

HP Business Service Management

for the Windows® operating system

Software Version: 9.01

Operations Manager *i* Extensibility Guide

Document Release Date: September 2010
Software Release Date: September 2010



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 Notices

© Copyright 2008-2010 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe® is a registered trademark of Adobe Systems Incorporated.

Java™ is a US trademark of Sun Microsystems, Inc.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Open Software Foundation® is a trademark of The Open Group in the U.S. and other countries.

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 (<http://www.apache.org/>).

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

This product includes software developed by the MX4J project (<http://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.

Support

Visit the HP Software Support Online web site at:

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

1	Introduction	15
	Prerequisites	16
	Section I: Content Development	17
2	Integration Content	19
	Topology	19
	Event Type and Health Indicators	19
	Correlation Rules	20
	Additional Event Processing	20
	Tools	20
	View Mappings	21
	Graphs	21
3	Topology	23
	Integrating New Applications	24
	Create New CI Types for the New Application	24
	Set Key Attribute Values for New CI Types	26
	Establish Relationships for the New Application	28
	Creating CIs and CI Relationships in the RTSM	29
	Creating CIs Using RTSM/HP Data Flow Management Discovery	29
	Creating CIs Using the HPOM Service Model and Topology Synchronization	29
	Considerations when Choosing a Discovery Method	30
	When to Use DFM	30
	When to Use Topology Synchronization	30
	When to Create CIs Manually	30
	Enrichment Rules	31
	Topology View of the New Application	31
	Impact Propagation	32
4	Event Type and Health Indicators	33
	Mapping Events to CIs	33
	Setting the Custom Message Attribute RelatedCiHint	34
	RelatedCiHint Values	35
	Creating Event Type and Health Indicators	36
	Analyze Events and Define Indicators	36
	Assigning Event Type Indicators to an Event	39
	Setting ETIs with the CMA “EtiHint”	39
	Setting ETIs with ETI Mapping Rules	40
	Assigning HIs to KPIs	41

5	Correlation Rules and Mapping	43
6	Additional Event Processing	45
7	Tools	47
8	View Mappings	49
9	Graphs	51
	Integrating Performance Data	51
10	Packaging Content	53
	Create RTSM Packages	53
	Save Topology Synchronization Files	54
	Create a Content Pack	54
	Upload the Content	56
	Upload RTSM Packages	56
	Copy Topology Synchronization Files	56
	Upload Content Packs	57
	Section II: Populating the Run-Time Service Model	59
11	Topology Synchronization Overview	61
	Basic Topology Synchronization	62
	Basic Topology Synchronization Architecture	63
	Running Basic Topology Synchronization	64
	Normal Mode	64
	Touch Mode	65
	Skip Services Option	65
	Dynamic Topology Synchronization	66
	Event-driven Synchronization	67
	Scheduled (or Time-driven) Synchronization	67
	Dynamic Topology Synchronization Architecture	68
	Comparing Basic and Dynamic Topology Synchronization	69
	Synchronization Packages and Mapping	70
	Scripting	70
	CI Resolution Using a Mapping Table	71
	Topology Synchronization File Locations	71
	Basic Topology Synchronization	71
	Dynamic Topology Synchronization	72
	Topology Synchronization Settings	72
	Uploading Synchronization Packages to the Database	73
	Uploading HPOM SPI Service Type Definitions to the Database	74

12 Synchronization Packages	75
Synchronization Packages Overview	75
Standard Out-of-the-box Synchronization Packages	76
Additional Out-of-the-box Synchronization Packages	77
Package Descriptor File: package.xml	77
Mapping Files	78
Context Mapping (Filtering): contextmapping.xml	78
Type Mapping: typemapping.xml	78
Attribute Mapping: attributemapping.xml	78
Relation Mapping: relationmapping.xml	78
Configuring Topology Synchronization: ACME Example	79
Configure Package Descriptor File: package.xml	79
Configure Context Mapping (Filtering) File: contextmapping.xml	79
Configure Type Mapping File: typemapping.xml	80
Configure Attribute Mapping File: attributemapping.xml	82
Configure Relation Mapping File: relationmapping.xml	83
Customizing Synchronization and Scripting	84
Synchronization Package Locations	84
13 Scripting	85
Groovy Scripts	86
Enabling and Disabling Scripts	86
Groovy Script Location	87
Script Variables	87
Handling Errors	88
Sample Script: preUpload.groovy	89
14 Testing and Troubleshooting	91
Validating XML Configuration Files	91
XSD Files	91
Validating Files Automatically	92
Validating Files Manually	92
Dumping Synchronization Data	94
Creating a Synchronization Data Dump	94
Data Dump Example	95
Viewing a Synchronization Data Dump	96
Validating Mapping Rules	96
Writing Rules	97
Simplifying Rule Development	97
Avoiding Complex XPath Queries	97
Matching Against Existing Attributes Only	97
Avoiding Broad XPath Expressions	97
Service Discovery Server Log File Configuration	97
Troubleshooting, Common Issues, and Tips	98
Limitations	99
Delta Detection Limitations	99
Event-driven Synchronization Limitation	99

15 Mapping Engine and Syntax.....	101
Common Mapping File Format	101
Mapping File Syntax	102
Rules	102
Rule Conditions.....	102
Condition Examples.....	102
Operator Elements	103
Operand Elements	106
Mapping Elements	112
Condition Examples.....	112
Filtering.....	113
Context Mapping	113
Type Mapping	114
Mapping	114
Attribute Mapping	115
Mapping to String values in the RTSM	115
Mapping to String values of a maximum length in the RTSM	115
Mapping to Integer values in the RTSM	115
Mapping to Boolean values in the RTSM.....	115
Mapping to Long values in the RTSM	115
Mapping to Date values in the RTSM	116
Mapping to Float values in the RTSM	116
Mapping to Byte values in the RTSM	116
Mapping to Double values in the RTSM	116
Mapping to StringList values in the RTSM.....	116
Mapping to IntList values in the RTSM	116
Attribute Mapping Example	117
Relation Mapping	118
Root Container Mapping.....	118
Message Alias Mapping for CI Resolution	118
XPath Navigation	120
Data Structure	120
CI Data Structure	120
Relationship Data Structure.....	120
Example of an XPath-Navigated Data Structure	122
XPath Expressions and Example Values.....	123

Section III: Event Processing Interface	125
16 Event Processing Interface	127
Event Processing Interface and Scripts	127
Entry Points for Running EPI Scripts	128
Before CI/ETI Resolution	128
After CI/ETI Resolution	128
Before Storing Events	128
After Storing Events	129
Specifying a Script	129
EPI Script Execution	129
Creating Scripts	130
EPI Troubleshooting	130
Log Files	130
Debugging	130
Log File Entries	130
17 Scripts For Custom Actions	131
Specifying Custom Actions Scripts	131
Creating Scripts	131
18 Creating EPI and Custom Action Scripts	133
Script Definition Attributes	133
Groovy Script API	134
Script Definition Format	134
EPI Groovy Script Samples	135
SimpleExampleEPI.groovy	135
RegExample.groovy	137
ResolveLocationFromDB.groovy	138
Custom Actions Groovy Script Samples	139
SimpleExample.groovy	139
TranslateEventTitle.groovy	140
Section IV: Integrating Events	141
19 HP BSM Integration Adapter	143
Policies	145
Integrating Events with XML Interceptor Policies	145
Integrating Events with HP BSM Integration Adapter	146

Section V: Integrating the Operations Management UI with Other Applications . . .	149
20 URL Launch of the Event Browser	151
Specifying a URL Launch	151
Default URL Launch	151
Specifying Optional Parameters	151
Parameters and Parameter Values	152
Defining Columns	153
Setting Filters	155
Filtering by String Attributes	156
Filtering by Time Properties	157
Filtering by Priorities	157
Filtering by CIs and CI Types	157
Filtering by Global CI ID	158
Filtering by ETIs and ETI Values	158
Filtering by Other Event Characteristics	159
URL launch of the Event Details	159
Section VI: Automating Operator Functions and Event Change Detection	161
21 Automating Operator Functions using the Event Web Service Interface	163
How to Access the Event Web Service	163
How to Detect New Events	165
Receiving Events as Atom Feeds	166
Specifying Parameters to Filter the Event List	166
How to Detect Event Changes	167
How to Modify Events	167
Modifying Events Using a REST Client	167
Modifying Events Using RESTClient	168
Modifying Events Using the Firefox Poster Extension	170
More About Firefox Poster Extension	172
Modifying Events Using the RestWsUtil.bat Utility	173
Example: How to Change an Event Title	173
Advanced Modification of Event Properties	174
Bulk Update of Events	175
22 REST Web Service Command-Line Utility	177
How to Call the Utility Help	177
Examples	178
Reading Events	178
Reading Custom Attributes of an Event	178
Reading Annotations of an Event	179
Creating a Custom Attribute	179
Creating an Annotation	180
Updating a Custom Attribute	180
Updating an Annotation	181
Updating the Title of an Event	181
Updating the Lifecycle State of an Event	182
Updating the Lifecycle State and the Severity of an Event	182

Updating the Event with Transfer Control to Connected Server Information	182
Bulk Event Update: Updating the State and the Severity of an Event.	182
Deleting a Custom Attribute	183
Deleting an Annotation	183
23 Event Web Service Query Language.	185
HTTP Query Parameters.	185
Filtering by Date and Time: watermark	186
Filtering by Event Attributes: query	187
Simple Filters.	187
Compound Filters	187
Paging	188
start_index	188
page_size	188
Combining start_index and page_size	188
Ordering	189
order_by	189
order_direction.	189
Data Inclusion.	189
include_closed	189
include_relationships.	190
Media Type	190
alt	190
Query Filter Criteria Properties and Supported Operators	191
Complex Attributes	192
Editable Properties	194
History Lines	196
Recorded Property Changes	196
Event Changes List	197
File Locations	197

Section VII: Integrating External Event Processes	199
24 Forwarding Events and Synchronizing Event Changes	201
Event Forwarding and Synchronization Process	201
Forwarding Events and Event Changes to an External Event Process	201
Groovy Script Adapter	202
Event Synchronization Web Service	202
Receiving Event Changes Back from an External Event Process	204
Performing a URL Launch of the Event Browser from an External Application	205
25 Integrating External Event Processes Using Groovy Scripts	207
Sample Groovy Script: Logfile Adapter	207
Configure a Connected Server using the Logfile Adapter	207
Configure an Event Forwarding Rule	209
Groovy Script Interface	210
Important File Locations	210
API Library	210
Javadoc API Documentation	210
OPR Event Schema	210
Event Integration Groovy Scripts	211
Groovy Script Methods	212
init	212
destroy	213
ping	213
Methods for Forwarding Events to a Connected Server	213
forwardEvent	214
forwardChange	215
Methods for Receiving Synchronization Data from a Connected Server	216
receiveChange	216
Additional Methods	217
Control Transfer	217
Annotations	217
addAnnotation(def author, def text)	217
updateAnnotations(def id, def author, def text)	218
Custom Attributes	218
addCustomAttribute(def name, def value)	218
updateCustomAttribute(def name, def value)	218
HTTP Requests and Responses	218
getHttpRequestHeader(def name)	218
setHttpResponseStatus(def httpStatusCode, def httpResponseMessage)	218
setHttpResponseHeader(def name, def value)	218
Methods for Retrieving Data from a Connected Server	219
getExternalEvent	219
Method for Supplying Data to a Connected Server	219
toExternalEvent	219
Management of Groovy Script Methods	220

26 Event Synchronization Web Service Interface	221
Overview	221
Forward Events and Event Changes from OPR Client	221
Forward Event	221
Forward Event Change	222
Web Service GET Request	222
Web Service Ping Request	222
Synchronize Event Changes Back from External Client	222
Event Update	223
Event Change Creation	223
Web Service Ping Request	223
Configuring Connected Servers	224
Event Attributes that Support Back Synchronization	224
Event Update: Logfile Adapter Examples	225
Set and Unset the Event Description	225
Set the Description	226
Unset the Description	226
Set and Unset the Event Title	226
Set the Title	226
Unset the Title	226
Change Multiple Properties in One Call	226
Set Title and Description	226
Event Change Creation: Logfile Adapter Examples	227
Annotations	228
Insert an Annotation	228
Update an Annotation	228
Cause	229
Set the Cause of an Event	229
Unset the Cause of an Event	229
Custom Attributes	230
Insert a Custom Attribute	230
Update a Custom Attribute	230
Description	231
Set the Description	231
Unset the Description	231
State	232
Set the State	232
Priority	232
Set the Priority	232
Severity	233
Set the Severity	233
Solution	233
Set a Solution	233
Unset a Solution	234
Title	234
Set the Title	234
Unset the Title	234
Change Multiple Properties in One Call	235
Set the Title and Description	235

27 Integrating External Event Processes: Frequently Asked Questions	237
Getting Started	238
Groovy Scripts and Programming	241
Integration Script Methods	242
Event Properties	246
Troubleshooting	247
Logging	247
28 Service Manager Integration	249
Configure the HP Service Manager Server as a Connected Server	249
Configure an Event Forwarding Rule	252
Configure URL Launch of Event Browser from HP Service Manager	253
Configure HP Service Manager Server	253
Configure URL Launch of HP Service Manager from the Event Browser	254
Mapping and Customization	255
Testing the Connection	255
Synchronizing Attributes	256
Uni-directional Synchronization: Operations Management to HP Service Manager	256
Bi-directional Synchronization	256
Attribute Synchronization using Groovy Scripts	257
Tips for Customizing Groovy Scripts	257
Controlling Attribute Synchronization	257
Mapping OPR Lifecycle States to BDM Lifecycle States	258
Mapping BDM Lifecycle States to OPR Lifecycle States	258
Syntax Errors	258

1 Introduction

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 seven sections.

- Section I: [Content Development](#) 17
- Section II: [Populating the Run-Time Service Model](#) 59
- Section III: [Event Processing Interface](#) 125
- Section IV: [Integrating Events](#) 141
- Section V: [Integrating the Operations Management UI with Other Applications](#) 149
- Section VI: [Automating Operator Functions and Event Change Detection](#) 161
- Section VII: [Integrating External Event Processes](#) 199

Section I: [Content Development](#) 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](#) 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](#), is developed further to illustrate how to create topology synchronization rules specific to a particular service model.

Section III: [Event Processing Interface](#) describes the role of event processing scripts and custom actions for modifying and enhancing events during event processing.

Section IV: [Integrating Events](#) provides integrators with information about integrating events from other applications such as HP Network Node Manager i (HP NNMi) with HP Business Service Management (BSM) Operations Management using HP BSM Integration Adapter.

Section V: [Integrating the Operations Management UI with Other Applications](#) describes how to integrate parts of the Operations Management user interface with an external application using a drill-down URL launch.

Section VI: [Automating Operator Functions and Event Change Detection](#) 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 VII: [Integrating External Event Processes](#) 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 using this guide are as follows:

- You should be familiar with the relevant HP Software products and components, including the associated documentation:
 - Operations Manager *i* (OMi).
 - HP Operations Manager (HPOM) for Windows and UNIX.
 - HP Business Service Management (BSM).
 - Run-Time Service Model (RTSM). Detailed administrative and operational knowledge is assumed.
- 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.

Section I: 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:

• Chapter 2, Integration Content	19
• Chapter 3, Topology	23
• Chapter 4, Event Type and Health Indicators	33
• Chapter 5, Correlation Rules and Mapping	43
• Chapter 6, Additional Event Processing	45
• Chapter 7, Tools	47
• Chapter 8, View Mappings	49
• Chapter 9, Graphs	51
• Chapter 10, Packaging Content	53

2 Integration Content

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

- [Topology](#)
- [Event Type and Health Indicators](#)
- [Correlation Rules](#)
- [Additional Event Processing](#)
- [Tools](#)
- [View Mappings](#)
- [Graphs](#)

Topology

Topology data is contained in an 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 [Chapter 3, Topology](#) on page 23.

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 [Chapter 4, Event Type and Health Indicators](#) on page 33.

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 [Chapter 5, Correlation Rules and Mapping](#) on page 43.

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 [Chapter 6, Additional Event Processing](#) on page 45.

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 [Chapter 7, Tools](#) on page 47.

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 [Chapter 8, View Mappings](#) on page 49.

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 [Chapter 9, Graphs](#) on page 51.

3 Topology

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

This chapter is structured as follows:

- [Integrating New Applications](#)
- [Topology View of the New Application](#)
- [Impact Propagation](#)
- [Enrichment Rules](#)

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.

Figure 1 shows the topology model of the “ACME” environment.

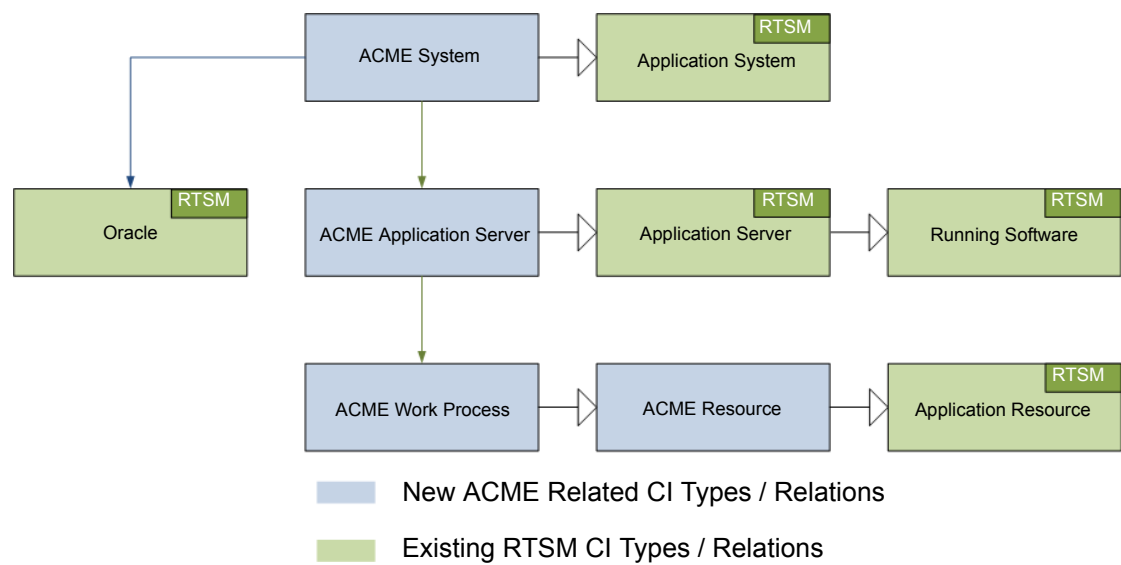


Figure 1 ACME topology model

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. Figure 1 on page 24 shows four new CI types, depicted in blue:

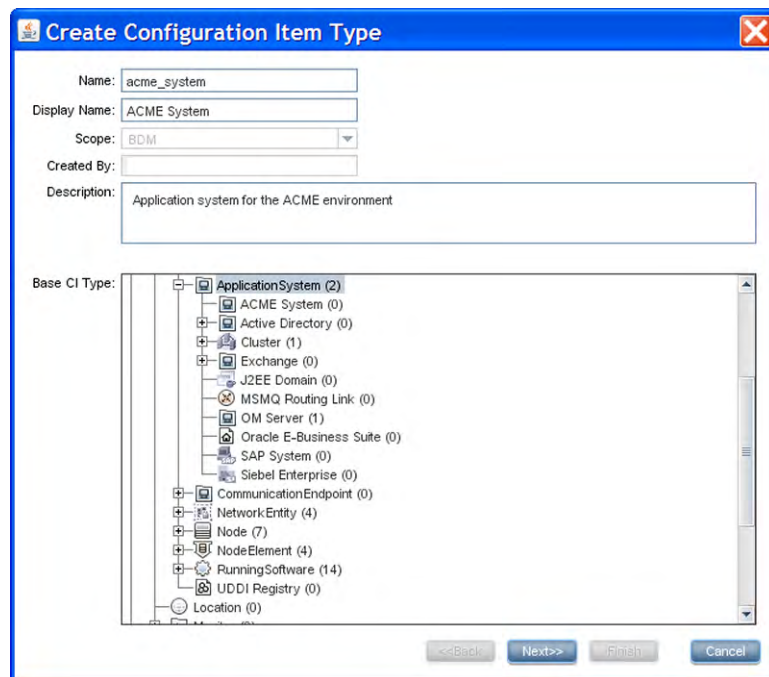
- ACME System
- ACME Application Server
- ACME Work Process
- ACME Resource

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 *HP Business Service Management 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](#) on page 26.

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.

Table 1 shows a list of potential key attribute for an ACME environment.

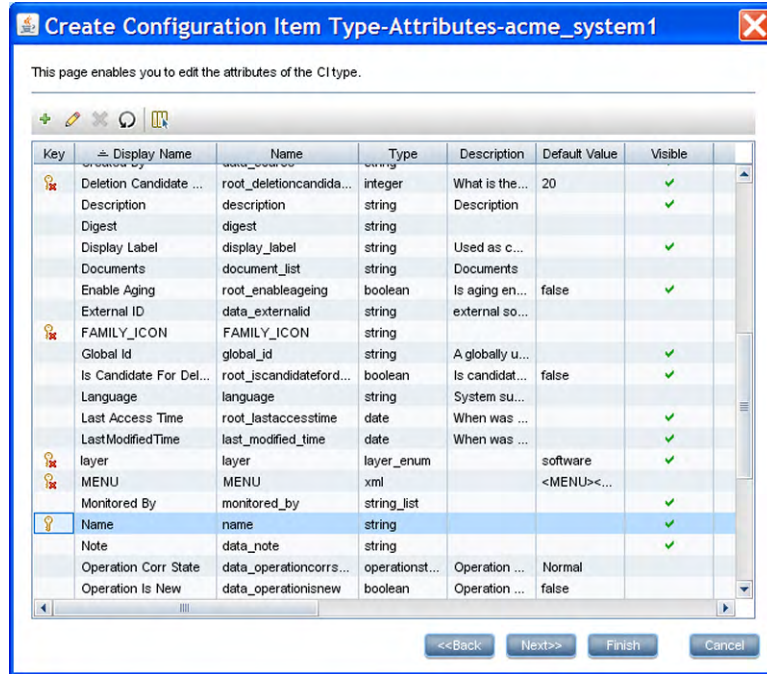
Table 1 List of CI Type Attributes for the ACME Environment

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

To set key attributes for the new CI types, continue from [step 5](#) on page 25.

- 1 In the Create Configuration Item Type-Attributes dialog, set the key attributes for the CI type, for example, ACME System.

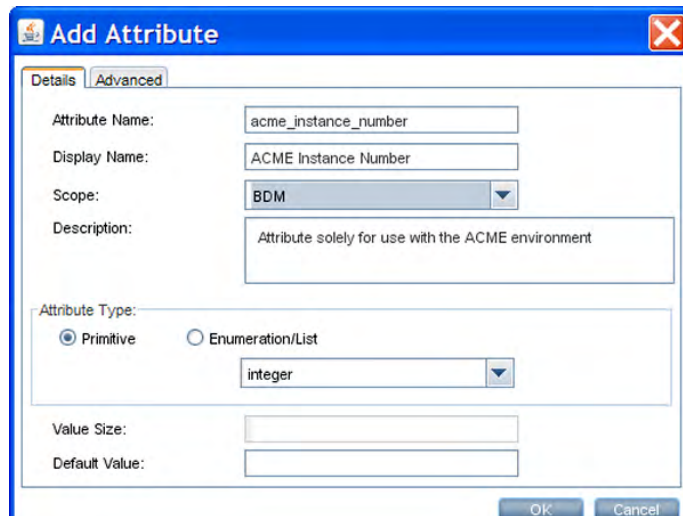
To identify your new CI type with an existing attribute, click in the column marked **Key** on the same row as the attribute you want to set as a key attribute. A small key icon then appears. (Click again if you do not wish to set that attribute.)



- 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](#) on page 27.
- 4 Click **Finish**.

Establish Relationships for the New Application

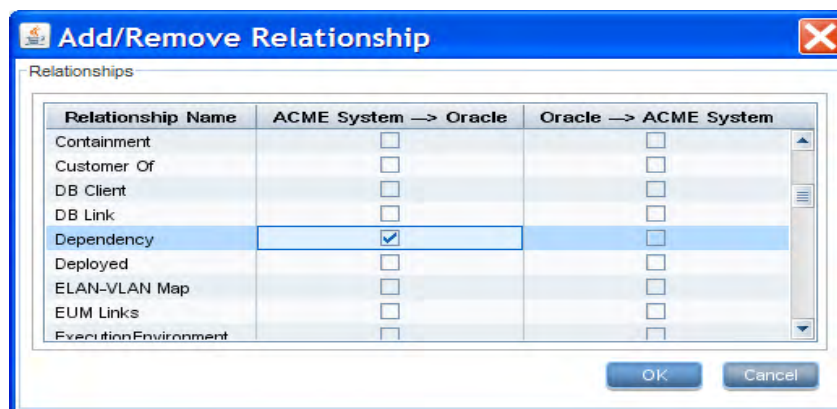
The next step in integrating the new application is to establish the relationships for the new application.

In the out-of-the-box RTSM model, there is a **membership** relationship between the ACME system and the ACME application server. As illustrated in [Figure 1](#) on page 24, 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 [Section II, Populating the Run-Time Service Model](#) on page 59.

- 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 (DFM) 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 DFM 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 DFM, 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 *Data Flow Content Guide* in the HP Business Service Management documentation library.

Creating CIs Using the HPOM Service Model and 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 [Section II, Populating the Run-Time Service Model](#) on page 59.

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 DFM

Use DFM if:

- You already use DFM to populate the RTSM, or are planning to use it.
- You do **not** already have a service model in HPOM. We recommend using DFM, 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) DFM 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

Figure 1 on page 24, 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 *HP Business Service Management Modeling Guide*.

Topology View of the New Application

Use the RTSM Modeling Studio to create a view to display the ACME topology. Figure 2 on page 31 shows the ACME topology view in the RTSM Modeling Studio.

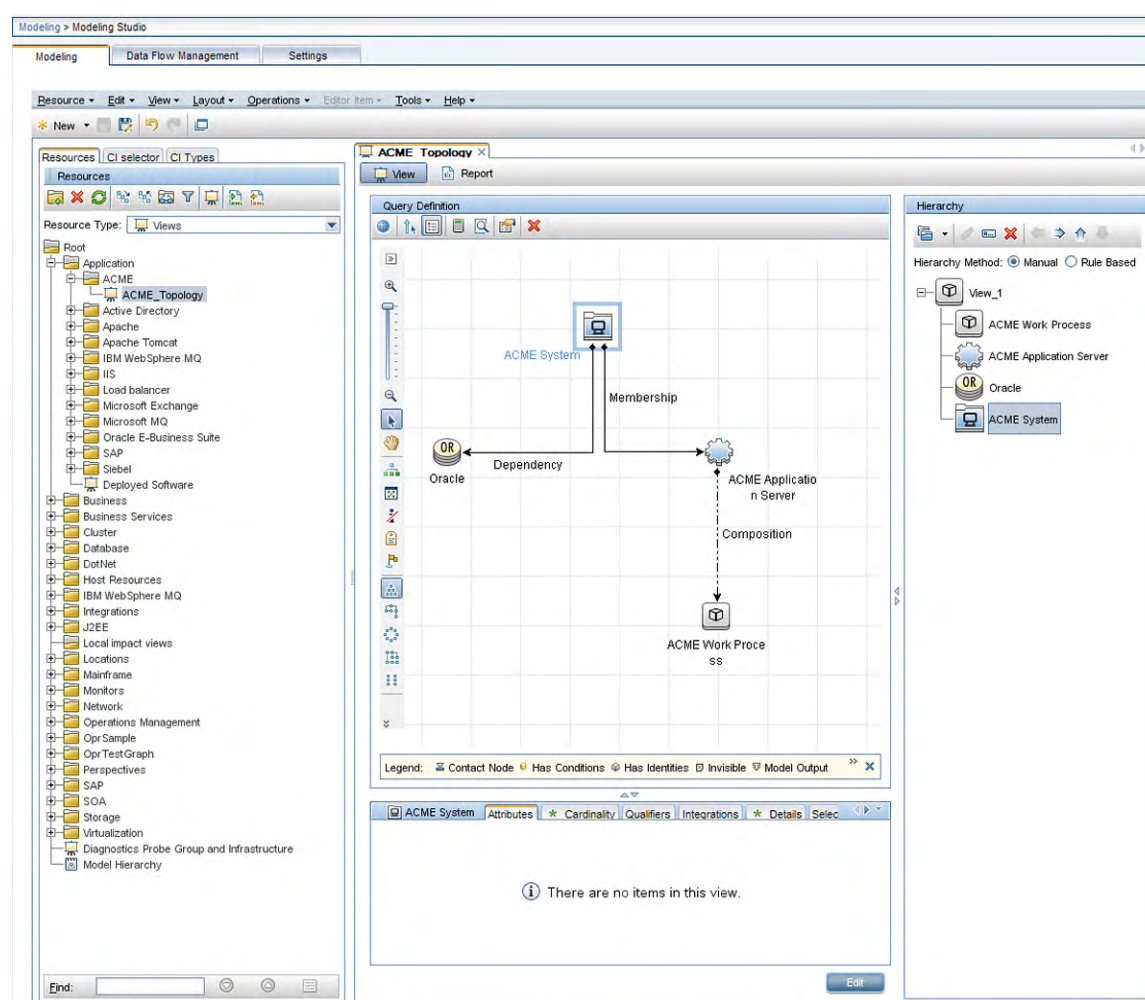


Figure 2 ACME topology view in RTSM Modeling Studio

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 *HP Business Service Management Modeling Guide*.

- ▶ 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.

Figure 3 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.

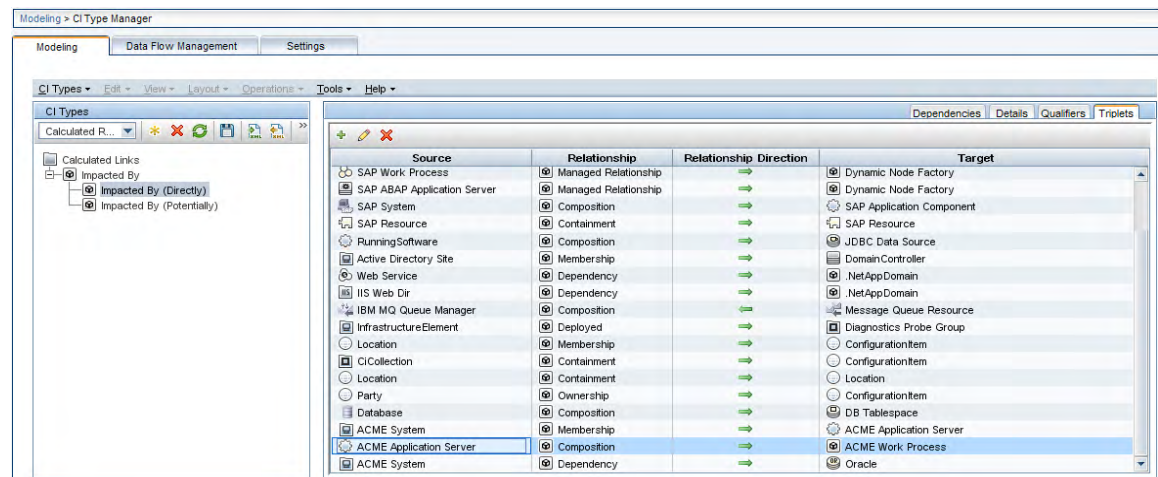


Figure 3 Creating an impact relation between the ACME Application Server and the ACME Work Process

4 Event Type and Health Indicators

This chapter 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`

Figure 4 shows an example of how to set the CMA `RelatedCiHint` in the HPOM message:

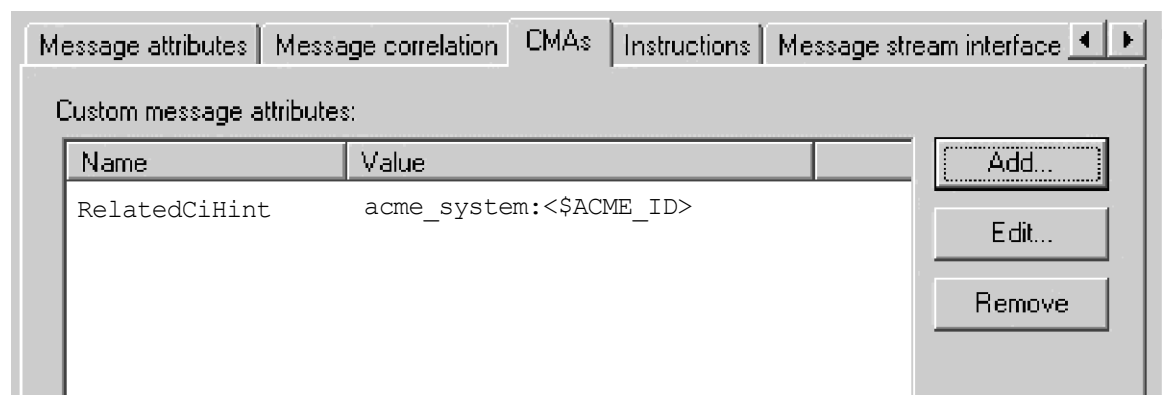


Figure 4 Setting custom message attribute `RelatedCiHint` in the HPOM message

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.

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 41, 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 16, and illustrated in [Figure 1](#) on page 16.

For these CI types, there are specific ETIs and HIs. [Table 2](#) on page 36 shows the result of an investigation into which ETIs/HIs are important within an ACME environment.

Table 2 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

Table 2 Overview of ETIs and HIs

CI Type Display Name	Category	Name	Value	Severity	Policy
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



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 → Design Operations Content → 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 → Design Operations Content → 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.

New Health Indicator

General

* Display Name: ACME System Status

* Name: ACME_System_Status

Type: Health Indicator with associated Event Type Indicator

Description: HI for ACME system status/availability

Application: Both Service Health and SLM

Units:

States

Display Name	Status	Icon
NORMAL [Default]	Normal	✓
CRITICAL	Critical	✗

Service Health

☐ Generate Events SEM Configuration

Default Rule:

Save Cancel Help

- 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.

New Indicator State

* Display Name: AVAILABLE

* Name: AVAILABLE

Default: ☒

Status: Normal

Icon: basic/ind6_grn.gif

Save Cancel Help

- 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”

Table 2 on page 36 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`.

Figure 5 on page 40 shows an example configuration of the CMA `EtiHint`.

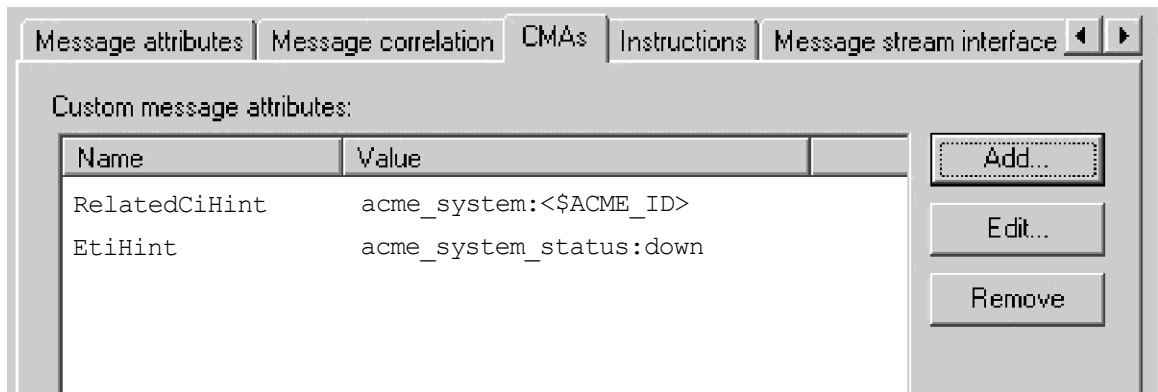


Figure 5 Custom message attribute EtiHint

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.

[Figure 6](#) on page 40 shows an example of a filter configuration for db connection down related messages.

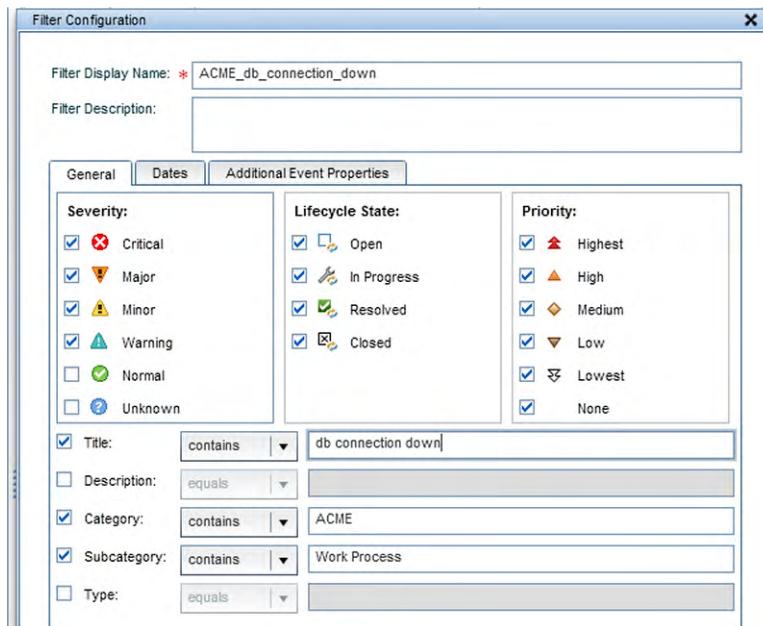


Figure 6 Filter example showing db connection down related messages

[Figure 7](#) on page 41 illustrates the configuration of the mapping rule itself for the Lost Database Connection ETI, using the filter configuration shown in [Figure 6](#) on page 40.

The screenshot shows a dialog box titled "ACME WP lost db connection - Create New Mapping Rule". It has three main sections: "General", "Filter Events", and "Map Event to Indicator".

- General:**
 - Display Name: * ACME WP lost db connection
 - Name: * ACME_WP_lost_db_connection
 - Description: (empty text box)
 - Active: ☒
- Filter Events:**
 - Event Filter: * ACME_db_connection_down (dropdown menu)
- Map Event to Indicator:**
 - Indicator: * Lost Database Connection (dropdown menu)
 - Map to Indicator Value:
 - ☐ Based on severity
 - ☒ Specific Indicator Value: * OCCURRED (dropdown menu)

Figure 7 Mapping rule configuration

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 [Table 2](#) on page 36, mostly represent the availability status of the specific CI. Therefore, those HIs are assigned to the Operations Availability KPI.

[Figure 8](#) on page 42 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.

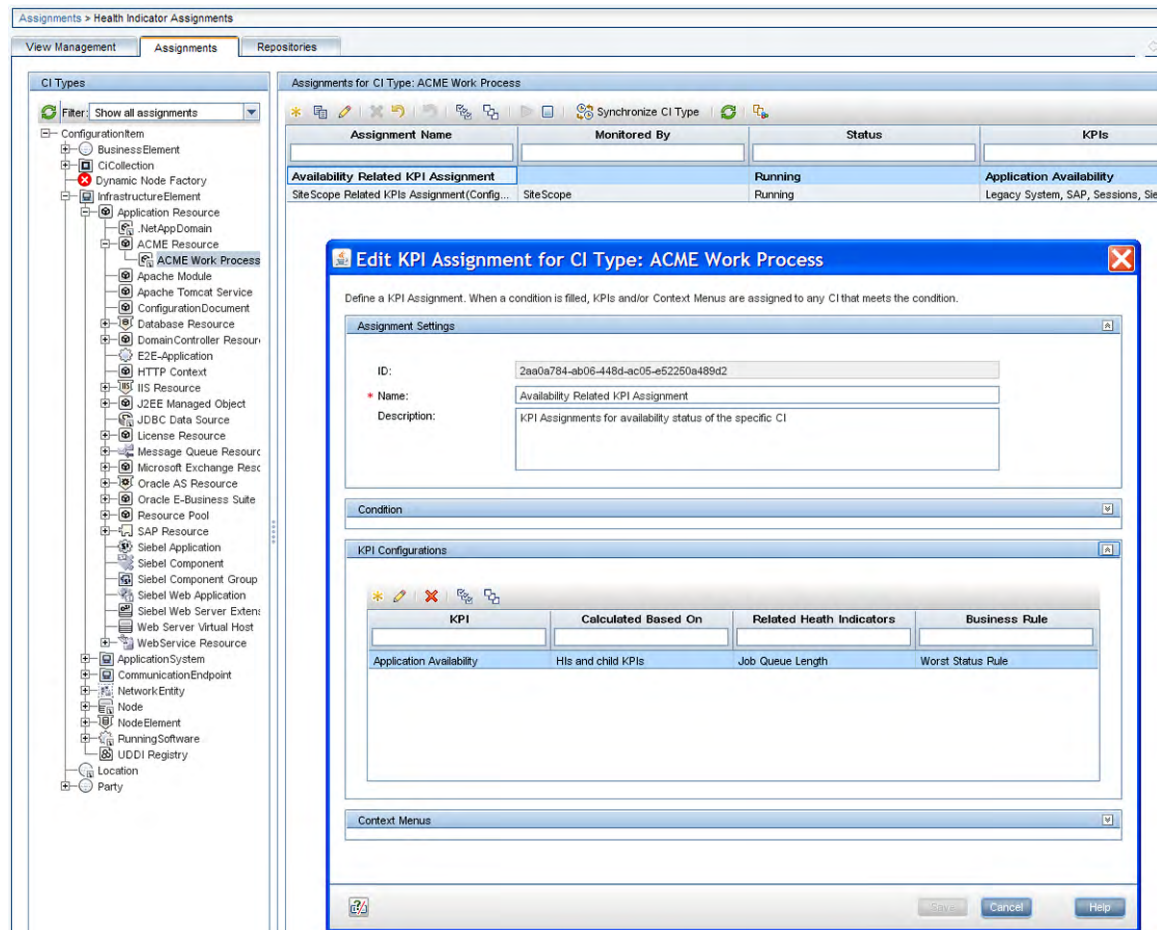


Figure 8 KPI assignment

When you have finished with assigning HIs to KPIs, perform a CI type synchronization for the assignment to take effect on existing CIs.

5 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 [Figure 1](#) on page 16, 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. [Figure 9](#) 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.

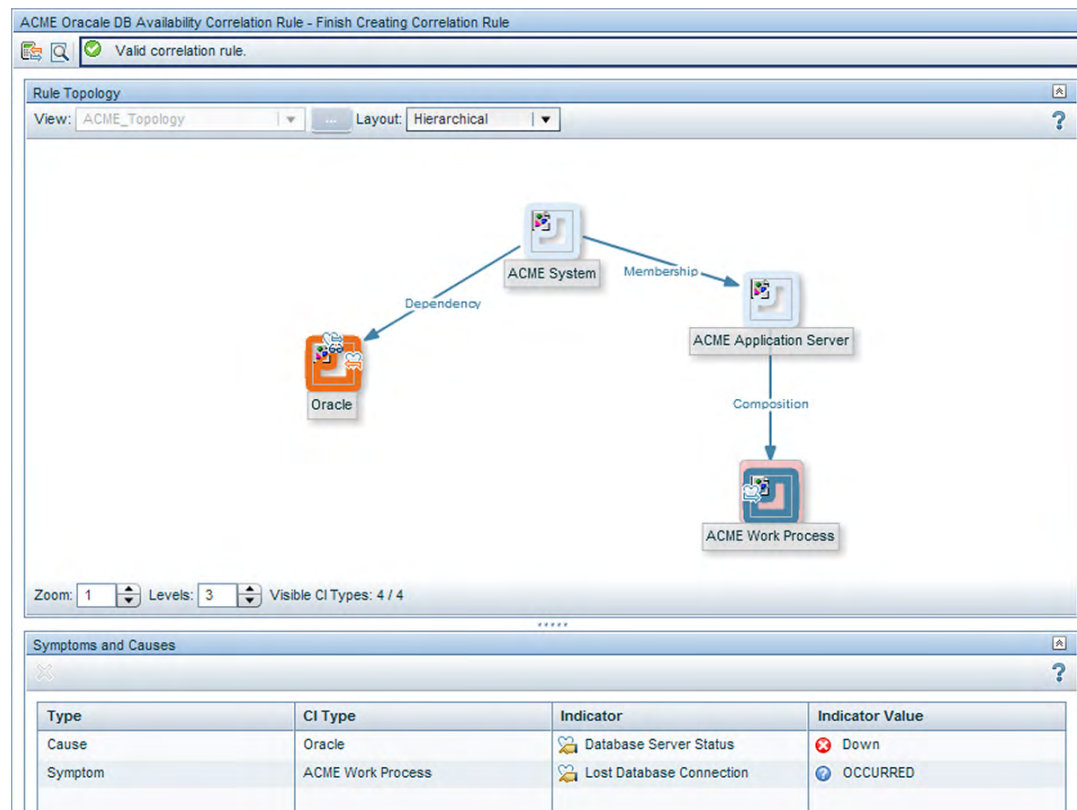


Figure 9 Correlation rule for the ACME landscape

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.

6 Additional Event Processing

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

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 [Chapter 16, Event Processing Interface](#) on page 127.

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 [Chapter 17, Scripts For Custom Actions](#) on page 131.

For details about how to create EPI and custom action scripts, including some sample Groovy scripts provided with the product, see [Chapter 18, Creating EPI and Custom Action Scripts](#) on page 133.

7 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.

Figure 10 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.

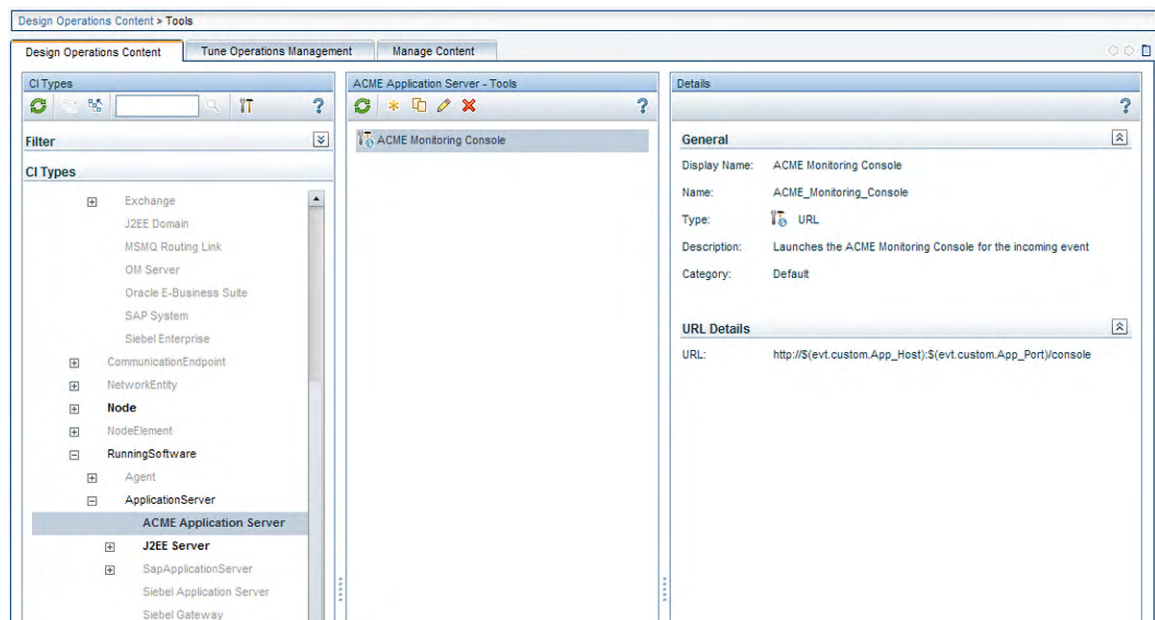


Figure 10 Tool for ACME application server

8 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.

Figure 11 shows the View Mappings pane. The CI type `ACME System` is mapped to (has an association with) the view `ACME_Topology`.

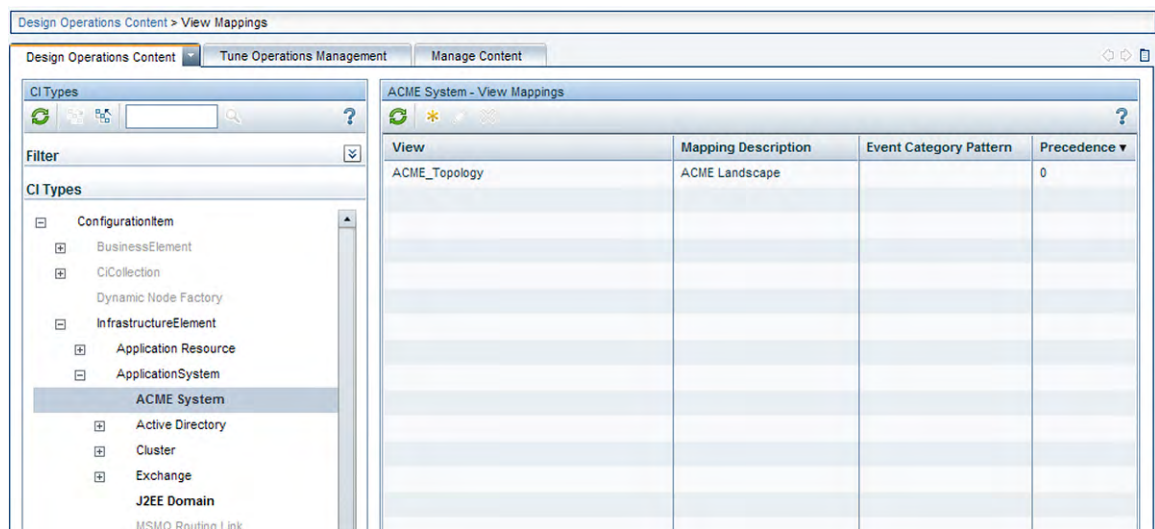


Figure 11 ACME System - View Mappings

9 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.

10 Packaging Content

This chapter 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 BSM Operations 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 BSM Operations 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 BSM Operations Management.

The workflow for creating an ACME RTSM package is summarized below:

- Create the ACME RTSM package.
For more information about how to create a content pack, see the HP Business Service Management online help.
- Add the CI types (see [Chapter 3, Topology](#) on page 15), and views (see [Chapter 8, View Mappings](#) on page 39) that you created in previous steps.

[Figure 12](#) on page 54 shows the selected items which are part of the ACME RTSM package. Usually CI types, TQLs and views are part of such a package.

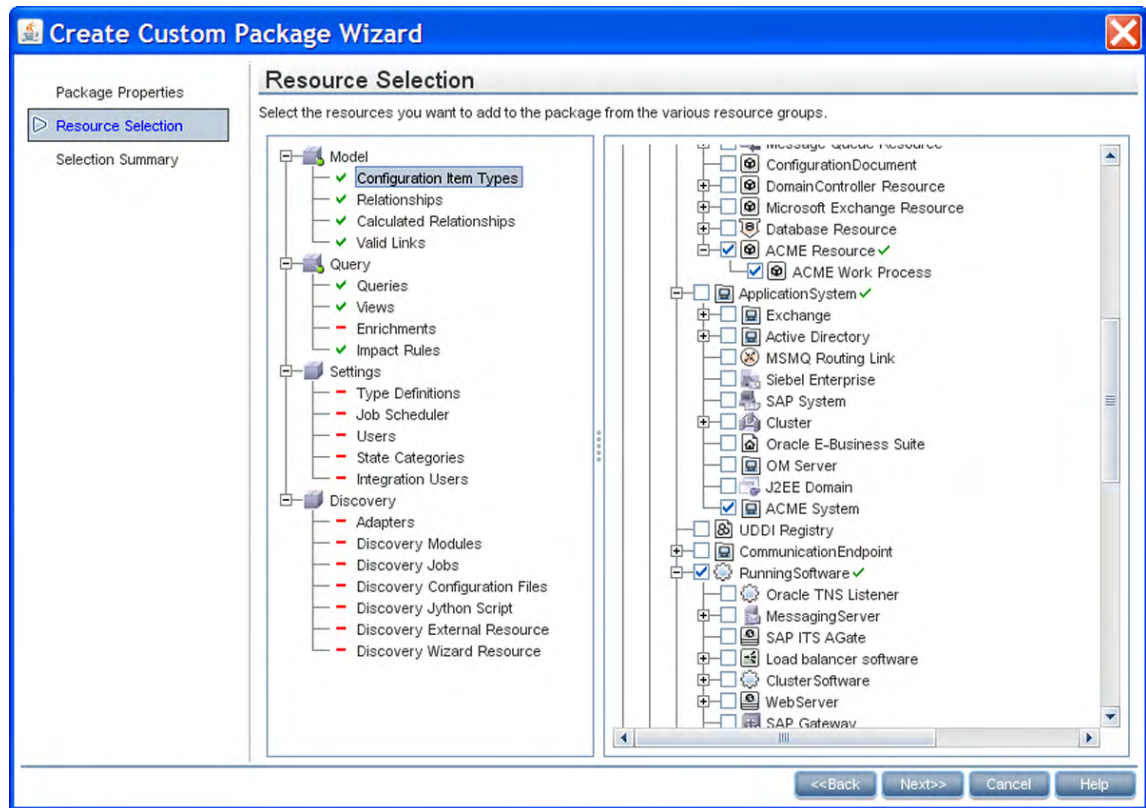


Figure 12 ACME RTSM package resource selection

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 [Section II, Populating the Run-Time Service Model](#) on page 59.

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 BSM Operations 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 BSM Operations Management.

[Figure 13](#) on page 56 shows a selection in the content manager within the ACME environment content pack.

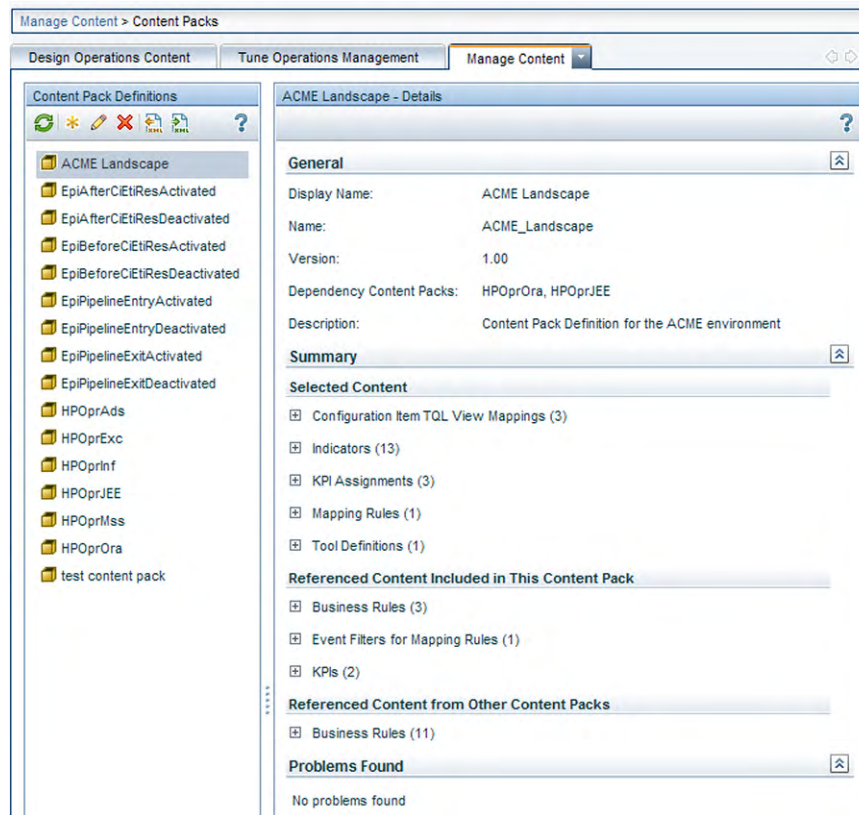


Figure 13 ACME environment content pack selection

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.

Section II: Populating the Run-Time Service Model

This section provides information for developers to create their own topology synchronization mapping rules to populate the Run-Time Service Model (RTSM) 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:

- [Chapter 11, Topology Synchronization Overview](#) 61
- [Chapter 12, Synchronization Packages](#)..... 75
- [Chapter 13, Scripting](#)..... 85
- [Chapter 14, Testing and Troubleshooting](#) 91
- [Chapter 15, Mapping Engine and Syntax](#) 101

11 Topology Synchronization Overview

Topology synchronization is used instead of, or in conjunction with, HP Data Flow Management (DFM) discovery to create CIs in the RTSM. 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).

Topology synchronization can be defined as a set of rules that determine how services, nodes, and node groups monitored by HP Operations Manager (HPOM) for Windows or UNIX are mapped to Operations Management configuration items (CIs) in the Run-Time Service Model (RTSM).

Topology synchronization is a standard part of the Operations Manager *i* license. It offers a solution, at no extra cost, 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. During installation, the topology synchronization rules are copied 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 their service model.

Topology synchronization utilizes the RTSM API, and uses Wiseman, JAXB, JDOM, JXPath, SPRING, Hibernate, and Groovy scripting technology.

There are two methods available for topology synchronization:

- [Basic Topology Synchronization](#) (see [page 62](#))

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.

- [Dynamic Topology Synchronization](#) (see [page 66](#))

Dynamic topology synchronization receives discovered topology data from *multiple* HPOM systems and updates CIs and CI relationships in the RTSM as soon as changes are discovered.

Both topology synchronization methods use the same synchronization packages to map discovery data to CIs and CI relationships in the RTSM.

Basic Topology Synchronization

Basic topology synchronization is a command-line tool, running on demand on the BSM data processing server.

To create CIs in the RTSM, 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.
- 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.



The definition of particular CI types is especially important when synchronizing the topologies of Operations Management and HP Operations Manager 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

Figure 14 provides an overview of the topology synchronization architecture.

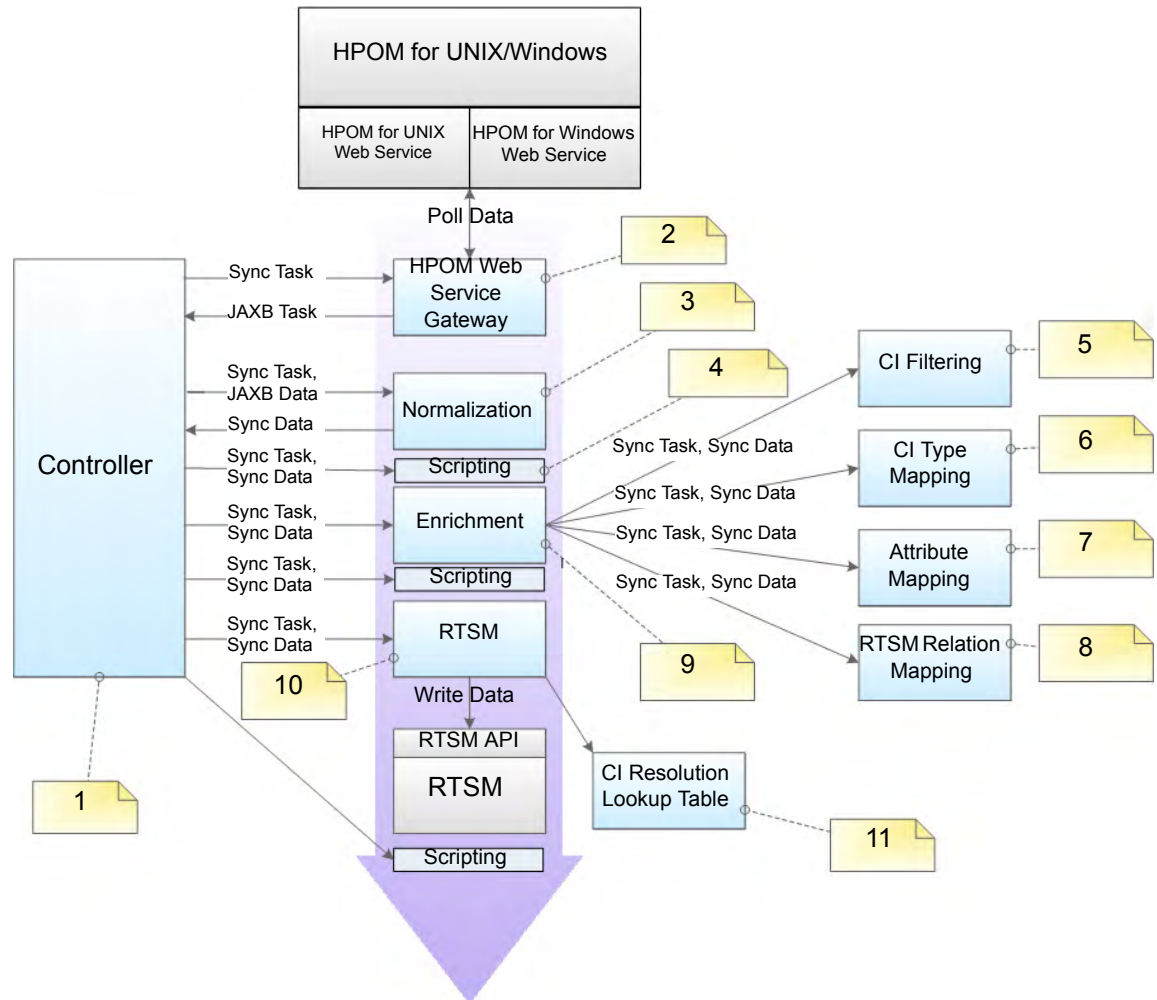


Figure 14 Basic Topology Synchronization Architecture

The numbers in the diagram refer to the following notes:

Reference	Refers to	Note
1	Controller	<ul style="list-style-type: none"> 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.

Reference	Refers to	Note
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 types.
7	Attribute Mapping	Map untyped HPOM relations to typed RTSM relations.
8	RTSM Relation Mapping	<ul style="list-style-type: none"> • 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 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` 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. The normal mode also performs delta detection and deletes elements from the RTSM that have been deleted in HPOM.

To run the `opr-startTopologySync` tool in normal mode, enter the following command:

```
<HPBSM root directory>/bin/opr-startTopologySync.bat
```


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` tool in touch mode, enter the following command:

```
<HPBSM root directory>/bin/opr-startTopologySync.bat -touch
```

For more details about RTSM aging, refer to the *HP Business Service Management 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` tool in normal mode with the option to skip the loading of HPOM services:

```
<HPBSM root directory>/bin/opr-startTopologySync.bat -skipservices
```

Dynamic Topology Synchronization

Dynamic topology synchronization makes it possible to share topology data from HPOM services, nodes, and node groups between any supported combination of multiple HPOM and BSM Operations Management instances, in a distributed, hierarchical environment.

An OMi instance can be configured to receive topology data from multiple HPOM and other BSM Operations Management instances. An BSM Operations Management instance can also be configured to forward topology data to other BSM Operations Management instances. Topology data is forwarded to the BSM Operations Management instance dynamically, that is, as soon as a topology change is discovered by an HPOM agent, and written to the HPOM service model. Dynamic topology synchronization enables near-real-time discovery of infrastructure changes, so for example, if a cluster package is switched from one node to another, this change is immediately forwarded, and the database updated.

For details of supported HPOM versions, see the *HP Business Service Management 9.00 Readme* (release notes).

Dynamic topology synchronization forwards only the discovered topology changes originating from an HPOM agent or an HPOM system to another supported HPOM or BSM Operations Management instance. However, changes in topology of an BSM 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 *ODB Developer Reference Guide*.

Once configured, dynamic topology synchronization runs continuously in the background. Unlike basic topology synchronization, you do not need to start it manually.

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 *HP Business Service Management Modeling Guide*.

To create CIs in the RTSM, based on mapping rules, and using the HPOM nodes, node groups and the HPOM service model 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.



The definition of particular CI types is especially important when synchronizing the topologies of Operations Management and HP Operations Manager 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 utilizes two complementary methods for synchronizing topologies:

- Event-driven Synchronization
- Scheduled (or Time-driven) Synchronization

Event-driven Synchronization

Dynamic topology synchronization provides dynamic updates to changes in topology data. Event-driven synchronization means that changes in a node or service model are forwarded immediately to a configured connected server.

Dynamic topology synchronization ensures programmatically that topology data in both the model database and the RTSM, sourced from the various HPOM services, nodes and node groups, is accurate, current, and almost instantly updated whenever something changes in the environment. Accurate and up-to-date CI topology data is essential for Topology-based Event Correlation (TBEC) and BSM-wide service health monitoring.

Advantages of event-driven synchronization include the following:

- Immediate (near real-time) synchronization
- Low resource consumption

Scheduled (or Time-driven) Synchronization

Scheduled synchronization is based on two auto-discovery policies (`DiscoverOMTypes` and `DiscoverOM`) configured on the HPOM management server, and deployed to the Operations agent. These policies are scheduled to run periodically (typically once a day). The auto-discovery policies discover services in the monitoring environment, and automatically populate your HPOM service hierarchy. This topology data is forwarded to Operations Management, and written to the RTSM.

Advantages of scheduled synchronization include the following:

- Guaranteed delivery
- Complete service model data can be sent later if required by running `ovagtrep -publish` (in the use case where dynamic topology synchronization is being used to replace basic topology synchronization)

The main disadvantage of scheduled synchronization is the high load it places on the system.

To maximize user benefits, by default both approaches are used in combination. However, it is also possible to use each approach separately if this fits better with user requirements.

Dynamic Topology Synchronization Architecture

Figure 15 provides an overview of the dynamic topology synchronization architecture.

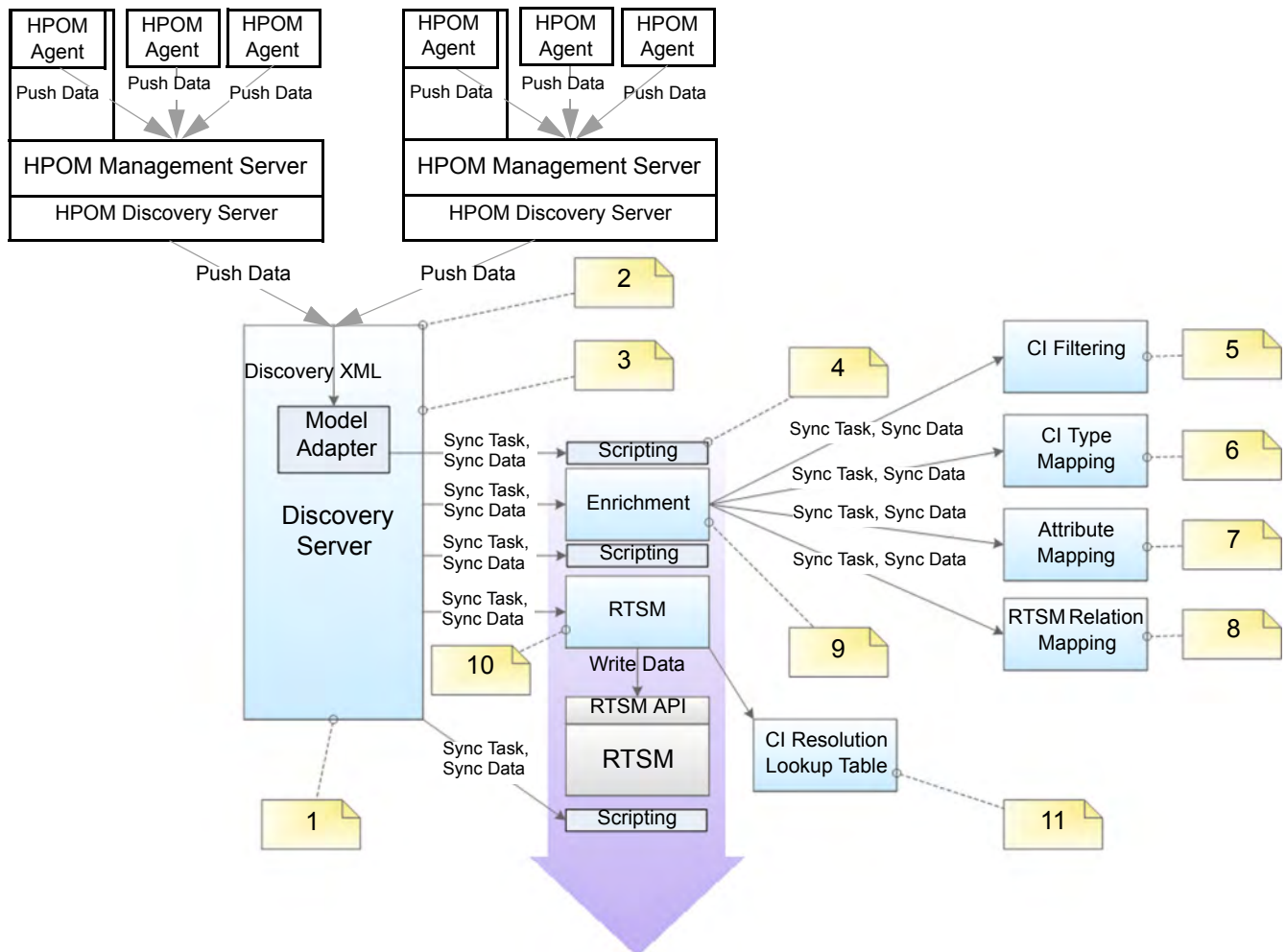


Figure 15 Dynamic Topology Synchronization Architecture

The reference numbers in the diagram refer to the following notes:

Reference	Refers to	Note
1	Discovery Server	<ul style="list-style-type: none"> Controls the synchronization process and passes data from one component to the other. Runs in <code>hpsbm_wde</code> process on the gateway servers
2	Receiving topology data on the Discovery Server	Receive topology data in the form of a discovery XML forwarded from an HPOM agent or another HPOM server.
3	Model Adapter	Convert the discovery XML to the synchronization data structure.
4	Scripting	Execute Groovy scripts to manipulate synchronization data.

Reference	Refers to	Note
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 types.
7	Attribute Mapping	Map HPOM relations that are not typed to RTSM relations.
8	RTSM Relation Mapping	<ul style="list-style-type: none"> 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.

Comparing Basic and Dynamic Topology Synchronization

Table 3 provides a comparison between basic and dynamic topology synchronization.

Table 3 Basic and Dynamic Topology Synchronization: a Comparison

Basic	Dynamic
Run from the command-line.	Once configured, runs automatically and 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 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 be updated in the RTSM, therefore resource consumption is much lower than with basic topology synchronization.
Requires additional 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. 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 78). 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 76.

- 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 77.

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 79.

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 Synchronization Packages to the Database](#) on page 73.

For more information about synchronization packages and mapping, see [Chapter 12, Synchronization Packages](#).

Scripting

Scripting enables you to perform additional processing and customizing during the synchronization process before the mapping and before and after the upload of data from HPOM nodes, node groups and services to the RTSM.

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 [Chapter 13, Scripting](#).

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. 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 processing server.

Basic Topology Synchronization

You can find the topology synchronization files for basic topology synchronization in the following locations:

Binaries:

```
<HPBSM root directory>/bin/opr-startTopologySync.bat  
<HPBSM root directory>/bin/opr-sd-tool.bat  
<HPBSM root directory>/opr/lib/opr-ts-*.jar
```

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:

```
<HPBSM root directory>/bin/opr-sd-tool.bat
```

```
<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 and HPOM topologies, 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.

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

➤ 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:

Infrastructure Settings → Applications → Operations Management → Operations Management - HPOM Topology Synchronization Settings

You can access the connection settings here:

Infrastructure Settings → Applications → Operations Management → Operations Management - HPOM Topology Synchronization Connection Settings

The screenshot shows the 'Infrastructure Settings Manager' interface. It has a 'Select Context' section with three radio buttons: 'Applications' (selected), 'Foundations', and 'All'. Below this, there are two main sections of settings:

Operations Management - HPOM Connection Settings

Name	Description	Value
Forwarding Retries	Configures the number of retries for the HPOM forwarder.	60
Forwarding Retry Interval	Specifies the time that the HPOM forwarder waits before trying to reconnect to an unreachable target server (seconds).	60
HPOM Host	Hostname of the HPOM system to which HP OMI connects.	
HPOM Password	Password that is used for the HPOM connection.	*****
HPOM Port	Port that is used for the HPOM connection.	443
HPOM Type	Configures the Operations Manager system type (HPOM for Windows or HPOM for UNIX).	OM for Windows
HPOM User	User name that is used for the HPOM connection.	
HTTPS HPOM Web Service Connection	Specifies whether HTTPS is used to connect to the HPOM web service.	true
Server-Based Flexible Management	Enables (true) forwarding to HPOM for existing events.	true

Operations Management - HPOM Topology Synchronization

Name	Description	Value
Dump Data	Enables (true) the saving of the data from all processing steps to the hard disk. This is not recommended for production systems, as it impacts performance.	false
Groovy Scripts	Enables (true) Groovy script usage to manipulate the synchronization data during the synchronization process.	true
Packages for Topology Sync	Semicolon-separated list of packages that are used for topology synchronizations.	default,nodegroups,operations-agent
Resolve IPs During Synchronization	Enables (true) IP resolution for nodes without IP address information in HPOM. Note: Enabling has a negative impact on synchronization performance.	false

Uploading Synchronization Packages to the Database

For both basic and dynamic topology synchronization, the command-line tool `opr-sd-tool` is provided to upload new or changed synchronization packages from the file system to the database, using the `-uploadpackage` command-line option.

To upload new or changed synchronization packages to the database, run the following command:

```
opr-sd-tool.bat -uploadpackage <path of synchronization package>
```

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.

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-sd-tool` 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:

```
opr-sd-tool.bat -uploadstd <path of MofFile>
```

12 Synchronization Packages

This chapter describes the topology synchronization packages that contain the rules for mapping HPOM services, nodes, and node groups to CIs in the RTSM.

The chapter is structured as follows:

- [Synchronization Packages Overview](#)
- [Standard Out-of-the-box Synchronization Packages](#)
- [Additional Out-of-the-box Synchronization Packages](#)
- [Package Descriptor File: package.xml](#)
- [Mapping Files](#)
- [Configuring Topology Synchronization: ACME Example](#)
- [Customizing Synchronization and Scripting](#)
- [Synchronization Package Locations](#)

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 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 HPOM services, nodes, and node groups into CIs in the RTSM.
- Synchronize CIs in the RTSM with data from specified services and nodes in 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 77).

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 78.

For basic information on mapping, see [Chapter 15, Mapping Engine and Syntax](#).

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 86.

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.
 - 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 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 page 76), the list would look like this:

```
default;nodegroups;operations-agent;HPOPrOra
```

- 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 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 of the package.

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 of the package.

- 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.



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 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. In other words, the 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 its 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.

Figure 16 shows a service tree from HPOM discovery.

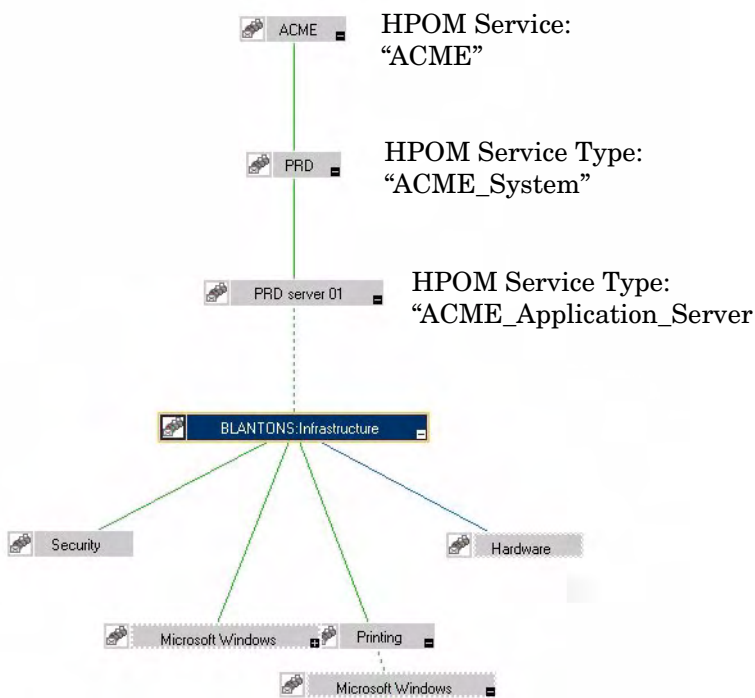


Figure 16 Service Tree from HPOM Discovery

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 of an HPOM service tree you want to include in the topology synchronization for mapping in the RTSM. The mapping rules contained in other configuration files are applied to the tagged services.

Figure 17 on page 80 represents the view for ACME in the RTSM.

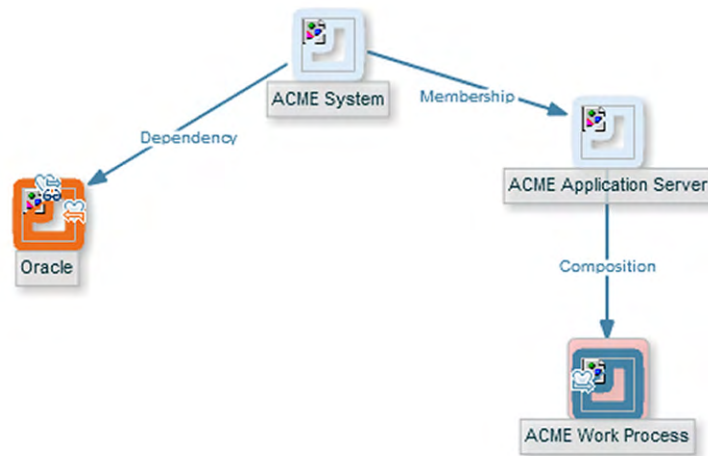


Figure 17 ACME View in the RTSM

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.

For the ACME example, the type mapping is defined in [Table 4](#) on page 81:

Table 4 Type Mapping for the ACME Example

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 service type attributes of a service in HPOM and the attributes of a CI in the RTSM.

For the ACME example, [Table 5](#) shows which HPOM service attributes are mapped to which CI attributes in the RTSM.

Table 5 Attribute Mapping for the ACME Example

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 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 data from HPOM nodes and services to the RTSM. One pre-mapping, one pre-upload, and one post-upload script can be associated with each synchronization package.

For details, see [Chapter 13, Scripting](#).

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>
```

13 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 from HPOM nodes and services to the RTSM.
- Post-upload scripts are executed after the upload of data from HPOM nodes and services 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 page 84.

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.



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 Operations Management host can edit scripts. This protects the scripts by the log-on security of the Operations Management host.

Groovy Scripts

Groovy scripting is supported. Groovy is a high level, object-oriented scripting language for the Java platform, which compiles 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, 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/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.

Table 6 shows the relationship between synchronization status (successful or unsuccessful synchronization) and script behavior.

Table 6 Relation to Synchronization Status

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 <code>AbortSyncOnError</code> 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 <code>abortSync(String)</code> 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.getConfigurationsItems()) {

    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");
}
}
```


14 Testing and Troubleshooting

This chapter contains information on:

- [Validating XML Configuration Files](#)
- [Dumping Synchronization Data](#)
- [Writing Rules](#)
- [Troubleshooting, Common Issues, and Tips](#)
- [Limitations](#)

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. We recommend using Eclipse, although you can use another editor of your choice that is capable of validating an XML file against a schema.

XML Schema Definition (XSD) 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 <code>package.xml</code> file in each synchronization package.
containmentrelations.xsd	Validates the <code>containmentrelations.xml</code> 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 mapping rules file, add the following on UNIX installations:

```
<?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 [Figure 18](#) on page 93 for an example.

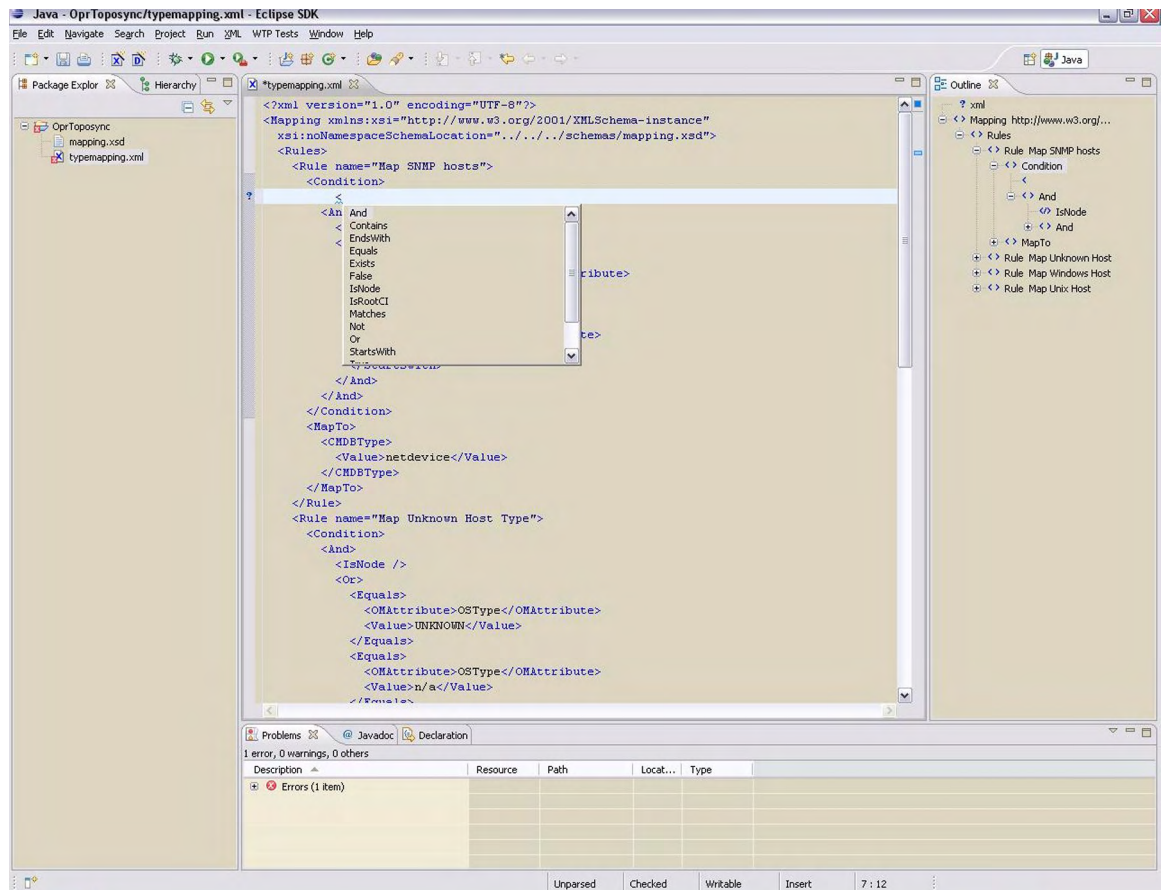


Figure 18 Example of Validating a File



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 by HPOM service elements to the RTSM and the data changed and added during the enrichment.
- Create a dump file to check XPath expressions of rules.

Creating a Synchronization Data Dump

A synchronization data dump contains the synchronized HPOM service elements 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:

```
<HPBSM root directory>/bin/opr-startTopologySync.bat
```

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 three subdirectories:

- **pre-enrichment**

Contains the synchronization data after the CI data structure has been normalized. The data reflects what has been loaded from the HPOM web service to the RTSM.

- **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:

- **Compare File Differences**

Using a file comparison tool of your choice you can easily see what has been changed during enrichment.

- **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.



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 91 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 HPOM services 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.

Service Discovery Server Log File Configuration

Dynamic topology synchronization only: You can edit the service discovery log to change the level of detail for debugging purposes.

To edit the service discovery log, follow these steps:

- 1 In a command prompt, run the following command:
`ovconfchg -edit`
- 2 You can see the log in a Notepad window. Edit the file by adding `LOG_LEVEL=4` to the `[om.svcdiscserver]` namespace.
`LOG_LEVEL=10` returns the most detailed log output.

Troubleshooting, Common Issues, and Tips

The log files specified under [Topology Synchronization File Locations](#) on page 71 are a good starting point for troubleshooting.

The most common issues are listed in [Table 7](#). The issues apply to topology synchronization generally, unless otherwise specified.

Table 7 Troubleshooting Common Issues

Symptom		Cause	Solution
Topology synchronization fails.	Both <i>Basic</i> and <i>Dynamic</i> :	Patch OMW_00070 or higher, and OMW_00057 for HPOM for Windows 8.1x were not installed.	Install Patch OMW_00070 or higher and OMW_00057 for HPOM for Windows. See the <i>HP Business Service Management 9.00 Readme</i> for details.
	Additionally for <i>Dynamic</i> :	OMi Enablement Patch OMW-00071 for HPOM for Windows 8.1x was not installed.	Install Patch OMW_00071 for HPOM for Windows. See the <i>HP Business Service Management 9.00 Readme</i> for details, including information about any required agent hotfixes or patches.
<i>Basic</i> 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.
<i>Dynamic</i> topology synchronization fails.		Synchronization package was changed on disk, but not uploaded to the database.	Run the <code>opr-sd-tool.bat</code> command-line tool to upload changes to synchronization packages to the database (see Uploading Synchronization Packages to the Database on page 73).

Table 7 Troubleshooting Common Issues

Symptom	Cause	Solution
All of a sudden, no more node CIs are created, and the synchronization fails.	The default synchronization package was removed in the Topology Synchronization settings.	Check if the default synchronization package was removed in 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 exception in the log file.	Data dump option is not enabled.	Enable the data dump option and check if the file in the <code><HPBSM root directory>/opr/tmp/datadump/post-enrichment</code> 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, the delta detection does not delete and replace the old RTSM 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 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 instances relating to the same (changed) managed node.

Workaround: To overcome this limitation, you need to manually delete the old RTSM instance.

Event-driven Synchronization Limitation

Event changes from the HPOM discovery server are registered with the WMI Listener, and forwarded through the WMI to the RTSM. 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. As a result, this could lead to a temporary discrepancy between the status in HPOM and the RTSM. This discrepancy will be reconciled the next time a scheduled synchronization takes place (typically once in 24 hours).

15 Mapping Engine and Syntax

Mapping is the mechanism used to map services, attributes, or nodes within HPOM to CIs in the RTSM. The file format, mapping syntax, and XPath query language used to navigate through the CI data structure is described in the following sections:

- [Common Mapping File Format](#)
- [Mapping File Syntax](#)
- [XPath Navigation](#)

Common Mapping File Format



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

This example illustrates the common parts of the mapping file:

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

The components of the mapping files are described in [Mapping File Syntax](#) on page 102.

Mapping File Syntax

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

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](#) below 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. You should 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 sub-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

Operations Manager Service ID

<OMId/>

Return type: String

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

- Services: OM ID is the service ID
- Nodes: OM ID is the unique ID
- Node Groups: OM ID is the Node Group identifier

Operations Manager Type

<OMType/>

Return type: String

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

CMDB Type

<CMDBType/>

Return type: String

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

Caption

<Caption/>

Return type: String

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

OM Attribute

<OMAttribute>[Name]</OMAttribute>

Return type: String

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

CMDB Attribute

Syntax:

<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 sub-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 back slash 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 120.

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 120.

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.



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

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

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. A 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

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

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

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

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 a service tree 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.

► 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 STD `exch_spi_std_server_role` and that are below a Service with a STD `exch_spi_std_server` are assigned to the exchange context.

Type Mapping

The service mapping maps the OM 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 service type attributes of a service in HPOM and the attributes of a CI in the RTSM.

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

Mapping to String values 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>
            <DescendantCICollection>
              <StartsWith>
                <OMType>
                  <Value>Exch2k7_Role_</Value>
                </StartsWith>
              </DependencyCI>
            </To>
            <Type>deployed</Type>
          </RelationTo>
          <RootContainer>
            <DependencyCI relationType="hosted_on">
              <True/>
            </DependencyCI>
          </RootContainer>
          <RedirectMessagesOf>
            <ChildCICollection/>
          </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 [Figure 19](#).

CI Data Structure

OMAttributes	Contains a map of all original RTSM CI attributes. The key of this map is the name of the RTSM CI attribute that references the RTSM value of the RTSM CI attribute.
Caption	Represents the name of the CI to be displayed in Service Navigator. Caption has the same value as the RTSM CI attribute <code>display_label</code> .
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 <code>Children</code> . 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 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`.

Figure 19 illustrates the data structure exposed to the navigation.

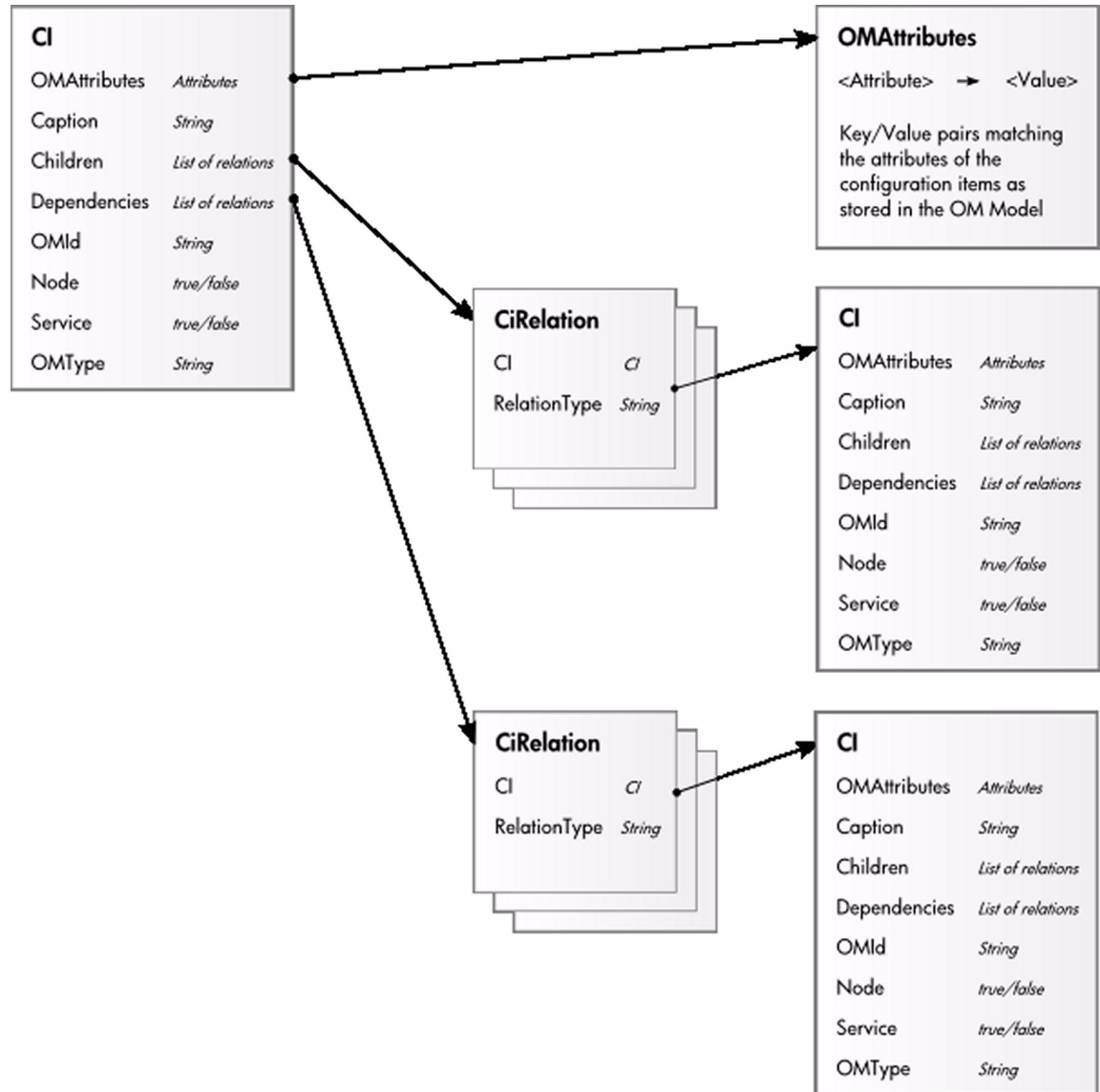


Figure 19 Data Structure Exposed to the Navigation

Example of an XPath-Navigated Data Structure

An example of an XPath-navigated data structure is shown in [Figure 20](#). 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).

[Figure 20](#) illustrates some XPath examples.

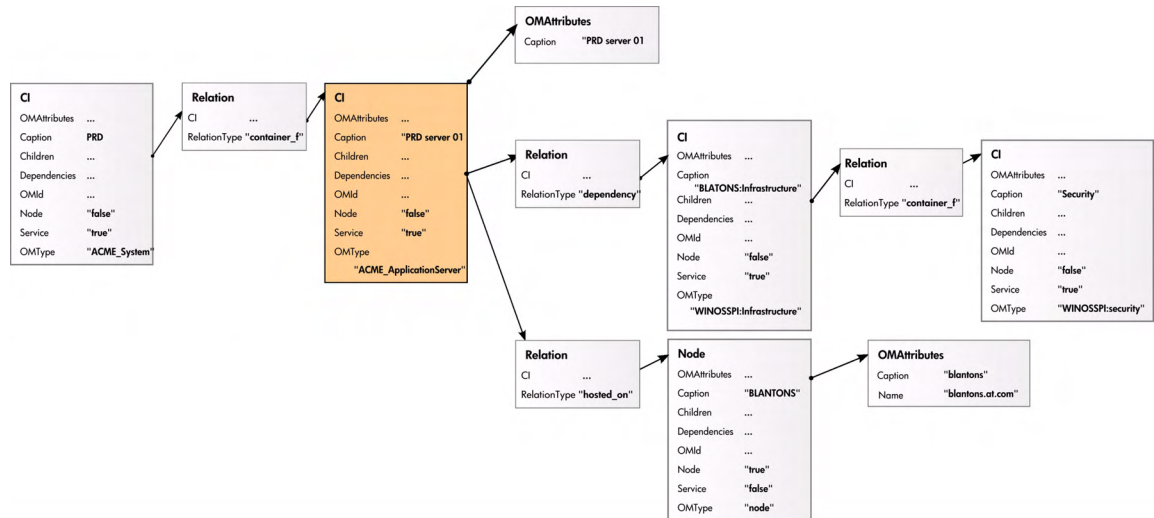


Figure 20 XPath Examples

XPath Expressions and Example Values

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

Table 8 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



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.

Section III: 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:

- [Chapter 16, Event Processing Interface](#) 127
- [Chapter 17, Scripts For Custom Actions](#) 131
- [Chapter 18, Creating EPI and Custom Action Scripts](#) 133

16 Event Processing Interface

This chapter 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.

Figure 21 provides an overview of the event processing, showing the event pipeline, and how the EPI script execution can be integrated into this process.

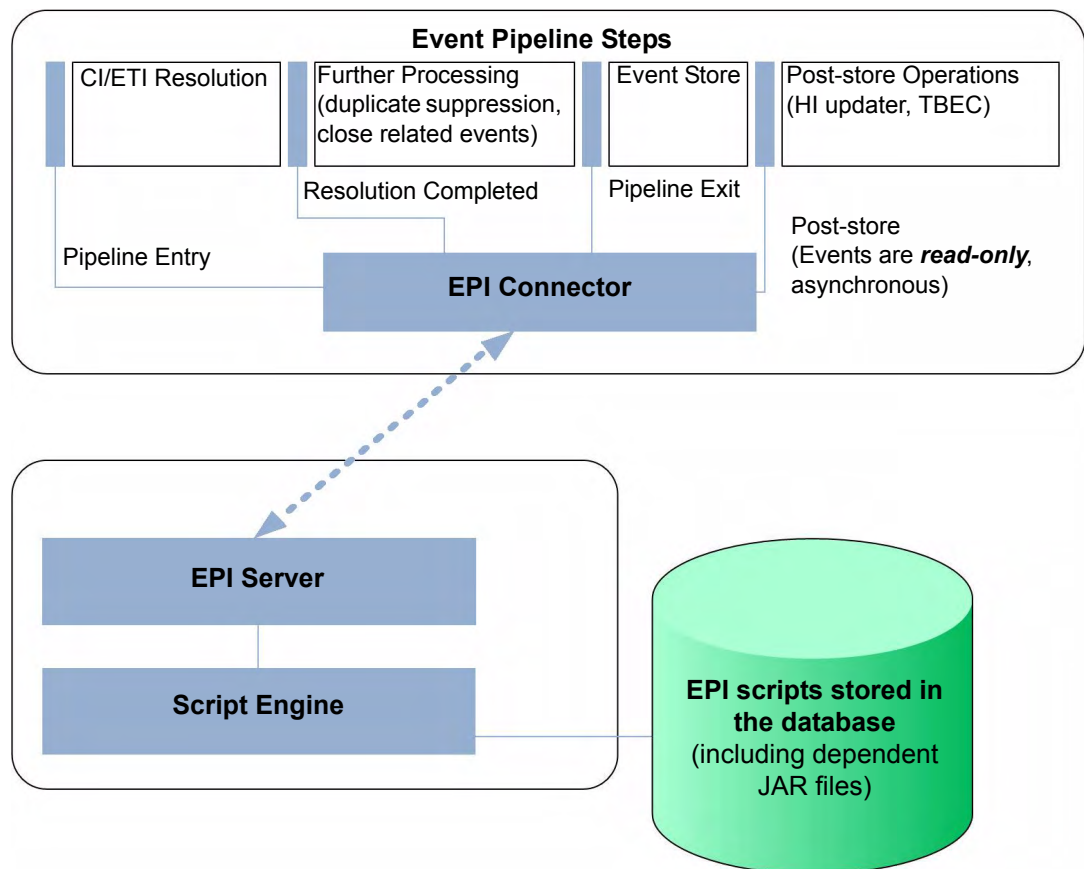


Figure 21 Event processing with pipeline entry points for scripts

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.



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 → Tune Operations Management → 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 [Chapter 18, Creating EPI and Custom Action Scripts](#) on page 133.

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 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.

Creating Scripts

EPI scripts and custom actions scripts share the same script definition format.

For more details about creating scripts, see [Chapter 18, Creating EPI and Custom Action Scripts](#) on page 133.

EPI Troubleshooting

This section contains information to help you troubleshoot the EPI.

Log Files

A good starting point for troubleshooting the EPI is to look at the log file that you can find at the following location:

```
<HPBSM root directory>/log/opr-epi-server/opr-epi-server.log
```

Debugging

To perform a debug, do the following:

- 1 Go to the following location:

```
<HPBSM root directory>/conf/core/Tools/log4j/opr-epi-server/  
opr-epi-server.properties
```

- 2 In the `opr-epi-server.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.

17 Scripts For Custom Actions

This chapter 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

- ▶ 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 HP Business Service Management 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 → Tune Operations Management → 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 [Chapter 18, Creating EPI and Custom Action Scripts](#) on page 133.

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 [Chapter 18, Creating EPI and Custom Action Scripts](#) on page 133.

18 Creating EPI and Custom Action Scripts

This chapter describes how to create EPI and custom action scripts.

This chapter is structured as follows:

- [Script Definition Attributes](#)
- [Script Definition Format](#)
- [Groovy Script API](#)
- [EPI Groovy Script Samples](#)
- [Custom Actions Groovy Script Samples](#)

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 custom action scripts, then you must implement a Groovy script that implements the methods defined by the following interface:

```
com.hp.opr.api.ws.adapter.scripting.Event
```

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 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.epi.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 is
    // disabled 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+(\w+)/, '$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.



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);
    }
}
```

Section IV: Integrating Events

This section provides integrators with information how to integrate events from other applications such as HP Network Node Manager i Software (NNMi) with HP Business Service Management (BSM) Operations Management using HP BSM Integration Adapter.

This section is structured as follows:

- [Chapter 19, HP BSM Integration Adapter](#) 143

19 HP BSM Integration Adapter

HP BSM Integration Adapter enables you to monitor event sources, and, if certain conditions apply, to forward the detected events as HP Business Service Management (BSM) events directly to the BSM Operations Management event browser. For BSM Integration Adapter to be able to convert the source events to BSM events, the event sources must make their data available as SNMP traps or in XML-formatted files.

SNMP event sources can range from simple hardware devices that send SNMP traps to sophisticated network management solutions such as HP Network Node Manager i (NNMi). (NNMi provides an out-of-the-box integration with BSM Integration Adapter, which converts NNMi incidents to SNMPv2c traps and forwards these SNMPv2c traps as BSM events to the BSM Operations Management event browser. For more information about this integration, see the *NNMi Deployment Guide*.)

BSM Integration Adapter can also monitor XML-formatted files. Because BSM Integration Adapter is agnostic to the content and syntax of the XML files, you can monitor any XML file. (If the application that you want to monitor does not log its events in XML format, consider writing a program or script that diverts and converts the application's output to XML-formatted files.)



BSM provides EMS (Enterprise Management Systems) software to facilitate the integration of data from other applications, both HP and third party. While BSM Integration Adapter is the recommended solution for integrating *events* into BSM, HP recommends that you use EMS to integrate other types of data, for example, metric data. See the *BSM Solutions and Integrations* guide for more information.

If you have the Operations Manager *i* (OMi) license, in the context of Operations Management, the BSM Operations Management event browser can receive events from HP Operations Manager (HPOM), or directly from BSM Integration Adapter. BSM Integration Adapter is the preferred method for sending events to Operations Management for new event integrations, if there is no need to use the HPOM event processing capabilities or console, or if HPOM is not part of the BSM deployment. For more information about deploying BSM, see the *BSM Deployment Guide*.

BSM Integration Adapter includes the HP Operations agent. The HP Operations agent collects and monitors the data on the event source, enriches these events with information that is meaningful to BSM users, and sends the events to BSM where they display in the Operations Management event browser. BSM Integration Adapter uses policies to configure the agent. For each policy, you decide what kinds of events to monitor, how often to monitor, what to look for in the events, and what to do if certain events are detected. [Figure](#) on page 144 shows the event flow from the event source through BSM Integration Adapter to BSM.

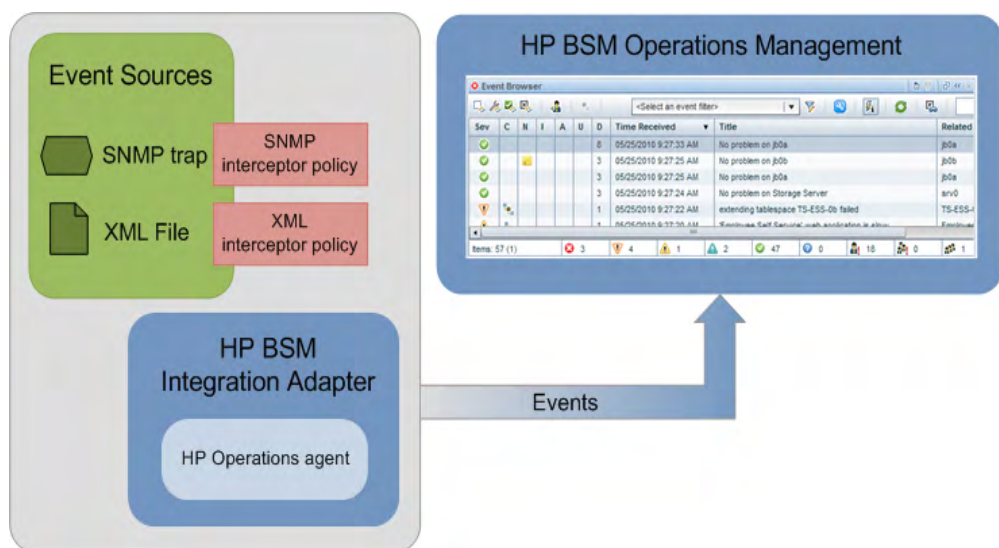


Figure 1 HP BSM Integration Adapter Conceptual Overview

HP Operations agents and BSM communicate using HTTPS, which is secure, reliable, and simplifies firewall configuration.

You typically install BSM Integration Adapter on the computer that provides the input events. For example, to integrate NNMi events into BSM, install BSM Integration Adapter on the NNMi management server.

The BSM Integration Adapter user interface is web based; you can therefore access it from anywhere using a supported web browser. Figure 2 on page 144 shows an example of the BSM Integration Adapter user interface.

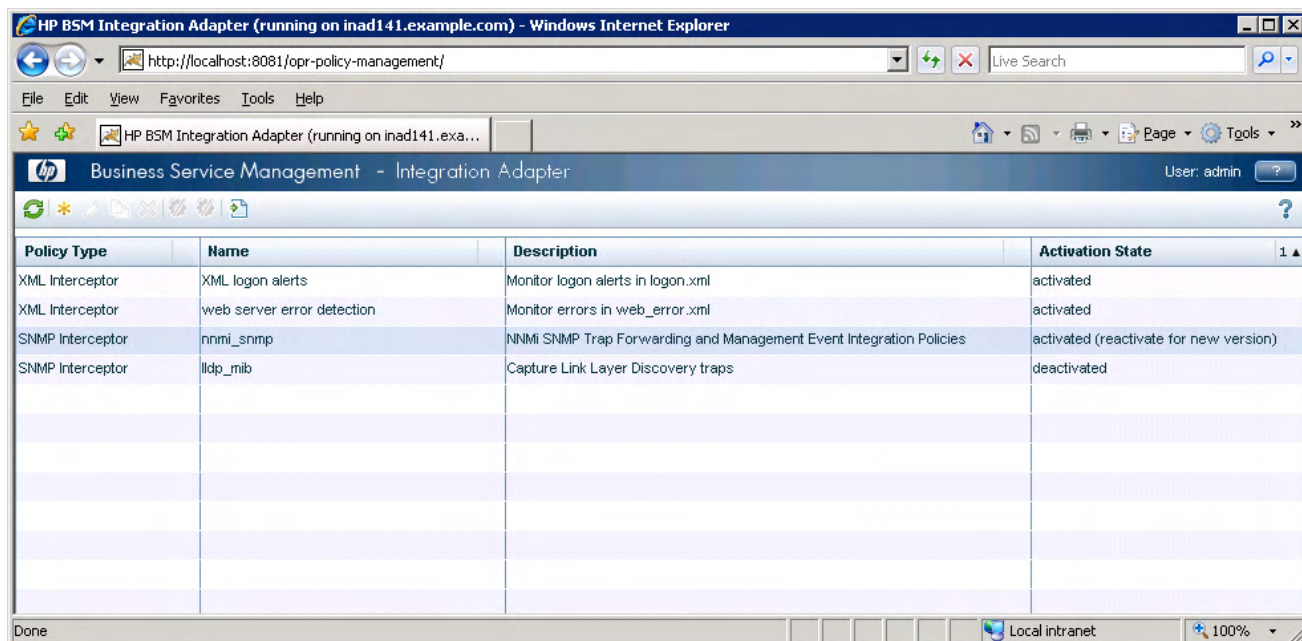


Figure 2 BSM Integration Adapter User Interface

For more information about BSM Integration Adapter, see the documentation provided with the product.

Policies

Policies are collections of configuration information that BSM Integration Adapter uses to configure the HP Operations agent to perform monitoring tasks. When you develop policies, you decide what kinds of events to monitor, how often to monitor, what to look for in the events, and what to do if certain events are detected.

BSM Integration Adapter enables you to create and edit policies to capture SNMP events and monitor XML files. You can import policies developed on other servers, for example, HP Operations Manager (HPOM) management servers, HP Network Node Manager i (NNMi) management servers, or other BSM Integration Adapter servers.



NNMi provides an SNMP interceptor policy that you can import into the BSM Integration Adapter policy repository. The policy includes an SNMPv2 trap definition for each of the management events and SNMP traps in the current NNMi configuration. For more information about the BSM Integration Adapter—NNMi integration, see the *NNMi Deployment Reference*.

Integrating Events with XML Interceptor Policies

XML interceptor policies are especially suited for integrating events from other applications. XML interceptor policies process XML files and send events to the Operations Management event browser when certain conditions apply. You can define the attributes of the BSM event based on information in the XML file. This enables you to process events generated by the applications and to convert them to BSM events.

XML interceptor policies process exactly the XML elements and attributes that you define. The XML syntax is not important to the policy, as long as the event information is embedded in XML elements and attributes.

If the application does not store its events in XML files, you may need to write a program or script that extracts the events from wherever they are stored, formats the data using XML syntax, and generates an XML file with the events. If you have control over the XML elements that are used in the XML file, choose XML elements and attributes that map to event attributes and values. This will simplify the policy.

Integrating Events with HP BSM Integration Adapter

The following list gives an overview of the tasks you must complete to integrate events into BSM using BSM Integration Adapter. For more information about each task, refer to the corresponding documentation.

Task 1: [Install and configure HP BSM Integration Adapter.](#)

Use the *HP BSM Integration Adapter Installation and Configuration Guide* to install and configure BSM Integration Adapter on the computer that serves as event source. During the configuration process you establish the connection between BSM Integration Adapter and BSM.

Task 2: [Choose the policy type.](#)

BSM Integration Adapter enables you to create and edit policies to capture SNMP events and monitor XML files. You can import policies developed on other servers, for example, HP Operations Manager (HPOM) management servers, HP Network Node Manager i (NNMi) management servers, or other BSM Integration Adapter servers.

- Create new SNMP or XML interceptor policies in BSM Integration Adapter.

You can use the BSM Integration Adapter user interface to create new SNMP and XML interceptor policies. For more information, see the BSM Integration Adapter online help.



If your event source does not provide data in XML file format, you may need to write a program or script that converts the event data to XML format. See also [Integrating Events with XML Interceptor Policies](#) on page 145.

- Import policies.

Policies that you import into the BSM Integration Adapter policy repository must support the XML-based policy exchange format. For more information about importing policies, see the BSM Integration Adapter online help.

Task 3: [Activate the policies in BSM Integration Adapter.](#)

When you create a new policy or import a policy, the policy exists in the BSM Integration Adapter policy repository but does not function yet. You must first activate the policy for it to start monitoring the corresponding event source. For more information, see the BSM Integration Adapter online help.

Task 4: [Optional. Set up configuration items in BSM.](#)

BSM tries to associate the events that it receives with a configuration item (CI) using CI resolution. This is not possible if the computer on which the event occurred is not set up as a CI in BSM. For more information about CI resolution, see “CI Resolution” in the Operations Management online help. For more information about creating CIs, see “Create CIs and Relationships in the RTSM” in the *Modeling* guide in the BSM Documentation library.

Section V: 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:

- [Chapter 20, URL Launch of the Event Browser](#) 151

20 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 chapter describes how to specify such a URL launch.

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 that 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`

- ▶ 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 [Table 1](#) on page 152.

You specify additional parameters for the URL launch in the following way:

`http://<hostname:port>/opr-console/opr-evt-browser?<set_parameters_here>`

- ▶ 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.
- ▶ 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

Table 1 on page 152 contains the parameters available for the URL launch of the Operations Management Event Browser:

Table 1 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 from into the browser.	One of: true , false	true
columns	Defines the columns to be displayed in the Event Browser.	One or more of: <Column key> (see Table 2 on page 153)	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

Table 1 Parameters for the URL Launch of the Event Browser

Parameter	Description	Possible Values	Default Values
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
sortField	Defines the column by which the Event Browser is sorted.	One of: <Column key> (see Table 2 on page 153)	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 [Table 2](#). [Table 3](#) on page 155 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.

Table 2 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

Table 2 Fixed Columns Keys to Define Which Columns are Displayed

Column Key	Explanation
eti	Event type indicator value
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

Table 3 Custom Attribute Column Key

Column Key	Explanation
<code>ca_<columnname></code>	Custom attribute column, where <code><columnname></code> 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

[Table 4](#) contains the filter parameters available for the URL launch of the Event Browser.

Table 4 Filter Parameters for the URL Launch of the Event Browser

Filter Parameter	Description	Values
<code>filter_assignment</code>	Applies an assignment filter to the Event Browser.	One or more of: <code>me</code> , <code>my_workgroups</code> , <code>others</code> , <code>nobody</code>
<code>filter_severities</code>	Applies a severity filter to the Event Browser.	One or more of: <code>unknown</code> , <code>normal</code> , <code>warning</code> , <code>minor</code> , <code>major</code> , <code>critical</code>
<code>filter_states</code>	Applies a lifecycle state filter to the Event Browser.	One or more of: <code>open</code> , <code>in_progress</code> , <code>resolved</code>

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 [Table 5](#) on page 156.

Table 5 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
```


Section VI: 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 [Chapter 24, Forwarding Events and Synchronizing Event Changes](#) on 201 .

This section is structured as follows:

- [Chapter 21, Automating Operator Functions using the Event Web Service Interface](#) . 163
- [Chapter 22, REST Web Service Command-Line Utility](#) 177
- [Chapter 23, Event Web Service Query Language](#) 185

21 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 update events using the Atom service.

► You can only modify events using the Event Web Service. You cannot create or delete an event itself.

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 151.

An integrator would typically be interested in:

- [How to Access the Event Web Service](#)
- [How to Detect New Events](#)
- [How to Detect Event Changes](#)
- [How to Modify Events](#)

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 [Table 6](#) on page 164.

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/event_list`

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 Table 6, 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/event_list/
532d3674-684f-419f-a752-b8681ee01a72`**

Table 6 List of OPR Event Services in the Service Document

OPR Event Service	URL
Annotation Service:	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list/{event}/annotation_list/{annotation}</code>
Annotations Service:	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list/{event}/annotation_list</code>
Automatic Action Service:	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list/{event}/auto_action</code>
Custom Attribute Service:	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list/{event}/custom_attribute_list/ {custom_attribute}</code>
Custom Attributes Service	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list/{event}/custom_attribute_list</code>
Event Change Service	<code>http://<bsmserver.example.com>/opr-console/rest/ event_change_list/{event_change}</code>
Event Changes Service	<code>http://<bsmserver.example.com>/opr-console/rest/ event_change_list</code>
Event Service	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list/{event}</code>
Events Service	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list</code>
History Line Service	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list/{event}/history_line_list/{history_line}</code>
History Lines Service	<code>http://<bsmserver.example.com>/opr-console/rest/ event_list/{event}/history_line_list</code>

Table 6 List of OPR Event Services in the Service Document

OPR Event Service	URL
Operator Action Service	<code>http://<bsmserver.example.com>/opr-console/rest/event_list/{event}/user_action</code>
Symptom Service	<code>http://<bsmserver.example.com>/opr-console/rest/event_list/{event}/symptom_list/{symptom}</code>
Symptoms Service	<code>http://<bsmserver.example.com>/opr-console/rest/event_list/{event}/symptom_list</code>

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 196.

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/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 190.
- **`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/event_list/?alt=atom`

- 2 You now see a list of events. Firefox and Internet Explorer represent the data in a slightly different way.

a 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 [Chapter 20, URL Launch of the Event Browser](#) on page 151.



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.

b 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 [Chapter 20, URL Launch of the Event Browser](#) on page 151.



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/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 [Chapter 23, Event Web Service Query Language](#) on page 185.

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.bat` command-line utility.



In this section, we describe a standard HTTP environment. For secure environments, use HTTPS.

Modifying Events Using a REST Client

In general, when using a REST client to update the 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/>

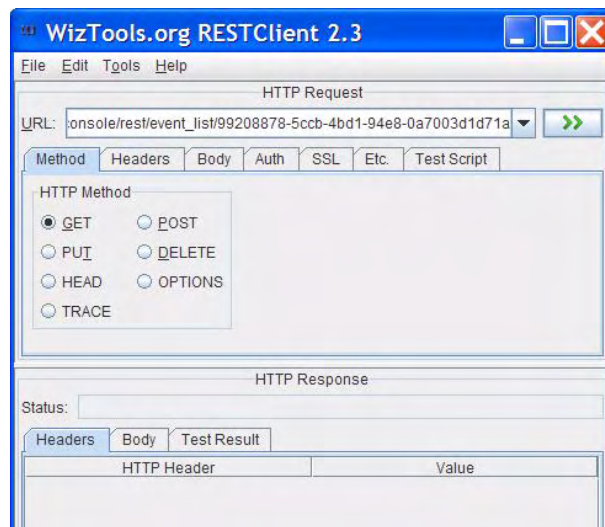
The RESTClient user interface is divided into two parts:


- 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/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.

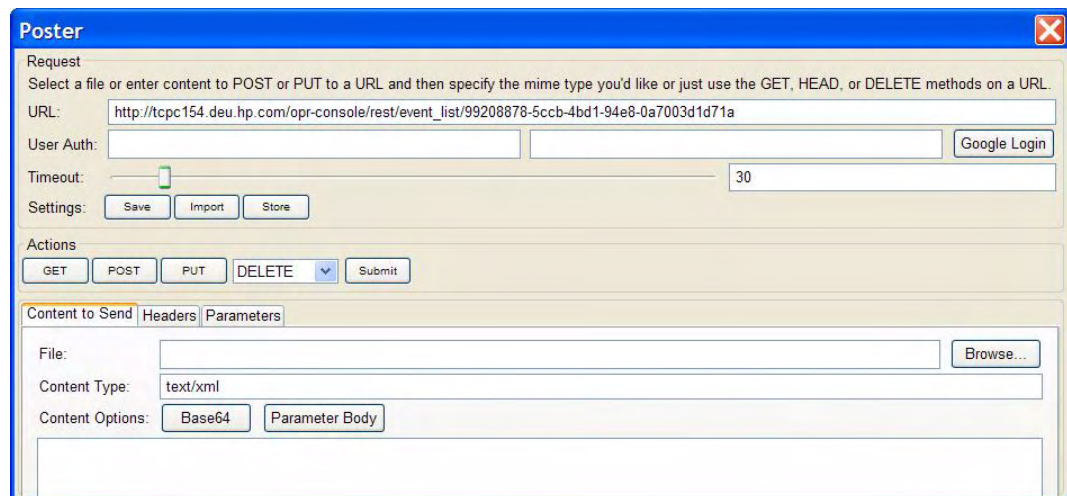
Modifying Events Using the Firefox Poster Extension

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.

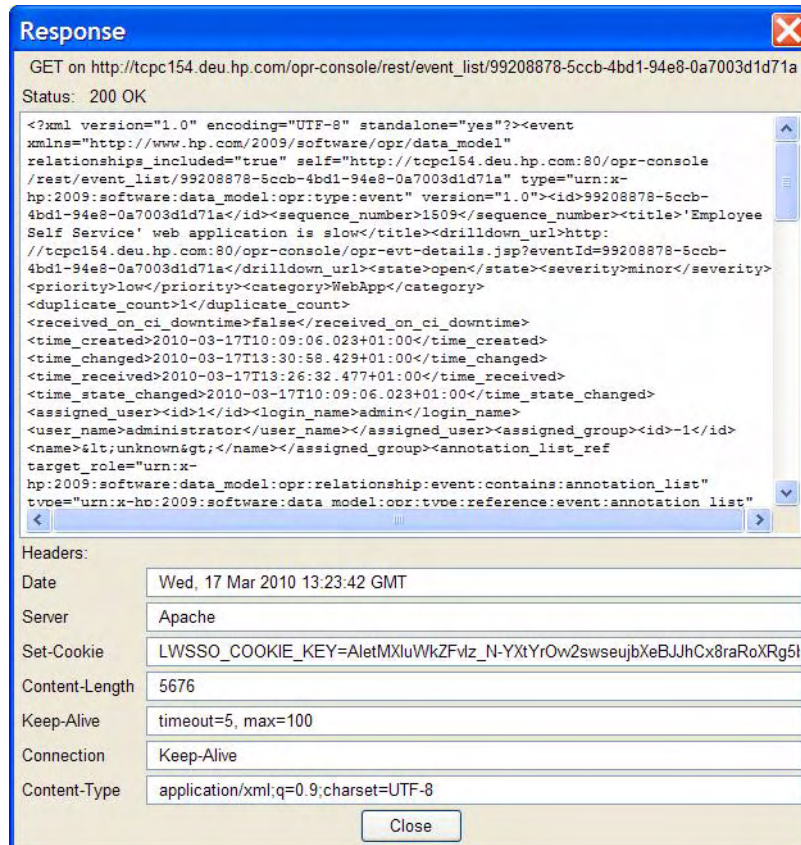
We will use an example of how to change the title and severity of an event using the Firefox Poster Extension. The steps required are as follows:

- 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/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.

➤ Not all properties can be updated. For a list of editable properties, see [Editable Properties](#) on page 194. 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.

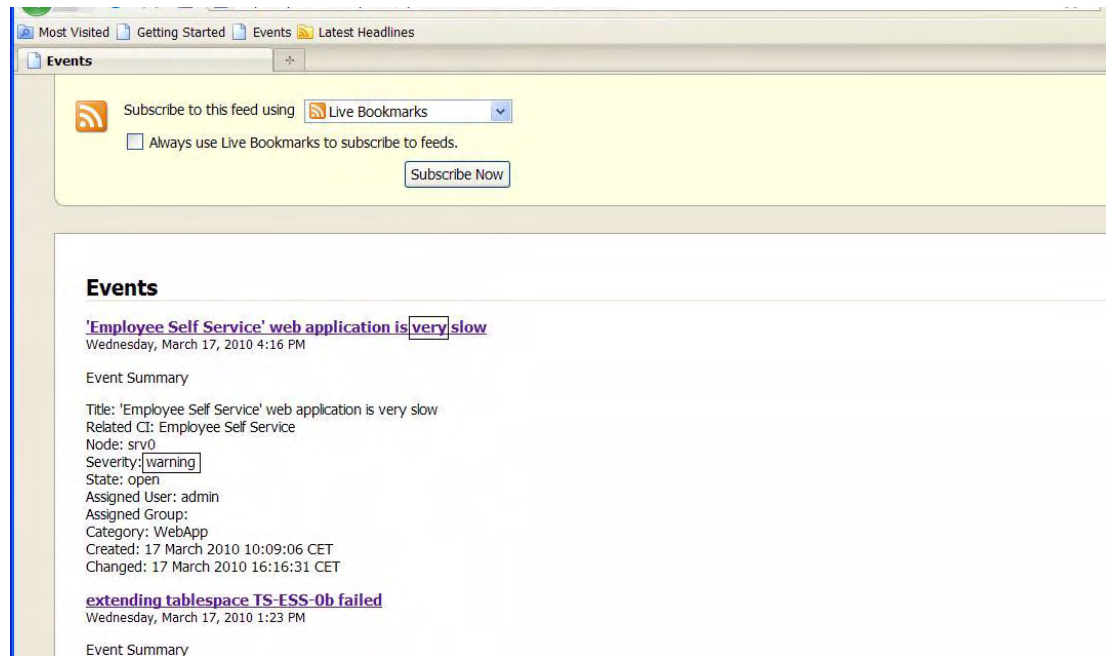
In this example, we will change the title of the event, and also the event severity, directly in the XML. For example, the event's original title was: "'Employee Self Service' web application is slow" and we want to edit the event title by inserting the word "very":

```
...<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.bat 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.bat` 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.bat` 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/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.bat` command-line utility, see [Chapter 22, REST Web Service Command-Line Utility](#) on page 177.

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 163.

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/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 that you cannot create or delete an event itself using the `RestWsUtil.bat` 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/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 [Chapter 23, Event Web Service Query Language](#) on page 185), and specifying the update in the payload of a PUT request.

For example:

**URL: http://my.host.com/opr-console/rest/event-list&query=title%20LIKE%20'%25db
down%25'**

HTTP Method: PUT

Payload: <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

Payload: <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.

22 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`

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:

```
C:\HPBSM\opr\bin>.\RestWsUtil.bat -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.bat` utility are as follows:

<code>-c, -create <in_file></code>	Create the resource in the specified XML document. HTTP POST operation.
<code>-content_type <type></code>	HTTP header content-type value to set for create and update operations. Default is <code>application/xml</code> .
<code>-d, -delete</code>	Delete the specified resource. The URL directly addresses the resource to delete. HTTP DELETE operation.
<code>-h, -help</code>	Print this message and exit.
<code>-o, -output <out_file></code>	Name of output file for text returned from the web service request. Default is <code>stdout</code> .
<code>-p, -port <port></code>	Set the port number. Default port is 80 for HTTP and 443 for HTTPS. This option cannot be specified in conjunction with the option <code>url</code> .

<code>-password <password></code>	Password for the specified user.
<code>-r, -read</code>	Read the specified resource. HTTP GET operation.
<code>-server <server></code>	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 <code>url</code> .
<code>-ssl</code>	Set the protocol to HTTPS. Default is to use HTTP. This option cannot be specified in conjunction with the option <code>url</code> .
<code>-update <in_file></code>	Update the resource with the specified XML document changes. HTTP PUT operation.
<code>-url <URL></code>	URL of the gateway server. This option cannot be specified in conjunction with the options <code>ssl</code> , <code>server</code> , or <code>port</code> .
<code>-username <login name></code>	User login name required for authentication.
<code>-v, -verbose</code>	Print verbose output.
<code>-version</code>	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.bat` utility follow.

Reading Events

Here is an example of using the `RestWsUtil.bat` command-line utility to read events and to send the results to an output file called `test.xml`.

```
.\RestWsUtil.bat -r -username admin -o test.xml -verbose
Password: *****
INFO: Read the resource located at: http://bsmserver.example.com/opr-console/rest/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.bat -r
```

```
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/event_list/
0695624b-93fa-40b1-8b0fc9b4ea07a4ec/custom_attribute_list
-username admin -verbose
Password: *****
INFO: Read the resource located at: http://localhost/opr-console/rest/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/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/event_list/0695624b-93fa-40b1-8b0f-c9b4ea07a4ec/
    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/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/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/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/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.bat -c newca.xml
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/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/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.bat -c newanno.xml
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/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/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/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.

```
C:\HPBSM\opr\bin>.\RestWsUtil.bat -update updateca.xml
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/event_list/
0695624b-93fa-40b1-8b0f-c9b4ea07a4ec/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/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/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/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/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.

```

C:\HPBSM\opr\bin>RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/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.

```
C:\HPBSM\opr\bin>RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/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.

```
C:\HPBSM\opr\bin>RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/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.

```
C:\HPBSM\opr\bin>RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/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.

```
C:\HPBSM\opr\bin>RestWsUtil -update update.xml -username admin
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/
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.bat -d
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/event_list/
0695624b-93fa-40b18b0fc9b4ea07a4ec/custom_attribute_list/MyNewCa -username admin -verbose
Password: *****
INFO: Deleting resource located at: http://localhost/opr-console/rest/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/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.bat -d
-url http://<fully qualified domain name of BSM gateway server>/opr-console/rest/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/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/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/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.
```


23 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 page 186.
query	Parameter for filtering by event attributes. For details, see Filtering by Event Attributes: query on page 187.
start_index	Paging query parameter used to define the item from which to start the query. For details, see Paging on page 188.
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 188.

Query Parameter	Description
order_by	Ordering parameter used to specify that the response feed should be ordered by the indicated field. For details, see Ordering on page 189.
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 189.
include_closed	Parameter used to specify whether closed events are included when querying the Events Service (<code>event_list</code>). Default value: <code>false</code> For details, see Data Inclusion on page 189.
include_relationships	Parameter used to specify whether relationships are included when querying the Events Service (<code>event_list</code>). Default value: <code>true</code> For details, see Data Inclusion on page 189.

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.

[Table 7](#) on page 191 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 is required. 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 [Table 8](#) on page 193.

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_created>2009-10-19T17:06:54.453%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/
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/
event_list?alt=atom&query=severity='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 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/
event_list?query=assigned_user='admin'%20AND%20title='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/
event_list?query=assigned_user='admin' AND
related_ci_info[ci_ref[target_id=XXX]]**

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/  
event_list?query=assigned_user='admin' AND (title='My Title' OR  
lifecycle_state='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/  
event_list?alt=atom&query=severity='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/
event_list?alt=atom&query=severity='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/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=xml`), which is supported by most resources (see [How to Detect New Events](#) on page 165).

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 and Supported Operators

Table 7 lists the available query filter criteria properties and supported operators.

Table 7 Query Filter Criteria Properties and Supported Operators

Property	Supported Operators	Supports order_by	Default for order_direction	Comment
application	=, !=, LIKE	Yes	ascending	
assigned_group	=, !=	assigned_group [id] only	ascending	group name
assigned_user	=, !=	assigned_user [id] only	ascending	user login name
category	=, !=, LIKE	Yes	ascending	
custom_attribute_list[<CA name>=<value>]	=, !=, LIKE			ca is an alias available for 'custom_attribute_list'
description	=, !=, LIKE	Yes	ascending	
eti_hint	=, !=, LIKE			
eti_value_id	=, !=, LIKE			
id	=			
key	=, !=, LIKE			
node_hints[hint=<value>]	=, !=, LIKE			See Complex Attributes on page 192 for further details
node_hints[node[core_id=<value>]]	=, !=			
node_hints[node[dns_name=<value>]]	=, !=			
node_hints[node[ip_address=<value>]]	=, !=			
node_ref[target_global_id=<value>]]	=, !=			
node_ref[target_id=<value>]]	=, !=			
object	=, !=, LIKE	Yes	ascending	
om_user	=, !=, LIKE	Yes	ascending	
original_data	=, !=, LIKE			
priority	=, <, >, >=, <=, !=	Yes	ascending	
related_ci[target_global_id (!)= 'XXX']	=, !=			See Complex Attributes on page 192 for further details
related_ci[target_id (!)= '<value>']	=, !=			
related_ci_hints[hint (!)= '<value>']	=, !=, LIKE			
sequence_number	=, <, >, >=, <=, !=	Yes	descending	
severity	=, !=	Yes	ascending	

Table 7 Query Filter Criteria Properties and Supported Operators

Property	Supported Operators	Supports order_by	Default for order_direction	Comment
source_ci[target_global_id='<value>']	=, !=			See Complex Attributes on page 192 for further details
source_ci[target_id='<value>']	=, !=			
source_ci_hints[hint='<value>']	=, !=, LIKE			
state	=, !=	Yes	ascending	To include closed events, you need to specify <code>include_closed=true</code>
sub_category	=, !=, LIKE	Yes	ascending	
time_changed	<, >	Yes	descending	Requires a value in XML Schema dateTime format
time_created	<, >	Yes	descending	
time_received	<, >	Yes	descending	
time_state_changed	<, >	Yes	descending	
title	=, !=, LIKE	Yes	ascending	
type	=, !=, LIKE	Yes	ascending	

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 is an example of a URL call containing a complex attribute:

```
http://<bsmserver.example.com>/opr-console/rest/  
event_list?query=related_ci_info[ci_ref[target_id='XXX']]
```



When specifying complex attributes, the square bracket characters ([]) must be escaped, for example, like this:

```
query=ca%5Btest_ca%5D=abc
```

[Table 8](#) on page 193 lists the characters that must be escaped in URLs.

Table 8 URL Escape Codes for Characters that Must Be Escaped

Character	Escape Codes
SPACE	%20
<	%3C
>	%3E
#	%23
%	%25
+	%2B
{	%7B
}	%7D
	%7C
\	%5C
^	%5E
~	%7E
[%5B
]	%5D
`	%60
;	%3B
/	%2F
?	%3F
:	%3A
@	%40
=	%3D
&	%26
\$	%24

Editable Properties

Event properties that you can edit within the Event Web Service are listed in [Table 9](#).

▶ 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 197 for access information for the Java API Documentation.

Table 9 List of Editable Event Properties

Name of Tag	Type	Description	Comment
title	String	Title of the event	
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		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: <code><name>User</name></code>
assigned_group	Group	User group assigned to the event	Example: <code><name>Users</name></code>
cause	ID	Identifier of the cause	
control_transferred_to	ID or name	ID or name of the connected server	Set in the <code>control_transferred_to</code> 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

Table 9 List of Editable Event Properties

Name of Tag	Type	Description	Comment
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 [Table 10](#). 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`.

Table 10 Recorded Property Changes

Name	ChangeType
annotation	OprAnnotationPropertyChange
application	OprStringPropertyChange
assigned_group	OprGroupPropertyChange
assigned_user	OprUserPropertyChange
category	OprStringPropertyChange
cause	OprStringPropertyChange
control_transfer_to	OprStringPropertyChange
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

Table 10 Recorded Property Changes

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 186 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`
`<HPBSM root directory>/opr/api/schema/OprDataModel.xsd`

Section VII: 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:

- [Chapter 24, Forwarding Events and Synchronizing Event Changes](#) 201
- [Chapter 25, Integrating External Event Processes Using Groovy Scripts](#) 207
- [Chapter 26, Event Synchronization Web Service Interface](#) 221
- [Chapter 27, Integrating External Event Processes: Frequently Asked Questions](#) 237
- [Chapter 28, Service Manager Integration](#) 249

24 Forwarding Events and Synchronizing Event Changes

This chapter 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

Figure 3 shows an overview of the event forwarding architecture, and the various routes that the forwarded event can take to reach the incident manager.

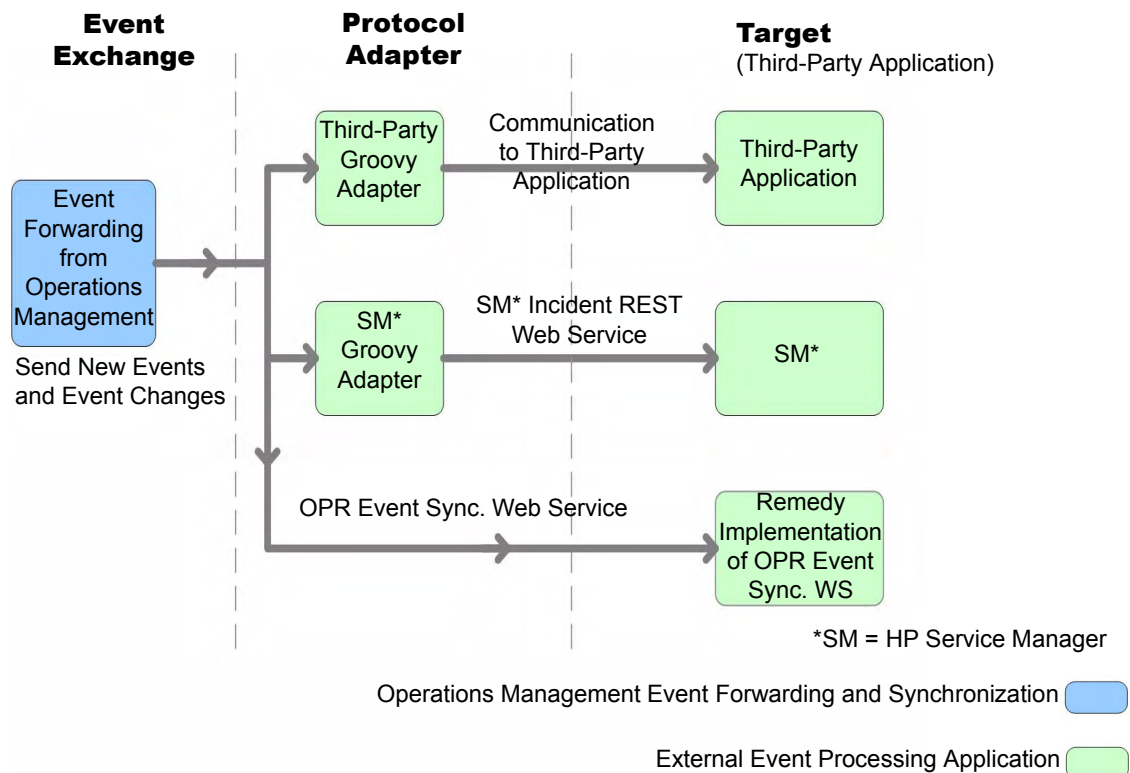


Figure 3 Overview of event forwarding architecture

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 Forwarding Rules 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 207)

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 [Chapter 25, Integrating External Event Processes Using Groovy Scripts](#) on page 207.

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 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.
- **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 [Chapter 26, Event Synchronization Web Service Interface](#) on page 221.

Receiving Event Changes Back from an External Event Process

Figure 4 shows an overview of how the target application synchronizes changes back to the OPR Event Synchronization Web Service.

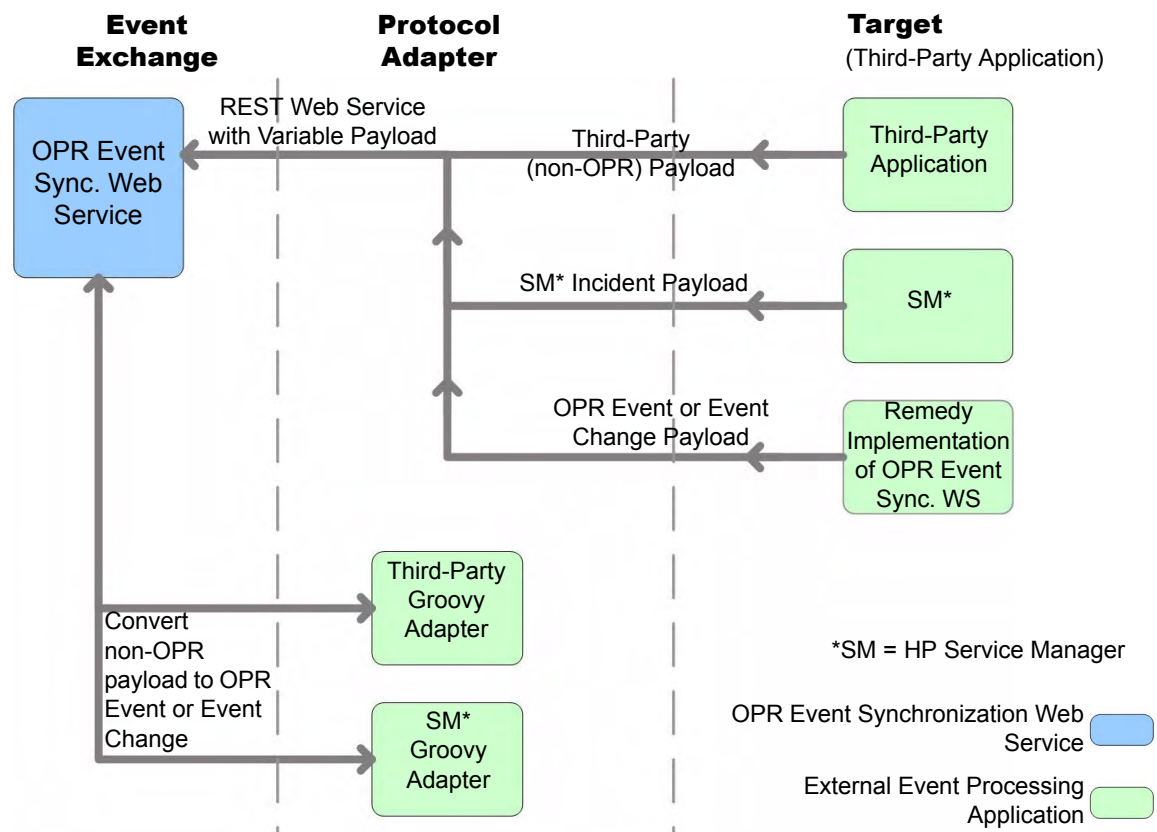


Figure 4 Receiving event changes back from the target application

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.

For details about how to use the Event Synchronization Web Service, see [Chapter 26, Event Synchronization Web Service Interface](#) on page 221.

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.

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 [Chapter 20, URL Launch of the Event Browser](#) on page 151.

25 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 chapter 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 241.

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 HP 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.

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 → Tune Operations Management → 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 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.

► 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, 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 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, do the following:

- a Make sure that the Supports 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, 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 Forwarding Rules manager in the Operations Management user interface:
Admin → Operations Management → Tune Operations Management → Forwarding Rules
- 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 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 [Configure a Connected Server using the Logfile Adapter](#) on page 207. 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. In the Forwarding Type field, select Synchronize and Transfer Control.

- 9 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 page 210.

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 page 210).

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 than Java, you may need to use the OPR Event Schema to generate classes to marshal and unmarshal 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

You must store event integration Groovy scripts in the following installation location:

```
<HPBSM root directory>/conf/opr/integration/<directory>
```

Where:

<directory> at the end of the above path denotes the adapter type that is displayed in the Connected Servers manager. All Groovy scripts for that adapter type must be located in the appropriate directory. The out-of-the-box scripts can be found at the following locations:

Logfile Adapter, type sample:

```
<HPBSM root directory>/conf/opr/integration/sample/LogfileAdapter.groovy
```

Service Manager Adapter, type sm:

```
<HPBSM root directory>/conf/opr/integration/sm/  
ServiceManagerAdapter.groovy
```

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 [Table 11](#).

Table 11 Properties for init() Method

Property	Description
Read Properties	
<code>installDir</code>	BSM Operations Management root directory.
<code>logger</code>	Type <code>org.apache.commons.logging.Log</code> . Used to access the logs.
<code>connectedServerId</code>	ID of the target connected server.
<code>connectedServerName</code>	Name of the target connected server.
<code>connectedServerDisplayName</code>	Display Name of the target connected server.
<code>node</code>	DNS name of the target connected server.
<code>port</code>	Type Integer. Port number of the web service, if any, for the target connected server.
<code>nodeSsl</code>	Type Boolean. Indicates if the target connected server web service requires SSL.
<code>drilldownNode</code>	DNS name of the target connected server where drill-down may be done.
<code>drilldownPort</code>	Type Integer. Drill-down port, if any, for the drilldown service on the target connected server drill-down node.
<code>drilldownNodeSsl</code>	Type Boolean. Indicates if the drill-down service requires SSL.
<code>maxTimeout</code>	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.

Table 12 lists the properties for the `ping()` method:

Table 12 Properties for ping() Method

Property	Description
Read Properties	
<code>installDir</code>	BSM Operations Management root directory.
<code>logger</code>	Type <code>org.apache.commons.logging.Log</code> . Used to access the logs.
<code>connectedServerName</code>	Name of the target connected server.
<code>connectedServerDisplayName</code>	Display Name of the target connected server.
<code>node</code>	DNS name of the target connected server.
<code>port</code>	Type Integer. Port number of the web service, if any, for the target connected server.
<code>nodeSsl</code>	Type Boolean. Indicates if the target connected server web service requires SSL.
<code>credentials</code>	Type <code>java.net.PasswordAuthentication</code> Credentials, if any, for the target connected server.
<code>locale</code>	Type: Locale The desired locale of the properties that are set.
Write Properties	
<code>outputDetail</code>	Detailed results of the ping operation. This is displayed in the Connected Servers configuration dialog, 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 [Table 13](#):

Table 13 Properties for forwardEvent() Method

Property	Description
Read Properties	
credentials	Type: java.net.PasswordAuthentication Credentials, if any, for the target connected server.
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.ci.OprForwardingInfo Holds information regarding the type of forwarding, that is: Notify, Notify and Update, Synchronize, or Synchronize and Transfer Control.
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	
OprEvent getEvent(final String id, final boolean 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.
Write Properties	
drilldownUrlPath	<i>Optional.</i> 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.

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 [Table 14](#):

Table 14 Properties for forwardChange() Method

Property	Description
Read Properties	
<code>changes</code>	Type: <code>com.hp.opr.api.ws.model.event.OprEventChange</code> The event changes to be forwarded.
<code>credentials</code>	Type: <code>java.net.PasswordAuthentication</code> Credentials, if any, for the target connected server.
<code>externalRefId</code>	Type: String The ID of the external event that should be updated.
<code>info</code>	Type: <code>com.hp.opr.api.ws.model.event.ci.OprForwardingInfo</code> Holds information regarding the type of forwarding, that is: Notify, Notify and Update, Synchronize, or Synchronize and Transfer Control.
<code>event</code>	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.
<code>causeExternalRefId</code>	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	
<code>OprEvent getEvent(final String id, final boolean includeRefs)</code>	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 <code>includeRefs</code> to true, otherwise false. See Event WS for more details on this flag.
Write Properties	
<code>None</code>	There are no write properties for this method.

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.



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 [Table 15](#):

Table 15 Properties for receiveChange() Method

Property	Description
Read Properties	
event	Type: <code>OprEvent</code> 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.
externalEventChange	Type: <code>String</code> 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.
info	Type: <code>com.hp.opr.api.ws.model.event.ci.OprForwardingInfo</code> Holds information regarding the type of forwarding, that is: Notify , Notify and Update , Synchronize , or Synchronize and Transfer Control .
locale	Type: <code>Locale</code> The desired locale of the web service caller.
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	
<code>OprEvent getEvent(final String id, final boolean includeRefs)</code>	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 <code>includeRefs</code> to true, otherwise false. See Event WS for more details on this flag.

Table 15 Properties for receiveChange() Method

Property	Description
Write Properties	
eventId	Must be set by the Groovy script adapter in cases where the connected server did a POST to the event_change web service.
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.

Additional Methods

Additional methods are provided for transferring the control of an event, annotations, custom attributes, and for accessing the HTTP requests and responses.

Control Transfer

The methods available for transferring control of an event are listed here.

`requestControl(def args)`

The `requestControl(def args)` 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(def args)`

The `returnControl(def args)` 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 the value of the specified HTTP header. 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. Parameters are of type String.

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 Table 16:

Table 16 Properties for `getExternalEvent()` Method

Property	Description
Read Properties	
<code>externalRefId</code>	The external reference ID, identifying the external event to get.
<code>credentials</code>	Type: <code>java.net.PasswordAuthentication</code> Credentials, if any, for the target connected server.
<code>locale</code>	Type: Locale The desired locale of the properties that are set.
Write Properties (All are of type String)	
<code>title</code>	<i>Optional.</i> Title of the external event.
<code>description</code>	<i>Optional.</i> Description of the external event.
<code>assignedUser</code>	<i>Optional.</i> User assigned to the external event.
<code>assignedGroup</code>	<i>Optional.</i> User group assigned to the external event.
<code>severity</code>	<i>Optional.</i> Severity of the event.
<code>priority</code>	<i>Optional.</i> Priority of the event.
<code>state</code>	<i>Optional.</i> 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 the Java Security Manager:

- access to the `System.exit()` call
- `java.awt.AWTPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/java/awt/AWTPermission.html`
- `java.io.SerializablePermission` for all targets:
`http://java.sun.com/javase/6/docs/api/java/io/SerializablePermission.html`
- `java.lang.management.ManagementPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/java/lang/management/ManagementPermission.html`
- `java.net.NetPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/java/net/NetPermission.html`
- `java.sql.SQLPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/java/sql/SQLPermission.html`
- `java.util.logging.LoggingPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/java/util/logging/LoggingPermission.html`
- `javax.management.MBeanServerPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/javax/management/MBeanServerPermission.html`
- `javax.management.MBeanTrustPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/javax/management/MBeanTrustPermission.html`
- `javax.management.remote.SubjectDelegationPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/javax/management/remote/SubjectDelegationPermission.html`
- `javax.net.ssl.SSLPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/javax/net/ssl/SSLPermission.html`
- `javax.security.auth.AuthPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/javax/security/auth/AuthPermission.html`
- `javax.security.auth.kerberos.DelegationPermission` for all targets:
`http://java.sun.com/javase/6/docs/api/javax/security/auth/kerberos/DelegationPermission.html`
- `java.security.SecurityPermission` for all targets *except* target `getProperty`:
`http://java.sun.com/javase/6/docs/api/java/security/SecurityPermission.html`
- `java.lang.RuntimePermission` only for target `exitVM`:
`http://java.sun.com/javase/6/docs/api/java/lang/RuntimePermission.html`

26 Event Synchronization Web Service Interface

This chapter describes the Event Synchronization Web Service and how to use it to integrate with external event processes.

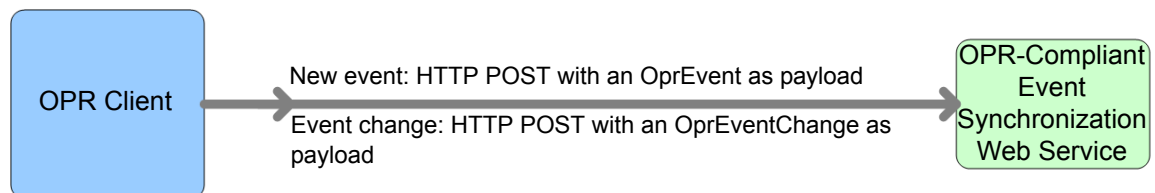
Overview

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 any changes 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 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.

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 expected response payload is the external event in the form of an `OprEvent`.

Web Service Ping Request

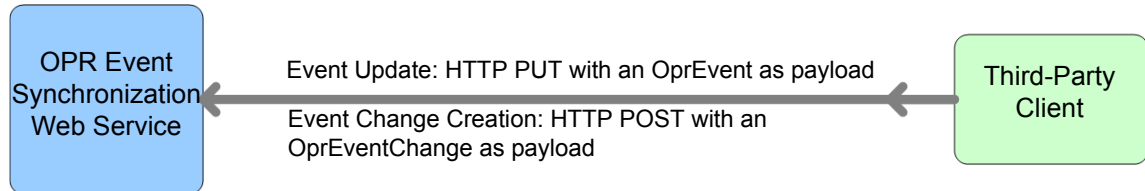
You can send a ping request to a third-party application. When sending a ping request to a third-party application, the following data is relevant:

- 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 an OPR event or event change object, as there is no Groovy script involved in this configuration to process the payload.

You can submit event changes with an event update, or with an event change creation. As a response payload, the Event Synchronization Web Service expects either an updated event, or an event change 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/synchronization/event/<event_id>`**
- Ping request (from third-party client): HTTP HEAD to base URL
- Expected Payload:
 - 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.

Event Change Creation

In the case of event changes, the following data is relevant:

- HTTP Method: POST
- Base URL: **`http://<bsmserver.example.com>/rest/synchronization/event_change/<event_id>`**
- Ping request (from third-party client): HTTP HEAD to base URL
- Expected Payload:
 - With a Groovy Script configured for connected server, the Groovy script defines the payload.
 - Without a Groovy Script configured for the connected server, the payload must be an `OprEventChange` object.

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
- URL: **`http://<gatewayhost>/opr-gateway/rest/synchronization`**

- Expected Payload: None

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 [Chapter 21, Automating Operator Functions using the Event Web Service Interface](#)), 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

Table 17

Attribute Name	Supported Operations		
	Update	Create	Delete
annotation	Yes	Yes	
assigned_user	Yes		set to null
assigned_group	Yes		set to null
cause	Yes		set to null
custom attribute	Yes	Yes	
description	Yes	Yes	set to null
priority	Yes		
severity	Yes		
solution	Yes		set to null

Table 17

Attribute Name	Supported Operations		
	Update	Create	Delete
state	Yes		
title	Yes		set to null
control_transferred_to	Yes		set to null

Event Update: Logfile Adapter Examples

► 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 226).

You can send test payloads to the Event Synchronization Web Service using the REST web service command-line utility `RestWsUtil.bat`, 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.bat` command-line utility, see [Chapter 22, REST Web Service Command-Line Utility](#) on page 177.

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:

```
C:\<HPBSM root directory>\opr\bin>.\RestWsUtil.bat -update update.xml
-url http://<bsmserver.example.com>/opr-gateway/rest/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.bat` command-line utility.

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 226).

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/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:

```
C:\HPBSM\opr\bin>.\RestWsUtil.bat -create create.xml
-url http://<bsmserver.example.com>/opr-gateway/rest/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.bat` 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:

```
<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>
```


27 Integrating External Event Processes: Frequently Asked Questions

This chapter 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 chapter is structured as follows:

Getting Started

- Is there any documentation available?
- Is there any Javadoc for the API?
- Is there a schema available for the `OprEvent` and `OprEventChange` object?
- Will I need a schema?
- Is there a Web Services Description Language (WSDL) available?
- Where can I find out more about REST?
- Are there any good toolkits to use for creating a REST client or server?
- Are there JAXB annotated classes already available for `OprEvent` and `OprEventChange`?
- Are there any sample implementations?
- How do I configure and use the Logfile Adapter sample?
- 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?
- I wrote a Groovy integration script. How do I install and test it?
- I automatically forwarded an event to my external server, but I cannot see the External Info tab in the Event Browser. What happened?
- Can I forward an event to more than one external system?
- I transferred control to an external system. Can I take it back?
- I transferred control to an external system. Can that system give it back?

Groovy Scripts and Programming

- Where can I find out more about Groovy programming?
- I see in the Logfile Adapter sample that the "?" character is sometimes used. What does it mean?
- The payload being sent by my connected server is XML. How can I parse it from the Groovy script?
- I want to map my external process severity to an Operations Management event severity. Is there an easy way to do that in Groovy?

Integration Script Methods

- What script methods must I implement?
- Which script methods are optional?
- Which script methods are optional?
- Must my class implement the `EventProcessAdapter` interface?
- When do `init()` and `destroy()` get called?

- What properties are available in the `init()` method argument?
- What properties are available in the `forwardEvent()` method argument?
- What properties are available in the `forwardChange()` method argument?
- What properties are available in the `receiveChange()` method argument?
- When processing `receiveChange()`, I would like to send back a particular response to the web service caller. How can I do that?
- What properties are available in the `getExternalEvent()` method argument?

Event Properties

- What event properties exist?
- I want to see what is the related CI for this event. How can I get this information?
- An `OprConfigurationItem` object does not define all key properties for all CIs. How do I get the other key properties?
- What are the possible class types that can be returned in the `JAXBElements` returned from `OprConfigurationItem.getAny()`?
- `OprConfigurationItem.getAny()` returns multiple objects with the same name. How can this happen?

Troubleshooting

- I configured a connected server, but it does not show up in the Event Browser context menu item "Transfer Control to". Why?
- I have forwarded an event to my external connected server. How can I tell if my script was called?
- I have forwarded an event to my external connected server. How can I tell if my script was called?

Logging

- Where can I view the log file for script execution?
- How can I change the logging level?
- How can I log from my script?

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 [Chapter 24, Forwarding Events and Synchronizing Event Changes](#) on page 201.
- For information about the Event Synchronization Web Service, see [Chapter 26, Event Synchronization Web Service Interface](#) on page 221.
- For information about integrations using Groovy scripts, see [Chapter 25, Integrating External Event Processes Using Groovy Scripts](#) on page 207.

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
- **Supports 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 Forwarding Rules 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 enable **Supports 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.transferControlTo = null
```

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 212.

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 [Table 11](#) on page 212.

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 initialized. 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 [Table 12](#) on page 213.

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 [Table 13](#) on page 214.

Example:

```
def forwardEvent(def args) {
    def timestamp = new Date()
    def extId = "urn:uuid:${args.event.getId()}"

    def msg = "### ${timestamp.toString()}: forwardEvent() called ###\n"
    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.e
vent.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 [Table 14](#) on page 215.

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 [Table 15](#) on page 216.

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.getExternalEventChange().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 web service caller.

What properties are available in the `getExternalEvent()` method argument?

For available properties for the `getExternalEvent()` method, see [Table 16](#) on page 219.

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/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?

The other key properties are added to the "any" list. Since they are only known at runtime, they are added to the list of "any" objects. They are a list of `JAXBElement` objects. Try the following sample Groovy code:

```
event.relatedCi.configurationItem.any.each { property ->
    def name = property.name.localPart
    def value = property.value
    def classType = property.declaredType

    logger.debug("Property: name=${name} value=${value} classType=${classType}")
}
```

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 sections 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 sections 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 file:

```
<HPBSM root directory>/conf/core/Tools/log4j/wde/  
opr-event-sync-adapter.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)
```


28 Service Manager Integration

This chapter 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](#)
- [Configure an Event Forwarding Rule](#)
- [Configure HP Service Manager Server](#)

Configure the HP Service Manager Server as a Connected Server

Synchronizing events and event changes between Operations Management and HP Service Manager depends on a server hosting Operations Management forwarding events to HP Service Manager, with these events and event changes being synchronized back from Service Manager. 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 → Tune Operations Management → 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.

► 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 Groovy Script Adapter**.
 - b In the External Event Processing Type field, select **sm**.
 - c In the Groovy Script File Name field, select **ServiceManagerAdapter.groovy**.
(Leave the Groovy Classpath field blank, as in this case, no external resources are required.)
 - d Click **Next**.
- 7 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 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.
 - b In the User Name field, enter the user name for the Integration User you set up previously. In this example, this is **falcon**.
 - c In the Password field, enter the password for the user you just specified, in this example **admin**. Repeat the password entry in the Password (Repeat) field.
 - d 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.
 - e If you do not want to use secure HTTP, make sure that the Use secure HTTP checkbox is *not* checked.
 - f 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` 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.

g Test the connection.

h Click **Next**.

- 8 If, in addition to purely forwarding events to HP Service Manager, 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 event drill-down.

➤ To enable event 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**.

- 9 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, enter a password that the HP Service Manager server requires to connect to the server hosting Operations Management, **Myqwer1_** in this example.

➤ 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](#) on page 249.

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 → Tune Operations Management → Forwarding Rules
- 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:

- a Either Select an existing filter, and then go to [step 8](#).
- b Or Create a new filter, as described from [step 5](#) to [step 7](#).

- 5 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, **FilterCritical**.

Uncheck the checkboxes for all severity levels except for the severity Critical.

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 target connected server you configured in the section [Configure the HP Service Manager Server as a Connected Server](#) on page 249. 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 either **Synchronize** or **Synchronize and Transfer Control**.

➤ **Notify** and **Notify and Update** are not supported forwarding types for HP Service Manager, so do not select either of these forwarding types. If you do select an unsupported forwarding type, an error is generated for any operation in HP Service Manager on the forwarded event.

For an explanation of the forwarding types, see the section entitled [Forwarding Events and Event Changes to an External Event Process](#) on page 201.

- 9 Click **OK**.

Configure URL Launch of Event Browser from HP Service Manager

Before an operator is 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 operator must be set up as a valid user in Operations Management.

Set up the operator in Operations Management with same user name that is used by the HP Service Manager operator to log on to HP Service Manager and to perform the URL call.

In this example, the valid user name to set up in Operations Management is **falcon**.

- ▶ Without this valid user name, an attempt to perform a URL launch of the Operations Management Event Browser from the calling application results in an empty browser window.

See also [Performing a URL Launch of the Event Browser from an External Application](#) on page 205.

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/synchronization/event/	General
username	Service_Manager_1 (This is the name of the HP Service Manager target server you configured in step 3 in the section Configure the HP Service Manager Server as a Connected Server on page 249.)	Header
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 (see [step 9](#) on page 251). 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**.

 Ensure that the rule is active. To make the rule active, select the rule and click **Enable**.

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 DRILLDOWN_ROOT_PATH='<web path to HP  
Service Manager>/  
index.do?ctx=docEngine&file=probsummary&query=number%3D';
```

Where **<web path to HP Service Manager>** is the path to the HP Service Manager login.

- 4 Change the value of **<web path to HP Service Manager>** 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 (refer to [step 8](#) on page 251).

Here is an example of how the drill-down URL looks:

```
http://smsserver.example.com/SM920/  
index.do?ctx=docEngine&file=probsummary&query=number%3D
```

So in this example, the value of **<FQDNS of HP Service Manager web tier server>** is `smsserver.example.com`, and the value of **<web path to HP Service Manager>** is `SM920`.

- 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.

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:

```
C:\HPBSM\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 **External Info** 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. There are two ways to do this:
 - a If the event drill-down connection is configured correctly, do the following:
 - Click the **Edit** button.
 - This opens a browser window that takes you directly to the incident in the Incident Details in HP Service Manager.
 - b If the event drill-down connection is not configured, do the following:
 - In the External Info tab in the Operations Management Event Browser, copy or note the incident ID from the External Id field.
 - In the HP Service Manager user interface, navigate to:
Incident Management → Search Incidents
 - Paste or enter the incident ID in the Incident Id field.
 - 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 on 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 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 External Info 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.



In multi-server environments, make sure that the Groovy script files are updated on *all* gateway servers.

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
```

Index

A

- ACID principle, scripts, 129
- ACME
 - content pack, 56
 - example
 - configuring topology synchronization, 79
 - topology model, 24
- add/remove relationship, 28
- additional synchronization packages, 77
- alt query parameter, 190
 - alt=atom, 165
 - alt=xml, 165
- AncestorCI operand element, 109
- And operator
 - element, 103
 - usage, 97
- API library, file location, 210
- Atom feeds, 165, 166
 - alt media type, 190
 - Firefox, 166
 - Internet Explorer, 166
- attribute mapping, 115
 - example, 117
 - file, configuring, 82
 - to Boolean values in RTSM, 115
 - to Byte values in RTSM, 116
 - to Date values in RTSM, 116
 - to Double values in RTSM, 116
 - to Float values in RTSM, 116
 - to Integer values in RTSM, 115
 - to IntList values in RTSM, 116
 - to Long values in RTSM, 115
 - to StringList values in RTSM, 116
 - to String values in RTSM, 115
 - to String values of max. length in RTSM, 115
- attributemapping.xml, configuring, 82

- attributes
 - create new, 27
 - matching, 97
 - setting key values for CI types, 26
 - synchronizing
 - control of, 257
 - using Groovy scripts, 257

B

- basic topology synchronization, architecture, 63
- BDM
 - lifecycle state, 258
 - Mapping Manager, 255
- bi-directional synchronization, 256
- BSM Integration Adapter, 143
 - overview, 144
 - policies, 145
- bulk event update, 175

C

- Caption
 - field, 120
 - operand element, 106
- ChildCICollection operand element, 109
- ChildCI operand element, 108
- Children field, 120
- CI resolution, 71
 - mapping table, 71
- CIs
 - creating, 29
 - using DFM discovery, 29
 - using HPOM service model, 29
 - using topology synchronization, 29
 - creating relationships in RTSM, 29
 - data structure, 120
 - mapping events to, 33
 - matching types, 97
 - RelatedCiHint values
 - for hosted on CIs, 35
 - for virtual CIs, 35

- CI types
 - create new attribute, 27
 - creating new, 24, 25
 - set key attributes, 26
 - CMA
 - EventTypeIndicator
 - setting ETIs with, 39
 - setting RelatedCiHint variable, 34
 - CMDBAttribute operand element, 106
 - CMDBType operand element, 106
 - command-line utility
 - REST web service, 177
 - calling the help, 177
 - usage examples, 178
 - common mapping file format, 101
 - comparing files, 96
 - complex attributes, 192
 - URL escape codes, 193
 - Condition operator element
 - description, 102
 - example, 102, 112
 - conditions
 - examples, 102, 112
 - rules, 102
 - connected server
 - configuration, 224, 249
 - using logfile adapter, 207
 - configure Service manager as, 249
 - testing connection, 255
 - containmentrelations.xsd file, 91
 - Contains operator element, 105
 - content
 - creating content pack, 54
 - workflow, 55
 - creating RTSM packages, 53
 - migration
 - saving topology synchronization files, 54
 - steps, 53
 - uploading, 56
 - uploading content packs, 57
 - pack
 - uploading, 57
 - packaging, 53
 - upload, 56
 - RTSM packages, 56
 - uploading content packs, 57
 - using on another system, 53
 - workflow for uploading, 56
 - content development, 53 to 57
 - content pack, 77
 - ACME, 56
 - creating, 54
 - definitions, 55
 - dependencies, 55
 - referenced content, 55
 - selection example, 55
 - uploading, 57
 - workflow for creating, 55
 - XML file, 55
 - context mapping, 78
 - file configuring, 79
 - contextmapping.xml
 - configuring, 79, 80
 - correlation rules, 20
 - ACME landscape, 43
 - creating new CI types, 24
 - custom actions
 - scripts, 131
 - creating, 130, 131
 - definition attributes, 133
 - specifying, 131
 - custom attributes, mapping, 255
 - custom message attribute. *See* CMA
- ## D
- datadump.xsd file, 91
 - data structure, 120
 - debugging,EPI, 130
 - default synchronization package, 76
 - dependencies,content pack, 55
 - Dependencies field, 120
 - DependencyCICollection operand element, 110
 - DependencyCI operand element, 110
 - DependentCICollection operand element, 111
 - DependentCI operand element, 111
 - DescendantCICollection operand element, 110
 - DescendantCI operand element, 109
 - destroy method, 213
 - DFM
 - creating CIs using, 29
 - directories
 - synchronization data dumps, 96
 - XSD files, 91
 - discovery
 - choosing a method, 30
 - DFM, 29

- domain names, 107
- drill-down, 208
 - to Service Manager, 251
- dumping synchronization data, 94 to 96
- dynamic topology synchronization, 66
 - architecture, 68

E

- Eclipse, 93
- editors, XML. *See* XML editors
- elements
 - mapping, 112
 - operand, 106 to 108
 - operator, 103 to 105
- Ends With operator element, 104
- enriched directory, 96
- enrichment rules, 31
- EPI
 - debugging, 130
 - log files, 130
 - scripts, 127
 - ACID principle, 129
 - definition attributes, 133
 - entry points
 - after CI/ETI resolution, 128
 - after storing events, 129
 - before CI/ETI resolution, 128
 - before storing events, 128
 - entry points for running, 127
 - execution, 129
 - specifying, 129
 - troubleshooting, 130
- Equals operator element, 104
- error messages, 92
- escape codes, 193
- ETIs, 19, 33
 - assigning to an event, 39
 - creating, 36
 - overview, 36
 - setting with mapping rules, 40
 - filter configuration example, 40
 - rule configuration example, 40
- event browser
 - URL launch, 151
 - default, 151
 - defining columns, 153
 - filters, 155
 - CIs and CI types, 157
 - ETI and ETI values, 158
 - global CI ID, 158

- other event characteristics, 159
 - priorities, 157
 - string attributes, 156
 - time properties, 157
- from Service Manager, 253
- optional parameters, 151
- parameters and values, 152
- specifying, 151

- event change
 - creation
 - logfile adapter examples, 227 to 235
 - list, 167
- event changes
 - detecting, 167
 - forwarding, 222
 - to external event process, 221
 - synchronizing
 - from external process, 204
- event details
 - URL launch, 159
- event forwarding rule, configuring, 252
- event list
 - filtering, 166
- Event Processing Interface. *See* EPI
- events
 - advanced modification of properties, 174
 - analysis, 36
 - assigning ETIs to, 39
 - attributes supporting back sync., 224
 - bulk update, 175
 - change creation, 223
 - changes
 - forwarding, 222
 - changes list, 197
 - defining indicators, 36
 - detecting new, 165
 - drill-down, 208
 - event changes service, 164
 - events service, 164
 - forwarding, 201, 221
 - architecture, 201
 - modes
 - Notify, 202
 - Notify and Update, 202
 - Synchronize, 202
 - Synchronize and Transfer Control, 202
 - rule configuration, 209, 252
 - to external application, 201
 - to external event process, 221
- mapping to CIs, 33
- modifying, 167, 173
 - Firefox Poster Extension, 170
- processing

- additional, 20
 - receiving as Atom feeds, 166
 - in Firefox, 166
 - in Internet Explorer, 166
 - synchronizing, 201
 - changes, 222
 - updates, 223
- Event Synchronization Web Service, 201, 202
 - event change creation, 223
 - event update, 223
 - forwarding event changes, 221
 - forwarding events, 201, 221
 - forwarding modes
 - Notify, 202
 - Notify and Update, 202
 - Synchronize, 202
 - Synchronize and Transfer Control, 202
 - HTTP method calls, 203
 - GET, 203
 - HEAD, 203
 - POST, 203
 - interface, 221
 - OPR client, 221
 - OPR-compliant, 221
- EventTypeIndicator
 - setting ETIs with, 39
- event type indicators. *See* ETIs
- event update
 - logfile adapter examples, 225 to 226
- Event Web Service
 - accessing, 163
 - alt=atom parameter, 165
 - alt=xml parameter, 165
 - detecting new events, 165
 - editable properties, 194
 - event_change_list, 164
 - event_list, 164
 - events service, 164
 - file locations
 - Java API documentation, 197
 - schema, 197
 - filtering
 - complex attributes, 192
 - history lines, 196
 - event changes list, 197
 - recorded property changes, 196
 - OPR event service list, 164
 - query filter criteria, 191
 - query language, 185
 - query parameters
 - combining page_size and start_index, 188
 - compound filters, 187
 - data inclusion, 189
 - HTTP, 185

- include_closed, 189
 - include_relationships, 190
 - media type, 190
 - order_by, 189
 - order_direction, 189
 - ordering, 189
 - page_size, 188
 - paging, 188
 - query, 187
 - simple filters, 187
 - start_index, 188
 - watermark, 186
- examples
 - XML file validation, 93
 - XPath, 122
- Exists operator element, 103
- expressions
 - regular, 105, 107
 - relative, 97
- external event process
 - forwarding events to, 221
 - integrating, 221
 - FAQ, 237 to 248
 - using Event Synchronization Web Service, 202
 - using Groovy scripts, 202, 207
 - synchronizing event changes from, 204
- External Info tab, 222

F

- False operator element, 103
- figures
 - data structure, 121
 - examples
 - file validation, 93
 - XPath, 122
- file locations, 71, 210
 - API library, 210
 - event integration Groovy scripts, 211
 - Event Web Service
 - Java API documentation, 197
 - schema, 197
 - Groovy scripts, 87
 - Java API documentation, 210
 - OPR event schema, 210
 - topology synchronization, 54, 56
 - basic, 71
 - dynamic, 72

- files
 - comparing, 96
 - content pack, 55
 - root elements, 96
 - XML, validating, 91
 - XSD, 91
- filtering, 78
 - context mapping, 113
 - Event Web Service
 - by date and time, 186
 - by event attributes, 187
 - compound filters, 187
 - query filter criteria, 191
 - simple filters, 187
- Firefox Poster Extension, 170
 - modifying events using, 170
- forwardChange method, 215
- forwardEvent method, 214
- forwarding events
 - from OPR client, 221
 - to external event process, 201
- forwarding rule
 - configuring, 209, 252
- forwarding types
 - Notify, 202
 - Notify and Update, 202
 - Synchronize, 204
 - Synchronize and Transfer Control, 204
- frequently asked questions
 - external event process integration, 237 to 248
- From CI Get operand element, 111

G

- GET
 - method call, 203
 - request, web service, 222
- getExternalEvent method, 219
- graphs, 21, 51
- Groovy scripts, 76
 - adapter, 202
 - Service Manager, 202
 - adding custom attributes, 255
 - API, 134
 - attributes synchronization, 257
 - custom actions
 - samples, 139
 - SimpleExample.groovy, 139
 - TranslateEventTitle.groovy, 140
 - customizing, 255
 - tips, 257
 - EPI
 - samples, 135
 - RegExample.groovy, 137
 - ResolveLocationFromDB.groovy, 138
 - SimpleExampleEPI.groovy, 135
- event integration
 - file locations, 211
- integrating external event process using, 207
- interface, 210
- location, 87
- logfile adapter, 207
- methods
 - additional, 217
 - destroy, 213
 - forwardChange, 215
 - forwardEvent, 214
 - getExternalEvent, 219
 - init, 212
 - management of, 220
 - ping, 213
 - receiveChange, 216
 - toExternalEvent, 219
- postUpload.groovy, 86
- preEnrichment.groovy, 86
- preUpload.groovy, 86

H

- HEAD method call, 203
- Health Indicators. *See* HIs
- HIs, 19, 33
 - assigning to KPIs, 41
 - creating, 36
 - overview, 36
- history lines, 196
 - event changes list, 197
 - recorded property changes, 196
- hosted on CIs
 - RelatedCiHint values, 35
- HPOM
 - connection settings, 72
 - service model, creating CIs using, 29
 - Topology Synchronization Connection Settings, 73
 - Topology Synchronization settings, 72, 73
- HP Service Manager integration, 249
- HP-UX operating system, 122
- HTTP
 - Event Synchronization Web Service
 - method calls, 203
 - GET, 203
 - HEAD, 203
 - POST, 203
 - query parameters, 185

I

- id field, 120
- impact propagation, 32
- include_closed query parameter, 189
- include_relationships query parameter, 190
- indicators
 - defining, 36
 - event type, 19, 36
 - health, 19, 36
- init method, 212
- Integration Adapter, 143
 - integrating events with, 146
 - overview, 144
 - policies, 145
- Is Deletion CI operator element, 105
- Is Node operator element, 104
- Is Root CI operator element, 104

J

- Java API documentation
 - Event Web Service, 197
 - file location, 210
- Java data structure, 120

K

- key attributes, CI type, 26
- key performance indicators. *See* KPIs
- KPIs, 33
 - assigning HIs to, 41

L

- List operand element, 108
- locations
 - synchronization data dumps, 96
 - synchronization packages
 - sync-packages directory, 84
 - XSD files, 91
- logfile adapter
 - configuring connected server with, 207
 - examples
 - event change creation, 227 to 235
 - event update, 225 to 226
- LogfileAdapter.groovy, 207
- log files, EPI, 130
- logs, error, 92

M

- mapping
 - attributes, 115
 - example, 117
 - to Boolean values in RTSM, 115
 - to Byte values in RTSM, 116
 - to Date values in RTSM, 116
 - to Double values in RTSM, 116
 - to Float values in RTSM, 116
 - to Integer values in RTSM, 115
 - to IntList values in RTSM, 116
 - to Long values in RTSM, 115
 - to StringList values in RTSM, 116
 - to String values in RTSM, 115
 - to String values of max. length in RTSM, 115
 - elements, 112
 - events to CIs, 33
 - file
 - format, 101
 - syntax, 102 to 112
 - Condition examples, 102
 - lifecycle states
 - BDM to OPR, 258
 - OPR to BDM, 258
 - relation, 118
 - message alias (for CI res.), 118
 - root container, 118
 - rules
 - setting ETIs with, 40
 - validating, 96
 - syntax, 102 to 123
 - to Boolean values in RTSM, 115
 - to byte values in RTSM, 116
 - to Date values in RTSM, 116
 - to Double values in RTSM, 116
 - to Float values in RTSM, 116
 - to Integer values in RTSM, 115
 - to IntList values in RTSM, 116
 - to Long values in RTSM, 115
 - to StringList values in RTSM, 116
 - to String values in RTSM, 115
 - to String values of max. length in RTSM, 115
 - type, 114
- mapping.xsd file, 91
- mapping files, 78
 - attribute
 - attributemapping.xml, 78
 - context
 - contextmapping.xml, 78
 - relation
 - relationmapping.xml, 78
 - topology synchronization, 75
 - type
 - typemapping.xml, 78

Matches operator element, 104

matching attributes, 97

messages, error, 92

N

navigation, XPath, 120

Network Node Manager i. *See* NNMi

NNMi, 145

Node field, 120

nodegroups synchronization package, 76

nodetypes.xsd file, 92

non-matching XPath expressions, 97

normalized directory, 96

normal mode, topology synchronization, 64

Notify and Update forwarding mode, 202

Notify forwarding mode, 202

Not operator element, 103

O

OMAttribute operand element, 106

OMAttributes field, 120

OMId operand element, 106

OMType operand element, 106

OMType operator element, 97

operand elements, 106 to 108

operations agent synchronization package, 76

operator elements, 103 to 105

OPR

client

forwarding events from, 221

event

schema file location, 210

services list, 164

lifecycle state, 258

opr-sd-tool.bat, 73, 74

opr-startTopologySync.bat

executing, 64

normal mode, 64

skipservices option, 65

touch mode, 65

Oracle applications, 122

order_by query parameter, 189

order_direction query parameter, 189

OriginServer operand element, 112

Or operator element, 103

P

package.xml file

configuring, 79

usage, 75, 77

package.xsd file, 91

package descriptor file

configuring, 79

packages

RTSM, 53

upload, 56

workflow for creating, 53

synchronization

mapping, 70

page_size query parameter, 188

ParentCI operand element, 108

ping

method, 213

web service request

from external application, 223

to external app., 222

policies

Integration Adapter, 145

interceptor, 145

POST method call, 203

post-upload scripts, 85

postUpload.groovy, 86

pre-mapping scripts, 85

preEnrichment.groovy, 86

pre-upload scripts, 85

preUpload.groovy, 86

properties, advanced event modification, 174

Q

queries, XPath, 97

query filter parameter, 187

query language, Event Web Service, 185

R

raw data directory, 96

receiveChange method, 216

referenced content, 55

regular expressions, 105, 107

- RelatedCiHint
 - best practice for setting, 35
 - setting CMA variable, 34
 - values, 35
 - hosted on CIs, 35
 - virtual CIs, 35
- relation data structure, 120
- relation mapping, 118
 - configuring file, 83
 - message alias (for CI res.), 118
 - root container, 118
- relationmapping.xml
 - configuring, 83
- relations
 - creating an impact relation, 32
 - cross-domain, 31
- relationships
 - add/remove, 28
 - creating, 28
- relative expressions, 97
- Replace operand element, 107
- REST
 - client
 - modifying events using, 167
 - web service
 - command-line utility, 177
 - help, 177
 - usage examples, 178
 - HTTP method calls, 203
 - GET, 203
 - HEAD, 203
 - POST, 203
 - web service utility, 173
- RestWsUtil.bat, 177
 - help, 177
 - modifying events using, 173
 - usage examples, 178
 - creating a custom attribute, 179
 - delete a custom attribute, 183
 - reading event custom attributes, 178
 - reading events, 178
 - updating a custom attribute, 180
 - updating an event title, 181
 - updating event lifecycle state, 182
 - updating event lifecycle state and severity, 182
- RTSM, 29
 - ACME view in, 80
 - creating packages, 53
 - packages
 - upload, 56
 - workflow to create, 53

- rules
 - conditions, 102
 - correlation, 20
 - ACME landscape, 43
 - enrichment, 31
 - mapping
 - configuring, 40
 - setting ETIs with, 40
 - validating, 96
 - names, 101
 - writing, 97

S

- schema, Event Web Service location, 197
- script definition
 - attributes, 133
 - Active, 133
 - Filter, 133
 - Name, 133
 - Read-only, 133
 - Timeout, 133
 - format, 134
- scripting, 70
 - error handling, 88
 - examples, 89
 - naming convention, 86
 - synchronization, 85
 - topology synchronization, 70
 - variables and scope, 87
- scripts
 - ACID principle, 129
 - custom actions, 131
 - creating, 130, 131
 - definition attributes, 133
 - samples, 139
 - SimpleExample.groovy, 139
 - TranslateEventTitle.groovy, 140
 - specifying, 131
 - definition format, 134
 - enabling and disabling, 86
- EPI
 - creating, 130
 - definition attributes, 133
 - entry points, 127
 - after CI/ETI resolution, 128
 - after storing events, 129
 - before CI/ETI resolution, 128
 - before storing events, 128
 - execution, 129
 - samples, 135
 - RegExample.groovy, 137
 - ResolveLocationFromDB.groovy, 138
 - SimpleExampleEPI.groovy, 135
 - specifying, 129

- Groovy
 - API, 134
 - EPI, 127
 - location, 87
 - postUpload.groovy, 86
 - preEnrichment.groovy, 86
 - preUpload.groovy, 86
- post-upload, 85
- pre-mapping, 85
- pre-upload, 85
- service discovery log, dynamic topology
 - synchronization, 97
- service document, 164
 - event changes service, 164
 - events service, 164
- Service Manager
 - drill-down to Operations Management, 253
 - event drill-down to, 251
 - Groovy script adapter, 202
 - integration, 249
 - testing connection, 255
 - server configuration, 253
 - URL launch from event browser, 254
- service type definitions, upload to database, 74
- skipservices option, basic topology synchronization, 65
- standard synchronization packages, 76
- start_index query parameter, 188
- Starts With operator element, 104
- structure, data, 120
- synchronization
 - attributes
 - bi-directional, 256
 - uni-directional, 256
 - customizing, 85
 - error handling, 88
 - example, 89
 - examples, 89
 - naming convention, 86
 - scope, 87
 - variables, 87
- synchronization data dump
 - creating, 94
 - overview, 94
 - viewing, 96
- synchronization packages, 75
 - additional, 77
 - enabling, 77
 - locations
 - sync-packages directory, 84
 - mapping, 70

- scripts, 84
- standard, 76
 - default, 76
 - nodegroups, 76
 - operations agent, 76
- topology synchronization
 - uploading to database, 73

Synchronize and Transfer Control forwarding type, 202, 204

Synchronize forwarding type, 202, 204

sync-packages directory, 84

syntax, mapping files, 102 to 112

T

target server, configuration, 224

toExternalEvent method, 219

tools, 20, 47

- ACME application server example, 47

topology

- ACME model, 24

- view, 31

topology data, 19, 23 to 32

topology synchronization, 61 to 123

- basic

- architecture, 63

- file locations, 71

- normal mode, 64

- skipservices option, 65

- starting, 64

- touch mode, 65

- CI resolution mapping table, 71

- common issues, 98

- comparison of basic and dynamic, 69

- configuration

- ACME example, 79

- dynamic, 66

- architecture, 68

- file locations, 72

- service discovery log, 97

- files

- copying, 56

- location, 54

- saving, 54

- Groovy scripts, 76

- HPOM

- connection settings, 72, 73

- Topology Synchronization settings, 72

- known limitations, 99

- mapping files, 75

- opr-sd-tool.bat, 73, 74

- overview, 61

- packages

- upload to database, 73
 - scripting, 70
 - service type definitions
 - upload to database, 74
 - settings, 72
 - synchronization packages, 70, 75
 - additional, 77
 - enabling, 77
 - standard, 76
 - default, 76
 - nodegroups, 76
 - operations agent, 76
 - Windows scheduler tasks, 64
- touch mode, topology synchronization, 65
- troubleshooting
 - EPI, 130
 - topology synchronization
 - common issues, 98
- True operator element, 103
- Type field, 120
- type mapping, 114
 - definition, 81
 - file
 - configuring, 80
 - typemapping.xml, 80

U

- uni-directional synchronization, 256
- uploading content, 56
 - content packs, 57
- URL
 - escape code, 193
 - launch
 - of event browser, 151, 205
 - default, 151
 - defining columns, 153
 - filters, 155
 - CI and CI types, 157
 - ETI and ETI values, 158
 - global CI ID, 158
 - other event characteristics, 159
 - priorities, 157
 - string attributes, 156
 - time properties, 157
 - from external application, 205
 - from Service Manager, 253
 - optional parameters, 151
 - parameters and values, 152
 - specifying, 151
 - of event details, 159
 - of Service Manager, 254

V

- validating
 - mapping rules, 96
 - XML files
 - automatically, 91
 - manually, 92 to 93
 - XPath Expressions, 96
- Value operand element, 108
- viewing synchronization data dumps, 96
- view mappings, 21, 49
 - ACME example, 49
 - pane, 49
 - View Manager, 49
- views
 - creating, 49
 - RTSM, 21
 - topology, 31
- virtual CIs, RelatedCiHint values, 35

W

- W3C, 91
- watermark query parameter, 186
- web service request
 - GET, 222
 - ping, 222, 223
- writing rules, 97

X

- XML, 55, 96
 - editors
 - validating
 - files, 92
 - XPath expressions, 96
 - writing rules, 97
 - files
 - synchronized CIs, 94
 - validating configuration, 91
 - format, 165
 - interceptor policies, 145
 - Schema Definition, 91
- XPath
 - examples, 122 to 123
 - expressions, 123
 - matching expressions, 97
 - navigation, 120
 - validating expressions, 96
 - writing queries, 97
 - XPathResultList
 - operand element, 108
 - operator element, 97

XPathResult operand element
description, 107
usage, 97

XSD. *See* XML Schema Definition

