

# HP Operations Manager i

for the Windows operating system

Software Version: 8.10

---

## Extensibility Guide



## Legal Notices

### Warranty

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

The information contained herein is subject to change without notice.

### Restricted Rights Legend

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

### Copyright Notices

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

### Trademark Notices

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

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

UNIX® is a registered trademark of The Open Group.

### Acknowledgements

This product includes ANTLR.

This product includes software developed by Andy Clark.

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

This product includes software developed by Daisuke Okajima and Kohsuke Kawaguchi (<http://relaxngcc.sf.net/>).

This product includes cryptographic software written by Eric Young ([eay@cryptsoft.com](mailto:eay@cryptsoft.com)).

This product includes software developed by the Indiana University Extreme! Lab (<http://www.extreme.indiana.edu/>).

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

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes software developed by the OpenSymphony Group (<http://www.opensymphony.com/>).

This product includes code licensed from RSA Data Security.

This product includes software written by Tim Hudson ([tjh@cryptsoft.com](mailto:tjh@cryptsoft.com)).

## 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](http://www.hp.com/go/hpsoftwaresupport)**

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

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

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

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

To find more information about access levels, go to:

**[http://h20230.www2.hp.com/new\\_access\\_levels.jsp](http://h20230.www2.hp.com/new_access_levels.jsp)**

1	Introduction	9
	Prerequisites	10
	Section I: Content Development	11
2	HP OMi Integration Artifacts	13
	Topology	13
	Event and Health Indicators	13
	Correlation Rules	14
	Tools	14
	Graphs	14
3	Topology	15
	Integrating New Applications	16
	Create New CI Types for the New Application	16
	Set Key Attribute Values for New CI Types	18
	Establish Relations for New Application	20
	Creating CIs and CI Relationships in the UCMDB	21
	Creating CIs Using UCMDB/HP DDM Discovery	21
	Creating CIs Using the HPOM Service Model and Topology Synchronization	21
	Considerations when Choosing a Discovery Method	22
	When to Use HP DDM	22
	When to Use Topology Synchronization	22
	When to Create CIs Manually	22
	Enrichment Rules	23
	Topology View of the New Application	23
	Impact Propagation	24
4	Event and Health Indicators	25
	Mapping Events to CIs	25
	Setting the Custom Message Attribute CiInfo	26
	CiInfo Values	27
	Creating Event and Health Indicators	28
	Analyze Events and Define Indicators	28
	Assigning Event Type Indicators to an Event	29
	Setting ETIs with the CMA “EventTypeIndicator”	29
	Setting ETIs with ETI Mapping Rules	29
	Assigning HIs to KPIs	31
5	Correlation Rules	33
6	HP OMi Tools	35
7	Graphs	37
8	View Mappings	39
9	Packaging Content	41
	Create UCMDB Packages	41

Save Topology Synchronization Files . . . . .	42
Save Graph Templates . . . . .	42
Create an HP OMi Content Pack . . . . .	43
Upload the Content . . . . .	45
Upload UCMDB Packages . . . . .	45
Copy Topology Synchronization Files . . . . .	45
Copy Graph Templates . . . . .	45
Upload HP OMi Content Packs . . . . .	45
<b>Section II: Topology Synchronization . . . . .</b>	<b>47</b>
<b>10 Topology Synchronization Overview . . . . .</b>	<b>49</b>
Topology Synchronization Architecture . . . . .	50
Synchronization Packages and Mapping . . . . .	51
Scripting . . . . .	51
CI Resolution Using a Mapping Table . . . . .	52
File Locations . . . . .	52
Topology Synchronization Settings . . . . .	53
Running Topology Synchronization . . . . .	54
Normal Mode . . . . .	54
Touch Mode . . . . .	55
<b>11 Synchronization Packages . . . . .</b>	<b>57</b>
Synchronization Packages Overview . . . . .	57
Standard Out-of-the-box Synchronization Packages . . . . .	58
Additional Out-of-the-box Synchronization Packages . . . . .	59
Package Descriptor File: package.xml . . . . .	59
Mapping Files . . . . .	60
Context Mapping (Filtering): contextmapping.xml . . . . .	60
Type Mapping: typemapping.xml . . . . .	60
Attribute Mapping: attributemapping.xml . . . . .	60
Relation Mapping: relationmapping.xml . . . . .	60
Configuring Topology Synchronization: ACME Example . . . . .	61
Configure Package Descriptor File: package.xml . . . . .	61
Configure Context Mapping (Filtering) File: contextmapping.xml . . . . .	61
Configure Type Mapping File: typemapping.xml . . . . .	62
Configure Attribute Mapping File: attributemapping.xml . . . . .	64
Configure Relation Mapping File: relationmapping.xml . . . . .	65
Customizing Synchronization and Scripting . . . . .	66
Synchronization Package Locations . . . . .	66
<b>12 Scripting . . . . .</b>	<b>67</b>
Groovy Scripts . . . . .	68
Enabling and Disabling Scripts . . . . .	68
Groovy Script Location . . . . .	68
Script Variables . . . . .	69
Handling Errors . . . . .	70

Sample Script: preUpload.groovy .....	71
<b>13 Testing, Deployment, and Troubleshooting .....</b>	<b>73</b>
Validating XML Configuration Files .....	73
HP OMi XSD Files .....	73
Validating Files Automatically .....	74
Validating Files Manually .....	74
Dumping Synchronization Data .....	76
Creating a Synchronization Data Dump .....	76
Data Dump Example .....	77
Viewing a Synchronization Data Dump .....	78
Validating Mapping Rules .....	78
Writing Rules .....	79
Simplifying Rule Development .....	79
Avoiding Complex XPath Queries .....	79
Matching Against Existing Attributes Only .....	79
Avoiding Broad XPath Expressions .....	79
Troubleshooting, Common Issues, and Tips .....	80
.....	80
<b>A Mapping Engine and Syntax .....</b>	<b>81</b>
Common Mapping File Format .....	81
Mapping File Syntax .....	82
Rules .....	82
Rule Conditions .....	82
Condition Examples .....	82
Operator Elements .....	82
Operand Elements .....	85
Mapping Elements .....	91
Condition Examples .....	91
Filtering .....	92
Context Mapping .....	92
Type Mapping .....	93
Mapping .....	93
Attribute Mapping .....	94
Mapping to String values in the UCMDB .....	94
Mapping to Integer values in the UCMDB .....	94
Mapping to Boolean values in the UCMDB .....	94
Relation Mapping .....	96
Root Container Mapping .....	96
Message alias mapping for CI resolution .....	96
XPath Navigation .....	98
Data Structure .....	98
CI Data Structure .....	98
Relationship Data Structure .....	98
Example of an XPath-Navigated Data Structure .....	100
XPath Expressions and Example Values .....	101





# 1 Introduction

This *HP Operations Manager i Extensibility Guide* provides information for content developers and integrators to customize and extend the functionality of HP Operations Manager i (HP OMi).

The guide is divided into two sections.

- Section I: [Content Development](#), starting on [page 11](#)
- Section II: [Topology Synchronization](#), starting on [page 47](#)

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 HP OMi. Full details about the concepts of HP OMi can be found in the *HP Operations Manager i Concepts Guide*.

Section II: [Topology Synchronization](#) provides information for developers to create their own topology synchronization mapping rules, to augment the out-of-the-box mapping rules provided by HP OMi to populate the Universal Configuration Management Database (UCMDB) with configuration items (CIs) and CI relationships from services, nodes, and node groups in HPOM.

For customers who have created their own service model in HPOM, they can reuse that model, and use the topology synchronization feature of HP OMi to create corresponding CIs and CI relationships in the UCMDB.

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.

# 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:
  - HP Operations Manager i (HP OMi). The *HP Operations Manager i Concepts Guide* is an essential reference for this document.
  - HP Operations Manager (HPOM) for Windows and UNIX.
  - HP Business Availability Center (HP BAC).
  - Universal Configuration Management Database (UCMDB). Detailed administrative and operational knowledge is assumed.
- Knowledge of Groovy scripting and syntax is required for creating Groovy scripts. HP OMi supports Groovy for its scripting capabilities, and uses Groovy scripts in the topology synchronization process.
- 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 with HP OMi according to customer requirements.
- Add monitoring capabilities for new applications and elements of the customer's IT environment.

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

This section is structured as follows:

[Chapter 2, HP OMi Integration Artifacts](#), page 13

[Chapter 3, Topology](#), page 15

[Chapter 4, Event and Health Indicators](#), page 25

[Chapter 5, Correlation Rules](#), page 33

[Chapter 6, HP OMi Tools](#), page 35

[Chapter 8, View Mappings](#), page 39

[Chapter 9, Packaging Content](#), page 41



## 2 HP OMi Integration Artifacts

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

- Topology
- Event and health indicators
- Correlation rules
- Tools
- Graphs

### Topology

Topology data for an HP OMi monitored environment is contained in an universal configuration management database (UCMDB). The UCMDB contains configuration item (CI) types for all monitored objects together with their relationship to other CIs.

To integrate a new application into an HP OMi monitoring solution, and create a topology view of the new application, it is necessary to:

- Create new CI types for the new application.
- Identify the key attribute values for the new CI types.
- Establish the relations for the new application.
- Create CIs and CI relationships in the UCMDB.

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

For more details about the role of topology data in integrating a new application, see [Chapter 3, Topology](#).

### Event and Health Indicators

To benefit from the advanced health-based monitoring features of HP OMi for your new application area, it is necessary to:

- Populate the UCMDB with CIs, and HP OMi events must be mapped to the correct CIs in the UCMDB.

- Transform events into data about the service health of CIs based on received events. 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 and Health Indicators](#).

## Correlation Rules

HP OMi simplifies the event management process by not only consolidating events from all sources in a central console, but also correlating events using topology-based event correlation.

Topology-based event correlation rules make associations between known a root-cause event and related symptom events. Symptom events are those events that occur as a consequence of the root-cause event. Topology-based event correlation 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.

HP OMi uses HI/ETI values to represent the events that have an impact on the configuration item types specified in the topology-based event correlation rule. These values are used to create correlation rules.

For more details about correlation rules, see [Chapter 5, Correlation Rules](#).

## 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 6, HP OMi Tools](#).

## Graphs

HP OMi provides graphs and charts that provide 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.

## 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](#) (see [page 16](#))
- [Topology View of the New Application](#) (see [page 23](#))
- [Impact Propagation](#) (see [page 24](#))
- [Enrichment Rules](#) (see [page 23](#))

# Integrating New Applications

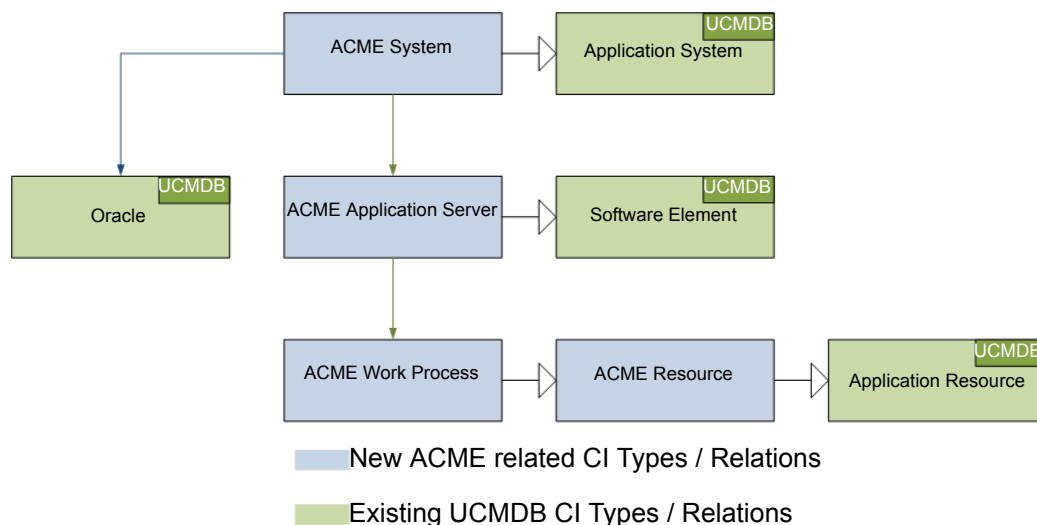
To integrate a new application into an HP OMi monitoring solution, and create a topology view of the new application, it is necessary to:

- Create new CI types for the new application.
- Set the key attribute values for the new CI types.
- Establish the relations for the new application.
- Create CIs and CI relationships in the UCMDB.

## 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 16 shows four new CI types, shown in blue:

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

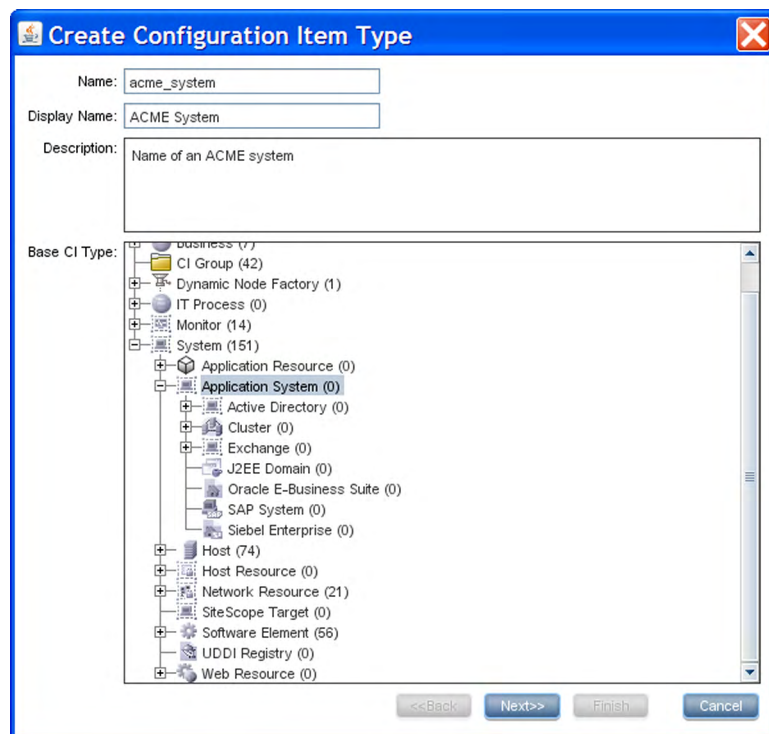
These new CI types are child elements of CI types that already exist in the UCMDB (shown in green).



Comprehensive details about how to create new CI types, and how to work with the concepts of the UCMDB, are described in the *HP Business Availability Center Model Management* guide.

To create a new CI type, do the following:

- 1 Navigate to the CI Type Manager:  
**Universal CMDB 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:  
**IT Universe → System → Application System**
- 4 Right-click, and select **New**. 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, 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 18.

## Set Key Attribute Values for New CI Types

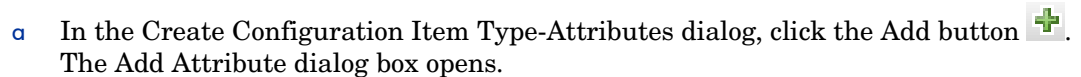
It is essential to identify the new CI types with unique key attributes. By setting unique key attributes, you can make sure there are no duplicate CIs created, for instance, by different discovery sources.

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 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.)



**Add Attribute**

**Details** | **Advanced**

Attribute Name:

Display Name:

Description:

Attribute Type:

☒ Primitive ☐ Enumeration/List

▼

Value Size:

Default Value:

- b In the Attribute Name field, enter the name of the new attribute.  
In the Display Name field, enter the display name of the new attribute:  
**acme\_installation\_number**  
Optional: In the Description field, enter a description for the new attribute: **ACME Installation Number**
  - 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 19.
  - 4 Click **Finish**.

## Establish Relations for New Application

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

In the out-of-the-box UCMDB model, there is a “member” relation between the ACME system and the ACME application server. As illustrated in [Figure 1](#) on page 16, our example ACME system depends on an Oracle database. This “depend” relation, however, does not exist in the out-of-the-box UCMDB model. You must create this relation using the CI Type Manager:

**Universal CMDB Administration → Modeling → CI Type Manager**

Management of Oracle databases is captured by the HP Operations Smart Plug-in for Databases, which is HP OMi-ready.

## Creating CIs and CI Relationships in the UCMDB

The next step in integrating the new application is to create CIs and CI relationships in the UCMDB.

There are three methods to create CIs and CI relationships:

- Create the CIs and CI relationships using discovery features of the UCMDB and HP Discovery and Dependency Mapping (HP DDM).
- Create the CIs and CI relationships required by HP OMi using the topology synchronization feature of HP OMi. 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 the topology synchronization feature of HP OMi, see [Section II, Topology Synchronization](#) on page 47.

- Create CIs manually in the UCMDB. 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 UCMDB/HP DDM Discovery

HP DDM 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:

- HP DDM Standard discovers hosts and hosts relationships. This is not sufficient to discover all CI types that are monitored by OMi.
- HP DDM Advanced discovers resources such as applications, databases, network devices, servers, and so on.
- HP DDM Advanced also allows users to extend the UCMDB model and write their own patterns.

It is also possible to query external data sources:

- Comma Separated Value (CSV) Files
- Properties Files
- Databases

For more details, see the *HP Discovery and Dependency Mapping (DDM) User Guide*, especially Chapter 8, “Discovery and Dependency Mapping Content” and Chapter 9, “Importing Data from External Sources”.

### 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.

These customers would use either the service discovery features of the HP Operations Smart Plug-ins or their own discovery mechanisms to create the service tree.

If this is the case, then they can use the topology synchronization feature of HP OMi to create CIs and CI relationships, based on the HPOM service model and topology synchronization mapping rules. HP OMi provides out-of-the-box mapping rules that are able to map service models created by the following HP Operations Smart Plug-ins, all of which are HP OMi-enabled:

- 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

Many customers have invested a lot of effort in creating their own service model, together with a corresponding manual or automatic service model creation process. They can reuse that model and create corresponding UCMDB configuration items automatically. All they need to do is to write corresponding topology synchronization mapping rules for their model.

For more details about the topology synchronization feature of HP OMi, see [Section II, Topology Synchronization](#) on page 47.

## Considerations when Choosing a Discovery Method

The following considerations are useful in helping you decide which discovery option to use to populate the UCMDB.

### When to Use HP DDM

Use HP DDM if:

- You already use HP DDM to populate the UCMDB, or are planning to use it.
- You do **not** already have a service model in HPOM. We recommend using HP DDM, as it is a much better discovery option for populating the UCMDB for HP OMi.

### 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) HP DDM to populate the UCMDB, *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.

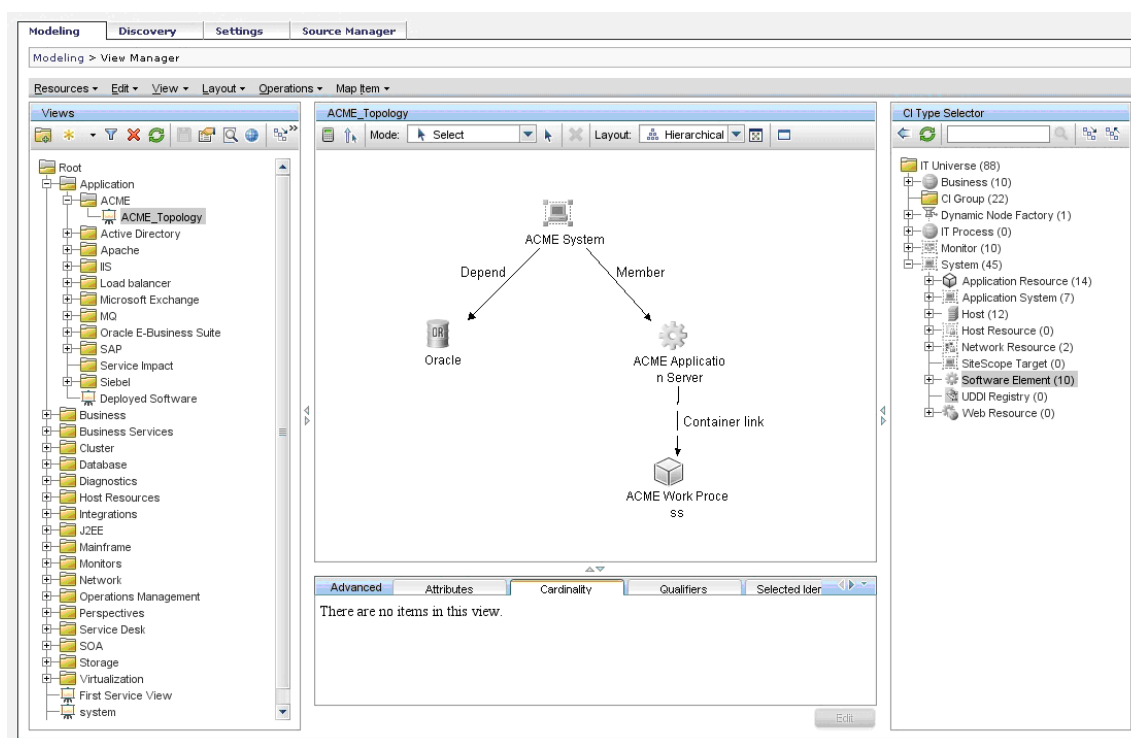
## Enrichment Rules

Figure 1 on page 16, shows the ACME example topology, 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 “depend” relation can be generated by an enrichment rule.

More information about how to create and maintain enrichment rules, see Chapter 9, “Enrichment Manager” of the *HP Business Availability Center Model Management* guide.

## Topology View of the New Application

Use the UCMDB view manager to create a view to display the ACME topology. Figure 2 shows the ACME topology view in the UCMDB view manager.



**Figure 2** ACME topology view in UCMDB view manager

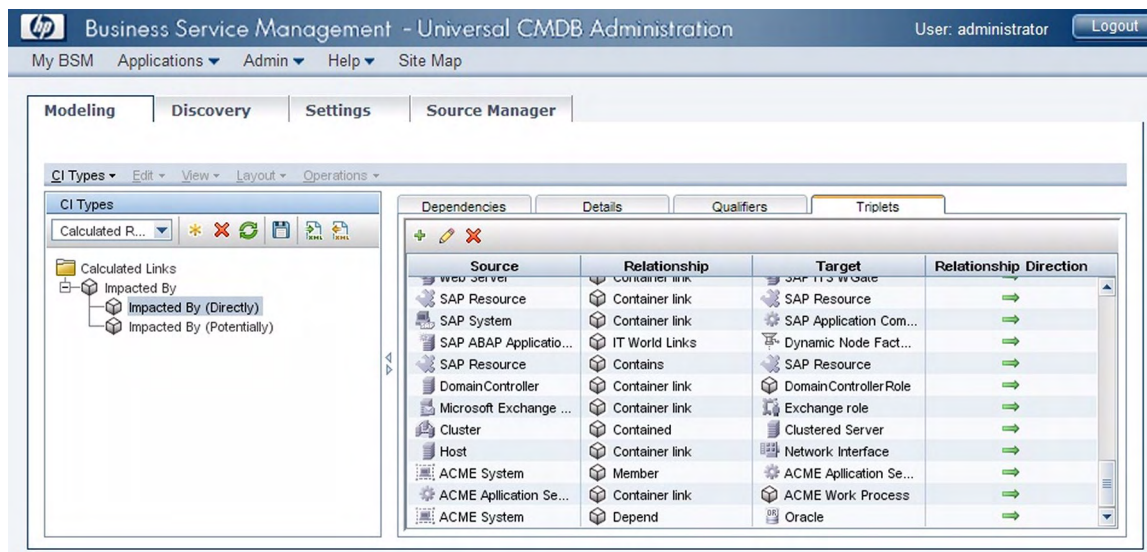


# Impact Propagation

Impact modeling is performed using calculated relations. Impact relationships are important for KPI calculations. For a detailed description of impact modeling in the UCMDB, including concepts, see Chapter 12 of the *HP Business Availability Center Model Management* 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 System.



**Figure 3** Creating an impact relation between the ACME Application Server and the ACME System



## 4 Event 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 of HP OMi.

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 features of HP OMi, it is necessary to:

- Populate the UCMDB with CIs, and map HP OMi events to the correct CIs in the UCMDB.
- 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 HP OMi is that HP OMi events are mapped to the correct CIs in the UCMDB.

If events are not mapped to the correct CIs:

- Setting health or event type 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 UCMDB 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.

HP OMi uses smart mapping technology 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 HP OMi uses this DNS name 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 UCMDB).

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, HP OMi evaluates 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

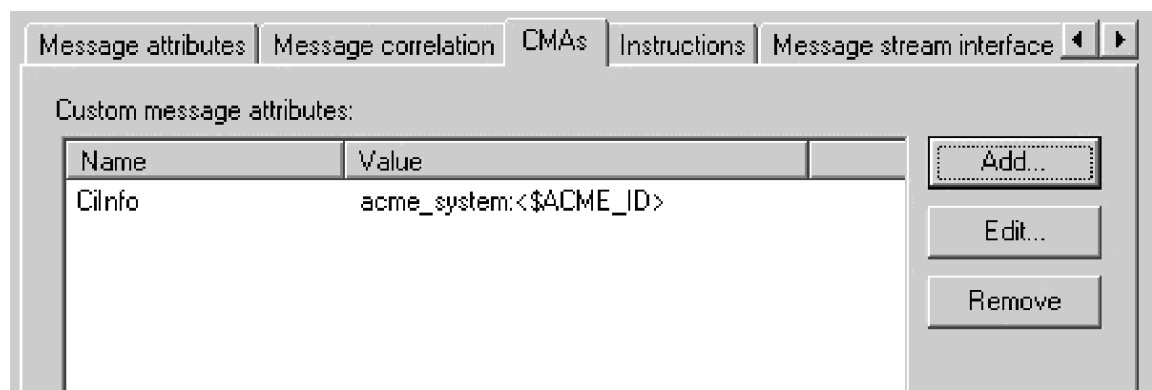
The separator (:) can be configured in the HP BAC Infrastructure Settings.

All the hints are then evaluated to find a matching CI in the UCMDB. 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 UCMDB 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 `CiInfo` in the HPOM message.

## Setting the Custom Message Attribute CiInfo

Figure 4 shows an example of how to set the CMA `CiInfo`:



**Figure 4** Setting custom message attribute `CiInfo`

The following considerations regarding best practice are important to bear in mind when setting the `CiInfo` variable:

- The CMA `CiInfo` should have sufficient hints to find the corresponding CI.
- It is necessary to differentiate between CIs that have a container-link relation to a host, and those that do not have such a relation.

## CiInfo Values

In general, the `CiInfo` 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) relation 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 and Health Indicators

After mapping the monitored HPOM events to the correct CIs in the UCMDB, the next step is to create event 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.

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 28 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_SystemStatus_001
ACME Application Server	HI	ACME Application Server Status	Unavailable	Major	ACME_SystemStatus_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	High	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.

## Assigning Event Type Indicators to an Event

In HP OMi, there are two ways to assign ETIs to an event:

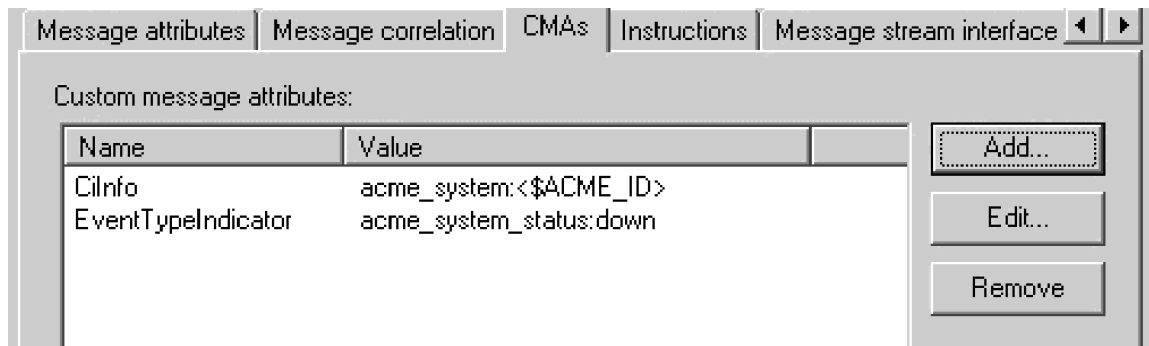
- Use ETI mapping rules
- Set the custom message attribute `EventTypeIndicator` within an HPOM policy

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/HP OMi events. However, it is also possible to assign an ETI to an event with ETI mapping rules.

### Setting ETIs with the CMA “EventTypeIndicator”

Table 2 on page 28 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 `EventTypeIndicator`.

Table 5 on page 29 shows an example configuration of the CMA `EventTypeIndicator`.



**Figure 5** Custom message attribute `EventTypeIndicator`

### Setting ETIs with ETI Mapping Rules

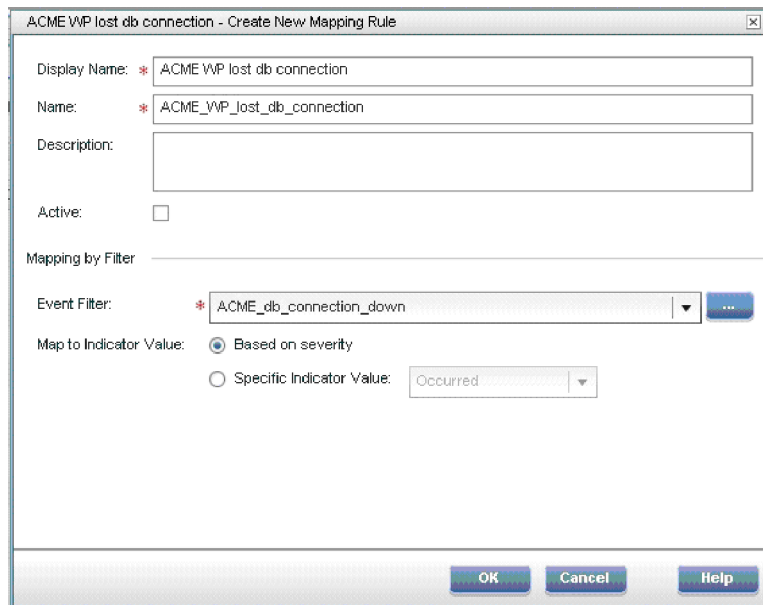
In the ACME example, all messages are sent using HP OMi policies. Therefore, the CMAs are used to set the ETIs. However, it is also possible to use mapping rules to set ETIs for incoming OMi messages.

Figure 6 on page 30 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 30 illustrates the configuration of the mapping rule itself for the Lost Database Connection ETI, using the filter configuration shown in Figure 6 on page 30.



**Figure 7 Mapping rule configuration**

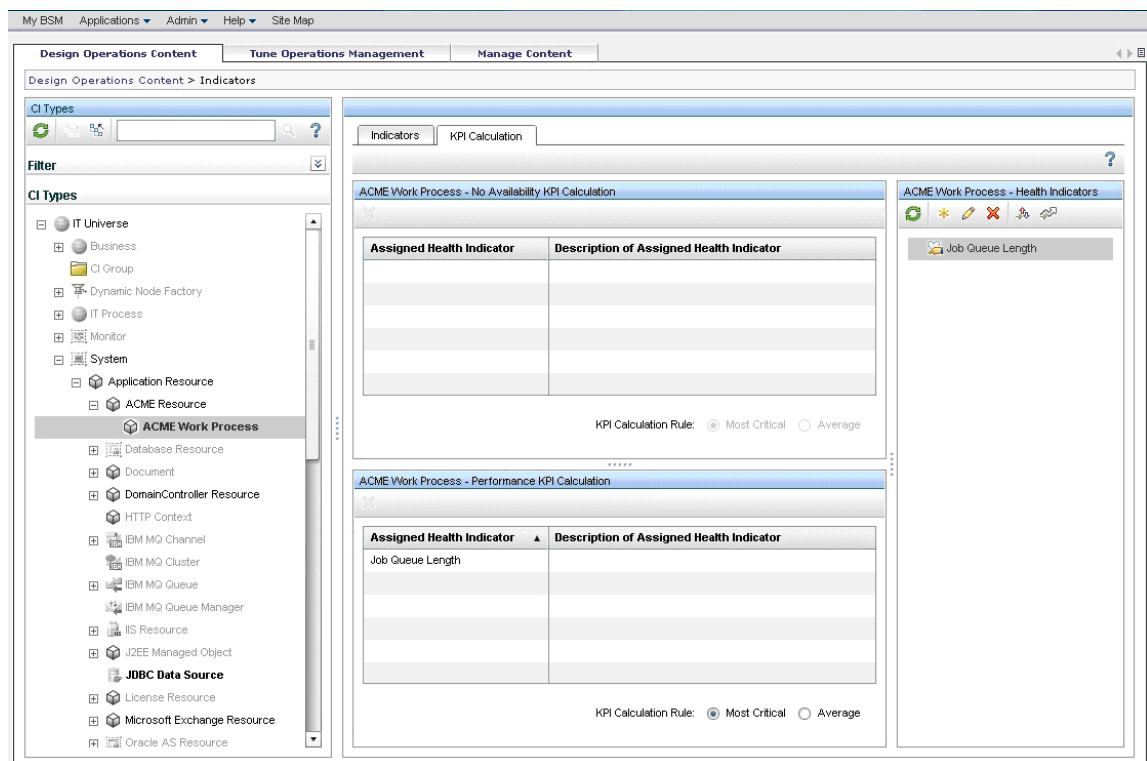
With this mapping rule, you match every incoming message where Category=ACME, Sub Category=Work Process and Title=db connection lost to the ETI Lost Database Connection for CI type ACME Work Process.

## 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 KPI Calculation tab of the HP OMi Indicator Manager.

The HIs from the ACME example, shown in [Table 2](#) on page 28, mostly represent the availability status of the specific CI. Therefore, those HIs are assigned to the Operations Availability KPI.

[Figure 8](#) on page 31 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.



**Figure 8 KPI assignment**



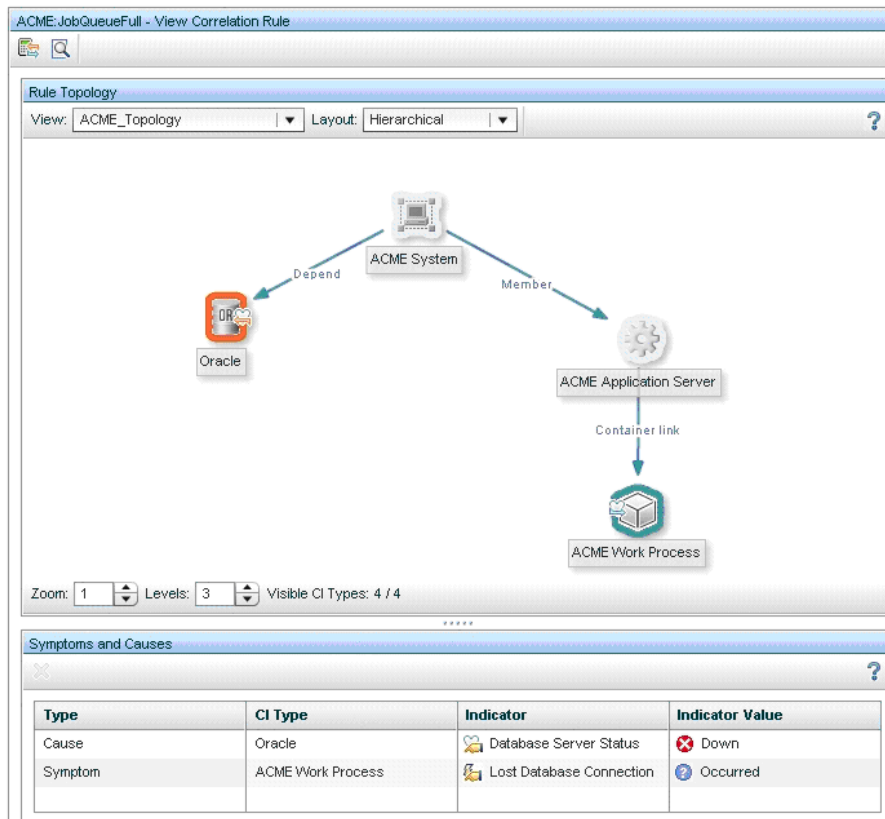


## 5 Correlation Rules

You can define correlation rules that correlate related events occurring in the same or different domains of the managed IT environment. 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.

## 6 HP OMi Tools

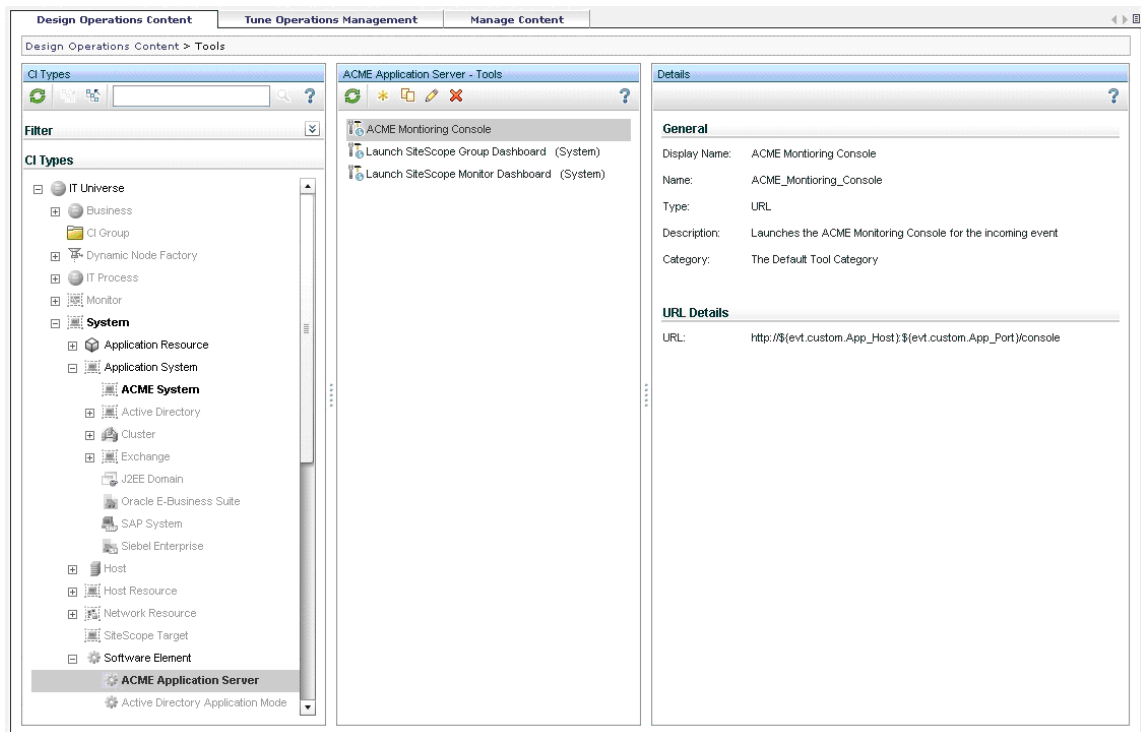
You can configure tools to help domain operators manage and monitor specific events and CIs, and to solve problems that occur in your new application area.

By assigning tools to a particular configuration-item 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 configuration item type.

In the ACME example, we provide a cross-URL launch from the HP OMi browser to the ACME administrator console.

For more details about tools, see the *HP Operations Manager i Concepts Guide*.

**Figure 10** shows an example of an URL launch to the ACME monitoring console. The tool uses the event attributes `App_Host` and `App_Port` to generate the appropriate URL.



**Figure 10** Tool for ACME application server



## 7 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 *HP OMi Online Help*.
- Map suitable graph families or categories to the ACME configuration item types.

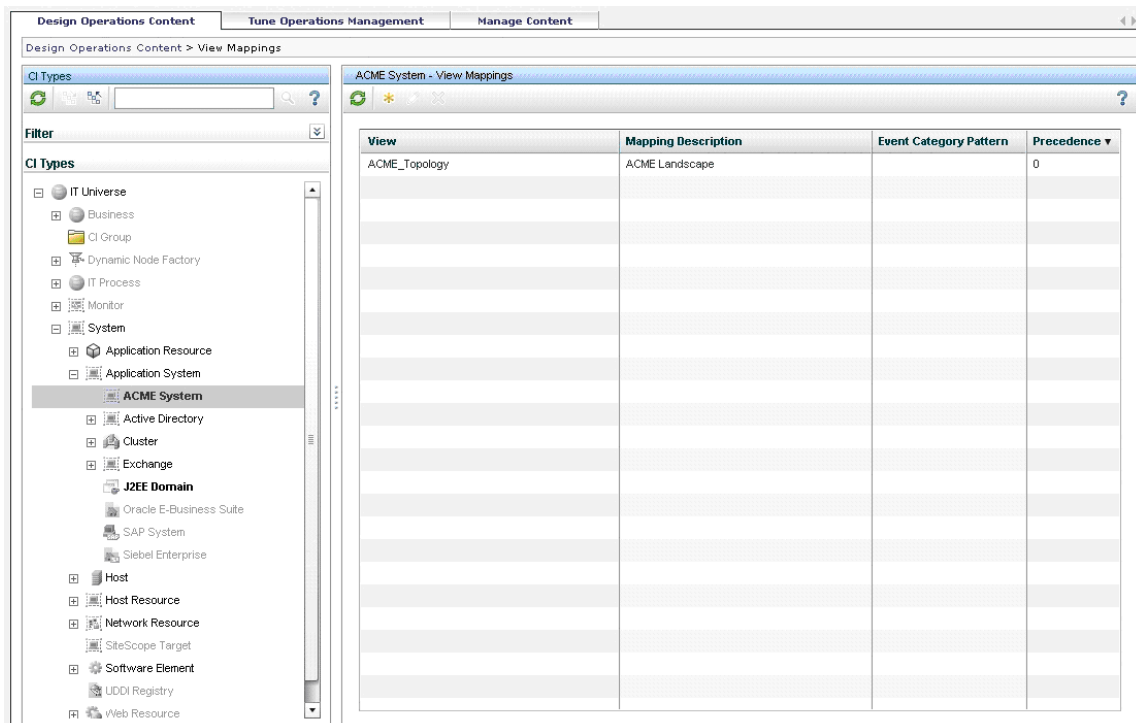


## 8 View Mappings

To ensure that views are available for selection and use in the HP OMi 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 View Manager enables users to create new views in the UCMDB. You can associate configuration item types to more than one view.

With HP OMi's View Mappings manager you can map configuration item types to UCMDB views. Only those views that are mapped to the configuration item type related to the selected event are available for display in the Selected Views drop-down list in the 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 Packaging Content

If you want to use the content you create on another HP OMi system, you need to:

- Create and export the UCMDB package.
- Create and export the ACME HP OMi content pack.
- Copy topology synchronization files.
- Copy graph templates.
- Upload the files to the other HP OMi system.

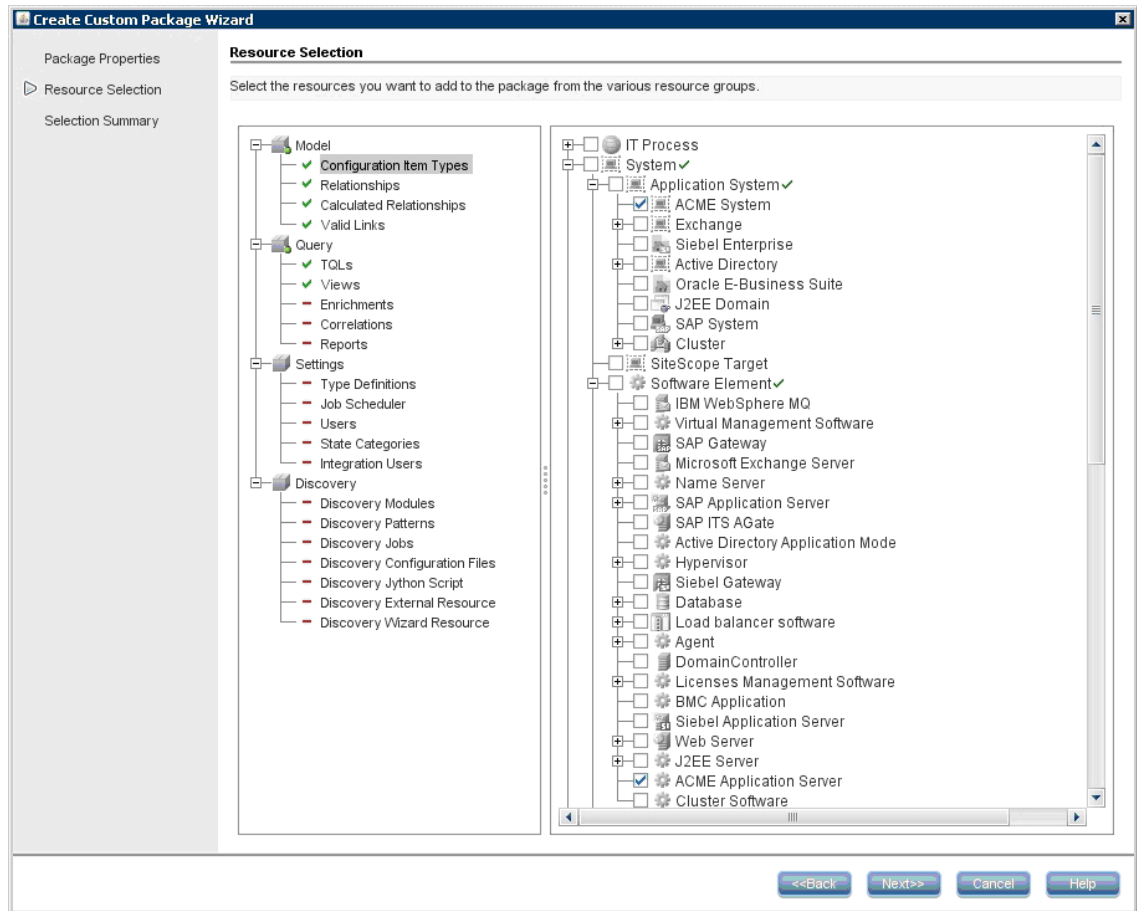
### Create UCMDB Packages

UCMDB packages are essential parts of HP OMi content. You can use UCMDB 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 an HP OMi server.

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

- Create the ACME UCMDB package.  
For more information about how to create an HP OMi content pack, see Chapter 7 of the *HP Operations Manager i Concepts Guide*.
- 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 42 shows the selected items which are part of the ACME UCMDB package. Usually CI types, TQLs and views are part of such a package.



**Figure 12 ACME UCMDB package selection**

## Save Topology Synchronization Files

Collect and save the topology synchronization files so that you can copy them to another HP OMi system.

You can find topology synchronization files in the following location:

```
%TOPAZ_HOME%\conf\opr\topology-sync\sync-packages
```

For more details about how to create topology synchronization mapping rules, see [Section II, Topology Synchronization](#) on page 47.

## Save Graph Templates

Collect and save the graph templates, described in [Chapter 7, Graphs](#) on page 37, so that you can copy them to another HP OMi system.

To migrate the graph templates from one HP OMi system to another, do the following:

- 1 Copy the graph template files (VPI\_Graphs\*.txt) from the directory %TOPAZ\_HOME%\opr\newconfig\OVPM on the source system to the same directory %TOPAZ\_HOME%\opr\newconfig\OVPM on the host system.
- 2 Execute the following script on the target HP OMi system.

**<OvBinDir>/pmiuploadtemplates.bat**

where *OvBinDir* is:

- %OvInstallDir%\bin on 32 bit HP OMi servers
- %OvInstallDir%\bin/win64 on 64 bit HP OMi servers.

The script `pmiuploadtemplates.bat` uploads the graph templates to the database tables.

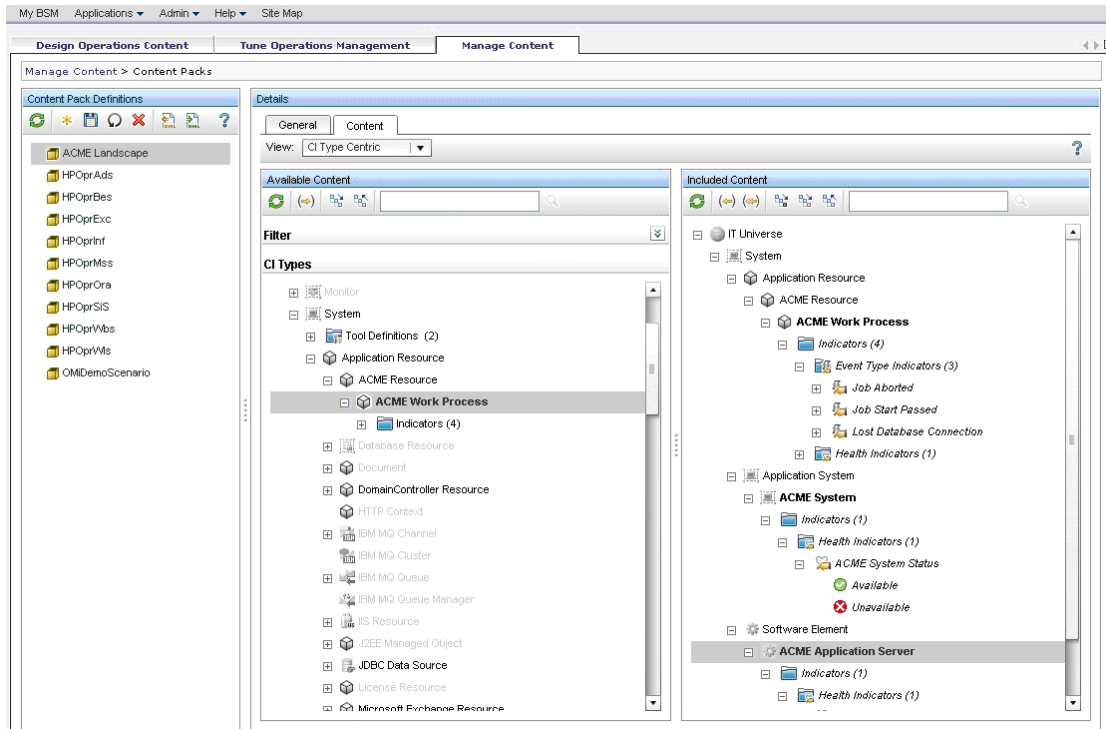
## Create an HP OMi Content Pack

A content pack can contain a complete snapshot of all (or any part of) the HP OMi rules, tools, mappings, and assignments that you define and configure to help users manage your IT environment with HP OMi.

The workflow for creating an ACME content pack is summarized below:

- Create the ACME content pack.  
For more information about how to create an HP OMi content pack, see Chapter 7 of the *HP Operations Manager i Concepts Guide*.
- Add all correlation rules, HIs, ETIs, tool and graph assignments, and view mappings that you created in previous steps.
- Export the content pack XML file (ACME.xml, for the ACME environment content pack) using the content manager. HP OMi enables you to exchange content between instances of HP OMi 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 HP OMi.

Figure 13 on page 44 shows a selection in the content manager within the HP OMi ACME environment content pack.



**Figure 13 ACME environment content pack selection**

# Upload the Content

Copy the exported files to another HP OMi system, and upload them to the other system in the following sequence:

- Upload the UCMDB packages.
- Copy topology synchronization files.
- Copy graph templates.
- Upload and apply the HP OMi content packs.

## Upload UCMDB Packages

To upload an UCMDB package:

- 1 Place the zip-file in a suitable directory of your choice in the file system.
- 2 Use the UCMDB package manager to upload the UCMDB package.

## Copy Topology Synchronization Files

Copy the topology synchronization files to the following location on the destination HP OMi system:

```
%TOPAZ_HOME%\conf\opr\topology-sync\sync-packages
```

## Copy Graph Templates

Copy the graph templates to the following location on the destination HP OMi system:

```
%TOPAZ_HOME%\opr\newconfig\OVPM
```

## Upload HP OMi Content Packs

To upload an HP OMi content pack:

- 1 Place the exported XML file in a suitable directory of your choice on the file system.
- 2 With HP OMi administrator rights, apply the content pack using the content manager.



## Section II: Topology Synchronization

This section provides information for developers to create their own topology synchronization mapping rules to populate the Universal Configuration Management Database (UCMDB) 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 contains the following chapters:

- [Chapter 10, Topology Synchronization Overview](#), page 49
- [Chapter 11, Synchronization Packages](#), page 57
- [Chapter 12, Scripting](#), page 67
- [Chapter 13, Testing, Deployment, and Troubleshooting](#), page 73
- [Appendix A, Mapping Engine and Syntax](#), page 81





# 10 Topology Synchronization Overview

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 HP Operations Manager i (HP OMi) configuration items (CIs) in the universal configuration management database (UCMDB).

Topology synchronization is a standard part of the Event Management Foundation license. It offers a solution, at no extra cost, to populate the UCMDB with CIs from services 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. Installation setup copies topology synchronization rules to the local file system on the installed machine. Customers can also write their own topology synchronization rules to populate the UCMDB based on their service model.

Topology synchronization is used instead of, or in conjunction with, HP DDM (Discovery and Dependency Mapping) discovery. Topology synchronization utilizes the UCMDB API, and uses Wiseman, JAXB, JDOM, XPath, SPRING, Hibernate, and Groovy scripting technology.

To create CIs in the UCMDB, based on mapping rules, and using the HPOM nodes, node groups and the HPOM service model as sources, topology synchronization does the following:

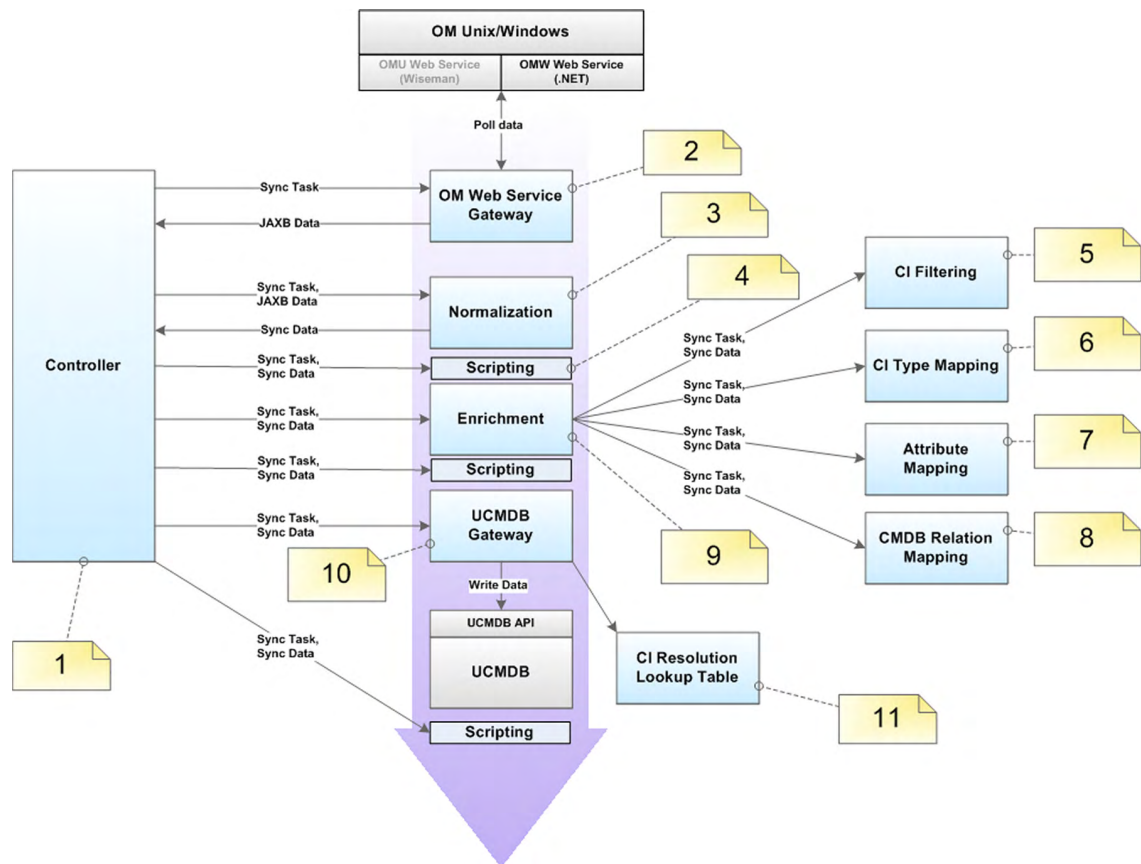
- Transfers the service model hierarchy (infrastructure-based service management data) from HPOM to HP OMi.
- Converts the service management data into a format compatible with HP OMi.
- Uploads the service management data to the UCMDB.
- Updates the data in the UCMDB on demand.
- Provides a mapping table for CI resolution.



The definition of particular CI types is especially important when synchronizing the topologies of HP OMi 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 UCMDB, the topology synchronization process aborts.

# Topology Synchronization Architecture

Figure 14 provides an overview of the topology synchronization architecture.



**Figure 14 Topology Synchronization Architecture**

The numbers in the diagram refer to the following notes:

Reference	Note
1	<ul style="list-style-type: none"> <li>Controls the synchronization process and passes data from one component to the other.</li> <li>Provides an executable class to launch the synchronization from the command line.</li> </ul>
2	Send the request to the HPOM Web Service and return the JAXB objects.
3	Convert the JAXB result to the internal synchronization data structure.
4	Execute Groovy scripts to manipulate synchronization data.
5	Use the mapping engine to specify, which CIs are part of the synchronized model, and which are not.
6	Map STDs for UCMDB types.
7	Convert the JAXB result to the internal synchronization data structure.

Reference	Note
8	<ul style="list-style-type: none"> <li>Create the relations for the UCMDB.</li> <li>Map the root container to the CI to be able to create the CI in the UCMDB according to the model.</li> </ul>
9	Control the whole enrichment process and return the modified sync data.
10	Write the synchronization data into the UCMDB.
11	For each synchronized CI, update a mapping table for the CI Resolution component that maps the HPOM Service ID to the UCMDB CI ID.

## Synchronization Packages and Mapping

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 UCMDB side.

A topology synchronization package consists of XML configuration files (see [Mapping Files](#) on page 60). These files are responsible for transforming HPOM services, nodes and node groups into CIs in the UCMDB, and synchronizing those CIs with data from specified services, nodes, and node groups in HPOM.

HP OMi delivers a set of topology synchronization packages:

- Standard packages provided out-of-the-box as part of the HP OMi installation package.  
For more information, see [Standard Out-of-the-box Synchronization Packages](#) on page 58.
- 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 59.

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 61.

For more information about synchronization packages and mapping, see [Chapter 11, 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 UCMDB.

HP OMi supports the use of Groovy scripts 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 12, 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 UCMDB. 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 UCMDB.

There are situations in which you would not use the mapping table shortcut, for example, where the service tree structure and UCMDB 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 UCMDB. There may be many CIs in the UCMDB 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 HP BAC Infrastructure Settings Manager:

**Infrastructure Settings → Operations Management → Operations Management - CI Resolver Settings → Use Topology Sync Shortcut**

## File Locations

Installation setup copies topology synchronization files to the following locations on the local file system on the HP OMi processing server:

### **Binaries:**

```
%TOPAZ_HOME%\bin\opr-startTopologySync.bat
%TOPAZ_HOME%\opr\lib\opr-ts-*.jar
```

### **Log files:**

```
%TOPAZ_HOME%\log\opr-topologysync
```

### **Log file configuration:**

```
%TOPAZ_HOME%\conf\core\Tools\log4j\opr-topologysync\opr-topologysync.
properties
```

### **Topology Synchronization Packages:**

```
%TOPAZ_HOME%\conf\opr\topology-sync\sync-packages
```

### **Schema files:**

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

# Topology Synchronization Settings

For the successful synchronization of HP OMi and HP Operations Manager topologies, make sure that the following settings are correctly configured:

- **HPOM Connection Settings**

The topology synchronization process needs to read the topology data from the HP Operations Manager web service (WS) during synchronization.

- **UCMDB Connection Settings**

The topology synchronization process needs to write information about new and changed configuration items to the UCMDB during the synchronization.

In the UCMDB Connection Settings, you can specify the following:

- Host: Host of the UCMDB
- Port: Port of the UCMDB connection
- User: User login used to connect to the UCMDB
- Password: Password of the user used to connect to the UCMDB

You can access the UCMDB Connection Settings here:

**Infrastructure Settings → Operations Management → UCMDB Connection Settings**

The OM Topology Synchronization pane contains the settings used by HP OMi to synchronize its topology with HP Operations Manager.

In the OM Topology Synchronization settings, you can:

- Enable or disable dump data
- Enable or disable the use of Groovy Scripts
- Specify which topology synchronization packages to use
- 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 HP OMi and successful synchronization of the object topology in the environments monitored by HP OMi and HP Operations Manager.

You can access the OM Topology Synchronization settings here:

**Infrastructure Settings → Operations Management → OM Topology Synchronization**

You can access the connection settings here:

**Infrastructure Settings → Operations Management → OM Connection Settings**

Infrastructure Settings Manager		
Select Context:		
<input checked="" type="radio"/> Applications	Operations Management	
<input type="radio"/> Foundations	Alerting	
<input type="radio"/> All		
Operations Management - OM Connection Settings		
Name	Description	Value
Enable event forwarding	When this setting is true all events will be forwarded to the defined OM Server	true
Enable server-based Flexible Management	Define if forwarding to HPOM for existing events should be enabled	true
Forwarding retries (number of retries)	This value configures the number of retries for the HPOM forwarder.	2
Forwarding retry interval (in seconds)	This value specifies the the number of seconds that the OM forwarder should wait before trying to reconnect to an unreachable target server	30
OM Host	Hostname that is used for the OM connection	tcvml13.deu.hp.com
OM Password	Password that is used for the OM connection	*****
OM Port	Port that is used for the OM connection	8444
OM Type	Configure if the Operations Manager system is an OM for Windows or OM for Unix	OM for Unix
OM User	Username that is used for the OM connection	opc_admin
Use HTTPS for the OM Webservice	If this setting is set to true, HTTPS is used to connect to the OM Webservice	true
Operations Management - OM Topology Synchronization		
Name	Description	Value
Dump data	If this setting is set to true, all data of the different processing steps is saved to the hard disk. This is not recommended for production systems, as it impacts the performance.	true
Enable usage of Groovy scripts	Use this setting to enable Groovy scripts used to manipulate the synchronization data during the synchronization process.	true
Packages used for Topology Sync	Semicolon separated list of packages, which are used for the topology synchronizations	default;nodegroups;operations-agent
Resolve IPs during synchronization	Use this setting to enable IP resolution for nodes without IP address information in OM. Please be aware, that setting this to true has a negative impact on the synchronization performance.	false

For more information, see the *HP Operations Manager i Installation and Deployment Guide* and the *HP Operations Manager i Online Help*.

## Running Topology Synchronization

You start topology synchronization by running the `opr-startTopologySync.bat` command-line tool, on demand, on the HP Business Availability Center (HP BAC) data processing server.



The binary file to use if you want to set up a topology-synchronization task in the Windows scheduler is:

```
%TOPAZ_HOME%\opr\bin\startTopologySync.bat
```

You can run the `opr-startTopologySync.bat` tool in two modes:

- Normal mode
- Touch mode

### Normal Mode

The normal mode loads the complete service model and synchronizes all configured data from HPOM to the UCMDB. The normal mode also performs delta detection and deletes elements from the UCMDB that have been deleted in HPOM.

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

```
%TOPAZ_HOME%\bin\opr-startTopologySync.bat
```

## Touch Mode

The touch mode prevents aging in the UCMDB by touching all elements from the previous synchronization. In touch mode, no new CIs are created in the UCMDB, and no CIs are deleted.

To run the `opr-startTopologySync.bat` tool in touch mode, run the following command:

```
%TOPAZ_HOME%\bin\opr-startTopologySync.bat -touch
```

For more details about UCMDB aging, refer to the *HP Business Availability Center Model Management* guide.





# 11 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 UCMDB.

The chapter is structured as follows:

[Synchronization Packages Overview](#), see [page 57](#)

[Standard Out-of-the-box Synchronization Packages](#), see [page 58](#)

[Additional Out-of-the-box Synchronization Packages](#), see [page 59](#)

[Package Descriptor File: package.xml](#), see [page 59](#)

[Mapping Files](#), see [page 60](#)

[Configuring Topology Synchronization: ACME Example](#), see [page 61](#)

[Customizing Synchronization and Scripting](#), see [page 66](#)

[Synchronization Package Locations](#), see [page 66](#)

## Synchronization Packages Overview

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 UCMDB 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 UCMDB.
- Synchronize CIs in the UCMDB 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 59](#)).

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 60](#).

For basic information on mapping, see [Appendix A, 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 68.

## Standard Out-of-the-box Synchronization Packages

HP OMi enables you to specify the content you want to update when synchronizing the topology between HP OMi and HP Operations Manager.

HP OMi provides 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 UCMDB.
  - 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, in addition to the host CI itself, and relates it to the host CI.
- **nodegroups**

In addition to creating the host CI itself, maps HPOM node groups to the UCMDB CI type `ci_group`, creates instances of the CI type `ci_group`, and creates relations for the contained nodes.

In the **Packages used for Topology Sync** setting in OM Topology Synchronization settings, you can list the packages whose contents HP OMi updates during the topology synchronization process:

**Infrastructure Settings → Operations Management → OM Topology Synchronization → Packages used 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:

```
%TOPAZ_HOME%\conf\opr\topology-sync\sync-packages
```

Additional topology synchronization packages are provided in the HP OMi content packs.

# Additional Out-of-the-box Synchronization Packages

Additional topology synchronization packages are provided out-of-the-box in HP OMi content packs. Content packs include the following:

- ActiveDirectory
- Exchange
- MS SQL Server
- Oracle
- WebSphere
- WebLogic
- Cluster
- Virtualization

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 HP BAC Infrastructure Settings:

**Infrastructure Settings → Operations Management → OM Topology Synchronization → Packages used for TopoSync**

Topology synchronization packages are written to the following directory:

```
%TOPAZ_HOME%\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 58), 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.

## 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
- 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 UCMDB 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 UCMDB.

### 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 UCMDB.

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`

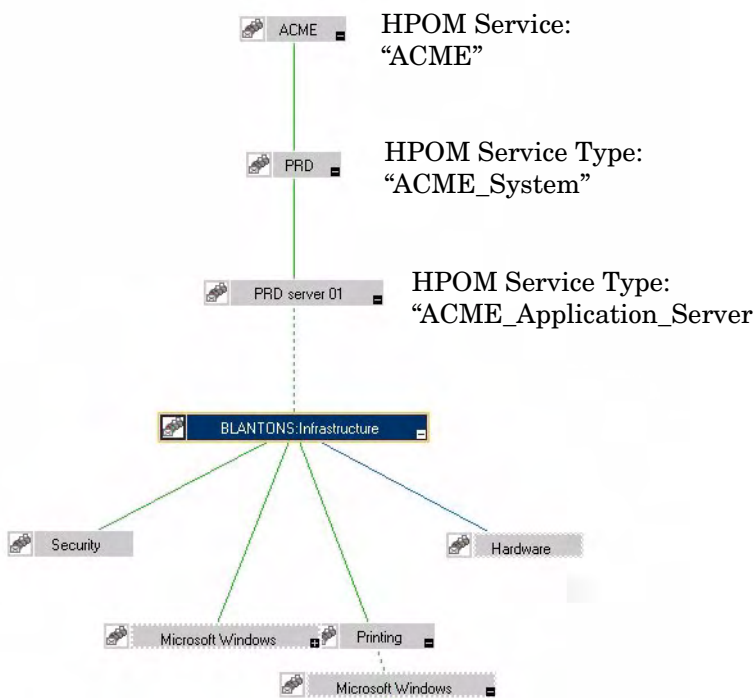
Using the relation mapping file `relationmapping.xml`, you can define the CI relationships created in the UCMDB between specified HPOM services.

Make sure that the specified HPOM services are created as CIs in the UCMDB, otherwise it is not possible for topology synchronization to create a relationship in the UCMDB.

# 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 15 shows a service tree from HPOM discovery.



**Figure 15 Service Tree from HPOM Discovery**

## Configure Package Descriptor File: package.xml

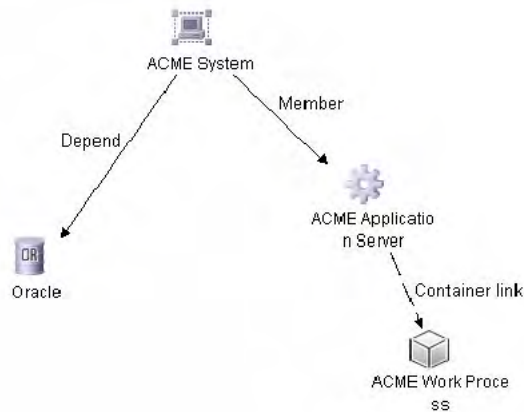
Configure the file `package.xml` to define the name and provide a description of your topology synchronization package, together with a priority level. For the ACME topology synchronization package, the `package.xml` file looks like this:

```
<Package>
  <Name>ACME</Name>
  <Description>Service to UCMDB 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 UCMDB. The mapping rules contained in other configuration files are applied to the tagged services.

Figure 16 on page 62 represents the view for ACME in the UCMDB.



**Figure 16 ACME View in the UCMDB**

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 UCMDB:

```

<?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 UCMDB.

For the ACME example, the type mapping is defined in [Table 3](#):

**Table 3    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 UCMDB. HPOM service elements of type `ACME_Application_Server` are mapped to CIs of type `acme_appserver` in the UCMDB.

```
<?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 UCMDB.

For the ACME example, [Table 4](#) shows which HPOM service attributes are mapped to which CI attributes in the UCMDB.

**Table 4 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 UCMDB.
- 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 UCMDB.

```
<?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 UCMDB 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 containment 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 UCMDB. One pre-mapping, one pre-upload, and one post-upload script can be associated with each synchronization package.

For details, see [Chapter 12, 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:

```
%TOPAZ_HOME%\conf\opr\topology-sync\sync-packages\<SyncPackageName>
```

## 12 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 UCMDB.
- Post-upload scripts are executed after the upload of data from HPOM nodes and services to the UCMDB.

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 66.

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 HP OMi host can edit scripts. This protects the scripts by the log-on security of the HP OMi host.

# Groovy Scripts

Groovy is an object-oriented programming language for the Java platform. It can be used as a scripting language for the Java platform. HP OMi supports Groovy for its scripting capabilities.

Groovy uses a Java-like curly bracket syntax that is dynamically compiled to JVM bytecodes, and that works well with other Java code and libraries. 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 are identified using fixed names within synchronization package directories. Each script runs at a defined point of the synchronization process:

- `preEnrichment.groovy` — script to be executed before mapping  
HP OMi runs the `preEnrichment.groovy` script before starting the topology synchronization's mapping process.
- `preUpload.groovy` — script to be executed before upload  
HP OMi runs the `preUpload.groovy` script after the mapping process but before writing any data to the UCMDB, for example, to create additional CIs or add extra details to existing CI instances.
- `postUpload.groovy` — script to be executed after upload  
HP OMi runs the `postUpload.groovy` script after saving the uploaded data in the UCMDB, 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 OM 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 OM Topology Synchronization settings in the HP BAC Infrastructure Settings Manager:

**Infrastructure Settings → Operations Management → Operations Management - OM 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:

%TOPAZ\_HOME%/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.

### **syncData**

Object Type: `com.hp.opr.ts.common.data.sync.SyncData`

Description: Provides access to the data that is synchronized.

For more information about the interfaces required for developing your own scripts, see the document *HP Operations Manager i Topology Synchronization API Documentation*, which you can download from the following web site:

**<http://h20230.www2.hp.com/selfsolve/manuals>**

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

**Table 5    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");
}
}
```





# 13 Testing, Deployment, and Troubleshooting

This chapter contains information on:

- [Validating XML Configuration Files](#) on page 73
- [Dumping Synchronization Data](#) on page 76
- [Writing Rules](#) on page 79
- [Troubleshooting, Common Issues, and Tips](#) on page 80

## 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. HP OMi provides XSD files 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>.

### HP OMi XSD Files

The HP OMi schema files are stored in the following directory:

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

The files are:

<b>package.xsd</b>	Validates the <code>package.xml</code> file in each synchronization package.
<b>containmentrelations.xsd</b>	Validates the <code>containmentrelations.xml</code> file.
<b>datadump.xsd</b>	Validates synchronization data files that are created through enabling data dumps or used as input for the enrichment simulator.
<b>mapping.xsd</b>	Validates the following mapping files contained in the synchronization packages: <ul style="list-style-type: none"><li>• Context mapping - <code>contextmapping.xml</code></li><li>• Type mapping - <code>typemapping.xml</code></li><li>• Attribute mapping - <code>attributemapping.xml</code></li></ul>

- 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 by HP OMi. 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

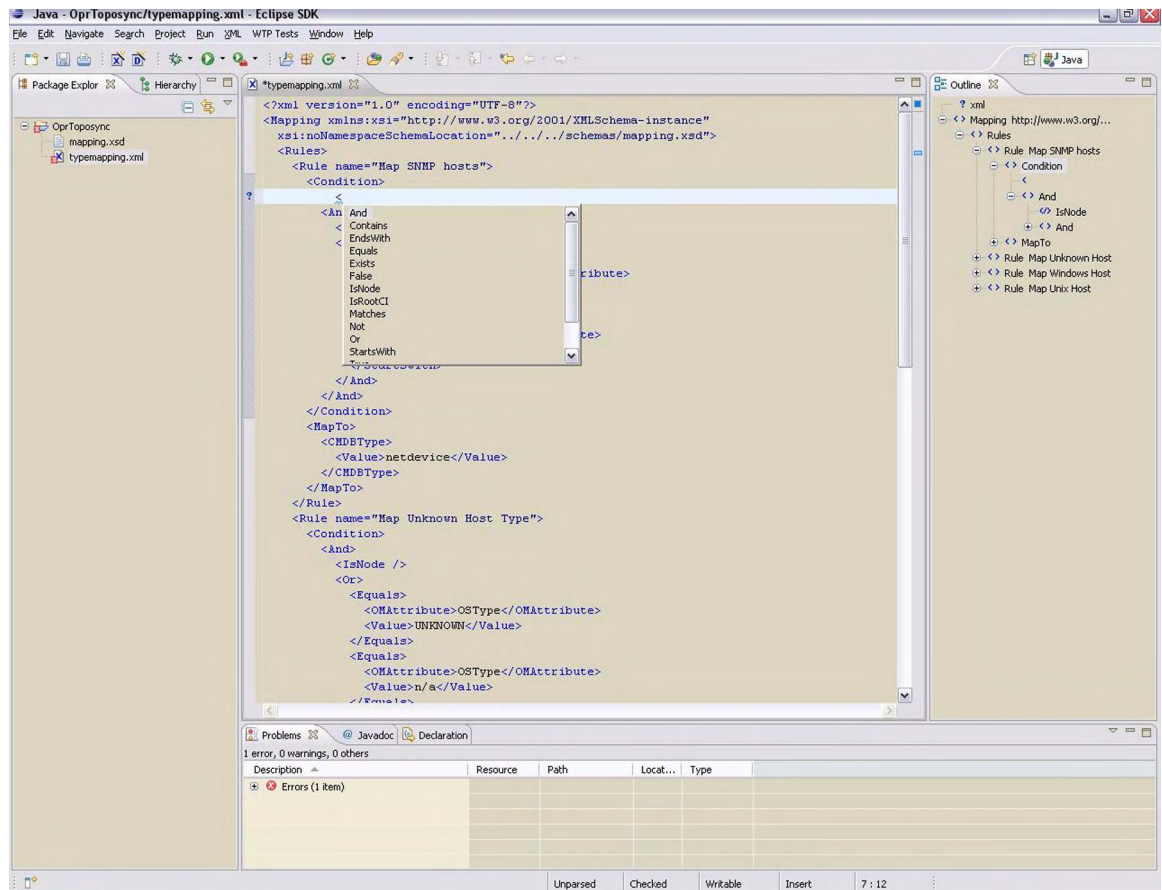
With a modern XML editor, you can validate a file against a schema. Eclipse, for example, can validate an XML file against a schema, if the top level element of the document contains a reference to an XSD file. To enable validation, add the following attributes to the top level element of an XML file:

```
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="%TOPAZ_HOME%/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 17](#) on page 75 for an example.



**Figure 17 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 UCMDB 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 OM Topology Synchronization settings in the HP BAC Infrastructure Settings Manager:

**Infrastructure Settings → Operations Management → OM Topology Synchronization → Dump data**

- 2 Change the value of **Dump data** to **true**.
- 3 Run the Topology Sync tool with the following command:

```
%TOPAZ_HOME%\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:

```
%TOPAZ_HOME%\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 UCMDB.
- **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`.

## 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 73 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.

## Troubleshooting, Common Issues, and Tips

The log files in %TOPAZ\_HOME%\log\opr-topologysync are a good starting point for troubleshooting.

The most common issues are listed in [Table 6](#).

**Table 6 Troubleshooting Common Issues**

Symptom	Cause	Solution
Topology synchronization fails.	Patch OMW_00029 and OMW_00032 for HPOM for Windows were not installed.	Install Patch OMW_00029 and OMW_00032 for HPOM for Windows.
	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.
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 UCMDB exception in the log file.	Data dump option is not enabled.	Enable the data dump option and check if the file in the %TOPAZ_HOME%\opr\tmp\datadump\post-enrichment directory contains all expected attributes for the CIs of your synchronization package.



# A Mapping Engine and Syntax

Topology synchronization leverages the mapping engine from HP Operations Manager Dependency Mapping Automation (HPOM DMA).

Mapping is the mechanism used to map CIs from the UCMDB to services, attributes, or nodes within HPOM. 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](#) on page 81
- [Mapping File Syntax](#) on page 82
- [XPath Navigation](#) on page 98

## 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 82.

# 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 may 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.

## 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.

## Operand Elements

### Operations Manager Service ID

<OMId/>

Return type: String

Returns the OM ID string of the CI as stored in Operations Manager. 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 Operations Manager. 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 UCMDB. 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 UCMDB or BAC.

### 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 UCMDB. 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 backslash in the CI caption by an underscore, you must declare the following:

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

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

For more information on applicable regular expressions, see:

**<http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html>**

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

```
<Replace regExp="true">
  <In>
    <Attribute>host_dnsname</Attribute>
  </In>
  <For>
    <Value>^[^\.]*\.[^\.]*</Value>
  </For>
  <By>
    <Value>$1</Value>
  </By>
</Replace>
```

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

## XPath Result

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

Return type: String

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

For more information on XPath, see [XPath Navigation](#) on page 98.

## 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 98.

## 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)]
```



```

</DescendantCIList>

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

```

**Return type:** List of CIs

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

**Optional elements:**

Operator:            Only CIs that match the operator will be returned.  
relationType:        Only follow relations with the specified relation type.

### **Dependency CI**

```

<DependencyCI>
    [Operator]
</DependencyCI>

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

```

**Return type:** CI

Returns the first dependency CI that matched the included operator.

**Optional elements:**

relationType:        Only follow relations with the specified relation type.

### **Dependency CI List**

```

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

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

```

**Return type:** CI

Returns the list of dependencies.

**Optional elements:**

Operator:            Only CIs that match the operator will be returned.  
relationType:        The dependency must have the specified relation type.

### **Dependent CI**

```

<DependentCI>
    [Operator]
</DependentCI>

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

```

**Return type:** CI

Returns the first dependent CI that matched the included operator.

Example for a dependent CI:

ServiceA → hosted\_on → HostB

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

Optional elements:

relationType: Only follow relations with the specified relation type.

### Dependent CI List

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

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

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.

## 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 UCMDB.

### Mapping to String values in the UCMDB

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetValue>
    [Operands]
  </SetValue>
</CMDBAttribute>
```

### Mapping to Integer values in the UCMDB

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

### Mapping to Boolean values in the UCMDB

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

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.

**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 18](#).

### CI Data Structure

<b>OMAttributes</b>	Contains a map of all original UCMDB CI attributes. The key of this map is the name of the UCMDB CI attribute that references the UCMDB value of the UCMDB CI attribute.
<b>Caption</b>	Represents the name of the CI to be displayed in Service Navigator. Caption has the same value as the UCMDB CI attribute <code>display_label</code> .
<b>Children</b>	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.
<b>Dependencies</b>	References a list of relations to dependent CIs. Similar to <code>Children</code> . However, referenced objects are contained in a different hierarchy.
<b>OMid</b>	Unique ID of a CI.
<b>Node</b>	Boolean value that indicates whether this is a node in HPOM.
<b>Type</b>	Contains the service type of an HPOM service.  This is not the display label of the CI Type.
<b>Service</b>	Boolean value that indicates whether this element a service in HPOM.  Node groups have Node and Service set to FALSE.

### Relationship Data Structure

<b>CI</b>	Contains the reference to the CI to which the current CI is related.
<b>RelationType</b>	Relationship type as stored in the UCMDB.  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.

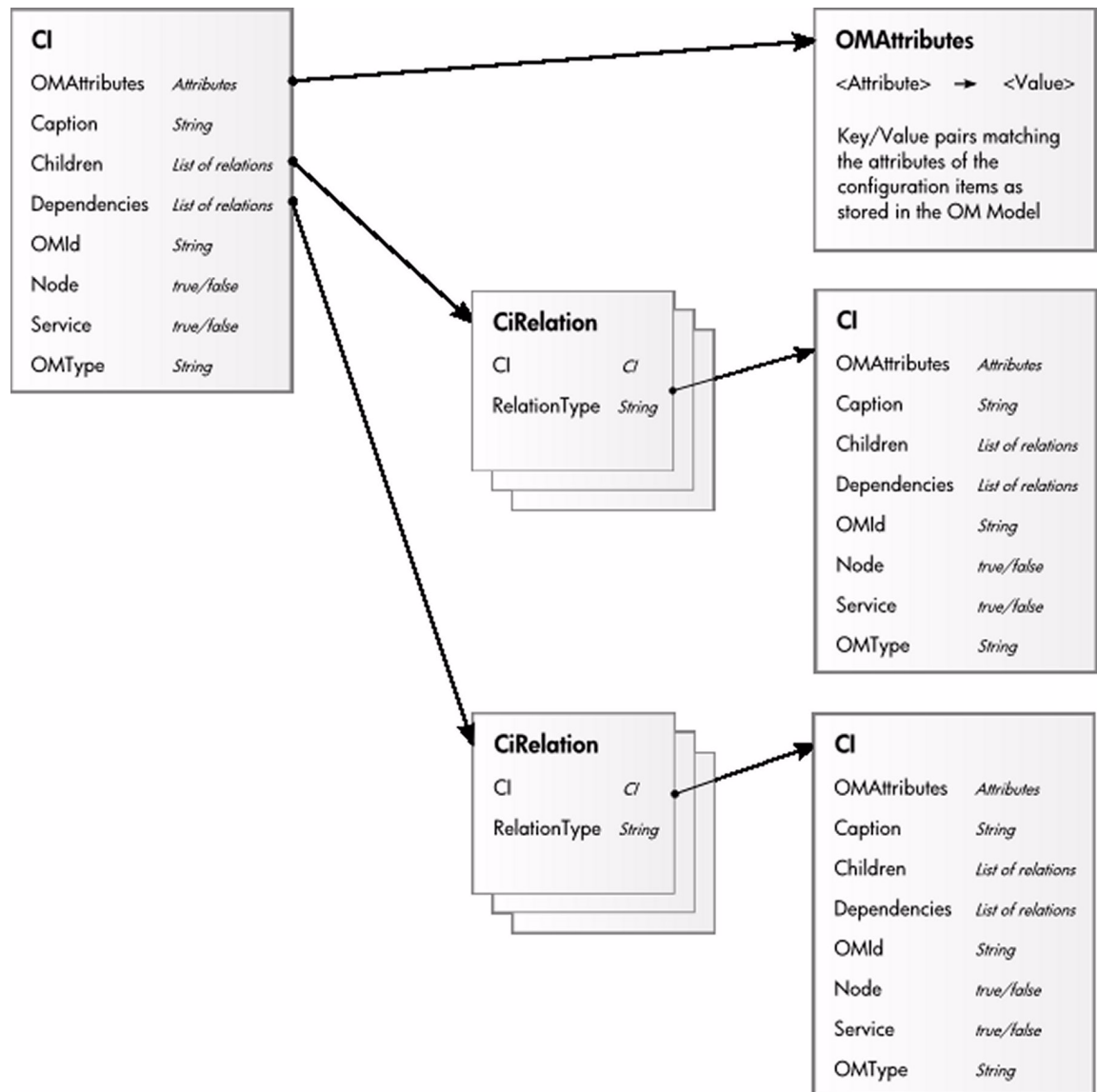
Contains `hosted_on` if it is a hosted on relation in HPOM.

*For nodes:*

Contains `container_node` or `dependency_node`.

Figure 18 illustrates the data structure exposed to the navigation.

**Figure 18 Data Structure Exposed to the Navigation**

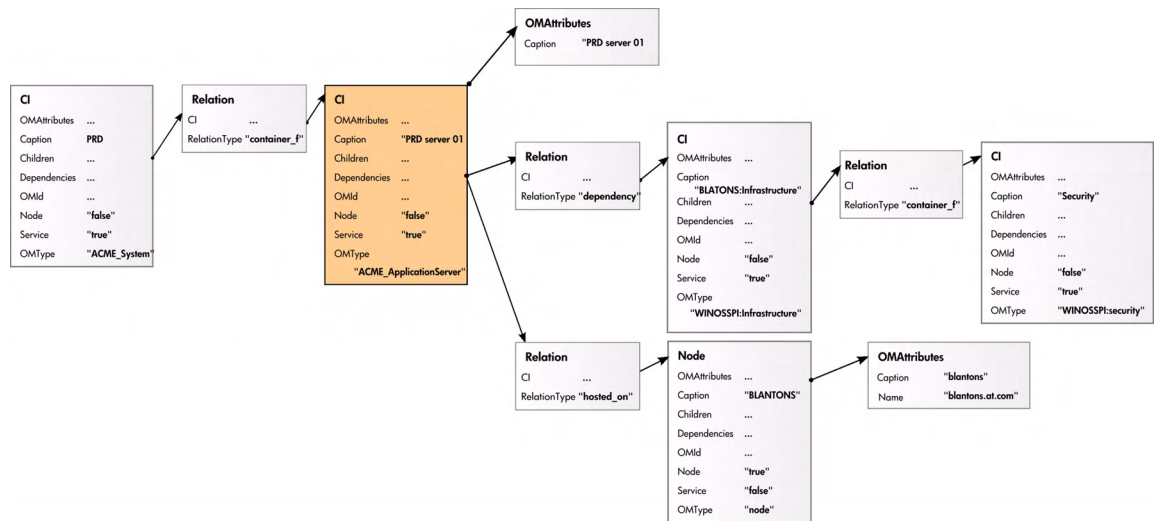


## Example of an XPath-Navigated Data Structure

An example of an XPath-navigated data structure is shown in [Figure 19](#). 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 19](#) illustrates some XPath examples.

**Figure 19 XPath Examples**



## XPath Expressions and Example Values

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

**Table 7 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.



# Index

## A

- ACME
  - example
    - configuring topology synchronization, 61
  - topology model, 16
- additional synchronization packages, 59
- And operator
  - element, 83
  - usage, 79
- attributemapping.xml
  - configuring, 64
- attribute mapping file
  - configuring, 64
- attributes
  - create new, 19
  - matching, 79
  - setting key values for CI types, 18
- Attributes field, 98

## C

- Caption field, 98
- Caption operand element, 85
- Children field, 98
- CiInfo
  - best practice for setting, 26
  - values, 27
    - hosted on CIs, 27
    - virtual CIs, 27
- CI resolution, 52
  - mapping table, 52

## CIs

- CiInfo values
  - for hosted on CIs, 27
  - for virtual CIs, 27
- creating, 21
  - using HP DDM discovery, 21
  - using HPOM service model, 21
  - using topology synchronization, 21
- creating relationships in UCMDB, 21
- data structure, 98
- mapping events to, 25
- matching types, 79

## CI types

- create new attribute, 19
- creating new, 16
- set key attributes, 18

## CMA

- EventTypeIndicator
  - setting ETIs with, 29
- setting CiInfo variable
  - CiInfo
    - setting CMA variable, 26

- common mapping file format, 81

- comparing files, 78

## Condition operator element

- description, 82
- example, 82, 91

## conditions

- examples, 82, 91
- rules, 82

- containmentrelations.xsd file, 73

- Contains operator element, 84

- content
  - creating content pack, 43
    - workflow, 43
  - creating UCMDB packages, 41
  - migration
    - saving topology synchronization files, 42
    - uploading, 45
    - uploading content packs, 45
    - uploading graph templates, 45
  - migration steps, 41
  - packaging, 41
  - packs
    - uploading, 45
  - upload, 45
    - UCMDB packages, 45
  - uploading content packs, 45
  - using on another HP OMi system, 41
  - workflow for uploading, 45
- content development, 11 to 45
- content pack, 43
  - creating, 43
  - workflow for creating, 43
  - XML file, 43
- content packs, 59
  - uploading, 45
- context mapping, 60
- contextmapping.xml
  - configuring, 61, 62
- context mapping file
  - configuring, 61
- correlation
  - rules, 14
- correlation rules, 14
  - ACME landscape, 33
- creating
  - synchronization data dumps, 76
- creating new CI types, 16
- custom message attribute. *See* CMA

## D

- datadump.xsd file, 73
- data structure, 98
- default synchronization package, 58
- Dependencies field, 98
- directories
  - synchronization data dumps, 78
  - XSD files, 73

- discovery
  - choosing a method, 22
  - HP DDM, 21
- domain names, 86
- dumping synchronization data, 76 to 78

## E

- Eclipse, 75
- editors, XML. *See* XML editors
- elements
  - mapping, 91
  - operand, 85 to 87
  - operator, 82 to 85
- Ends With operator element, 84
- enriched directory, 78
- enrichment rules, 23
- Equals operator element, 84
- error messages, 74
- ETIs, 13, 25
  - assigning to an event, 29
  - overview, 28
  - setting with mapping rules, 29, 30
    - filter configuration example, 29
    - rule configuration example, 30
- events
  - analysis, 28
  - assigning ETIs to, 29
  - defining indicators, 28
  - mapping to CIs, 25
- EventTypeIndicator
  - setting ETIs with, 29
- event type indicators. *See* ETIs
- examples
  - conditions, 82, 91
  - XML file validation, 75
  - XPath, 100
- Exists operator element, 83
- expressions
  - regular, 84, 86
  - relative, 79

## F

- False operator element, 82
- figures
  - data structure, 99
  - examples
    - file validation, 75
    - XPath, 100



file location  
    topology synchronization, 42, 45

file locations, 52  
    Groovy scripts, 68

files  
    comparing, 78  
    XML  
        validating, 73  
    XSD, 73

filtering, 60

## G

graphs, 14  
    templates  
        saving, 42  
        uploading, 45  
graph templates  
    uploading, 45  
Groovy scripts, 58  
    location, 68  
    postUpload.groovy, 68  
    preEnrichment.groovy, 68  
    preUpload.groovy, 68

## H

Health Indicators. *See* HIs

HIs, 13, 25  
    assigning to KPIs, 31  
    overview, 28

hosted on CIs  
    CiInfo values, 27

HP DDM  
    creating CIs using, 21

HPOM  
    service model  
        creating CIs using, 21

HPOM connection settings, 53

HP-UX operating system, 100

## I

id field, 98

impact propagation, 24

indicators  
    defining, 28  
    event type, 13, 28  
    health, 13, 28

Is Node operator element, 83

## J

Java data structure, 98

## K

key attributes, CI type, 18

key performance indicators. *See* KPIs

KPIs, 25  
    assigning HIs to, 31

## L

List operand element, 87

locations  
    synchronization data dumps, 78  
    synchronization packages  
        sync-packages directory, 66  
    XSD files, 73

logs, error, 74

## M

mapping  
    elements, 91  
    events to CIs, 25  
    file  
        format, 81  
        syntax, 82 to 91  
    rules  
        setting ETIs with, 30  
        setting ETIs with, 29  
        validating, 78  
    syntax, 82 to 101

mapping.xsd file, 73

mapping files, 60  
    attribute  
        attributemapping.xml, 60  
    context  
        contextmapping.xml, 60  
    relation  
        relationmapping.xml, 60  
    topology synchronization, 57  
    type  
        typemapping.xml, 60

Matches operator element, 84

matching attributes, 79

messages, error, 74

## N

navigation, XPath, 98

Node field, 98

- nodegroups synchronization package, 58
- nodetypes.xsd file, 74
- non-matching XPath expressions, 79
- normalized directory, 78
- normal mode
  - topology synchronization, 54
- Not operator element, 83

**O**

- OM Connection Settings, 53
- OM Topology Synchronization settings, 53
- OMType
  - operator element, 79
- operand elements, 85 to 87
- operations-agent synchronization package, 58
- operator elements, 82 to 85
- opr-startTopologySync.bat
  - executing, 54
  - normal mode, 54
  - touch mode, 55
- Oracle applications, 100
- Or operator element, 83

**P**

- package.xml, configuring, 61
- package.xml file
  - usage, 57, 59
- package.xsd file, 73
- package descriptor file
  - configuring, 61
- packages
  - synchronization
    - mapping, 51
  - UCMDB, 41
  - upload, 45
  - workflow for creating, 41
- postUpload.groovy, 68
- post-upload scripts, 67
- preEnrichment.groovy, 68
- pre-mapping scripts, 67
- prerequisites (for using this guide), 10
- preUpload.groovy, 68
- pre-upload scripts, 67

**Q**

- queries, XPath, 79

**R**

- raw data directory, 78
- regular expressions, 84, 86
- relation data structure, 98
- relationmapping.xml
  - configuring, 65
- relation mapping file
  - configuring, 65
- relations
  - creating, 20
  - creating an impact relation, 24
  - cross-domain, 23
- relative expressions, 79
- Replace operand element, 86
- rules
  - conditions, 82
  - correlation
    - ACME landscape, 33
  - enrichment, 23
  - mapping
    - configuring, 30
    - setting ETIs with, 29
    - validating, 78
  - names, 81
  - writing, 79
- rules, correlation, 14

**S**

- scripting, 51
  - error handling, 70
  - examples, 71
  - naming convention, 68
  - synchronization, 67
  - topology synchronization, 51
  - variables and scope, 69
- scripts
  - enabling and disabling, 68
  - Groovy
    - location, 68
    - postUpload.groovy, 68
    - preEnrichment.groovy, 68
    - preUpload.groovy, 68
  - post-upload, 67
  - pre-mapping, 67
  - pre-upload, 67
- selection example, 43

- standard synchronization packages, 58

- Starts With operator element, 84

- structure, data, 98

- synchronization

  - customizing, 67

    - error handling, 70

    - example, 71

    - examples, 71

    - naming convention, 68

    - scope, 69

    - variables, 69

- synchronization data dump

  - creating, 76

  - overview, 76

  - viewing, 78

- synchronization packages, 57

  - additional, 59

  - enabling, 59

  - locations

    - sync-packages directory, 66

  - mapping, 51

  - scripts, 66

  - standard, 58

    - default, 58

    - nodegroups, 58

    - operations-agent, 58

- sync-packages directory, 66

- syntax, mapping

  - files, 82 to 91

## T

- tools, 14, 35

  - ACME application server example, 35

- topology

  - ACME model, 16

  - view, 23

- topology data, 13, 15 to 24

- topology synchronization, 47 to 101

  - architecture, 50

  - configuration

    - ACME example, 61

  - files

    - copying, 45

    - location, 42

    - saving, 42

  - Groovy scripts, 58

  - HPOM connection settings, 53

  - mapping files, 57

  - normal mode, 54

  - overview, 49

  - scripting, 51

  - settings, 53

  - starting, 54

  - synchronization packages, 51, 57

    - additional, 59

    - enabling, 59

    - standard, 58

      - default, 58

      - nodegroups, 58

      - operations-agent, 58

  - touch mode, 55

  - UCMDB connection settings, 53

  - Windows scheduler tasks, 54

- touch mode

  - topology synchronization, 55

- troubleshooting

  - common issues, 80

- True operator element, 82

- Type field, 98

- type mapping

  - definition, 63

  - file

    - configuring, 62

    - typemapping.xml, 62

## U

- UCMDB, 21

  - ACME view in, 62

  - connection settings, 53

  - creating packages, 41

  - packages

    - upload, 45

    - workflow to create, 41

- uploading content, 45

  - content packs, 45

  - copying graph templates, 45

## V

- validating
  - mapping rules, 78
  - XML files
    - automatically, 73
    - manually, 74 to 75
  - XPath Expressions, 78
- Value operand element, 87
- viewing synchronization data dumps, 78
- view mappings, 39
  - ACME example, 39
  - pane, 39
  - View Manager, 39
- views
  - creating, 39
  - topology, 23
- virtual CIs
  - CiInfo values, 27

## W

- W3C, 73
- writing rules, 79

## X

- XML editors
  - validating
    - files, 74
    - XPath expressions, 78
  - writing rules, 79
- XML files
  - content pack, 43
  - root elements, 78
  - synchronized CIs, 76
  - validating configuration, 73
- XML Schema Definition, 73
- XPath
  - examples, 100 to 101
  - expressions, 101
  - matching expressions, 79
  - navigation, 98
  - validating expressions, 78
  - writing queries, 79
- XPathResultList
  - operand element, 87
  - operator element, 79
- XPathResult operand element
  - description, 87
  - usage, 79
- XSD. *See* XML Schema Definition