets the payload that is defined by the client.

If a client wants to submit a new event or a change to an event, but not invoke the Groovy script, the sub-paths `event_list` or `event_change_list` must be used. These paths will require native OPR objects (for example, `OprEvent` or `OprEventChangeList`). The sub-paths `event_list` and `event_change_list` also enable clients to submit one or more events or event changes.
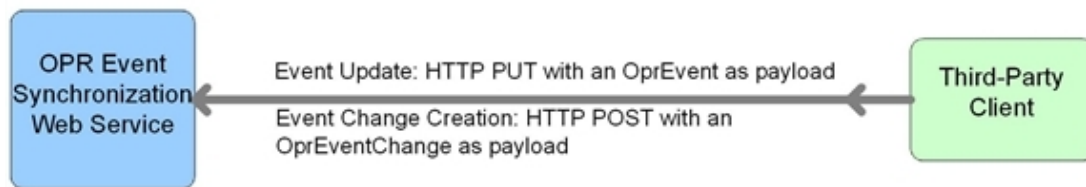
Event Updates

Event Get

In the case of event get, the following data is relevant:

- HTTP Method: GET

- Base URL: `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event/<event_id>`

- Payload returned:

  - With a Groovy script configured for the connected server, the Groovy script `toExternalEvent()` method defines the payload.

  - Without a Groovy script configured for the connected server, the payload is an `OprEvent` object.

Event Update

In the case of event updates, the following data is relevant:

- HTTP Method: PUT

- Base URL: `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event/<event_id>`

- Ping request (from third-party client): HTTP HEAD to base URL

- Payload received:

  - With a Groovy script configured for the connected server, the Groovy script defines the payload.

  - Without a Groovy script configured for the connected server, the payload must be an `OprEvent` object.

  See also "Event Update: Logfile Adapter Examples" on page 358.

Event Change Creation

In the case of event changes, the following data is relevant:

- HTTP Method: POST

- Base URL: `http://<bsmserver.example.com>/rest/9.10/synchronization/event_change/<event_id>`

- Ping request (from third-party client): HTTP HEAD to base URL

- Payload received:

  - With a Groovy script configured for the connected server, the Groovy script defines the payload.

  - Without a Groovy script configured for the connected server, the payload must be an `OprEventChange` object.

  See also "Event Change Creation: Logfile Adapter Examples" on page 360.

Web Service Ping Request

You can also send a ping request *from* a third-party application. When sending a ping request from a third-party application, the following data is relevant:

- HTTP Method: HEAD

- Base URL: `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization`

- Expected Payload: None

Event List Updates

The following URLs point to the sub-path `event_list`. Requests to this sub-path do not call the Groovy script and always have input or output of OPR data structures, for example, `OprEvent` and `OprEventList`.

Event Get

In the case of event get, the following data is relevant:

- HTTP Method: GET

- Base URL: `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_list/<event_id>`

- Payload returned:

  - The payload is always an `OprEvent` object.

  - The Groovy script is not called for the sub-path `event_list`.

Event Update

In the case of event updates, the following data is relevant:

- HTTP Method: PUT

- Base URL: `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_list/<event_id>`

- Payload received and returned:

  - The payload received must be an `OprEvent` object.

  - The payload returned is always an `OprEvent` object.

  - The Groovy script is not called for the sub-path `event_list`.

See also .

Event Submit

You can also submit an event *from* a third-party application. When submitting an event from a third-party application, the following data is relevant:

- HTTP Method: POST

- Base URL: `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_list`

- Payload received:

  - The payload must be an `OprEvent` or `OprEventList` object.

  - For an `OprEventList` object, the media type must be `text/xml;type=collection` or `application/xml;type=collection`.

  - The Groovy script is not called for the sub-path `event_list`.

See also .

Event List Changes

The following URLs point to the sub-path `event_change_list`. Requests to this sub-path do not call the Groovy script and always have input or output of OPR data structures, for example, `OprEvent` and `OprEventList`.

Event Change Creation

In the case of event changes to the sub-path `event_change_list`, the following data is

relevant:

- HTTP Method: POST

- Base URL: `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_change_list/`

- Payload received:

  - The payload received must be an `OprEventChange` or `OprEventChangeList` object..

  - For an `OprEventChangeList` object, the media type must be `text/xml;type=collection` or `application/xml;type=collection`.

  - The Groovy script is not called for the sub-path `event_change_list`.

  See also "Event Change Creation for event_change_list Examples" on page 370.

Configuring Connected Servers

The broad steps involved when using the Event Synchronization Web Service can be summarized as follows:

1. Configure a target server. You do this in the Connected Servers manager. The name of the server is the name that must be used when authenticating at the Event Synchronization Web Service. You also specify a password for this user during configuration of the connected server.

2. Forward the event to the target connected server. This can be done by using the Event Browser context menu to manually forward an event, or by configuring a forwarding rule and then triggering that rule.

3. Until an event is forwarded to the target connected server, the server cannot synchronize back any changes. In contrast to the Event Web Service (see "Automating Operator Functions using the Event Web Service Interface" on page 281), which allows the authenticated Operations Management user to view and update any event that the Operations Management user has access to, the connected server is only allowed to synchronize back changes for events that have been first forwarded to it.

4. The connected server synchronizes back changes to the event.

For details of how to configure connected servers, refer to the Operations Management online help.

Event Attributes that Support Back Synchronization

Supported Attribute Updates

Attribute Name	Supported Operations		
	Update	Create	Delete
`annotation`	Yes	Yes	
`assigned_user`	Yes		set to null
`assigned_group`	Yes		set to null

Attribute Name	Supported Operations		
	Update	Create	Delete
cause	Yes		set to null
custom attribute	Yes	Yes	
description	Yes	Yes	set to null
priority	Yes		
severity	Yes		
solution	Yes		set to null
state	Yes		
title	Yes		set to null
control_transferred_to	Yes		set to null

Event Update: Logfile Adapter Examples

Note: The examples showing event updates in this section are specific to the Logfile Adapter (the sample Groovy script adapter provided with the product), and would work only for the Logfile Adapter, or for accessing the Event Synchronization Web Service directly without having a Groovy script configured for the target connected server.

You can submit event changes with an event update. The HTTP method in this case is PUT, and the expected payload for the Logfile Adapter is an `OprEvent`. Each Groovy script adapter defines its own expected payload, so the expected payload would be different for other Groovy script adapters.

The Event Synchronization Web Service supports sending multiple properties in a single payload. The following examples each show just one changed property, with the exception of the last sample, which shows two changed properties (see "Change Multiple Properties in One Call" on page 360).

You can send test payloads to the Event Synchronization Web Service using the REST web service command-line utility `RestWsUtil`, supplied with the product. When using the utility, make sure that the event you are trying to update was first forwarded to the target connected server you are using to authenticate with the Event Synchronization Web Service.

For details about the `RestWsUtil` command-line utility, see "REST Web Service Command-Line Utility" on page 303.

An example call for an event update is shown below. The following sample call for an event update is for a connected server named `logger` with password set to `Password1`, and attempts to update an event with the ID `0695624b-93fa-40b1-8b0fc9b4ea07a4ec`. The sample call looks like this:

```
RestWsUtil -update update.xml-url http://<bsmserver.example.com>/opr-
gateway/rest/9.10/synchronization/event/0695624b-93fa-40b1-
8b0fc9b4ea07a4ec -username logger -password Password1 -verbose
```

The samples that follow show possible XML payloads for the call. The payloads are contained in the `update.xml` file referenced in the call to the `RestWsUtil` command-line utility.

> **Note:** An Event REST WS sample is located at:
>
> *<HPBSM>*`/opr/examples/event-ws`
>
> This can be compiled and packaged into a war file for deployment into a J2EE container. See the README.txt in this directory for instructions on how to build this webapp and deploy it for testing.
>
> A Event REST WS template is located at:
>
> *<HPBSM>*`/opr/examples/synchronization-ws`
>
> This can be used as a starting point for developers who want to develop their own Event Synchronization web service. See the README.txt in this directory for instructions on how to build this webapp. The "TODO" sections must be completed by the developer.

Set and Unset the Event Description

The Event Synchronization Web Service supports only the set or unset operations for the description of an event. Insert, update, or delete operations are not supported.

Set the Description

Here is a sample payload to set the description:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <description>This is a new description</description>
</event>
```

Unset the Description

Here is a sample payload to unset the description:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <description/>
</event>
```

Set and Unset the Event Title

The Event Synchronization Web Service supports only the set or unset operations for the title of an event. Insert, update, or delete operations are not supported.

Set the Title

Here is a sample payload to set the title:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <title>This is a new title</title>
</event>
```

Unset the Title

Here is a sample payload to unset the title:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <title/>
</event>
```

Change Multiple Properties in One Call

The Event Synchronization Web Service supports sending multiple properties in a single payload. Here we show just two changed properties.

Set Title and Description

The title and description can be set in a single call. The order of the attributes is unimportant and attributes not specified do not affect the current values.

Here is a sample payload to set the event title and description in a single update call:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
  <title>This is a new title</title>
  <description>This is a new description</description>
</event>
```

Event Change Creation: Logfile Adapter Examples

You can submit event changes with an event change creation. The HTTP method in this case is POST, and the expected payload for the LogfileAdapter is an `OprEventChange`. The examples in this section show sample XML payloads for each supported update. If a Groovy script is not configured for the connected server, the Event Synchronization Web Service expects an `OprEventChange` object. Each of the samples that follow represent an OprEventChange object.

The Event Synchronization Web Service supports sending multiple properties in a single payload. The following examples each show just one changed property, with the exception of the last sample, which shows two changed properties (see "Change Multiple Properties in One Call" on page 367).

The examples use an event ID of `531d2673-683f-429f-a742-b8680ee01a76`. Change this event ID to the ID of the specific event for which you wish to create a change object. The correct HTTP method for the creation of an event change is the POST method.

You can also use the event change list web service to get examples of event change objects. You can access the event change list web service using this URL:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_change_list
```

An example call for an event change creation is shown below.

The following sample call for an event change creation is for a connected server named `logger` with password set to `Password1`, and attempts to create an event change for an event with ID `531d2673-683f-429f-a742-b8680ee01a76`. For an event change, the event ID must be passed in the payload, and is not part of the URL. The sample call looks like this:

```
RestWsUtil -create create.xml-url http://<bsmserver.example.com>/opr-
gateway/rest/9.10/synchronization/event_change/<event id>-username
logger -password Password1 -verbose
```

The samples that follow show possible XML payloads for the call. The payloads are contained in the `create.xml` file referenced in the call to the `RestWsUtil` command-line utility.

Annotations

The Event Synchronization Web Service supports only insert and update operations for annotations. The delete operation is not supported.

Insert an Annotation

Here is a sample payload showing an annotation insert:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <annotation_property_change>
      <property_name>annotation</property_name>
      <current_value xsi:type="xs:string">This is a new
Annotation.</current_value>
      <change_operation>insert</change_operation>
    </annotation_property_change>
  </changed_properties>
</event_change>
```

Update an Annotation

Here is a sample payload showing an update to an annotation with the ID of `1c108839-9584-45f4-93ab-798bf49797c7`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <annotation_property_change>
      <property_name>annotation</property_name>
      <annotation_id>1c108839-9584-45f4-93ab-798bf49797c7</annotation_
id>
      <current_value xsi:type="xs:string">This is an updated
Annotation.</current_value>
      <change_operation>update</change_operation>
    </annotation_property_change>
```

```
    </changed_properties>
</event_change>
```

Cause

The Event Synchronization Web Service supports only the set or unset operations for the cause of an event. Insert, update, or delete operations are not supported. You set or unset the cause on the symptom event.

Set the Cause of an Event

Here is a sample payload showing how to set the cause of an event to be the event with the ID of 9915e504-f5a2-471a-ab98-6184a7382d32:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
      <property_name>cause</property_name>
      <current_value xsi:type="xs:string">9915e504-f5a2-471a-ab98-
6184a7382d32</current_value>
    </string_property_change>
  </changed_properties>
</event_change>
```

Unset the Cause of an Event

Here is a sample payload showing how to unset the cause of an event:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
      <property_name>cause</property_name>
      <current_value xsi:type="xs:string"/>
    </string_property_change>
  </changed_properties>
</event_change>
```

Custom Attributes

The Event Synchronization Web Service supports only insert and update operations for custom attributes. The delete operation is not supported.

Insert a Custom Attribute

Here is a sample payload showing how to insert a custom attribute, `custom_attribute`, with the value `Some CA value`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <custom_attribute_property_change>
      <property_name>custom_attribute</property_name>
      <key>TestCA</key>
      <current_value xsi:type="xs:string">Some CA value</current_
value>
      <change_operation>insert</change_operation>
    </custom_attribute_property_change>
  </changed_properties>
</event_change>
```

Update a Custom Attribute

Here is a sample payload showing how to update a custom attribute, `custom_attribute`, with the value `Some updated CA value`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <custom_attribute_property_change>
      <property_name>custom_attribute</property_name>
      <key>TestCA</key>
      <current_value xsi:type="xs:string">Some updated CA
value</current_value>
      <change_operation>update</change_operation>
    </custom_attribute_property_change>
  </changed_properties>
</event_change>
```

Description

The Event Synchronization Web Service supports only the set or unset operations for the description of an event. Insert, update, or delete operations are not supported.

Set the Description

Here is a sample payload showing how to set the description in an event change creation to `Some description of the event`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
      <property_name>description</property_name>
      <current_value xsi:type="xs:string">Some description of the
event.
      </current_value>
    </string_property_change>
  </changed_properties>
</event_change>
```

Unset the Description

Here is a sample payload showing how the description in an event change creation is unset:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
      <property_name>description</property_name>
      <current_value xsi:type="xs:string"/>
    </string_property_change>
  </changed_properties>
</event_change>
```

State

The only operation that is possible on the state of an event is to give the state a new value. Insert, update, or delete operations are not applicable for event state changes.

The possible values for the state of an event are: `open`, `in_progress`, `resolved`, `closed`.

Set the State

Here is a sample payload showing how the state in an event change creation is set to the value `in_progress`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <state_change>
      <property_name>state</property_name>
      <current_value xsi:type="xs:string">in_progress</current_value>
    </state_change>
  </changed_properties>
</event_change>
```

Priority

The only operation that is possible on the priority of an event is to give the priority a new value. Insert, update, or delete operations are not applicable for priority changes.

You can set the priority to one of the following values: `none`, `lowest`, `low`, `medium`, `high`, `highest`.

Set the Priority

Here is a sample payload showing how the priority in an event change creation is set to `low`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <priority_property_change>
      <property_name>priority</property_name>
      <current_value xsi:type="xs:string">low</current_value>
    </priority_property_change>
  </changed_properties>
</event_change>
```

Severity

The only operation that is possible on the severity of an event is to give the severity a new value. Insert, update, or delete operations are not applicable for severity changes.

You can set the severity to one of the following values: `critical`, `major`, `minor`, `warning`, `normal`.

Set the Severity

Here is a sample payload showing how the severity in an event change creation is set to `normal`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <severity_property_change>
      <property_name>severity</property_name>
      <current_value xsi:type="xs:string">normal</current_value>
    </severity_property_change>
  </changed_properties>
</event_change>
```

Solution

The Event Synchronization Web Service supports only the set or unset operations for the solution of an event. Insert, update, or delete operations are not applicable for solution changes.

Set a Solution

Here is a sample payload showing how the solution in an event change creation is set to `Some solution to the event`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
      <property_name>solution</property_name>
      <current_value xsi:type="xs:string">Some solution to the
event.</current_value>
    </string_property_change>
  </changed_properties>
</event_change>
```

Unset a Solution

Here is a sample payload showing how the solution in an event change creation is unset:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
```

```
        <property_name>solution</property_name>
        <current_value xsi:type="xs:string"/>
      </string_property_change>
    </changed_properties>
</event_change>
```

Title

The Event Synchronization Web Service supports only the set or unset operations for the title of an event. Insert, update, or delete operations are not applicable for title changes.

Set the Title

Here is a sample payload showing how the title in an event change creation is set to `Some title for the event`:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
      <property_name>title</property_name>
      <current_value xsi:type="xs:string">Some title for the
event.</current_value>
    </string_property_change>
  </changed_properties>
</event_change>
```

Unset the Title

Here is a sample payload showing how the title in an event change creation is unset:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
      <property_name>title</property_name>
      <current_value xsi:type="xs:string"/>
    </string_property_change>
  </changed_properties>
</event_change>
```

Change Multiple Properties in One Call

The Event Synchronization Web Service supports sending multiple properties in a single payload.

Here, we show two changed properties.

Set the Title and Description

Here is a sample payload to set the event title and description in a single update call:

```xml
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>
  <changed_properties>
    <string_property_change>
      <property_name>title</property_name>
      <current_value xsi:type="xs:string">Some title for the
event.</current_value>
    </string_property_change>
    <string_property_change>
      <property_name>description</property_name>
      <current_value xsi:type="xs:string">Some description for the
event.</current_value>
    </string_property_change>
  </changed_properties>
</event_change>
```

Event Update to event_list Example

You can update events using OPR data structure only (and not call the Groovy script). To update events using OPR data structures, call update using the `event_list` sub-path. The payload is the same as, for example, the standard put.

The title and description can be set in a single call as follows. The order of the attributes is unimportant and attributes not specified do not affect the current values.

```xml
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>This is a new title</title>
   <description>This is a new description</description>
</event>
```

Event Submit Examples

You can submit new events using the URL `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_list`. The HTTP method in this case is the POST method and the expected payload is an `OprEvent` or `OprEventList`. For an `OprEventList` object, the media type must be `text/xml;type=collection` or `application/xml;type=collection`.

If the submitter does not add an `OprForwardingInfo` entry for itself in the event, one will be automatically added with `ForwardingType` set to notify. This allows the caller to subsequently read the event from the Event Synchronization Web Service.

The following samples show the XML payload. The correct HTTP method to set is the POST method.

Submit a New Event

Here is a sample payload showing submittal of a single event:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>This is a new event</title>
   <description>This is a description</description>
</event>
```

Submit a New Event with Synchronization Requested

Here is a sample payload showing an event submittal with synchronization requested:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>This is a new event</title>
   <description>This is a description</description>
   <forwarding_info_list>
      <forwarding_info>
         <dns_name>extclt.example.com</dns_name>
         <external_id>IM10219</external_id>
         <external_url>http://extclt.example.com:8081/webtier-
9.20/index.d-
o?ctx=-
docEngine&file=probsummary&query=number%3D%22IM10219%22</external_url>
         <forwarding_type>synchronize</forwarding_type>
         <state>originated</state>
      </forwarding_info>
   </forwarding_info_list>
</event>
```

Submit a New Event with Synchronization and Transfer Control Requested

Here is a sample payload showing an event submittal with synchronization and transfer control requested:

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>This is a new event</title>
   <description>This is a description</description>
   <forwarding_info_list>
      <forwarding_info>
         <dns_name>extclt.example.com</dns_name>
         <external_id>IM10219</external_id>
         <external_url>http://extclt.example.com:8081/webtier-
9.20/index.d-
```

```
o?ctx=-
docEngine&file=probsummary&query=number%3D%22IM10219%22</external_url>
        <forwarding_type>synchronize_and_transfer_
control</forwarding_type>
        <state>originated</state>
      </forwarding_info>
    </forwarding_info_list>
</event>
```

Submit a List of New Events

The media type for `OprEventList` objects must be `text/xml;type=collection` or `application/xml;type=collection`.

Here is a sample payload showing submittal of a list of events:

```
<event_list xmlns="http://www.hp.com/2009/software/opr/data_model">
   <event><title>e0</title><severity>critical</severity></event>
   <event><title>e1</title><severity>normal</severity></event>
   <event><title>e2</title><severity>major</severity></event>
   <event><title>e3</title><severity>minor</severity></event>
   <event><title>e4</title><severity>warning</severity></event>
</event_list>
```

Event Change Creation for event_change_list Examples

You can submit new events changes using the URL `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_change_list`. The HTTP method in this case is the POST method and the expected payload is an `OprEventChange` or `OprEventChangeList`. For an `OprEventChangeList` object, the media type must be `text/xml;type=collection` or `application/xml;type=collection`.

The following samples show the XML payload. The correct HTTP method to set is the POST method.

Submit a New Event Change

Here is a sample payload showing submittal of a title change:

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <event_ref>
      <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
   </event_ref>
   <changed_properties>
      <string_property_change>
         <property_name>title</property_name>
         <current_value xsi:type="xs:string">Some title for the
```

```
event.</current_value>
      </string_property_change>
   </changed_properties>
</event_change>
```

Submit a List of New Event Changes

The media type must be `text/xml;type=collection` or
`application/xml;type=collection`.

Here is a sample payload showing submittal of a title change for one event, and a state change for
another event:

```
<event_change_list xmlns="http://www.hp.com/2009/software/opr/data_
model">
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <event_ref>
      <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
   </event_ref>
   <changed_properties>
      <string_property_change>
         <property_name>title</property_name>
         <current_value xsi:type="xs:string">Some title for the
event.</current_value>
      </string_property_change>
   </changed_properties>
   </event_change>
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <event_ref>
      <target_id>ad726f4f-ba51-409f-b429-b445b791ac9d</target_id>
   </event_ref>
   <changed_properties>
      <state_change>
         <property_name>state</property_name>
         <current_value xsi:type="xs:string">in_progress</current_
value>
      </state_change>
   </changed_properties>
</event_change>
</event_change_list>
```

Integrating External Event Processes: Frequently Asked Questions

This section contains a collection of frequently asked questions (FAQs) related to integrating external event processes. The FAQs are intended to assist developers writing scripts for a connected server configured for external event processing.

This section is structured as follows:

Getting Started

This section contains FAQs related to basic information required to get started.

Is there any documentation available?

Yes. Read this guide (Operations Manager i Extensibility Guide).

- For information about the event forwarding and synchronization process, see "Forwarding Events and Synchronizing Event Changes" on page 330.

- For information about the Event Synchronization Web Service, see "Event Synchronization Web Service Interface" on page 352.

- For information about integrations using Groovy scripts, see "Integrating External Event Processes Using Groovy Scripts" on page 335.

Is there any Javadoc for the API?

Yes. You can find the Javadoc here:

<HPBSM root directory>`/opr/api/doc/opr-external-api-javadoc.zip`

You need to unzip this file.

Is there a schema available for the OprEvent and OprEventChange object?

Yes. You can find the schema here:

<HPBSM root directory>`/opr/api/schema/OprDataModel.xsd`

Will I need a schema?

If you integrate an external application using a Groovy script, you most likely do *not* need the OprEvent schema. The schema is mainly needed if you intend your integration to communicate directly with the Event Synchronization Web Service endpoints using `OprEvent` and

`OprEventChange` objects with no intervention by a Groovy script. For example, if you implement a REST web service that will accept the HTTP POST method call of an `OprEvent` object (as described in the schema), and a REST client that either performs HTTP POST methods calls of `OprEventChange` objects, or performs HTTP PUT method calls for `OprEvent` objects to submit those objects with changes to the Event Synchronization Web Service. In this case, the connected server would be configured for a web service integration and not a Groovy script integration.

Additionally, if your client or service is written in Java, you can directly use the Java Architecture XML Binding (JAXB) annotated objects in the `opr-external-api.jar` file to marshal and unmarshal the XML. Only if you are programming in a different language should you need to use the schema directly.

Is there a Web Services Description Language (WSDL) available?

No. The Event Synchronization Web Service is a simple REST-based web service. In general, REST-based web services do not have a WSDL.

Where can I find out more about REST?

A good place to start is the following Wikipedia entry:

http://en.wikipedia.org/wiki/Representational_State_Transfer

Are there any good toolkits to use for creating a REST client or server?

Use the Apache Wink toolkit, which you can get here:

http://incubator.apache.org/wink/index.html

Are there JAXB annotated classes already available for OprEvent and OprEventChange?

Yes. The `<HPBSM root directory>/lib/opr-external-api.jar` file contains JAXB annotated classes. If you are programming in Java, you can use these classes directly instead of generating classes from the schema. See the Javadoc API Documentation for details about the classes provided.

Are there any sample implementations?

Yes. Take a look at the sample located here:

```
<HPBSM root
directory>/conf/opr/integration/sample/LogfileAdatper.groovy
```

This is a sample Groovy script adapter that accepts forward requests and writes them to a log file.

How do I configure and use the Logfile Adapter sample?

When configuring this adapter, configure the connected server with the following parameters:

- node - localhost (do not use the DNS name here)

- port - 80 for standard http, 443 if SSL is enabled

- drilldown node - `<DNS name of your Operations Management node>`

- drilldown port - 80 for standard http, 443 if SSL is enabled

- **Enable Synchronize and Transfer Control** is selected

After you configure the connected server, you can forward events manually from the event browser to the Logfile Adapter. Just select **Transfer Control to ...** from the Event Browser context menu. You can also configure an automatic forwarding rule using the Event Forwarding manager.

Once an event has been forwarded, you should see a new tab, called External Info, in the Event Details of the for the forwarded event. Selecting this tab calls the `LogfileAdapter.groovy` script `getExternalInfo()` method to fetch the external information for this event and then display it in the tab. If the call succeeds, the fields on the left will be populated with data.

I wrote a client to access the Event Synchronization Web Service to update an event, but I always receive an HTTP 401, Unauthorized error message. How do I authenticate?

Make sure you have configured a password for your connected server. This is the password on the last page of the wizard, or the "Incoming Connection" page. You can also find the user name on this page.

Make sure the event you are updating has first been successfully forwarded to the connected server you are authenticating.

I wrote a Groovy integration script. How do I install and test it?

- Create a new directory in the following location:

  <HPBSM root directory>`/conf/opr/integration`

- Add your script to the new directory.

- Configure a new external process connected server and select your script from the drop-down menu.

- Create a forwarding rule with the title "`my_forward_test`" to forward events automatically to your target connected server.

- Use the `sendEvent` tool to create a test event:

  <HPBSM root directory>`/opr/support/sendEvent.bat -t "my_forward_`
  `test"`

- Check to see if the event was delivered to your server.

Alternatively, you can select **Enable Synchronize and Transfer Control** in the Connected Server Outgoing Connection screen. You can then manually forward the event from the Event Browser. You can then also check the external event by selecting the External Info tab.

I automatically forwarded an event to my external server, but I cannot see the External Info tab in the Event Browser. What happened?

The External Info tab is only for events that have transferred control to an external server. If you forwarded the event using one of the other types of forwarding, for example, Notify, Notify and Update, or Synchronize, then you will not be able to get the external status on the External Info tab. Currently, there is no interface in the Event Browser to view other types of forwarding.

Can I forward an event to more than one external system?

Yes. You can forward an event to as many systems as you like, but you can transfer control to only one system.

I transferred control to an external system. Can I take it back?

This is currently not possible.

I transferred control to an external system. Can that system give it back?

Yes. The external system must generate an update using the Event Synchronization Web Service. When the Groovy script method `receiveChange()` is called, you can set the `args` property as like this:

```
args.returnControl
```

This returns control back to the local Operations Management instance, assuming the connected server had control and is logged in.

Groovy Scripts and Programming

This section contains frequently asked questions related to Groovy scripts and programming.

Where can I find out more about Groovy programming?

There is a beginners tutorial that you can access here:

http://groovy.codehaus.org/Beginners+Tutorial

I see in the Logfile Adapter sample that the "?" character is sometimes used. What does it mean?

Consider this line of code:

```
def username = args.credentials?.userName
```

If the value of `credentials` is null, there will be no attempt at runtime to de-reference it. Null will simply be assigned to `username`. If the value of `credentials` is not null, `userName` will be assigned to `username`.

The payload being sent by my connected server is XML. How can I parse it from the Groovy script?

A simple way to parse the XML from Groovy is to use the XmlSlurper. See the sample code below and the `LogfileAdapter.receiveChange()` method.

The following sample assumes a payload with an OPR event (XML). This payload will vary depending upon the connected server sending the update.

```
  def receiveChange(def args) {
    def timestamp = new Date()
    def externalEvent = args.externalEventChange
    def msg = """### ${timestamp.toString()}: receiveChange() called
###
    parameter externalEvent: ${externalEvent}\n\n"""
    m_logfile.append(msg)
    if ((externalEvent == null) || (externalEvent.length() == 0))
      return false;
    // check if this is an event or event_change
    def xmlNode = new XmlSlurper().parseText(externalEvent);
    if (xmlNode.name().equals("event"))
      return handleEvent(args, xmlNode)
    else if (xmlNode.name().equals("event_change"))
      return handleEventChange(args, xmlNode)
    else {
      def err = "Unexpected object type: ${obj.getClass
().canonicalName}"
```

```
      m_logfile.append("${err}\n\n")
      m_logger.error(err);
      return false
    }
  }
  def handleEvent(def args, def event) {
    m_logger.debug("Change request received with ${EVENT_TAG}
record.")
    // Update the event properties if present in XML
    if (event."title".size())
      args.title = event."title".text()
    if (event."description".size())
      args.description = event."description".text()
    if (event."solution".size())
      args.solution = event."solution".text()
    if (event."severity".size()) {
      def text = event."severity".text()
      def severity = severityMap."${text}"
      if (severity)
        args.severity = severity
      else {
        args.setHttpResponseStatus(400, "Invalid severity: ${text}")
        return false
      }
    }
...
```

I want to map my external process severity to an Operations Management event severity. Is there an easy way to do that in Groovy?

Use a map as follows:

```
// Map for severity mapping
  static def severityMap = ["0": OprSeverity.unknown,
                            "1": OprSeverity.normal,
                            "2": OprSeverity.warning,
                            "3": OprSeverity.minor,
                            "4": OprSeverity.major,
                            "5": OprSeverity.critical]
...
def externalSeverity = "2"
def oprSeverity = severityMap."${externalSeverity}"
```

Integration Script Methods

This sections contains frequently asked questions related to integration script methods.

What script methods must I implement?

You must implement the following methods in any integration script:

- `init()`

- `destroy()`

- `forwardEvent()`

For more information about script methods, see "Groovy Script Methods" on page 339.

Which script methods are optional?

The following script methods are optional:

- `forwardChange()`: Only needed if your script adapter supports the following forwarding modes: `Notify and Update`, `Synchronize`, or `Synchronize and Transfer Control`.

- `receiveChange()`: Only needed if your adapter supports the following forwarding modes: `Synchronize` or `Synchronize and Transfer Control`.

- `getExternalEvent()`: Only needed if your script adapter supports populating the External Info tab in the Event Browser.

- `toExternalEvent()`: Only needed if your connected server needs to perform a GET HTTP method call at the Event Synchronization Web Service, to retrieve the current properties of the event that originated in Operations Management.

Must my class implement the EventProcessAdapter interface?

No. Your script must only implement the required methods. The interface is provided for documentation and for those that wish to use an integrated development environment (IDE) such as Eclipse or IntelliJ.

When do init() and destroy() get called?

The `init()` method gets called when the script is first loaded. There is one instance loaded per connected server and done so on the first access to the connected server. It remains loaded until the connected server configuration is updated, or the script is changed. At that time, the `destroy()` method is called, and the script is reloaded. Once loaded, the script may maintain state between calls, for example, it may leave connections open to a remote server and reuse those connections on subsequent calls.

What properties are available in the init() method argument?

For available properties for the `init()` method, see "Properties for init() Method" on page 340.

Example:

```
def init(def args) {
    m_logger = args.logger
    m_initArgs = args
    def logfileDir = new File("${args.installDir}${File.separator}
${LOG_DIR_REL}")
    if (!logfileDir.exists())
      logfileDir.mkdirs()
    m_logfile = new File(logfileDir, LOGFILE_NAME)
    if (!m_logfile.exists())
      m_logfile.createNewFile()
    m_logger.debug("Logfile Adapter initalized. INSTALL_
DIR=${args.installDir}")
    def timestamp = new Date()
    def msg = """### ${timestamp.toString()}: init() called ###
    parameter connected server ID: ${m_initArgs.connectedServerId}
```

```
    parameter connected server name: ${m_initArgs.connectedServerName}
    parameter connected server display name: ${m_
initArgs.connectedServerDisplayName}
    parameter node: ${m_initArgs.node}
    parameter port: ${m_initArgs.port}
    parameter ssl:  ${m_initArgs.nodeSsl}
    parameter drilldown node: ${m_initArgs.drilldownNode}
    parameter drilldown port: ${m_initArgs.drilldownPort}
    parameter drilldown ssl:  ${m_initArgs.drilldownNodeSsl}\n\n"""
    m_logfile.append(msg)
  }
```

For details, see the `com.hp.opr.api.ws.adapter.InitArgs` Javadoc.

What properties are available in the ping() method argument?

For available properties for the `ping()` method, see

Example:

```
def ping(def args) {
    args.outputDetail = "success"
    return true
}
```

For details, see the `com.hp.opr.api.ws.adapter.PingArgs` Javadoc.

What properties are available in the forwardEvent() method argument?

For available properties for the forwardEvent() method, see

Example:

```
def forwardEvent(def args) {
    def timestamp = new Date()
    def extId = "urn:uuid:${args.event.getId()}"
    def msg = """### ${timestamp.toString()}: forwardEvent() called
###
    event.id: ${args.event.id}
    event.title: ${args.event.title}
    event.state: ${args.event.state}
    event.external.id: ${extId}\n\n"""
    m_logfile.append(msg)
    // Set the external reference ID
    args.externalRefId = extId
    // Make a drilldown to the original event as an example
    args.drilldownUrl =
      new URL("http://${m_initArgs.drilldownNode}:${m_
initArgs.drilldownPort}${ROOT_DRILLDOWN_PATH}${args.event.getId()}")
    return true
  }
```

For more details on the argument passed to the method `forwardEvent()`, see the
`com.hp.opr.api.ws.adapter.ForwardEventArgs` Javadoc.

What properties are available in the forwardChange() method argument?

For available properties for the `forwardChange()` method, see "Properties for forwardChange() Method" on page 344.

Example:

```
def forwardChange(def args) {
    def timestamp = new Date()
    StringBuffer buff = new StringBuffer()
    buff.append("### ${timestamp.toString()}: forwardChange() called
###\n")
    buff.append("    parameter externalRefId: ${args.externalRefId}
\n")
    buff.append("    change headline: ${args.changes.headline}\n")
    args.changes.changedProperties.each {
      def propertyChange ->
      buff.append("    changed property:
${propertyChange.propertyName}=${propertyChange.currentValue}\n")
    }
    buff.append("\n")
    m_logfile.append(buff.toString())
    return true
  }
```

For more details on the argument passed to the method `forwardChange()`, see the `com.hp.opr.api.ws.adapter.ForwardChangeArgs` Javadoc.

What properties are available in the receiveChange() method argument?

For available properties for the `receiveChange()` method, see "Properties for receiveChange() Method" on page 346.

Example:

```
def receiveChange(def args) {
    def timestamp = new Date()
    def msg = """### ${timestamp.toString()}: receiveChange() called
###
    parameter externalEvent: ${args.getExternalEventChange()}\n\n"""
    m_logfile.append(msg)
    def jc = javax.xml.bind.JAXBContext.JAXBContext.newInstance(
com.hp.opr.api.ws.model.event.OprEvent.class)
    def event = jc.createUnmarshaller().unmarshal(
new CharArrayReader(args.externalEventChange.toCharArray()))
    if (event instanceof com.hp.opr.api.ws.model.event.OprEvent) {
      if (event.titleUpdated)
        args.title = event.title
      if (event.descriptionUpdated)
        args.description = event.description
      if (event.solutionUpdated)
        args.solution = event.solution
      return true
    } else {
```

```
        def err = "Unexpected object type: ${obj.getClass
().canonicalName}"
        m_logfile.append("${err}\n\n")
        m_logger.error(err);
        return false
    }
  }
```

For more details on the argument passed to the method `receiveChange()`, see the
`com.hp.opr.api.ws.adapter.ReceiveChangeArgs` Javadoc.

When processing receiveChange(), I would like to send back a particular response to the web
service caller. How can I do that?

The `args` has a method to allow you to control the response:

```
args.setHttpResponseStatus(400, "My response message")
```

You can set the HTTP status and payload to anything you wish. If the value is less than 300, the
payload is processed after the receiveChange() method is called. Then the status and message are
returned to the web service caller, otherwise the HTTP status and message are returned
immediately to the we service caller.

What properties are available in the getExternalEvent() method argument?

For available properties for the `getExternalEvent()` method, see "Properties for
getExternalEvent() Method" on page 350.

Example:

```
def getExternalEvent(def args) {
    def timestamp = new Date()
    def msg = """### ${timestamp.toString()}: getExternalEvent()
called ###\n\n"""
    m_logfile.append(msg)
    args.assignedUser = "logger"
    args.assignedGroup = "logging group"
    args.state = "open"
    args.severity = "normal"
    args.priority = "none"
    return true
  }
```

For more details on the argument passed to the method `getExternalEvent()` see the
`com.hp.opr.api.ws.adapter.GetExternalEventArgs` Javadoc.

Event Properties

This sections contains frequently asked questions related to event properties.

What event properties exist?

All properties available in the Event Browser, or that can be found in the Event Web Service are
available in the `OprEvent` object. Details can be found in the Javadoc for `OprEvent`.

For an example go to the Event Web Service and list the events:

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
```

The XML output will give you a good idea of the properties that are available. This XML output is directly generated from the `OprEvent` object.

I want to see what is the related CI for this event. How can I get this information?

`event.relatedCi` returns an object with properties describing the related CI. It is of type `OprRelatedCi`. `event.relatedCi.configurationItem` contains the key properties of the CI, and, if the CI is part of another CI, it contains the CI it is part of: `event.relatedCi.configurationItem.partOf`. "partOf" is of type `OprRelatedCi`, so this will continue until there are no more parts. This should provide you with enough details to identify the CI in an external system.

An OprConfigurationItem object does not define all key properties for all CIs. How do I get the other key properties?

Use the `OprConfigurationItem` utility method to get the other properties: `getProperty (name)`, or for a map of all properties call `getProperties()`.

The OprConfigurationItem in the event only has the key properties. How do I get the other properties?

Call the utility method `getCi(id)` for the CI you want. All properties will be set in the CI that is returned from this method. This utility method is available in the `args` for `forwardEvent()`, `forwardChange()`, and `receiveChanges()`.

What are the possible class types that can be returned in the JAXBElements returned from OprConfigurationItem.getAny()?

The possible class types are:

- String

- Boolean

- Integer

- Long

- Float

- Double

- Date

OprConfigurationItem.getAny() returns multiple objects with the same name. How can this happen?

If the CI property is a list, you will get multiple entries.

Troubleshooting

This section contains frequently asked questions related to troubleshooting the connected server.

I configured a connected server, but it does not show up in the Event Browser context menu item "Transfer Control to". Why?

Check the following:

- Make sure the connected server is "Active". Found on "General" tab.

- Make sure the connected server supports "Ownership Transfer". Found on "Outgoing Connection" tab.

I have forwarded an event to my external connected server. How can I tell if my script was called?

Try the following:

- Switch logging to debug level.

- Check the logfile.

Logging

This section contains frequently asked questions related to log files.

Where can I view the log file for script execution?

You can view the log file at this location:

<HPBSM root directory>`/log/opr-event-sync-adapter.log`

How can I change the logging level?

For `getExternalEvent()` method calls, you need to edit the following properties file:

<HPBSM root directory>`/conf/core/Tools/log4j/EJB/opr-event-sync-adapter.properties`

For all other method calls, edit the following properties files:

<HPBSM root directory>`/conf/core/Tools/log4j/wde/opr-event-sync-adapter.properties`

<HPBSM root directory>`/conf/core/Tools/log4j/opr-ctxm-server/opr-event-syncadapter.properties`

You need to set the `loglevel` parameter towards the top of the file. The file contains possible values.

How can I log from my script?

The `args` passed to the `init()` method has a property called `logger`. Use this logger for logging. For example:

```
def logger = args.logger
logger.info("This is an info log")
logger.warn("This is a warning log")
logger.error("This is a error log")
logger.debug("This is a debug log")
logger.error("This is a error log with an exception", exception)
```

Integrating an External Event Processing Service Defined by a WSDL

You can integrate an external event processing system that exposes its interfaces using a standard web service and a WSDL (Web Services Description Language) description using the steps described in this section.

HP recommends that you implement the integration in stages that correspond to the four defined forwarding types. Each stage builds upon the previous to create a more fully functional integration.

- **All forwarding types.** You must implement the Groovy script `init()` method for all forwarding types. The method is called whenever the Groovy script is initialized.

- **Notifiy.** This is the minimum and requires implementing the `forwardEvent()` method in the Groovy script to forward events to the target server.

- **Notify and Update.** This requires implementing the `forwardChange()` method in the Groovy script to forward changes to the target server.

- **Synchronize.** This requires implementing the `receiveChange()` method in the Groovy script to receive changes from the target server.

  The target server must be able to call the OPR Event Synchronization Web Service to post the changes to Operations Management when a change occurs on the target server. The payload of the web service request is passed to the Groovy script `receiveChange()` method for interpretation. The web service call may be of any type, for example SOAP or REST based.

- **Synchronize and Transfer Control.** No additional Groovy script methods are required for this implementation.

- *Optional.* **Ping support.** This requires implementing the Groovy script `ping()` method.

For more information about Groovy script methods, see "Groovy Script Methods" on page 339.

Here is a summary of the configuration steps required to implement an integration for the notify forwarding type:

- "Generate Java Code from WSDL" below

- "Deploy the JAR Files" on page 385

- "Modify the Groovy Script to Access the Service " on page 385

- "Configure the External Event Processing Application as a Connected Server" on page 385

- "Test the External Event Creation" on page 387

To support the other forwarding types, the appropriate methods in the Groovy script must be implemented in a similar fashion as the `forwardEvent()` method.

Generate Java Code from WSDL

The following example uses Apache Axis to generate Java code from a WSDL file.

1. Download and install the following prerequisites:

   Java JDK 1.6

   Apache Axis2

   Apache Ant 1.7

2. Create a working directory, for example `integration`.

3. Copy the WSDL file to that directory, for example `integration/service.wsdl`.

4. Create a directory for the generated code, for example `integration/gen`.

5. Change to the `gen` directory.

6. Run Apache Asix2 and generate the code from the WSDL file:

   ```
   wsdl2java -uri file:../service.wsdl
   ```

7. Edit the generated `build.xml` file to correctly set the `classpath` in the manifest of the JAR file:

   a. Open the `build.xml` file in a text editor.

   b. Locate the Ant target `jar.client`.

   c. Add the following Ant path declaration just after the path declaration `axis2.class.path` at the beginning of the file:

   ```
   <path id="axis2.client.class.path">
      <fileset dir="${axis2.home}">
         <include name="lib/*.jar"/>
      </fileset>
   </path>
   <pathconvert property="axis2.client.class.path.string"
                pathsep=" ">
      <path refid="axis2.client.class.path" />
      <flattenmapper />
   </pathconvert>
   ```

   d. In the jar task, add the following manifest directive to specify the *classpath* in the JAR file manifest.

   As there are many Axis2 JAR files needed at execution time, it is simpler if the manifest of the generated JAR file is able to resolve the dependencies at runtime.

   ```
   <jar destfile="${lib}/${name}-test-client.jar">
      <fileset dir="${classes}">
         <exclude name="**/META-INF/*.*"/>
         <exclude name="**/lib/*.*"/>
         <exclude name="**/*MessageReceiver.class"/>
         <exclude name="**/*Skeleton.class"/>
      </fileset>
      <manifest>
         <attribute name="Created-By"
                    value="Developer name goes here" />
         <attribute name="Class-Path"
   ```

```
                                value="${axis2.client.class.path.string}" />
        </manifest>
    </jar>
```

e. Save the changed `build.xml` file.

f. Run `ant` at the command line. This builds the jar file needed to access the service. The output is available the `build/lib` directory.

Deploy the JAR Files

1. Create new integration directories on the BSM gateway server test system:

   <HPBSM root directory>`/conf/opr/integration/test`

   <HPBSM root directory>`/conf/opr/integration/test/lib`

2. Copy the generated JAR file to the new `lib` directory.

3. Copy the Axis2 JAR files `AXIS2_HOME/lib/*` to the new `lib` directory.

Modify the Groovy Script to Access the Service

1. Make a copy of the Logfile Adapter Groovy script:

   <HPBSM root directory>`/conf/opr/integration/sample/LogfileAdapter.groovy`

   For example:

   <HPBSM root directory>`/conf/opr/integration/test/TestAdapter.groovy`

2. Edit the new `TestAdapter.groovy` script and add calls to the classes in the Axis2 generated JAR file:

   - You must first call the `forwardEvent()` script method. You need to create a target object for a given Operations Management event.

   - The Axis2 code generator generated a stub class. You need to construct this class and then, using the event passed in the `args` to `forwardEvent()`, create an external event.

   - Each service will be different, so there are no specifics to supply here. It is expected that once the external event is successfully created the `externalID` is set in the `args` before the method is exited.

Configure the External Event Processing Application as a Connected Server

Synchronizing events and event changes between Operations Management and the external event processing application depends on a server hosting Operations Management forwarding events to the external event processing application. To achieve this you must configure the external event processing application as a target connected server in the Connected Servers manager.

For full details about how to configure a connected server, see the Connecting Servers section of the Operations Management online help.

To configure the external event processing application server as a target connected server, perform the following steps:

1. Navigate to the Connected Servers manager in the Operations Management user interface:

   Admin > Operations Management > Setup > Connected Servers

2. Click the New ( ) button to open the Create New Server Connection dialog.

3. In the Display Name field, enter a name for the target the external event processing application server. By default, the Name field is filled automatically. For example, if you enter `ExtEvtProcAppServer` as the Display Name for the target the external event processing application server, `ExtEvtProcAppServer` is automatically inserted in the Name field. Of course, you can specify your own name in the Name field, if you want to change it from the one suggested automatically.

   Optional: Enter a description for the new target server.

   Make sure that you check the Active checkbox.

   Click **Next**.

4. Select `External Event Processing` to choose the server type suitable for an external event processing application.

   Click **Next**.

5. Enter the Fully Qualified DNS Name of the external event processing application target server.

   Click **Next**.

6. Establish the type of integration. In the Integration Type dialog, you can choose between using a Groovy script adapter, or the Event Synchronization Web Service.

   a. Select **Call Groovy Script Adapter**.

   b. Select the external event processing type, for example **test**. (The entries in the drop-down list correspond to the directories that contain the files for the integrations.)

   c. In the Groovy Script File Name field, select **TestAdapter.groovy**.

   d. Set the Groovy Classpath field to: *<HPBSM root directory>*`/conf/opr/integration/test/lib/`*<your jar file>*

   e. Click **Next**.

7. In the Outgoing Connection dialog, provide the credentials (user name, password, and port number) to connect to the external event processing application target server and to forward events to that server.

   Select **Enable Synchronize and Transfer Control** for initial testing. When the Enable Synchronize and Transfer Control flag is set, an Operations Management operator is then able to transfer ownership of the event to the target connected server. If the flag is not set, then the option Synchronize and Transfer Control does not appear in the list of forwarding types when configuring forwarding rules.

   Also, note that if the Enable Synchronize and Transfer Control flag is not set for any target connected server, the Transfer Control to option does not appear at all in the Event Browser context menu.

If a specific server is configured without the Enable Synchronize and Transfer Control flag set, then that server is not available in the Event Browser context menu as a server to which you can transfer ownership.

Click **Next**.

8. Complete the remaining dialogs and then click **Finish**.

    The target external event processing application server appears in the list of Connected Servers.

Test the External Event Creation

1. On the system running Operations Management, open an Event Browser.

2. Select an event.

3. Right-click the event and select **Transfer Control > *<external event processing application target server>***.

4. Verify that the event appears in the external event processing application target server.

Service Manager Integration

This section describes how to connect to HP Service Manager and to forward events to HP Service Manager, and how forwarded events and subsequent event changes are synchronized back from HP Service Manager to Operations Management.

Here is a summary of the configuration steps required:

- "Configure the HP Service Manager Server as a Connected Server" below
- "Configure an Event Forwarding Rule" on page 390
- "Configure HP Service Manager Server" on page 393

Configure the HP Service Manager Server as a Connected Server

Synchronizing events and event changes between Operations Management events and HP Service Manager incidents requires configuring a Connected Server within OMi to correctly identify the target Service Manager instance. The first step to achieve this is to configure HP Service Manager as a target connected server in the Connected Servers manager.

For full details about how to configure a connected server, see the Connecting Servers section of the Operations Management online help.

To configure the HP Service Manager server as a target connected server, perform the following steps:

1. Navigate to the Connected Servers manager in the Operations Management user interface:

   Admin > Operations Management > Setup > Connected Servers

2. Click the New (🌼) button to open the Create New Server Connection dialog.

3. In the **Display Name** field, enter a name for the target HP Service Manager server. By default, the Name field is filled automatically. For example, if you enter `Service Manager 1` as the Display Name for the target HP Service Manager server, `Service_Manager_1` is automatically inserted in the Name field. Of course, you can specify your own name in the Name field, if you want to change it from the one suggested automatically.

   > **Note:** Make a note of the name of the new target server (in this example, `Service_Manager_1`). You need to provide it later on as the `username` when configuring the HP Service Manager server to communicate with the server hosting Operations Management.

   Optional: Enter a description for the new target server.

   Make sure that you check the **Active** checkbox.

   Click **Next**.

4. Select `External Event Processing` to choose the server type suitable for an external incident manager like HP Service Manager.

   Click **Next**.

5. Enter the Fully Qualified DNS Name of the HP Service Manager target server.

   Click **Next**.

6. Next, you need to establish the type of integration. In the Integration Type dialog, you can choose between using a Groovy script adapter, or the Event Synchronization Web Service.

   a. As a Service Manager Groovy script adapter is provided for integrating with HP Service Manager, select **Call Script Adapter**.

   b. In the Script Name field, select **sm:ServiceManagerAdapter**.

   c. Click **Next**.

7. In HP Service Manager, set up an Integration User with user name and password. This is the user name and password needed to access the HP Service Manager target server.

8. In the Operations Management user interface, the next step is to provide the credentials (user name, password, and port number) to connect to the HP Service Manager target server and to forward events to that server. In the Outgoing Connection dialog, enter the following values:

   a. In the **User Name** field, enter the user name for the Integration User you set up in HP Service Manager.

   b. In the **Password** field, enter the password for the user you just specified. Repeat the password entry in the **Password (Repeat)** field.

   c. In the **Port** field, specify the port configured on the HP Service Manager side for the integration with Operations Management. To find the port number to enter:

      ○ Go to the following file:

        <HP Service Manager root directory>`/HP/Service Manager`
        <version>`/Server/RUN/sm.ini`

      ○ In the `sm.ini` file, you will find two port entries, depending on whether you want to use a secure HTTP connection: the `httpPort`, with default port number `13080`, and `httpsPort`, with default port number `13443`. The actual values for the ports can differ from these default values depending on how they are configured. Enter the appropriate value in the Port field.

   d. If you do not want to use secure HTTP, make sure that the Use secure HTTP checkbox is *not* checked.

      If Use Secure HTTP is selected, download and install a copy of the target server's SSL certificate by clicking the link "Retrieve from Server", or "Import from File" if the certificate is available in a local file.

   e. Make sure that the **Supports Synchronize and Transfer Control** checkbox is checked. When the Supports Synchronize and Transfer Control flag is set, an Operations Management operator is then able to transfer ownership of the event to the target connected server. If the flag is not set, then the option Synchronize and Transfer Control does not appear in the list of forwarding types when configuring forwarding rules.

      Also, note that if the Supports Synchronize and Transfer Control flag is not set for any target connected server, the Transfer Control to option does not appear at all in the Event Browser context menu.

If a specific server is configured without the Supports Synchronize and Transfer Control flag set, then that server is not available in the Event Browser context menu as a server to which you can transfer ownership.

f. Test the connection.

g. Click **Next**.

9. If, in addition to automatically generating HP Service Manager incidents from OMi events, you want to also be able to drill-down into HP Service Manager, you need to specify the fully qualified DNS name and port of the HP Service Manager system where you want to perform incident drill-down.

> **Note:** To enable incident drill-down to HP Service Manager, you must install a web tier client for your HP Service Manager server according to your HP Service Manager server install/configuration instructions.
>
> In the Event Drilldown dialog of the Connected Servers manager, configure the server where you installed the web tier client along with the configured port used.
>
> If you do not specify a server in the Event Drilldown dialog of the Connected Servers manager, it is assumed that the web tier client is installed on the server used for forwarding events and event changes to HP Service Manager, and receiving event changes back from HP Service Manager.
>
> If nothing is configured in the Event Drilldown dialog, and the web tier client is not installed on the HP Service Manager server machine, the web browser will not be able to find the requested URL.

Click **Next**.

10. The next thing to do is to enable event changes to be synchronized back from HP Service Manager to Operations Management. For this you need to provide credentials for the HP Service Manager server to access the server hosting Operations Management.

a. In the Incoming Connection dialog, select the **Support Event BackSync** checkbox, and then enter a password that the HP Service Manager server requires to connect to the server hosting Operations Management, `Myqwer1_` in this example.

> **Note:** Make a note of this password (in this example, `Myqwer1_`). You need to provide it later on when configuring the HP Service Manager server to communicate with the server hosting Operations Management. This password goes with the server name (`Service_Manager_1`) you configured in step 3.

b. Click **Finish**. The target HP Service Manager server appears in the list of Connected Servers.

Configure an Event Forwarding Rule

The next step is to configure an event forwarding rule that determines which events are forwarded automatically to HP Service Manager.

Refer to the Operations Management online help for full details about configuring filters.

To configure a forwarding rule, carry out the following steps:

1. Navigate to the Forwarding Rules manager in the Operations Management user interface:

   Admin > Operations Management > Event Automation > Event Forwarding

2. Click the New (✳) button to open the Create New Forwarding Rule dialog.

3. In the **Display Name** field, enter a name for the forwarding rule, in this example `Forward Critical (Sync and Transfer Control)`.

   Optional. Enter a description for the forwarding rule you are creating.

   Make sure the **Active** checkbox is checked. A rule must be active in order for its status to be available in HP Service Manager.

4. Click the browse button next to the Event Filter field. The Select an Event Filter dialog opens.

   In the Select an Event Filter dialog box, do one of the following:

   ▪ Select an existing filter

   ▪ Create a new filter as follows:

      i. Click the New (✳) button to open the Filter Configuration dialog.

      ii. In the **Filter Display Name** field, enter a name for the new filter, in this example, **FilterCritical**.

         Uncheck the checkboxes for all severity levels except for the severity Critical.

         Click **OK**.

      iii. You should see your new filter in the Select an Event Filter dialog (select it, if it is not already highlighted).

         Click **OK**.

5. Under **Target Servers**, select the target connected server you configured in the previous section "Configure the HP Service Manager Server as a Connected Server" on page 388. In this example, this is `Service Manager 1`.

   Click the Add (➕) button next to the target servers selection field. You can now see the connected server's details. In the **Forwarding Type** field, select the forwarding type.

   For an explanation of the forwarding types, see the section entitled "Forwarding Events and Event Changes to an External Event Process" on page 330.

   Click **OK**.

Configure URL Launch of Event Browser from HP Service Manager

Before operators are able to perform event drill-down from HP Service Manager into the Operations Management user interface using a URL launch of the Event Browser, the operators must be set up as valid users in BSM with appropriate permissions in Operations Manager i:

- **User account requirements**

  If Single Sign-On (SSO) authentication is configured, set up each user in BSM with the *same* user name that is used by the HP Service Manager operator to log onto HP Service Manager and to perform the URL call. (The password of each BSM user can be empty or any string.) After successfully logging into HP Service Manager, the BSM users can launch the Operations Management Event Browser without further authentication.

  If HP Service Manager is not configured to use SSO authentication, set up each user with the *same* user name that is used by the HP Service Manager operator and specify a valid password. The users are required to enter their user name and password when launching the Operations Management Event Browser.

- **Required user permissions**

  You must grant the permission `Events assigned to user` including the required actions to each BSM user. You can optionally grant the permission to view events not assigned to each user.

  Note: Without valid user names, or if a user does not have the required viewing permissions, any attempt to perform a URL launch of the Operations Management Event Browser from HP Service Manager results in an empty browser window.

See also "Performing a URL Launch of the Event Browser from an External Application" on page 334.

Configure URL Launch of HP Service Manager from the Event Browser

To be able to perform a URL launch of HP Service Manager from the Operations Management Event Browser using the web tier client, do the following:

1. Navigate to the Groovy script `ServiceManagerAdapter.groovy` in following location:

   `<HPBSM root directory>/conf/opr/integration/sm/ServiceManagerAdapter.groovy`

2. Open the `ServiceManagerAdapter.groovy` Groovy script.

3. Locate the following text in the Groovy script:

   `private static final String SM_WEB_TIER_NAME = 'webtier-9.30'`

4. Change the value of `webtier-9.30` to the value required to access the HP Service Manager web tier client.

   The full drill-down URL is made up like this:

   `http://<FQDNS of HP Service Manager web tier server>/<web path to HP Service Manager>/<URL query parameters>`

   where `<FQDNS of HP Service Manager web tier server>` is the fully qualified DNS name of the HP Service Manager server where the web tier client is installed. This part of the URL is added automatically (together with `http://`) according to the values that you

provided when you configured the HP Service Manager as a target connected server in the Connected Servers manager (refer to "Configure the HP Service Manager Server as a Connected Server" on page 388).

Here is an example of how the drill-down URL looks:

```
http://smserver.example.com/SM930/index.do?ctx=docEngine&file=
probsummary&query=number%3D
```

So in this example, the you must replace `webtier-9.30` with `SM930`. All the other parts of the URL are configured automatically.

5. In the HP Service Manager web tier configuration file `web.xml`, set the value of the `querySecurity` parameter from the default value (`true`) to **`false`**.

   For more details, see the section Web parameter: querySecurity in the HP Service Manager online help.

Configure HP Service Manager Server

The next step is to configure HP Service Manager server to integrate with Operations Management.

To configure the HP Service Manager server, complete the following steps in the HP Service Manager:

1. From the left hand pane of the HP Service Manager user interface, navigate to:

   Tailoring > Integration Manager

2. Click **Add** to add a new configuration.

3. Select the **SMOMi** integration template from the Integration Template field. Click **Next**.

4. *Optional*. Change the log level to the desired value.

   Optional. Change the description, for example, to `This is for SMOMi integration.`

   Click **Next**.

5. In the General Parameters tab, replace the existing entries with the following values:

Name	Value	Category
`omi.server.url`	`http://`<*BSM_gateway_FQDN*>`/` `opr-gateway/rest/9.10/` `synchronization/event/`	General
`username`	`Service_Manager_1` (This is the name of the HP Service Manager target server you configured previously in the section "Configure the HP Service Manager Server as a Connected Server" on page 388).	Header

Name	Value	Category

`omi.eventdetail.baseurl`	`http://`*`<BSM_gateway_FQDN>`*`/`
	`opr-console/opr-evt-`
	`details.jsp?`
	`eventId=`
		General

6. In the Secure Parameters tab, set the password to the one you specified in the Incoming Connection dialog when configuring the target connected server in the section "Configure the HP Service Manager Server as a Connected Server" on page 388. In our example, this is `HPqwer1_`.

   Click **Next**.

7. In the Integration Instance Fields dialog, click **Next**.

8. In the Integration Instance Mapping dialog, click **Finish**.

> **Note:** Ensure that the rule is active. To make the rule active, select the rule and click **Enable**.

Mapping and Customization

You can add your own custom attributes in a Groovy script and then map these custom attributes to HP Service Manager to the appropriate field in HP Service Manager. You can also change how attributes are mapped from Operations Management to HP Service Manager. The mapping is done in the BDM Mapping Manager in HP Service Manager:

System Administration > Ongoing Maintenance > BDM Mapping Management

For full details about mapping attributes, see the HP Service Manager online help.

Testing the Connection

To test the connection, send an event to the server hosting Operations Management that matches the filter you defined (in our example filter, the severity value is `Critical`), and then verify that the event is forwarded to HP Service Manager as expected.

To test the connection, do the following:

1. On the system running Operations Management, open an Event Browser.

2. On the system running Operations Management, open a command prompt and change to the following directory:

   `<HPBSM root directory>``\opr\support`

3. Send an event using the following command:

   `sendevent -s critical -t test111-1`

4. Verify that the event appears in the Operations Management Event Browser.

5. Select the **Forwarding** tab.

6. In the External Id field, you should see a valid HP Service Manager incident ID.

7. Next, verify that the incident appears in the Incident Details in HP Service Manager:

    If the event drill-down connection is configured correctly, click the **Edit** button. A browser window opens, which takes you directly to the incident in the Incident Details in HP Service Manager.

    If the event drill-down connection is not configured, do the following:

    a. In the Forwarding tab in the Operations Management Event Browser, copy or note the incident ID from the External Id field.

    b. In the HP Service Manager user interface, navigate to:

        Incident Management→Search Incidents

    c. Paste or enter the incident ID in the Incident Id field.

    d. Click the **Search** button. This takes you to the incident in the Incident Details.

8. Close the incident in HP Service Manager.

9. Verify that the change in the state of the incident (it is now `closed`) is synchronized back to Operations Management. You should not be able to see the event that was closed in HP Service Manager in the active Event Browser, but it should now be in the History Browser.

Synchronizing Attributes

Not all attributes are synchronized back from HP Service Manager to Operations Management by default. There are some attributes that are subject to a one-time, uni-directional update from Operations Management to HP Service Manager, and there are some that are subject to bi-directional synchronization.

Uni-directional Synchronization: Operations Management to HP Service Manager

The following attributes are transferred to HP Service Manager from Operations Management one a one-time basis, that is, when the event was initially created, and the transfer of control of the event was configured in the Connected Servers manager.

- Title

- Severity

- Priority

- Operator: the operator assigned to the event who forwarded the event

- Category

- Subcategory

- Related CI

For these attributes, there is no back synchronization from HP Service Manager to Operations Management.

Bi-directional Synchronization

Attributes that support bi-directional synchronization between Operations Management and HP Service Manager are:

- Description

- Lifecycle state (the state is only updated when the state changes to closed)

- Solution

- Operations Management event annotations are synchronized to HP Service Manager activity log

- Contents under the Forwarding tab in the Event Details

Attribute Synchronization using Groovy Scripts

If you want to change the out-of-the-box behavior regarding which attributes are updated, you can specify this in a Groovy script. In the Groovy script, you would specify which fields are updated in HP Service Manager, and which fields are updated in Operations Management. You can also specify custom attributes in the Groovy script.

Tips for Customizing Groovy Scripts

This section provides some tips about customizing Groovy scripts. Below we show just a few selected examples of what you can customize. You can look at the configuration section of a Groovy script to see further items that can be modified.

In the configuration section of a Groovy script, you can define and modify the attributes that are to be synchronized between Operations Management and HP Service Manager. The configuration section of a Groovy script also contains the default value mappings for lifecycle state, severity, and priority. You can also modify these, and it is possible to define the mappings for in-going and out-going requests differently.

More advanced configuration can be done in other parts of the Groovy script if required.

The beginning and the end of the configuration section of a Groovy script is marked as follows:

```
//

// configuration section to customize the Groovy script

// BEGIN

...

...

//

// configuration section to customize the Groovy script

// END
```

Before you modify a Groovy script, make a copy of the original (out-of-the-box) script. This is because a patch, service pack or hotfix may be delivered with a new version of the Groovy script that may overwrite the original script. Make sure that you copy your customized Groovy script to a

safe location. It may be necessary to merge your changes with the new Groovy script delivered with the patch, service pack or hotfix.

The mapping from Operations Management to HP Service Manager is compliant to BDM 1.1 incident web service specifications. The mapping of the BDM 1.1 incident web service to HP Service Manager is specified in HP Service Manager in the BDM Mapping Manager. For more information about the BDM Mapping Manager, see the BDM Mapping Manager section of the HP Service Manager online help.

Controlling Attribute Synchronization

You can control how updates to certain attributes are synchronized between Operations Management and HP Service Manager by setting some Boolean variables to true or false.

Here are two examples:

- `private static final SyncTitleToSMOnUpdate = false;`

  This line of the Groovy script disables the synchronization of changes to the title made in Operations Management to HP Service Manager.

- `private static final Boolean SyncTitleToOPROnUpdate = false;`

  This line of the Groovy script disables the synchronization changes to the title made in HP Service Manager to Operations Management.

The title is a required attribute in HP Service Manager, and is set, independently of the flags above, using the title given in Operations Management during the creation of the incident.

Mapping OPR Lifecycle States to BDM Lifecycle States

You can map the Operations Management (OPR) lifecycle state to a (BDM) lifecycle state in HP Service Manager by modifying the Groovy script.

Here are two examples:

- `private static final Map OPR2BDMLifecycleState = ["open": null, "in_ progress": null, "resolved": null, "closed": "closed"];`

  In this example, only the OPR lifecycle state "closed" is mapped to the BDM lifecycle state "closed". A `null` is an instruction not to change the state in HP Service Manager.

- `private static final Map OPR2BDMLifecycleState = ["open": null, "in_ progress": null, "resolved":"resolved", "closed": "closed"];`

  In this example, the OPR state `resolved` sets the BDM state to `resolved`.

  It is not required to map OPR lifecycle state `open` to SM lifecycle state `open`, because the initial lifecycle of an incident is set during the creation of the incident.

Mapping BDM Lifecycle States to OPR Lifecycle States

You can specify the mapping from BDM lifecycle states to the known OPR states with the following configuration line:

```
private static final Map BDM2OPRLifecycleState = ["open": null, "work-
in-progress": null, "resolved": null, "closed": "closed"];
```

In this example the (OPR) event is closed when the (BDM) incident is closed. A `null` is an instruction not to change the event state if the incident state was changed in HP Service Manager.

If you map the BDM lifecycle state `open` to OPR lifecycle state `open`, the following would happen. If an incident is closed and then is reopened again in HP Service Manager, corresponding event in Operations Management would be reopened.

Syntax Errors

If you get a syntax error when customizing your Groovy scripts, look at the log file `opr-event-sync-adapter.log` for information about how to resolve the error. You can find the log file here:

`<HPBSM root directory>/log/opr-event-sync-adapter.log`

Customizing the Service Manager 9.2 Integration

The ServiceManagerAdapter groovy script is provided for Event Forwarding to Service Manager. You can customize this script for your installations.

To customize the ServiceManagerAdapter groovy script, open the Scripts manager (⬛), select the **sm:ServiceManagerAdapter** script and open it for editing (✏). The Edit Script window opens. The contents of the script are displayed in the **Script** tab.

> **Tip:** Copying the script text into a text editor of your choice. When you have finished editing, copy the edited text back to the Edit Script window and save the script.

Near the beginning of the script, there are two sections used to modify the default behavior of the BSM event synchronization with Service Manager.

To access

Select **Admin > Operations Management > Setup > Connected Servers >** ⬛ button

Configuring the ServiceManagerAdapter Script

This section controls which event and incident properties are synchronized to and from Service Manager, and is contained within the following comments:

- `BEGIN Configuration: Customization of properties for synchronization`
- `END Configuration: Customization of properties for synchronization`

The ServiceManagerAdapter script can be configured section contains constants, 6 "maps" and "8" sets to allow configuring the synchronization of the properties. Each is described below:

Service Manager Drilldown Constant

The first variable that can be adjusted is `SM_WEB_TIER_NAME`. Set this value to the base name of the web application deployed in the Tomcat container on the Service Manager system. This web application is used for drilling down into Service Manager. The name is used in the URL path for the drilldown. It must match the base name of the web application (".war" is removed). The default is shown below:

`private static final String SM_WEB_TIER_NAME = 'webtier-9.30'`

BSM Administrator User

The `BSM_ADMINISTRATOR_LOGIN_NAME` variable is used to contain the name of the BSM Administrator user. By default this is set to `admin`.

For Operations Management events that are forwarded automatically by a forwarding rule, the `_is_recorded_by` attribute is set to the user specified in the `BSM_ADMINISTRATOR_LOGIN_NAME` variable.

For Operations Management events that are forwarded manually, the `_recorded_by` attribute is set to the user that initiated the forward request.

```
private static final String BSM_ADMINISTRATOR_LOGIN_NAME = 'admin'
```

Enumeration Value Maps

Maps are defined to map enumerated values of Operations Management event properties to values on Service Manager incident properties. These maps, in general, should not be customized, but they provide a list of possible values that can be specified in the sets described below. For details on each map, view the actual values defined in the script.

- **MapOPR2SMStatus**: Maps the Operations Management event `state` to the Service Manager incident `status`

- **MapSM2OPRState**: Maps the Service Manager incident `status` to the Operations Management event `state`

- **MapOPR2SMUrgency**: Maps the Operations Management event `severity` to the Service Manager incident `urgency`

- **MapSM2OPRSeverity**: Maps the Service Manager incident `urgency` to the Operations Management event `severity`

- **MapOPR2SMPriority**: Maps the Operations Management event `priority` to the Service Manager incident `priority`

- **MapSM2OPRPriority**: Maps the Service Manager incident `priority` to the Operations Management event `priority`

Custom Property Maps

The following maps allow a user to map any top-level Operations Management event property to any top-level Service Manager incident property.

- **MapOPR2SMCustomAttribute**: Maps the specified Operations Management custom attributes to a Service Manager incident property for synchronization.

  Add a CA name to the map along with a Service Manager incident property name (XML tag name).

  Target Service Manager Incident property name of "activity_log" will append the CA change to the Service Manager incident activity log.

  > **Note:** Only top-level Service Manager incident properties are supported in this map.

- **MapSM2OPRCustomAttribute**: Maps the specified Service Manager incident properties to an Operations Management event custom attribute for synchronization.

Add a Service Manager incident property name to the map along with an Operations Management event custom attribute name.

> **Note:** Only top-level Service Manager incident properties are supported in this map.

Examples:

The following synchronizes the Operations Management event custom attribute `MyCustomCA` to the Service Manager incident `activity_log` and the custom attribute `MyCustomCA1` to the Service Manager incident property `SMCustomAttribute`.

```
private static final Map<String, String> MapOPR2SMCustomAttribute =
["MyCustomCA" : "activity_log", "MyCustomCA1" : "SMCustomAttribute"]
```

The following synchronizes the Service Manager incident property `incident_status` to the Operations Management custom attribute `SMIncidentStatus`.

```
private static final Map<String, String> MapSM2OPRCustomAttribute =
["incident_status" : "SMIncidentStatus"]
```

Synchronization Change Sets

The following sets define which properties and enumerated values are synchronized whenever a change occurs in the Operations Management event or Service Manager incident. The properties synchronized upon a change as standard are marked in **bold**. For each list, the value of "*" can be specified. In this case, all possible properties or enumerated values are synchronized for the specified list.

> **Note:** When the Service Manager incident is created, all possible Operations Management event properties and enumerated values are set in the Service Manager incident. The following sets are mainly used for synchronization of changes.

SyncOPRPropertiesToSM

Operations Management event properties to synchronize to a corresponding Service Manager Incident property on change:

- title
- **description**
- **state**
- severity
- priority
- **solution**
- assigned_user
- assigned_group

SyncOPRPropertiesToSMActivityLog

Operations Management event properties to synchronize to a corresponding Service Manager Incident activity log on change:

- **title**

- description

- **state**

- **severity**

- **priority**

- solution

- **annotation**

- **duplicate_count**

- **custom_attribute**

- **cause**

- **symptom**

- control_transferred_to

- **assigned_user**

- **assigned_group**

SyncSMPropertiesToOPR

Service Manager Incident properties to synchronize to a corresponding Operations Management Event property on change:

- name

- **description**

- **incident_status**

- urgency

- priority

- **solution**

SyncOPRStatesToSM

Operations Management event states to synchronize to the Service Manager incident status on change:

> **Note:** state must be included in SyncOPRPropertiesToSM or this list is ignored.

- open

- in_progress

- in_progress

- resolved

- **closed**

SyncOPRSeveritiesToSM

Operations Management event severities to synchronize to the Service Manager incident urgency on change:

> **Note:** `severity` must be included in `SyncOPRPropertiesToSM` or this list is ignored.

- **critical**

- **major**

- **minor**

- **warning**

- **normal**

- **unknown**

SyncSMStatusToOPR

Service Manager incident status to synchronize to the Operations Management event states on change:

> **Note:** `status` must be included in `SyncSMPropertiesToOPR` or this list is ignored.

- accepted

- assigned

- open

- reopened

- pending-change

- pending-customer

- pending-other

- pending-vendor

- referred

- suspended

- work-in-progress

- rejected

- replaced-problem

- resolved

- cancelled

- **closed**

SyncSMUrgenciesToOPR

Service Manager incident urgencies to synchronize to the Operations Management event severities on change:

> **Note:** `urgency` must be included in `SyncSMPropertiesToOPR` or this list is ignored.

Acceptable values are **1-4**.

SyncSMPrioritiesToOPR

Service Manager incident priorities to synchronize to the Operations Management event priorities on change:

> **Note:** `priority` must be included in `SyncSMPropertiesToOPR` or this list is ignored.

Acceptable values are **1-4**.

Examples:

The following example synchronizes the Operations Management title, state and description to the Service Manager incident whenever the corresponding property is changed in the Operations Management event.

```
private static final Set SyncOPRPropertiesToSM = ["title", "state",
"description"]
```

The following example will synchronize the Operations Management states resolved and closed to the Service Manager incident whenever the corresponding property is changed in the Operations Management event.

```
private static final Set SyncOPRStatesToSM = ["resolved", "closed"]
```

> **Note:** The properties that are synchronized to the Service Manager activity log are concatenated together for each change and then appended to the Service Manager incident activity log.

Localization

This section is provided to enable localizing some of the text that is displayed in the:

- Operations Manager Event Browser Forwarding tab (not available for an Event Channel deployment)
- HP Service Manager incident activity log

This sections is contained within the following comments:

- `BEGIN Localization: Customization of text values for language localization`
- `END Localization: Customization of text values for language localization`

The following sections describe the text that can be localized.

Forwarding tab

The Service Manager incident properties urgency and priority are of type integer. In order to display a more meaningful value in the Forwarding tab, maps are provided to display a string. These strings may be localized for display in the browser.

- **Service Manager Urgency values**

  The text value will be displayed in the Forwarding tab.

  > **Note:** This text may be localized for the desired locale.

  ```
  privatestaticfinal Map SMUrgency = ["1": "1 - Critical", "2": "2 -
  High", "3": "3 - Average", "4": "4 - Low"]
  ```

- **Service Manager Priority values**

  The text value will be displayed in the Forwarding tab.

  > **Note:** This text may be localized for the desired locale.

  ```
  privatestaticfinal Map SMPriority = ["1": "1 - Critical", "2": "2 -
  High", "3": "3 - Average", "4": "4 - Low"]
  ```

Service Manager Incident Activity Log

Synchonization from BSM to Service Manager appends various text to the Service Manager incident activity log. This text may be localized as follows.

- General Locale Setting: Used mainly in formatting of dates. May be changed, for example to Locale.JAPAN. See Java *Locale* class documentation for all possible values.

  ```
  private static final Locale LOCALE = Locale.getDefault()
  ```

- Annotation date format: See Java *SimpleDateFormat* class documentation for details on the syntax. Script default is below.

  ```
  private static final String ANNOTATION_DATE_FORMAT = "yyyy.MM.dd
  HH:mm:ss z"
  ```

- Description: In Service Manager the incident description is a required attribute. In case it is not set in BSM this value is taken. An empty string is NOT allowed.

  ```
  private static final String EMPTY_DESCRIPTION_OVERRIDE = "<none>"
  ```

- Log Text: The following text is prefixed to the appropriate Operations Management event property when synchronizing it to an Service Manager Incident activity log.

  NOTE: This text may be localized for the desired locale. Defaults are as shown below.

  ```
  private static final String ACTIVITY_LOG_TITLE = "[Title]\n"
  ```

  ```
  private static final String ACTIVITY_LOG_TITLE_CHANGE = "Event title
  changed to: "
  ```

  ```
  private static final String ACTIVITY_LOG_STATE = "[State]\n"
  ```

```
private static final String ACTIVITY_LOG_STATE_CHANGE = "Event state
changed to: "

private static final String ACTIVITY_LOG_DESCRIPTION = "
[Description]\n"

private static final String ACTIVITY_LOG_DESCRIPTION_CHANGE = "Event
description changed to: "

private static final String ACTIVITY_LOG_SOLUTION = "[Solution]\n"

private static final String ACTIVITY_LOG_SOLUTION_CHANGE = "Event
solution changed to: "

private static final String ACTIVITY_LOG_ASSIGNED_USER = "[Assigned
User]\n"

private static final String ACTIVITY_LOG_ASSIGNED_USER_CHANGE =
"Event assigned user changed to: "

private static final String ACTIVITY_LOG_ASSIGNED_GROUP = "[Assigned
Group]\n"

private static final String ACTIVITY_LOG_ASSIGNED_GROUP_CHANGE =
"Event assigned group changed to: "

private static final String ACTIVITY_LOG_SEVERITY = "[Severity]\n"

private static final String ACTIVITY_LOG_SEVERITY_CHANGE = "Event
severity changed to: "

private static final String ACTIVITY_LOG_PRIORITY = "[Priority]\n"

private static final String ACTIVITY_LOG_PRIORITY_CHANGE = "Event
priority changed to: "

private static final String ACTIVITY_LOG_CONTROL_TRANSFERRED_TO = "
[Control Transferred To]\n"

private static final String ACTIVITY_LOG_CONTROL_TRANSFERRED_TO_
CHANGED = "Event control transfer state changed to: "

private static final String ACTIVITY_LOG_ANNOTATION = "[Annotation]
\n"

private static final String ACTIVITY_LOG_CA = "[Custom Attribute]\n"

private static final String ACTIVITY_LOG_CAUSE = "[Cause] "

private static final String ACTIVITY_LOG_OMI_CAUSE = "[OMi Cause] "

private static final String ACTIVITY_LOG_OMI_SYMPTOM = "[OMi
Symptom] "

private static final String ACTIVITY_LOG_DUPLICATE_COUNT = "
[Duplicate Count] "

private static final String ACTIVITY_LOG_PREVIOUS = "previous "

private static final String ACTIVITY_LOG_CURRENT = "current "
```

Operations Management Event to BDM Incident Property Mapping Table

The standard integration synchronizes the following Operations Manager Event properties to Service Manager Incident properties:

Operations Management UI (Event)	Operations Management UI (Forwarding)	Operations Management Web Service (Event)	Service Manager Web Service (Incident)	Service Manager Object	Service Manager UI	Comments
-	-	-	global_id	id	-	There is no mapping and no synchronization for `global_id` between an Operations Management event and a Service Manager incident.
ID	-	id	external_process_reference	external.process.reference	-	Set only on Service Manager incident creation.
-	External ID	control_transferred_to external_id	reference_number	number	Incident ID	Set only on Service Manager incident creation.

Operations Management UI (Event)	Operations Management UI (Forwarding)	Operations Management Web Service (Event)	Service Manager Web Service (Incident)	Service Manager Object	Service Manager UI	Comments
Title	-	title	name	brief.d-escription	Title	Set on incident creation. Standard configuration: only Operations Management title changes are synchronized to the Service Manager incident activity log. If Operations Management event title exceeds 256 characters, or contains a newline character, it is truncated and the entire title is prepended to the description.

Operations Management UI (Event)	Operations Management UI (Forwarding)	Operations Management Web Service (Event)	Service Manager Web Service (Incident)	Service Manager Object	Service Manager UI	Comments
Description	-	description	description	action	Description	Can be a combination of the Operations Management event title and description. See title comments for details. Standard configuration: synchronized in both directions.
Solution	-	solution	solution	Resolution	Solution	Standard configuration: synchronized in both directions.
Lifecycle State	-	state	incident_status	problem.status	Status	Set on incident creation. Standard configuration: only 'closed' state is synchronized.
Severity	Severity	severity	urgency	severity	Urgency	Set on incident creation. Standard configuration: all value changes are synchronized.

Operations Management UI (Event)	Operations Management UI (Forwarding)	Operations Management Web Service (Event)	Service Manager Web Service (Incident)	Service Manager Object	Service Manager UI	Comments
Priority	Priority	priority	priority	priority.code	Priority	Set on incident creation. Standard configuration: all value changes are synchronized.
Assigned User	-	assigned_user login_name	is_requested_by party display_label	contact.name	Contact	By default not synchronized.
-	Assigned User	-	has_assigned party display_label	assignee.name	Assignee	Operations Management queries the Service Manager Incident Web Service in real time for display in Forwarding Tab only. No synchronization between the Operations Management event and the Service Manager incident.

Operations Management UI (Event)	Operations Management UI (Forwarding)	Operations Management Web Service (Event)	Service Manager Web Service (Incident)	Service Manager Object	Service Manager UI	Comments
-	-	control_transferred_to initiated_by	is_recorded_by party display_label	opened.by	Opened By	By default not synchronized.
Assigned Group	Assigned Group	assigned_group name	has_assigned_group functional_group display_label	assignment	Assignment Group	Operations Management queries the Service Manager Incident Web Service in real time for display in Forwarding Tab only. No synchronization between the Operations Management event and the Service Manager incident.
Category	-	category	category	sub_category	Area	Category is set by Operations Management only on Service Manager incident creation, but is by default ignored in Service Manager .

Operations Management UI (Event)	Operations Management UI (Forwarding)	Operations Management Web Service (Event)	Service Manager Web Service (Incident)	Service Manager Object	Service Manager UI	Comments
Sub-category	-	sub_category	sub_category	product_type	Subarea	Subcategory is set by Operations Management only on Service Manager incident creation, but is by default ignored in Service Manager .
Related CI	-	related_ci	is_registered_for	logical.name	Affected CI	Affected CI is set by Operations Management only on Service Manager incident creation.
Annotations	-	annotation_list	activity_log description	update.action	Activity Log	Synchronized from Operations Management to Service Manager only. OOTB synchronization enabled.

Operations Management UI (Event)	Operations Management UI (Forwarding)	Operations Management Web Service (Event)	Service Manager Web Service (Incident)	Service Manager Object	Service Manager UI	Comments
Custom Attributes	-	custom_attribute_list	activity_log description	update.action	Activity Log	Can be configured for synchronization to a specific Service Manager incident property, or the activity log. For Service Manager activity log, synchronization is from Operations Management to Service Manager only. Standard configuration: synchronization disabled.

Oper-ations Man-agement UI (Event)	Oper-ations Man-agement UI (For-warding)	Oper-ations Man-agement Web Service (Event)	Service Manager Web Service (Inci-dent)	Service Manager Object	Service Manager UI	Comments
Cause	-	cause-> control_ trans-ferred_to external_id	is_ caused_ by -master_ ref-erence_ number	Links the two SM Incidents	Related Records	If the cause event has been synchronized to Service Manager, the two Service Manager Incidents are related, otherwise information about the Operations Management cause is appended to the Service Manager incident activity log. Standard configuration: syn-chronization enabled.
Symptoms	-	symptom_ list	activity_ log descrip-tion	update.action	Activity Log	Syn-chronization is from Operations Management to Service Manager only. Standard configuration: syn-chronization enabled.

Operations Management UI (Event)	Operations Management UI (Forwarding)	Operations Management Web Service (Event)	Service Manager Web Service (Incident)	Service Manager Object	Service Manager UI	Comments
Duplicate Count	-	duplicate_count	activity_log description	update.action	Activity Log	Synchronization is from Operations Management to Service Manager only. Standard configuration: synchronization enabled.

Error Handling

The Operations Management event synchronization interface that forwards events and subsequent changes uses the error handling algorithms described in this section to determine if a forward request should be retried or discarded.

Error handling is different depending on the type of integration:

- "Groovy Script Integration" below
- "Web Service Integration" on the next page
- "HP Service Manager Integration" on page 417

Groovy Script Integration

If you implement a Groovy script to integrate your application, error handling is as follows.

For `forwardEvent()` and `forwardChange()`:

- Return of true: The request is marked as FORWARDED and the forward request is removed from the queue.
- Return of false: The request will be put back in the queue. Retries are done once a minute until the request succeeds or the request is in the queue longer than the Event Forwarding Expiration Time (default is 12 hours and may be changed in the Infrastructure Settings). New events and updates are forwarded in the order the forward request was received. Should a request fail due to a Non-RuntimeException error it will block all other requests for this server. All new events in the queue are delivered before delivering any outstanding updates.

For `forwardEvents()`:

- Return of true: All forward event requests are marked as FORWARDED and the forward requests are removed from the queue. If all events have had the forward status set to FORWARDED, the exception is logged.
- Return of false: If there are events that are not marked as FORWARDED, then the first event without a result is handled in the normal exception processing. The remaining events are left queued and retried later.

  See also standard error handling for `forwardEvent()` defined in the `com.hp.opr.api.ws.adapter.ExternalProcessAdapter` interface.

For `forwardChanges()`:

- Return of true: All forward change requests are marked as FORWARDED and the forward requests are removed from the queue. If all change requests have had the forward status set to FORWARDED, the exception is logged.
- Return of false: If there are changes that are not marked as FORWARDED, then the first change without a result is handled in the normal exception processing. The remaining changes are left queued and retried later.

  See also standard error handling for `forwardEvent()` defined in the `com.hp.opr.api.ws.adapter.ExternalProcessAdapter` interface.

For `receiveChanges()`:

- Error handling is the same as for `receiveChange()` defined in the `com.hp.opr.api.ws.adapter.ExternalProcessAdapter` interface.

If the Groovy script throws an exception, it is handled as follows:

- `org.apache.wink.client.ClientWebException`: The exception is queried for the HTTP status code returned by the server. If an HTTP status code is found the status code is handled just as in the Web Service Integration case described in "Web Service Integration" below. If the status code does not exist, the exception is handled just as any other exception, see below for more information.

- Any exception other than `ClientWebException`: The exception is recursively searched for the root cause exception and then interpreted as follows:

  - RuntimeException: Error is logged to the `opr-event-sync-adapter.log` log file and the request is marked as FAILED. There is no retry for this request.

  - Non-RuntimeException: Examples would be IOException, SocketException, and so on. Error is logged to the `opr-event-sync-adapter.log` log file. In this case it is expected that the connection to the server will recover at some future time and be able to send the request. The request will be put back in the queue. Retries are done once a minute until the request succeeds or the request is in the queue longer than the Event Forwarding Expiration Time (default is 12 hours and may be changed in the Infrastructure Settings). New events and updates are forwarded in the order the forward request was received. Should a request fail due to a Non-RuntimeException error it will block all other requests for this server. All new events in the queue are delivered before delivering any outstanding updates.

Web Service Integration

If you implement an Event Synchronization Web Service endpoint that can be directly called by Operations Management, error handling is as follows.

- **HTTP status of 2xx returned.** Depending upon the particular call, for example, a 200 (OK) or 201 (Created) are both acceptable for POST of a new event, 200 (OK) is expected for a PUT of an event update. HTTP status 202 (accepted), is also accepted for PUT or POST.

  - The object returned must be an OPR Event with the ID set to the external event ID for POST.

  - The object returned for PUT is ignored.

- **HTTP status of 3xx returned.** These are redirection status. Redirection is not supported by the service. They are treated the same as 4xx, therefore the request will be marked as FAILED and no further retries are done.

- **HTTP status of 4xx returned.** Any status of 4xx is considered an error in the client request, therefore the request will be marked as FAILED and no further retries are done.

- **HTTP status of 5xx returned.** Any status of 5xx is considered an error on the server. In this case it is expected that the server will recover at some future time and be able to accept the request. The request will be put back in the queue. Retries are done once a minute until the request succeeds or the request is in the queue longer than the Event Forwarding Expiration Time (default is 12 hours and may be changed in the Infrastructure Settings). New events and updates are forwarded in the order the forward request was received. Should a request fail due to a 5xx error it will block all other requests for this server. All new events in the queue are delivered

before delivering any outstanding updates.

- Any type of IOException encountered while trying to communicate with the server will result in the request being re-queued and then retried as described for HTTP status of 5xx.

HP Service Manager Integration

The `ServiceManagerAdapter` Groovy script uses the Apache Wink client to communicate with HP Service Manager. It therefore will throw a `ClientWebException` when there is an HTTP error status returned by HP Service Manager. See "Groovy Script Integration" on page 415 for an indication on how this type of error will be handled for HP Service Manager integrations.

Part 5

Reporting in BSM

Chapter 14

Generic Reporting Engine API

The recommended method for creating API-level queries to the profile database is building queries using the Custom Query Builder. The Custom Query Builder enables the building of queries using a graphical user interface, and facilitates the generation of reports, extraction of data in different formats, and generation of query URLs that can be used with third-party or custom tools. For details, see Building a Custom Query Using Custom Query Builder in the BSM User Guide.

The Generic Reporting Engine API also enables manual creation of queries using the following methods:

- **Web browser.** The request is sent as an HTML query and the data is returned as HTML or as a CSV (Comma Separated Values) file that can be opened with Microsoft Excel or processed with a custom tool.

- **Web Service.** The return object contains the data in CSV format.

The remainder of this chapter describes how to create queries manually.

Prerequisite Knowledge

Users of the API should be familiar with SQL syntax and BSM administration and applications. Users of the API through the Web Service should also be familiar with the SOAP specification and an object-oriented programming language such as C++ or Java.

Permissions

For a query to access the data using the API query syntax described below, the user and password parameters passed in the query must be those of a user with either System Viewer or Superuser permissions. (For details on setting permissions in the Permissions Manager, see Permissions Overview in the BSM Platform Administration Guide.)

Configuration

To configure the API options, select **Admin > Platform > Setup and Maintenance > Infrastructure Settings**:

- Select **Foundations.**

- Select **Generic Data Engine Open API**.

- In the Generic Data Engine Open API - Generic Data Engine Open API Settings table, locate:

  - **Maximum Rows.** Change the maximum number of data rows returned.

  - **Enable Open API.** Enable or disable the use of the Generic Reporting Engine API, as required.

Getting Metadata on the Samples

When building queries, you must know the data representation of the sample. For information on commonly queried samples and descriptions of their fields, see Data Samples in the BSM

Application Administration Guide.

Advanced Sample Retrieval

Users with special reporting needs can retrieve a list of all samples and their fields using the MBean Inspector. Access the MBean Inspector page by entering the following URL in your browser:

http://<server>[:port]/jmx-console/HtmlAdaptor?action=inspectMBean&name=Topaz%3Aservice%3DMeta-Data+Manager

The default port number is 8080. If this port is incorrect, consult your system administrator for the correct port number.

Enter your JMX console authentication credentials. If you do not know your authentication credentials, contact your system administrator.

On the MBean Inspector page, click the **Invoke** button next to the operation **showMetaDataDBMapping**. The bean returns the list of the fields in each sample.

Data Returned

The same data is returned whether the request is made from a browser or with the Web Service. With a browser, the data resides in the response body, and for the Web Service, the data resides in the return object.

Web Browser Response Body

When the query is submitted from a browser, the response CSV or HTML contains either the data, or an error code and message. If the number of rows to be returned exceeds the maximum, the last row of the data is `Returned X of Y rows`, where *X* is the number of rows returned and *Y* is the actual number of rows that fulfil the conditions of the query. If there is an error at the engine level, the HTTP success code is returned, but the body of the response is <error code>, <error message>.

Web Service Return Object

The Web Service return object contains the following:

- **retval.** The data or an error message.

- **errorCode.** The error code (type int). Possible error codes are:

  - 0 - Success

  - 100 - Authorization error

  - 101 - Processing error

  - 102 - Open API has been disabled

- **origRowCount.** The actual number of rows the query should have returned (type int). If the number of rows to be returned exceeds the maximum, the **origRowCount** field is set to the actual number of rows that the query would have returned had the maximum not been exceeded.

Querying with a Browser

When querying with a browser, the getData service is called with the URL:

```
http://<server>[:port]
/topaz/gd-
eope-
napi/GdeOpenApi?method=getData&user=<username>&password=<password>&query=<query>
```

The URL can include an optional resultType parameter:

```
http://<server>[:port]
/topaz/gd-
eope-
napi/GdeOpenApi?method=getData&user=<username>&password=<password>&query=<query>&resu
```

The port specification is only necessary for non-standard installations. Consult your system administrator for the correct port number.

The default return type is HTML. If `resultType=csv` is specified, a comma separated values file is returned.

Using the Web Service

The API Web Service enables submitting a query consisting of a username, password, and an SQL-like select statement. The engine returns an error description if it cannot parse the statement or if there is a problem running the query. If there is no error, the results of the query are returned.

The SOAP WSDL is at:

```
http://<server>[:port]/topaz/gdeopenapi/services/GdeWsOpenAPI?wsdl
```

The port specification is only necessary for non-standard installations. Consult your system administrator for the correct port number.

Supported SQL Syntax

The language supported is a subset of SQL and supports these keywords, modifiers, and operators:

- SELECT

- WHERE

- FROM

- TOP

- HAVING

- Aliasing with the AS keyword

- Logical operators OR , AND , NOT

- DISTINCT modifier (only supported for select list items)

- IN operator. Inner selects can be used to return the values for the IN operator.

- BETWEEN operator

- IS NULL (IS NOT NULL is not supported)

- LIKE. The wildcard character is the asterisk (*). Do not use the percent sign (%). The asterisk can not be used by itself (LIKE *). It must be used with other characters.

- Mathematical operators: +, -, *, /, (,)

- Comparators: =, IS, !=, <>, >, >=, <, <=

- ORDER BY and the ASC and DESC modifiers

Supported Functions

The supported functions are as follows:

- MAX

- MIN

- SUM

- COUNT

- AVG

- STDDEV

- SUMOFSQR

- LOG

- CEIL

- FLOOR

- MOD

- SQRT

- REPLACENULL (equivalent to Oracle's NVL and Microsoft SQL Server's ISNULL)

- IF

-

Query Limitations

The following limitations apply to queries submitted to the service:

- Only one monitor type can be selected in a single query.

- The asterisk (*) is not supported as a wildcard character except in combination with the LIKE operator. It is supported as the multiplication operator.

- Inner selects and joins are not supported, with one exception: an inner select can be used to return the values for an IN clause.

- The ORDER BY clause requires a column number, for example ORDER BY 1. ORDER BY column name is not supported.

- The engine requires that queries contain a time limitation (that is, a condition for the time_stamp field) in the WHERE clause.

- The GROUP BY clause is not supported. It is unnecessary because the engine treats all fields that do not have an aggregate function as GROUP BY fields.

- When manually defining a filter that consists of strings containing white space or special characters (for example, `where bb_guid IN (a b, c)`), you must enclose the white space or special character string with quotes (for example, `where bb_guid IN (`a b', c)`). When you create filters on the Filter Builder page, BSM automatically adds the quotes. Special characters are defined as any characters other than digits, letters, and the following characters: "_", "$", "#".

- When defining a filter that consists of strings containing one or more single quote characters, you must add a second single quote character beside each instance. For example, change `szTransactionName = ('Login_to_O'Brien')` to `szTransactionName = ('Login_to_O''Brien')`.

- The columns in the returned data are labeled `Column 0, Column 1`, and so on. To return meaningful column labels, use the SQL AS operator. For example, `Select time_stamp as TimeStamp`. With this use of the AS operator, the column label is `TimeStamp`.

- The "COUNT (DISTINCT <field>)" syntax is not supported. Instead use the "COUNT DISTINCT (<field>)" syntax.

Date-Time Values

Time in queries and return data is specified in seconds since January 1, 1970. You can use Microsoft Excel to convert between time values in seconds and date- time.

Time is most commonly used for time stamp fields.

To get a GMT time for use in a query:

Enter the date and time in a Date-formatted cell and in another cell, formatted as General, enter the formula:

```
=(<date cell> - 25569 ) * 86400
```

To correct for a local time zone:

Add the time zone offset times 3600 seconds to the result. For example, for Central Europe (GMT + 2):

```
=(<date cell> - 25569 ) * 86400 + ( 2 * 3600 )
```

To view a time value from a query as a GMT date in Excel:

Use a Date format for the cell and enter the formula:

```
=<time stamp> / 86400 + 25569
```

To correct for a local time zone:

Subtract the time zone offset times 3600 seconds from the time stamp. For example, for the Eastern United States, standard time (GMT - 3):

```
=(<time stamp> - ( -3 * 3600 ) )/ 86400 + 25569
```

byTime Function

The Generic Reporting Engine SQL supports the function **byTime**, which returns data grouped by time periods. For example, querying the average response time of a transaction for the past day without the byTime function returns one value. You can use the byTime function to view the average response time of the transaction for each hour of the past day. It this case, a value is returned for each hour of the past 24 hours.

The function syntax is:
byTime(<*timefield* >, <*step value*>, <*number of step*>, <*offset*>)

Argument	Description
timefield	Usually a timestamp field
step value	One of: 10 - Second 20 - Minute 30 - Hour 40 - Day 50 - Week 60 - Month 70 - Quarter 80 - Year
number of step	The number of the units specified in *step value* to group.
offset	Time zone offset from GMT in hours. Positive numbers indicate time zones East of GMT. Negative numbers indicate time zones west of GMT.

For example, to return one value for each 3 days, corrected to one hour east of GMT:

byTime(time_stamp, 40, 3, 1)

Query Examples

Below are several examples of query URLs that retrieve different types of data from the database.

Example of ss_t Sample

This example illustrates retrieving the average value for SiteScope samples on a given measurement and monitor:

http:/-
/myServer/topaz/gd-
eopenapi/GdeOpenApi?method=getData&user=admin&password=admin&query=select
szMeasurementName, szMonitorName, avg(dValue) from ss_t where u_iStatus=1 and time_
stamp > 123456 and szMeasurementName = `myMeasurmentName' and szMonitorName =
`myMonitorName'

Example of trans_t Sample

This example illustrates retrieving the average response time, grouped by minutes and offset to
GMT + 3 for Springfield_infra_ems_login transactions in the Springfield_Location application for a
given period from BPM data:

http:/-
/myServer/topaz/gd-
eopenapi/GdeOpenApi?method=getData&user=admin&password=admin&query=select byTime
(time_stamp, 20, 1, 3.0), application_name as ApplicationName, szTransactionName as
TransactionName, AVG(dResponseTime) from trans_t where time_stamp>=1126594800.64 and
time_stamp<1126596000.64 and application_name='Springfield_Location' and
szTransactionName='Springfield_infra_ems_login'

Example of rum_action_t Sample

This example illustrates retrieving the total server time for each URL as measured by RUM:

http:/-
/myServer/topaz/gd-
eopenapi/GdeOpenApi?method=getData&user=admin&password=admin&query=select
application_name as ApplicationName,action_name as ActionName,action_descriptor,AVG(tot_
server_time) from rum_action_t where time_stamp>=1304197200.64 and time_
stamp<1306702800.64 and application_name='EC2%20jpetstore' and action_name='Sign In'

Example of rum_application_stats_t Sample

This example illustrates retrieving the average server time of the application's actions on each of
the servers serving the application as measured by RUM:

```
http:/-
/MyServer/topaz/gd-
eopenapi/GdeOpenApi?method=getData&user=admin&password=admin&query=select
 application_name as ApplicationName,server_host_name as hostName,Avg
(tot_server_time) as serverTime from rum_application_stats_t where
time_stamp>=1304197200.64 and time_stamp<1306702800.64 and
application_name='EC2 jpetstore' group by application_name, server_
host_name
```