HP Universal CMDB

Universal Discovery Content Guide – HP Integrations

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notice

Trademark Notices

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

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

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

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

Java and Oracle are registered trademarks of Oracle Corporation and/or its affiliates.

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

UNIX® is a registered trademark of The Open Group.

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

- This product includes OpenLDAP code from OpenLDAP Foundation (http://www.openldap.org/foundation/).

- This product includes GNU code from Free Software Foundation, Inc. (http://www.fsf.org/).This product includes JiBX code from Dennis M. Sosnoski.

- This product includes the XPP3 XMLPull parser included in the distribution and used throughout JiBX, from Extreme! Lab, Indiana University.

- This product includes the Office Look and Feels License from Robert Futrell (http://sourceforge.net/projects/officelnfs).

- This product includes JEP - Java Expression Parser code from Netaphor Software, Inc. (http://www.netaphor.com/home.asp).

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: **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

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

Chapter 1: HP APM Push Integration

This section includes the following:

- "Overview" below

- "Supported Versions" on the next page

- "How to Integrate UCMDB and HP APM" on page 12

- "View UCMDB Data in HP APM" on page 21

- "Customize the Integration" on page 22

- "Developer References" on page 26

- "Troubleshooting and Limitations" on page 34

Overview

The integration between HP APM and HP Universal CMDB (UCMDB) enables you to share information from UCMDB with HP APM.

You can use the integration to automate the creation and update of requests in HP APM, freeing you from repetitive and manual input of information in HP APM. This also ensures that HP APM is kept up to date with real, accurate, discovered data in your environment.

The following table provides an overview of the HP APM integration with UCMDB:

Integration direction	From UCMDB to HP APM
Integration technology	Push adapter
Pushed data	CIs created in UCMDB are pushed to HP APM to create requests in HP APM
HP Universal CMDB adapter	HP APM Push Adapter (APMPushAdapter)

How Data is Synchronized Between HP APM and UCMDB

When referring to the concept of data information, it is important to distinguish between a UCMDB CI (Configuration Item) and an HP APM Application. Both are defined in a different Data Model, and there must be a conversion before transferring CIs in UCMDB to Applications in HP APM.

The following graphic shows the high-level components of the integration:
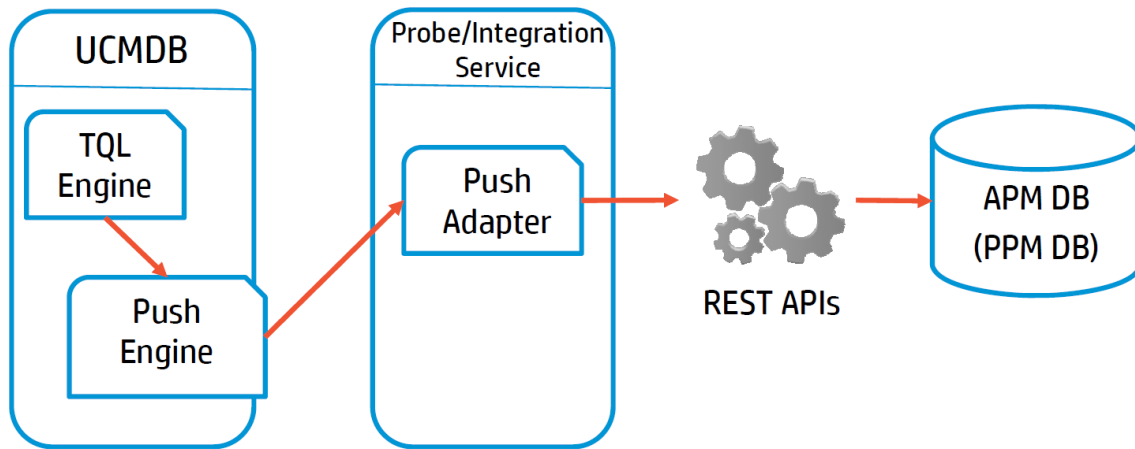
> **Note:** The Push Adapter is executed in the Data Flow Probe/Integration Service process.

UCMDB stores its information using CIs. The integration chooses which data to pull from UCMDB by defining integration TQL queries. Each TQL query defines a superset of data relevant for the integration.

The **UCMDB Push Engine**:

- Retrieves the required data from UCMDB, using the given TQL query.

- Filters the data to include only the data that has changed since the last execution of this synchronization.

- Splits the data into multiple chunks without breaking consistency.

- Sends the information to the Probe/Adapter.

The Push Adapter is a generic framework for easily configuring push adapters, using only XML and Groovy. It allows easy mapping of the data from the UCMDB data model into the HP APM data model, and the transfer of this converted data into the HP APM database through REST APIs called from HP APM.

For more information about push adapter, see **Developing Push Adapters** in the *HP Universal CMDB Developer Reference Guide*.

For details about REST APIs that this integration call from HP APM, see "REST APIs Called in the Integration" on page 32.

For entity mappings and field mappings between HP APM and UCMDB, see "Default Entity and Field Mappings between HP APM and UCMDB" on page 27.

Supported Versions

The HP APM adapter supports the following:

- Universal CMDB version 10.00 and later

- HP Project and Portfolio Management Center (PPM Center) version 9.22 (and later) where HP APM for PPM 9.20 is installed

How to Integrate UCMDB and HP APM

To set up integration between UCMDB and HP APM, you must complete the following steps:

- "Deploy the APM Push Adapter" below

- "Create an Integration Point between HP APM and UCMDB" on the next page

- "(Optional) Push CI Data from UCMDB to HP APM" on page 17

- "Schedule Data Push Jobs" on page 21

Deploy the APM Push Adapter

To integrate HP APM with UCMDB, administrators must deploy the APM Push Adapter.

> **Note:** If your UCMDB version is 10.10, you can skip the steps below and proceed to "Create an Integration Point between HP APM and UCMDB" on the next page. The APM Push Adapter is deployed on UCMDB version 10.10 by default.

To download the APM Push Adapater zip package and deploy it:

1. Obtain the APM Push Adapter from the HP Live Network.

   a. Go to the HP Live Network page for Integrations and Other Content for Discovery and Dependency Mapping: https://hpln.hp.com//node/32/contentfiles?dir=17109

   b. If you have not logged in, the HP Passport sign-in page opens.

      Provide your User ID and Password, and click **Sign-in**.

   c. Click **APM integration package** to download the `APMPushAdapter.zip` package.

2. Start the UCMDB server.

3. Open a browser and log on to UCMDB as an administrator.

   Example of the UCMDB login page URL:
   http://<*UCMDBServerName.Domain*>:8080/ucmdb-ui/login_page.jsp

4. From the left navigation bar, click the **Administration** tab.

5. Click **Package Manager**.

6. Click **Deploy packages to server (from local disk)** ⟐.

   The Deploy Packages to Server dialog opens.

7. Click **Add** ✚.

8. Browse to the folder where `APMPushAdapter.zip` is located. Select the zip file, and click **Open**.

9. Click **Deploy**.

10. Click **OK** when the following confirmation message displays: `Resources were deployed successfully`.

Create an Integration Point between HP APM and UCMDB

1. Log on to UCMDB as an administrator.

2. Navigate to **Data Flow Management > Integration Studio**.

   UCMDB displays a list of existing integration points.

3. Click the **New Integration Point** ✳ button.

   The New Integration Point dialog box opens.

4. Complete the **Integration Properties** and **Adapter Properties** fields as shown in the following table:

Field (*Required)	Description
Integration Properties section	
*Integration Name	Type the name (unique key) of the integration point.
Integration Description	Type a description of the current integration point.
*Adapter	Click the **Select Adapter** ⋯ button and select **HP Software Products > APM > APM Push Adapter** from the Select Adapter dialog.
*Is Integration Activated?	Select this option to indicate the integration point is active.

Field (*Required)	Description
Adapter Properties section	
*Protocol Type	Select **http** or **https** from the drop-down list
*Hostname/IP	Type the hostname or IP Address of the PPM Server. For example, `16.166.16.16` or `hostname`.
*Port	Type the communication port of the PPM Server. The default value is `80`. Example: `30000`
*Path	Type the path of the PPM Server. The default value is `itg`.

Field (*Required)	Description

*Credentials ID	Click the **Select Credential Id** [...] button, then select **Generic Protocal** or **DefaultDomain > Generic Protocal** in the Protocal pane, and from the Credentials list, select a credential or create a new credential that is to be used by UCMDB to access HP APM.
	
	To create a credential for this integration point,
	
	a. Click the **Select Credential Id** [...] button. The Choose Credentials dialog opens.
	
	b. Click the **Create new connection details for selected protocol type** ✳ button. The Generic Protocol Parameters dialog opens.
	
	c. Provide values for the following fields and click **OK**:
	
	   ○ **Network Scope:** Use the default value `ALL`.
	   ○ **User Label:** Type a label for the credential.
	   ○ **User Name:** Provide the user name for the HP APM account that is to be used by UCMDB to access HP APM.
	   ○ **Password:** Click [...] and provide the password for the HP APM account that is to be used by UCMDB to access HP APM.
	
	d. Click **OK** twice.
	
	Tip: Users of any of the following security groups can be used as Credentials ID:
	
	▪ APM User
	▪ APM Administrator
	▪ APM Analyst

Field (*Required)	Description
*Data Flow Probe	The name of the Data Flow Probe/Integration service used to execute the synchronization from. Select **IntegrationService** for this integration. **Note:** If the **IntegrationService** option does not exist, consult with your UCMDB administrator for the best selection for your requirements.
Additional Probes	Not required for this integration point.

Below is an example of the completed dialog:
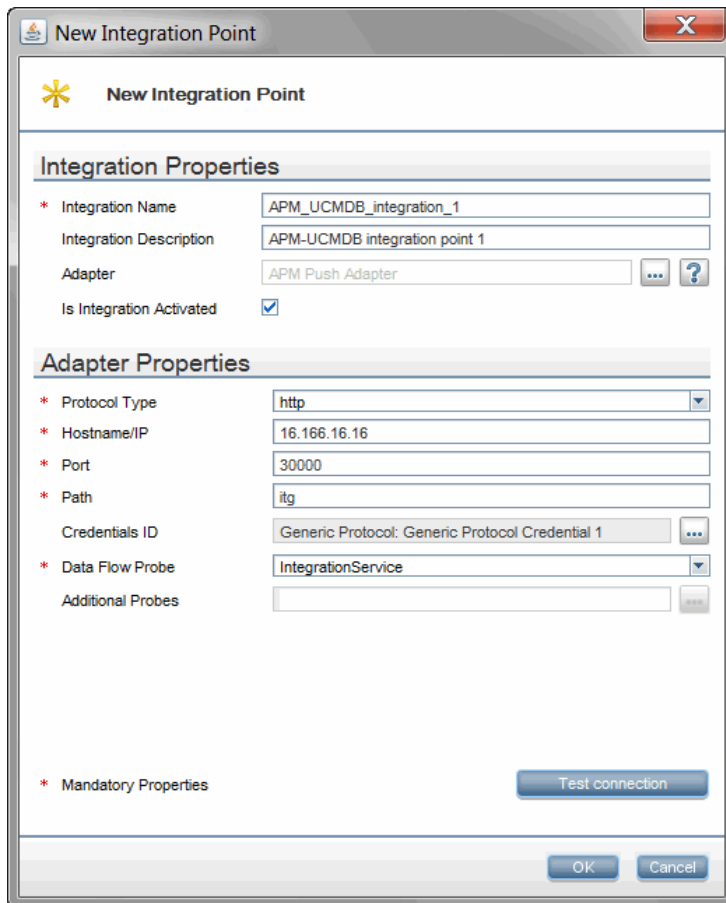
5. Click **Test Connection** to make sure there is a valid connection.

6. Click **OK**.

The integration point is created and its details are displayed. (It is not saved to the server until you click on the **OK** button)

UCMDB creates a default data push job when creating the integration point. The default data push job includes everything, it runs immediately and performs a Full Synchronization.

> **Note:** The first Full Synchronization may take a while to complete.

If needed you may create or edit the existing job. To create or edit and run a customized data push job, see "(Optional) Push CI Data from UCMDB to HP APM" below.

For instructions about scheduling the data push job, see "Schedule Data Push Jobs" on page 21.

(Optional) Push CI Data from UCMDB to HP APM

Data push jobs copy or update CI or CI relationship records from the local UCMDB system to your HP APM system.

To run a customized data push job, complete the following steps:

1. Log on to UCMDB as an administrator.

2. Navigate to **Data Flow Management > Integration Studio**.

   UCMDB displays a list of existing integration points.

3. Select the integration point you created for HP APM.

4. Select the default data push job **APM Push**.

   Or, if the default data push job does not satisfy your needs, you may add a new data push job as follows:

   a. Click the **New Integration Job** ✳ button in the right panel.

   b. In the **Name** field, type a unique name for the job.

   c. Click the **Add Query** ➕ button to add existing TQL queries to the job.

      UCMDB creates a default data push job when creating the integration point for HP APM. The following table lists the Topology Query Language (TQL) queries in the default data push job. If required, you may create, update, or remove TQL queries for the push job. You may also need to update the mapping.

      > **Note:** To access these OOTB TQL queries for push, navigate to **Modelling >**

> **Modeling Studio > Resources**, select **Queries** from the drop-down list for the **Resource Type** field and then navigate to **Root > Integration > APM Push**.

TQL Query	Description
APM Location Push	Pushes Location CIs. Mapping XML: `pushMappingAPMLocation.xml`
APM Process Push	Pushes BusinessProcess CIs. Mapping XML: `pushMappingAPMProcess.xml`
APM Process Relation Clear Push	Clears old relations between processes in HP APM. This TQL query synchronization must have been run BEFORE the 'APM Process Relation Push' TQL query synchronization. Mapping XML: `pushMappingAPMProcessRClear.xml`
APM Process Relation Push	Pushes relations between Processes (pushed by APM Process Push) to other business elements or to processes. BusinessProcess CIs must have been pushed before this TQL query synchronization in the 'APM Process Push' TQL query synchronization. Mapping XML: `pushMappingAPMProcessR.xml`
APM Server Push	Pushes Node CIs (Computers, Network Devices, etc.). Mapping XML: `pushMappingAPMServer.xml`
APM Server Relation Clear Push	Clears old relations between Servers in HP APM. This TQL query synchronization must have been run BEFORE the 'APM Server Relation Push' TQL query synchronization. Mapping XML: `pushMappingAPMServerRClear.xml`

TQL Query	Description
APM Server Relation Push	Pushes relations between Servers (pushed by APM Server Push) to other servers. Node CIs must have been pushed before this TQL query synchronization in the 'APM Server Push' TQL query synchronization. Mapping XML: `pushMappingAPMServerR.xml`
APM Application Push	Pushes BusinessApplication CIs. Location, BusinessProcess, and/or Node CIs must have been pushed before this TQL query synchronization. Mapping XML: `pushMappingAPMApplication.xml`
APM Application Relation Clear Push	Clears old relations (except for the downstream relations) between Applications in HP APM. This TQL query synchronization must have been run BEFORE the 'APM Application Relation Push' TQL query synchronization. Mapping XML: `pushMappingAPMApplicationRClear.xml`
APM Application Relation Push	Pushes relations (except for the downstream relations) between Applications (pushed by APM Application Push) to other business elements or to nodes. BusinessApplication CIs must have been pushed before this TQL query synchronization in the 'APM Application Push' TQL query synchronization. Mapping XML: `pushMappingAPMApplicationR.xml`
APM Application Relation Clear Down Push	Clears old downstream relations between Applications in HP APM. This TQL query synchronization must have been run BEFORE the 'APM Application Relation Down Push' TQL query synchronization. Mapping XML: `pushMappingAPMApplicationDownRClear.xml`

TQL Query	Description
APM Application Relation Down Push	Pushes downstream relations between Applications (pushed by APM Application Push) to other business elements or to nodes. BusinessApplication CIs must have been pushed before this TQL query synchronization in the 'APM Application Push' TQL query synchronization. Mapping XML: `pushMappingAPMApplicationDownR.xml`

d. Select the **Allow Deletion** option for each query.

This allows deletion of synchronized data in HP APM when data in UCMDB are deleted. Otherwise requests created in HP APM as a result of synchronization remain even when their original data in UCMDB are deleted.

> **Note:** For scheduling configuration, see "Schedule Data Push Jobs" on the next page.

e. Click **OK.**

f. Save the integration point.

5. Run the job manually to see if the integration job works properly:

a. To push all the relevant data for the job, click the All Data Synchronization button (or **Full Synchronization**).

b. To push only the changes in the data since the job last executed, click the Changes Synchronization button (or **Delta Synchronization**).

6. The job is in **Running** status. Wait for the job to complete; click the **Refresh** button multiple times as needed until the job is completed.

You can also look at the Query tab as that gives a progress bar against the query it is running.

7. When the job is completed, the job status becomes one of the following depending on the results:

- Succeeded
- Passed with failures
- Failed

8. Click the **Statistics** tab to view the results; if any errors occur, click the **Query Status** tab and **Job Errors** tab for more information. For more information about errors, see "Troubleshooting and Limitations" on page 34.

Schedule Data Push Jobs

UCMDB allows you to schedule job executions directly from a data push job.

1. Log on to UCMDB as an administrator.

2. Navigate to **Data Flow Management > Integration Studio.**

   UCMDB displays a list of existing integration points.

3. Select the integration point you created for the APM - UCMDB integration.

4. Select the APM Push job.

5. Click the **Edit Integration Job** button.

   The Edit Integration Job dialog opens.

   > **Note:** UCMDB allows you to define two different schedules for two types of data push:
   > **Changes Synchronization** (or **Delta Synchronization**) and **All Data Synchronization**
   > (or **Full Synchronization**). It is recommended to use the Changes Sync schedule to only
   > synchronize changes and avoid synchronizing the entire set of data each time.

6. Define a schedule for Changes Synchronization or Delta Synchronization.

   a. Click on the **Changes Synchronization** tab (or **Delta Synchronization**).

   b. Select the **Scheduler enabled** option.

   c. Select the scheduling options you want to use.

7. Click the **All Data Synchronization** tab (or **Full Synchronization**) and select the scheduling
   options you want to use.

8. Click **OK**.

9. Save the integration point.

View UCMDB Data in HP APM

After a push job is successfully completed, you can search for and verify that the pushed
CI/relationship data is in HP APM.

To view UCMDB data in HP APM,

1. Log on to PPM Center.

2. On the **Open** menu, click **Application Portfolio > Search Entities**.

   The Search Entities page opens.

3. In the **Entities** section, click one of the following entities:

   - Application
   - Location
   - Process
   - Server

   The Search: APM - *<Entity>* page opens.

   In this example, click **Application** and the Search: APM - Application page opens.

4. Click **Search.**

   The Search Results page displays request search results.

5. Click any **Application No** to view an APM - *<Entity>* request.

Customize the Integration

This section includes:

- "Overview" below

- "Customize an Existing Mapping" on the next page

- "Add a New Mapping to the Integration" on page 24

Overview

This section contains details about the architecture of the integration.

Data Flow Architecture

1. The Push Engine executes the TQL query.

2. For a differential flow, the data is compared to the last synchronized data, and only the changes are forwarded.

3. Data is converted into Composite CIs (instances of data according to the TQL Root elements).

4. Data is then pushed to the Push Adapter.

5. The Push Adapter loads the correct mapping for the specific TQL query.

6. All dynamic_mappings are executed and saved to maps, to allow usage in the next mapping stage.

   For more information, see **Developing Push Adapters** in the *HP Universal CMDB Developer Reference Guide*.

7. Data is sent to HP APM database via REST APIs from HP APM, where REST APIs converts data to HP APM compatible data.

Integration TQL Queries

A TQL query used for the integration must contain a root query node.

Any attribute used in the mapping flow of the Push Adapter must be marked in the selected layout of the query node. Each TQL query may only have one mapping.

For more information, see **Data Flow Management > Integration Studio > Integration Jobs Pane**.

Customize an Existing Mapping

This example shows you how to add the NAME attribute to the integration including the TQL query and Push Adapter Mapping. It allows the integration to push the NAME attribute to Location in HP APM.

After completing the following steps, you may run the job with the customized mapping:

1. **Add the NAME attribute to the APM Location Push TQL query layout.**

   In this step we add the NAME attribute of the Location to the integration TQL query (APM Location Push TQL) so that we can use the attribute and value in the mapping.

   a. Navigate to **Modeling > Modeling Studio > Resources** and select the **Queries Resource Type**.

   b. Navigate to **Query: Root > Integration > APM Push > APM Location Push**.

   c. Select **Root**, right-click and select **Query Node Properties**.

   d. Go to the Element Layout tab.

   e. Move the **Name** to the Specific Attributes box.

   f. Click **OK**.

   g. Save the Query.

2. **Add the NAME Mapping to the pushMappingAPMLocation.xml push adapter mapping.**

   In this step we take the value from the TQL result and remodel it to the HP APM Data Model.

a. Navigate to **Data Flow Management > Adapter Management > Packages > APMPushAdapter > Configuration Files > pushMapingAPMLocation.xml.**

b. Navigate to the **<target_ci_type name="fields">** XML tag.

c. Below the tag, add the following XML tag to hold the value of the Description:

```
<target_mapping name="REQ.DESCRIPTION" datatype="STRING" value="APMPushFu
nctions.subString(Root['name'],200)"/>
```

where, "REQ.DESCRIPTION" is the request Name field token of Location.

d. Click **OK**.

Add a New Mapping to the Integration

This example shows how to add a new TQL query and push-mapping to the integration. It also shows how to push Locations from UCMDB to HP APM. It consists of the following steps:

Step 1: Create a TQL Query

1. Navigate to **Modeling > Modeling Studio > New > Query**.

2. From the **CI Types** tab, add a **Location** to the query.

3. Right-click the **Location Query Node** and select **Query Node Properties**.

4. Rename the **Element Name** to **Root**.

5. Navigate to the **Element Layout** tab.

6. Select **Select attributes for layout**.

7. In the **Attributes condition** drop down, select **Specific Attributes**, and add the **Name** attribute

8. Click **OK**.

9. Save the query to **Root > Integration > APM Push > APM Location Push**.

Step 2: Create a Push-Mapping

1. Navigate to **Data Flow Management > Adapter Management > APMPushAdapter**.

2. Click the **Create new resource** ✳ button and select **New Configuration File**.

3. Type the following Name: **APMPushAdapter/mappings/pushMappingAPMLocation.xml**.

4. Select the **APMPushAdapter** package.

5. Click **OK**.

6. Copy the following into the newly created XML file:

```xml
<integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <info>
    <source name="UCMDB" versions="10.0" vendor="HP"/>
    <target name="APM" versions="2.0" vendor="HP"/>
 </info>
 <import>
    <scriptFile path="mappings.scripts.APMPushFunctions"/>
 </import>
 <targetcis>
    <source_instance_type query-name="APM Location Push" root-element-nam
e="Root" >
       <target_ci_type name="request">
         <target_mapping name="uuid" datatype="STRING" value="Root['globa
l_id']"/>
         <target_mapping name="requestType" datatype="STRING" value="'APM
- Location'"/>
         <target_mapping name="description" datatype="STRING" value="Root[
'name']"/>
         <target_ci_type name="fields">
           <target_mapping name="REQ.DESCRIPTION" datatype="STRING" value=
"APMPushFunctions.subString(Root['name'],200)"/>
         </target_ci_type>
       </target_ci_type>
    </source_instance_type>
  </targetcis>
</integration>
```

In the following line:

```
<target_mapping name="requestType" datatype="STRING" value="'APM - Locatio
n'"/>
```

The value is the request type name in HP APM. In this example, it is `APM - Location`.

7. Click **OK**.

Step 3: Create a Job with the New TQL Query

1. Navigate to **Data Flow Management > Integration Studio**.

2. Create an Integration Point with HP APM.

3. In the **Integration Jobs** tab, click the **New Integration Job** ✱ button .

4. Insert a job name in the **Name** field.

5. Click the ➕ button, and choose the **APM Location Push query**.

6. Click **OK**.

Step 4: Run the Job

1. Click on the job created in .

2. Click the All Data Synchronization 🔄 button (or **Full Synchronization** 🔄).

3. Wait for the job to finish. You should click the **Refresh** 🔄 button to see progress.

4. Make sure that the status is **Succeeded**.

Step 6: View the Results

1. Log on to .

2. On the **Open** menu,

   - Click **Application Portfolio > Search Entities**, then in the **Entities** section, click **Location**. Or,

   - Click **Search > Requests**, then from the **Request Type** drop-down list, select **APM - Location**.

3. Click **Search**.

   The Search Results page displays request search results.

Developer References

This section includes the following:

- "Adapter" below

- "Default Entity and Field Mappings between HP APM and UCMDB" on the next page

- "REST APIs Called in the Integration" on page 32

Adapter

This integration job uses the adapter called APMAdapter.

Input CI Type

destination_config

Triggered CI Data

Name	Value
adapterId	${ADAPTER.adapter_id}
attributeValues	${SOURCE.attribute_values}
credentialsId	${SOURCE.credentials_id}
destinationId	${SOURCE.destination_id}

Adapter Parameters

Name	Value
credentialsId	
domain	itg
host	
port	80
probeName	
protocalType	http

Default Entity and Field Mappings between HP APM and UCMDB

The following sections describe out of the box mappings that are available with the APM Push Adapter for integration with HP APM and UCMDB.

The following table provides an overview of type mappings between HP APM entities and UCMDB CI Types:

HP APM Entity	PPM Center Request Type	UCMDB CI Type	Remarks
Application	APM - Application	BusinessApplication	For detailed mappings, see "Default Field Mappings between HP APM Application and UCMDB BusinessApplication" on the next page.

HP APM Entity	PPM Center Request Type	UCMDB CI Type	Remarks
Process	APM - Process	BusinessProcess	For detailed mappings, see "Default Field Mappings between HP APM Process and UCMDB BusinessProcess" on page 30.
Location	APM - Location	Location	For detailed field mappings, see "Default Field Mappings between HP APM Location and UCMDB Location" on page 30.
Server	APM - Server	Node	For detailed field mappings, see "Default Field Mappings between HP APM Server and UCMDB Node" on page 31.

Default Field Mappings between HP APM Application and UCMDB BusinessApplication

The following table describes the default field mappings that can be modified for the integration between the HP APM entity of **Application** and the UCMDB CI Type of **BusinessApplication**.

HP APM Field Name and Field Type	UCMDB CI Attribute and Field Type
Name KNTA_PROJECT_NAME Text Field - 300	Name name string
Updated By [a]	(N/A)
Create On [b] CREATION_DATE Date	(N/A)
Purpose APM_APP_PURPOSE Text Field - 4000	Description description string
Business Criticality [c] APM_RATING_BUSINESS_CRIT Drop Down List	BusinessCriticality business_criticality integer
Created By[a] CREATED_BY Auto Complete List	(N/A)

HP APM Field Name and Field Type	UCMDB CI Attribute and Field Type
Supported Processes APM_SUPPORTED_PROCESSES Auto Complete List	Name (of CI Type **BusinessProcess**) name (of CI Type **BusinessProcess**) string
Downstream Applications APM_DOWNSTREAM_APPS Auto Complete List	Name (of downstream CI Type **BusinessApplication**) name (of downstream CI Type **BusinessApplication**) string
Upstream Applications APM_UPSTREAM_APPS Auto Complete List	Name (of upstream CI Type **BusinessApplication**) name (of upstream CI Type **BusinessApplication**) string
Service Level Agreement APM_APP_SLA Text Field - 200	Name (of CI Type **ServiceLevelAgreement**) name (of CI Type **ServiceLevelAgreement**) string
Servers APM_SERVER_LIST Auto Complete List	Name (of CI Type **Node**) name (of CI Type **Node**) string
Database APM_DATABASE_LIST Text Field - 200	Name (of CI Type **Database**) name (of CI Type **Database**) string

a. The HP APM account you provided when creating the integration point (see "Create an Integration Point between HP APM and UCMDB" on page 13).

b. Time when the request is created for the first time in HP APM.

c. The following mapping rule is used for this mapping:
```
<target_mapping name="REQD.APM_RATING_BUSINESS_CRIT" datatype="STRING"
value="APMPushFunctions.getPropertyValue('bc', Root['business_
criticality'].toString() , '')"/>
```
where the definition of 'bc' token is defined as follows in the `server.properties` field:

```
bc.0=0 - Least critical
bc.1=1 - Slightly critical
bc.2=2 - Less than average
bc.3=3 - More than average
bc.4=4 - Critical
bc.5=5 - Highly critical
```

When synchronizing the **Business Criticality** field from UCMDB to HP APM, if the value is '1' in UCMDB, then the field value will be set to '1 - Slightly critical' in APM.

Default Field Mappings between HP APM Process and UCMDB BusinessProcess

The following table describes the default field mappings that can be modified for the integration between the HP APM entity of Process and the UCMDB CI Type of BusinessProcess.

HP APM Field Name, Database ID, and Field Type	UCMDB CI Attribute, Database ID, and Field Type
Process Name DESCRIPTION Text Field - 200	Name name string
Parent Process APM_PARENT Auto Complete List	Name of parent **BusinessProcess** name of parent **BusinessProcess** string
Description APM_DESCRIPTION Text Area - 4000	Description description string
Created By [a] CREATED_BY Auto Complete List	(N/A)
Created On [b] CREATION_DATE Date	(N/A)
a. The HP APM account you provided when creating the integration point (see "Create an Integration Point between HP APM and UCMDB" on page 13)	
b. Time when the request is created for the first time in HP APM.	

Default Field Mappings between HP APM Location and UCMDB Location

The following table describes the default field mappings that can be modified for the integration between the HP APM entity of Location and the UCMDB CI Type of Location.

HP APM Field Name, Database ID, and Field Type	UCMDB CI Attribute, Database ID, and Field Type
Location Name DESCRIPTION Text Field - 200	Name name string

HP APM Field Name, Database ID, and Field Type	UCMDB CI Attribute, Database ID, and Field Type
Description APM_DESCRIPTION Text Field - 4000	Description description string
Address APM_LOC_ADDRESS Text Field - 200	StreetAddress+ExtendedStreetAddress street_address+extended_street_address string+string
Postal Code APM_LOC_ZIPCODE Text Field - 20	PostalCode postal_code string
Country APM_LOC_COUNTRY Text Field - 200	CountryOrArea country_or_area string
Region APM_LOC_REGION DDL	Region region string
Longitude APM_LONGITUDE Text Field - 40	Longitude longitude string
Latitude APM_LATITUDE Text Field - 40	Latitude latitude string
City APM_LATITUDE Text Field - 200	~~City[a]~~ ~~city~~ ~~string~~
State/Province APM_STATE Text Field - 200	~~State[a]~~ ~~state~~ ~~string~~
a. The CI attribute is removed from UCMDB version 10.x, but exists in earlier versions of UCMDB. For UCMDB instances that upgraded from an earlier version to 10.x, this CI attribute exists but is read-only. If the CI attribute has a value, the value can be synchronized to HP APM, otherwise the HP APM field remains empty after you run the synchronization push job in UCMDB.	

Default Field Mappings between HP APM Server and UCMDB Node

The following table describes the default field mappings that can be modified for the integration between the HP APM entity of Server and the UCMDB CI Type of Node.

HP APM Field Name, Token, and Component Type	UCMDB CI Attribute Display Name, Name, and Type
Server Name DESCRIPTION Text Field - 200	Name name string
Description APM_DESCRIPTION Text Area - 4000	Description description string (value size: 1000)
IP Address APM_IP_ADDRESS Text Field, Max Length: 15	IP Address (of **IpAddress**) ip_address (of **IpAddress**) string
OS APM_OS Drop-down List	OsDescription os_description string
Running Software APM_RUNNING_SOFTWARE Text Area - 4000	ProductName:Name (of **RunningSoftware**) product_name:name (of **RunningSoftware**) product_name_enum:string
Location APM_LOCATION Auto Complete List	Name (of **Location**) name (of **Location**) string

REST APIs Called in the Integration

The following HP Demand Management REST APIs are called in this integration to convert data from UCMDB into HP APM compatible data:

- Get a request

  For more information, see **Get Details of a Request** section of the *RESTful Web Services Guide* for 9.20.

- Create a request

  For more information, see **Create/Update a Request** section of the *RESTful Web Services Guide* for 9.20.

- Update a request

  For more information, see **Create/Update a Request** section of the *RESTful Web Services Guide* for 9.20.

- Delete a request

  Added in version 9.22. For more information, see "Delete a Request" on the next page

Delete a Request

Request: `http://<PPM_Server_IP>:<port>/itg/rest/dm/requests/{reqId}`

HTTP Method: DELETE

Description: Delete a request with a specific ID. To perform this operation, you must be in one or more of the authorized security groups for the create/update action.

Request path variables:

Attribute	Description	Required?
reqId	Request ID. Indicates which request will be deleted.	Yes

Response entity body:

- **on success**: When the operation is successfully executed, no message is returned. However, the REST API automatically calls the GET operation and returns the following message in Response Header (with empty message body):

```
Status Code: 204 No Content
Cache-Control: no-cache
Date: Mon, 26 Aug 2013 09:20:29 GMT
Expires: -1
Pragma: no-cache
```

- **on failure**: The following message codes are returned if the operation fails:

Message Code	Message	Cause	Possible Corrective Action
PPMC_WSE108	Not Found	The quest is not found.	The request ID is not passed or the request of this ID doesn't exist.
PPMC_WSE001	Internal Server Error	There was an error when you tried to delete a request.	N/A
PPMC_WSE116	Internal Server Error	There was an error when you tried to delete a request.	N/A

Limitation:

This operation supports field security check, but it ignores user interface (UI) rules or status dependency. Such constraints have to be validated and enforced on the client side before this operation is invoked.

Troubleshooting and Limitations

This section includes the following:

- "Limitations" below

- "Troubleshooting Problems" on the next page

- "Logs" on the next page

Limitations

- The Data Flow Probe or Integration Service must be installed on a Windows OS.

- For requests created in HP APM from CIs pushed from UCMDB, any changes made in HP APM are overwritten when you run the data push job in UCMDB.

- The APM Application request form holds a single value for Location, therefore it is designed to push only one value for Location of the Application from the UCMDB BusinessApplication.

  In the definition of Location of the Application (see the `pushMappingAPMApplicationR.xml` file), the Location of the Server is used for the Location of the Application is used. For an Application that contains multiple servers, select one of the servers and then you can get its Location.

- For value mappings between UCMDB and HP APM, certain mapping rules are followed.

  For example, when synchronizing the Business Criticality field of Application, the following mapping rule is used for the mapping:

  ```
  <target_mapping name="REQD.APM_RATING_BUSINESS_CRIT" datatype="STRING" value="
  APMPushFunctions.getPropertyValue('bc', Root['business_criticality'].toString(
  ) , '')"/>
  ```

  where the definition of 'bc' token is defined as follows in the `<APMPushAdapter.zip>`/mappings/scripts/**server.properties** file:

  ```
  bc.0=0 - Least critical
  bc.1=1 - Slightly critical
  bc.2=2 - Less than average
  bc.3=3 - More than average
  bc.4=4 - Critical
  bc.5=5 - Highly critical
  ```

  When synchronizing the **Business Criticality** field from UCMDB to HP APM, if the value is '1' in UCMDB, then the field value will be set to '1 - Slightly critical' in APM.

If you need to use this value mapping for other fields from UCMDB to APM, make sure you customize the mapping by following the example above.

- This integration does not support synchronizing values in languages that are not supported by UCMDB. For example, Simplified Chinese.

- The "**;**" character is not supported. If a UCMDB CI name contains a "**;**" character, it would be treated as a separator and two entries would show up in HP APM after synchronization.

Troubleshooting Problems

- **Problem:** Some UCMDB CIs include characters that are not supported in APM Entities. For example, semicolon (;).

  Solution: The suggested solution is to modify the content synchronized from UCMDB to APM. You can use the Replace function to replace the unsupported characters for this field mapping in the XML mapping file. However, note that this may cause inconsistent content between UCMDB and APM.

  An example, in the `pushMappingAPMApplication.xml` file, the semicolon character (;) is replaced with a space:

  ```
  <target_mapping name="REQ.KNTA_PROJECT_NAME" datatype="STRING" value="APMPushF
  unctions.stringReplace(Root['name'], ';', ' ')"/>
  ```

- **Problem:** For some fields, the field value lengths between HP APM and UCMDB are different, therefore you may need to customize the field mapping. You can follow the example below:

  Example

  Use a substring as illustrated below to limit the field length to 200 characters:

  ```
  <target_mapping name="REQ.DESCRIPTION" datatype="STRING" value="APMPushFunctio
  ns.subString(APMPushFunctions.stringReplace(Root['name'],';',' '),200)"/>
  ```

Logs

The push adapter framework uses a different logging system for the normal fcmdb.adapters.*.log files.

To change the level of the log files to debug, edit the following file:

- On the Data Flow Probe machine:

  ```
  ..\DataFlowProbe\conf\log\fcmdb.push.properties
  ```

- If using the integration service, on the UCMDB server:

  ```
  ..\UCMDB Server\Integrations\conf\log\fcmdb.push.properties
  ```

Change the log level to DEBUG:

```
loglevel=DEBUG
```

The integration generates `fcmdb.push.*` logs in the following folder:

- On the Data Flow Probe machine:

  ```
  ..\DataFlowProbe\runtime\log\
  ```

- If using the integration service, on the UCMDB server:

  ```
  ..\UCMDB Server\Integrations\runtime\log\
  ```

Chapter 2: HP Asset Manager Integration

This chapter includes:

Overview

HP Asset Manager is an asset lifecycle management solution with modular components allowing an IT organization to measure and communicate the value it provides to the business it supports.

UCMDB-Asset Manager integration is implemented by the Asset Manager adapter (AMAdapter) pulling CIs and relationships from Asset Manager to UCMDB.

> This chapter covers the use of the **Asset Manager 9.02 adapter** and the **Asset Manager 9.02 Update 1 adapter**.
>
> For information about the **Asset Manager Population and Federation adapter 9.30**, see "Populating HP Universal CMDB from HP Asset Manager" in the *HP Service Asset and Configuration Management (SACM) Solution Configuration Guide Version 9.30* . If you have an HP Passport, you can access this document from the HP Software Product Manuals web site (http://support.openview.hp.com/selfsolve/manuals), selecting your product - Universal CMDB (Application Mapping) - version number, and operating system, and searching. To register for an HP passport, go to http://h20229.www2.hp.com/passport-registration.html, or click the **New users - please register** link on the HP Passport login page.

Supported Versions

The Asset Manager adapter supports Asset Manager Versions 5.22 and later.

How to Integrate Asset Manager with UCMDB

> This documentation covers the use of the Asset Manager 9.02 adapter, and the Asset Manager 9.02 Update 1 adapter. There is an *HP SACM Solution Configuration Guide* available for each. You can access this document from HP Live Network (https://hpln.hp.com/) by searching for "HP Solution Content for Asset Manager".

This task consists of the following steps:

1. Prerequisites - Deploying the Asset Manager package

   Ensure the following steps are completed as detailed in "Integrating Asset Manager with HP Universal CMDB" in the *HP SACM Solution Configuration Guide*.

   a. **Create the Asset Manager SQL views**

   b. **Deploying the Asset Manager integration package to HP Universal CMDB**

   c. **Making some CI attributes visible**

d. **Mapping the location types in Asset Manager and HP Universal CMDB**

e. Only relevant for the Asset Manager 9.02 adapter: **Adding the asset_tag attribute**

2. Create the integration point

In the Integration Studio, create a new integration point:

a. Provide a name and description for the integration point.

b. Ensure the **Is Integration Activated** option is enabled.

c. Under **Integration Properties > Adapter**, select **HP Software Products > Asset Manager >**:

   ○ **Asset Manager Adapter 9.02 Update 1** for SACM 9.02 Update 1

   ○ **Asset Manager Adapter** for SACM 9.02 or earlier

d. Under Adapter Properties:

Field	Value
Hostname\IP	<host name or IP address of computer hosting the Asset Manager database>\<name of the instance used by the database>
Port	Type the port to access the Asset Manager database.
Credentials ID	See **Create credentials**, below.
DB Name/SID	Type the database identifier used by Asset Manager.
DB Type	Select the database type. For example: SQL Server.
Data Flow Probe	Select an appropriate probe.
Additional Probes	Leave empty.

e. Create credentials

   i. Click the ellipsis to the right of the **Credentials ID** property.

   ii. Select the **Generic DB Protocol (SQL)** protocol in the left pane.

   iii. Click the **Create new connection details for selected protocol type** ![button] button.

   iv. Populate the fields on the **SQL Protocol Parameters** page, **General** section, as follows:

Field	Value
Network Scope	Use the default value.
User Label	Type a label for the credential.

v. Populate the fields on the **SQL Protocol Parameters** page, **SQL** section, as follows:

Field	Value
Database Type	Select the DBMS type.
Port Number	The port to access the database.
Connection Timeout	The time in milliseconds after which the probe stops trying to connect to the database.
User Name	The name of the user used to connect to the database.
Password	The password of the user needed to connect to the database.
Instance Name	For Oracle, enter the name of the instance.
Encryption Method	Choose **None** or **SSL**. For Oracle, choose **None**.
Trust Store File Path	Enter the full path to the SSL trust store file. **Note:** Only available if you have chosen an encryption method.
Trust Store Password	The SSL trust store password. **Note:** Only available if you have chosen an encryption method.

Note: For details about integration points and credentials, see the *HP Universal CMDB Data Flow Management Guide*.

    f. Click **Test Connection** to verify the connection is successfully established.

    g. Click OK.

    h. Save the integration point.

3. Run the appropriate integration data flow

- Population

- Federation

> **Note:** For details on running these, see "Integrating Asset Manager with HP Universal CMDB" in the *HP SACM Solution Configuration Guide.*

Adapter

This section contains details about the adapter called **AMAdapter** which the job uses.

Input CIT

destination_config

Triggered CI Data

Name	Value
adapterId	${ADAPTER.adapter_id}
attributeValues	${SOURCE.attribute_values}
credentialsId	${SOURCE.credentials_id}
destinationId	${SOURCE.destination_id}

Parameters

Name	Value
dbname	AssetManager
dbtype	SQLServer
port	1433

Troubleshooting and Limitations

When running a diff population job, ensure **amComputer.dtLastModif** for related CIs is updated.

For more information, see the *HP SACM Solution Configuration Guide.*

Chapter 3: HP Asset Manager Push Integration

This chapter includes:

Quick Start

> **Note:** This section is only for **advanced users** who want to start using Asset Manager Push Integration quickly, without reading the full documentation. It therefore provides the minimum information required before you run your first integration.

Before starting the integration for the first time, you must complete the following:

- "Validate Pre-Loaded Data in Asset Manager" on page 45

- "Update Asset Manager Schema" on page 47

- "Create AMPushAdapterAPI Package with Required AM API Files" on page 49

- "Install a Database Client" on page 50

Overview

Integration between HP Universal CMDB (UCMDB) and HP Asset Manager enables you to share information from UCMDB with Asset Manager. Common use cases include Hardware, Installed Software and Business Services.

You can use the integration to automate the creation and update of Asset and Portfolio information in Asset Manager. This ensures Asset Manager is kept up to date with real, accurate, discovered data in your environment.

> **Note:** This Integration replaces the Connect-It Scenarios used for syncing Hardware and Software information from DDMI 9.3x and below to Asset Manager. Also, this integration replaces the Connect-It Scenarios used to sync Business Services and Business Applications from UCMDB to Asset Manager.

How Data is Synchronized Between UCMDB and Asset Manager

When referring to the concept of data information, it is important to distinguish between a UCMDB **CI** (Configuration Item) and an Asset Manager **Asset**. Both are defined in a different Data Model, and there must be a conversion before transferring CIs in UCMDB to Assets in Asset Manager.

The following graphic shows the high-level components of the integration:
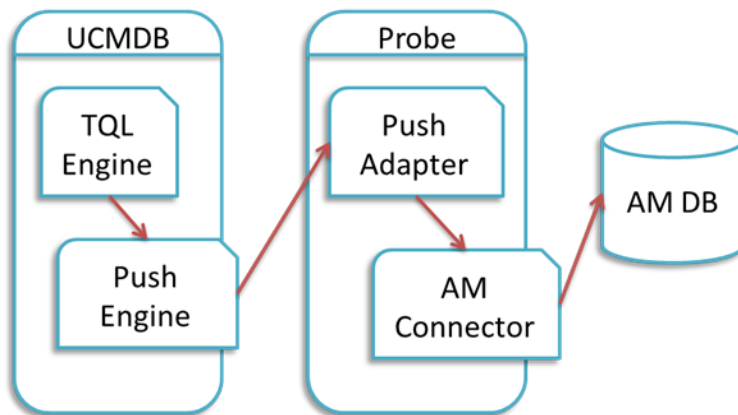
> **Note:** The Push Adapter and AM Connector are executed in the Data Flow Probe/Integration Service process.

UCMDB stores its information using CIs. The integration chooses which data to pull from UCMDB by defining integration TQL queries. Each TQL query defines a superset of data relevant for the integration.

The **UCMDB Push Engine:**

- Retrieves the required data from UCMDB, using the given TQL query.

- Filters the data to include only the data that has changed since the last execution of this synchronization.

- Splits the data into multiple chunks without breaking consistency.

- Sends the information to the Probe/Adapter

The **Push Adapter** is a generic framework for easily configuring push adapters, using only XML and Groovy[1]. It allows easy mapping of the data from the UCMDB data model into the Asset Manager Data model, and the transfer of this converted data into the AM Connector. For more information, see **Developing Push Adapters** in the *HP Universal CMDB Developer Reference Guide*.

The **AM Connector** is a component that connects to the Push Adapter, built specifically to reconcile, push, and handle the complex logic needed to synchronize data into Asset Manager.

Supported Versions

This integration supports HP Asset Manager versions 5.20, 5.21, 5.22, 9.30, 9.31 and later.

---

[1]Groovy is an agile and dynamic language, natively supported by the Java Virtual Machine. It allows simple scripting capabilities, while maintaining all the strengths and capabilities of Java. It can execute simple String manipulation, and use 3rd party libraries. For more information, see http://groovy.codehaus.org/

How to Integrate UCMDB and Asset Manager

To set up integration between UCMDB and Asset Manager, you must complete the following steps:

- "Validate Pre-Loaded Data in Asset Manager" below

- "Set Up Asset Manager" below

- "Set Up UCMDB" on page 49

- "Push CI Data from UCMDB to Asset Manager" on page 52

Validate Pre-Loaded Data in Asset Manager

For the integration to succeed, it requires there to be some basic data already in the Asset Manager database.

This data may either be imported during the database creation (using the Asset Manager Application Designer), or may be added later. For more information, see the **Asset Manager Documentation** – **Administration**.

For **hardware synchronization** the required data is:

- Shared Data

- UNSPSC Product Classification

- Portfolio – Line-of-business data

- Virtualization – Line-of-business data

- Business services management – Line-of-business data

For **software synchronization** the required data is:

- Software Asset Management – Line-of-business data

Set Up Asset Manager

To set up Asset Manager you must complete the following steps:

Create an Account with Administrative Rights

For the integration, any user with administrative rights will suffice. Asset Manager OOTB installations include an administrator account.

The details of the default Administrator user are:

- **User:** Admin

- **Password:** <empty>

The following example shows how to create a new user (named Integration-Admin) with administration rights, specifically for the integration.

1. Log on to Asset Manager as an administrator.

2. Go to **Organization Management > User actions > Add a user**.

    a. In **ID #**, type: **Integration-Admin**.

    b. In **Name**, type: **Integration-Admin**.

    c. In **First**, type: **Integration-Admin**.

    d. Click **Next**.

    e. Click **Next**.

    f. Click **Finish**.

3. Go to **Organization Management > Organization > Employees**.

4. Select the newly created User and go to the **Profile** tab.

5. In **User name**, type: **Integration-Admin**.

6. In **Password**, type: **<A password you would like to use>**.

7. In the **Password Administration** pane, ensure **Never Expires** is selected.

8. In the **Profile** pane, ensure **Administration rights** is selected.

9. Click **Modify**.

Update Asset Manager Schema

The default Asset Manager database schema includes column lengths that may be significantly shorter than their counterparts in the UCMDB database schema. For attributes used for reconciliation, this may be critical and may cause creation of multiple records.

To fix this issue, you are recommended to change the **Asset Manager Column Sizes** to the values shown in the following table of Asset Manager Attributes.

Table	Name	Default Max Length	New Value	New Value as of AM 9.31[*]
amComputer	**TcpIpHostName**	40	255	255
amComputer	**WorkGroup**	40	250	250
amPortfolio	**Folder**	128	250	250
amAsset	**SerialNo**	36	250	250
amModel	**Name**	80	250	250
amCompany	**Name**	30	100	100
amSoftInstall	**Folder**	128	255	255
amSoftInstall	**Field1**	26	255	255
amSoftInstall	**TechnicalInfo**	128	255	255
amBrand	**Name**	64	250	250
amEmplDept	**UserName**	100	200	200
amEmplDept	**UserDomain**	100	200	200
amBrand	**FullName**			500
amComputer	**FullName**			500
amModel	**FullName**			500

[*] Also available in Asset Manager versions 5.22 and 9.3 with an appropriate hotfix for column length limit.

> **Note:**
>
> - The new values in the table are only a suggestion, and you may need to change them according to actual data per customer use case.
>
> - DB2 default table space page size of 4K may be too small in some cases; using 8K or higher is recommended.

Prepare Asset Manager for Parallel Push

Enabling Parallel Push significantly increases the performance of the push. However, some advance preparation is necessary. Different actions are needed for different database types, as shown in the following table:

Database	Action
DB2	*Mandatory*: follow **Eliminating locks and deadlocks** in the *Asset Manager Tuning Guide*. **Limitation**: DB2 parallel push is not supported in UCMDB 10.01.
Oracle	*Optional*: follow **Eliminating locks and deadlocks** in the *Asset Manager Tuning Guide*.

SQL Server	The following are all mandatory steps:
	
	 1. Alter the **SQLServer Schema** isolation level:
	
	 Execute the following command on the database, replacing **<AMSchema>** with the real schema name:
	
	 <pre>ALTER DATABASE <AMSchema> SET READ_COMMITTED_SNAPSHOT ON
	ALTER DATABASE <AMSchema> SET ALLOW_SNAPSHOT_ISOLATION ON
	GO</pre>
	 Note: If the execution takes too long, you may need to disconnect all connections to the database. One possible way is to restart the database service, then execute the command. If you restart the SQL Server, and an Integration Point has already been created in UCMDB, you should restart the UCMDB Probe as well to avoid the issue of dead connections.
	
	 2. Alter Asset Manager database options:
	
	    a. Open the Asset Manager Client and connect to the appropriate database schema
	
	    b. Navigate on the top menus to **Administration > Database options**
	
	    c. For option **'Sql Server specifics'|'Isolation command before starting a write transaction'** change the current value to **set transaction isolation level snapshot**
	
	 3. Follow **Eliminating locks and deadlocks** in the *Asset Manager Tuning Guide*.

Set Up UCMDB

To set up UCMDB you must complete the following steps:

Create AMPushAdapterAPI Package with Required AM API Files

> **Note:** If you want to create several integrations to **different versions of Asset Manager**, follow this procedure for each version of Asset Manager you want to integrate to. Otherwise, see "Integrating to a Single Version of Asset Manager" below.

In order for the adapter to connect to the appropriate Asset Manager version, you must supply the Data Flow Probe/Integration Service with the appropriate Asset Manager API DLLs and Jars, as follows:

1. Copy the files below:

    - **<Asset Manager Installation folder>\x64*.dll**

    - **<Asset Manager Installation folder>\websvc\lib*.jar**

2. Create a package called **AMPushAdapterAPI_<AM Version Number>.zip**. For example, for version 9.3 the package is **AMPushAdapterAPI_9.3.zip**.

3. Paste the copied files to:

    <AMPushAdapterAPI_{AM Version Number}.zip>\discoveryResources\AMPushAdapter\amVersion\<AM Version Number>

    For example, for version 9.3 the path is:

    AMPushAdapterAPI_9.3.zip\discoveryResources\AMPushAdapter\amVersion\9.3

4. Add the subfolder **AMPushAdapter/amVersion/<AM Version Number>/*.*** to the **additionalClasspath** property in the **globalSettings.xml** file.

5. Deploy the **AMPushAdapterAPI_<AM Version Number>.zip** package.

Integrating to a Single Version of Asset Manager

If you want to create an integration to only a single version of Asset Manager, take the following steps instead:

1. Copy these files

    <Asset Manager Installation folder>\x64*.dll

    <Asset Manager Installation folder>\websvc\lib*.jar

2. Paste the copied files to the Data Flow Probe\Integration service\lib directory.

3. Restart the probe.

Install a Database Client

You must install database client software according to the type of database the Asset Manager schema is installed on, as detailed in the following table:

Database	Client Software
DB2	1. Download and Install "IBM Data Server Client" 64 bit for windows on your Data Flow Probe/Integration Service computer. This may be downloaded from: http://www-01.ibm.com/support/docview.wss?rs=4020&uid=swg21385217 2. Create a connection to the DB2 database of Asset Manager. You may create this using the DB2 Control Center. **Note:** Remember the Database Alias you define in the connection, because you need it when creating the integration point. 3. Copy the **db2cli64.dll** file from the DB2 client bin directory (By default: C:\Program Files\IBM\SQLLIB\BIN) to the **<Data Flow Probe/Integration Service>\lib** folder. 4. Restart the Data Flow Probe/Integration Service.
Oracle	**Oracle client windows 64 bit** For example: Oracle Database 11g Release 2 Client (11.2.0.1.0) for Microsoft Windows (x64). 1. Download the client installation from: http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html 2. Install the client, in Administrator mode, on the Data Flow Probe/Integration Service computer. 3. Copy **oci.dll** from the <**Oracle Client installation directory**> into: **<Probe/Integration Service installation directory>\lib**. 4. Restart the Data Flow Probe/Integration Service.
SQL Server	None required.

Create an Integration Point in UCMDB

1. Log on to UCMDB as an administrator.

2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.

3. Click the ⊛ button. The New Integration Point dialog box is displayed.

4. Complete the Integration and Adapter Properties fields as shown in the following table:

Field	Required	Description
Integration Name	Yes	Type the name (unique key) of the integration point.
Integration Description	No	Type a description of the current integration point.
Adapter	Yes	Select **HP Software Products > Asset Manager > Asset Manager Push Adapter**
Is Integration Activated	Yes	Select this option to indicate the integration point is active.
Hostname/IP	Yes	Type the hostname or IP Address of the Asset Manager database.
DB Type	Yes	Select the database type your Asset Manager schema is located on.
DB Port	Yes	Type the communication port of the Asset Manager Data Base.
DB Name/SID	Yes	**DB2**: type in the name of the Database Alias you defined in the database connection. **Oracle**: type the service name. **SQL Server**: type the name of the schema.

Field	Required	Description
Credentials ID	Yes	Select **Asset Manager Protocol**. To create a new protocol, click the [...] button. Under **Asset Manager Protocol** complete: ▪ **Asset Manager User Name:** An AM administrator's user name. ▪ **Asset Manager Password:** An AM administrator's password. ▪ **DB User Name:** The AM database user's name. ▪ **DB Password:** The AM database user's password.
AM Version	Yes	Select the version of Asset Manager this integration point is to connect to.
Enable Parallel Push	Yes	Select to allow parallel (multi-threaded) data push to Asset Manager. This improves performance. Note that you must configure SQL Server & DB2 to support parallel push. See "Prepare Asset Manager for Parallel Push".
Data Flow Probe	Yes	Select the name of the Data Flow Probe/Integration service used to execute the synchronization from.
Additional Probes	No	Select additional probes to use when pushing to AM in order to increase redundancy.
Default owner name	No	Not required for this integration point. **Note**: This field only appears in a Multi-Tenant Enabled UCMDB.

5. Click **Test Connection** to make sure there is a valid connection.

6. Click **OK**.

The integration point is created and its detailed are displayed.

UCMDB creates a default data push job when creating the integration point. If needed you may create or edit the existing job. For more information, see "Work with Data Push Jobs" in the *HP Universal CMDB Data Flow Management Guide*.

Push CI Data from UCMDB to Asset Manager

Data push jobs copy CI or CI relationship records from your UCMDB system to your Asset Manager system. To run a data push job, complete the following steps:

1. Log on to UCMDB as an administrator.

2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.

3. Select the integration point you created for Asset Manager.

4. Add a new data push job as follows:

   a. Click the ![*] button on the right panel.

   b. In the Name field, type a unique name for the job.

   c. Click the ![+] button to add existing TQL queries to the job.

   UCMDB Creates a default data push job when creating an integration point.The following table lists the Topology Query Language (TQL) queries in the default data push job. If required, you may create, update, or remove TQL queries for the push job. You may also need to update the mapping. See "How to Customize an Existing Mapping" on page 73. To access these OOTB TQL queries for push, go to **Modeling > Modeling Studio > Resources**, select **Queries for Resource Type** and then go to **Root > Integration > AM Push**.

   > **Note:** If you want to use cluster with Asset Manager, you must install the latest version of the SAM Best Practice package in Asset Manager.

TQL Query	Description
AM Business Element Push	Pushes Business Applications, Business Services and Business Infrastructure CIs. Mapping XML: pushMappingAMBusinessElement.xml

TQL Query	Description
AM Business Element Relations Push	Pushes relationships between Business Elements (pushed by AM Business Element Push Query) to other business elements or to nodes. Business Elements must have been pushed before this TQL query synchronization in the 'AM Business Element Push' TQL query synchronization. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingBeRelations.xml
AM Computer Push	Pushes nodes (Computers, Network Devices, etc.). Also pushes IPs, Interfaces, Disk Devices, Physical Ports, Hardware Boards, Display Monitors, CPUs, Printers, Inventory Scanners, File Systems, and Assets. Minimal attributes for pushing a Node: ○ Serial Number ○ Vendor or Discovered Vendor ○ Model or Discovered Model ○ Node Role **Note:** These are required values, and depend on the capability of the data source to report them. Mapping XML: pushMappingAMCoumputer.xml
AM Computer Relations Push	Pushes relationships between Computers to any node (Computers, Network Devices, etc.). Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingAMComputerRelations.xml

TQL Query	Description
AM Host Server And Running LPAR VM Relations Push	Pushes relationships between Host and Guest (virtualized) systems of LPAR type. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingHostToVMLpar.xml
AM Host Server And Running Solaris VM Relations Push	Pushes relationships between Solaris Host and Guest (Virtualized) operating systems. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingHostToVMSolaris.xml
AM Host Server And Running VM Relations Push	Pushes relationships between Host and Guest (Virtualized) operating systems. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingHostToVM.xml
AM Installed Software Sync	Pushes Installed Software and User_Software_Utilization CIs. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingNormalizedSW.xml (Possible alternate mapping: pushMappingSWNonNorm.xml See "Installed Software" on page 60.)
AM Net Device Relations Push	Pushes relationships between Network Devices to Network Devices. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingAMNetDeviceConnections.xml

TQL Query	Description
AM Oracle LMS Push	Pushes the Oracle Running Software and its Oracle LMS data. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingOracleLMS.xml
AM Software Sync Hypervisor	Pushes Hypervisor Installed Software. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingHypervisor.xml
AM Cluster Push	Pushes Cluster CIs Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingCluster.xml
AM Cluster Node Relations Push	Pushes cluster relations between Computer Elements (pushed by AM Computer Push) and Cluster. Cluster Elements must have been pushed before this TQL query synchronization in the 'AM Cluster Push' TQL query synchronization. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingClusterNodeRelations.xml
AM Cluster Runningsoftware Push	Pushes Cluster Runningsoftware CIs Nodes must have been pushed before this TQL query synchronization in the 'AM Cluster Push' TQL query synchronization. Mapping XML: pushMappingClusterRunningsoftware.xml

TQL Query	Description
AM Cluster Runningsoftware Relations Push	Pushes cluster runningsoftware relations between Computer Elements (pushed by AM Computer Push) and Cluster. Cluster Runningsoftware Elements must have been pushed before this TQL query synchronization in the 'AM Cluster Runningsoftware Push' TQL query synchronization. Nodes must have been pushed before this TQL query synchronization in the 'AM Computer Push' TQL query synchronization. Mapping XML: pushMappingClusterRunningsoftwareRelations.xml

d. Select or unselect the **Allow Deletion** option for each query. (The setting determines if this TQL query is allowed to delete data from Asset Manager, though the actual action on delete is defined by CI type in the mapping xml.)

> **Note:** For scheduling configuration, see "How to Schedule Data Push Jobs" on the next page.

e. Click **OK**.

f. Save the integration point.

5. Run the job manually to see if the integration job works properly:

a. To push all the relevant data for the job, click the 🔁 button.

b. To push only the changes in the data since the job last executed, click the 🔁 button.

6. Wait for the job to complete; click the 🔄 button multiple times as needed until the job is completed.

7. When the job is completed, the job status becomes one of the following depending on the results:

- Succeeded

- Completed

- Failed

8. Click the **Statistics** tab to view the results; if any errors occur, click the **Query Status** tab and **Job Errors** tab for more information. For more information about errors, see "Troubleshooting and Limitations".

> **Note:** For details about these tabs and managing the integration, see **Integration Jobs Pane** in the *HP Universal CMDB Data Flow Management Guide*.

If the job completes successfully, you can view the UCMDB CI data in Asset Manager.

How to View UCMDB Data in Asset Manager

After a push job is successfully completed, you can search for and verify that the pushed CI/relationship data is in Asset Manager.

Nodes

This includes computers, network devices, etc.

To view:

1. Log on to Asset Manager as an administrator.

2. Go to **Portfolio Management > Asset Configurations > IT equipment > Computers and virtual machines**.

3. In the opened dialog box, for **IP name**, type the name of the computer you are searching for.

4. You may use '<name prefix>%' for easier searching. For example, searching IP name for **mycomp%** returns computer mycomp1, mycomp2, etc.

5. Browse the computer for the different data.

Business Elements

This includes Business Applications, Business Services and Business Infrastructures.

1. Log on to Asset Manager as an administrator.

2. Go to **Asset Lifecycle > IT Services and virtualization > Business services > Business services**.

3. Browse the different Services.

For more information about viewing data in Asset Manager, see the Asset Manager Documentation.

How to Schedule Data Push Jobs

UCMDB allows you to schedule job executions directly from a data push job.

1. Log on to UCMDB as an administrator.

2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.

3. Select the integration point you created for the UCMDB - AM integration.

4. Select the push job.

5. Click the ✎ button.

> **Note:** UCMDB allows you to define two different schedules for two types of data push: **Changes Synchronization** and **All Data Synchronization**. It is recommended to use the Changes Sync schedule to only synchronize changes and avoid synchronizing the entire set of data each time.

6. Define a schedule for Changes Sync.

   a. Click on the **Changes Synchronization** tab.

   b. Select the **Scheduler enabled** option.

   c. Select the scheduling options you want to use.

7. Click the **All Data Synchronization** tab and select the scheduling options you want to use.

8. Click **OK**.

9. Save the integration point.

Installed Software

The Integration supports the following different flows for pushing Installed Software to Asset Manager. You may switch between these flows.

Note: The flows below show a simplified high-level flow of the different Installed Software synchronization behavior. The actual behavior may be more complex in some cases, mainly for performance improvement. See also .

Normalized Installed Software

This flow uses an InventoryModel to catalog each exact Software Version. Therefore, if the AM Operator decides to map a certain Software version to a different model, he only has to do it once to the Inventory Model, and does not have to process all the Installed Software in AM. This flow allows using either the SAI Version ID, or the attributes name, version, and vendor, to correctly reconcile the Installed Software, and uses the information to automatically create Models according to major versions as needed.

1. Each Installed Software is first mapped to an Inventory Model. If one does not exist, it creates one. The mapping is done according to the SAI Version ID which is an inventory ID from the Universal Discovery Scanner, or by using the Installed Software's name, version, and vendor.

2. It then sees if the InventoryModel has a final mapping to a Model. If it is a new InventoryModel, or the InventoryModel has no final mapping to a Model, it attempts to search for one with the same name and version. If one is found, it connects the InventoryModel to it; otherwise it creates a new Model.

3. It then connects the Installed Software to the Model as well.

Note: Normalized Installed Software is the default flow.

Normalized Installed Software – No Model Creation

This flow uses an InventoryModel to catalog each exact Software Version. Therefore, if the AM Operator decides to map certain Software version to a different model, he only has to do it once to the InventoryModel, and does not have to process all the Installed Software in AM. This flow does not automatically create a Model. Instead, the Model must be connected to the InventoryModel by a different flow, or manually by an Asset Manager Operator.
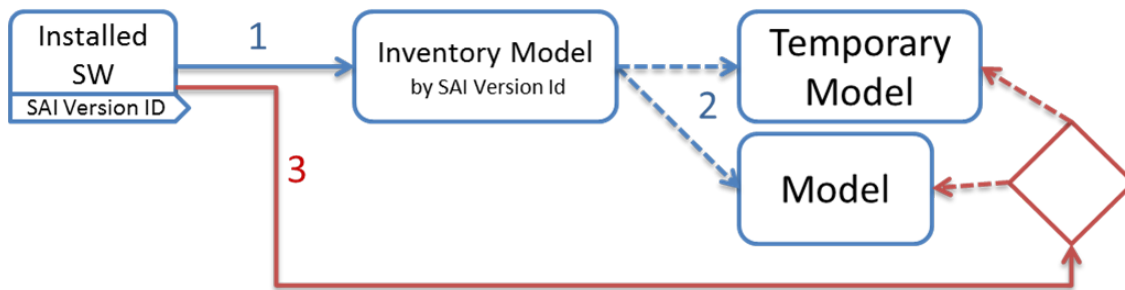
1. Each Installed Software is first mapped to an Inventory Model. If one does not exist, it creates one. The mapping is done according to the SAI Version ID, which is an inventory ID from the Universal Discovery Scanner, or by using the Installed Software attributes: name, version, and vendor.

2. It then sees if the InventoryModel has a final mapping to a Model. If not, it chooses the temporary model (an Unknown Software Model).

3. It then connects the Installed Software to the Model found in the step 2.

4. Later, an Asset Manager Operator manually connects each Inventory Model to a final Model, as he wishes.

Non-Normalized Installed Software

This flow pushes Installed Software and Models only. (It does not map or use the Inventory Models in any way).

Each Installed Software is mapped to a matching Model which is created if not found.

Switching between Installed Software Flows

Switching to Normalized Installed Software Flow

This is the default flow, enabled OOTB for this integration. If the flow was changed and you would like to return to this flow:

- Change the Installed Software push TQL query name (original name: AM Software Push) to **AM Installed Software Push**.

Switching to Normalized Installed Software – No Model Creation

In Adapter Management, edit **am-push-mapping.xml**:

1. Go to **ci-type="Complete_amModel"**

2. Change **operation-type** to **optional_reference**

Switching to Non-Normalized Installed Software

Change the Installed Software push TQL query name (original name: AM Installed Software Push) name to: **AM Software Non Norm**.

How to Tailor the Integration

This section includes:

- "Integration Architecture" on the next page

- "How to Change Adapter Settings" on page 72

- "How to Customize an Existing Mapping" on page 73

- "How to Add a New Mapping to the Integration" on page 74

Note: For more detailed information about customizing the mapping, see "Developing Enhanced Generic Push Adapters" in the *HP Universal CMDB Developer Reference Guide*.

Integration Architecture

This section contains details about the architecture of the integration.

Data Flow Architecture

1. The Push Engine executes the TQL query.

2. For a differential flow, the data is compared to the last synchronized data, and only the changes are forwarded.

3. Data is converted into Composite CIs (instances of data according to the TQL Root elements).

4. Data is then pushed to the Push Adapter.

5. The Push Adapter loads the correct mapping for the specific TQL query.

6. All **dynamic_mappings** are executed and saved to maps, to allow usage in the next mapping stage.

   For more information, see "Developing Enhanced Generic Push Adapters" in the *HP Universal CMDB Developer Reference Guide*.

7. Data is mapped from the UCMDB data Model into the AM Data model according to the mapping XML.

8. Data is sent to the AM Connector.

9. AM Connector orders all the data in a set of dependency trees, starting with the records that do not depend on any other record.

10. AM Connector attempts to merge any duplicate records

11. AM Connector starts reconciling and pushing any record without any dependencies, or a record whose dependencies have already been reconciled/pushed to Asset Manager.

    a. AM Connector first tries to reconcile with existing records

    b. If it finds a match, it attempts to update that record,

    c. If it does not find a match, it attempts to create a new record.

12. AM Connector deletes any records it is required to delete in AM, as permitted by action-on-delete.

Integration TQL Queries

A TQL query used for the integration must contain a root query node.

Any attribute using in the mapping flow of the Push Adapter must be marked in the selected layout of the query node.

Each TQL query may only have one mapping.

For more information, see **Data Flow Management > Integration > Integration Studio > Integration Jobs Pane**.

Reconciliation Proposals

When pushing data to Asset Manager, there is an option to create a reconciliation proposal (RP). A reconciliation proposal should be created if there is a change in a specific attribute that may need AM Operator validation or action to support AM business processes.

The OOTB configuration creates a reconciliation proposal record when the memory size of the pushed computer has decreased compared to the AM computer.

How to use Reconciliation Proposals

In the OOTB configuration **lMemorySizeMb** is marked for attribute-reconciliation. The update script calls the **updateMemorySize** function. This function verifies if the memory size of the computer was decreased. It initializes all the parameters that are passed to the function **validateReconcUpdateAdvance**. Calls **validateReconcUpdateAdvance** and return its returned value.

validateReconcUpdateAdvance is a function that returns the value that should be set to the attribute according to the Reconciliation Proposal status.The following table describes its parameters:

Parameter	Description
AMApiWrapper	The wrapper that is used to communicate with the AM.
newVal	The value of the attribute in the pushed data.
oldVal	The value of the attribute that is retrieved from AM.

Parameter	Description
recordId	The primary key of the table that the attribute belongs to.
strCode	The prefix of the **code** field in the reconciliation proposal.
strName	The name of the reconciliation proposal.
path	The name of the attribute.
recordTable	The table that the attribute belongs to.

validateReconcUpdateAdvancereturns the value that should be set for the attribute, according to the Reconciliation Proposal status.

In order to create a reconciliation proposal flow on a different attribute, the following steps must be completed:

1. Add the **<attribute-reconciliation>** tag for this attribute.

2. The update-script should call a new function that initializes the parameters passed to **validateReconcUpdateAdvance**, and returns the value returned from **validateReconcUpdateAdvance**.

> **Note:** It is suggested to use the **updateMemorySize** function as a reference.

Asset Manager Rules and Flows

Asset Manager has its own set of rules and flows that are enforced by the Asset Manager API. Some customizations may need to later these rules and flows as well. For more information, see the Asset Manager documentation.

Data Mapping

For details, see **Developing Push Adapters** in the *HP Universal CMDB Developer Reference Guide*.

Push Mapping

This section includes tables explaining each XML tag and the attributes available for configuration.

Basic Information

Attributes in the **<am-mapping>** tag.

Attribute	Description
ci-type	Name used in push adapter mapping to recognize this record type.
primary-key	The primary key column in AM database schema.
operation-type	Defines what operations may be done with the record: - **insert** Only allows creation of new records; if it already exists, an exception is thrown. - **update** - Only allows updates of an existing record; if it does not exist, an exception is thrown. - **update_else_insert** - If the record exists it is updated, otherwise the record is created. - **reference-only** - The record is only used for referencing by other record (and is not updated). An exception is thrown if the record does not exist. - **ignore** - The record is unaffected by operations. - **insert_else_reference** - If the record does not exist, it is created. Otherwise it is only used as a reference and is not be updated; see **reference-only**.

Attribute	Description
parallel-push-allowed	If enabled with the *enabled-parallel-push* configuration of the integration point, will attempt to push to the entity with multiple threads in order to increase performance.
merge-allowed	If enabled and this entity is an exact duplicate of another entity in the chunk, it merges both entities into one and fixes any relevant references.
errorcode-override	If used together with the adapter specific errors, allows the printing of a customized error message to the UI if the push or reconciliation of this entity fails.
target-ci-type	Real name of the AM database table to push this record to. If missing , it uses ci-type instead.
from-version	Use this mapping only from (and including) this version. The version is taken from the integration point configuration.
to-version	Use this mapping only up to (and including) this version. The version is taken from the integration point configuration.

Reconciliation

Reconciliation defined for each mapping may include more than one set of reconciliation rules. When attempting to reconcile the record with existing ones in the AM database, the AM Connector tries each of the reconciliation sets until it finds a matching record. Priority is defined by order of reconciliation rules. If no record in the AM database matches this record, an insert operation occurs if the operation type permits it.

Name	Type	Description
reconciliation	Tag	Parent XML tag for all reconciliation configuration.
reconciliation-keys	Tag	Represents a single reconciliation rule that may be made of one or more attributes. All attributes inside the rule must match in order for the reconciliation of this rule to be successful.
reconciliation-key	Tag	Represents a single attribute used for reconciliation as part of the reconciliation-keys rule.
Ignore-case	Attribute	Part of the reconciliation-key tag. Specifies that this attribute comparison ignores case.

Name	Type	Description
reconciliation-advanced	Tag	Allows definition of the reconciliation rule by manually defining the WHERE clause of the AQL (Asset Query Language). Uses GString (Groovy String) to generate the replacement String. Any variable or property defined in this record or its parent during the mapping stage (in the Push Adapter) may be used as a variable in the GString. (See http://groovy.codehaus.org/Strings+and+GString for more information). **Note:** AMPushAdvancesReconciliationException may be thrown inside this tag to skip to the next rule.
follow-parent	Tag	Used for defining overflow tables. See the AM documentation for more information on overflow tables. When using follow-parent, no other reconciliation may be used as this target CI has a 1:1 connection with its parent, and it uses the parent reconciliation to push data to AM.
am-prefix	Attribute	Part of the follow-parent tag. Defines the name that the parent target CI uses to reference to this table. (To find out the correct value, navigate to **AM Application Designer > Edit Links**.)

Example:

```
<reconciliation>
  <reconciliation-advanced>Portfolio.CMDBId = '${if(globalId==null) { throw
new com.hp.ucmdb.adapters.ampush.exception.
AMPushAdvancesReconciliationException
('Not enough reconciliation data') }else{ return globalId}}'
  </reconciliation-advanced>
  <reconciliation-keys>
    <reconciliation-key ignore-case="true">AssetTag</reconciliation-key>
  </reconciliation-keys><reconciliation-keys>
    <reconciliation-key>TcpIpHostName</reconciliation-key>
    <reconciliation-key>Workgroup</reconciliation-key>
  </reconciliation-keys>
</reconciliation>
```

Target CI Validation

This tag allows the definition of a validation rule that is executed on specific attribute values: the new one held in memory, and the old one stored in the AM database.

The following table shows the attributes of the **<target-ci-validation>** tag:

Name	Description
attribute-name	The attribute that you want to use for validation.
validation-script	A Groovy based script that returns **true** if this record is to be pushed, and **false** if it is not to be pushed. The script may access any external Groovy code in the path in order to run the evaluation. • **vNewVal** - Attribute value of the record in memory. • **vOldVal** - Attribute value of the record in the AM database.
failed-validation-error-code	This optional attribute holds the error code that appears if there is a validation failure. The arguments that can be referenced in the error message are: • {0} - The validated attribute name. • {1} - The property value in UCMDB. • {2} - The property value in Asset |Manager. • {3} - The validation script. • {4} - The additional message from the **additional-failure-message** attribute, or 'null' if there is no additional message.
additional-failure-message	This optional attribute holds an additional error message that can be referenced by the error message in the **properties.error** file. See **failed-validation-error-code**, above.

Example:

```
<target-ci-validation attribute-name="dtLastScan" validation-script="mappings.scripts.AMR
econciliationAdvanced.isDateAfter(vNewVal,vOldVal)"/>
```

Reference Attribute

A reference attribute defines a column that references another record from a different or same table. This record is not pushed, or reconciled against existing AM database records, until this reference is resolved. Resolved references are replaced by a reference ID that represents the primary ID of the referenced record.

The following table shows the attributes of the **<reference-attribute>** tag:

Name	Description
ci-name	The CI-type of the referenced record.

Name	Description
datatype	The value type of the record.
name	The column in the current record that is to be populated by the reference ID.
reference-direction	According to the tree created by the Push Adapter, the value specifies if the referenced record is a parent or child of the current record.

Example:

```
<reference-attribute ci-name="SW_amModel" datatype="STRING" name="lModelId"r
eference-direction="child"/>
```

Attribute Reconciliation

This tag allows the AM connector to decide what to do with an attribute value according to the existing value in the AM database.

The following table shows the attributes of the **<attribute-reconciliation>** tag:

Name	Description
attribute-name	The attribute to be reconciled.
update-script	The script to execute in case of an update operation on the record. The returned value by the groovy script will be push to AM as the value of this attribute. • **vNewVal** - Attribute value of the record in memory. • **vOldVal** - Attribute value of the record in the AM database.
Insert-script	The script to execute in case of an insert operation on the record. The value returned by the Groovy script is pushed to AM as the value of this attribute. **vNewVal** - Attribute value of the record in memory.

Example:

```
<attribute-reconciliation attribute-name="AssetTag" update-script=
"mappings.scripts.AMPushFunctions.fIsEmpty(vOldVal) ? vNewVal : vOldVal"/>
```

Action on Delete

This tag allows customization of the behavior on receipt of a delete notification for a record.

Note: No deletion occurs if the **Allow Delete** option in the job definition is disabled.

The following actions are possible

- **<ignore>** - Do nothing.

- **<delete-ci>** - Delete this record from the AM database.

- **<set-attribute-value>** - Change the value of one or more attributes in the AM database.

Example:

```
<action-on-delete>
     <set-attribute-value name="bMissing" datatype="BOOLEAN" value="1"/>
</action-on-delete>
```

Enum Attribute

This tag allows a specific enum attribute to be pushed in a serial mode, when the adapter is configured to push data in parallel mode.

> **Note:** This option exists to prevent duplicate key exceptions occurring when several threads push the same enum value.

The following table shows the attributes of the **<enum-attribute>** tag:

Name	Description
attribute-name	The enum attribute name.
itemized-name	The itemized list format (amOS) of the enum.

Ignored Attributes

This tag allows specific attributes to be ignored and not pushed to the AM database. This capability is commonly used with the **from-version** and **to-version** attributes or tags, to ignore certain attributes for specific versions of Asset Manager.

The following table shows the attributes of the **<ignored-attributes>** tag:

Name	Description
from-version	Ignore this attribute only from (and including) this version. The version is taken from the integration point configuration.
to-version	Ignore this attribute only up to (and including) this version. The version is taken from the integration point configuration.

Example:

```
<ignored-attributes>
    <ignored-attribute>lSeq</ignored-attribute>
</ignored-attributes>
```

How to Change Adapter Settings

1. Go to **Data Flow Management > Adapter Management > AMPushAdapter > Adapters**.

2. Right-click **AMPushAdapter** and click **Edit Adapter Source**.

3. Scroll down to the **<adapter-settings>** tag.

4. Edit the settings as required and click the Save button.

   The following table shows the Settings relevant to the AM Push Adapter:

Setting	Default	Description
replication.chunk.size	4,000	Defines the requested number of CIs and Relationships sent in each chunk. It is possible to adjust this setting to try and improve performance of the server and the Probe.
replication.chunk.root.limit	850	Defines the maximum requested number of Roots sent in each chunk. This works with **replication.chunk.size** as a limiter on the amount of data sent in each chunk.
PushConnector.class.name		The name of the Java class used to load the AM Connector.
parallel.thread.pool.size	8	The amount of threads used in Parallel Push Mode. The more threads, the more CPU used. Increasing the pool size may lower performance.
mapping.size.fuse	100,000	The maximum number of records the AM Connector may to retrieve in <dynamic_mapping> .
transaction.deadlock.max.retry.count	3	The maximum number of retries the AM Connector attempts to push a deadlocked database transaction.

How to Customize an Existing Mapping

This example shows you how to add the **BarCode/RFID** attribute to the integration, including the TQL query, Push Adapter Mapping and AM Mapping configuration. It allows the integration to both push the BarCode/RFID attribute to Asset Manager, as well as use it for reconciliation of a Hardware Asset.

After completing the following steps, you may run the job with the customized mapping:

1. **Add the BarcodeOrRfidTag attribute to the AM Computer Push TQL query layout.**

   In this step we add the attribute of the Asset CI to the integration TQL query so that we can use the attribute and value in the mapping.

   a. Go to **Modeling > Modeling Studio > Resources** and select the **Queries** Resource Type.

   b. Go to **Queries: Root > Integration > AM Push > AM Computer Push**.

   c. Select **Asset**, right-click and select **Query Node Properties**.

   d. Go to the **Element Layout** tab.

   e. Move the **BarcodeOrRfidTag** to the **Specific Attributes** box.

   f. Click **OK**.

   g. Save the Query.

2. **Add the BarCode Mapping to the pushMappingAMComputer.xml push adapter mapping.**

   In this step we take the value from the TQL result and remodel it to the Asset Manager Data Model.

   a. Go to **Data Flow Management > Adapter Management > Packages > AMPushAdapter > Configuration Files > pushMapingAMComputer.xml**.

   b. Go to the **<target_ci_type name="amComputer">** XML tag.

   c. Add the variable to hold the value of the barcode:

   ```
   <variable name="vBarCode" datatype="STRING"   value="Root.Asset[0]['barco
   de_or_rfid_tag"/>
   ```

   d. Go to the **<target_ci_type name="amAsset">** XML tag.

   e. Below the tag, add the following XML tag:

```
<target_mapping name="BarCode" datatype="STRING" value="vBarCode"/>
```

    f.   Click **OK**.

3. **Add the BarCode to the am-push-mapping.xml configuration.**

> **Note:** This example is specific to Asset Manager version 9.3 and above.

    a.   Go to **Data Flow Management > Adapter Management > Packages > AMPushAdapter > Configuration Files > am-push-mapping.xml**.

    b.   Go to the **<am-mapping ci-type="amComputer" from-version="9.3">** XML tag.

    c.   Add the following rule to the advanced reconciliation rule:

```
<reconciliation-advanced> Portfolio.Asset.BarCode ='${if(vBarCode==nul
l)
{ throw new com.hp.ucmdb.adapters.ampush.exception.AMPushAdvancesRecon
ciliationException
('Not enough reconciliation data') }else{ return vBarCode}}'</reconcil
iation-advanced>
```

This rule retrieves the computer whose asset has the same barCode value as vBarCode. If the VBarCode is null, an exception is thrown stating that there is insufficient reconciliation data. The exception thrown orders the AM Connector to try the next available reconciliation rule.

    d.   To avoid overwriting an existing value, add the following to **<am-mapping ci-type="amAsset">**:

```
<attribute-reconciliation attribute-name="BarCode" update-
script="mappings.scripts.AMPushFunctions.fIsEmpty(vOldVal) ? vNewVal :
vOldVal"/>
```

    e.   Click **OK**.

How to Add a New Mapping to the Integration

This example shows how to add a new TQL query, push-mapping, and am-mapping to the integration. It also shows how to push Locations from UCMDB to Asset Manager. It consists of the following steps:

Step 1: Create a TQL Query

1. Go to **Modeling > Modeling Studio > New > Query**.

2. From the **CI Types** tab, add a **Location CIT** to the query.

3. Right-click the **Location Query Node** and select **Query Node Properties**.

4. Rename the **Element Name** to **Root**.

5. Go to the **Element Layout** tab.

6. Select **Select attributes for layout**.

7. In the **Attributes condition** drop down, select **Specific Attributes**, and add the following attributes:

   a. **Name**

   b. **LocationType**

   c. **LocationBarCode**

8. Click **OK**.

9. Save the query to **Root > Integration > AM Push > AM Location Push**.

Step 2: Create a Push-Mapping

1. Go to **Data Flow Management > Adapter Management > AMPushAdapter**.

2. Click the ✳ button and select **New Configuration File**.

3. Type the following Name: **AMPushAdapter/mappings/hw/pushMappingAMLocation.xml**.

4. Select the **AMPushAdapter** package.

5. Click **OK**.

6. Copy the following into the newly created XML file:

```
<integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:no
NamespaceSchemaLocation="/am-push-adapter/src/
main/resources/schema/am-push-adapter.xsd">
  <info>
    <source name="UCMDB" versions="10.0" vendor="HP"/>
    <target name="AssetManager" versions="9.3" vendor="HP"/>
  </info>
  <import>
    <!--Allows the XMLs to use the AMPushFunctions groovy      methods--
>
    <scriptFile path="mappings.scripts.AMPushFunctions"/>
```

```
    </import>
    <targetcis>
      <!--Query:  Location(Root) -->
      <source_instance_type query-name="AM Location Push"      root-elemen
t-name="Root" >
      <!-- Single_amLocation is the name we will refer to this CI      in t
he am-mapping.xml-->
        <target_ci_type name="Single_amLocation">
        <!--Retrieve the LocationBarCode from the Location CI          an
d place it in the amLocation record-->
        <!—-'name' is the column name in Asset Manager's       table --
>
        <!—-'datatype' is the type of the data of this column -->
        <!—-'value' the script to execute in order to retrieve        the
value of this target_mapping-->
        <target_mapping name="BarCode" datatype="STRING"         value="Ro
ot['location_bar_code']"/>
        <target_mapping name="LocationType" datatype="STRING"         valu
e="Root['location_type']"/>
        <target_mapping name="Name" datatype="STRING"         value="Root[
'name']"/>
<!--Marks the location as created on the fly (So we can distinguish from
AM UI created Locations)-->
        <target_mapping name="bCreatedOnTheFly"         datatype="BOOLEAN"
value="1"/>
      </target_ci_type>
</source_instance_type>
    </targetcis>
</integration>
```

7. Click **OK**.

Step 3: Create an AM-Mapping

1. Go to **Data Flow Management >Adapter Management > AMPushAdapter**.

2. Click **am-push-mapping.xml**.

3. Insert the new mapping:

```
<!-- 'ci-type' is the name of the ci as appear in the      mapping file.
   'target-ci-type' the actual table name in Asset Manager
    this target ci is mapped to -->
<am-mapping ci-type="Single_amLocation" primary-key="lLocaId"
 target-ci-type="amLocation" operation-type="update_else_insert"
 parallel-push-allowed="true" from-version="9.3">
  <reconciliation>
```

```
   <reconciliation-keys>
     <!-- The reconciliation of 'location' will be done by the
      two attributes: Name, LocationType.
      Those are the two identifiers of Location CIT in UCMDB-->
      <reconciliation-key>Name</reconciliation-key>
      <reconciliation-key>LocationType</reconciliation-key>
   </reconciliation-keys>
 </reconciliation>
 <!-- In case the CI has been deleted from the UCMDB,
   it won't effect the amLocation in Asset Manager -->
 <action-on-delete>
    <ignore/>
 </action-on-delete>
</am-mapping>
```

4. Click **OK**.

Step 4: Create a Job with the New TQL Query

1. Go to **Data Flow Management > Integration Studio**.

2. Create an integration point with Asset Manager:

   a. In the **Integration Jobs** tab, click the ⁂ button.

   b. Insert a job name in the **Name** field.

   c. Click the ✚ button, and choose the **AM Location Push** query.

   d. Click **OK**.

Step 5: Run the Job

1. Click on the job created in "Step 4: Create a Job with the New TQL Query".

2. Click the 🔁 button.

3. Wait for the job to finish. You should click the 🔄 button to see progress.

4. Make sure that the status is **Succeeded**.

Step 6: View the Results

1. Go to the **Asset Manager Client**.

2. Go to **Organization Management > Organization > Location**.

3. Validate that the Location pushed in the previous steps is displayed.

Frequently Asked Questions

What is a Root CI node?

A Root node is a TQL node that represents the CI type that is created via the push to Asset Manager from the TQL Structures. Usually the rest of the TQL structure contains information that can be incorporated within the Root CI type and is used to enrich the record in Asset Manager. The Root is the heart of the Composite CI (or Instance), and if it is deleted from UCMDB we send a delete notification to Asset Manager for the entire record.

When is a new Asset created in Asset Manager?

In the out of the box integration we create Assets for 4 types of UCMDB CIs:

- **Nodes**

- **Business Elements**

- **Printers**

- **Display Monitors**

Whenever UCMDB sends a CI of one of these types, the integration first tries to detect if this Asset already exists in Asset Manager, using the defined reconciliation rules. If a matching Asset is found, it is updated, otherwise a new Asset is created.

How do I control the action taken when a CI is deleted in UCMDB?

See .

I deleted a Node CI in UCMDB - Why isn't it deleted in Asset Manager?

The default Action on Delete for nodes in the integration is to do nothing.

You may either change the action to delete the Asset in Asset Manager by changing the <action-on-delete> xml mapping in the am-push-mapping.xml:

```
<am-mapping ci-type="amComputer" …>
        …
        …
        <action-on-delete>
              <delete-ci/>
        </action-on-delete>
</am-mapping>
```

Or you may change the action to set the Asset as Missing in Asset Manager by changing the <action-on-delete> xml mapping in the am-push-mapping.xml:

```
<am-mapping ci-type="amComputer" …>
```

```
      …
      …
    <action-on-delete>
      <set-attribute-value name="Portfolio.seAssignment" datatype="INTEGER"
value="6"/>
    </action-on-delete>
</am-mapping>
```

Validate that the **Allow Delete** check box in the job configuration is selected.

I deleted an Installed Software CI in UCMDB - Why isn't it deleted in Asset Manager?

The default Action on delete for Installed Software in the integration is to mark it as missing.

You may change the action to delete the Soft Installed in Asset Manager, by changing the <action-on-delete> xml mapping in the am-push-mapping.xml:

```
<am-mapping ci-type="Complete_amSoftInstall" …>
        …
        …
        <action-on-delete>
              <delete-ci/>
        </action-on-delete>
</am-mapping>
```

Due to the complexity and amount of flows available for Installed Software you must also change these mappings to match:

- SW_amSoftInstall

- Complete_amSoftInstall_User

- soft_Hyper_amSoftInstall

- SW_amSoftInstall_User

Is it possible to avoid overwriting an attribute in Asset Manager?

Yes. By using the Attribute Reconciliation feature, you choose to never overwrite an existing value.

Example:

```
<attribute-reconciliation attribute-name="AssetTag" update-script=
"mappings.scripts.AMPushFunctions.fIsEmpty(vOldVal) ? vNewVal : vOldVal"/>
```

See "Integration Architecture" on page 63.

What is the different between the mapping XMLs and the am-push-mapping.xml?

The Mapping XMLs (for example: pushMappingAMBusinessElement.xml) define the way we convert the data from the UCMDB data model into the Asset Manager Data Model and are executed by the Push Adapter.

For more information, see **Developing Push Adapters** in the *HP Universal CMDB Developer Reference Guide*.

The **am-push-mapping.xml** is the Asset Manager Connector configuration file. It configures the way we reconcile and handle the data, before we update Asset Manager with the record.

Should I select the 'Enable Parallel' Feature?

Asset Manager configured over an Oracle database supports parallel push out of the box.

Asset Manager configured over an SQL Server database, needs some tuning before enabling this capability. See "Prepare Asset Manager for Parallel Push" on page 48.

Why does an integration point that synchronizes only the AM Installed Software Push TQL query, keep failing?

The AM Installed Software Push TQL query contains both a query node of Installed Software and a query node of Node. The Node in this mapping is only referenced, and is mapped to Asset Manager by saving its Asset Manager ID from an earlier run. Before pushing this TQL query, the same integration point must push the Node to Asset Manager (using the AM Computer Push TQL).

How can I push nodes without Model Name or Serial Number information, to Asset Manager?

To avoid pushing nodes not yet fully discovered, we avoid sending ones without a Model Name or Serial Number that provide us with a physical identification of the Asset. If you would like to push these nodes as well, simply remove the appropriate condition of the node from the **AM Computer Push** TQL query.

However removing the **Node Role** condition (filtering nodes without a Node Role) from the TQL query is not recommended, as the integration will not know what type of an Asset/Portfolio to create in Asset Manager.

Troubleshooting and Limitations

- **Limitation:** A single probe may only connect to one version of Asset Manager. (Use of multiple instances of the same version is supported). This is due to the JVM limitation of loading only one Asset Manager API per process.

- **Limitation:** The out of the box implementation does not support synchronization of mobile devices.

- **Limitation:** The Data Flow Probe or Integration Service must be installed on a Windows OS.

- **Limitation:** DB2 parallel push mode is not supported in UCMDB 10.01.

- **Problem: Missing DDLs or jars.**

  When testing the connection of the integration point, an error with the following phrase appears:

  Asset Manager DLLs and/or Jars are missing

  Solution: See "Create AMPushAdapterAPI Package with Required AM API Files" on page 49

- **Problem: First time synchronization has many failed CIs.**

  The first synchronization in the integration creates a large number of enum values in the Asset Manager database. In some cases, when enabling the parallel push for the first synchronization, it may cause a very large number of deadlocks during the push that is more than the Adapter's auto deadlock handling mechanism can handle.

  Solution 1: You may try to re-synchronize the failed CIs until they all pass.

  Solution 2: Add the enum attribute that caused the duplicate key exception to **am-mapping** in the **am-push-mapping.xml**.

- **Problem: A push integration fails with the error message 'Only one connected Asset CI is allowed'**

  When running a push integration of Computers to Asset Manager, one of the reconciliation attributes used is Asset Tag. If there is more then one Asset CI connected to the Node CI, it means there is more then one Asset Tag for a single Node; this is not a valid state.

  Solution: Remove the CIT Computer from the incorrect Asset CIs. This should ensure the Node is connected to either one or no Asset CIs.

- **Problem: Some CIs in a Relations Push fail.**

  The Relations flow (TQL query) assumes that you schedule (either in the same job, or in a different job) the different flows that this Relations push depends on, to run before this flow. The following table shows the dependencies:

TQL Query	Depends On
AM Business Element Push	
AM Business Element Relations Push	**AM Business Element Push** and **AM Computer Push**
AM Computer Push	
AM Computer Relations Push	**AM Computer Push**

TQL Query	Depends On
AM Host Server And Running LPAR VM Relations Push	**AM Computer Push**
AM Host Server And Running Solaris VM Relations Push	**AM Computer Push**
AM Host Server And Running VM Relations Push	**AM Computer Push**
AM Installed Software Push	**AM Computer Push**
AM Net Device Relations Push	**AM Computer Push**
AM Oracle LMS Push	**AM Computer Push**
AM Software Hypervisor Push	**AM Computer Push**
AM Software Solaris Push	**AM Computer Push**

- **Problem: Some CIs in a Software Push fail.**

  The software flows assume that you schedule the computer push flow (either in the same job, or in a different job) to run first. See "Why does an integration point that synchronizes only the AM Installed Software Push TQL query, keep failing? " on page 80.

- **Problem: Missing Root in a TQL Query.**

  Solution: The integration TQL query must contain a Root Element (1 if it is a CI, 1 or more if it is a Relationship). Update your TQL query by renaming one of the query Elements to **Root**. Make sure your mapping xml is updated accordingly.

  See "What is a Root CI node?" on page 78.

- **Problem: Error in Test Connection. Unable to connect to this database engine.**

  The following solution assumes you are connecting to a DB2 or Oracle database.

  Solution: Validate the following:

  1. The database client is installed on the Data Flow Probe/Integration Service machine. See "Install a Database Client" on page 50.

  2. The installed client is a 64 bit version.

  3. The client is installed on the actual Probe selected in the integration point configuration.

- **Problem: Multiple Assets are created in Asset Manager for a single UCMDB Node.**

Solution 1: This can happen when the maximum length of an attribute is too short compared to the attribute's value in UCMDB. It causes the attribute value to truncate when pushed to Asset Manager. However, on a different execution, when attempting to reconcile the attribute, there will not be a match because of the truncated value in asset Manager. Therefore, increase the attribute length. See "Update Asset Manager Schema" on page 47.

Solution 2: Check and fix customized or changed reconciliation rules.

- **Problem: Error in test connection: Module Ssl : Unable to load dynamic library (libeay64.dll).**

  This error occurs when the Windows operating system of the probe is missing the Visual C++ 2008 SP1 or later.

  Solution 1: Download and install the Microsoft Visual C++ 2008 SP1 Redistributable Package (x64). You may download this from:
  http://www.microsoft.com/download/en/details.aspx?id=2092.

  Thereafter, reboot Windows and restart the Probe.

  Solution 2: Run Windows update and retrieve the Visual C++ 2008 SP1 or later update. Thereafter, reboot Windows and restart the Probe.

- **Problem: Reconciliation gaps when pushing and populating Business Elements using reconciliation by Global-ID.**

  Solution: To populate Business Elements using reconciliation by Global-ID, complete the following:

  a. Go to **Dataflow Management > Adapter Managent > Packages** and select **AMPushAdapter**.

  b. Select **Configuration Files > pushMappingAMBusinessElement.xml**.

  c. Replace the file content with the following:

  ```
  <integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noN
  amespaceSchemaLocation="/am-push-adapter/src/main/resources/schema/am-push-
  adapter.xsd">
  <info>
    <source name="UCMDB" versions="10.0" vendor="HP"/>
    <target name="AssetManager" versions="9.3" vendor="HP"/>
  </info>
  <import>
    <scriptFile path="mappings.scripts.AMPushFunctions"/>
  </import>
  <targetcis>
    <!--Query:  BusinessElement(root class = Business Service)(Root) -->
    <source_instance_type query-name="AM Business Element Push*" root-
  ```

```
element-name="Root" >
   <target_ci_type name="IS_amAsset">
   <variable name="globalId"  datatype="STRING" value="Root['global_id']"
/>
   <variable name="vAssetTag"  datatype="STRING" value="AMPushFunctions.u
Case(Root['name'])"/>
   <target_mapping name="AssetTag" datatype="STRING" value="vAssetTag"/>
   <target_mapping name="Label" datatype="STRING" value="Root['name']"/>
   <target_ci_type name="IS_amPortfolio">
    <target_mapping name="CMDBId" datatype="STRING" value="globalId"/>
   <target_ci_type name="IS_amModel">
    <target_mapping name="Name" datatype="STRING" value="Root['name']"/>
   <target_ci_type name="IS_amNature">
     <target_mapping name="Code" datatype="STRING" value="'BIZSVC'"/>
   </target_ci_type>
    <target_ci_type name="IS_amModel-Parent">
      <target_mapping name="BarCode" datatype="STRING" value="AMPushFunct
ions.getParentBarCodeFromType(Root['root_class'])"/>
     </target_ci_type>
    </target_ci_type>
   </target_ci_type>
  </target_ci_type>
  </source_instance_type>
 </targetcis>
</integration>
```

d. Click **OK**.

e. In the same directory select the **am-push-mapping.xml** file.

f. Delete the **am-mapping** of **ci-type="IS_amAsset"** and replace with the following:

```
<am-mapping ci-type="IS_amAsset" primary-key="lAstId" operation-type="upda
te_else_insert" target-ci-type="amAsset"  merge-allowed="true" parallel-pu
sh-allowed="true">
 <reconciliation>
  <reconciliation-advanced>PortfolioItem.CMDBId = '${globalId}'</reconcili
ation-advanced>
 <reconciliation-keys>
  <reconciliation-key ignore-case="true">AssetTag</reconciliation-key>
 </reconciliation-keys>
 </reconciliation>
 <reference-attribute ci-name="IS_amPortfolio" datatype="STRING" name="Por
tfolioItem" reference-direction="child"/>
 <attribute-reconciliation attribute-name="AssetTag" update-script="mappin
gs.scripts.AMPushFunctions.fIsEmpty(vOldVal) ? vNewVal : vOldVal"/>
 <action-on-delete>
  <delete-ci/>
```

```
    </action-on-delete>
  </am-mapping>

  <am-mapping ci-type="IS_amPortfolio" primary-key="lPortfolioItemId" target-
  ci-type="amPortfolio" operation-type="update_else_insert" parallel-push-al
  lowed="true">
   <reconciliation>
    <follow-parent am-prefix="PortfolioItem"/>
   </reconciliation>
   <reference-attribute ci-name="IS_amModel" datatype="STRING" name="lModelI
  d" reference-direction="child"/>
   <action-on-delete>
    <delete-ci/>
   </action-on-delete>
  </am-mapping>
```

g. Click **OK**.

Logs

The push adapter framework uses a different logs then the normal fcmdb.adapters.*.log files.

To change the level of the log files to debug, edit the following file:

- On the Data Flow Probe machine:
  ..\DataFlowProbe\conf\log\fcmdb.push.properties

- If using the integration service, on the UCMDB server:
  ..\UCMDB Server\Integrations\conf\log\fcmdb.push.properties

Change the log level to DEBUG:

loglevel=DEBUG

The integration generates **fcmdb.push.*** logs in the following folder:

- On the Data Flow Probe machine:
  ..\DataFlowProbe\runtime\log

- If using the integration service, on the UCMDB server:
  ..\UCMDB Server\Integrations\runtime\log

Chapter 4: HP Configuration Manager – Federating KPI Data

This chapter includes:

Overview

The federation mechanism that is built into HP Universal CMDB enables UCMDB to be used as a contact repository for sharing data among external applications, without duplicating it. By federating data from Configuration Manager to UCMDB, external applications can consume its analysis information in various ways:

- Use UCMDB's reporting functionality to generate and schedule reports on top of Configuration Manager's data.

- Consume Configuration Manager's data in other HP applications, such as HP Business Service Management.

- Use Configuration Manager's analysis data as a basis for making decisions in other applications.

Configuration Manager exposes the following data for federation:

- **Policy compliance status** data includes information about current policy result data for managed CIs and the associated policies.

- **Authorization status** data includes information about the authorization status of managed CIs.

UCMDB provides the class model for the schema for the model to be shared, and uses a federation TQL query as the way to consume data in UCMDB on the fly. For details, see "Federating KPIs" in the *HP Universal CMDB Configuration Manager User Guide*.

KPI means **Key Performance Indicator**. The UCMDB provides the **CMKpiAdapter** to federate KPI information about policy and authorization status from Configuration Manager. The data that is federated from Configuration Manager populates the **Kpi** and **KpiObjective** CITs, and can be retrieved by TQL as described in .

How to Consume Federated KPI Data from Configuration Manager

This workflow provides a brief overview of the steps to be performed in UCMDB, in order to consume federated KPI data from Configuration Manager.

This task includes the following steps:

- "Create an Integration Point to Federate KPI Data" below

- "Create KPI Reports" on the next page

Create an Integration Point to Federate KPI Data

1. In the Integration Studio, create a new integration point.

2. Set the following adapter properties:

Field	Description
Adapter	Click the [...] button and select **CMKpiAdapter**.
Credentials ID	Do the following: a. Click the [...] button. b. Select **Generic Protocol** and click **OK**. c. Click the [+] button to add the credentials to connect to the Configuration Manager database. Enter credentials for the user who has Manage, Authorize, and Access to UI permissions. d. When finished, click **OK**.
Hostname/IP	Provide the host name or IP address of the Configuration Manager database.
Integration Name	Enter a name for the new integration point.
Port	Enter the port number that is used for communication with the Configuration Manager database.
Use SSL	Select **False**. You cannot use secured communication to federate data from Configuration Manager.

3. Click **Test Connection** to make sure that you have configured the integration point correctly.

4. Click **OK** to save the integration point.

5. Select the KPI and KPIObjective CI types in the Supported and Selected CI Types tree.

6. Click the 💾 button to save the integration point.

For further details about creating integration points, see the section about the Integration Studio in the *HP Universal CMDB Data Flow Management Guide*.

Create KPI Reports

You can create KPI reports based on the CIs in a view, a custom TQL query, or business services.

1. In UCMDB, create a new view based on a custom TQL or copy an existing view.

> **Note:** When using a custom TQL query, make sure you take into account the limitations of the data capacity when using federation. You should filter the CIs in the TQL query to take this limitation into account. For details, see .

2. For each configuration item that you want to associate with a policy, attach the selected CI to the Kpi CI type and the Kpi CI type to the KpiObjective CI type, using composition links. The cardinality should be 0..* if you also want to obtain results for CIs that do not have associated KPI information.

> **Note:** If you want to create a business services report, select the BusinessService CI type when creating the TQL query.
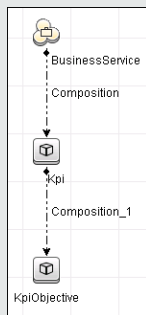
3. Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.

4. Set the hierarchy. An example is shown below:
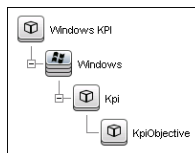
5. Add properties for the KpiObjective CI type to the report layout: An example is shown below:

6. If desired, you can schedule these reports to be created periodically. For details, see the *HP Universal CMDB Data Flow Management Guide*.

For details about creating reports, see the section about reports in the *HP Universal CMDB Modeling Guide*.

Troubleshooting and Limitations

- Federation only works with CIs in the actual state. Therefore:

  - Policy compliance is federated only for CIs in the actual state.

  - The authorization status for CIs that were deleted from the actual state is not shown.

- SSL communication for KPI data federation is not supported.

- The maximum number of CIs that can be federated is configurable. For details about changing this number, edit the value of the Max Num To Federate setting in the Infrastructure Settings Manager in UCMDB. For details about changing settings, see the Infrastructure Settings Manager chapter in the *HP Universal CMDB Administration Guide*. The recommended number of CIs is no more than 20,000, if large views have been enabled in Configuration Manager. For details about enabling support for large views, see the section describing large capacity planning in the interactive *HP Universal CMDB Deployment Guide*.

- If the test connection fails, click **Details** and check the first error in the stack trace for more information.

Chapter 5: HP Configuration Manager – Federating Policy Data

This chapter includes:

Overview

The federation mechanism that is built into HP Universal CMDB enables UCMDB to be used as a contact repository for sharing data among external applications, without duplicating it. By federating data from Configuration Manager to UCMDB, external applications can consume its analysis information in various ways:

- Use UCMDB's reporting functionality to generate and schedule reports on top of Configuration Manager's data.

- Consume Configuration Manager's data in other HP applications, such as HP Business Service Management.

- Use Configuration Manager's analysis data as a basis for making decisions in other applications.

Configuration Manager exposes **Policy compliance status** data (which includes information about current policy result data for managed CIs and the associated policies) for federation.

UCMDB provides the class model for the schema for the model to be shared, and uses a federation TQL query as the way to consume data in UCMDB on the fly. For details, see "Federating Policy Compliance Data" in the *HP Universal CMDB Configuration Manager User Guide*.

UCMDB provides the **CMPolicyAdapter** to federate policy data from Configuration Manager, which populates the **Policy** and **PolicyResult** CITs, and can be retrieved by TQL as described in "Create Policy Reports Based on CIs in a View or Custom TQL query" on page 94 and "Create summary policy reports based on the CIs in a view or a custom TQL query" on page 95.

How to Consume Federated Policy Data from Configuration Manager

This workflow provides a brief overview of the steps to be performed in UCMDB, in order to consume federated data from Configuration Manager.

This task includes the following steps:

- "Create an Integration Point to Federate Policy Compliance Data" below

- "Create Policy Reports Based on CIs in a View or Custom TQL query" on the next page

- "Create summary policy reports based on the CIs in a view or a custom TQL query" on page 95

Create an Integration Point to Federate Policy Compliance Data

1. In the Integration Studio, create a new integration point.

2. Set the following adapter properties:

Field	Description
Adapter	Click the ⋯ button and select **CMPolicyAdapter**.
Credentials ID	Do the following: a. Click the ⋯ button. b. Select **Generic DB Protocol (SQL)** and click **OK**. c. Click the ➕ button to add the credentials to connect to the Configuration Manager database. These should be the same credentials that were provided during the installation of Configuration Manager. d. When finished, click **OK**.
DB Name/SID	The database name or schema ID.
DB Type	Specify Oracle or MSSQL, as required.
Hostname/IP	Provide the host name or IP address of the Configuration Manager database.
Integration Name	Enter a name for the new integration point.
Port	Enter the port number that is used for communication with the Configuration Manager database.

3. Click **Test Connection** to make sure that you have configured the integration point correctly.

4. Click **OK** to save the integration point.

5. Select the Policy and PolicyResults CI types in the Supported and Selected CI Types tree.

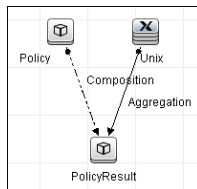6. Click the ⊞ button to save the integration point.

For further details about creating integration points, see the section about the Integration Studio in the *HP Universal CMDB Data Flow Management Guide*.

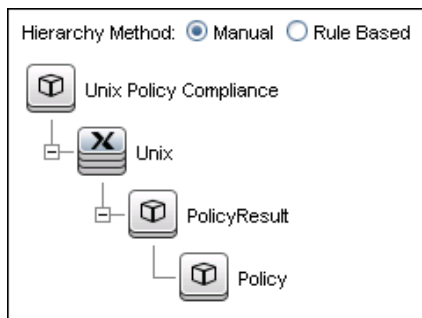## Create Policy Reports Based on CIs in a View or Custom TQL query

1. Create an integration point as described in , if one does not already exist.

2. In UCMDB, create a new view with a custom TQL query, or copy an existing view.

   > **Note:** When using a custom TQL query, make sure you take into account the limitations of the data capacity when using federation. You should filter the CIs in the TQL query to take this limitation into account. For details, see

3. For each configuration item that you want to associate with a policy, attach the Policy CI type and the selected CI to the PolicyResult CI type, using composition and aggregation links accordingly. The cardinality should be 0..* if you also want to obtain results for CIs that do not have associated policy information. An example is shown below:


4. Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.

5. Set the hierarchy. An example is shown below:

6. Add properties for the Policy CI type to the report layout: An example is shown below:

7. Add properties for the PolicyResult CI type to the report layout. An example is shown below:
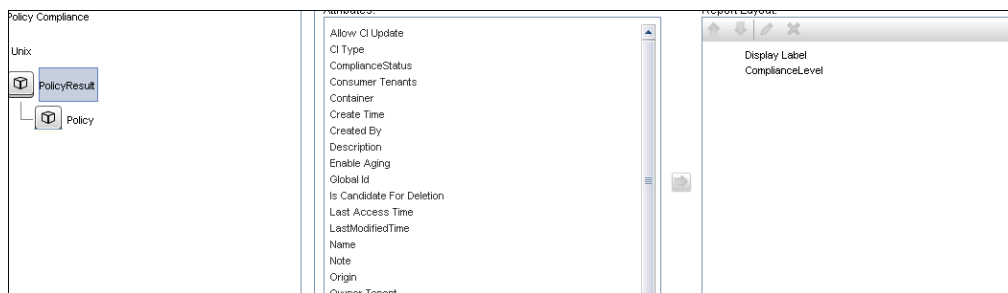
8. If desired, you can schedule these reports to be created periodically. For details, see the *HP Universal CMDB Data Flow Management Guide*.
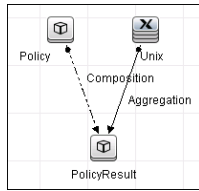
For details about creating reports, see the section about reports in the *HP Universal CMDB Modeling Guide*.

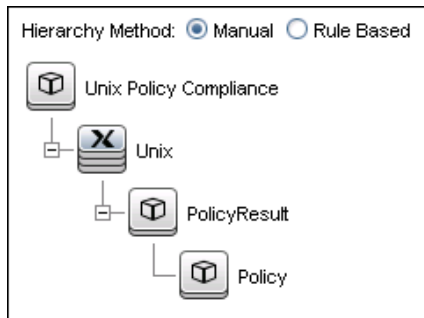## Create summary policy reports based on the CIs in a view or a custom TQL query

1. Create an integration point as described in , if one does not already exist.

2. In UCMDB, create a new view or copy an existing view.

> **Note:** When using a custom TQL query, make sure you take into account the limitations of the data capacity when using federation. You should filter the CIs in the TQL query to take this limitation into account. For details, see .

3. For each configuration item that you want to associate with a policy, attach the Policy CI type and the selected CI to the PolicyResult CI type, using composition and aggregation links accordingly. The cardinality should be 0..* if you also want to obtain results for CIs that do not have associated policy information. An example is shown below:

4. Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.

5. Set the hierarchy. An example is shown below:

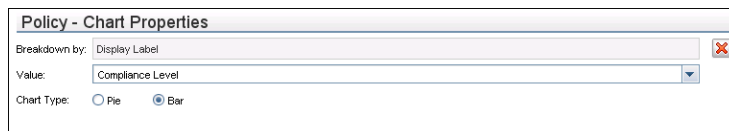6. Create an aggregation function for the Policy CI type. An example is shown below:
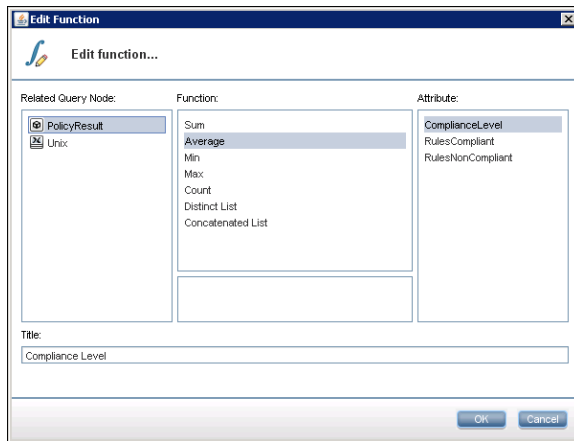
7. Add properties for the Policy CI type to the report layout. An example is shown below:

8. Add properties for the ConfigurationItem CI type to the report layout. An example is shown below:

9. Change the report format to a bar chart. An example is shown below:

10. If desired, you can schedule these reports to be created periodically. For details, see *HP Universal CMDB Data Flow Management Guide*.

For details about creating reports, see the section about reports in the *HP Universal CMDB Modeling Guide*.

Troubleshooting and Limitations

- Federation only works with CIs in the actual state. Therefore:

  - Policy compliance is federated only for CIs in the actual state.

  - The authorization status for CIs that were deleted from the actual state is not shown.

- The maximum number of CIs that can be federated is configurable. For details about changing this number, edit the value of the Max Num To Federate setting in the Infrastructure Settings Manager in UCMDB. For details about changing settings, see the Infrastructure Settings Manager chapter in the *HP Universal CMDB Administration Guide*. The recommended number of CIs is no more than 20,000, if large views have been enabled in Configuration Manager. For details about enabling support for large views, see the section describing large capacity planning in the *HP Universal CMDB Configuration Manager Deployment Guide*.

- If the test connection fails, click **Details** and check the first error in the stack trace for more information.

Chapter 6: HP Discovery and Dependency Mapping Inventory Integration

This chapter includes:

Overview

This document describes how to integrate Discovery and Dependency Mapping Inventory (DDMI) with UCMDB. Integration occurs by populating the UCMDB database with devices, topology, and hierarchy from DDMI and by federation with DDMI's supported classes and attributes. This enables change management and impact analysis across all business services mapped in UCMDB.

According to UCMDB reconciliation rules, if a CI is mapped to another CI in the CMDB, it is updated during reconciliation; otherwise, it is added to the CMDB.

Supported Versions

DDMI integration has been developed and tested on HP Universal CMDB version 7.5.2 or later with ED version 2.20 or DDMI version 7.5.

DDMI Adapter

Integration with DDMI is performed using a DDMI adapter, which is based on the Generic DB Adapter. This adapter supports full and differential population for defined CI types as well as federation for other CI types or attributes.

The DDMI adapter supports the following features:

- Full population of all instances of the selected CI types.

- Identifying changes that have occurred in DDMI, to update them in UCMDB.

- Implementing **Remove** in DDMI. When a CI is removed in DDMI, it is not physically deleted from the database, but its status is changed to indicate that the CI is no longer valid. The DDMI adapter interprets this status as an instruction to remove the CI when needed.

- Federation of defined CI types and attributes.

Out-of-the-box integration with DDMI includes population of the following classes:

- Node (some of the attributes are populated and some are federated)

- Layer2 connection

- Location that is connected to the node

- IP address

- Interface

In addition, the following classes can be defined as federated from DDMI:

- Asset

- CPU

- File system

- Installed software

- Printer

- Cost center

The following classes and attributes should be marked as federated by the DDMI adapter for the proper functionality of the Actual State feature of Service Manager:

- **Classes**

  - Person

  - Asset

  - CPU

  - Installed software

  - Printer

  - Windows service

- **Node attributes**

  - DiscoveredOsVendor

  - DiscoveredModel

  - Description

  - DomainName

  - DiscoveredLocation

  - NetBiosName

Note: Avoid marking the **CreateTime** and **LastModifiedTime** attributes as federated, as it may lead to unexpected results.

How to Populate the CMDB with Data from DDMI

This task describes how to install and use the DDMI adapter, and includes the following steps:

- "Define the DDMI integration" below

- "Define a population job (optional)" on the next page

- "Run the population job" on the next page

1. **Define the DDMI integration**

   a. In UCMDB, navigate to **Data Flow Management > Integration Studio**.

   b. Click the **new integration point** [icon] button to open the new integration point Dialog Box.

      ○ Click [...], select the DDMI adapter, and click **OK**.

        Each out-of-the-box adapter comes predefined with the basic setup needed to perform integration with UCMDB. For information about changing these settings, see "Integration Studio Page" in the *HP Universal CMDB Data Flow Management Guide*.

      ○ Enter the following information, and click **OK**:

Name	Description
Credentials	Allows you to set credentials for integration points. For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.
Hostname/IP	The name of the DDMI server.
Integration Name	The name you give to the integration point.
Is Integration Activated	Select this check box to create an active integration point. You clear the check box if you want to deactivate an integration, for instance, to set up an integration point without actually connecting to a remote machine.
Port	The port through which you access the DDMI database.

   c. Click **Test connection** to verify the connectivity, and click **OK**.

   d. Click **Next** and verify that the following message is displayed: **A connection has been successfully created**. If it does not, check the integration point parameters and try again.

2. **Define a population job (optional)**

   The DDMI adapter comes out-of-the-box with the DDMI Population job, which runs the following predefined queries: **hostDataImport**, **networkDataImport**, **printerDataImport**, and **Layer2DataImport**. For details about these queries, see "Predefined Queries for Population Jobs" on page 104. This job runs according to a default schedule setting.

   You can also create additional jobs. To do this, select the Population tab to define a population job that uses the integration point you defined in "Define the DDMI integration" on the previous page. For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

3. **Run the population job**

   Activate the population job in one of the following ways:

   - To immediately run a full population job, click . In a full population job, all appropriate data is transferred, without taking the last run of the population job into consideration.

   - To immediately run a differential population job, click . In a differential population job, the previous population time stamp is sent to DDMI, and DDMI returns changes from that time stamp to the present. These changes are then entered into the UCMDB database.

   - To schedule a differential population job to run at a later time or periodically, define a scheduled task. For details, see "Define Tasks that Are Activated on a Periodic Basis" in the *HP Universal CMDB Administration Guide*.

How to Federate Data with DDMI

The following steps describe how to define the CI types that will be federated with DDMI.

1. In UCMDB, navigate to **Data Flow Management > Integration Studio**.

2. Select the integration point that you defined in "Define the DDMI integration" on the previous page.

3. Click the **Federation** tab. The panel shows the CI types that are supported by the DDMI adapter.

4. Select the CI types and attributes that you want to federate.

5. Click **Save** .

How to Customize the Integration Data Model in UCMDB

Out-of-the-box CIs for DDMI integration can be extended in one of the following ways:

To add an attribute to an existing CI type:

If the attribute you want to add does not already exist in the CMDB, you need to add it. For details, see "Add/Edit Attribute Dialog Box" in the *HP Universal CMDB Modeling Guide*.

1. Go to the **orm.xml** file as follows: **Data Flow Management > Adapter Management > DDMiAdapter > Configuration Files > orm.xml**.

2. Locate the **generic_db_adapter.[CI type]** to be changed, and add the new attribute.

3. Ensure that the TQL queries that include this CI Type have the new attribute in their layouts, as follows:

   a. In the Modeling Studio, right-click the node where you want to include the attribute.

   b. Select **Query Node Properties**.

   c. Click **Advanced layout settings** and select the new attribute.

   For details about selecting attributes, see "Layout Settings Dialog Box" in the *HP Universal CMDB Modeling Guide.* For limitations on creating this TQL query, see

To add a new CI type to the DDMI Adapter:

1. In UCMDB, create the CI type that you want to add to the adapter, if it does not already exist. For details, see "Create a CI Type" in the *HP Universal CMDB Modeling Guide*.

2. Go to the **orm.xml** file as follows: **Data Flow Management > Adapter Management > DDMiAdapter > Configuration Files > orm.xml**.

3. Map the new CI type by adding a new entity called **generic_db_adapter.[CI type]**.

4. In the **orm.xml** file, ensure that the new CI type has the following mappings:

   a. the **data_note** attribute is mapped to the **NMID_StatusInAppliance** column (this attribute is used for checking the CI's status).

   b. the **last_modified_time** and **create_time** attributes are mapped to the **Device_UpdatedDt** and **Device_FirstFoundDt** columns.

   For details, see "The orm.xml File" in the *HP Universal CMDB Developer Reference Guide*.

5. Create queries to support the new CI types that you added. Make sure that all mapped attributes have been selected in the Advanced Layout settings:

   a. In the Modeling Studio, right-click the node where you want to include the attribute.

   b. Select **Query Node Properties**.

   c. Click **Advanced layout settings** and select the new attribute.

For details about selecting attributes, see "Layout Settings Dialog Box" in the *HP Universal CMDB Modeling Guide.* For limitations on creating this TQL query, see "Troubleshooting and Limitations" on the next page.

6. In UCMDB, navigate to **Data Flow Management > Integration Studio**.

7. Edit the DDMI integration point to support the new CI type by selecting it either for population or for federation.

8. If the new CI type is for population, edit the population job that you created in "Define a population job (optional)" on page 102 to include the new TQL query.

Predefined Queries for Population Jobs

The following TQL queries (located in the Modeling Studio in the **Integration\Data In** folder) are provided out-of-the-box if you use the DDMI adapter when you create an integration point:

- **hostDataImport** - use to import nodes. Imported data includes nodes whose NodeRole attribute is either null, or contains **desktop**, **server**, or **virtualized_system**. Nodes are identified either by their interface or IP address. Information also includes the location of the nodes (building, floor and room).

- **networkDataImport** - use to import nodes that are not imported with **hostDataImport**. Similar to **hostDataImport**, except that it imports nodes whose NodeRole is not null and does not contain the following strings: **desktop**, **server**, **virtualized_system**, or **printer**.

- **printerDataImport** - use to import printers. Similar to **networkDataImport**, except that it does import nodes whose NodeRole contains the string **printer**.

- **Layer2DataImport** - use to import Layer2 connections between pairs of nodes through their interfaces. Information also includes the nodes and their IP addresses.

DDMI Adapter Configuration Files

The adapter includes the following configuration files:

- **orm.xml.** The Object Relational mapping file in which you map between UCMDB classes and database tables.

- **discriminator.properties.** Maps each supported CI type (also used as a discriminator value in **orm.xml**) to a list of possible corresponding values of the discriminator column, **DeviceCategory_ID**.

- **replication_config.txt.** Contains a comma-separated list of non-root CI and relations types that have a **Remove** status condition in the DDMI database. This status condition indicates that the

device has been marked for deletion.

- **fixed_values.txt.** Includes a fixed value for the attribute **ip_domain** in the class IP (**DefaultDomain**).

For details on adapter configuration, see "Developing Generic Database Adapters" in the *HP Universal CMDB Developer Reference Guide*.

Troubleshooting and Limitations

> **Note:** Only queries that meet these requirements are visible to the user when selecting a query for a population job.

- Queries that are used in population jobs should contain one CI type that is labeled with a **Root** prefix, or one or more relations that are labeled with a **Root** prefix.

  The root node is the main CI that is synchronized; the other nodes are the contained CIs of the main CI. For example, when synchronizing the **Node** CI Type, that graph node is labeled as **Root** and the resources are not labeled **Root**.

- The TQL graph must not have cycles.

- A query that is used to synchronize relations should have the cardinality 1...* and an OR condition between the relations.

- The adapter does not support compound relations.

- The TQL graph should contain only CI types and relations that are supported by the DDMI adapter.

- ID conditions on the integration TQL query are not supported.

Chapter 6: HP IT Executive Scorecard

This chapter includes:

Overview

HP Universal CMDB (UCMDB) is based on three key elements: a rich data model, visualization, and federation to additional data repositories. UCMDB provides impact analysis, change tracking, and reporting capabilities to transform UCMDB data into comprehensible, actionable information that helps answer critical questions and solve business problems. UCMDB can provide valuable information about services, and their operational status.

The purpose of the integration of UCMDB as a data source is to bring quality management information into the Data Warehouse.

You use the UCMDB push adapter to access data links and nodes from UCMDB, which are then pushed into the Data Warehouse. You must deploy the push adapter package when the UCMDB Content Pack is activated in the Data Source Management UI, and before the first run of the ETL in order to get the Node, Application and Services topology from UCMDB.

Supported Versions

UCMDB supports integration with Executive Scorecard version 9.31 or later.

How to Push Data from UCMDB to Executive Scorecard

1. **Prerequisite - Set up Executive Scorecard**

   Perform the steps "Activate the Integration" and "Connect to UCMDB on a Secured Connection" as described in the Executive Scorecard manual *Content Reference Guide for the Integration with HP Universal Configuration Management Database*.

2. **Prerequisite - Ensure that the push adapter package has been deployed**

   > **Note:** If you have installed UCMDB version 10.10 or later with CP14, the push adapter package is installed by default. You can skip this step.

   a. In UCMDB, go to **Data Flow Management > Adapter Management**.

   b. In the Resources pane, search for **XSAdapter**.

3. **Create an integration point**

   a. Complete the integration and adapter properties fields as shown in the following table:

Field	Required	Description
Integration Properties		

Field	Required	Description
Integration Name	Yes	Enter a unique name for the integration point.
Integration Description	No	Enter a description of the current integration point.
Adapter	Yes	Click the ⋯ button and select **HP BTO Products > Executive Scorecard > Executive Scorecard**.
Is Integration Activated	Yes	Select this check box to create an active integration point. You clear the check box if you want to deactivate an integration, for instance, to set up an integration point without actually connecting to a remote machine.
Adapter Properties		
Hostname/IP	Yes	The hostname or IP Address of the Executive Scorecard database.
Port	Yes	**1433** (the communication port of the Executive Scorecard database).
Database	Yes	The staging database name of the Data Warehouse in Executive Scorecard.
Schema	Yes	**dws** (the name of the database schema).

Field	Required	Description
Credentials ID	Yes	Do the following: i. Click the [...] button. ii. Select **Generic DB Protocol (SQL)** and click **OK**. iii. Click the ✳ button to add new credentials. Enter the following information: ○ **Database Type**: Select Microsoft SQL Server. ○ **Port Number**: Leave the default value (1433). ○ **Connection Timeout**: Leave the default value (20000). ○ **User Name** and **Password**: Enter the user name and password for connecting to the Executive Scorecard Data Warehouse database. iv. When finished, click **OK**.
Data Flow Probe	Yes	The name of the Data Flow Probe on which the integration runs. Select a Data Flow Probe whose range includes the IP address of the Executive Scorecard database.
Additional Probes	No	Select additional probes to use when pushing to AM in order to increase redundancy.

For details about how to create an integration point, see "How to Set Up an Integration Point" in the *HP Universal CMDB Data Flow Management Guide*.

   b. Click **OK**.

   c. Click **Test Connection**. If the connection fails, verify that the information provided is correct.

   d. Click **OK** to close the dialog box and return to the Integration Studio.

4. **Edit the existing integration job (optional)**

A default integration job called **XS Push job** is created when you create an integration point using the **Executive Scorecard** adapter. You can edit this job to change the default queries or the job's schedule.

    a. Select the integration point that you created in step 4 and click **XS Push job**.

    b. Click [pencil icon].

    c. Update the TQL queries listed or schedule as required.

    d. When you are finished, click **OK**.

5. **Run the job**

    a. In the Integration Studio, select the integration point that you created in step 4.

    b. Select **XS Push job** and click [icon] to run a full synchronization.

Executive Scorecard Adapter

This section contains information about the **XSAdapter**.

Input CIT

destination_config

Triggered CI Data

Name	Value
adapterId	${ADAPTER.adapter_id}
attributeValues	${SOURCE.attribute_values}
credentialsId	${SOURCE.credentials_id:FED#_NA}
destinationId	${SOURCE.destination_id}

Parameters

Name	Value
credentialsId	
database	
host	
port	
schema	dws

Chapter 7: HP Network Automation (NA) Integration

This chapter includes:

Overview

HP Network Automation (NA) provides an enterprise class solution that tracks and regulates configuration and software changes across routers, switches, firewalls, load balancers, and wireless access points. NA provides visibility into network changes, enabling the IT staff to identify and correct trends that could lead to problems, while mitigating compliance issues, security hazards, and disaster recovery risks. NA also captures full audit trail information about each device change.

Network engineers can use NA to pinpoint the following:

- Which device configuration changed?

- What exactly was changed in the configuration?

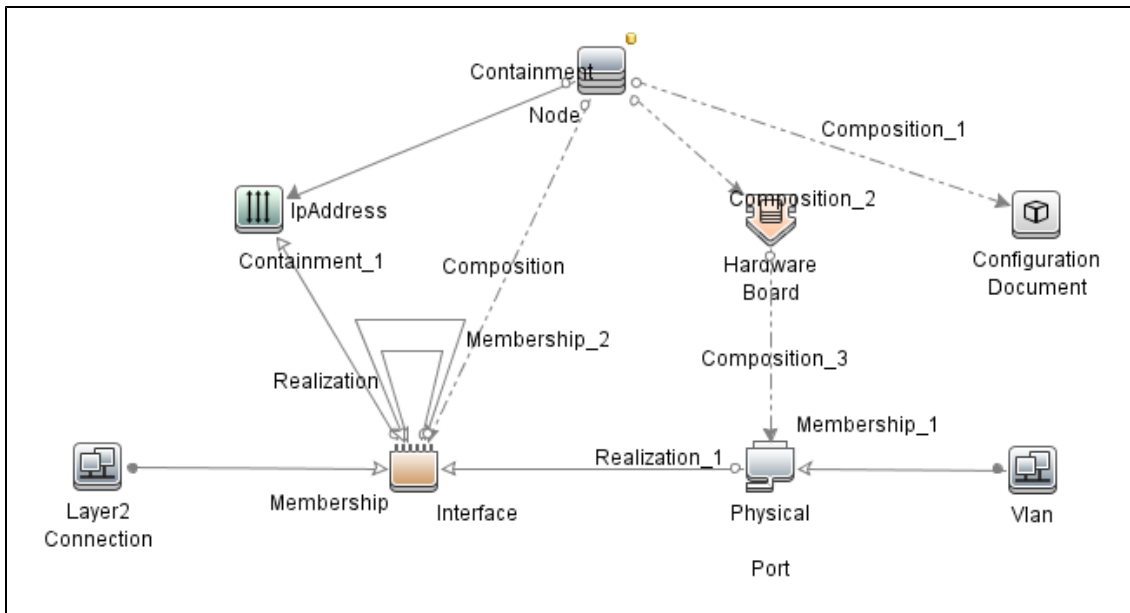- Who made the change?

- Why the change was made?

In addition, NA can enforce security and regulatory policies at the network level by making sure that configurations comply with pre-defined standards.

Supported Versions

This integration supports HP Network Automation 9.22.

Topology

The diagram below depicts the HP NA topology.

How to Pull Data Topology from an HP NA Server using a Java Client

The HP NA server provides the Java client library, which allows executing predefined commands and retrieving topology data remotely. Communication is done over RMI in binary format.

This section describes how to pull topology data from the HP NA server to the UCMDB using a Java client.

Prerequisites

1. Ensure that there is connectivity to the HP NA server on the RMI port. The default port is 1099.

2. The user performing the connection must have the correct permissions to execute the **list** and **show** commands.

Integration Flow

1. Create a new integration point in the Integration Studio based on the **Pull topology from HP Network Automation** adapter as follows:

   a. In UCMDB, go to **Data Flow Management > Integration Studio**.

   b. Click the **New Integration Point** button. The **New Integration Point** dialog box opens.

   c. Enter a name and description for the adapter.

   d. Check the **Is Integration Activated** check box.

   e. In the Adapter field, click the **Select Adapter** button. The Select Adapter dialog box opens.

   f. Go to HP Software Products>HP Network Automation, select **Pull Topology from HP Network Automation**, and click **OK**.

   g. In the Adapter Properties section, define the following properties:

   > **Note:** Most of the properties listed below are assigned default values that are displayed after selecting **Pull Topology from HP Network Automation** in the previous step.

   - **Credentials ID.** Click the **Select Credentials ID** button and in the Choose Credentials dialog box enter the HP NA Java protocol credentials to be used for connection.

- ○ **queryTopologyPerDevice.** This parameter affects how information about devices is queried. When disabled, topology entities are retrieved in single queries. When enabled, separate query is issued to retrieve data per device. The default value is false.

- ○ **remoteJVMClasspath.** This is the class path of the external JVM process. The default value assigned here should not be changed.

- ○ **reportDeviceConfigs.** This parameter specifies whether the latest configuration of devices should be reported as **configuration_document** CIs. The default value is false.

- ○ **runInSeparateProcess.** This parameter enables the job to run in an external JVM process, separate from the main probe process. The default value is **true** and this value should not be changed.

- ○ **Data Flow Probe.** This parameter defines the Probe to be used for integration. Select **DataFlowProbe** to explicitly choose the probe, or **Auto-Select** if you want the probe to be automatically selected according to where the destination IP is assigned.

- ○ **Trigger CI instance.** This parameter defines the IP Address of the target HP NA server.

h. Click **OK** in the New Integration Point dialog box.

2. Run the full synchronization of the **HP Network Automation by Java job** with the integration point that you just created. This job performs the following internal actions:

a. Establishes a connection to the target HP NA server.

b. Performs a query to retrieve IDs of all devices in the system.

c. Retrieves all devices in chunks. The default chunk size is 500 devices.

d. Retrieves all ports on all devices. Depending on the value of parameter **queryTopologyPerDevice**, the ports are retrieved per device or from all devices at once.

e. Retrieves all device modules. Depending on the value of parameter **queryTopologyPerDevice**, the device modules are retrieved per device or from all devices at once.

f. Retrieves all VLANs.

g. Retrieves connections between devices. Depending on the value of parameter **queryTopologyPerDevice**, the connections are retrieved per device or from all devices at once.

h. Retrieves configuration of each device as textual data (if **reportDeviceConfigs=true**).

i. Analyzes topology. Port channels are identified with related child ports, port aliases are identified along with the parent port.

j. Reports the topology. The job forms result vectors with information about devices and their connections, while trying to not exceed 10000 objects in one vector. Vectors are sent to the UCMDB server.

Pull Topology from HP NA Adapter

This section contains information on the Pull Topology from HP NA Adapter.

Input CIT

IpAddress

Triggered CI Data

Name	Value
ip_address	${SOURCE.name}

Used Scripts

- network_automation.py

- na_discover.py

- na_integration_by_java.py

Discovered CITs

- Composition

- ConfigurationDocument

- Containment

- HardwareBoard

- Interface

- IpAddress

- Layer2Connection

- Membership

- Node

- PhysicalPort

- Realization

- Vlan

Limitations

Delta synchronization is not supported by the Pull Topology from HP NA adapter. It is not possible to run the HP Network Automation by Java job and synchronize only changes since the last run.

Chapter 8: HP Network Node Manager (NNMi) Integration

This chapter includes:

Overview

You integrate NNMi with UCMDB using the Data Flow Management (DFM) application.

When you activate the NNMi integration, DFM retrieves Layer 2 network topology data from NNMi and saves the data to the UCMDB database. Users can then perform change management and impact analysis.

Use Cases

The documentation for this integration is based on the following use cases:

- **Use Case 1:** A UCMDB user wants to view the Layer 2 network topology supporting servers and applications. The requirement is to use NNMi as the authoritative source for that information with access through the UCMDB application.

- **Use Case 2:** An NNMi operator wants to view the impact of a network access switch infrastructure failure where the impact data is available in UCMDB. The NNMi operator selects an incident or a node in NNMi and then enters a request for impacted CIs.

Supported Versions

Out of the box, the following software versions are supported:

- Data Flow Probe version 9.02 or later

- HP NNMi version 8.1, 8.11, and 9.x

NNMi – UCMDB Integration Architecture
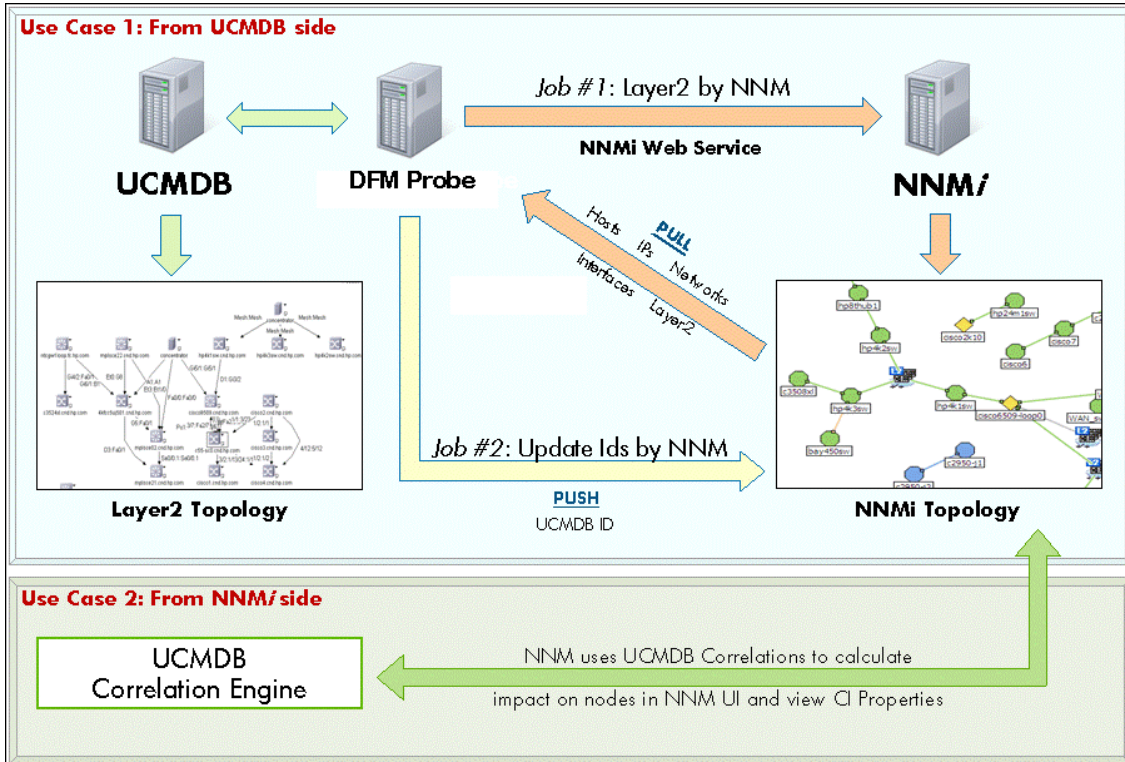
Topology

Layer2 by NNM Job

> **Note:** For a list of discovered CITs, see "Discovered CITs" on page 127.
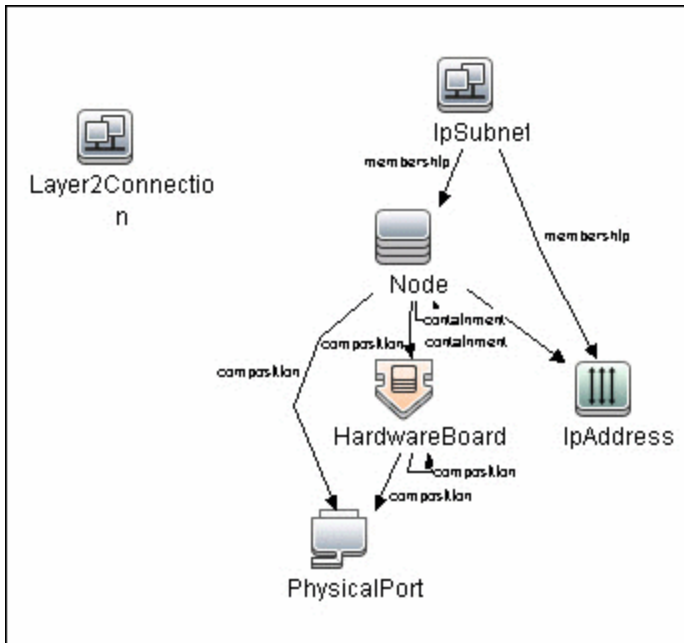
How to Run NNMi–UCMDB Integration

This task includes the steps to run the NNMi-UCMDB integration jobs.

> **Note:** To avoid conflict, do not run the UCMDB Layer 2 discovery jobs when running the NNMi integration.

This task includes the following steps:

1. **Run NNMi Integration**

   In the Integration Studio, create a new integration point.

   a. Provide a name and description for the integration point.

   b. Under **Integration Properties > Adapter**, select the appropriate adapter:

- **Pull Topology from NNMi**: Use this adapter to run population jobs.

  This job connects to the NNMi Web service and retrieves NNMi discovenodes, IPs, networks, interfaces, physical ports, VLANs, hardware boarLayer 2 connection information to create a Layer 2 topology in UCMDB.

- **NNMI Update IDs**: Use this adapter to run push jobs. This job:

  - Retrieves the UCMDB IDs of the NNMi hosts from the UCMDB Server using the UCMDB Web Services API.

  - Updates the UCMDB_ID custom attribute on the corresponding node object on the NNMi server using the NNMi Web service.

c. Under **Adapter Properties > Data Flow Probe**, select the **Data Flow Probe**.

d. Under **Adapter Properties > Trigger CI instance** select:

- **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI; or

- **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

> **Note:** For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.

e. Under **Adapter Properties > Credentials ID** select the appropriate credentials for connection to the NNMi server.

f. Save the Integration Point.

g. Run the job.

> **Note:** For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

2. **Validate results**

   Verify that data was discovered using the NNMi integration jobs.

   a. For the **NNMi population** job:

   - In UCMDB, navigate to **Managers > Modeling** > **IT Universe Manager.**

   - In the **CI Selector** pane, select **View Browser**.

- In the **View** drop-down menu, select **Layer 2**. Select a view. The view displays the CIs and relationships discovered by the integration job.

b. For the **NNMi push** job:

- In NNMi, open an NNMi node that was discovered in UCMDB.

- On the **Custom Attributes** tab, look for the **UCMDB_ID** custom attribute. This attribute should contain the UCMDB ID of the corresponding host in UCMDB.

How to Manually Add the IpAddress CI of the NNMi Server

> **Note:** When you installed HP Universal CMDB, you may have installed a bundled UCMDB that uses a Foundation license. If your UCMDB installation has a Foundation license deployed, use the steps in this section to manually add an **IpAddress** CI. If any other license (Basic or Advanced) is deployed on the UCMDB server, discover the IPAddress CI as described in "How to Run NNMi–UCMDB Integration" on page 119.

To manually add the IpAddress CI of the NNMi server

1. Verify that the Data Flow Probe is correctly installed and connected to the UCMDB Server.

2. Add the IP of the NNMi server to the Data Flow Probe range:

   In the **Data Flow Probe Setup** module, select the Probe that is to be used for the NNMi integration, and add the IP address of the NNMi server to its range. For details, see the section describing how to add Probe range in the HP Universal CMDB Data Flow Management Guide.

3. Insert the **Address** CI of the NNMi server in the CMDB:

   a. In **Modeling** > **IT Universe Manager,** in the CI Selector pane, click the **Browse Views** tab and select **Network Topology** from the **View** drop-down menu.

   b. Click the **New CI** ✳ button.

   c. In the New CI dialog box, select the **IpAddress** CIT from the tree and enter the following values:

Field	Description
IP Address	The IP address of the NNMi server.
IP Domain Name	The UCMDB domain name (for example, `DefaultDomain`).
IP Probe Name	The name of the Data Probe (for example, `DefaultProbe`).

   d. Save the **IpAddress** CI.

How to Set Up HP NNMi–HP UCMDB Integration

The following steps describe how to configure NNMi to communicate with UCMDB:

Configure the connection between NNMi and UCMDB

On the NNMi management server, do the following:

1. In the NNMi console, open the **HP NNMi–HP UCMDB Integration Configuration** form (**Integration Module Configuration > HP UCMDB**).

2. Select the **Enable Integration** check box to activate the remaining fields on the form.

3. Enter the information for connecting to the NNMi management server.

4. Enter the information for connecting to the UCMDB server.

5. Click **Submit** at the bottom of the form.

   A new window displays a status message. If the message indicates a problem with connecting to the UCMDB server, re-open the **HP NNMi–HP UCMDB Integration Configuration** form (or press **ALT+LEFT ARROW** in the message window), and then adjust the values for connecting to the UCMDB server as suggested by the text of the error message.

Customize the integration

On the NNMi management server, do the following:

1. In the NNMi console, open the **HP NNMi–HP UCMDB Integration Configuration** form (**Integration Module Configuration > HP UCMDB**).

2. Enter values for the following fields:

   - HP UCMDB Correlation Rule Prefix

   - HP UCMDB Impact Severity Level (1–9)

3. Click **Submit** at the bottom of the form.

NNMi Integration Job

Adapter Parameters

Parameter	Description
discoverDisabledIps	Defines whether the integration should discover disabled IPs. When set to false, the integration does not discover disabled IPs. **Default**: false.
discoverLayer2	Defines whether the integration should discover the Layer2Connection CIs from NNMi. When set to true, the integration fetches all the Layer2Connections-related data, iteratively querying for a specified number of Layer2Connections from NNMi (based on value of the **pageSizeLayer2** parameter), then querying for Network Interfaces on the ends of Layer2Connection and Nodes hosting these interfaces with instant push of collected topology to UCMDB. **Default:** true
discoverNodes	Defines whether the integration should discover all the Nodes that are registered in NNMi, regardless of their inclusion into Layer2 Topology or VLANs. When set to true, integration fetches all the Nodes with connected IpAddresses, Interfaces, HardwareBoards, Physical Ports and IpSubnets, iteratively querying for a specified number of Nodes with related data from NNMi (based on value of the **pageSizeNodes** parameter) and instantly pushing collected topology into UCMDB. **Default:** true
discoverNonManagedInterfaces	Defines whether the integration should discover non-managed interfaces. When set to false, the integration does not discover non-managed interfaces. **Default**: false.

Parameter	Description
discoverNonManagedNodes	Defines whether the integration should discover non-managed nodes. When set to false, the integration does not discover non-managed nodes. **Default**: false. When this parameter is set to true, **discoverDisabledIps** and **discoverNonManagedInterfaces** are ignored for non-managed nodes; the integration will discover everything, including disabled ips and non-managed interfaces.
discoverPhysicalPorts	Defines whether the integration should discover physical ports. When set to false, the integration does not discover physical ports. **Default**: false. When this parameter is set to false, the integration does not discover **VLANs** and **HardwareBoards**.
discoverVlans	Defines whether the integration should discover all the VLANs that are registered in NNMi. When set to true, integration fetches all the VLANs with member Physical Ports, Hardware Boards and Nodes hosting those Physical Ports and Node-related topology, iteratively querying for a specified number of VLANs (based on the value of pageSizeVlans parameter), getting all the necessary related topology and instantly reporting it back to UCMDB. **Default:** true This parameter is ignored if **discoverPhysicalPorts** is set to false.

Parameter	Description
discoveryMode	Specifies the adapter's mode of execution, either related or full. **Default:** related - In related read mode, only a small subset of the topology is read, starting from the root entity. For example, the following is possible: - read several Nodes - read all entities related to these Nodes, including Interfaces, Ip Addresses, Ports, and so on. - send date to UCMDB - continue with other Nodes - In full read mode, the entire topology is read into memory before the data is processed and result vectors are formed. This approach requires more memory, but reduces the number of requests to the NNMi server. The job sends the topology in chunks after more than 10,000 objects are accumulated for sending. The Data Flow Probe then processes these chunks and sends them to the UCMDB server. Use the **remoteJVMArgs** parameter of the integration adapter to increase the memory limits for the remote Java virtual machine that executes the job, in case there is a lot of data in NNMi. For example, setting the value of the **remoteJVMArgs** parameter to **Xmx4096m** enables the remote JVM to consume up to 4 GB of memory. As a general rule, reading 2 million objects from NNMi consumes approximately 10-12 GB of memory. For full read mode, it is recommended to set all page sizes to 1000 (using the **pageSizeLayer2**, **pageSizeNodes**, and **pageSizeVlans** parameters.
pageSizeLayer2	Defines the number of Layer2Connection CIs to fetch from NNMi per one query. **Default:** 5
pageSizeNodes	Defines the number of Nodes to fetch from NNMi per one query. **Default:** 10

Parameter	Description
pageSizeVlans	Defines the number of VLANs to be queries from NNMi per one query. **Default:** 1
requestCustomAttributes	Enables global querying for custom attributes of entities in the script. The script must still be modified to read the values and report them (this code modification is expected to be made by the customer). Without modification to the script, setting this parameter to true has no effect. For performance reason, by default, custom attributes are not read from the NNMi server . **Default:** false

Discovered CITs

- Composition

- Containment

- HardwareBoard

- Interface

- IPAddress

- IpSubnet

- Layer2Connection

- Membership

- Node

- PhysicalPort

- Realization

- Vlan

Note: To view the topology, see .

How to Customize Integration

This section describes how to customize the NNM Integration package in order to report additional attributes or entities, or to perform other changes.

Included Scripts

The NNM Integration package includes the scripts detailed in the following table:

Name	Description
nnmi_api.py	Implements API for accessing and retrieving data in NNM. Includes definition of base entities, such as **Node** or **Interface**, collections, topology and fetcher classes.
nnmi_filters.py	Support module used by **nnmi_api.py**. Includes definition of filters, allowing creation of advanced expressions when querying data in NNM. In general, you should not modify this script.
nnmi.py	Uses API provided by **nnm_api.py** to retrieve data, form topology into **ObjectStateHolder** objects and send result to UCMDB.
NNM_Integration.py	Main entry point of the integration. Contains **DiscoveryMain** method which is called by probe, and uses **nnmi.py**.
NNM_Integration_Utils.py	Previous version of NNM integration implementation, used by **CheckCred.py** script which validates credentials.
NNM_Integration_Utils_9.py	Previous version of NNM integration implementation for UCMDB 9.0
NNM_Update_Ids.py	Support module which updates custom attribute in NNM with UCMDB ID of the CI.

Customization Step by Step

The following steps illustrate customization of integration with an example, showing how you discover and report an additional attribute for an Interface. In this example, the attribute is the **managementMode** property in the NNM interface.

1. **Prerequisites**

   a. You need to know the **name** of the attribute in NNM. Here it is **managementMode**.

   b. You need to know the **name of a method** which should be called on the stub to read the corresponding property of entity, in this case Interface. This information is in the corresponding WSDL file or API documentation.

2. **Property Retrieval**

For the new property to be reported, it should be retrieved first. To do that, you must modify the **nnmi_api.py** script.

a. Locate Entity class

Open **nnm_api.py** and find the corresponding Entity class. Entity class names follow the pattern **Nms*Entity**. For this example, you want **NmsInterfaceEntity**.

```
nnmi_api.py

class NmsInterfaceEntity(BaseNmsEntity):
    ...
```

b. Add new entry to **field_names** tuple

Add a new field to the **field_names** tuple. This tuple is used for real-time entity introspection, mainly debugging.

> **Note:** For use in Python classes, translate a camel cased field name into an underscore separated field name. For example: **management_mode**.

```
nnmi_api.py

    field_names = (
        'id',
        'name',
        'hosted_on_id',
        'connection_id',
        'if_index',
        'if_alias',
        'if_descr',
        'if_name',
        'if_speed',
        'physical_address',
        'if_type',
        'uuid',
        'management_mode',

    )
```

c. Modify constructor of the Entity to read the new property

Modify **__init__ method** of the entity to read the new property. Modification depends on whether you need property post-processing.

    i. Simple read

      Call the method on the stub.

```
nnmi_api.py
class NmsInterfaceEntity(BaseNmsEntity):
    ...
    def __init__(self, item, fetcher):
        ...
        self.if_type          = item.getIfType()

        self.uuid             = item.getUuid()

        self.management_mode = item.getManagementMode()
```

ii. Read with post processing

Delegate reading of the property to a new method which performs post processing for the retrieved value. In this case the new method is added to an **NmsInterfaceEntity** class called **_get_management_mode** which just strips the value of white spaces and makes it lowercase.

```
nnmi_api.py
class NmsInterfaceEntity(BaseNmsEntity):
    ...
    def __init__(self, item, fetcher):
        ...
        self.if_type          = item.getIfType()

        self.uuid             = item.getUuid()

        self.management_mode = self._get_management_mode(item)

        ...
    def _get_management_mode(self, item):
        management_mode = item.getManagementMode()

        if management_mode:
            return management_mode.strip().lower()
        else:
            return ''
```

3. **Property reporting**

Once the modifications to **nnmi_api.py** are done, the new property is available, but it is not reported yet. To add reporting for this property, you need to modify the **nnmi.py** script.

a. Locate **Builder** class

Open the **nnmi.py** script and find the corresponding **Builder** class for the entity being reported. All **Builder** names match the ***Builder** pattern. In your case, you need the **InterfaceBuilder** class.

```
nnmi.py
class InterfaceBuilder:
    ...
```

b. Locate a place where **ObjectStateHolder** is created

Each **Builder** has a build method and several other methods which assist in the creation of the **ObjectStateHolder**. Find where this **ObjectStateHolder** is created either directly or indirectly by calling the method in the **modeling.py** module. In your example, **ObjectStateHolder** is created in the method **_createInterfaceOsh**.

```
nnmi.py
class InterfaceBuilder:
    ...
    def _createInterfaceOsh(self, interface):
        interfaceOsh = modeling.createInterfaceOSH( ...

        self._setIsPseudoAttribute(interface, interfaceOsh)

        return interfaceOsh
```

c. Add new attribute reporting

Now you add reporting of the new property. Here, you report the new value to the attribute **interface_management_mode** of the **Interface** CIT which you previously created.

```
nnmi.py
class InterfaceBuilder:
    ...
    def _createInterfaceOsh(self, interface):
        interfaceOsh = modeling.createInterfaceOSH( ...

        self._setIsPseudoAttribute(interface, interfaceOsh)

        interfaceOsh.setAttribute('interface_management_mode',
interface.management_mode)

        return interfaceOsh
```

Troubleshooting and Limitations

This section describes troubleshooting and limitations for NNMi Integration.

- **Problem:** The NNMi Web service responds with a **cannot interrogate model** message.

  Solution: This message usually indicates that the Web services request made to the NNMi server is incorrect or too complex to process. Check the NNMi jbossServer.log file for details.

- **Problem:** If an excessive number of nodes are to be updated with the same UCMDB ID, it may take a while for the update adapter to complete.

  Solution: The volume of data retrieved from the NNMi server might be large. The recommended memory requirements for the Data Probe process is 1024 MB. Since the NNMi Web service enables updating the individual nodes one at a time, the time to update the nodes may take a while.

- **Problem:** The **Layer 2 by NNM** job finishes with the following warning: `Failed to get any Layer 2 links from NNM.`

  Solution: Refer to technical article KM629927 on the HP support Web site at http://support.openview.hp.com.

- **Problem:** Either of the NNMi integration jobs fails with the following error in the DFM log files: `com.hp.ov.nms.sdk.node.NmsNodeFault: Cannot interrogate model.`

  Solution: This error typically means that the NNMi server failed to process the Web services call. Check the following two logs on the NNMi server for exceptions when the integration was activated:

  - jbossServer.log

  - sdk.0.0.log

- **Problem:** Either of the NNMi integration jobs fail with the following error: `Could not find Discovery Probe 'DefaultProbe'. Task for TriggerCI will not be created.`

  Solution:

  a. Right-click the job and select **Go To Adapter**.

  b. Click the **Adapter Management** tab.

  c. Select the **Override default Probe selection** check box, and enter the name of the Probe used for the NNMi integration in the **Probe** field.

  d. Click **Save** to save the adapter, then reactivate the job against the **IpAddress** CI of the NNMi server.

Limitation

When integrating with multiple NNMi servers, each node (host) may only be managed by one NNMi server at the same time.

Chapter 9: HP Service Anywhere Push Integration

This section includes:

Overview

Integration between HP Universal CMDB (UCMDB) and HP Service Anywhere ("**SAW**") enables you to share information from UCMDB with Service Anywhere. Common use cases include Hardware, Installed Software, Running Software, and Business Services.

You can use the integration to automate the creation and update of information in Service Anywhere. This ensures Service Anywhere is kept up to date with real, accurate, discovered data in your environment.

How Data is Synchronized Between UCMDB and Service Anywhere

The following graphic shows the high-level components of the integration:
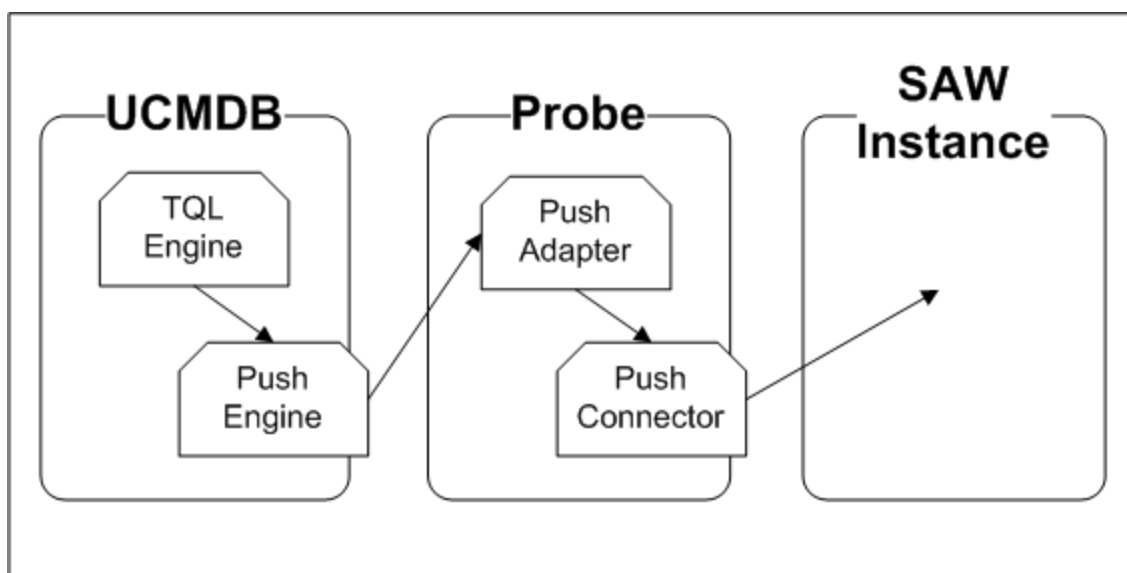
Note: The Push Adapter and Push Connector are executed in the Data Flow Probe/Integration Service process.

UCMDB stores its information using CIs. The integration chooses which data to pull from UCMDB by defining integration TQL queries. Each TQL query defines a superset of data relevant for the integration.

The **UCMDB Push Engine:**

- Retrieves the required data from UCMDB, using the given TQL query.

- If set for differential synchronization, filters the data to include only the data that has changed since the last execution of this synchronization.

- Splits the data into multiple chunks without breaking consistency.

- Sends the information to the Probe/Adapter

The **Push Adapter** is a generic framework for easily configuring push adapters, using only XML and Groovy[1]. It allows easy mapping of the data from the UCMDB data model into the Service Anywhere data model, and the transfer of this converted data into the Push Connector. For more information, see **Developing Push Adapters** in the *HP Universal CMDB Developer Reference Guide*.

The **Push Connector** is a component that connects to the Push Adapter, built specifically to reconcile, push, and handle the complex logic needed to synchronize data into Service Anywhere.

Supported Versions

This integration supports HP Service Anywhere version 1.00 and later.

How to Integrate UCMDB and Service Anywhere

To set up integration between UCMDB and Service Anywhere, you must complete the following steps:

1. "Set up UCMDB" below

2. "Push CI Data from UCMDB to Service Anywhere" on page 138

Set up UCMDB

To set up UCMDB, complete the following steps:

Create an Integration Point in UCMDB

1. Log on to UCMDB as an administrator.

2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.

3. Click **New Integration Point** [*]. The New Integration Point dialog box is displayed.

4. Complete the Integration and Adapter Properties fields as shown in the following table:

---

[1]Groovy is an agile and dynamic language, natively supported by the Java Virtual Machine. It allows simple scripting capabilities, while maintaining all the strengths and capabilities of Java. It can execute simple String manipulation, and use 3rd party libraries. For more information, see http://groovy.codehaus.org/

Field	Required	Description
Integration Name	Yes	Type the name (unique key) of the integration point.
Integration Description	No	Type a description of the current integration point.
Adapter	Yes	Select **HP Software Products > Service Anywhere >Service Anywhere Push Adapter**
Is Integration Activated	Yes	Select this option to indicate the integration point is active.
Hostname/IP	Yes	Type the external hostname or IP Address of the Service Anywhere instance.
Port	Yes	The external communication port of the Service Anywhere instance. **Note:** The Adapter automatically populates this field with the default SSL port 443.
Web application Root context	Yes	The Root context of Service Anywhere. **Note:** The Adapter automatically populates this field.
Protocol	Yes	The protocol used to connect with Service Anywhere. The only valid protocols are HTTPS and HTTP. **Note:** ■ The Adapter automatically populates this field with the HTTPS protocol. ■ For SSL connection to a specific Service Anywhere instance, you must first import the certificate from the Service Anywhere instance to the probe keystore. The keystore path is: `\\<UCMDBHOST>\c:\hp\UCMDB\DataFlowProbe\ bin\jre\lib\security\cacerts`

Field	Required	Description
Credentials ID	Yes	Click ⸬ and select Generic protocol. Do **one** of the following: ▪ Select the appropriate credential ▪ Create a new credential. Fore more information, see "How to Create a New Credential" on the next page
Data Flow Probe	Yes	Select the name of the Data Flow Probe/Integration service used to execute the synchronization from. **Note:** Do not use Auto-Select for this field.
Additional Probes	No	Select additional probes to use when pushing to Service Anywhere in order to increase redundancy.

5. Click **Test Connection** to make sure there is a valid connection.

6. Click **OK**.

   The integration point is created and its detailed are displayed.

   UCMDB creates a default data push job when creating the integration point. If needed you may create or edit the existing job. For more information, see "Work with Data Push Jobs" in the *HP Universal CMDB Data Flow Management Guide*.

How to Create a New Credential

If you need to create a new credential, you must complete the following steps:

1. Select Generic Protocol

2. Click **Create new connection details for selected protocol type** ✳.

3. Populate the fields on the Generic Protocol Parameters page, as follows:

Field	Value
Network Scope	Use the default value.
User Label	Type a label for the credential.
User Name	The name of the user used to connect to Service Anywhere. **Note:** The user must have integration level permission in Service Anywhere.
Password	The password of the user needed to connect to Service Anywhere.

Push CI Data from UCMDB to Service Anywhere

Data push jobs copy CI or CI relationship records from your UCMDB system to your Service Anywhere system. To run a data push job, complete the following steps:

1. Log on to UCMDB as an administrator.

2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.

3. Select the integration point you created for Service Anywhere.

4. Select the default **SAW Push** job, or any other job you have created for this integration.

   UCMDB Creates a default data push job when creating an integration point.The following table lists the Topology Query Language (TQL) queries in the default data push job. If required, you may create, update, or remove TQL queries for the push job. You may also need to update the mapping. See "Customize an Existing Mapping" on page 147.

   To access the out-of-the-box TQL queries for push, go to **Modeling > Modeling Studio > Resources**, select **Queries** for Resource Type, and then go to **Root > Integration > Service Anywhere Push**.

> **Note:** The order in which the TQL queries are resolved is important.

TQL Query	Description
SAW_BusinessElement	Pushes Business Activity, Business Application, Business Function, Business Process, Business Service, Business Infrastructure, Business Transaction, CI Collection, Functional Groups, and Organization CIs. **Note:** Persons are not synchronized in this integration. You must import Persons into Service Anywhere using LDAP. **Mapping XMLs:** ■ businessElementCisMappings.xml ■ businessElementrelationsMappings.xml
SAW_Computer	Pushes nodes (Computers, Network Devices, etc.) and Location CIs. **Mapping XMLs:** ■ computerCisMappings.xml ■ computerRelationsMappings.xml
SAW_ComputerElement	Pushes Buffer, File System, File System Export, Installed Software, and Logical Volume CIs. **Mapping XMLs:** ■ computerCisMappings.xml ■ computerRelationsMappings.xml

TQL Query	Description
SAW_ComputerPhysicalElements	Pushes CPU, Disk Device, Environmental Sensor, Fan, Hardware Board, Memory Unit, Physical Ports, and Power Supply CIs. **Mapping XMLs:** - computerPhysicalEelementCisMappings.xml - computerPhysicalElementRelationsMappings.xml
SAW_ComputerRelations	Pushes only relationships. **Note:** The relationships that are pushed depend on the CIs pushed in the preceding TQL queries. **Mapping XMLs:** - computerRelationsMappings.xml
SAW_NetworkEntity	Pushes Interface, IpAddress, IpSubnet, Layer2 Connection, and VLAN CIs. **Mapping XMLs:** - computerCisMappings.xml - computerRelationsMappings.xml
SAW_RunningSoftware	Pushes Application Server, Cluster Resource Group, Cluster Software, Failover Cluster, IpService Endpoint, Running Software,and URI Endpoint CIs. **Mapping XMLs:** - runningSoftwareCisMappings.xml - runningSoftwareRelationsMappings.xml

5. Run the job manually to see if the integration job works properly:

   a. To push all the relevant data for the job, click **Full Synchronization** .

   b. To push only the changes in the data since the job last executed, click **Delta Synchronization** .

6. Wait for the job to complete; click **Refresh** multiple times as needed until the job is completed.

7. When the job is completed, the job status becomes one of the following depending on the results:

    ▪ Succeeded

    ▪ Completed

    ▪ Failed

8. Click the **Statistics** tab to view the results; if any errors occur, click the **Query Status** tab and **Job Errors** tab for more information.

    > **Note:** For details about these tabs and managing the integration, see **Integration Jobs Pane** in the *HP Universal CMDB Data Flow Management Guide*.

    If the job completes successfully, you can view the UCMDB CI data in Service Anywhere.

Schedule Data Push Jobs

To schedule job executions directly from a data push job:

1. Log on to UCMDB as an administrator.

2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.

3. Select the integration point you created for the UCMDB - Service Anywhere.

4. Select the push job.

5. Click **Edit Integration Job** .

    > **Note:** UCMDB allows you to define two different schedules for two types of data push: **Changes Synchronization** and **All Data Synchronization**. It is recommended to use the Changes Synchronization schedule to only synchronize changes and avoid synchronizing the entire set of data each time.

6. Define a schedule for Changes Sync.

    a. Click on the **Changes Synchronization** tab.

    b. Select the **Scheduler enabled** option.

      c.   Select the scheduling options you want to use.

7. Click the **All Data Synchronization** tab and select the scheduling options you want to use.

8. Click **OK**.

9. Save the integration point.

> **Note:** The out-of-the-box default job schedule is for a Changes Synchronization data push every 2 hours.

Tailor the Integration

This section includes:

Integration Architecture

This section includes:

- "Data Flow Architecture" below

- "Integration TQL Queries" on the next page

- "Service Anywhere Rules and Flows" on the next page

- "Data Mapping" on the next page

- "Push Mapping" on the next page

Data Flow Architecture

1. The Push Engine executes the TQL query.

2. For a differential flow, the data is compared to the last synchronized data, and only the changes are forwarded.

3. Data is then pushed to the Push Adapter.

4. The Push Adapter loads the correct mapping for the specific TQL query.

5. Data is mapped from the UCMDB data Model into the Service Anywhere Data model according to the mapping XML.

6. Data is sent to the Push Connector.

7. The Push Connector orders all the data in a set of dependency trees, starting with the records that do not depend on any other record.

8. The Push Connector starts reconciling and pushing any record without any dependencies, or a record whose dependencies have already been reconciled/pushed to Service Anywhere.

a. The Push Connector first tries to reconcile with existing records

b. If it finds a match, it attempts to update that record,

c. If it does not find a match, it attempts to create a new record.

Integration TQL Queries

Any attribute using in the mapping flow of the Push Adapter should be marked in the selected layout of the query node.

For more information, see **Data Flow Management > Integration Studio > Integration Jobs Pane**.

Service Anywhere Rules and Flows

For details of Service Anywhere rules and flows, refer to the Service Anywhere documentation.

Data Mapping

For details, see Developing Push Adapters in the *HP Universal CMDB Developer Reference Guide*.

Push Mapping

The following table describes variables in the Generic Push Adapter XML mapping file that are specifically required for integration with Service Anywhere.

Variable	Description
externalId	The external ID of the CI in UCMDB.
serviceName	The Web service name used in the SOAP request.
attributesHierarchy	The location of the CI attributes in the SOAP request.
keysHierarchy	The location of the CI keys in the SOAP request.
keyTag	The key in the SOAP request that passes the value of the ID.
keyVal	The ID value from UCMDB.

Variable	Description
retrieveRequestName	The name of the SOAP request for a retrieval operation.
createRequestName	The name of the SOAP request for a creation operation.
deleteRequestName	The name of the SOAP request for a delete operation.
updateRequestName	The name of the SOAP request for an update operation.

Change Adapter Settings

To change adapter settings:

1. Go to **Data Flow Management > Adapter Management > SAWPushAdapter > Adapters**.

2. Right-click **SAWPushAdapter** and click **Edit Adapter Source**.

3. Scroll down to the **<adapter-settings>** tag.

4. Edit the settings as required and click the Save button.

   The following table shows the Settings relevant to the SAW Push Adapter:

Setting	Default	Description
replication.chunk.size	4,000	Defines the requested number of CIs and Relationships sent in each chunk. It is possible to adjust this setting to try and improve performance of the server and the Probe.
is.new.generic.push.adapter	true	Enables use of the correct generic Push Adapter XSD.
PushConnector.class.name	com.hp.ucmdb.adapters. sawpush.SAWPushAdapter	The name of the Java class used to load the Push Connector. **Note:** Only change this setting if you are not using the out-of-the-box connector.
support.ci.type.mapping	true	Specifies the mapping of single CIs to single CIs.
parallel.thread.pool.size	5	The amount of threads used in Parallel Push Mode. The more threads, the more CPU used. Increasing the pool size may lower performance.

Customize an Existing Mapping

This example shows you how to add a new attribute to the integration, including the TQL query, and Push Adapter Mapping. It enables the integration to push the attribute to Service Anywhere.

After completing the following steps, you may run the job with the customized mapping:

1. **Add the new attribute to an existing Service Anywhere TQL query layout.**

   In this step you add the attribute of a specific CI to the integration TQL query, so that you can use the attribute and value in the mapping.

   a. Go to **Modeling > Modeling Studio >Resources** and select the **Queries** Resource Type.

   b. Go to **Queries > Root > Integration > Service Anywhere Push**.

   c. Select the TQL query containing the CI to which you want to add a new attribute, and double-click. The query is displayed in the main pane.

   > For example, if you are adding a new attribute **bt_example** to the Business Transaction CI, double-click the SAW_BusinessElement query.

   d. Select the specific CI for which you want to add an attribute, right-click and select **Query Node Properties**. The **Query Node Properties** dialog box is displayed.

   e. Go to the **Element Layout** tab.

   f. Move the new attribute to the **Specific Attributes** box.

   g. Click **OK**.

   h. Save the query.

2. **Add the new attribute to the corresponding push adapter mapping.**

   In this step, you take the value from the TQL result, and remodel it to the Service Anywhere data model.

   a. Go to **Data Flow Management > Adapter Management > Packages > SAWPushAdapter > Configuration Files**, and select the XML file corresponding to the CI to which you have added a new attribute.

   > For example, if you are adding a new attribute **bt_example** to the **Business Transaction** CI, select the **businessElementCisMappings.xml** file.

   b. Go to the source_ci_type XML tag for the element that matches the name of the element in

the TQL you edited.

> Continuing the previous example, you go to:
>
> ```
> <source_ci_type query-element-name="BusinessTransaction*"
>  extends="AbstractBusinessNode">
> ```

c. Add the target mapping to hold the value of the new attribute.

> Continuing the previous example:
>
> ```
> <target_mapping name="bt_example" datatype="STRING"
>  value ="BusinessTransaction.attributeExists('bt_example') ? BusinessT
> ransaction['bt_example']: 'MISSING'"/>
> ```

d. Click **OK**.

Add a New Mapping to the Integration

This example shows how to add a new TQL query and push-mapping to the integration for a CIT called *Example*. It consists of the following steps:

Step 1: Create a TQL Query

1. Go to **Modeling > Modeling Studio > New > Query**.

2. From the **CI Types** tab, add **Example CIT** to the query.

3. Right-click the **Example Query Node** and select **Query Node Properties**.

4. Go to the **Element Layout** tab.

5. Select **Select attributes for layout**.

6. In the **Attributes condition** drop down, select **Specific Attributes**, and add the new attributes that you want to map and push to Service Anywhere.

7. Click **OK**.

8. Save the query to **Modeling > Modeling Studio > Resources > Queries > Root > Integration > Service Anywhere Push**.

Step 2: Create a Push-Mapping

1. Go to **Data Flow Management > Adapter Management > SAWPushAdapter**.

2. Click **Create new resource** ✶ and select **New Configuration File**.

3. Type the a name for the file.

4. Select the **SAWPushAdapter** package.

5. Click **OK**.

6. Copy the following into the newly created XML file, replacing **ExampleCI** with the name of the relevant CI:

```
<integration
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../generic-push-adapter.xsd">
<info>
 <source name="UCMDB" versions="10.0" vendor="HP"/>
 <target name="ServiceAnywhere" versions="1.0" vendor="HP"/>
</info>
<import>
 <scriptFile path="mappings.scripts.SAWPushFunctions"/>
</import>
<targetcis>
 <source_ci_type query-element-name="ExampleCI*">
  <target_ci_type name="ExampleCI">
    <variable name="externalId" datatype="STRING"
     value="ExampleCI['external_id_obj']"/>
    <variable name="serviceName" datatype="STRING"
     value="'CMDB_UDM_1_6'"/>
    <variable name="attributesHierarchy" datatype="STRING"
     value="'model.instance'"/>
    <variable name="keysHierarchy" datatype="STRING"
     value="'model'"/>
    <variable name="keyTag" datatype="STRING" value="'ID'"/>
    <variable name="keyVal" datatype="STRING"
     value="ExampleCI['cmdb_id']"/>
    <variable name="retrieveRequestName" datatype="STRING"
     value="'RetrieveExampleCI'"/>
    <variable name="createRequestName" datatype="STRING"
     value="'CreateExampleCI'"/>
    <variable name="deleteRequestName" datatype="STRING"
     value="'DeleteExampleCI'"/>
    <variable name="updateRequestName" datatype="STRING"
     value="'UpdateExampleCI'"/>
    <target_mapping name="Name" datatype="STRING"
```

```
        value="ExampleCI.attributeExists('name') ?
        ((ExampleCI['name']==null)
        ?ExampleCI['display_label']:ExampleCI['name']) :
        'MISSING'"/>
      <target_mapping name="Description" datatype="STRING"
       value="ExampleCI.attributeExists('description')
       ? ExampleCI['description'] : 'MISSING'"/>
      <target_mapping name="DisplayLabel"  datatype="STRING"
       value="ExampleCI.attributeExists('display_label')
       ? ExampleCI['display_label'] : 'MISSING'"/>
      <target_mapping name="GlobalId"  datatype="STRING"
       value="ExampleCI.attributeExists('global_id')
       ? ExampleCI['global_id'] : 'MISSING'"/>
    </target_ci_type>
  </source_ci_type>
 </targetcis>
</integration>
```

7. Click **OK**.

Step 3: Create a Job with the New TQL Query

1. Go to **Data Flow Management > Integration Studio**.

2. Create an Integration Point with Service Anywhere.

3. In the **Integration Jobs** tab, click **New Integration Job** ✳ .

4. Insert a job name in the **Name** field.

5. Click **Add Query** ✚ , and choose the newly created query.

6. Click **OK**.

Step 4: Run the Job

1. Click on the job created in "Step 3: Create a Job with the New TQL Query".

2. Click **Full Synchronization** 🔁 .

3. Wait for the job to complete; click **Refresh** 🔄 multiple times as needed until the job is completed.

4. Make sure that the status is **Succeeded**.

Step 5: View the Results

1. Go to Service Anywhere.

2. Validate that the new attribute, pushed in the previous steps, is displayed.

Troubleshooting and Limitations

The Service Anywhere push adapter assumes that relationships comply with the Universal Data Model specification. The adapter pushes CIs that comply with the specification, and ignores those that do not comply.

Chapter 10: HP ServiceCenter/Service Manager Integration

This chapter includes:

Note: This adapter is a specific configuration of the ServiceDesk Adapter.

Overview

The ServiceCenter/Service Manager adapters support the push to and retrieval of data from HP ServiceCenter and HP Service Manager. These adapters connect to, send data to, and receive data from ServiceCenter/Service Manager using the Web Service API. Every request to ServiceCenter/Service Manager to calculate a federated query or to push data is made through these adapters. These adapters are compatible with HP ServiceCenter version 6.2, and HP Service Manager, versions 7.0x, 7.1x, and 7.2x-9.2x (following changes to the WSDL configuration).

The adapters are provided with preconfigured jobs to transfer Incident, Problem, and Planned Change CI types between ServiceCenter/Service Manager and UCMDB.

Data Push

The data push framework uses the adapter to push CIs and relationships to HP Service Manager. Once a CI has been pushed to HP Service Manager, an Actual State flow may be triggered in HP Service Manager, and selecting a tab in HP Service Manager enables you to view the most updated data available on the CI in UCMDB.

For details about setting up a data push flow, see "Data Push Tab" in the *HP Universal CMDB Data Flow Management Guide*.

Federation

The adapter supports three external CI types: Incident, Problem, and Planned Change. The adapter retrieves the CIs of these types from ServiceCenter/Service Manager with the required layout and by a given filter (using reconciliation and/or a CI filter). Each of these CITs can be related to one of the following UCMDB internal CITs: Host, Business Service, Application. Each UCMDB internal CIT includes a reconciliation rule in the ServiceCenter/Service Manager configuration that can be changed dynamically. For details, see "Reconciliation Data Configuration" on page 160. Note that there are no internal relationships between adapter-supported CITs.

The modeling of the supported CITs and virtual relationships is supplied with the Adapter. You can add attributes to a CIT. For details, see "How to Add an Attribute to the ServiceCenter/Service Manager CIT" on page 169.

For details about setting up a federation flow, see "Federation Tab" in the *HP Universal CMDB Data Flow Management Guide*.

Supported Versions

UCMDB is delivered with four different Service Manager adapters, for different versions of HP ServiceCenter/HP Service Manager. When you define an integration, choose the correct adapter according to your Service Manager version.

For documentation about **ServiceManagerAdapter9.x**, see the *HP Service Manager 9.3x Integration Guide*.

Data Push Flow

You can configure the data push flow options for the Service Manager integration by updating the following UCMDB, Service Manager and adapter XML files:

- **xslt files.** Maps the UCMDB graph to the Service Manager request.

- **smSyncConfFile.** Maps a tql name to an xslt file. This resource should be changed when adding a new TQL query.

Multi-Threading

By default, the ServiceDesk Adapter uses six concurrent threads to push data to Service Manager. To configure the ServiceDesk Adapter multi-thread settings, edit the **sm.properties** file, located in:

Data Flow Management > Adapter Management > ServiceManagerAdapter corresponding to Service Manager version > Configuration Files

Error Handling

The ServiceCenter/Service Manager adapter has a mechanism that permits the capture of CIs that failed in a push job due to specific errors, and instead of failing the entire push job, attempts to send them again in future executions. In such a case, the statistics display the **Successful with warnings** status.

By default, only the error of `locked CI` (Error 3) triggers this mechanism.

To configure error handling, navigate to **Adapter Management > ServiceManagerAdapterX-X > Configuration Files > sm.properties** and set the required values.
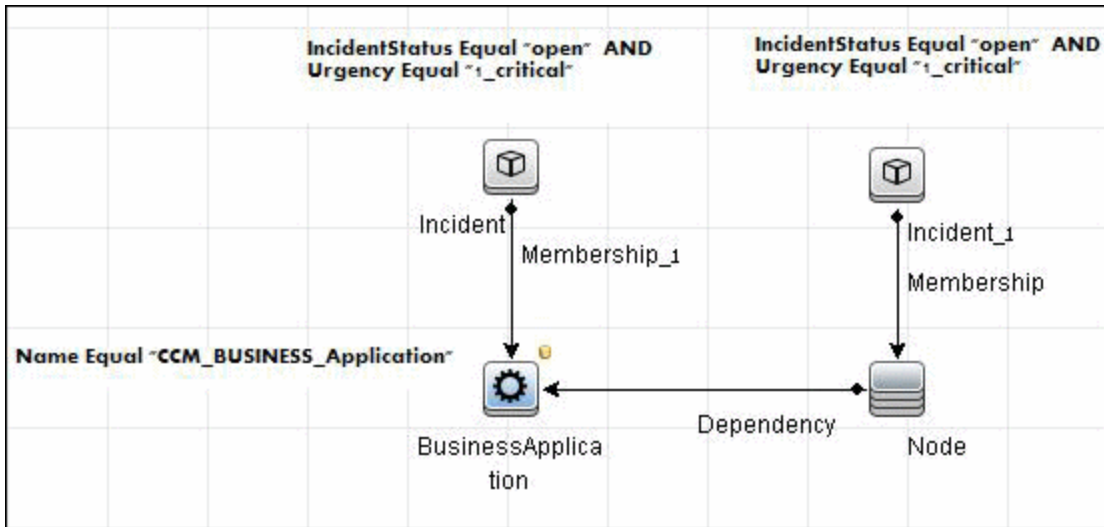
Federation Use Cases

The following use cases (which include TQL query examples) describe how the adapter can be used:

- A user needs to display all unplanned changes to all hosts running a specific application during the last 24 hours:
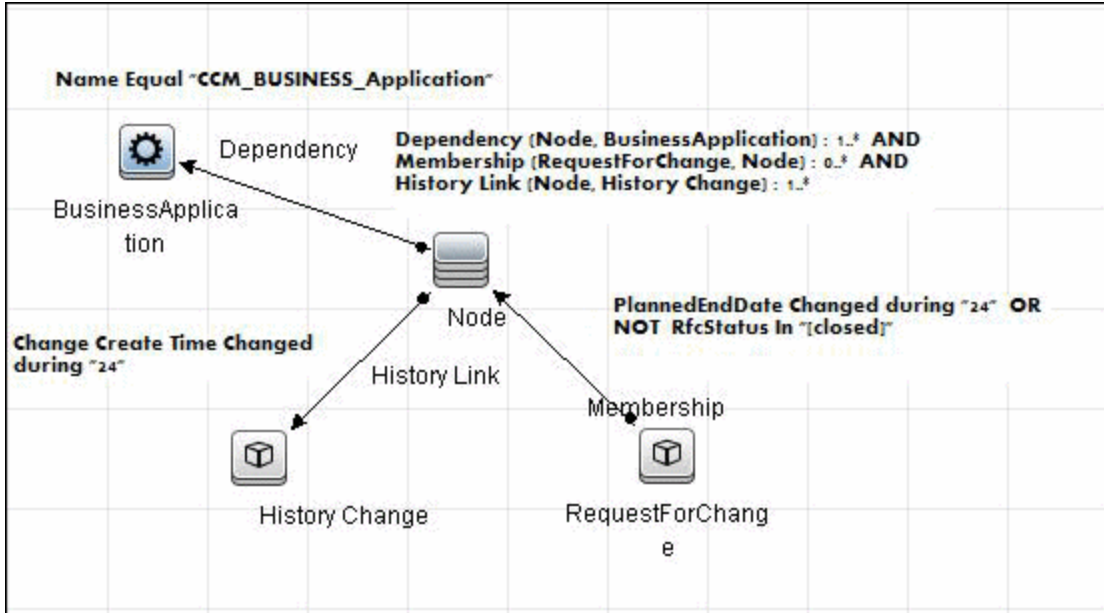
- A user needs to see all open critical incidents on an application and its hosts:

Viewing the Actual State

UCMDB exposes a Web Service for the use of Service Manager. The Web Service receives the CMDB ID and customer ID as input and returns extended data for the CI, which includes properties and related CIs.

The call to the Web Service is done in the Actual State tab in HP Service Manager, when Service Manager is configured to work with UCMDB.

The Web Service executes the query in the **Integration\SM Query** folder that matches the type of CI sent. If more than one matching query exists, an exception is thrown.

The layout that is defined in the TQL query is the layout that is synchronized.

It is common for some parts of the executed query to be federated (for example, from DDMI, Asset Manager, SMS, and so on).

This section includes:

- "Predefined Queries" below

- "Configuration" below

Predefined Queries

Out-of-the-box queries are located in the **Integration\SM Query** folder. Queries are selected according to the class type of the CI.

- **hostExtendedData.** Used for retrieving real time extended information (Asset, Person, WindowsService, Printer, InstalledSoftware, and CPU) about a certain CI of type Node.

- **applicationExtendedData.** Used for retrieving real time extended information about Business Applications.

- **businessServiceExtendedData.** Used for retrieving real time extended information about Business Services.

Configuration

WSDL and XML Schema URLs for the Web Service

- **WSDL:**

  ```
  http://[machine_name]:8080/axis2/
  services/ucmdbSMService
  ```

- **XML Schema:**

  ```
  http://[machine_name]:8080/axis2/
  services/ucmdbSMService?xsd=xsd0
  ```

Manipulating the Result Using Transformations

In some cases you may want to apply additional transformations to the resulting XML (for example, to sum up all the disks' sizes and add those as an additional attribute to the CI). To add invoke additional transformation on the TQL results, place a resource named **[tql_name].xslt** in the adapter configuration as follows: <SF> generic?**Adapter Management > ServiceDeskAdapter7-1 > Configuration Files > [tql_name].xslt**.

There is a resource named **example_calculated_attribute.xslt** that demonstrates how to sum the disk sizes using xslt.

Using Global IDs

It is possible to use the Global ID instead of the CMDB ID to work with the Actual State flow. This may be needed in multiple CMDB environments, where a non-CMS UCMDB is integrated with Service Manager. To use global IDs instead of CMDB IDs, navigate to **Adapter Management > ServiceManagerAdapterX-X > Configuration Files > sm.properties** and set **use.global.id=true**.

For details about multiple CMDB environments, see "Integrating Multiple CMDBs" in the *HP Universal CMDB Data Flow Management Guide*.

If CIs were previously pushed to Service Manager from a different CMDB instance, duplicates may occur, as the CIs will not reconcile.

Compressing Location Topology to an Attribute

Due to the limitation of the Data Push flow, it is not possible to push topologies that have CIs that are not connected directly to the Root. To be able to push locations to Service Manager, an enrichment is used to concatenate the location topology to a single attribute (Calculated Location) on the Node.

The enrichments are found in the **Location** folder:

- Location_1Enrhicment

- Location_2Enrichment

- Location_3Enrichment

The xslt transformer then inflates the attribute back to separate XML tags with the following xslt code:

```
<xsl:variable name="calculatedLocation" select="@calculated_location"/>
    <Building>
        <xsl:value-of select="substring-after($calculatedLocation,'
         Building:')"/>
    </Building>
    <Floor>
        <xsl:value-of select="substring-before(substring-after
         ($calculatedLocation,'Floor:'),' Building:')"/>
    </Floor>
    <Room>
        <xsl:value-of select="substring-before(substring-after
```

```
        ($calculatedLocation,'Room:'),' Floor:')"/>
    </Room>
```

The serviceDeskConfiguration.xml File

The **serviceDeskConfiguration.xml** Adapter configuration file contains three parts:

The first part, which is defined by the **ucmdbClassConfigurations** element, contains the external CIT configuration that the Adapter supports. For details, see "External CITs Configuration" below.

The second part, defined by the **reconciliationClassConfigurations** element, contains reconciliation data information for appropriate UCMDB CITs. For details, see "Reconciliation Data Configuration" on page 160.

The third part, defined by the **globalConnectorConfig** element, includes the global configuration for a specific connector implementation. For details, see "Global Configuration" on page 163.

This section includes the following topics:

- "External CITs Configuration" below

- "Adding Attributes to a CIT" on page 160

- "Reconciliation Data Configuration" on page 160

- "Changing the Reconciliation Rule of a CIT" on page 163

- "Reconciliation of a Host by IP Address or Name" on page 163

- "Global Configuration" on page 163

External CITs Configuration

Each CIT that is supported by the adapter is defined in the first section of the adapter configuration file.

This section, **ucmdbClassConfiguration**, represents the only supported CIT configuration. This element contains the CIT name as defined in the UCMDB class model (the **ucmdbClassName** attribute), mapping for all its attributes (the **attributeMappings** element), and a private configuration for a specific connector implementation (the **classConnectorConfiguration** element):

- The **ucmdbClassName** attribute defines the UCMDB class model name.

- The **attributeMappings** element contains attributeMapping elements.

  The **attributeMapping** element defines the mapping between the UCMDB model attribute name (the **ucmdbAttributeName** attribute) to an appropriate ServiceCenter/Service Manager attribute name (the **serviceDeskAttributeName** attribute).

  For example:

```
<attributeMapping ucmdbAttributeName="problem_brief_description"
serviceDeskAttributeName="brief.description"/>
```

This element can optionally contain the following converter attributes:

- The `converterClassName` attribute. This is the converter class name that converts the UCMDB attribute value to the ServiceDesk attribute value.

- The `reversedConverterClassName` attribute. This is the converter class name that converts the ServiceDesk attribute value to the UCMDB attribute value.

- The `classConnectorConfiguration` element contains the configuration for the specific connector implementation for the current external CIT. Wrap this configuration in CDATA if it contains special XML characters (for example, `&` replacing &).

  The useful fields of the Service Manager `classConnectorConfiguration` element are as follows:

  - The **device_key_property_names** element contains the fields names in the WSDL information of the current object that can contain the device ID (for example, `ConfigurationItem`). Each field should be added as a `device_key_property_name` element.

  - The **id_property_name** element contains the field name in the WSDL information that contains the ID of the current object.

The following example shows the **ucmdbClassConfiguration** section of the **serviceDeskConfiguration.xml** file. The section includes the **ucmdbClassName** element for the Incident CIT with a ServiceCenter connector implementation:

```
<ucmdbClassConfiguration ucmdbClassName="it_incident">
        <attributeMappings>
            <attributeMapping ucmdbAttributeName="incident_id" serviceDeskAt
tributeName="IncidentID"/>
            <attributeMapping ucmdbAttributeName="incident_brief_descriptio
n" serviceDeskAttributeName="BriefDescription"/>
            <attributeMapping ucmdbAttributeName="incident_category" service
DeskAttributeName="Category"/>
            <attributeMapping ucmdbAttributeName="incident_severity" service
DeskAttributeName="severity"/>
            <attributeMapping ucmdbAttributeName="incident_open_time" servic
eDeskAttributeName="OpenTime"/>
            <attributeMapping ucmdbAttributeName="incident_update_time" serv
iceDeskAttributeName="UpdatedTime"/>
            <attributeMapping ucmdbAttributeName="incident_close_time" servi
ceDeskAttributeName="ClosedTime"/>
            <attributeMapping ucmdbAttributeName="incident_status" serviceDe
skAttributeName="IMTicketStatus"/>
        </attributeMappings>
 <classConnectorConfiguration>
```

```
            <![CDATA[ <class_configuration connector_class_name="com.mercury
.topaz.fcmdb.adapters.serviceDeskAdapter
.serviceCenterConnector.impl.SimpleServiceCenterObjectConnector">
        <device_key_property_names>
          <device_key_property_name>ConfigurationItem</device_key_property_nam
e>
        </device_key_property_names>
        <id_property_name>IncidentID</id_property_name>
        <keys_action_info>
          <request_name>RetrieveUcmdbIncidentKeysListRequest</request_name>
<response_name>RetrieveUcmdbIncidentKeysListResponse</response_name>
        </keys_action_info>
        <properties_action_info>
          <request_name>RetrieveUcmdbIncidentListRequest</request_name>
          <response_name>RetrieveUcmdbIncidentListResponse</response_name>
        </properties_action_info>
      </class_configuration> ]]>
        </classConnectorConfiguration>
      </ucmdbClassConfiguration>
```

Adding Attributes to a CIT

To add an attribute to the UCMDB model for an adapter-supported CIT:

1. Go to **Data Flow Management > Adapter Management** and select the **ServiceManagerAdapter** that corresponds to your version of Service Manager.

2. Select **Configuration Files > ServiceDeskConfiguration.xml** file and add an `attributeMapping` element to the appropriate `ucmdbClassConfiguration` element.

3. Verify that ServiceCenter/Service Manager externalizes this attribute in its Web Service API.

4. Click **Save**.

Reconciliation Data Configuration

Each UCMDB CIT that can be related to the adapter-supported CIT is defined in the second section of the **serviceDeskConfiguration.xml** file.

This section, `reconciliationClassConfigurations`, represents the reconciliation data configuration for one UCMDB CIT. The element includes the following attributes:

- **ucmdbClassName.** This is the CIT name as defined in the UCMDB class model.

- **concreteMappingImplementationClass.** This is the class name of the concrete implementation for the `ConcreteMappingEngine` interface. Use this attribute to map between instances of UCMDB CITs and external Adapter CITs. The default implementation that is used is:

  ```
  com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.mapping.impl. OneNodeMappi
  ngEngine
  ```

An additional implementation exists that is used only for the host reconciliation CIT for reconciliation by the IP of the host:

```
com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter
.mapping.impl. HostIpMappingEngine
```

The **reconciliationClassConfiguration** element can contain one of the following elements:

- The **reconciliationById** element. This element is used when the reconciliation is done by ID. In this case, the text value of this element is the ServiceDesk field name that contains the CMDB ID. For example:

```
<reconciliationById>UcmdbID</reconciliationById>
```

In this example, the ServiceDesk field `UcmdbID` contains the CMDB ID of the appropriate host.

- The **reconciliationData** element. This element is used if the reconciliation is done by comparing attributes. You can run reconciliation with one attribute or several attributes by using the logical operators OR and/or AND.

If you run reconciliation with one attribute, the **reconciliationData** child element should be a **reconciliationAttribute** element. The **reconciliationAttribute** element contains an appropriate UCMDB attribute name (the **ucmdbAttributeName** attribute) and an appropriate ServiceDesk attribute name (the **serviceDeskAttributeName** attribute). This element can also contain a **ucmdbClassName** attribute that defines the appropriate UCMDB CIT name. By default, the current reconciliation UCMDB CIT name is used.

You can also use the **converterClassName** and **reversedConverterClassName** attributes; they should contain the converter class name that converts the UCMDB attribute value to the ServiceDesk attribute value, or vice versa.

For example:

```
<reconciliationData>
 <reconciliationAttribute ucmdbAttributeName="name" serviceDeskAttributeName="
NetworkName" converterClassName="com.mercury.topaz.fcmdb.adapters.
serviceDeskAdapter.converter.PropertyValueConverterToUpperCase" />
 </reconciliationData>
```

For reconciliation to run with two or more attributes, use a logical operator between reconciliation attributes.

The logical operator AND can contain several **reconciliationAttribute** elements (the minimum is 2). In this case the reconciliation rule contains an AND operator between attribute comparisons.

For example:

```
<reconciliationData>
<AND>
 <reconciliationAttribute ucmdbAttributeName="name"
```

```
serviceDeskAttributeName="NetworkName" converterClassName="com.mercury.topaz.f
cmdb.adapters.
serviceDeskAdapter.converter.PropertyValueConverterToUpperCase" />
<reconciliationAttribute ucmdbClassName="ip_address" ucmdbAttributeName="name"
serviceDeskAttributeName="NetworkAddress" />
</AND>
</reconciliationData>
```

In this example, the reconciliation rule follows this format: **node.name= NetworkName** and **ip_
address.name= NetworkAddress**.

The logical operator OR can contain several **reconciliationAttribute** and AND elements. In
this case, the reconciliation rule contains an OR operator between attributes and AND
expressions. Since XML does not assure the order of elements, you should provide a priority
attribute to each sub-element of OR element type. The comparison between OR expressions is
calculated by these priorities.

For example:

```
<reconciliationData>
<OR>
                    <reconciliationAttribute ucmdbAttributeName="primary_dns_nam
e" serviceDeskAttributeName="NetworkDNSName" priority="2" />
<AND priority="1" >
                <reconciliationAttribute ucmdbAttributeName="name" serviceDesk
AttributeName="NetworkName" converterClassName="com.mercury.topaz.fcmdb.adapte
rs.
serviceDeskAdapter.converter.PropertyValueConverterToUpperCase"/>
                <reconciliationAttribute ucmdbClassName="ip_address" ucmdbAtt
ributeName="name" serviceDeskAttributeName="NetworkAddress" />
</AND>
</OR
    </reconciliationData>
```

In this example the reconciliation rule follows this format: (**node.primary_dns_name=
NetworkDNSName OR (node.name= NetworkName and ip_address.name=
NetworkAddress)**). Since the AND element takes a priority attribute of value 1, the
(node.name= NetworkName and ip_address.name= NetworkAddress) condition is checked
first. If the condition is satisfied, the reconciliation is run. If not, the **.host_dnsname=
NetworkDNSName** condition is checked.

The additional sub-element of the **reconciliationClassConfiguration** element is
classConnectorConfiguration. The **classConnectorConfiguration** element contains the
configuration for a specific connector implementation for the current reconciliation CIT. This
configuration should be wrapped by CDATA if it contains some special XML characters (for
example, & replacing &).

Changing the Reconciliation Rule of a CIT

1. In **serviceDeskConfiguration.xml**, update the appropriate `reconciliationData` element with the new rule.

2. Call to the JMX to reload the adapter: **FCmdb Config Services > loadOrReloadCodeBaseForAdapterId**, using the appropriate customer ID and `ServiceDeskAdapter` adapter ID, or go to the Integration Points pane and reload the adapter from there. For details, see "Integration Point Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Reconciliation of a Host by IP Address or Name

To run reconciliation on a host by **ip_address** or **name**, place the following `ReconciliationData` element in the Adapter configuration file:

```
                    <reconciliationData>
                         <OR>
                              <reconciliationAttribute priority="1" ucmdbClass
Name="ip_address" ucmdbAttributeName="ip_address" serviceDeskAttributeName="Netw
orkAddress" />
                              <reconciliationAttribute priority="2" ucmdbClass
Name="node" ucmdbAttributeName="name" serviceDeskAttributeName="NetworkName" con
verterClassName="com.mercury.topaz.fcmdb.adapters
.serviceDeskAdapter.converter.PropertyValueConverterToUpperCase" />
                         </OR>
                    </reconciliationData>
```

Global Configuration

The third section of the Adapter configuration file contains the global configuration for the specific connector implementation. This configuration, `globalConnectorConfig`, should be wrapped by CDATA if it contains some special XML characters (for example, `&` replacing `&`).

The useful fields of the Service Manager **globalConnectorConfig** element are as follows:

1. The **date_pattern** element contains the date adapter with which the Service Manager works.

   The default is `MM/dd/yy HH:mm:ss`.

   If the date adapter is wrong, an FTQL returns wrong date condition results.

2. The **time_zone** element defines the time zone of Service Manager. The default is the `UCMDB` `server time zone`.

   To check the Service Manager date adapter and time zone:

   a. **Service Manager version 7**: Access **Menu Navigation > System Administration > Base System Configuration > Miscellaneous >System Information Record**. Click the **Date Info** tab.

b. **ServiceCenter version 6.1**: Access **Menu Navigation > Utilities > Administration > Information >System Information**. Click the **Date Info** tab.

3. The **max_query_length** element defines the maximal query length in a Service Manager Web service request. The default value is `1000000`.

4. The **name_space_uri** element defines the name space URI to connect to the Service Manager Web service. The default value is `http://servicecenter.peregrine.com/PWS`.

5. The **web_service_suffix** element defines the Service Manager Web service center URI suffix. The default value is `sc62server/ws`. It is used when the URL is created.

How to Deploy the Adapter – Typical Deployment

This section describes a typical deployment of the adapter.

This task includes the following steps:

1. "How to Deploy the ServiceDesk Adapter" below.

2. "How to Add an Attribute to the ServiceCenter/Service Manager CIT" on page 169.

How to Deploy the ServiceDesk Adapter

This section explains where to place the files needed for deployment.

This task includes the following steps:

- "Add a ServiceCenter/Service Manager External Data Source" below

- "Configure HP ServiceCenter 6.2" on the next page

- "Configure HP Service Manager 7.0/7.1 " on page 168

- "Define data push jobs (optional)" on page 169

- "Run the jobs" on page 169

- "Select Classes for Federation" on page 169

1. **Add a ServiceCenter/Service Manager External Data Source**

   a. In UCMDB, navigate to **Data Flow Management > Integration Studio**.

   b. Click the **New Integration Point** button ✳ to add an integration point.

      ○ Click ⬚ , select the ServiceDesk Adapter that matches your version of Service Manager, and click **OK**.

Each out-of-the-box adapter comes predefined with the basic setup needed to perform integration with UCMDB. For information about changing these settings, see "Integration Studio Page" in the *HP Universal CMDB Data Flow Management Guide*.

○ Enter the following information, and click **OK**:

Name	Description
CMDB State (Data Push)	The state of the source machine. Values are: ○ Actual ○ Authorized **Note:** This field is visible only on a UCMDB for which authorized state has been defined.
Credentials	Allows you to set credentials for integration points. For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.
Hostname/IP	The name of the server on which HP Service Manager is running
Integration Name	The name you give to the integration point.
Is Integration Activated	Select this check box to create an active integration point. You clear the check box if you want to deactivate an integration, for instance, to set up an integration point without actually connecting to a remote machine.
Port	The server port at which HP Service Manager is connected.

c. Click **Test connection** to verify the connectivity, and click **OK**.

d. Click **Next** and verify that the following message is displayed: **A connection has been successfully created**. If it does not, check the integration point parameters and try again.

e. Continue with "Configure HP ServiceCenter 6.2" below or .

2. **Configure HP ServiceCenter 6.2**

If you are connecting to HP ServiceCenter 6.2, perform the following procedure. If you are connecting to HP Service Manager 7.0/7.1, skip this step.

a. Open HP ServiceCenter, then the ServiceCenter client.

b. Display **WSDL Configuration** in the Navigator (**Main Menu > Menu navigation > Toolkit**):
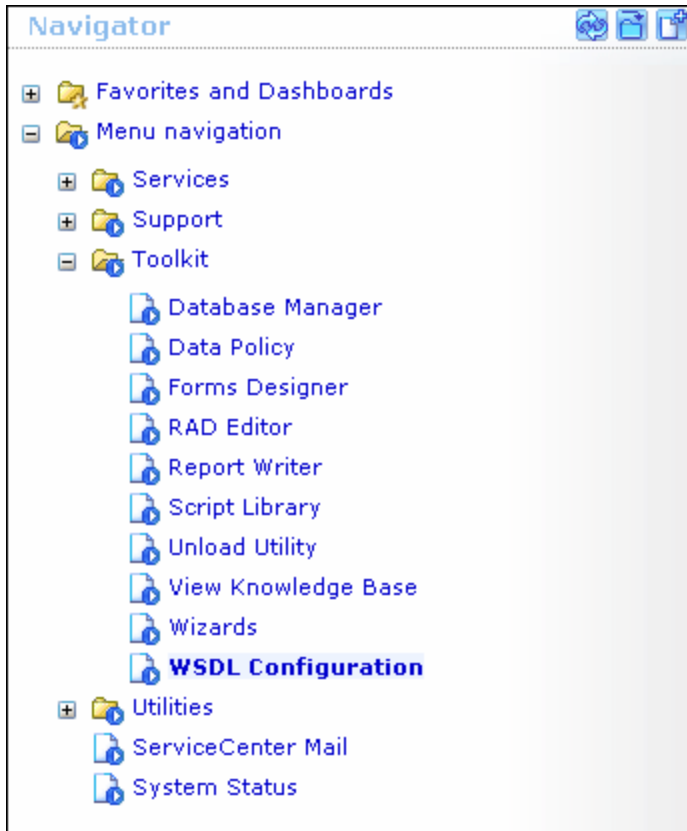
c. In the Name field, enter **device** and press **Enter**:
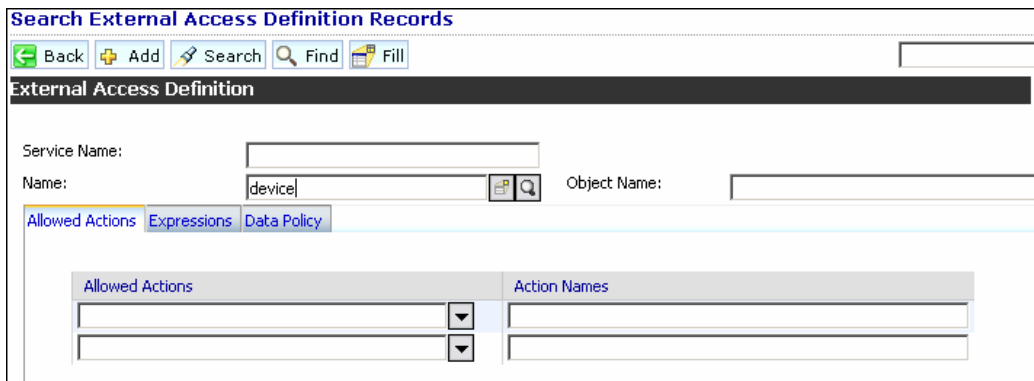
d. Select the **Data Policy** tab and ensure that the `network.name` attribute is not empty (its value should be **NetworkName**). Change the value to **false**. Save your changes.

e. After saving, click the **Cancel** button.

f. In the Object Name field type **Change** and press **Enter**.

g. Select the Data Policy tab and ensure that:

   ○ The **header,coordinator** attribute is not empty (its value should be **Coordinator**). Change the value to **false**.

   ○ The **header,orig.operator** attribute is not empty (its value should be **OpenedBy**). Change the value to **false**.

h. Save the changes.

i. Restart ServiceCenter: Select **Start > Programs > ServiceCenter 6.2 > Server > Console** to open the ServiceCenter Console.
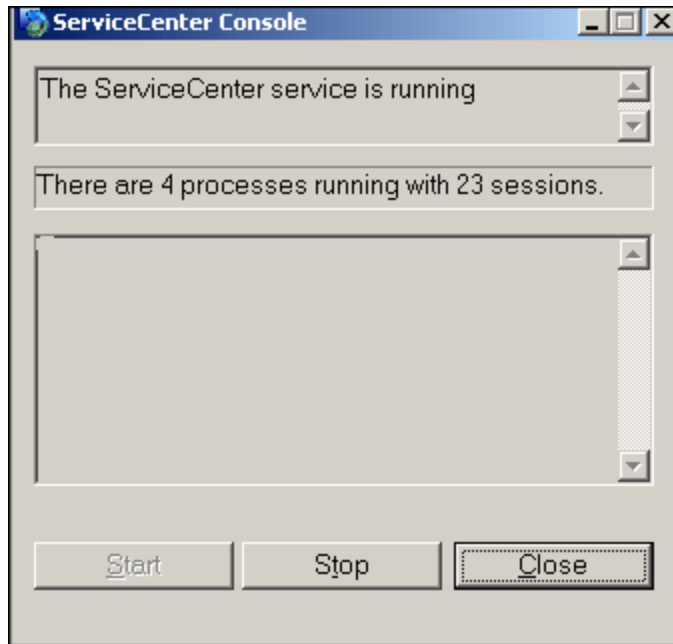
j. Click **Stop** and then **Start**.

k. Continue with "How to Add a New Attribute to an Existing CI Type" on page 176.

3. **Configure HP Service Manager 7.0/7.1**

If you are connecting to HP Service Manager 7.0/7.1, perform the following procedure. If you are connecting to HP ServiceCenter 6.2, skip this step.

a. Import the unload file relevant to the Service Manager version with which you are working: **ucmdbIntegration7_0x.unl** or **ucmdbIntegration7_1x.unl**. To do so, in Service Manager, click **Menu Navigation > Tailoring > Database Manager**.

○ Right-click the detail button and select **Import/Load**.

○ In the HP Service Manager File Load/Import page, click **Specify File** and browse to the following unload file:

C:\hp\UCMDBServer\runtime\fcmdb\CodeBase\ServiceManagerAdapter7-1

The file is loaded via the file browser.

○ Enter the description in the **Import Description** box.

○ Select **winnt** in the **File Type** list.

○ Select a display option.

○ Click **Load FG** to start loading.

b. Continue with "Add an Attribute to the UCMDB Model" on page 176.

4. **Define data push jobs (optional)**

> **Note:** The Data Push flow is relevant for HP Service Manager version 7.1 and later only.

The Service Manager 7.1x-9.2x adapter comes out-of-the-box with the SM History-based Changes push job and the SM Topology Comparison RMI job, which use the queries described below.

- The SM History-based Changes push job uses the following predefined queries: **hostData**, **networkData**, **printerData**, **applicationData**, and **businessServiceData**.

- The SM Topology Comparison RMI job uses of the following predefined queries: **hostRelationsData**, **applicationRelationsData**, and **businessServiceRelationsData**.

For details about these queries, see "Predefined Queries for Data Push Jobs" on page 178.

Each of these jobs runs according to a default schedule setting.

You can also create additional jobs. To do this, select the Data Push tab to define data push jobs that uses the integration point you defined in "Add a ServiceCenter/Service Manager External Data Source" on page 164. For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

5. **Run the jobs**

a. Run the Changes Job, and then run the RMI job.

b. Click the **Refresh Statistics** button [icon] (**Data Flow Management > Integration Studio > Statistics** tab) to review the jobs' statistics. Compare the statistics to the TQL queries by using the **Calculate Query Result Count** button [icon] in the Modeling Studio.

c. In Service Manager, verify that the CIs have been pushed correctly.

6. **Select Classes for Federation**

The adapter contains the following predefined classes for federation: **request_for_change**, **problem**, and **incident**.

How to Add an Attribute to the ServiceCenter/Service Manager CIT

This section explains how to retrieve additional data from ServiceCenter or Service Manager by adding an attribute to the CIT.

This task includes the following steps:

- "Add an attribute to the UCMDB model" below

- "Export attributes from HP ServiceCenter by changing the configuration" below

- "Export attributes from HP Service Manager by changing the configuration" on the next page

- "Modify the Adapter Configuration File" on page 174

1. **Add an attribute to the UCMDB model**

   Edit the Incident CIT to add the new attribute to UCMDB as follows:

   a. Go to **Modeling > CI Type Manager**.

   b. In the CI Types pane, select **IT Process Record > Incident**.

   c. Select the Attributes tab and add the new attribute.

   d. Continue with "Export attributes from HP ServiceCenter by changing the configuration" below or "Export attributes from HP Service Manager by changing the configuration" on the next page.

2. **Export attributes from HP ServiceCenter by changing the configuration**

   If you are connecting to HP ServiceCenter, perform the following procedure.

   a. In HP ServiceCenter, open the ServiceCenter client.

   b. Select **Window > Open Perspective > Administration**:
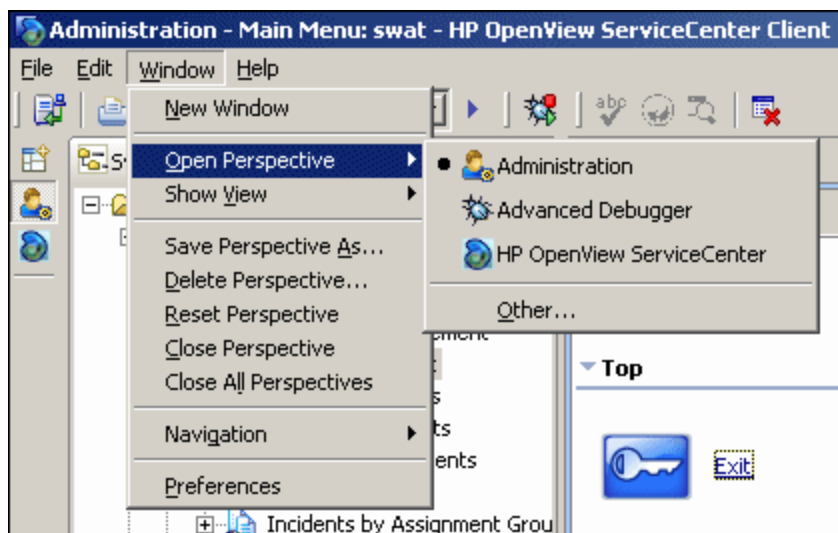
   c. Select **Incident Management > All Open Incidents**, and select one of the incidents you created.

> **Note:** Verify that the value in the Class field is the one that you want to report to
> UCMDB.

d. Search for the value you entered in the Class field (that is, **myclass**), in the XML file
displayed below. This is the CI name in ServiceCenter.

e. Display **WSDL Configuration** in the Navigator (**Main Menu > Menu navigation >
Toolkit**). Locate the Object Name field, enter **Incident** and press **Enter**.

f. Select the **Data Policy** tab. Enter a name for the CI mentioned in the XML file (that is,
class). Change the value to **false**. Save your changes.

g. Restart ServiceCenter: Select **Start > Programs > ServiceCenter 6.2 > Server >
Console** to open the ServiceCenter Console.

h. Click **Stop** and then **Start**.

3. **Export attributes from HP Service Manager by changing the
configuration**

If you are connecting to HP Service Manager, perform the following procedure.

a. In the HP Service Manager client, restore the bottom right pane by clicking the **Restore**
button. Click the **Detail Data** tab.

b. Open one of the incidents you created: Select **Incident Management > Search Incidents**. Click the search button (you can filter the fields to limit the search).

Note: Verify that the value in the Class field is the one that you want to report to HP Universal CMDB.

c. Search for the value you entered in the Class field (that is, **myclass**), in the XML file displayed below. This is the CI name in Service Manager.
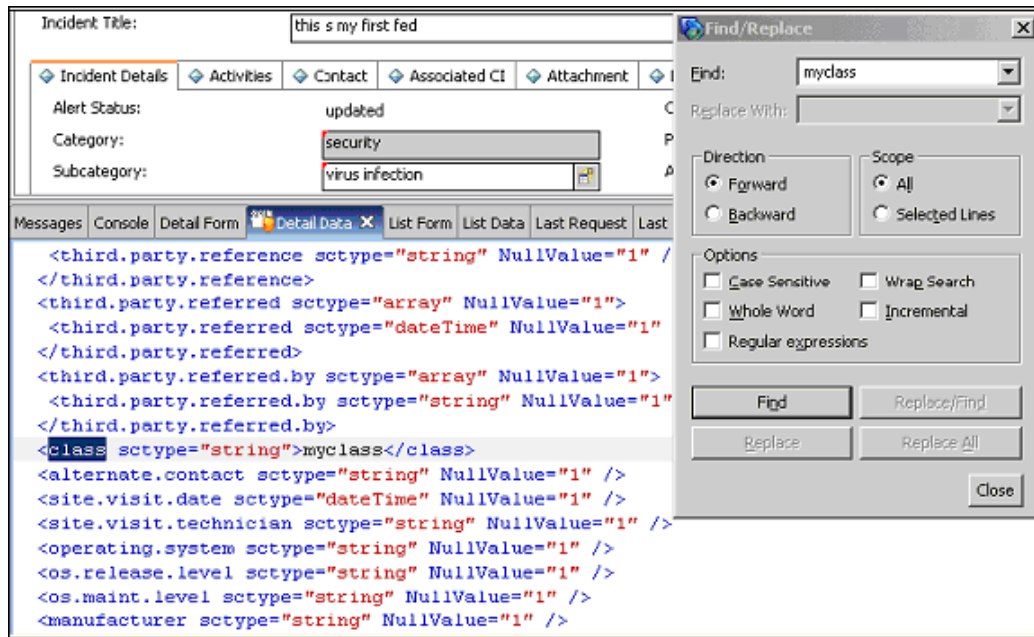
d. Display **WSDL Configuration** in the Navigator (**Main Menu > Menu Navigation > Tailoring**). Locate the Object Name field, enter **UcmdbIncident** and press **Enter**.

e. Select the **Data Policy** tab.

f. Select the **Fields** tab and ensure that the CI name mentioned in the XML file (that is, **class**) appears in the Field list with **ClassName** as its caption. If this attribute does not appear in the Field list, add it and save your changes.

g. Continue with "Modify the Adapter Configuration File" below.

4. **Modify the Adapter Configuration File**

   Perform this procedure for all configurations.

   a. Go to **Data Flow Management > Adapter Management** and select the ServiceManagerAdapterthat corresponds to your version of Service Manager. Continue and select **Configuration Files > ServiceDeskConfiguration.xml**.

   b. Edit the **ServiceDeskConfiguration.xml** file by navigating to **Data Flow Management > Adapter Management > ServiceManagerAdapter** (the one that corresponds to your version of Service Manager) **> Configuration Files > ServiceDeskConfiguration.xml**

   c. Add the new attribute line under the Incident area: Locate the following marker:

   ```
   <ucmdbClassConfiguration ucmdbClassName="it_incident">
   <attributeMappings>
   ```

   d. Add the following line:

```
<attributeMapping ucmdbAttributeName="incident_class" ServiceDeskAttribute
Name="ClassName"/>
```

where:

- ○ `ucmdbAttributeName="incident_class"` is the value defined in the CI Type Manager

- ○ `ServiceDeskAttributeName="ClassName"` is the valued defined in
  ServiceCenter/Service Manager

e. Click **Save**.

How to Communicate with Service Manager over SSL

The following procedure explains how to open communication with Service Manager over SSL.

This task includes the following steps:

- "Add an SM Self-signed Certificate to the UCMDB Trusted Stores" below

- "Add the SM External Data Source Using Communication Over SSL " on the next page

1. **Add an SM Self-signed Certificate to the UCMDB Trusted Stores**

   a. Copy the SM self-signed certificate to a directory. (To export SM self-signed certificates, refer to the Service Manager documentation).

   b. Locate the JRE security folder, by default located in:
      C:\hp\UCMDB\UCMDBServer\bin\jre\lib

   c. Back up the **cacerts** file by renaming it.

   d. Open a command line window and execute the following commands (to import the previously created or copied certificate):

   For HP Universal CMDB 8.0x:

   ```
   cd C:\hp\UCMDB\UCMDBServer \jre\bin"
   keytool.exe -import -keystore     C:\hp\UCMDB\UCMDBServer\j2f\JRE\lib\secu
   rity\cacerts" -trustcacerts -file
   <full path to SM self-signed certificate>
   ```

   For HP Universal CMDB 9.00 or later:

   ```
   cd C:\hp\UCMDB\UCMDBServer\bin\jre\bin
   keytool.exe -import -keystore
   C:\hp\UCMDB\UCMDBServer\bin\jre\lib\security\cacerts -trustcacerts -file
   <full path to SM self-signed certificate>
   ```

e. Restart the UCMDB service.

2. **Add the SM External Data Source Using Communication Over SSL**

   a. In UCMDB, navigate to **Data Flow Management > Integration Studio**.

   b. Define an integration point using the following parameters: In the new integration point dialog box, choose the **ServiceDeskAdapter** for your version of ServiceCenter or Service Manager, and enter the user name, password, and URL. The URL field should contain: **https://<SM server name>:13443/sc62server/ws**.

      For details, see "New Integration Point/Edit Integration Point Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

How to Add a New Attribute to an Existing CI Type

Perform the following steps to add a new attribute to an existing CI type.

This task includes the following steps:

- "Add an Attribute to the UCMDB Model" below

- "Add the Attribute to the Layout of the TQL Query" below

- "Map the Attribute in the SM Adapter Configuration" below

- "Map the Field in the Service Manager Web Service" on the next page

1. **Add an Attribute to the UCMDB Model**

   a. Go to **Modeling > CI Type Manager**.

   b. Select the CI type to which you want to add the attribute.

   c. Select the Attributes tab and add the new attribute.

2. **Add the Attribute to the Layout of the TQL Query**

   a. Go to **Modeling > Modeling Studio**.

   b. Select the query that contains the CI type you want to change (located in the **Integration\SM Sync** folder).

   c. Right-click the node of the CI type you are changing and select **Query Node Properties**.

3. **Map the Attribute in the SM Adapter Configuration**

   a. Go to **Data Flow Management > Adapter Management** and select the ServiceManagerAdapter that corresponds to your version of Service Manager.

   b. Select Configuration Files, and choose the xslt file that contains the CI type you changed.

   c. Add the attribute at the file.device XML tag or at the concrete file XML tag of the type (depends on the Service ManagerWeb Service).

4. **Map the Field in the Service Manager Web Service**

   For details, refer to the Service Manager documentation.

How to Add a New CI Type

Perform the following steps to add a new CI type to the UCMDB class model.

This task includes the following steps:

- "Add the CI Type to the UCMDB Class Model" below
- "Define a TQL Query for Synchronizing the CI Type" below
- "Map the Attribute in the SM Adapter Configuration" below
- "Map the CI Type in the SM Adapter Configuration" on the next page
- "Create and Map the Field in the Service Manager Web Service" on the next page
- "Update the Data Push Job" on the next page

1. **Add the CI Type to the UCMDB Class Model**

   a. Go to **Modeling > CI Type Manager**.

   b. Add the new CI type and its valid relations.

2. **Define a TQL Query for Synchronizing the CI Type**

   a. Go to **Modeling > Modeling Studio**.

   b. In the **Integration\SM Sync** folder, create a new query.

   The new TQL query should include the new CI type (which should be labeled as `Root`) and all the related CIs that are connected to the root node for the additional data. For example: in the **hostData** query, IpAddress and Interface are the additional data of the node.

   The TQL query should also contain the layout that you want to synchronize.

3. **Map the Attribute in the SM Adapter Configuration**

   a. Go to **Data Flow Management > Adapter Management** and select the ServiceManagerAdapter that corresponds to your version of Service Manager.

   b. Select Configuration Files, and choose the xslt file that contains the CI type you changed.

   c. Add the attribute at the file.device XML tag or at the concrete file XML tag of the type

(depends on the Service Manager Web Service).

4. **Map the CI Type in the SM Adapter Configuration**

   a. Go to **Data Flow Management > Adapter Management** and select the ServiceManagerAdapter that corresponds to your version of Service Manager.

   b. Select Configuration Files.

   c. Create a new xslt file for the new CI type and map all the attributes and related CIs to it.

   d. Open **smSyncConfFile.xml** and add a mapping between the new TQL query and the new xslt file.

5. **Create and Map the Field in the Service Manager Web Service**

   For details, refer to the Service Manager documentation.

6. **Update the Data Push Job**

   a. Go to **Data Flow Management > Integration Studio**.

   b. Configure the Data Push job to include the new TQL query.

Predefined Queries for Data Push Jobs

The following TQL queries (located in the Modeling Studio in the **Integration\SM Sync** folder) are provided out-of-the-box if you use the ServiceCenter/Service Manager adapters when you create an integration point.

This section includes:

- "Queries for Data Push Changes Job (SM History-based Changes job)" below

- "Queries for a Data Push RMI job (SM Topology Comparison RMI job)" on the next page

Queries for Data Push Changes Job (SM History-based Changes job)

These queries are used to create a job of type Changes (for pushing CIs):

- **hostData** – use to push nodes. Pushed data includes nodes whose NodeRole attribute is either empty, or contains desktop, server or virtualized_system. Nodes are identified either by their interface or IP address. Information also includes the location of the nodes (building, floor, and room). Due to limitations of the Changes flow, the location information is saved using an enrichment in the Calculated Location attribute.

- **networkData** – use to push nodes that are not pushed with the **hostData** query. This query is similar to **hostData**, except that it pushes nodes whose NodeRole attribute is not empty and does not contain the following strings: `desktop`, `server`, `virtualized_system`, or `printer`.

- **printerData** – use to push printers (network printers). This query is similar to **networkData**,

except that it does push nodes where the NodeRole attribute contains the string `printer`.

- **applicationData** – use to push Business Applications.

- **businessServiceData** – used to push Business Services.

For details, see "Integration Jobs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

> **Note:**
>
> - Select the Allow Delete check box if you want your Data Push job to send deletes of CIs & Links to Service Manager.
>
> - The Changes flow is required for integration with Service Manager because it creates a single CI out of a topology, which matches the Service Manager specification.

Queries for a Data Push RMI job (SM Topology Comparison RMI job)

These queries are used to create a job of type RMI (for pushing Relations):

- **hostRelationsData** – use to push Layer2 (Physical) connections between pairs of nodes through their interfaces.

- **applicationRelationsData** – use to push logical relations between Business Applications to other Business Applications and nodes.

- **businessServiceRelationsData** – use to push logical relations between Business Services to other Business Services, applications and nodes.

For details, see "Integration Jobs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Flow and Configuration

The ServiceCenter/Service Manager adapter receives data and a TQL definition from the Data Push engine, transforms it into a SOAP call for each instance of the TQL query's results, and sends the SOAP requests to Service Manager.

The transformation between the UCMDB class model to the Service Manager class model is done by an XSLT engine.

This section also includes:

- "Parse the TQL Definition" below

- "XSLT Transformation" on page 182

Parse the TQL Definition

The TQL definition must have one Root node (in which case it will be considered a CI synchronization TQL) or several Root links (in which case it will be considered a Relations

synchronization TQL).

Example of an out-of-the-box TQL query for synchronizing a node CI type:
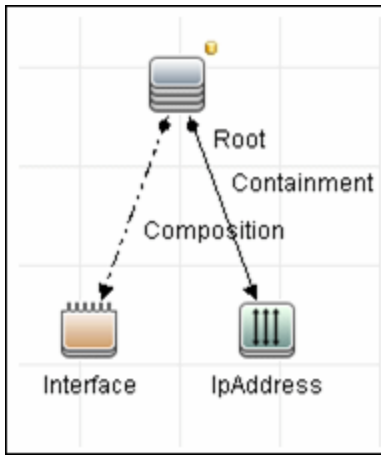
To XML

The result of the TQL query is divided into instances according to the Root node/links, and each instance is given an XML representation.

XML Schema

Each TQL query is automatically assigned a schema according to the structure of the TQL adapter and the layout attributes chosen.

Example of an XML schema for a TQL query:

This example displays the XML schema for a TQL query using a UCMDB JMX located at **http:// [cmdb_machine]:8080/jmx-console/HtmlAdaptor, service=FCmdb Config Services, createXMLSchemaFromTql(**

java.lang.String createXMLSchemaFromTql()

Produce XML Schema for a tql. XML with this schema will be used in some target adapters for transformation purposes.

Param	ParamType	ParamValue	ParamDescription
customerId	int	1	Customer id
tqlName	java.lang.String	applicationData	Name of the TQL Query

[Invoke]

XML schema for a **networkData** TQL query example:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <<xs:element name="node">
    <xs:complexType>
      <xs:sequence>
       <xs:element name="ip_addresss" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                         <xs:sequence>
                             <xs:element name="ip_address" minOccurs="0" maxOccur
s="unbounded">
                                  <xs:complexType>
                                      <xs:attribute name="friendlyType" type="xs:s
tring"/>

                                      <xs:attribute name="id" type="xs:string"/>
                                      <xs:attribute name="ip_netmask" type="xs:str
ing"/>

                                      <xs:attribute name="name" type="xs:string"/>
                                  </xs:complexType>
                             </xs:element>
                         </xs:sequence>
                    </xs:complexType>
               </xs:element>

               <xs:element name="interfaces" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                         <xs:sequence>
                             <xs:element name="interface" minOccurs="0" maxOccurs
="unbounded">
                                  <xs:complexType>
                                      <xs:attribute name="friendlyType" type="xs:s
tring"/>
```

```
                                        <xs:attribute name="id" type="xs:string"/>
                                        <xs:attribute name="mac_address" type="xs:st
ring"/>
                                </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="calculated_location" type="xs:string"/>
        <xs:attribute name="default_gateway_ip_address" type="xs:string"/>
        <xs:attribute name="discovered_os_name" type="xs:string"/>
        <xs:attribute name="discovered_os_version" type="xs:string"/>
        <xs:attribute name="friendlyType" type="xs:string"/>
        <xs:attribute name="global_id" type="xs:string"/>
        <xs:attribute name="id" type="xs:string"/>
        <xs:attribute name="node_role" type="xs:string"/>
        <xs:attribute name="primary_dns_name" type="xs:string"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Example of XML for a **networkData** query:

```
<node customer_id="1" discovered_os_name="windows 2010"
    discovered_os_version="build45-2a" friendlyType="Net Device"
    global_id="bdef388c1b1b3db863ce442a96b54e53" id="bdef388c1b1b3db863ce442a96b
54e53"
    default_gateway_ip_address="1.2.3.4"
    calculated_location="Room:234 Floor:2 Building:M54"
node_role="&lt;Values&gt;&lt;Value&gt;firewall&lt;
/Value&gt;&lt;/Values&gt;" primary_dns_name="myDNS.com">
    <ip_addresss direction="outgoing" linkType="Containment">
        <ip_address customer_id="1" friendlyType="IpAddress"
            id="91757d9d45f166437c1864e931f59e16" ip_address="16.59.64.1"/>
        <ip_address customer_id="1" friendlyType="IpAddress"
            id="f91bf4c40b06e460b51af2178181843d" ip_address="16.59.66.1"/>
    </ip_addresss>
</node>
```

XSLT Transformation

Mapping a TQL name to XSLT

To map between the TQL names and the XSL files, navigate to **Data Flow Management >
Adapter Management > ServiceManagerAdapter** (the one that corresponds to your version of
Service Manager) **> Configuration Files > smSyncConfFile.xml**.

Example of XML for configuring a **hostData** query:

The file includes the names of the Service Manager requests for each operation (create, update,
and delete).

```
<tql name="hostData" xslFile="host_data.xslt">
   <!-- this is host->ip,interface,sm_host tql -->
   <request type="Create" name="CreateucmdbComputerRequest"/>
   <request type="Update" name="UpdateucmdbComputerRequest"/>
   <request type="Delete" name="DeleteucmdbComputerRequest"/>
</tql>
```

The **smSyncConfFile.xml** file must be updated when you add a new TQL query that will be synchronized with Service Manager.

Result after transformation

This sample shows the results after **host_data1.xslt** is executed on the original XML file.

```
<UpdateucmdbNetworkRequest>
        <model>
            <keys/>
            <instance>
                <file.device>
                    <UCMDBId>bdef388c1b1b3db863ce442a96b54e53</UCMDBId>
                    <CustomerId>1</CustomerId>
                    <Subtype>firewall</Subtype>
                    <Building>M54</Building>
                    <Floor>2</Floor>
                    <Room>234</Room>
                    <DefaultGateway>1.2.3.4</DefaultGateway>
                    <OS>windows 2010</OS>
                    <DNSName>myDNS.com</DNSName>
                </file.device>
                <file.networkcomponents>
                    <OSVersion>build45-2a</OSVersion>
                    <addlIPAddr>
                        <addlIPAddr>
                            <AddlIPAddress>16.59.64.1</AddlIPAddress>
                            <AddlSubnet/>
                        </addlIPAddr>
                        <addlIPAddr>
                            <AddlIPAddress>16.59.66.1</AddlIPAddress>
                            <AddlSubnet/>
                        </addlIPAddr>
                    </addlIPAddr>
                </file.networkcomponents>
            </instance>
        </model>
    </UpdateucmdbNetworkRequest>
```

XSLT references

XSLT is a standard language for transforming XML documents into other XML documents. The adapter uses the built-in Java 1.5 Xalan XSLT 1.0 transformer. For details about XSLT see:

http://www.w3.org/TR/1999/REC-xslt-19991116

http://www.w3schools.com/xsl/

http://www.zvon.org/xxl/XSLTutorial/Output/index.html

Reuse of XSLT parts

In addition to the standard XSLT specifications, the adapter? supports the use of an XSLT preprocessor that scans XSL files for comments such as **<!--import:[file_name]-->** in the XSLT, and replaces them with the contents of **[file_name]**.

Service Manager WSDL

Tools such as SoapUI or SoapSonar can be used to view the WSDL files.

Service Manager Web Services are dynamic and can be modified. For details on how to edit or add new Service Manager Web Services, refer to the Service Manager documentation.

Service Manager Result SOAP request

For details on how to enable printing of SOAP requests, see "Logs" below.

Using Mapping Tools

An automatic tool (such as Mapforce) can be used to create XSLT mappings between the CMDB XML schema and the Service Manager XML schema.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for the ServiceCenter/Service Manager adapter.

Changes Flow Limitations

- A query should contain one CI that is labeled as `Root` or one or more relations that are labeled as `Root_<postfix%gt;`.

  The root node is the main CI that is synchronized, and the other nodes are the contained CIs of the main CI. For example, when synchronizing Nodes, the query node of (Node) will be labeled as `Root` and the host resources will not be root.

- The TQL graph must not contain cycles.

- The TQL query must only contain the Root CI, and optionally CIs that are directly connected to it.

- A query that is used to synchronize relations should have cardinality 1...* and OR condition between them.

- Any conditions must reside on the Root CI only.

- If you want to synchronize only specific Roots from a TQL query, you must configure the required condition on these Roots, and then, configure the same condition in the TQL that synchronize the relationships that are linked to the Roots.

- Compound relations are not supported.

- Subgraphs are not supported.

- if one of the TQL queries that are used for synchronization (including layout changes) is edited, the changes will not be synchronized until a full data push job has been manually run. Results from a previous synchronization will not be deleted from the Service Manager server.

- Changes to NodeRole only will not be detected and will not update CI for the next Data Push job.

Logs

Use the **fcmdb.adapters.log** file to troubleshoot the Service Desk adapter (located in the **UCMDBServer\runtime\log** folder).

To view the complete SOAP request and response in addition to other information, use the **fcmdb.properties** file to change the adapter's log level to debug: **log4j.category.fcmdb.adapters=debug,fcmdb.adapters**.

Do not forget to change the log level back to **error** when you are finished debugging. For example, if the **fcmdb.adapters.log** of an Service Manager integration names SM01, for each single CI sent the log will show:

```
DEBUG — SM01 >> Source CI tree is: (The XML as outputted by the ucmdb goes here)
INFO - SM01 >> ======== start run soap message
INFO - SM01 >> ========== create urs required time = 0
DEBUG - SM01 >> Run message: (The XML Send after Xslt Transformation goes here)
DEBUG - SM01 >> Response message: (The XML response goes here)
INFO - SM01 >> ======== stop run soap message. The required time = 390
```

In multi-threaded push flows the thread name indicates the chunk number and thread number:

```
[SM01_pushObjectWorkerThread-<ChunkID>::<ThreadID>]
```

Actual State

To troubleshoot the Actual State flow, use a SOAP testing tool such as SoapUI or SoapSonar to run a SOAP request similar to this:

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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XML
Schema" xmlns:types="http://schemas.hp.com/ucmdb/1/types"%gt;
  <soap:Body>
    <types:getAllCIProperties>
      <types:ID>17868889fd660853e16a474e10df5de3</types:ID>
    </types:getAllCIProperties>
  </soap:Body>
</soap:Envelope>
```

You will obtain a response similar to this:

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header />
<soapenv:Body>
<types:getAllCIPropertiesResponse xmlns:types="http://schemas.hp.com/ucmdb/1/typ
es">
<types:CI id="17868889fd660853e16a474e10df5de3" type="Windows">
<types:prop type="string">
<types:name>Host Name</types:name>
<types:value>LABM2AM209</types:value>
</types:prop>
<types:prop type="string">
<types:name>Host Operating System</types:name>
<types:value>Windows 2003 Server Enterprise Edition </types:value>
```

```
</types:prop>
<types:prop type="string">
<types:name>Host Vendor</types:name>
<types:value>Microsoft Windows</types:value>
</types:prop>
<types:prop type="string">
<types:name>Host DNS Name</types:name>
<types:value>labm2am209.devlab.ad</types:value>
</types:prop>
<types:prop type="string">
<types:name>Asset Tag</types:name>
<types:value>GB8718DS72___</types:value>
</types:prop>
<types:complexProp className="IP" size="1">
<types:item>
<types:prop type="string">
<types:name>IP Address</types:name>
<types:value>16.59.56.161</types:value>
</types:prop>
<types:prop type="string">
<types:name>IP Network Mask</types:name>
<types:value />
</types:prop>
</types:item>
</types:complexProp>
...

</types:CI>
</types:getAllCIPropertiesResponse>
</soapenv:Body>
</soapenv:Envelope>
```

If errors occur, review the following files for exceptions:

- **C:\hp\UCMDB\UCMDBServer\runtime\log\error.log**

- **C:\hp\UCMDB\UCMDBServer\runtime\log\cmdb.operation.log**

Chapter 11: HP Systems Insight Manager (HP SIM) Integration

This chapter includes:

Overview

HP Universal CMDB (UCMDB) can discover data center infrastructure information stored in an HP Systems Insight Manager (HP SIM) system. Integration involves synchronizing devices, topology, and the hierarchy of a data center infrastructure in the UCMDB database (CMDB). This enables change management and impact analysis across all business services mapped in UCMDB, from an infrastructure point of view.

UCMDB initiates discovery on the HP SIM server through Web service calls. Synchronized configuration items (CIs) include nodes such as Windows, and UNIX servers, network devices, printers, clusters, cellular/partitioned systems, blade enclosures, and racks. Some server components, for example, CPU, are also synchronized. The integration also synchronizes relationships between blade servers and blade enclosures, virtual machines, physical servers, and so on. The synchronization uses an XML-based mapping that dynamically changes synchronized CIs and attributes without requiring a code change.

For details on nodes and attributes in HP SIM, refer to the Database tables section of the *HP SIM Technical Reference* guide.

Supported Versions

This integration solution supports HP SIM versions 5.1, 5.2, 5.3, 6.0, 6.1, 6.2, 6.3, 7.0, 7.1, and 7.2.

HP SIM Integration Mechanism

DFM uses the HP SIM Web service API to retrieve node information from the HP SIM database. DFM also enables you to specify extended attributes that should be retrieved for each node.

HP SIM represents hosts (blade enclosures, racks, servers, and so on) as Nodes; UCMDB has separate CITs for each such host. To represent hosts correctly in UCMDB, a two-level mapping is used, to enable integration customization without code changes. This makes the integration completely customizable and dynamic.

For details on jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

The following sections describe the levels of mapping:

HP SIM Node to HP UCMDB Node Mapping

All IP-enabled systems are represented as **Nodes** in HP SIM and each node has attributes (for example, operating device type and operating system name) that can be used to classify nodes as specific CITs in UCMDB. The first level of mapping involves setting parameters on the **SIM Integration** job. This job includes **HostCitIdentifierAttributes** and **HostCitIdentifierMap** parameters that are used for the mapping:

- **HostCitIdentifierAttributes**. This attribute specifies the names of HP SIM Node attributes that are used for the mapping. This parameter uses the **DeviceType** and **OSName** out-of-the-box Node attributes. The parameter accepts comma-separated node attribute names, is case

sensitive, and expects each node attribute name to be enclosed in single quotes.

- **HostCitIdentifierMap**. This attribute specifies the mapping between values of the above HP SIM Node attributes and corresponding UCMDB CITs. This parameter accepts a comma-separated list of value pairs, where each value pair takes the following format:

```
'node attribute value':'UCMDB CI Type'
```

Both attributes are case-sensitive and must be enclosed in single quotes. Each Node-attribute value is one possible value of one or more Node attribute names specified in the **HostCitIdentifierAttributes** parameter. Each UCMDB CIT is the name (not the display name) of the UCMDB CIT to which this value maps.

This parameter has out-of-the-box mappings as follows:

HP SIM Node Attribute	UCMDB CIT
'AIX'	'unix'
'Complex'	'complex'
'Embedded'	'management_processor'
'Enclosure'	'enclosure'
'HPUX'	'unix'
'Hypervisor'	'unix'
'LINUX'	'unix'
'MgmtProc'	'management_processor'
'Printer'	'netprinter'
'Rack'	'rack'
'Server'	'node'
'Solaris'	'unix'
'Switch'	'switch'
'WINNT'	'nt'
'Workstation'	'node'

Example mapping based on the above settings:

- If the **DeviceType** attribute of a node has the value **Switch**, in UCMDB the node is represented as a **Switch** CIT.

- If the **OSName** attribute of a node has the value **WINNT**, in UCMDB the node is represented as an **NT** CIT (Display name: **Windows**).

The DFM script parses these mapping parameters from left to right and does not stop on success, so the rightmost match is considered final. This means that if a node has **DeviceName** = **Server** and **OSName** = **HPUX**, the rightmost match is **OSName** with value **HPUX**. The resulting CIT for this node in UCMDB is **unix** because **HPUX** maps to **unix**.

Node Attribute to CI Type and CI Attribute Mapping

Once the nodes are mapped to CITs using DFM job parameters as described in "HP SIM Node to HP UCMDB Node Mapping" on page 189, individual node attributes (including extended node attributes) are mapped to corresponding attributes (or CITs, as appropriate) using a generic UCMDB integration framework. The framework uses an XML file in which source and target CIT and attribute names are specified.

A sample XML mapping file (**SIM_To_UCMDB_Sample_MappingFile.xml**) that includes all node CITs mapped in the"HP SIM Node to HP UCMDB Node Mapping" on page 189 section is included in the **SIM_Integration** package. The sample file includes host resources (for example, CPU, Disk) and relationship mapping information, to build relationships between various nodes (for example, Blade Enclosure to server, virtual machine host to guest, and so on).

Using this framework, you can map additional CITs without any code changes. For example, to map HBAs, add a new section to the XML file. Define the node attributes that identify an HBA and its attributes. Relationships between HBAs and HOSTs are also required.

How to Discover HP SIM Data Center Infrastructure

This task describes how to discover data center infrastructure information stored in an HP Systems Insight Manager (HP SIM) system.

This task includes the following steps:

- "Prerequisites" below

- "Perform setup on the Probe machine" on the next page

- "Enable chunking - optional" on page 193

- "Run the job" on page 193

1. **Prerequisites**

   Important: If you set up an HTTPS connection to connect to the SIM WebService API (that is, an SSL enabled HTTP connection), the **SIM Integration** job performs **no validation** of any certificates presented by the HP SIM server. The job trusts any certificate issued by the HP SIM server and uses it for SSL enabled communication.

   The following additional requirements must be satisfied for the mapping file to be valid for HP SIM (for details on the mapping files, see "HP SIM Integration Mechanism" on page 189):

- Verify that source and target are **HP SIM** and **HP UCMDB** respectively.

- Verify that attribute names specified in the **HostCitIdentifierAttributes** parameter are included as attributes of each host CIT in the XML file.

  That is, the **OSName** and **DeviceType** attributes must be included for each **host_node** (Computer), **chassis** (Chassis), **netprinter** (Net Printer), **switch** (Switch), **nt** (Windows), **unix** (UNIX), **hp_complex** (Complex), and **management_processor** (Management Processor) CIT.

- Verify that default attributes (that is, non-extended attributes) of a node have a **Node.** prefix in the mapping file.

  That is, you should specify attributes such as **OSName**, **DeviceType**, and **IPAddress** as **Node.OSName**, **Node.DeviceType**, and **Node.IPAddress**.

- Verify that each Node CIT has the following attribute mapping to enable the generation of the **host_key** attribute:

```
<target_attribute name="host_key" datatype="StrProp" >
    <map type="direct" source_attribute="host_key" />
</target_attribute>
```

  > **Note:** The **host_key** attribute is the primary key attribute on Node and derived CITs. Since HP SIM uses a different type of key attribute, the XML definition for the **host_key** attribute is included in the mapping file, to enable generation of the **host_key** primary key attribute.

- Verify that the IP Address mapping section has the following attribute to enable automatic population of the IP domain attribute:

```
<target_attribute name="ip_domain" datatype="StrProp">
    <map type="direct" source_attribute="ip_domain" />
</target_attribute>
```

  > **Note:** For details on the list of HP SIM nodes and attributes, refer to the HP SIM documentation.

2. **Perform setup on the Probe machine**

   a. Copy **mxpartnerlib.jar** from this directory:

      C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\ discoveryResources\hpsim

to this directory:
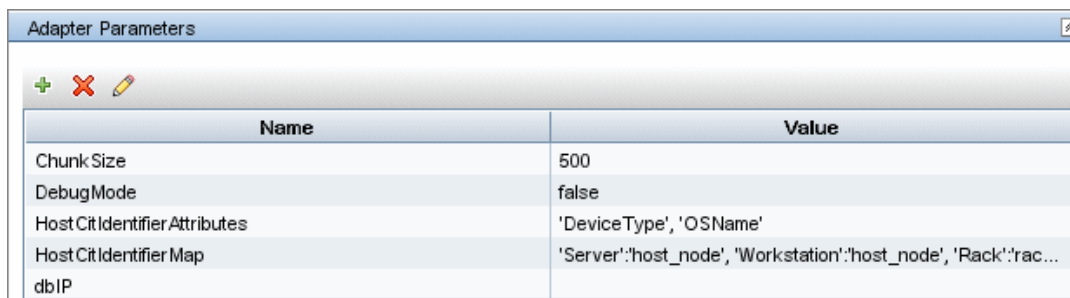
C:\hp\UCMDB\DataFlowProbe\content\lib

b. Open **C:\hp\UCMDB\DataFlowProbe\bin\WrapperEnv.conf** for editing.

c. Comment out line ~51 with a hash sign (**#**) at the beginning so that it looks as follows:

```
#set.SYSTINET_CLASSES=%lib%/webservice;.........
```

d. Save and close the file.

e. Restart the Probe.

3. **Enable chunking - optional**

   If the HP SIM server being discovered contains or manages a large number of nodes (more than 1,000), you should consider enabling chunking (**Data Flow Management > Adapter Management > select an adapter > Adapter Management tab > Adapter Parameters pane**):

   a. To reduce load on the SIM server, if necessary, you can set the **ChunkSize** parameter in the **SIM Integration** adapter to a lower value than the default **500**.

   b. Populate the **dbIP** parameter in the **SIM Integration** adapter with the IP address of the HP SIM CMS database.

   c. Populate the **SIM Database ...** fields in the HP SIM protocol with connection details for the HP SIM CMS database.

   > **Note:** HP SIM CMS database details (except for the password) are located in the **Systems Insight Manager\config\database.props** file on the HP SIM server.

4. **Run the job**

   > **Note:** For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

In the Integration Studio, create a new integration point.

a. Provide a name and description for the integration point.

b. Under **Integration Properties > Adapter**, select the **Systems Insight Manager** adapter.

c. Complete the **dbIP** field with the IP address of the HP SIM CMS database.

d. Under **Adapter Properties > Data Flow Probe**, select the **Data Flow Probe**.

e. Under **Adapter Properties > Trigger CI instance** select:

    i. **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI or

    ii. **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

> **Note:** For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.
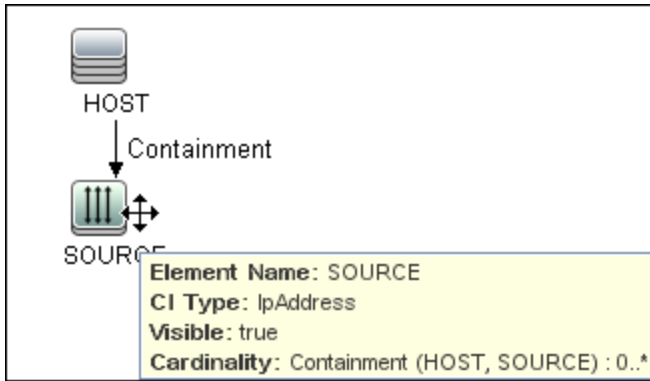
f. Save the integration point.

g. Run the job.

SIM WebService Ports Job

Trigger Query
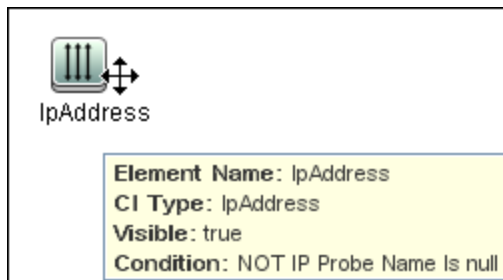
Adapter
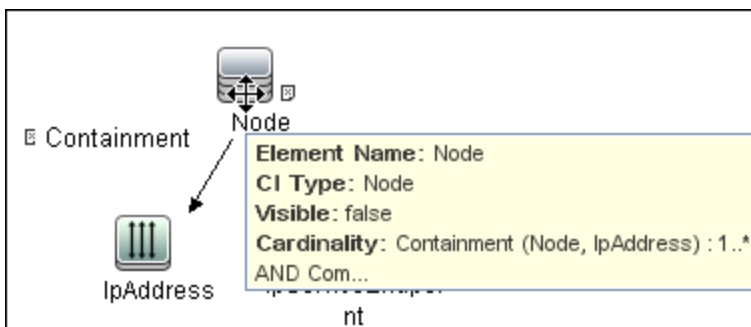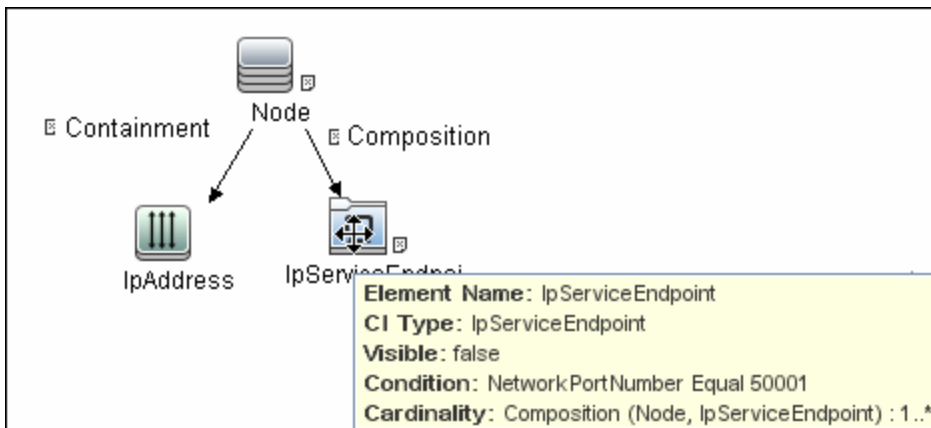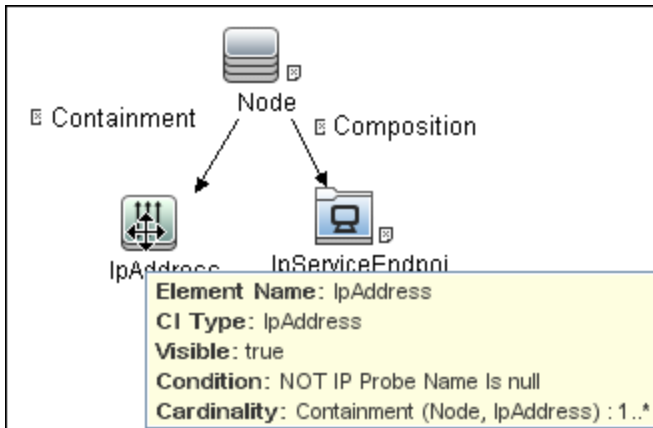
- Input query:

Discovered CITs

- Composition

- Containment

- IpAddress

- IpServiceEndpoint

- Node

- Usage

SIM Integration by WebServices Job

This section includes details about the job.

Trigger Query

Discovered CITs

- Chassis

- Composition

- Computer

- Containment

- Cpu

- Dependency

- Enclosure

- HP Complex

- Interface

- IpAddress

- LogicalVolume

- Management Processor

- Membership
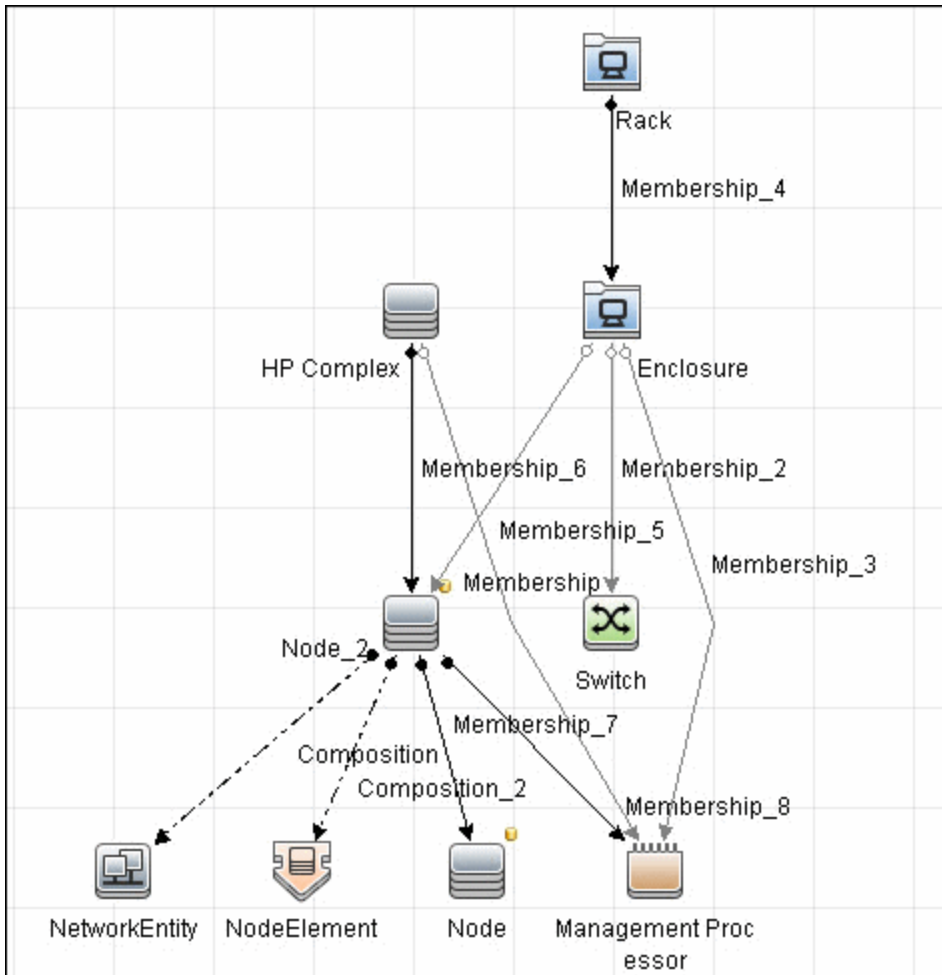
- Net Printer

- Node

- Process

- Rack

- Switch

Instance Views

The package includes two adapter views that show all nodes and resources retrieved from HP SIM, as well as relationships between these nodes.

This section includes the following topics:

-

-

Host Infrastructure View

This view shows relationships between Chassis, Blade Enclosures, Servers, Workstations, Virtual Machine hosts to guests, and so on. This view also shows the interdependence between various nodes in an environment, to enable change management and correlation.

You can use this view, for example, to identify all the servers housed within a specific blade enclosure and all virtual machines running on servers within this blade enclosure. This enables analysis of the impact of shutting down a blade enclosure (say, for a firmware upgrade) on virtual machines. If UCMDB knows of services provided by these virtual machines and which business service these services are part of, it becomes possible to analyze the impact of a blade enclosure outage all the way to a business service.

Hosts and Resources from HP SIM
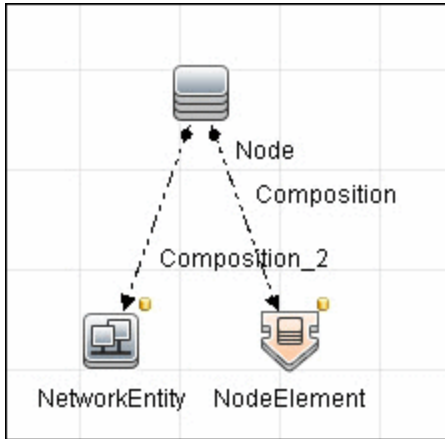
This view shows Node CIs retrieved from HP SIM with associated HostResource and NetworkResource CIs also retrieved from HP SIM.

Troubleshooting and Limitations

This section describes troubleshooting and limatations for HP SIM integration.

> **Note:** The following limitation only applies when (a) using UCMDB Version 9.00-9.03, **and** (b) the user manually increased the out-of-the-box value in the pattern execution option **maximum threads** to greater than 1.

Limitation: If there are multiple HP SIM servers in the environment and this integration is used to integrate with all of them, you should create a new integration job for each HP SIM server and schedule them to run separately. This is because the integration uses XML files to process results from HP SIM, and running the integration against multiple HP SIM servers simultaneously causes the XML files to be overwritten (because the file name is static).